SHARDA GROUP

HINDUSTAN COLLEGE OF SCIENCE & TECHNOLOGY

HINDUSTAN COLLEGE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE FILE ON

DATA STRUCTURES
(KCS-301)

COURSE: **B.TECH.**

DEPARTMENT: **IT (13)**

YEAR/SEMESTER: **2ND / 3RD**

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:29:39+05'30'
Foxit PhantomPDF Version: 10.1.1

SUBJECT TEACHER:
AJAY RAJ PARASHAR
ASST. PROFESSOR (IT)

VISION OF THE COLLEGE

HCST strives to impart a holistic knowledge-centric environment to serve humanity by providing research-oriented technical education to nurture global leaders and entrepreneurs.

MISSION OF THE COLLEGE

1. **Create an ecosystem to foster a culture of innovation, research, academic excellence, and entrepreneurship.**
2. **Nurture technically competent and socially committed global leaders with high moral and ethical values.**
3. **Impart outcome-based education to facilitate students for their holistic development.**

Index		
Sr. No	**Content**	**Remark**
1	Academic Calendar	
2	Syllabus with Course outcomes (COs)	
3	Course outcome (CO)-Program outcome (PO) mapping	
4	Course outcome (CO)-Program specific outcome (PSO) mapping	
5	Class/Section Time table	
6	Individual Time table	
7	Lecture Plan and Lecture plan Execution	
8	University question papers	
9	Question Bank	
10	Attendance Record	
11	Unit Test- Question papers with CO & Results	
12	List of weak students and plan for weak students	
13	Subject notes (handwritten / presentation)	
14	Additional topics covered (other than syllabus)	
15	Assignments	
16	Tutorial Sheets	
17	Sessional Marks with breakups	
18	Result of Subject (last 3 Years)	

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:34:33+05'30'
Foxit PhantomPDF Version: 10.1.1

HINDUSTAN COLLEGE OF SCIENCE & TECHNOLOGY, Farah (MATHURA)
TENTATIVE ACADEMIC CALENDAR FOR ODD SEMESTER: 2022-23
(For 3rd, 5th & 7th Semester)

August 2022

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
	*Kailash Mela	2	3	4	5	6
7	8	9 *Muharram	10	11	12 Raksha Bandhan	13 Registration of VII Sem. (B.Tech.)
14	15 Swatantrata Diwas	16 Commencement of Classes VII Sem. (B.Tech.)	17 Aptitude Workshop start for VII sem.	18	19 Janmashtami	20
21 Aptitude Workshop end for VII sem.	22	23	24	25	26.	27 Registration of III & V Sem. (B.Tech.)
28	29 Commencement of Classes for III & V Sem. (B.Tech.) & National Sports Day	30	31			

September 2022

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19 CR's Meeting-VII Sem.	20 CR's Meeting-V Sem	21 CR's Meeting-III Sem.	22	23	24
25	26 CT-1	27 CT-1	28 CT-1	29	30	

October 2022

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
						1
2 Gandhi Jayanti	3	4 *Mahanavmi	5 Duseehra	6	7 CT-1 marks upload on ERP	8
9 *Id-E-Milad	10	11	12	13	14	15 Innovation Day
16	17 CR's Meeting-VII Sem.	18 CR's Meeting-V Sem	19 CR's Meeting-III Sem	20	21	22
23	24 *Diwali	25	26 Govardhan Puja	27 Bhai Dooj	28	29 CT-2
30	31 CT-2					

November 2022

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
		1 CT-2	2	3	4	5
6	7	8 Guru Nanak Jayanti	9 CT-2 marks upload on ERP	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24 Guru Teg Bahadur Shaheedi Diwas	25	26
27	28	29	30			

December 2022

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
				1 # PUT for V & VII Sem.	2 # PUT	3 # PUT
4	5 # PUT	6 # PUT	7 # PUT	8	9 #AKTU Practical exam Start (V & VII Sem)	10
11	12	13	14	15 # End Semester theory Exam start for V & VII sem.	16 PUT marks upload on ERP	17
18	19	20	21	22	23	24
25 Christmas Day	26	27	28	29 Guru Gobind Singh Jayanti	30	31

January 2023

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24 # End Semester theory Exam start for I & III sem.	25	26 Gantantra Diwas	27	28

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:34:03+05'30'
Foxit PhantomPDF Version: 10.1.1

February 2023

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
29	30	31				
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				

Note: Academic Calendar may be subjected to changes as and when necessary.

- ✓ * Subject to visibility of Moon/Local celebration date
- ✓ # Subjected to AKTU Examination.
- ✓ CT-1, CT-2 & PUT will be conducted after completing 1/3rd, 2/3rd & complete syllabus respectively.
- ✓ As per AKTU, minimum attendance required to appear in CT-1, CT-2 & PUT will be 70%, 75% & 75% respectively from official date of registration.
- ✓ Before CR meeting general online feedback must be conducted.
- ✓ Book Distribution must be start from the next day of registration.
- ✓ Each department must prepare the departmental academic calendar & submit to DA office.
- ✓ Evaluate answer sheets and show them to students within 7 days of completion of the subject paper..
- ✓ Display the solution to the students of CT-1, CT-2 & PUT after complete the examination.

Dr. Rajeev Kumar Upadhyay
Director

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:33:50+05'30'
Foxit PhantomPDF Version: 10.1.1

Mr. Vijay Katta
Assistant Dean Academics

Hindustan College of Science and Technology, Farah-Mathura

DEPARTMENT OF INFORMATION TECHNOLOGY

Date: 20th Nov, 2017

Vision of the Department

Information Technology undergraduate program empowers students with research and training-based education to become global technical leaders and entrepreneurs, creating positive societal impact.

Mission of the Department

1. Foster innovation, research, and entrepreneurship through real-world projects and industry collaborations.
2. Develop ethical global leaders with theoretical and practical education, critical thinking, and social responsibility.
3. Provide outcome-based education for holistic development, integrating technical skills with leadership and teamwork.

PEOs

1. Equip students with IT knowledge, critical thinking abilities, and leadership qualities through research-oriented education to become global entrepreneurs.
2. Prepare students to solve real-world problems and positively impact society through practical experience and industry collaborations.
3. Develop socially responsible global leaders with high moral and ethical values through community engagement and personal/professional growth opportunities.

PSOs

1. Equip students with the latest IT knowledge and skills to tackle real-world challenges.
2. Foster leadership, critical thinking, problem-solving, and communication skills for IT careers.
3. Encourage entrepreneurship and innovation through research, start-up projects, industry collaborations, and business skills.

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA3012478C8CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:33:35+05'30'
Foxit PhantomPDF Version: 10.1.1

HOD-IT
Head
Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

DATA STRUCTURES (KCS-301)

Course Outcomes (COs):

CO No.	Statement of Course Outcome	Bloom's Knowledge Level
After completion of the course, the student will be able to		
CO1	Represent Array and Linked list in an efficient manner and determine the computational efficiency of the algorithms.	K1, K2
CO2	Analyze the concepts of Stack and queue data structure in problem-solving and understanding the concept of recursion, application of recursion	K2
CO3	Explore Tree data structure and its variants and explore the working of advanced trees	K3
CO4	Identify the importance and application of Graph data Structure with problem-solving techniques.	K3
CO5	Apply various searching and sorting algorithms	K4

Unit	Topic	Proposed Lecture
I	Introduction: Basic Terminology, Elementary Data Organization, Built in Data Types in C. Algorithm, Efficiency of an Algorithm, Time and Space Complexity, Asymptotic notations: Big Oh, Big Theta and Big Omega, Time-Space trade-off. Abstract Data Types (ADT) Arrays: Definition, Single and Multidimensional Arrays, Representation of Arrays: Row Major Order, and Column Major Order, Derivation of Index Formulae for 1-D,2-D,3-D and n-D Array Application of arrays, Sparse Matrices and their representations. Linked lists: Array Implementation and Pointer Implementation of Singly Linked Lists, Doubly Linked List, Circularly Linked List, Operations on a Linked List. Insertion, Deletion, Traversal, Polynomiai Representation and Addition Subtraction & Multiplications of Single variable & Two variables Polynomial.	08
II	Stacks: Abstract Data Type, Primitive Stack operations: Push & Pop, Array and Linked Implementation of Stack in C, Application of stack: Prefix and Postfix Expressions, Evaluation of postfix expression, Iteration and Recursion- Principles of recursion, Tail recursion, Removal of recursion Problem solving using iteration and recursion with examples such as binary search, Fibonacci numbers, and Hanoi towers. Tradeoffs between iteration and recursion. Queues: Operations on Queue: Create, Add, Delete, Full and Empty, Circular queues, Array and linked implementation of queues in C, Dequeue and Priority Queue.	08
III	Trees: Basic terminology used with Tree, Binary Trees, Binary Tree Representation: Array Representation and Pointer (Linked List) Representation, Binary Search Tree, Strictly Binary Tree, Complete Binary Tree. A Extended Binary Trees, Tree Traversal algorithms: Inorder, Preorder and Postorder, Constructing Binary Tree from given Tree Traversal, Operation of Insertion, Deletion, Searching & Modification of data in Binary Search. Threaded Binary trees, Traversing Threaded Binary trees. Huffman coding using Binary Tree. Concept & Basic Operations for AVL Tree, B Tree & Binary Heaps	08
IV	Graphs: Terminology used with Graph, Data Structure for Graph Representations: Adjacency Matrices, Adjacency List, Adjacency. Graph Traversal: Depth First Search and Breadth First Search, Connected Component, Spanning Trees, Minimum Cost Spanning Trees: Prims and Kruskal algorithm. Transitive Closure and Shortest Path algorithm: Warshall Algorithm and Dijikstra Algorithm.	08
V	Searching: Concept of Searching, Sequential search, Index Sequential Search, Binary Search. Concept of Hashing & Collision resolution Techniques used in Hashing. Sorting: Insertion Sort, Selection, Bubble Sort, Quick Sort, Merge Sort, Heap Sort and Radix Sort.	08

Text Books:

1. Aaron M. Tenenbaum, Yedidyah Langsam and Moshe J. Augenstein, "Data Structures Using C and C++", PHI Learning Private Limited, Delhi India
2. Horowitz and Sahani, "Fundamentals of Data Structures", Galgotia Publications Pvt Ltd Delhi India.
3. Lipschutz, "Data Structures" Schaum's Outline Series, Tata McGraw-hill Education (India) Pvt. Ltd.
4. Thareja, "Data Structure Using C" Oxford Higher Education.
5. AK Sharma, "Data Structure Using C", Pearson Education India.
6. Rajesh K. Shukla, "Data Structure Using C and C++" Wiley Dreamtech Publication.
7. Michael T. Goodrich, Roberto Tamassia, David M. Mount "Data Structures and Algorithms in C++", Wiley India.
8. P. S. Deshpandey, "C and Data structure", Wiley Dreamtech Publication.
9. R. Kruse etal, "Data Structures and Program Design in C", Pearson Education. Berztiss, AT: Data structures, Theory and Practice, Academic Press.
10. Jean Paul Trembley and Paul G. Sorenson, "An Introduction to Data Structures with applications", McGraw Hill.
11. Adam Drozdek "Data Structures and Algorithm in Java", Cengage Learning

CO-PO Mapping

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	3	3	3	2			1		2	2	3	2	3	
CO2	3	3	3	2	2			2		2	3	3	2	3	
CO3	3	3	3	3	3			2		2	3	3	3	3	
CO4	3	3	3	3	3			2		2	3	3	3	3	
CO5	3	3	3	2	2			1		1	2	3	3	3	
Avg	**3**	**3**	**3**	**2.6**	**2.4**	**0**	**0**	**1.6**	**0**	**1.8**	**2.6**	**3**	**2.6**	**3**	**0**

Program Outcomes (POs)

Engineering Graduates will be able to:

1. Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and computing to solve Computer Science and Engineering related problems.

2. Problem Analysis: Demonstrate the ability to identify, formulate and solve engineering problems related to Computer Science and Engineering.

3. Design / Development of Solutions: Demonstrate the ability to design, analyze and interpret data and implement solutions for software based real life problems.

4. Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis, and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities related to Computer Science and Engineering with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and Team Work: Function effectively as an individual and as a member or leader to diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

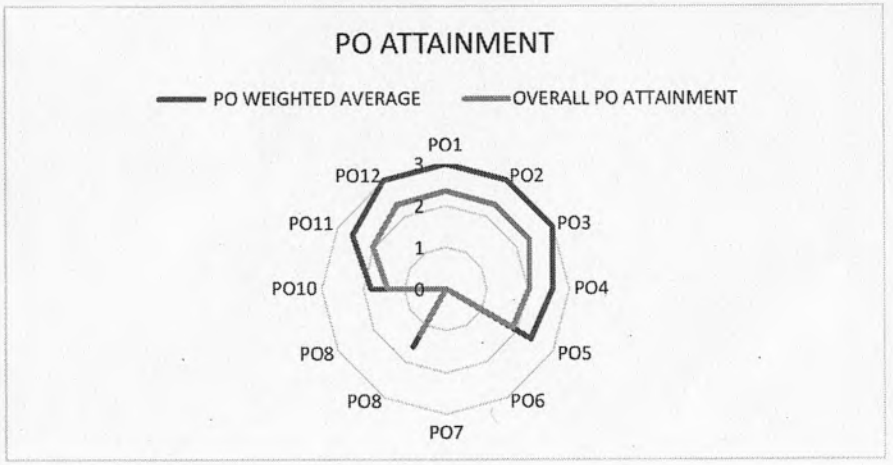
12. Life-Long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

RAJEEV KUMAR UPADHYAY

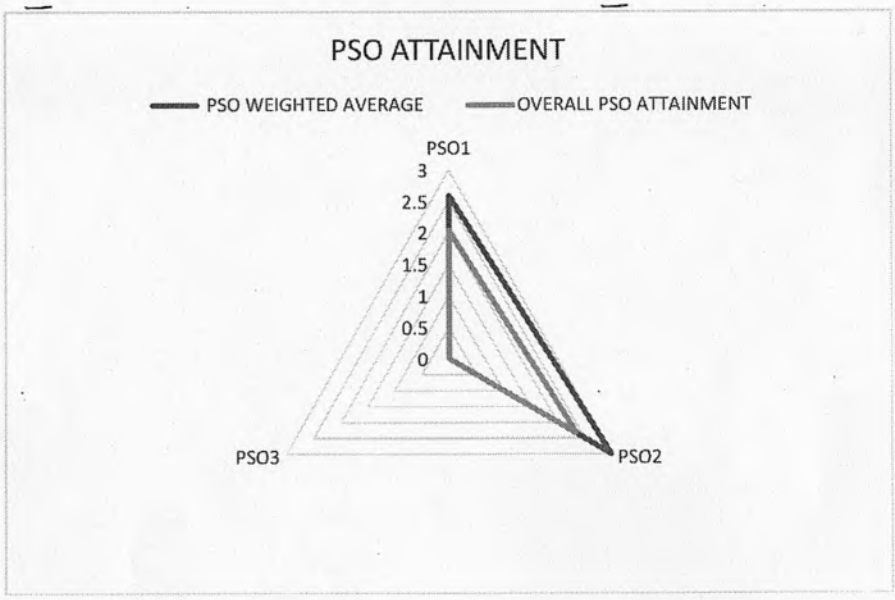
Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:32:56+05'30'
Foxit PhantomPDF Version: 10.1.1

PROGRAM AND PROGRAM SPECIFIC OUTCOME ATTAINMENT CALCULATIONS (2020-21)

Evaluation of PO Attainment			
AVERAGE CO ATTAINMENT			2.34
	PROGRAM OUTCOME	PO WEIGHTED AVERAGE	OVERALL PO ATTAINMENT
	PO1	3	2.34
	PO2	3	2.34
	PO3	3	2.34
	PO4	2.6	2.03
	PO5	2.4	1.88
	PO6	0	0.00
	PO7	0	0.00
	PO8	1.6	1.25
	PO8	0	0.00
	PO10	1.8	1.41
	PO11	2.6	2.03
	PO12	3	2.34



Evaluation of PSO Attainment			
AVERAGE CO ATTAINMENT			2.34
	PROGRAM SPECIFIC OUTCOMES	PSO WEIGHTED AVERAGE	OVERALL PSO ATTAINMENT
	PSO1	2.6	2.03
	PSO2	3	2.34
	PSO3	0	0.00



RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:32:16+05'30'
Foxit PhantomPDF Version: 10.1.1

HINDUSTAN COLLEGE OF SCIENCE & TECHNOLOGY, FARAH, MATHURA

DEPARTMENT OF INFORMATION TECHNOLOGY

CLASS TIME TABLE FOR ODD SEMESTER 2022-23

SESSION: **2022-23** W.E.F. **07/11/22** CLASS TEACHER: MR. VIBHU SHARMA

YEAR/SEM: II/ **3 IT-A** ROOM NO: **221-A** COUNSELORS: **MR. VIBHU SHARMA(A1)**

DR. SANJEEV KUMAR OJHA (A2)

Day \ Time	I	II	III	IV	1:00-01:30	V	VI	VII	VIII
	09:40-10:30	10:30-11:20	11:20-12:10	12:10-01:00		01:30-02:20	02:20-03:10	03:10-04:00	04:00-04:50
MONDAY	DS	COA	DE	DSTL		TC	CSS	COA	DS(T) (A1) DE(T) (A2)
TUESDAY	DSTL	PDP		TC		CO LAB (A1) RNO.321 IIT LAB (A2)		COA	DE
WEDNESDAY	DE	DS	TC	DS(T)(A2) DE(T)(A1)		PDP		COA(T) (A1) DSTL (T) (A2)	LIBRARY
THURSDAY	DSTL	COA	DE	DS		CO LAB (A2) RNO.321 DS LAB (A2)		DSTL LAB (A2) LAB#5 IIT LAB (A1)	
FRIDAY	DS	CSS	DSTL	DSTL(T)(A1) COA(T)(A2)		TC	CSS	DS LAB(A2) Lab#6 DSTL LAB(A1) Lab#5	
SATURDAY									

Class Room No

Subject Code	Name of the Subject	Lecture	Tutorial (A)		Name of the Faculty
KOE-039	DIGITAL ELECTRONICS	221-A			MRS. RUPALI MAHAJAN
KAS-301	TECHNICAL COMMUNICATION (TC)				DR. MAMTA SHARMA
KCS-301	DATA STRUCTURE (DS)				MR. AJAY PARASHAR
KCS-302	(COA)				MR. VIBHU SHARMA
KCS-303	DISCRETE STRUCTURE & THEORY OF LOGIC(DSTL)				MRS. DEEPTI MITTAL
KNC-301	COMPUTER SYSTEM SECURITY(CSS)				DR. SANJEEV KUMAR OJHA

Practical Room No.

Lab Code	Name of the Lab	Lab No (A)	Name of the Faculty
KCS-351	DS LAB	Lab #6	MR. AJAY PARASHAR
KCS-352	CO LAB	R NO:321	MR. VIBHU SHARMA
KCS-353	DSTL LAB	Lab #5	MRS. DEEPTI MITTAL
KCS-354	INTERNSHIP (IIT LAB)	R NO:221-A	DR. SANJEEV KUMAR OJHA

(Mrs. Deepti Mittal)
TT/ IC, IT Dept.

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:31:59+05'30'
Foxit PhantomPDF Version: 10.1.1

(Dr.Shankar Thawkar)
HOD, IT Dept.

Head
Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

HINDUSTAN COLLEGE OF SCIENCE & TECHNOLOGY, FARAH, MATHURA

DEPARTMENT OF INFORMATION TECHNOLOGY

CLASS TIME TABLE FOR ODD SEMESTER 2022-23

SESSION: **2022-23** W.E.F. **10/11/22** CLASS TEACHER: MR. VIBHU SHARMA

YEAR/SEM: II/ **3 IT-A** ROOM NO: **221-A** COUNSELORS: **MR. VIBHU SHARMA(A1)**

DR. SANJEEV KUMAR OJHA (A2)

Day \ Time	I	II	III	IV	1:00-01:30	V	VI	VII	VIII
	09:40-10:30	10:30-11:20	11:20-12:10	12:10-01:00		01:30-02:20	02:20-03:10	03:10-04:00	04:00-04:50
MONDAY	DS	COA	DE	DSTL		TC	CSS	COA	DS(T) (A1) DE(T) (A2)
TUESDAY	DSTL	PDP		TC		CO LAB (A1) RNO.321 IIT LAB (A2)		COA	CSS
WEDNESDAY	DE	DS	TC	DS(T)(A2) DE(T)(A1)		PDP		COA(T) (A1) DSTL (T) (A2)	LIBRARY
THURSDAY	DSTL	COA	DE	DS		CO LAB (A2) RNO.321 DS LAB (A1)		DSTL LAB (A2) LAB#5 IIT LAB (A1)	
FRIDAY	DS	CSS	DSTL	DSTL(T)(A1) COA(T)(A2)		TC	DE	DS LAB(A2) Lab#6 DSTL LAB(A1) Lab#5	
SATURDAY									

		Class Room No		
Subject Code	Name of the Subject	Lecture	Tutorial (A)	Name of the Faculty
KOE-039	DIGITAL ELECTRONICS	221-A		MRS. RUPALI MAHAJAN
KAS-301	TECHNICAL COMMUNICATION (TC)			DR. MAMTA SHARMA
KCS-301	DATA STRUCTURE (DS)			MR. AJAY PARASHAR
KCS-302	(COA)			MR. VIBHU SHARMA
KCS-303	DISCRETE STRUCTURE & THEORY OF LOGIC(DSTL)			MRS. DEEPTI MITTAL
KNC-301	COMPUTER SYSTEM SECURITY(CSS)			DR. SANJEEV KUMAR OJHA

		Practical Room No.	
Lab Code	Name of the Lab	Lab No (A)	Name of the Faculty
KCS-351	DS LAB	Lab #6	MR. AJAY PARASHAR
KCS-352	CO LAB	R NO:321	MR. VIBHU SHARMA
KCS-353	DSTL LAB	Lab #5	MRS. DEEPTI MITTAL
KCS-354	INTERNSHIP (IIT LAB)	R NO:221-A	DR. SANJEEV KUMAR OJHA

(Mrs. Deepti Mittal)
TT/IC, IT Dept.

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:31:45+05'30'
Foxit PhantomPDF Version: 10.1.1

(Dr.Shankar Thawkar)
HOD, IT Dept.

Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

HINDUSTAN COLLEGE OF SCIENCE AND TECHNOLOGY, FARAH, MATHURA

DEPARTMENT OF INFORMATION TECHNOLOGY

SESSION: 2022-23 (ODD)

MR. AJAY RAJ PARASHAR

Day \ Time	I	II	III	IV	1:00-01:30	V	VI	VII	VIII
	09:40-10:30	10:30-11:20	11:20-12:10	12:10-01:00		01:30-02:20	02:20-03:10	03:10-04:00	04:00-04:50
MONDAY	DS		WT						DS(T)(A1)
TUESDAY	WT							IIT LAB 5IT	
WEDNESDAY		DS		DS (T) (A2)				WT	
THURSDAY		WT		DS		DS LAB(A1) Lab#6		WT LAB LAB#6	
FRIDAY	DS		WT					DS LAB(A2) Lab#6	
SATURDAY									

MRS. DEEPTI MITTAL
TIME TABLE I/C

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:31:30+05'30'
Foxit PhantomPDF Version: 10.1.1

DR. SHANKAR THAWKAR
HOD, IT

Head
Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

Hindustan College of Science & Technology

Agra-Delhi Highway (NH-2), Farah,
Distt. Mathura-281122 U.P.
0565-2763366
director.hcst@sgei.org

General Report

Session: 2021 **Program:** B.Tech(IT) **Semester:** Sem III **Sub Code:** KCS301 **Sub Name:** Data Structure (KCS301)

Unit	SNo	Topic	No Of Class
Unit I	1	Introduction To C Programming	1
	2	Identifiers, Keywords, Data Types, Statements	1
	3	If-else, Switch Case, Loops (for, While, Do-while)	1
	4	Functions: Declaration, Definition, Calling	1
	5	Arrays And Structures	1
	6	Introduction To Data Structure: Basic Terminology	1
	7	Algorithm, Efficiency, Time And Space Complexity	1
	8	Asymptotic Notations: Big-oh, Time-space Trade-off	1
	9	Abstract Data Types (adt), Arrays	1
	10	Representation Of Arrays: Row & Column Major Order	1
	11	Sparse Matrices And Their Representations	1
	12	Linked Lists: Array & Dynamic Implementation	1
	13	Doubly Linked List, Circularly Linked List	1
	14	Operations On A Linked List: Insertion, Deletion And Traversal	1
	15	Polynomial Representation And Addition	1
	16	Generalized Linked List	1
Unit II	17	Stacks: Abstract Data Type, Push & Pop	1
	18	Array And Linked Implementation Of Stack In C	1
	19	Application Of Stack: Prefix And Postfix Expressions	1
	20	Application Of Stack: Prefix And Postfix Expressions	1
	21	Evaluation Of Postfix Expression	1
	22	Recursion, Tower Of Hanoi Problem	1
	23	Principles Of Recursion, Tail Recursion, Removal Of Recursion	1
	24	Queues, Operations On Queue: Create, Add, Delete, Full And Empty	1
	25	Array And Linked Implementation Of Queues In C	1
	26	Circular Queues And Their Array And Linked Implementation In C	1
	27	Dequeue And Priority Queue	1
Unit III	28	Searching: Sequential Search, Binary Search	1
	29	Insertion Sort, Selection Sort	1
	30	Bubble Sort, Quick Sort	1
	31	Two Way Merge Sort	1
	32	Heap Sort, Radix Sort	1
	33	Hashing, Storage Management	1
Unit IV	34	Graphs: Introduction & Terminology	1
	35	Sequential And Linked Representations Of Graphs	1
	36	Graph Traversal: Depth First Search And Breadth First Search	1
	37	Connected Component, Spanning Trees	1
	38	Minimum Cost Spanning Trees: Prims And Kruskal Algorithm	1
	39	Shortest Path Algorithm: Warshal Algorithm	1
	40	Shortest Path Algorithm: Dijikstra Algorithm	1
	41	Introduction To Activity Networks	1
Unit V	42	Trees: Basic Terminology, Binary Trees	1
	43	Binary Tree Representation: Array And Dynamic Representation	1
	44	Complete Binary Tree, Algebraic Expressions	1
	45	Extended Binary Trees	1
	46	Array And Linked Representation Of Binary Trees	1
	47	Tree Traversal Algorithms: Inorder, Preorder And Postorder	1
	48	Threaded Binary Trees, Traversing Threaded Binary Trees	1
	49	Huffman Algorithm	1

Report Generated By : 08ALIT560 On 24 Dec 2022 15:53:12 PM
Powered By : Global Infoways

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:31:14+05'30'
Foxit PhantomPDF Version: 10.1.1

General Report

Unit	SNo	Topic	No Of Class
Unit V	50	Bst, Insertion And Deletion, Complexity	1
	51	Avl Trees, B Trees & B+ Trees	1

College = 'HCST' AND Session = '2021' AND Program = 'B.Tech(IT)' AND Semester = 'Sem III' AND Subject Code = 'KCS301'

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:31:01+05'30'
Foxit PhantomPDF Version: 10.1.1

Hindustan College of Science & Technology

Agra-Delhi Highway (NH-2), Farah,

Distt. Mathura-281122 U.P.

0565-2763366

director.hcst@sgei.org

Faculty Wise Topic

Session: 2021 **Program:** B.Tech(IT) **Semester:** Sem III **Sub Code:** KCS301 **Sub Name:** Data Structure (KCS301)

Unit	SNo	Topic	Emp Code	Emp Name	Att Date	Period
Unit I	1	Introduction To C Programming	08ALIT560	AJAY RAJ PARASHAR	01/09/2022	P1(10:10 AM-11:00 AM)
	2	Identifiers, Keywords, Data Types, Statements	08ALIT560	AJAY RAJ PARASHAR	02/09/2022	P1(10:10 AM-11:00 AM)
	3	If-else, Switch Case, Loops (for, While, Do-while)	08ALIT560	AJAY RAJ PARASHAR	03/09/2022	P1(10:10 AM-11:00 AM)
	4	Functions: Declaration, Definition, Calling	08ALIT560	AJAY RAJ PARASHAR	05/09/2022	P1(10:10 AM-11:00 AM)
	5	Arrays And Structures	08ALIT560	AJAY RAJ PARASHAR	06/09/2022	P4(12:41 PM-13:30 PM)
	6	Introduction To Data Structure: Basic Terminology	08ALIT560	AJAY RAJ PARASHAR	07/09/2022	P2(11:01 AM-11:50 AM)
	7	Algorithm, Efficiency, Time And Space Complexity	08ALIT560	AJAY RAJ PARASHAR	08/09/2022	P4(12:41 PM-13:30 PM)
	8	Asymptotic Notations: Big-oh, Time-space Trade-off	08ALIT560	AJAY RAJ PARASHAR	09/09/2022	P1(10:10 AM-11:00 AM)
	9	Abstract Data Types (adt), Arrays	08ALIT560	AJAY RAJ PARASHAR	12/09/2022	P1(10:10 AM-11:00 AM)
	10	Representation Of Arrays: Row & Column Major Order	08ALIT560	AJAY RAJ PARASHAR	14/09/2022	P2(11:01 AM-11:50 AM)
	11	Representation Of Arrays: Row & Column Major Order	08ALIT560	AJAY RAJ PARASHAR	15/09/2022	P4(12:41 PM-13:30 PM)
	12	Sparse Matrices And Their Representations	08ALIT560	AJAY RAJ PARASHAR	16/09/2022	P1(10:10 AM-11:00 AM)
	13	Linked Lists: Array & Dynamic Implementation	08ALIT560	AJAY RAJ PARASHAR	19/09/2022	P1(10:10 AM-11:00 AM)
	14	Doubly Linked List, Circularly Linked List	08ALIT560	AJAY RAJ PARASHAR	21/09/2022	P2(11:01 AM-11:50 AM)
	15	Operations On A Linked List: Insertion, Deletion And Traversal	08ALIT560	AJAY RAJ PARASHAR	29/09/2022	P4(12:11 PM-13:00 PM)
	16	Polynomial Representation And Addition	08ALIT560	AJAY RAJ PARASHAR	30/09/2022	P1(09:40 AM-10:30 AM)
	17	Generalized Linked List	08ALIT560	AJAY RAJ PARASHAR	06/10/2022	P4(12:11 PM-13:00 PM)
Unit II	18	Stacks: Abstract Data Type, Push & Pop	08ALIT560	AJAY RAJ PARASHAR	11/10/2022	P1(09:40 AM-10:30 AM)
	19	Array And Linked Implementation Of Stack In C	08ALIT560	AJAY RAJ PARASHAR	12/10/2022	P2(10:31 AM-11:20 AM)
	20	Application Of Stack: Prefix And Postfix Expressions	08ALIT560	AJAY RAJ PARASHAR	13/10/2022	P3(11:21 AM-12:10 PM)
	21	Application Of Stack: Prefix And Postfix Expressions				
	22	Evaluation Of Postfix Expression	08ALIT560	AJAY RAJ PARASHAR	18/10/2022	P1(09:40 AM-10:30 AM)
	23	Recursion, Tower Of Hanoi Problem	08ALIT560	AJAY RAJ PARASHAR	19/10/2022	P1(09:40 AM-10:30 AM)
	24	Principles Of Recursion, Tail Recursion, Removal Of Recursion	08ALIT560	AJAY RAJ PARASHAR	20/10/2022	P1(09:40 AM-10:30 AM)
	25	Queues, Operations On Queue: Create, Add, Delete, Full And Empty	08ALIT560	AJAY RAJ PARASHAR	03/11/2022	P4(12:11 PM-13:00 PM)
	26	Array And Linked Implementation Of Queues In C	08ALIT560	AJAY RAJ PARASHAR	04/11/2022	P1(09:40 AM-10:30 AM)
	27	Circular Queues And Their Array And Linked Implementation In C	08ALIT560	AJAY RAJ PARASHAR	07/11/2022	P4(12:11 PM-13:00 PM)
	28	Dequeue And Priority Queue	08ALIT560	AJAY RAJ PARASHAR	09/11/2022	P2(10:31 AM-11:20 AM)
t III	29	Searching: Sequential Search, Binary Search	08ALIT560	AJAY RAJ PARASHAR	09/11/2022	P4(12:11 PM-13:00 PM)
	30	Insertion Sort, Selection Sort	08ALIT560	AJAY RAJ PARASHAR	10/11/2022	P1(09:40 AM-10:30 AM)
	31	Bubble Sort, Quick Sort	08ALIT560	AJAY RAJ PARASHAR	11/11/2022	P1(09:40 AM-10:30 AM)
	32	Two Way Merge Sort	08ALIT560	AJAY RAJ PARASHAR	14/11/2022	P1(09:40 AM-10:30 AM)
	33	Heap Sort, Radix Sort	08ALIT560	AJAY RAJ PARASHAR	16/11/2022	P2(10:31 AM-11:20 AM)
	34	Hashing, Storage Management	08ALIT560	AJAY RAJ PARASHAR	17/11/2022	P4(12:11 PM-13:00 PM)
Unit IV	35	Graphs: Introduction & Terminology	08ALIT560	AJAY RAJ PARASHAR	23/11/2022	P1(09:40 AM-10:30 AM)
	36	Sequential And Linked Representations Of Graphs	08ALIT560	AJAY RAJ PARASHAR	25/11/2022	P1(09:40 AM-10:30 AM)
	37	Graph Traversal: Depth First Search And Breadth First Search	08ALIT560	AJAY RAJ PARASHAR	30/11/2022	P1(09:40 AM-10:30 AM)
	38	Connected Component, Spanning Trees	08ALIT560	AJAY RAJ PARASHAR	01/12/2022	P4(12:11 PM-13:00 PM)
	39	Minimum Cost Spanning Trees: Prims And Kruskal Algorithm	08ALIT560	AJAY RAJ PARASHAR	02/12/2022	P1(09:40 AM-10:30 AM)
	40	Shortest Path Algorithm: Warshal Algorithm	08ALIT560	AJAY RAJ PARASHAR	07/12/2022	P2(10:31 AM-11:20 AM)
	41	Shortest Path Algorithm: Dijikstra Algorithm	08ALIT560	AJAY RAJ PARASHAR	08/12/2022	P4(12:11 PM-13:00 PM)
	42	Introduction To Activity Networks	08ALIT560	AJAY RAJ PARASHAR	12/12/2022	P5(13:31 PM-14:20 PM)
Unit V	43	Trees: Basic Terminology, Binary Trees	08ALIT560	AJAY RAJ PARASHAR	14/12/2022	P2(10:31 AM-11:20 AM)
	44	Binary Tree Representation: Array And Dynamic Representation	08ALIT560	AJAY RAJ PARASHAR	15/12/2022	P1(09:40 AM-10:30 AM)

Report Generated By : 08ALIT560 On 24 Dec 2022 15:55:05 PM

Powered By : Global Infoways

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:30:48+05'30'
Foxit PhantomPDF Version: 10.1.1

Faculty Wise Topic

Unit	SNo	Topic	Emp Code	Emp Name	Att Date	Period
Unit V	45	Complete Binary Tree, Algebraic Expressions	08ALIT560	AJAY RAJ PARASHAR	21/12/2022	P1(09:40 AM-10:30 AM)
	46	Extended Binary Trees	08ALIT560	AJAY RAJ PARASHAR	21/12/2022	P2(10:31 AM-11:20 AM)
	47	Array And Linked Representation Of Binary Trees	08ALIT560	AJAY RAJ PARASHAR	22/12/2022	P1(09:40 AM-10:30 AM)
	48	Tree Traversal Algorithms: Inorder, Preorder And Postorder	08ALIT560	AJAY RAJ PARASHAR	23/12/2022	P2(10:31 AM-11:20 AM)
	49	Threaded Binary Trees, Traversing Threaded Binary Trees				
	50	Huffman Algorithm				
	51	Bst, Insertion And Deletion, Complexity				
	52	Avl Trees, B Trees & B+ Trees				

College = 'HCST' AND Session = '2021' AND Program = 'B.Tech(IT)' AND Semester = 'Sem III' AND Subject Code = 'KCS301'

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:30:26+05'30'
Foxit PhantomPDF Version: 10.1.1

UNIVERSITY QUESTION PAPER

Printed Pages: 2
Paper Id: 233380

Sub Code: KCS- 301
Roll No.

B. TECH.
(SEM III) THEORY EXAMINATION 2022-23
DATA STRUCTURE

Time: 3 Hours *Total Marks: 100*

Note: 1. Attempt all Sections. If require any missing data; then choose suitably.

SECTION A

1. Attempt *all* questions in brief. **2 x 10 = 20**

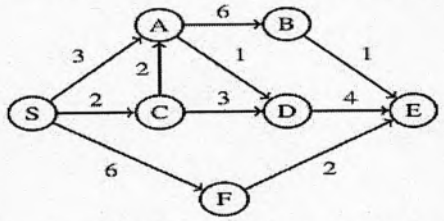
(a) Define best case, average case and worst case for analyzing the complexity of a program.
(b) Differentiate between binary search tree and a heap.
(c) Write the condition for empty and full circular queue.
(d) What do you understand by tail recursion?
(e) Construct an expression tree for the following algebraic expression:
(a - b) / ((c * d) + e)
(f) Differentiate between internal sorting and external sorting.
(g) What are the advantages and disadvantages of array over linked list?
(h) Write an algorithm for Breadth First Search (BFS) traversal of a graph.
(i) In a complete binary tree if the number of nodes is 1000000. What will be the height of complete binary tree.
(j) Which data structure is used to perform recursion and why?

SECTION B

2. Attempt any *three* of the following: **10x3=30**

(a) Assume that the declaration of multi-dimensional arrays X and Y to be,
X (-2:2, 2:22) and Y (1:8, -5:5, -10:5)
(i) Find the length of each dimension and number of elements in X and Y.
(ii) Find the address of element Y (2, 2, 3), assuming Base address of Y = 400 and each element occupies 4 memory locations.

(b) What is Stack? Write a C program for linked list implementation of stack.
(c) Write an algorithm for Quick sort. Use Quick sort algorithm to sort the following elements: **2, 8, 7, 1, 3, 5, 6, 4**
(d) Write the Dijkstra algorithm for shortest path in a graph and also find the shortest path from 'S' to all remaining vertices of graph in the following graph:



(e) The order of nodes of a binary tree in inorder and postorder traversal are as follows:

In order : **B, I, D, A, C, G, E, H, F.**

Post order: **I, D, B, G, C, H, F, E, A.**

(i) Draw the corresponding binary tree.
(ii) Write the pre order traversal of the same tree.

SECTION C

3. Attempt any *one* part of the following: 10x1=10

(a) How to represent the polynomial using linked list ? Write a C program to add two polynomials using linked list.

(b) Discuss doubly linked list. Write an algorithm to insert a node after a given node in singly linked list.

4. Attempt any *one* part of the following: 10x1=10

(a) Write an algorithm for converting infix expression into postfix expression. Trace your algorithm for infix expression Q into its equivalent postfix expression P,

Q: A + (B * C – (D / E ^ F) * G) * H

(b) What is circular Queue? Write a C code to insert an element in circular queue?

5. Attempt any *one* part of the following: 10x1=10

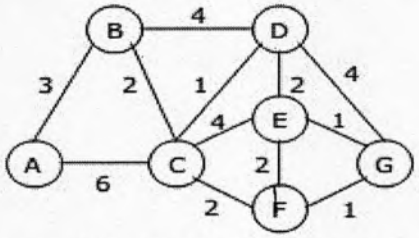
(a) What is Hashing? Explain division method to compute the hash function and also explain the collision resolution strategies used in hashing.

(b) Write an algorithm for Heap Sort. Use Heap sort algorithm, sort the following sequence:

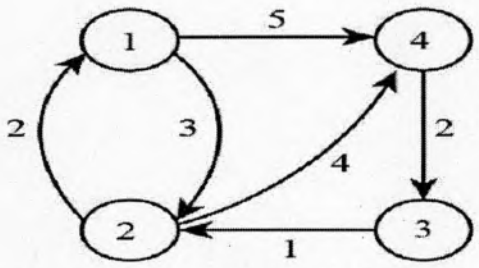
18, 25, 45, 34, 36, 51, 43, and 24.

6. Attempt any *one* part of the following: 10x1=10

(a) What is spanning tree? Write down the Prim's algorithm to obtain minimum cost spanning tree. Use Prim's algorithm to find the minimum cost spanning tree in the following graph:



(b) Write and explain the Floyd Warshall algorithm to find the all pair shortest path. Use the Floyd Warshall algorithm to find shortest path among all the vertices in the given graph:



7. Attempt any *one* part of the following: 10x1=10

(a) Discuss left skewed and right skewed binary tree. Construct an AVL tree by inserting the following elements in the order of their occurrence:

60, 2, 14, 22, 13, 111, 92, 86.

(b) What is B-Tree? Write the various properties of B- Tree. Show the results of inserting the keys **F, S, Q, K ,C, L, H, T, V, W, M, R, N, P, A, B** in order into a empty B-Tree of order **5.**

Roll No: | | | | | | | | | | | | | |

BTECH
(SEM III) THEORY EXAMINATION 2021-22
DATA STRUCTURE

Time: 3 Hours *Total Marks: 100*

Note: Attempt all Sections. If you require any missing data, then choose suitably.

SECTION A

1. Attempt *all* questions in brief. 2X10 = 20

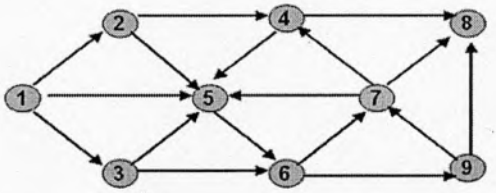
Q No	Questions	CO
(a)	Convert the infix expression (A+B) *(C-D) $E*F to postfix. Give the answer without any spaces.	1
(b)	Rank the following typical bounds in increasing order of growth rate: $O(\log n)$, $O(n^4)$, $O(1)$, $O(n^2 \log n)$	2
(c)	Draw the binary search tree that results from inserting the following numbers in sequence starting with 11: 11, 47, 81, 9, 61, 10, 12,	3

(d)	What does the following recursive function do for a given Linked List with first node as head?
	void fun1(struct node* head)
	{
	if(head == NULL)
	return;
	fun1(head->next);
	printf("%d ", head->data);
	}
		4

(e)	Define a sparse matrix. Suggest a space efficient representation for space matrices.	5
(f)	List the advantages of doubly linked list over single linked list.	1
(g)	Give example of one each stable and unstable sorting techniques.	2
(h)	Write advantages of AVL tree over Binary Search Tree (BST)	3
(i)	What is tail recursion? Explain with a suitable example.	4
(j)	Write different representations of graphs in the memory.	5

SECTION B

2. Attempt any *three* of the following: 10X3 = 30

Q No	Questions	CO
(a)	Write advantages and disadvantages of linked list over arrays. Write a 'C' function creating new linear linked list by selecting alternate elements of a linear linked list.	1
(b)	Write algorithms of insertion sort. Implement the same on the following numbers; also calculate its time complexity. 13, 16, 10, 11, 4, 12, 6, 7	2
(c)	Differentiate between DFS and BFS. Draw the breadth First Tree for the above graph. 	3
(d)	Differentiate between liner and binary search algorithm. Write a recursive function to implement binary search.	4
(e)	What is the significance of maintaining threads in Binary Search Tree? Write an algorithm to insert a node in thread binary tree.	5

SECTION C

3. Attempt any *one* part of the following: 10X1 = 10

Q No	Questions	CO
(a)	Suppose a three dimensional array A is declared using A[1:10, -5:5, -10:5) (i) Find the length of each dimension and the number of elements in A (ii) Explain Row major order and Column Major Order in detail with explanation formula expression.	1

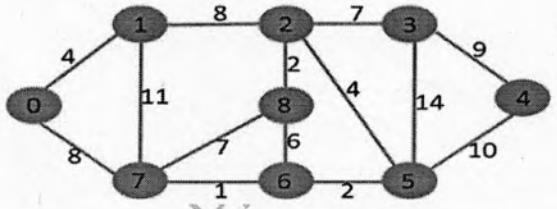
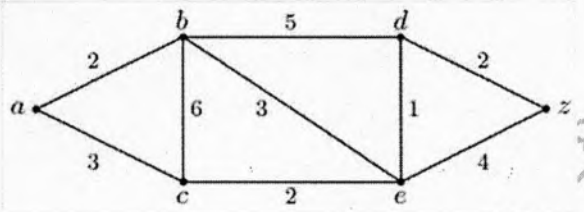
BTECH
(SEM III) THEORY EXAMINATION 2021-22
DATA STRUCTURE

(b)	Discuss the representation of polynomial of single variable using linked list. Write 'C' functions to add two such polynomials represented by linked list.	1

4. **Attempt any *one* part of the following:** **10 X1 = 10**

Q No	Questions	CO
(a)	(i) Use the merge sort algorithm to sort the following elements in ascending order. 13, 16, 10, 11, 4, 12, 6, 7. What is the time and space complexity of merge sort? (ii) Use quick sort algorithm to sort 15,22,30,10,15,64,1,3,9,2. Is it a stable sorting algorithm? Justify.	2
(b)	(i) The keys 12, 17, 13, 2, 5, 43, 5 and 15 are inserted into an initially empty hash table of length 15 using open addressing with hash function h(k) = k mod 10 and linear probing. What is the resultant hash table? (ii) Differentiae between linear and quadratic probing techniques.	2

5. **Attempt any *one* part of the following:** **10X1 = 10**

Q No	Questions	CO
(a)	Use Dijkstra's algorithm to find the shortest paths from source to all other vertices in the following graph. 	3
(b)	Apply Prim's algorithm to find a minimum spanning tree in the following weighted graph as shown below. 	3

6. **Attempt any *one* part of the following:** **10X1 = 10**

Q No	Questions	CO
(a)	(i) Write an iterative function to search a key in Binary Search Tree (BST). (ii) Discuss disadvantages of recursion with some suitable example.	4
(b)	(i) What is Recursion? (ii)Write a C program to calculate factorial of number using recursive and non-recursive functions.	4

7. **Attempt any *one* part of the following:** **10X1 = 10**

Q No	Questions	CO
(a)	(i) Why does time complexity of search operation in B-Tree is better than Binary Search Tree (BST)? (ii) Insert the following keys into an initially empty B-tree of order 5 a, g, f, b, k, d, h, m, j, e, s, i, r, x, c, l, n, t, u, p (iii) What will be the resultant B-Tree after deleting keys j, t and d in sequence?	5
(b)	(i) Design a method for keeping two stacks within a single linear array so that neither stack overflow until all the memory is used. (ii) Write a C program to reverse a string using stack.	5

PAPER ID-311107

Roll No: | | | | | | | | | | | | | |

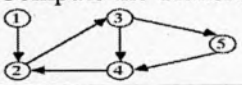
B. TECH
(SEM III) THEORY EXAMINATION 2020-21
DATA STRUCTURES

Time: 3 Hours *Total Marks: 100*

Note: **1.** Attempt all Sections. If require any missing data; then choose suitably.

SECTION A

1. Attempt *all* questions in brief. **2 x 10 = 20**

Q no.	Question	Marks	CO
a.	Define Time-Space trade-off.	2	1
b.	Differentiate Array and Linked list.	2	1
c.	Explain Tail Recursion with suitable example.	2	2
d.	Write the full and empty condition for a circular queue data structure.	2	2
e.	Examine the minimum number of interchanges needed to convert the array 90, 20, 41,18, 13, 11, 3, 6, 8,12, 7, 71, 99 into a maximum heap.	2	3
f.	Differentiate sequential search and binary search.	2	3
g.	Compute the Transitive closure of following graph. 	2	4
h.	Write short notes on adjacency multi list representation a Graph.	2	4
i.	What is the importance of threaded binary tree?	2	5
j.	Write short notes on min heap.	2	5

SECTION B

2. Attempt any *three* of the following:

Q no.	Question	Marks	CO
a.	Consider a multi-dimensional Array *A*[90] [30] [40] with base address starts at 1000. Calculate the address of A[10] [20] [30] in row major order and column major order. Assume the first element is stored at *A*[2][2][2] and each element take 2 byte.	10	1
b.	Evaluate the following postfix expression using stack. **2 3 9 * + 2 3 ^ - 6 2 / +** , show the contents of each and every steps. also find the equivalent prefix form of above expression. Where ^ is an exponent operator.	10	2
c.	Explain any three commonly used hash function with the suitable example? A hash function H defined as H(key) =key%7, with linear probing, is used to insert the key 37,38,72,48,98,11,66 into a table indexed from 0 to 6. what will be the location of key 11? Justify your answer, also count the total number of collisions in this probing.	10	3
d.	Write an algorithm for Breadth First search (BFS) and explain with the help of suitable example.	10	4
e.	If the in order of a binary tree is B,I,D,A,C,G,E,H,F and its post order is I,D,B,G,C H,F,E,A then draw a corresponding binary tree with neat and clear steps from above assumption.	10	5

Roll No: | | | | | | | | | | | | | |

SECTION C

3. Attempt any *one* part of the following:

Q no.	Question	Marks	CO
a.	Consider the two dimensional lower triangular matrix (LTM) of order N ,Obtain the formula for address calculation in the address of row major and column major order for location LTM[j][k], if base address is BA and space occupied by each element is w byte.	10	1
b.	Write a C program to insert a node at k^{th} position in single linked list.	10	1

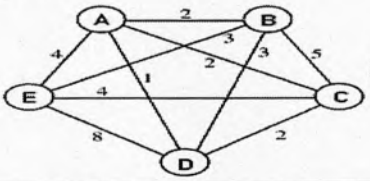
4. Attempt any *one* part of the following:

Q no.	Question	Marks	CO
a.	Convert the following infix expression to reverse polish notation expression using stack. $$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$	10	2
b.	Write a C program to implement stack using single linked list.	10	2

5. Attempt any *one* part of the following:

Q no.	Question	Marks	CO
a.	Write an algorithm for merge sort and apply on following elements 45,32,65,76,23,12,54,67,22,87.	10	3
b.	Write a C program for Index Sequential Search.	10	3

6. Attempt any *one* part of the following:

Q no.	Question	Marks	CO
a.	Describe Prim`s algorithm and find the cost of minimum spanning tree using Prim`s Algorithm. 	10	4
b.	Apply the Floyd warshall's algorithm in above mentioned graph (i.e. in Q.no 6a)	10	4

7. Attempt any *one* part of the following:

Q no.	Question	Marks	CO
a.	Write Short notes of following (a) Extended Binary Trees (b) Complete Binary Tree (c) Threaded Binary Tree.	10	5
b.	Insert the following sequence of elements into an AVL tree, starting with empty tree 71,41,91,56,60,30,40,80,50,55 also find the minimum array size to represent this tree.	10	5

Sub Code:KCS301

Paper Id: 110321 Roll No:

B. TECH.
(SEM III) THEORY EXAMINATION 2019-20
DATA STRUCTURES

Time: 3 Hours *Total Marks: 100*

Note: 1. Attempt all Sections. If require any missing data; then choose suitably.

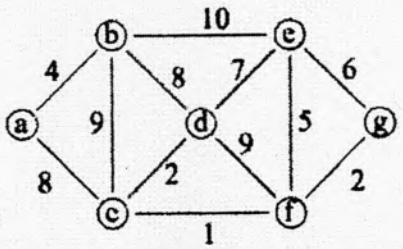
SECTION A

1. Attempt *all* questions in brief. **2 x 10 = 20**

Qno.	Question	Marks	CO
a.	How can you represent a sparse matrix in memory?	2	CO1
b.	List the various operations on linked list.	2	CO1
c.	Give some applications of stack.	2	CO2
d.	Explain Tail recursion.	2	CO2
e.	Define priority queue. Given one application of priority queue.	2	CO3
f.	How does bubble sort work? Explain.	2	CO3
g.	What is Minimum cost spanning tree? Give its applications.	2	CO4
h.	Compare adjacency matrix and adjacency list representations of graph.	2	CO4
i.	Define extended binary tree, full binary tree, strictly binary tree and complete binary tree.	2	CO5
j.	Explain threaded binary tree.	2	CO5

SECTION B

2. Attempt any *three* of the following: **3 x 10 = 30**

Qno.	Question	Marks	CO
a.	What are the merits and demerits of array? Given two arrays of integers in ascending order, develop an algorithm to merge these arrays to form a third array sorted in ascending order.	10	CO1
b.	Write algorithm for Push and Pop operations in stack. Transform the following expression into its equivalent postfix expression using stack: A + (B * C – (D / E ↑ F) * G) * H	10	CO2
c.	How binary search is different from linear search? Apply binary search to find item 40 in the sorted array: 11, 22, 30, 33, 40, 44, 55, 60, 66, 77, 80, 88, 99. Also discuss the complexity of binary search.	10	CO3
d.	Find the minimum spanning tree in the following graph using Kruskal's algorithm: 	10	CO4
e.	What is the difference between a binary search tree (BST) and heap? For a given sequence of numbers, construct a heap and a BST. 34, 23, 67, 45, 12, 54, 87, 43, 98, 75, 84, 93, 31	10	CO5

Sub Code:KCS301

Paper Id: 110321 Roll No:

SECTION C

3. Attempt any *one* part of the following: 1 x 10 = 10

Qno.	Question	Marks	CO
a.	What is doubly linked list? What are its applications? Explain how an element can be deleted from doubly linked list using C program.	10	CO1
b.	Define the following terms in brief: (i) Time complexity (iii) Space complexity (ii) Asymptotic Notation (iv) Big O Notation	10	CO1

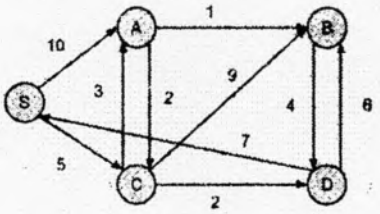
4. Attempt any *one* part of the following: 1 x 10 = 10

Qno.	Question	Marks	CO
a.	(i) Differentiate between iteration and recursion. (ii) Write the recursive solution for Tower of Hanoi problem.	10	CO2
b.	Discuss array and linked representation of queue data structure. What is dequeue?	10	CO2

5. Attempt any *one* part of the following: 1 x 10 = 10

Qno.	Question	Marks	CO
a.	Why is quick sort named as quick? Show the steps of quick sort on the following set of elements:25, 57, 48, 37, 12, 92, 86, 33 Assume the first element of the list to be the pivot element.	10	CO3
b.	What is hashing? Give the characteristics of hash function. Explain collision resolution technique in hashing.	10	CO3

6. Attempt any *one* part of the following: 1 x 10 = 10

Qno.	Question	Marks	CO
a.	Explain warshall's algorithm with the help of an example.	10	CO4
b.	Describe the Dijkstra algorithm to find the shortest path. Find the shortest path in the following graph with vertex 'S" as source vertex. 	10	CO4

7. Attempt any *one* part of the following: 1 x 10 = 10

Qno.	Question	Marks	CO
a.	Can you find a unique tree when any two traversals are given? Using the following traversals construct the corresponding binary tree: INORDER: H K D B I L E A F C M J G PREORDER: A B D H K E I L C F G J M Also find the Post Order traversal of obtained tree.	10	CO5
b.	What is a B-Tree? Generate a B-Tree of order 4 with the alphabets (letters) arrive in the sequence as follows: a g f b k d h m j e s i r x c l n t u p	10	CO5

Printed Pages: 02
Paper Id: 1 1 0 3 0 3

Sub Code: RCS305
Roll No.

B. TECH.
(SEM III) THEORY EXAMINATION 2018-19
DATA STRUCTURES

Time: 3 Hours *Total Marks: 70*

Note: 1. Attempt all Sections. If require any missing data; then choose suitably.

SECTION A

1. Attempt *all* questions in brief. **2 x 7 = 14**

a. How the graph can be represented in memory? Explain with suitable example.

b. Write the syntax to check whether a given circular queue is full or empty?

c. Draw a binary Tree for the expression: A * B - (C + D) * (P /Q)

d. Differentiate between overflow and underflow condition in a linked list.

e. What do you understand by stable and in place sorting?

f. Number of nodes in a complete tree is 100000. Find its depth.

g. What is Recursion? Give disadvantages of recursion.

SECTION B

2. Attempt any *three* of the following: **7 x 3 = 21**

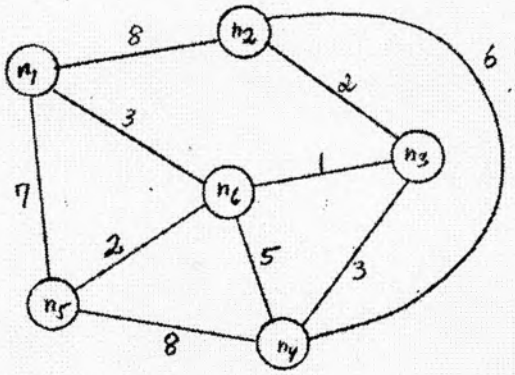
a. What do you understand by time and space trade off? Define the various asymptotic notations. Derive the O-notation for linear search.

b. Consider the following infix expression and convert into reverse polish notation using stack: A + (B * C – (D / E ^ F) * H)

c. Explain Huffman algorithm. Construct Huffman tree for MAHARASHTRA with its optimal code.

d. What is a height balanced Tree? Why height balancing of Tree is required? Create an AVL Tree for the following elements: a, z, b, y, c, x, d, w, e, v, f

e. Write the Floyd Warshall algorithm to compute the all pair shortest path. Apply the algorithm on following graph:



SECTION C

3. Attempt any *one* part of the following: **7 x 1 = 7**

(a) Write a program in c to delete a specific element in single linked list. Double linked list takes more space than single linked list for storing one extra address. Under what condition, could a double linked list more beneficial than single linked list.

(b) Suppose multidimensional arrays P and Q are declared as P (-2: 2, 2: 22) and Q (1: 8, -5: 5, -10: 5) stored in column major order
(i) Find the length of each dimension of P and Q
(ii) The number of elements in P and Q
(iii) Assuming Base address (Q) = 400, W=4, Find the effective indices E1, E2, E3 and address of the element Q [3, 3, 3].

4. Attempt any *one* part of the following: **7 x 1 = 7**

(a) Explain Tower of Hanoi problem and write a recursive algorithm to solve it.

(b) Explain how a circular queue can be implemented using arrays. Write all functions for circular queue operations.

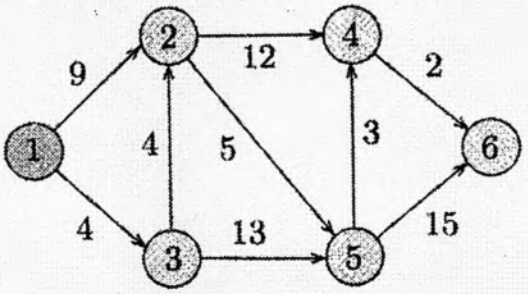
5. Attempt any *one* part of the following: **7 x 1 = 7**

(a) Write the algorithm for deletion of an element in binary search tree.

(b) Construct the binary tree for the following.
In-order: Q, B, K, C, F, A, G, P, E, D, H, R
Preorder: G, B, Q, A, C, K, F, P, D, E, R, H
Find the Post Order of the Tree.

6. Attempt any *one* part of the following: **7 x 1 = 7**

(a) By considering vertex '1' as source vertex, Find the shortest paths to all other vertices in the following graph using Dijkstra's algorithms. Show all the steps.



(b) Explain in detail about the graph traversal techniques with suitable examples.

7. Attempt any *one* part of the following: **7 x 1 = 7**

(a) Write algorithm for Quick sort. Trace your algorithm on the following data to sort the list: 2, 13, 4, 21, 7, 56, 51, 85, 59, 1, 9, 10. How the choice of pivot element effects the efficiency of algorithm.

(b) Construct a B-tree of order 5 created by inserting the following elements
3, 14, 7, 1, 8, 5, 11, 17, 13, 6, 23, 12, 20, 26, 4, 16, 18, 24, 25, 19
Also delete elements 6, 23 and 3 from the constructed tree.

Printed Pages: 02 **Sub Code: RCS305**

Paper ID: | 1 | 0 | 0 | 4 | **Roll No.** | | | | | | | | | | |

B TECH

(SEM III) THEORY EXAMINATION 2017-18

DATA STRUCTURES

Time: 3Hours ***Max. Marks: 70***

Note: Attempt all Sections. Assume missing data, if any.

SECTION A

1. Attempt *all* questions in brief: **2 x 7 = 14**

a. Define the term Data Structure. List some linear and non-linear data structures stating the application area where they will be used.

b. Discuss the concept of "successor" and "predecessor" in Binary Search Tree.

c. Convert the following arithmetic infix expression into its equivalent postfix expression.

Expression: A-B/C+D*E+F

d. Explain circular queue. What is the condition if circular queue is full?

e. Calculate total number of moves for Tower of Hanoi for n=10 disks.

f. List the different types of representation of graphs.

g. Explain height balanced tree. List general cases to maintain the height.

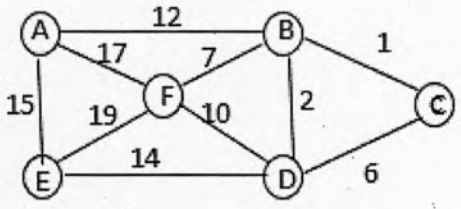
SECTION B

2. Attempt any *three* of the following: **7 x 3 = 21**

a. What do you understand by time space trade off? Explain best, worst and average case analysis in this respect with an example

b. Use quick sort algorithm to sort 15,22,30,10,15,64,1,3,9,2. Is it a stable sorting algorithm? – Justify.

c. Define spanning tree. Also construct minimum spanning tree using prim's algorithm for the given graph.



d. Define tree, binary tree, complete binary tree and full binary tree. Write algorithms or function to obtain traversals of a binary tree in preorder, postorder and inorder.

e. Construct a B-tree on following sequence of inputs.

10, 20, 30, 40, 50, 60, 70, 80, 90

Assume that the order of the B-tree is 3.

SECTION C

3. Attempt any *one* part of the following: **7 x 1 = 7**

(a) What are the various asymptotic notations? Explain Big O notation.

(b) Write an algorithm to insert a node at the end in a Circular linked list.

4. Attempt any *one* part of the following: 7 x 1 = 7

(a)What is a Stack .Write a C program to reverse a string using stack.

(b)Define the recursion. Write a recursive and non recursive program to calculate the factorial of the given number.

5. Attempt any *one* part of the following: 7 x 1 = 7

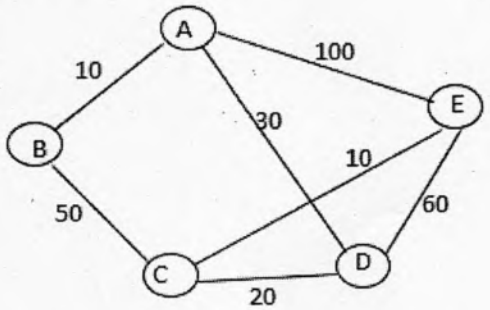
(a) Draw a binary tree with following traversals:
Inorder: B C A E G D H F I J
Preorder: A B C D E G F H I J

(b) Consider the following AVL Tree and insert 2, 12, 7and 10 as new node. Show proper rotation to maintain the tree as AVL.

6. Attempt any *one* part of the following: 7 x 1 = 7

(a)What is a Threaded Binary Tree? Explain the advantages of using a threaded binary tree.

(b)Describe Dijkstra's algorithm for finding shortest path. Describe its working for the graph given below.



7. Attempt any *one* part of the following: 7 x 1 = 7

(a)Write short notes on:

i. Hashing Technique
ii. Garbage collection

(b)Explain the following:

i. Heap Sort
ii. Radix Sort.

QUESTION BANK

DATA STRUCTURES QUESTION BANK

UNIT I

2 MARKS

1.Define data structure.

The data structure can be defined as the collection of elements and all the possible operations which are required for those set of elements. Formally data structure can be defined as a data structure is a set of domains D, a set of domains F and a set of axioms A. this triple (D,F,A) denotes the data structure d.

2. What do you mean by non-linear data structure? Give example.

The non-linear data structure is the kind of data structure in which the data may be arranged in hierarchical fashion. For example- Trees and graphs.

3. What do you linear data structure? Give example.

The linear data structure is the kind of data structure in which the data is linearly arranged. For example- stacks, queues, linked list.

4. List the various operations that can be performed on data structure.

Various operations that can be performed on the data structure are

- Create
- Insertion of element
- Deletion of element
- Searching for the desired element
- Sorting the elements in the data structure
- Reversing the list of elements.

5. What is abstract data type? What are all not concerned in an ADT?

The abstract data type is a triple of D i.e. set of axioms, F-set of functions and A-Axioms in which only what is to be done is mentioned but how is to be done is not mentioned. Thus ADT is not concerned with implementation details.

6. List out the areas in which data structures are applied extensively.

Following are the areas in which data structures are applied extensively.

- Operating system- the data structures like priority queues are used for scheduling the jobs in the operating system.
- Compiler design- the tree data structure is used in parsing the source program. Stack data structure is used in handling recursive calls.

- Database management system- The file data structure is used in database management systems. Sorting and searching techniques can be applied on these data in the file.
- Numerical analysis package- the array is used to perform the numerical analysis on the given set of data.
- Graphics- the array and the linked list are useful in graphics applications.

- Artificial intelligence- the graph and trees are used for the applications like building expression trees, game playing.

7. What is a linked list?

A linked list is a set of nodes where each node has two fields 'data' and 'link'. The data field is used to store actual piece of information and link field is used to store address of next node.

8. What are the pitfall encountered in singly linked list?

Following are the pitfall encountered in singly linked list

- The singly linked list has only forward pointer and no backward link is provided. Hence the traversing of the list is possible only in one direction. Backward traversing is not possible.
- Insertion and deletion operations are less efficient because for inserting the element at desired position the list needs to be traversed. Similarly, traversing of the list is required for locating the element which needs to be deleted.

9. Define doubly linked list.

Doubly linked list is a kind of linked list in which each node has two link fields. One link field stores the address of previous node and the other link field stores the address of the next node.

10. Write down the steps to modify a node in linked lists.

- Enter the position of the node which is to be modified.
- Enter the new value for the node to be modified.
- Search the corresponding node in the linked list.
- Replace the original value of that node by a new value.
- **Display the messages as "The node is modified".**

11. Difference between arrays and lists.

In arrays any element can be accessed randomly with the help of index of array, whereas in lists any element can be accessed by sequential access only.

Insertion and deletion of data is difficult in arrays on the other hand insertion and deletion of data is easy in lists.

12. State the properties of LIST abstract data type with suitable example.

Various properties of LIST abstract data type are

(i) It is linear data structure in which the elements are arranged adjacent to each other.

(ii) It allows to store single variable polynomial.

(iii)If the LIST is implemented using dynamic memory then it is called linked list.

Example of LIST are- stacks, queues, linked list.

13. State the advantages of circular lists over doubly linked list.

In circular list the next pointer of last node points to head node, whereas in doubly linked list each node has two pointers: one previous pointer and another is next pointer. The main advantage of circular list over doubly linked list is that with the help of single pointer field we can access head node quickly. Hence some amount of memory get saved because in circular list only one pointer is reserved.

14. What are the advantages of doubly linked list over singly linked list?

The doubly linked list has two pointer fields. One field is previous link field and another is next link field. Because of these two pointer fields we can access any node efficiently whereas in singly linked list only one pointer field is there which stores forward pointer.

15. Why is the linked list used for polynomial arithmetic?

We can have separate coefficient and exponent fields for representing each term of polynomial. Hence there is no limit for exponent. We can have any number as an exponent.

16. What is the advantage of linked list over arrays?

The linked list makes use of the dynamic memory allocation. Hence the user can allocate or de allocate the memory as per his requirements. On the other hand, the array makes use of the static memory location. Hence there are chances of wastage of the memory or shortage of memory for allocation.

17. What is the circular linked list?

The circular linked list is a kind of linked list in which the last node is connected to the first node or head node of the linked list.

18. What is the basic purpose of header of the linked list?

The header node is the very first node of the linked list. Sometimes a dummy value such - 999 is stored in the data field of header node.

This node is useful for getting the starting address of the linked list.

19. What is the advantage of an ADT?

- **Change:** the implementation of the ADT can be changed without making changes in theclient program that uses the ADT.
- **Understandability:** ADT specifies what is to be done and does not specify theimplementation details. Hence code becomes easy to understand due to ADT.
- **Reusability:** the ADT can be reused by some program in future.

20. What is static linked list? State any two applications of it.

- The linked list structure which can be represented using arrays is called static linked list.
- It is easy to implement, hence for creation of small databases, it is useful
- The searching of any record is efficient, hence the applications in which the record need to be searched quickly, the static linked list are used.

10 MARKS

1. Explain the insertion operation in linked list. How nodes are inserted after a specified node.
2. Write an algorithm to insert a node at the beginning of list?
3. Discuss the merge operation in circular linked lists.
4. What are the applications of linked list in dynamic storage management?
5. How polynomial expression can be represented using linked list?
6. What are the benefit and limitations of linked list?
7. Define the deletion operation from a linked list.
8. What are the different types of data structure?
9. Explain the operation of traversing linked list. Write the algorithm and give an example.

UNIT II

2 MARKS

1. Define Stack

A Stack is an ordered list in which all insertions (Push operation) and deletion (Pop operation) are made at one end, called the top. The topmost element is pointed by top. The top is initialized to -1 when the stack is created that is when the stack is empty. In a stack S = (a1,an), a1 is the bottom most element and element a is on top of element ai-1. Stack is also referred as Last In First Out (LIFO) list.

2. What are the various Operations performed on the Stack?

The various operations that are performed on the stack are

CREATE(S) – Creates S as an empty stack.

PUSH(S,X) – Adds the element X to the top of the stack.

POP(S) – Deletes the top most elements from the stack.

TOP(S) – returns the value of top element from the stack.

ISEMTPTY(S) – returns true if Stack is empty else false.

ISFULL(S) - returns true if Stack is full else false.

3.Write the postfix form for the expression -A+B-C+D?

A-B+C-D+

4.What are the postfix and prefix forms of the expression?

A+B*(C-D)/(P-R)

Postfix form: ABCD-*PR-/+

Prefix form: +A/*B-CD-PR

5.Explain the usage of stack in recursive algorithm implementation?

In recursive algorithms, stack data structures is used to store the return address when a recursive call is encountered and also to store the values of all the parameters essential to the current state of the function.

6.**Define Queue.**

A Queue is an ordered list in which all insertions take place at one end called the rear, while all deletions take place at the other end called the front. Rear is initialized to -1 and front is initialized to 0. Queue is also referred as First In First Out (FIFO) list.

7.**What are the various operations performed on the Queue?**

The various operations performed on the queue are

CREATE(Q) – Creates Q as an empty Queue.

Enqueue(Q,X) – Adds the element X to the Queue.

Dequeue(Q) – Deletes a element from the Queue.

ISEMTPTY(Q) – returns true if Queue is empty else false.

ISFULL(Q) - returns true if Queue is full else false.

8.How do you test for an empty Queue?

The condition for testing an empty queue is rear=front-1. In linked list implementation of queue the condition for an empty queue is the header node link field is NULL.

9.**Write down the function to insert an element into a queue, in which the queue is implemented as an array. (May 10)**

Q – Queue

X – element to added to the queue Q IsFull(Q)

– Checks and true if Queue Q is full Q->Size -

Number of elements in the queue Q Q->Rear –

Points to last element of the queue Q Q->Array

– array used to store queue elements void

```
enqueue (int X, Queue Q) {
        if(IsFull(Q))
          Error ("Full queue");
        else    {
                Q->Size++;
```

```
        Q->Rear = Q->Rear+1;
        Q->Array[ Q->Rear ]=X;
    }
}
```

10..Define Dequeue.

Deque stands for Double ended queue. It is a linear list in which insertions and deletion are made from either end of the queue structure.

11.**Define Circular Queue.**

Another representation of a queue, which prevents an excessive use of memory by arranging elements/ nodes $Q_1,Q_2,\ldots Q_n$ in a circular fashion. That is, it is the queue, which wraps around upon reaching the end of the queue

12. List any four applications of stack.

- Parsing context free languages
- Evaluating arithmetic expressions
- Function call
- Traversing trees and graph
- Tower of Hanoi

10 MARKS

1. Write an algorithm for Push and Pop operations on Stack using Linked list. (8)
2. Explain the linked list implementation of stack ADT in detail?
3. Define an efficient representation of two stacks in a given area of memory with n words and explain.
4. Explain linear linked implementation of Stack and Queue?
    a. Write an ADT to implement stack of size N using an array. The elements in the stack are to be integers. The operations to be supported are PUSH, POP and DISPLAY. Take into account the exceptions of stack overflow and stack underflow. (8)
    b. A circular queue has a size of 5 and has 3 elements 10,20 and 40 where F=2 and R=4. After inserting 50 and 60, what is the value of F and R. Trying to insert 30 at this stage what happens? Delete 2 elements from the queue and insert 70, 80 & 90. Show the sequence of steps with necessary diagrams with the value of F & R. (8 Marks)
5. Write the algorithm for converting infix expression to postfix (polish) expression?

6. Explain in detail about priority queue ADT in detail?
7. **Write a function called 'push' that takes two parameters: an integer variable and a stack into** which it would push this element and returns a 1 or a 0 to show success of addition or failure.
8. What is a DeQueue? Explain its operation with example?
9. Explain the array implementation of queue ADT in detail?
10. Explain the addition and deletion operations performed on a circular queue with necessary algorithms.(8) (Nov 09)

UNIT III

2 MARKS

1.Define tree?

Trees are non-liner data structure, which is used to store data items in a shorted sequence. It represents any hierarchical relationship between any data Item. It is a collection of nodes, which has a distinguish node called the root and zero or more non-empty sub trees T1, **T2,….Tk. each of which are connected by a directed edge from the root.**

2. Define Height of tree?

The height of n is the length of the longest path from root to a leaf. Thus all leaves have height zero. The height of a tree is equal to a height of a root.

3. Define Depth of tree?

For any node n, the depth of n is the length of the unique path from the root to node n. Thus for a root the depth is always zero.

4. What is the length of the path in a tree?

The length of the path is the number of edges on the path. In a tree there is exactly one path form the root to each node.

5. Define sibling?

Nodes with the same parent are called siblings. The nodes with common parents are called siblings.

6. Define binary tree?

A Binary tree is a finite set of data items which is either empty or consists of a single item called root and two disjoin binary trees called left sub tree max degree of any node is two.

7. What are the two methods of binary tree implementation?

Two methods to implement a binary tree are,

a. Linear representation.

b. Linked representation

8. What are the applications of binary tree?

Binary tree is used in data processing.

a. File index schemes

b. Hierarchical database management system

9. List out few of the Application of tree data-structure?

- Ø The manipulation of Arithmetic expression
- Ø Used for Searching Operation
- Ø Used to implement the file system of several popular operating systems
- Ø Symbol Table construction
- Ø Syntax analysis

10. Define expression tree?

Expression tree is also a binary tree in which the leafs terminal nodes or operands and non-terminal intermediate nodes are operators used for traversal.

11. Define tree- traversal and mention the type of traversals?
Visiting of each and every node in the tree exactly is called as tree traversal.

Three types of tree traversal

1. Inorder traversal
2. Preoder traversal
3. Postorder traversal.

12. Define in -order traversal?

In-order traversal entails the following steps;

a. Traverse the left subtree

b. Visit the root node

c. Traverse the right subtree

13. Define threaded binary tree.

A binary tree is threaded by making all right child pointers that would normally be null point to the in order successor of the node, and all left child pointers that would normally be null point to the in order predecessor of the node.

14. What are the types of threaded binary tree?

i. Right-in threaded binary tree
ii. Left-in threaded binary tree
iii. Fully-in threaded binary tree

15. Define Binary Search Tree.

Binary search tree is a binary tree in which for every node X in the tree, the values of all the keys in its left subtree are smaller than the key value in X and the values of all the keys in its right subtree are larger than the key value in X.

16.What is AVL Tree?

AVL stands for Adelson-Velskii and Landis. An AVL tree is a binary search tree which has the following properties:

1.The sub-trees of every node differ in height by at most one.

2.Every sub-tree is an AVL tree.

Search time is O(logn). Addition and deletion operations also take O(logn) time.

17. List out the steps involved in deleting a node from a binary search tree.

- Deleting a node is a leaf node (ie) No children
- Deleting a node with one child.
- Deleting a node with two Childs.

18. What is 'B' Tree?

A B-tree is a tree data structure that keeps data sorted and allows searches, insertions, and deletions in logarithmic amortized time. Unlike self-balancing binary search trees, it is optimized for systems that read and write large blocks of data. It is most commonly used in database and file systems.

19. Define complete binary tree.

If all its levels, possible except the last, have maximum number of nodes and if all the nodes in the last level appear as far left as possible

10 MARKS

1. Explain the AVL tree insertion and deletion with suitable example.
2. Describe the algorithms used to perform single and double rotation on AVL tree.
3. Explain about B-Tree with suitable example.
4. Explain about B+ trees with suitable algorithm.
5. Explain the tree traversal techniques with an example.
6. Construct an expression tree for the expression (a+b*c) + ((d*e+f)*g). Give the outputs when you apply inorder, preorder and postorder traversals.
7. How to insert and delete an element into a binary search tree and write down the code for the insertion routine with an example.
8. What are threaded binary tree? Write an algorithm for inserting a node in a threaded binary tree.
9. Create a binary search tree for the following numbers start from an empty binary search tree. 45,26,10,60,70,30,40 Delete keys 10,60 and 45 one after the other and show the trees at each stage.

UNIT- IV

2 MARKS

1. **Write the definition of weighted graph?**
   A graph in which weights are assigned to every edge is called a weighted graph.

2. **Define Graph?**

A graph G consist of a nonempty set V which is a set of nodes of the graph, a set E which is the set of edges of the graph, and a mapping from the set of edges E to set of pairs of elements of V. It can also be represented as G=(V, E).

3.**Define adjacency matrix?**

The adjacency –matrix is an n x n matrix A whose elements aij are given by
aij = 1 if (vi, vj) Exists =0 otherwise

4.**Define adjacent nodes?**

Any two nodes, which are connected by an edge in a graph, are called adjacent nodes. For example, if an edge x∈E is associated with a pair of nodes

(u,v) where u, v ∈V, then we say that the edge x connects the nodes u and v.

5.**What is a directed graph?**
A graph in which every edge is directed is called a directed graph.

6.**What is an undirected graph?**
A graph in which every edge is undirected is called an undirected graph.

7.**What is a loop?**
An edge of a graph, which connects to itself, is called a loop or sling.

8.**What is a simple graph?**
A simple graph is a graph, which has not more than one edge between a pair of nodes.

9.**What is a weighted graph?**
A graph in which weights are assigned to every edge is called a weighted graph.

10.**Define indegree and out degree of a graph?**

In a directed graph, for any node v, the number of edges, which have v as their initial node, is called the out degree of the node v.

Outdegree: Number of edges having the node v as root node is the outdegree of the node v.

11.Define path in a graph?
The path in a graph is the route taken to reach terminal node from a starting node.

12.**What is a simple path?**

i. A path in a diagram in which the edges are distinct is called a simple path.

ii. It is also called as edge simple.

13.**What is a cycle or a circuit?**

A path which originates and ends in the same node is called a cycle or circuit.

14.**What is an acyclic graph?**

A simple diagram, which does not have any cycles, is called an acyclic graph.

15.**What is meant by strongly connected in a graph?**

An undirected graph is connected, if there is a path from every vertex to every other vertex. A directed graph with this property is called strongly connected.

16.**When a graph said to be weakly connected?**

aij = 1 if (vi, vj) Exists =0 otherwise

When a directed graph is not strongly connected but the underlying graph is connected, then the graph is said to be weakly connected.

17.**Name the different ways of representing a graph? Give examples (Nov 10)**

a. Adjacency matrix

b. Adjacency list

18.**What is an undirected acyclic graph?**

When every edge in an acyclic graph is undirected, it is called an undirected acyclic graph. It is also called as undirected forest.

19.**What is meant by depth?**

The depth of a list is the maximum level attributed to any element with in the list or with in any sub list in the list.

20.**What is the use of BFS?**

BFS can be used to find the shortest distance between some starting node and the remaining nodes of the graph. The shortest distance is the minimum number of edges traversed in order to travel from the start node the specific node being examined.

21.**What is topological sort?**

It is an ordering of the vertices in a directed acyclic graph, such that: If there is a path from u to v, then v appears after u in the ordering.

22.**Write BFS algorithm**

1. Initialize the first node's dist number and place in queue

2. Repeat until all nodes have been examined

3. Remove current node to be examined from queue

4. Find all unlabeled nodes adjacent to current node

5. If this is an unvisited node label it and add it to the queue

6. Finished.

23.**Define biconnected graph?**

A graph is called biconnected if there is no single node whose removal causes the graph to break into two or more pieces. A node whose removal causes the graph to become disconnected is called a cut vertex.

24.**What are the two traversal strategies used in traversing a graph?**

a. Breadth first search
b. Depth first search

25.**Articulation Points (or Cut Vertices) in a Graph**

A vertex in an undirected connected graph is an articulation point (or cut vertex) if removing it (and edges through it) disconnects the graph. Articulation points represent vulnerabilities in a connected network – single points whose failure would split the network into 2 or more disconnected components. They are useful for designing reliable networks.

10 MARKS

1. Explain the various representation of graph with example in detail?
2. Explain Breadth First Search algorithm with example?
3. Explain Depth first and breadth first traversal?
4. What is topological sort? Write an algorithm to perform topological sort?
5. (i) write an algorithm to determine the biconnected components in the given graph.
   (ii)determine the biconnected components in a graph.
6. Explain the various applications of Graphs.

UNIT – V

2 MARKS

1.What is meant by Sorting?

Sorting is ordering of data in an increasing or decreasing fashion according to some linear relationship among the data items.

2. List the different sorting algorithms.

- Bubble sort
- Selection sort
- Insertion sort
- Shell sort
- Quick sort
- Radix sort
- Heap sort
- Merge sort

3. Why bubble sort is called so?

The bubble sort gets its name because as array elements are sorted they gradually **"bubble" to their proper positions, like bubbles risin**g in a glass of soda.

4. State the logic of bubble sort algorithm.

The bubble sort repeatedly compares adjacent elements of an array. The first and second elements are compared and swapped if out of order. Then the second and third elements are compared and swapped if out of order. This sorting process continues until the last two elements of the array are compared and swapped if out of order.

5. What number is always sorted to the top of the list by each pass of the Bubble sort algorithm?

Each pass through the list places the next largest value in its proper place. In essence, each item **"bubbles" up to the location where it belongs.**

6. When does the Bubble Sort Algorithm stop?

The bubble sort stops when it examines the entire array and finds that no "swaps" are needed. The bubble sort keeps track of the occurring swaps by the use of a flag.

7. State the logic of selection sort algorithm.

It finds the lowest value from the collection and moves it to the left. This is repeated until the complete collection is sorted.

8. What is the output of selection sort after the 2nd iteration given the following sequence?

16 3 46 9 28 14

Ans: 3 9 46 16 28 14

9.How does insertion sort algorithm work?

In every iteration an element is compared with all the elements before it. While comparing if it is found that the element can be inserted at a suitable position, then space is created for it by shifting the other elements one position up and inserts the desired element at the suitable position. This procedure is repeated for all the elements in the list until we get the sorted elements.

10.What operation does the insertion sort use to move numbers from the unsorted section to the sorted section of the list?

The Insertion Sort uses the swap operation since it is ordering numbers within a single list.

11. How many key comparisons and assignments an insertion sort makes in its worst case?

The worst case performance in insertion sort occurs when the elements of the input array are in descending order. In that case, the first pass requires one comparison, the second pass **requires two comparisons, third pass three comparisons,....kth pass requires (k-1)**, and finally the last pass requires (n-1) comparisons. Therefore, total numbers of comparisons are:

f(n) = 1+2+3+.....+(n-k)+.....+(n-2)+(n-1) = n(n-1)/2 = O(n2)

12. Which sorting algorithm is best if the list is already sorted? Why?

Insertion sort as there is no movement of data if the list is already sorted and

complexity is of the order O(N).

13. Which sorting algorithm is easily adaptable to singly linked lists? Why?

Insertion sort is easily adaptable to singly linked list. In this method there is an array link of pointers, one for each of the original array elements. Thus the array can be thought of as a linear link list pointed to by an external pointer first initialized to 0. To insert the k^{th} element the linked list is traversed until the proper position for x[k] is found, or until the end of the list is

reached. At that point x[k] can be inserted into the list by merely adjusting the pointers without shifting any elements in the array which reduces insertion time.

14. Why Shell Sort is known diminishing increment sort?

The distance between comparisons decreases as the sorting algorithm runs until the last phase in which adjacent elements are compared. In each step, the sortedness of the sequence is increased, until in the last step it is completely sorted.

15. Which of the following sorting methods would be especially suitable to sort alist L consisting of a sorted list followed by a few "random" elements?

Quick sort is suitable to sort a list L consisting of a sorted list followed by a few "random" elements.

16.What is the output of quick sort after the 3rd iteration given the following sequence?
24 56 47 35 10 90 82 31

Pass 1:- (10) 24 (56 47 35 90 82 31)

Pass 2:- 10 24 (56 47 35 90 82 31)

Pass 3:- 10 24 (47 35 31) 56 (90 82)

17.Mention the different ways to select a pivot element.

The different ways to select a pivot element are

- Pick the first element as pivot
- Pick the last element as pivot
- Pick the Middle element as pivot
- Median-of-three elements
- Pick three elements, and find the median x of these elements
- Use that median as the pivot.
- Randomly pick an element as pivot.

18.What is divide-and-conquer strategy?

- Divide a problem into two or more sub problems
- Solve the sub problems recursively
- Obtain solution to original problem by combining these solutions

19. Compare quick sort and merge sort.

Quicksort has a best-case linear performance when the input is sorted, or nearly sorted. It has a worst-case quadratic performance when the input is sorted in reverse, or nearly sorted in reverse.

Merge sort performance is much more constrained and predictable than the performance of quicksort. The price for that reliability is that the average case of merge sort is slower than the average case of quicksort because the constant factor of merge sort is larger.

20.Define Searching.

Searching for data is one of the fundamental fields of computing. Often, the difference between a fast program and a slow one is the use of a good algorithm for the data set. Naturally, the use of a hash table or binary search tree will result in more efficient searching, but more often than not an array or linked list will be used. It is necessary to understand good ways of searching data structures not designed to support efficient search.

21.What is linear search?

In Linear Search the list is searched sequentially and the position is returned if the key element to be searched is available in the list, otherwise -1 is returned. The search in Linear Search starts at the beginning of an array and move to the end, testing for a match at each item.

22.**What is Binary search?**

A binary search, also called a dichotomizing search, is a digital scheme for locating a specific object in a large set. Each object in the set is given a key. The number of keys is always a power of 2. If there are 32 items in a list, for example, they might be numbered 0 through 31 (binary 00000 through 11111). If there are, say, only 29 items, they can be numbered 0 through 28 (binary 00000 through 11100), with the numbers 29 through31 (binary 11101, 11110, and 11111) as dummy keys.

23.Define hash function?

Hash function takes an identifier and computes the address of that identifier in the hash table using some function.

10 MARKS

1. Write an algorithm to implement Bubble sort with suitable example.
2. Explain any two techniques to overcome hash collision.
3. Write an algorithm to implement insertion sort with suitable example.
4. Write an algorithm to implement selection sort with suitable example.
5. Write an algorithm to implement radix sort with suitable example.
6. Write an algorithm for binary search with suitable example.
7. Discuss the common collision resolution strategies used in closed hashing system.
8. Given the input { 4371, 1323, 6173, 4199, 4344, 9679, 1989 } and a hash function of h(X)=X (mod 10) show the resulting:
    a. Separate Chaining hash table
    b. Open addressing hash table using linear probing
9. Explain Re-hashing and Extendible hashing.

10. Show the result of inserting the keys 2,3,5,7,11,13,15,6,4 into an initially empty extendible hashing data structure with M=3.

11. what are the advantages and disadvantages of various collision resolution strategies?

UNIT TEST PAPER WITH RESULTS

Hindustan College of Science & Technology (064)

Department of Information Technology

Class Test-2 (CT-2)

Subject Name with code: DATA STRUCTURES (KCS-301) **Max Marks: 30**
Sem. /Section: 3rd Sem (IT-A) **Time: 01:30 Hrs.**

Course Name: B.Tech. (IT)

Course Outcomes (COs):

At the end of the course the student should be able to:

1. Describe how arrays, linked lists, stacks, queues, trees, and graphs are represented in memory, used by the algorithms, and their common applications.
2. Discuss the computational efficiency of the sorting and searching algorithms.
3. Implementation of Trees and Graphs and performing various operations on these data structures.
4. Understanding the concept of recursion, application of recursion, and its implementation and removal of recursion.
5. Identify the alternative implementations of data structures with respect to their performance to solve a real-world problem.

Q. No.	Question	Marks	CO	Bloom's Knowledge Level (KL)
Section A Attempt All the parts (No Choice		**(6X1 = 06)**		
A1	Arrange the following computing functions as per their growth rate: **$\log_2 n$, n, $n\log_2 n$, n^2, n^3, 2^n**	1	2	K2
A2	What is a Data Structure? What are the factors that influence the choice of a particular data structure?	1	1	K1
A3	Differentiate between malloc () and calloc () with an example.	1	1	K1
A4	Write a function in 'C' to calculate the sum of diagonal elements in a 2d array.	1	5	K3
A5	Differentiate between Linear and Binary Search in terms of advantages and limitations.	1	2	K4
A6	State whether the following statement is 'True' or 'False' with reason in support of your answer. ***"Structures facilitate the heterogenous collection of data in C"***	1	1	K2
Section B Attempt any 03 Questions from this section.		**(3X5 =15)**		
B1	Write a "C" function to delete a specific node mentioned by a user, from Single Linked List.	5	5	K3
B2	Implement the following functions in 'C' without using any predefined functions: a. Reverse a String b. Copy one String to Another	5	4	K2
B3	Write a "C" program to reverse a Single Linked List.	5	5	K3
B4	Write an algorithm for Selection Sort. Show the step-by-step sorting procedure for the following list of elements: **30, 12, 38, 8, 5, 15, 1, 40.**	5	2	K2

B5	What do you mean by Sparse Matrix? Consider the below matrix and write a program using the function to convert a Sparse Matrix to its Triplet Representation. $\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix}$	5	5	K6
	Section C Attempt any 02 Questions from this section		**(2X4.5 =09)**	
C1	What is a Doubly Linked List? Write the algorithm/function in 'C' to insert an element at the beginning of the Doubly Linked List.	4.5	5	K3
C2	Write a program in 'C' to implement Bubble Sort technique.	4.5	2	K2
C3	Write functions in 'C' for the following: a. Create a single linked list on 'n' nodes b. Count the no. of nodes in single linked list	4.5	1	K2

Hindustan College of Science & Technology (064)

Department of Information Technology

Class Test-1 (CT-1)

Subject Name with code: DATA STRUCTURES (KCS-301) **Max Marks: 30**
Sem. /Section: 3rd Sem (IT-A) **Time: 01:30 Hrs.**

Course Name: B.Tech. (IT)

Course Outcomes (COs):

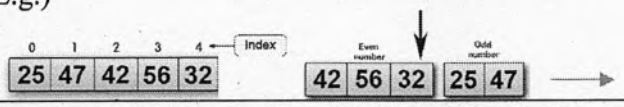
At the end of the course the student should be able to:

1. Describe how arrays, linked lists, stacks, queues, trees, and graphs are represented in memory, used by the algorithms, and their common applications.
2. Discuss the computational efficiency of the sorting and searching algorithms.
3. Implementation of Trees and Graphs and performing various operations on these data structures.
4. Understanding the concept of recursion, application of recursion, and its implementation and removal of recursion.
5. Identify the alternative implementations of data structures with respect to their performance to solve a real-world problem.

Q. No.	Question	Marks	CO	Bloom's Knowledge Level (KL)
Section A Attempt All the parts (No Choice		**(6X1 = 06)**		
A1	Suppose that it is known that the running time of one algorithm is always about N(logN) and the running time of another algorithm is always N^3. What can you say about the relative performance of algorithms?	1	2	K2
A2	What do you understand by ADT? Give example.	1	1	K1

A3	Give using "Big-O" notation, the worst-case running time of the following procedure with its cost and repetition details:
	`p = 0`
	`for (i = 1; p <= n; i++)`
	`{`
	`  p = p + i;`
	`}`
		1	2	K3

A4	What is Environment Stack in reference to Space Complexity?	1	1	K1
A5	What do you understand by Non-Linear Data Structures?	1	1	K2
A6	Why it is beneficial to use dynamic memory allocation instead of static? How can we access Dynamic Memory allocated in RAM?	1	1	K3
Section B Attempt any 03 Questions from this section.		**(3X5 =15)**		
B1	Write a program in "C" to delete an element in a single-dimensional array from a particular location.	5	1	K3
B2	Solve the following: a. Given a 2D array **A [100, 50].** Find the address of element **A [99,49]** considering the base address of 10 and each element requiring 4 bytes for storage. Follow row-major order and column-major order. b. Given the base address of an array **A [-10.... +2]** as 999, each element size is 2 bytes in the memory. Find the address of **A [-1].**	5	1	K2

B3	Write a program in C to separate odd and even integers in separate arrays. (E.g.) 	5	5	K3
B4	Write a program in "C" to perform Matrix Multiplication.	5	1	K3
B5	Write a function in "C" to find the Highest Element in each row of the 2D array and print it.	5	5	K3
	Section C Attempt any 02 Questions from this section		**(2X4.5 =09)**	
C1	Write a program in "C" to find the transpose of a square matrix and check whether they are symmetric or not.	4.5	5	K3
C2	How many ways can we measure any algorithm's growth rate? Explain how to analyze **"Big-O" and "Big-Θ"** notation with its graph and example.	4.5	2	K4
C3	Write a note on the following with the help of a suitable example: a. Contiguous vs Non-Contiguous Structures b. Practical Algorithm Design Issues	4.5	1	K1

DEPARTMENT OF INFORMATION TECHNOLOGY
SLOW & ADVANCED LEARNERS (DATA STRUCTURES) 2017-21

S NO.	ROLL NO.	STUDENT NAME	DS		AVG	STATUS
			KCS301_CT1	KCS301_CT2		
1	1606413023	KASHIF KHAN	0.00	1.00	1.00	SLOW
2	1706413001	ADAM SIRAJ	12.00	12.00	12.00	SLOW
3	1706413002	ADITYA KUMAR	0.00	0.00	0.00	SLOW
4	1706413003	AMAN VERMA	8.00	6.00	7.00	SLOW
5	1706413004	AMBIKESH MISHRA	8.00	10.00	9.00	SLOW
6	1706413005	ANANT CHITRANSH	3.00	8.00	6.00	SLOW
7	1706413006	ANSHU GUPTA	1.00	3.00	2.00	SLOW
8	1706413007	APOORVA SINGH	5.00	9.00	7.00	SLOW
9	1706413008	DEVENDRA BAGHEL	5.00	6.00	6.00	SLOW
10	1706413009	DIVYANSH DIXIT	13.00	11.00	12.00	SLOW
11	1706413010	GAURAV JAIN	3.00	5.00	4.00	SLOW
12	1706413011	HIMANSHU SHARMA	9.00	12.00	11.00	SLOW
13	1706413012	HIMANSHU SAXENA	5.00	4.00	5.00	SLOW
14	1706413013	KRISHAN VEER BAGHEL	6.00	8.00	7.00	SLOW
15	1706413014	MANSI SHARMA	8.00	12.00	10.00	SLOW
16	1706413015	MEHUL CHATURVEDI	8.00	8.00	8.00	SLOW
17	1706413016	MOHIT SHARMA	6.00	6.00	6.00	SLOW
18	1706413017	MONA SHARMA	7.00	5.00	6.00	SLOW
19	1706413018	NANCY SHARMA	8.00	10.00	9.00	SLOW
20	1706413019	NAVIN KUMAR	0.00	12.00	6.00	SLOW
21	1706413020	NIKHIL YADAV	2.00	8.00	5.00	SLOW
22	1706413022	NOORAIN KHAN	7.00	0.00	4.00	SLOW
23	1706413023	PRASHANT PATHAK	8.00	9.00	9.00	SLOW
24	1706413024	PRASHANT KUMAR	3.00	8.00	6.00	SLOW
25	1706413025	PRATIBHA SHARMA	8.00	9.00	9.00	SLOW
26	1706413026	PRIYAM DUBEY	4.00	12.00	8.00	SLOW
27	1706413027	RAJEEV	2.00	6.00	4.00	SLOW
28	1706413028	SAGAR SINGHAL	5.00	8.00	7.00	SLOW
29	1706413029	SAMBHAV GOYAL	12.00	14.00	13.00	AVERAGE
30	1706413030	SARTHAK JAIN	3.00	10.00	7.00	SLOW
31	1706413031	SHIVAM SAXENA	7.00	9.00	8.00	SLOW
32	1706413032	SHIVANI PARASHAR	9.00	0.00	5.00	SLOW
33	1706413033	SHUBHAM RAJPUT	0.00	6.00	3.00	SLOW
34	1706413035	SUYASH GUPTA	16.00	14.00	15.00	AVERAGE
35	1706413037	VISHAL KURELE	3.00	8.00	6.00	SLOW
36	1706413038	VISHPENDRA CHAHAR	5.00	11.00	8.00	SLOW

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:55:27+05'30'
Foxit PhantomPDF Version: 10.1.1

MR. AJAY PARASHAR
SUBJECT TEACHER

DEPARTMENT OF INFORMATION TECHNOLOGY
SLOW & ADVANCED LEARNERS (DATA STRUCTURES) 2018-22

S NO.	ROLL NO.	STUDENT NAME	DS		AVG	STATUS
			KCS301_CT1	KCS301_CT2		
1	1806413001	ADARSH KUMAR JADON	7.00	17.00	12.00	SLOW
2	1806413002	AMAN NAGAR	0.00	0.00	0.00	SLOW
3	1806413004	Aparna Kulshrestha	25.00	25.00	25.00	ADVANCED
4	1806413005	ARIF ALAM	19.00	4.00	12.00	SLOW
5	1806413006	ASEEM MITHWANI	15.00	12.00	14.00	AVERAGE
6	1806413007	AYUSHI GUPTA	16.00	8.00	12.00	SLOW
7	1806413008	Bharat Singh	8.00	13.00	11.00	SLOW
8	1806413010	DIVYANSH JOHARI	6.00	8.00	7.00	SLOW
9	1806413012	HARSH AGARWAL	13.00	17.00	15.00	AVERAGE
10	1806413015	NITIN SINGH RANA	0.00	0.00	0.00	SLOW
11	1806413018	PRANSHI CHAURASIA	10.00	17.00	14.00	AVERAGE
12	1806413019	PRIYANSHU SAXENA	0.00	0.00	0.00	SLOW
13	1806413020	RAJAT SHARMA	23.00	20.00	22.00	ADVANCED
14	1806413021	ROHIT PATHAK	16.00	17.00	17.00	AVERAGE
15	1806413022	SATYAM SINGH RATHORE	12.00	17.00	15.00	AVERAGE
16	1806413023	SAURABH SHUKLA	23.00	21.00	22.00	ADVANCED
17	1806413024	SHARAD SHAKYA	5.00	12.00	9.00	SLOW
18	1806413025	SHIVAM CHOUDHARY	0.00	4.00	2.00	SLOW
19	1806413026	SHIVAM CHAHAR	16.00	14.00	15.00	AVERAGE
20	1806413027	YASH MANGHNANI	19.00	12.00	16.00	AVERAGE

RAJEEV KUMAR UPADHYAY

MR. AJAY PARASHAR
SUBJECT TEACHER

DEPARTMENT OF INFORMATION TECHNOLOGY
SLOW & ADVANCED LEARNERS (DATA STRUCTURES) 2019-23

S NO.	ROLL NO.	STUDENT NAME	DS		AVG	STATUS
			KCS301_CT1	KCS301_CT2		
1	1900640130001	Aman Kumar Soni	17.00	16.00	17.00	AVERAGE
2	1900640130002	Himanshu Moolchandani	14.00	12.00	13.00	AVERAGE
3	1900640130004	Kunal	15.00	14.00	15.00	AVERAGE
4	1900640130005	Sejal Jain	22.00	17.00	20.00	AVERAGE
5	1900640130006	Uravashi Verma	18.00	12.00	15.00	AVERAGE
6	1900640130007	Viplav Kant Rai	17.00	0.00	9.00	SLOW
7	1900640130008	Vivek Sharma	15.00	14.00	15.00	AVERAGE

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:55:04+05'30'
Foxit PhantomPDF Version: 10.1.1

MR. AJAY PARASHAR
SUBJECT TEACHER

DEPARTMENT OF INFORMATION TECHNOLOGY
SLOW & ADVANCED LEARNERS (DATA STRUCTURES) 2020-24

S NO.	ROLL NO.	STUDENT NAME	DS		AVG	STATUS
			KCS301_CT1	KCS301_CT2		
1	2000640130001	Abhay Goyal	0.00	0.00	0.00	SLOW
2	2000640130003	Aman Ahmed	15.00	14.00	15.00	AVERAGE
3	2000640130004	Ayush Mittal	0.00	0.00	0.00	SLOW
4	2000640130005	Pankaj Sakhrani	0.00	0.00	0.00	SLOW
5	2000640130006	Piyusha Varshney	24.00	25.00	25.00	ADVANCED
6	2000640130007	Sandeep Singh	0.00	8.00	4.00	SLOW
7	2000640130008	Tushant Shastri	11.00	4.00	8.00	SLOW
8	2000640130009	Udit Upadhyay	0.00	0.00	0.00	SLOW
9	2000640130010	Umesh Singh	12.00	9.00	11.00	SLOW
10	2000640130011	Vansh Upadhyay	3.00	2.00	3.00	SLOW

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:54:50+05'30'
Foxit PhantomPDF Version: 10.1.1

MR. AJAY PARASHAR
SUBJECT TEACHER

DEPARTMENT OF INFORMATION TECHNOLOGY
SLOW & ADVANCED LEARNERS (DATA STRUCTURES) 2021-25

S NO.	ROLL NO.	STUDENT NAME	DS		AVG	STATUS
			KCS301_CT1	KCS301_CT2		
1	1806413014	KUSHAGRA KULSHRESHTHA	3.00	4.00	4.00	SLOW
2	2100640130002	Adarsh Gautam	0.00	0.00	0.00	SLOW
3	2100640130003	Aditya Pratap Singh	2.00	0.00	1.00	SLOW
4	2100640130004	Aditya Rajpal	0.00	0.00	0.00	SLOW
5	2100640130006	Akshit	9.00	3.00	6.00	SLOW
6	2100640130007	Aman Chaudhary	21.00	12.00	17.00	AVERAGE
7	2100640130008	Ananya Dubey	16.00	0.00	8.00	SLOW
8	2100640130010	Ankit Arora	0.00	0.00	0.00	SLOW
9	2100640130011	Ankit Gautam	0.00	0.00	0.00	SLOW
10	2100640130014	Astha	8.00	8.00	8.00	SLOW
11	2100640130017	Bishakha Kumari Shaw	0.00	14.00	7.00	SLOW
12	2100640130019	Devesh Kumar	23.00	20.00	22.00	ADVANCED
13	2100640130020	Dheeraj Yadav	0.00	0.00	0.00	SLOW
14	2100640130021	Dhwani Agarwal	22.00	21.00	22.00	ADVANCED
15	2100640130023	Diya Gupta	18.00	18.00	18.00	AVERAGE
16	2100640130024	Gaurav Raj	3.00	10.00	7.00	SLOW
17	2100640130025	Govind Tyagi	0.00	0.00	0.00	SLOW
18	2100640130026	Harsh Sharma	0.00	0.00	0.00	SLOW
19	2100640130027	Harsh Tripathi	15.00	0.00	8.00	SLOW
20	2100640130028	Harshul Chawla	21.00	25.00	23.00	ADVANCED
21	2100640130030	Krati Maheshwari	24.00	20.00	22.00	ADVANCED
22	2100640130034	Nikhil Rathore	7.00	0.00	4.00	SLOW
23	2100640130035	Nikunj Maheshwari	17.00	6.00	12.00	SLOW
24	2100640130036	Nishant Dixit	17.00	17.00	17.00	AVERAGE
25	2100640130037	Nishkarsh Rathore	1.00	0.00	1.00	SLOW
26	2100640130039	Palak Sharma	6.00	4.00	5.00	SLOW
27	2100640130041	Parth Singh Sikarwar	0.00	0.00	0.00	SLOW
28	2100640130042	Prashant Pal Singh	0.00	0.00	0.00	SLOW
29	2100640130043	Prashant Upreti	1.00	2.00	2.00	SLOW
30	2100640130044	Prateek Chaudhary	16.00	23.00	20.00	AVERAGE
31	2100640130046	Priyanshi Gupta	12.00	14.00	13.00	AVERAGE
32	2100640130048	Priyanshu Verma	0.00	0.00	0.00	SLOW
33	2100640130049	Rohit Kumar Sharma	0.00	0.00	0.00	SLOW
34	2100640130050	Sachin Gautam	15.00	13.00	14.00	AVERAGE
35	2100640130051	Sahil Ali	12.00	12.00	12.00	SLOW
36	2100640130052	Sakar Talwar	23.00	18.00	21.00	AVERAGE
37	2100640130053	Satyam Singh	20.00	25.00	23.00	ADVANCED
38	2100640130054	Shivani	9.00	0.00	5.00	SLOW
39	2100640130056	Suryansh Singh	0.00	0.00	0.00	SLOW
40	2100640130057	Udit Kukreti	16.00	16.00	16.00	AVERAGE
41	2100640130058	Vikash Solanki	15.00	16.00	16.00	AVERAGE
42		Prateek Mudgal	8.00	12.00	10.00	SLOW

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:54:36+05'30'
Foxit PhantomPDF Version: 10.1.1

43		Priyanshi Chaudhary	12.00	9.00	11.00	SLOW
44		Shivam Sorot	7.00	0.00	4.00	SLOW
45		Mohit Kumar Sharma	0.00	2.00	1.00	SLOW
46		Gagan Deep Baghel	0.00	4.00	2.00	SLOW
47		Sudhanshu Vaibhav	0.00	0.00	0.00	SLOW

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:54:23+05'30'
Foxit PhantomPDF Version: 10.1.1

MR. AJAY PARASHAR
SUBJECT TEACHER

Remedial classes for slow learners

Hindustan College of Science and Technology, Farah, Mathura

Department of Information Technology

Schedule for Subject Wise Academic Counselling/ Remedial Classes

w.e.f: 13th Dec2021

Date	Day	Semester	10:30 AM - 11:30 AM	11:30 AM - 12:30 PM	01:30 PM - 02:30 PM
13th Dec 2021	Monday	III Sem	DS	DSTL	MATHS-IV
14th Dec 2021	Tuesday	III Sem	COA	DS	DSTL
15th Dec 2021	Wednesday	III Sem	DS	DSTL	MATHS-IV

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:54:07+05'30'
Foxit PhantomPDF Version: 10.1.1

HOD, IT

Head
Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

Individual academic counseling is done by concerned subject teacher/counselor

Hindustan College of Science and Technology, Farah, Mathura

Department of Information Technology

Session: 2021-22

Name of Activity	Discussion of topic in faculty cabin/laboratory/library for clearing doubts (Slow Learner Activity)	
Date	a. 13/Dec/21	b. 14/Dec/21
	c. 15/Dec/21	d.
	e.	f.
	g.	h.
Venue	Room No. 230 (IT Dept.)	
Organized by	IT Dept.	
Name of Faculty	Mr. Ajay Raj Parashar	
Participated by	All Slow Learners.	
Content	a. Implementation of LL	b. Tree Traversals / Creation-B Tree
	c. BFS/DFS	d.
	e.	f.
	g.	h.
Objective	To clear doubts and focus on weak students	
Outcome of Activity	Doubts cleared.	

Name & Signature of Faculty

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:53:53+05'30'
Foxit PhantomPDF Version: 10.1.1

HOD, IT

Head
Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

Individual academic counseling is done by concerned subject teacher/counselor

Hindustan College of Science and Technology, Farah, Mathura

Department of Information Technology

Session: 2021-22

Name of Activity	Discussion of topic in faculty cabin/laboratory/library for clearing doubts (Advanced/Average Learner Activity)	
Date	a. 13/Dec/21	b. 14/Dec/21
	c. 15/Dec/21	d.
	e.	f.
	g.	h.
Venue	Room No. 230 (IT Dept.)	
Organized by	IT Dept.	
Name of Faculty	Mr. Ajay Raj Parashar	
Participated by	Advanced / Average Learners.	
Content	a. Gate Topics/ Reversal of LL	b. Advance problems on Trees
	c. Gate quest. on Graphs	d.
	e.	f.
	g.	h.
Objective	To prepare students for advanced questions.	
Outcome of Activity	Many Topics covered.	

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:53:40+05'30'
Foxit PhantomPDF Version: 10.1.1

Name & Signature of Faculty

HOD, IT

Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

Formation of study groups to promote group studies

Hindustan College of Science and Technology, Farah, Mathura

Department of Information Technology

III Sem, IT-A (2021-22), ODD Sem / Study Group for Data Structures

S.No.	Roll No	Student Name	Group
1	2000640130001	ABHAY GOYAL	G1
2	2000640130003	AMAN AHMED	
3	2000640130004	AYUSH MITTAL	
4	2000640130005	PANKAJ SAKHRANI	
5	2000640130006	PIYUSHA VARSHNEY	G2
6	2000640130007	SANDEEP SINGH	
7	2000640130009	UDIT UPDAHYAY	
8	2000640130008	TUSHANT SHASTRI	G3
9	2000640130010	UMESH SINGH	
10	2000640130011	VANSH UPADHYAY	

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:53:29+05'30'
Foxit PhantomPDF Version: 10.1.1

Name & Signature of Faculty

HOD, IT

Head
Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

Initiatives Taken to Improve the performance of Weak Students

Subject: Data Structures / Semester:IIISection:A

Subject Code: KCS-301

The following efforts were taken to improve the quality of weak students:

1. Basics of the subjects were cleared through remedial classes.
2. Notes on important topics were provided.
3. Theoretical portions were explained through presentations.
4. Programs are explained by performing live coding in front of students.

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:53:18+05'30'
Foxit PhantomPDF Version: 10.1.1

Mr. Ajay Raj Parashar
Subject Teacher

DR. SHANKAR THAWKAR
HOD, IT

Head
Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

SUBJECT NOTES

UNIT-1

Unit-01

Introduction to Data Structures

Data: Any raw or unorganized facts and figures (such as alphabets, numbers, or symbols).

- They refer to, or represent, conditions, ideas, or objects.
- Data is limitless and present everywhere in the universe.
- Data are plain facts. The word "data" is plural for "datum."

Information: Stimuli that has meaning in some context for its receiver.

- Information is data that has been converted into a more useful or intelligible form.
- It is the set of data that has been organized for direct utilization of mankind, as information helps human beings in their decision-making process.
- Examples are: Time Table, Merit List, and Report card etc.

A **Data type** refers to a named group of data which share similar properties or characteristics and which have common behavior among them. Three fundamental data types used in C programming are int for integer values, float for floating-point numbers and char for character values.

But, sometimes a need arises to treat a group of different data types as a single unit. **Data Structure** is a way of collecting and organizing data in such a way that we can perform operations on these data in an effective way.
For example, we have data player's name "Virat" and age 26. Here "Virat" is of String data type and 26 is of integer data type.

We can organize this data as a record like Player record. Now we can collect and store player's records in a file or database as a data structure. For example: "Dhoni" 30, "Gambhir" 31, "Sehwag" 33

Data structure is representation of the logical relationship existing between individual elements of data.

or

In other words, a Data structure is a way of organizing all data items that considers not only the elements stored but also their relationship to each other.

Data structures are the building blocks of a program. And hence the selection of particular data structure stresses on the following two things:

- The data structures must be rich enough in structure to reflect the relationship existing between the data.
- And the structure should be simple so that we can process data effectively whenever required.

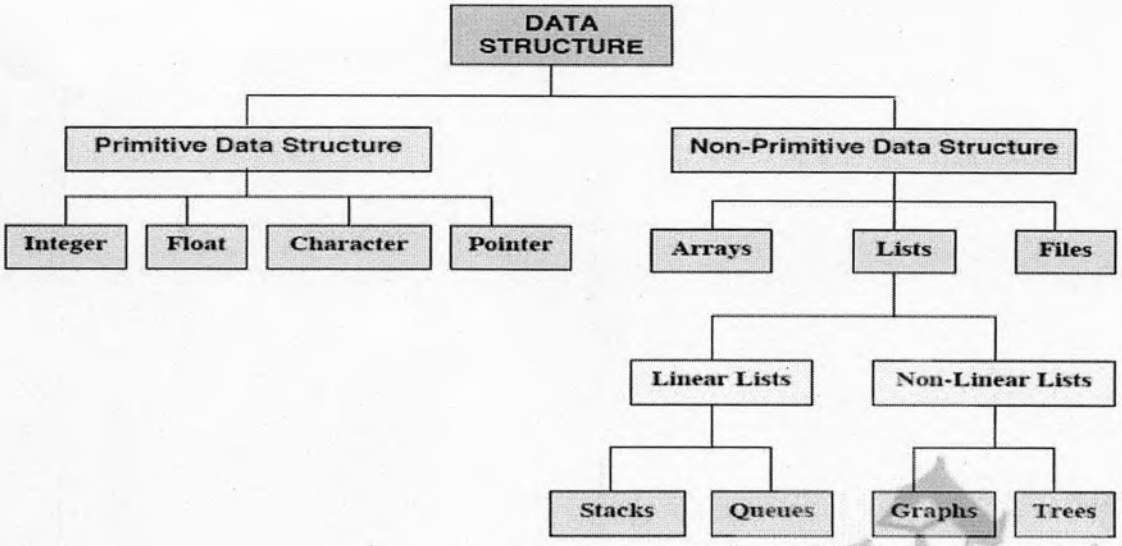
Data structures affect the design of both structural and functional aspects of a program.

Algorithm + Data structure = Program

Algorithm is a step-by-step procedure to solve a particular function i.e., it is a set of instructions written to carry out certain tasks and the data structure is the way of organizing the data with their logical relationship maintained.

Classification of Data Structures:
Data structures are divided into two types:

1. Primitive data structures (built-in)
2. Non-primitive data structures (user defined)



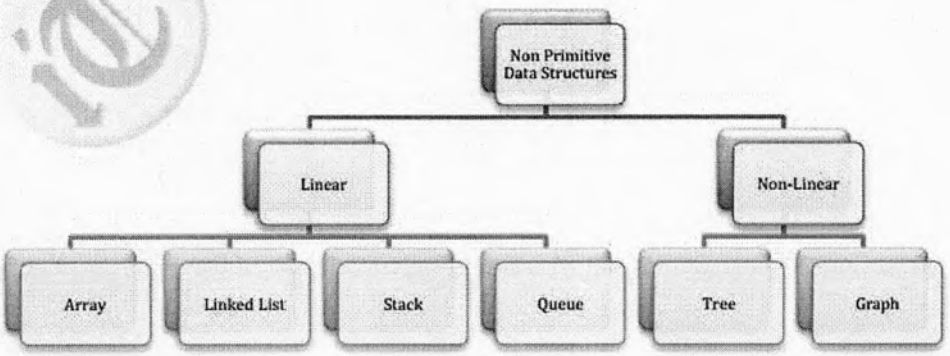
Primitive Data Structures are the basic data structures that **directly operate upon the machine instructions**. They have different representations on different computers. Integers, floating-point numbers, character-constants, string constants and pointers come under this category.

Non-primitive Data Structures are more complicated data structures and are **derived from primitive data** structures. They emphasize on grouping same or different data items with relationship between each data item. Arrays, lists and files come under this category.

Based on the structure and arrangement of data, non-primitive data structures can be further classified into:

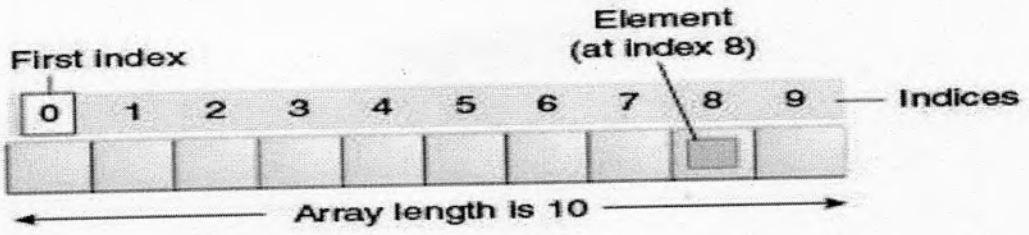
Linear: A data structures is said to be linear if its elements form a sequence or linear list. Although the way they are stored in memory need not be in a sequential manner.
E.g. Array, Linked List, Stack, Queue

Non-Linear: A data structure is said to be non linear if the data is not arranged in sequence. The insertion and deletion is not possible in linear fashion.
E.g. Trees, Graphs

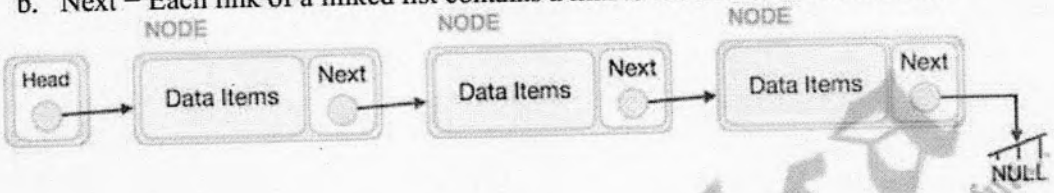


Linear Data Structures

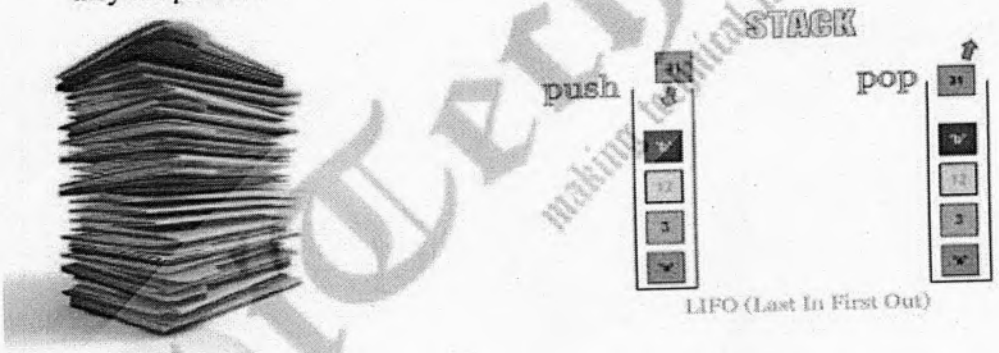
1. **Arrays:** Arrays a kind of data structure that is used to store a collection of data of the same type. All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



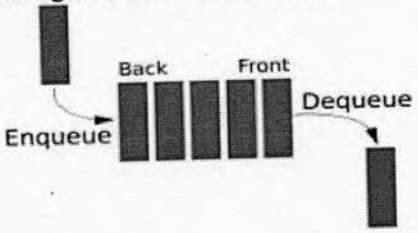
2. **Linked List:** It is a sequence of links (nodes), which contains items. Each link contains a connection to another link. Following are the important terms to understand the concept of Linked List.
   a. Data – Each link of a linked list can store a data called an element.
   b. Next – Each link of a linked list contains a link to the next link called Next.



3. **Stack:** It is a linear data structure, which follows a particular order in which the operations are performed. The order may be LIFO (Last In First Out) or FILO(First In Last Out). Mainly the following operations are performed in the stack:
   a. **Push:** Adds an item in the stack.
   b. **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed.

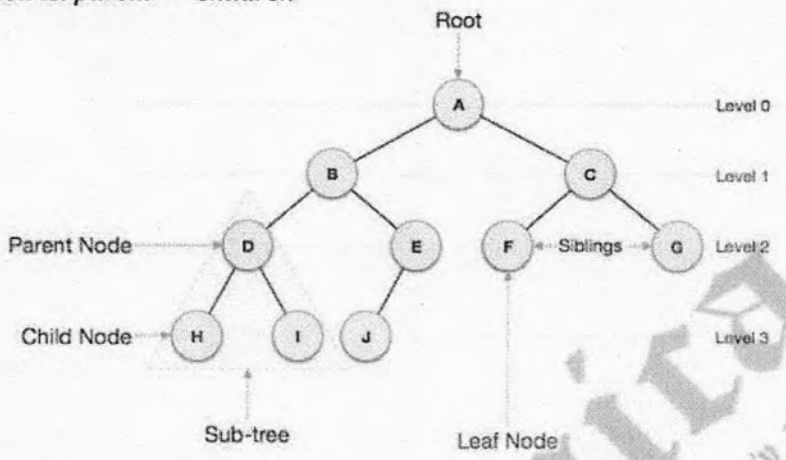


4. **Queue:** A queue or FIFO (first in, first out) is an abstract data type that serves as a collection of elements, with two principal operations:
   a. **enqueue,** the process of adding an element to the collection.(The element is added from the rear side) and
   b. **dequeue,** the process of removing the first element that was added. (The element is removed from the front side). It can be implemented by using both array and linked list.

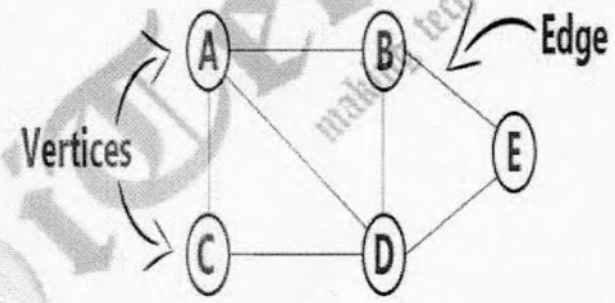


Non-Linear Data Structures

5. **Tree:** A tree is a *nonlinear* data structure; it can be empty with no nodes or a tree is a structure consisting of one node called the root and zero or one or more sub trees. A tree has following general properties:
   a. One node is distinguished as a root;
   b. Every node (exclude a root) is connected by a directed edge *from* exactly one other node; A direction is: *parent -> children*



6. **Graph:** Graph is a data structure that consists of following two components:
   a. A finite set of vertices also called as nodes.
   b. A finite set of ordered pair of the form (u, v) called as edge. The pair is ordered because (u, v) is not same as (v, u) in case of directed graph (di-graph). The pair of form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.



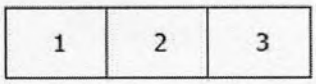
Organization of data

The collections of data you work with in a program have some kind of structure or organization. No matter how complex your data structures are they can be broken down into two fundamental types:

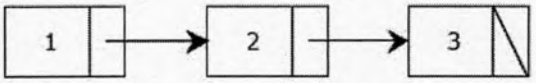
Contiguous: In contiguous structures, terms of data are kept together in memory (either RAM or in a file). An array is an example of a contiguous structure. Since each element in the array is located next to one or two other elements.

Non-Contiguous: In contrast, items in a non-contiguous structure are scattered in memory, but we linked to each other in some way. A linked list is an example of a non-contiguous data structure. Here, the nodes of the list are linked together using pointers stored in each node.

Figure below illustrates the difference between contiguous and non-contiguous structures.



(a) Contiguous



(b) non-contiguous

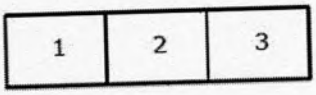
Contiguous structures:

Contiguous structures can be broken drawn further into two kinds:

- Those that contain data items of all the same size, and those where the size may differ. The first kind is called the array. Figure shows an example of an array of numbers. In an array, each element is of the same type, and thus has the same size.

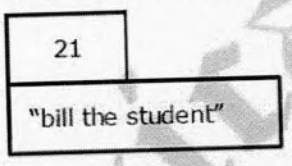
- The second kind of contiguous structure is called structure; figure (b) shows a simple structure consisting of a person's name and age. In a struct, elements may be of different data types and thus may have different sizes.

```
int arr[3] = {1, 2, 3};
```



(a) Array

```
struct cust_data
{
        int age;
        char name[20];
};

cust_data bill= {21, "bill the student"};
```

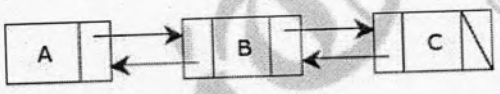


(b) struct

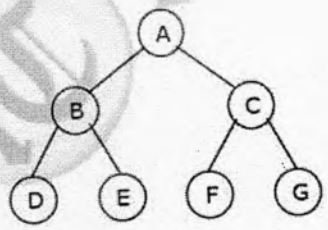
Non-contiguous structures:

Non-contiguous structures are implemented as a collection of data-items, called nodes, where each node can point to one or more other nodes in the collection.

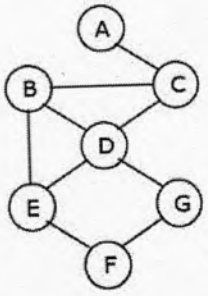
- The simplest kind of non-contiguous structure is linked list. A linked list represents a linear, one-dimension type of non-contiguous structure, where there is only the notation of backwards and forwards.
- A tree such is an example of a two-dimensional non-contiguous structure. Here, there is the notion of up and down and left and right. In a tree each node has only one link that leads into the node and links can only go down the tree.
- The most general type of non-contiguous structure, called a graph has no such restrictions.



(a) Linked List



(b) Tree



(c) graph

Basic Data Structure Operations:

The basic operations that are performed on data structures are as follows:

1. **Traversing:** Accessing each record exactly once so that certain items in the record may be processed. (This accessing and processing is sometimes called "visiting" the record).
2. **Searching:** Searching operation finds the presence of the desired data item in the list of data item. It may also find the locations of all elements that satisfy certain conditions.
3. **Inserting:** Inserting means addition of a new data element in a data structure.
4. **Deleting:** Deleting means removal of a data element from a data structure.

The following two operations, which are used in special situations, will also be considered:

5. **Sorting:** Sorting is the process of arranging all data items in a data structure in a particular order say for example, either in ascending order or in descending order.
6. **Merging:** Combining the records of two different sorted files into a single sorted file.

Algorithm:

An algorithm is a finite sequence of instructions, each of which has a clear meaning and can be performed with a finite amount of effort in a finite length of time.

No matter what the input values may be, an algorithm terminates after executing a finite number of instructions. In addition every algorithm must satisfy the following criteria:

- **Input:** there are zero or more quantities, which are externally supplied;
- **Output:** at least one quantity is produced;
- **Definiteness:** each instruction must be clear and unambiguous;
- **Finiteness:** if we trace out the instructions of an algorithm, then for all cases the algorithm will terminate after a finite number of steps;
- **Effectiveness:** every instruction must be sufficiently basic that it can in principle be carried out by a person using only pencil and paper. It is not enough that each operation is definite, but it must also be feasible.

Practical Algorithm design issues:

Choosing an efficient algorithm or data structure is just one part of the design process. There are two basic design goals that we should strive for in a program:

1. Try to save time (Time complexity).
2. Try to save space (Space complexity).

The performance of a program is the amount of computer memory and time needed to run a program.

Time Complexity:

The time needed by an algorithm expressed as a function of the size of a problem is called the TIME COMPLEXITY of the algorithm. **The time complexity of a program is the amount of computer time it needs to run to completion.**

The limiting behavior of the complexity as size increases is called the **asymptotic time complexity.**

Space Complexity:

The space complexity of a program is the amount of memory it needs to run to completion. The space need by a program has the following components:

- **Instruction space:** Instruction space is the space needed to store the compiled version of the program instructions. The amount of instructions space that is needed depends on factors such as:
  - The compiler used to complete the program into machine code.
  - The compiler options in effect at the time of compilation.
  - The target computer.
- **Data space:** Data space is the space needed to store all constant and variable values. Data space has two components:
  - Space needed by constants and simple variables in program.
  - Space needed by dynamically allocated objects such as arrays and class instances.
- **Environment stack space:** The environment stack is used to save information needed to resume execution of partially completed functions.

Complexity of Algorithms

The complexity of an algorithm M is the function f(n) which gives the running time and/or storage space requirement of the algorithm in terms of the size 'n' of the input data.

Mostly, the storage space required by an algorithm is simply a multiple of the data size 'n'. Complexity shall refer to the running time of the algorithm.

The function f(n), gives the running time of an algorithm, depends not only on the size 'n' of the input data but also on the particular data. The complexity function f(n) for certain cases are:

1. **Best Case:** The minimum possible value of f(n) is called the best case.
2. **Average Case:** The expected value of f(n).
3. **Worst Case:** The maximum value of f(n) for any key possible input.

The field of computer science, which studies efficiency of algorithms, is known as analysis of algorithms.

Time Complexity Analysis:
To understand the concept of time complexity, let us consider we have two students and they have been assigned the task of finding whether a no is prime or not. Both of them write their programs as follows:

Ram
```
for i=2 to n-1
        if i divides n
        n is not prime
```

Shyam
```
for i=2 to √n
        if i divides n
        n is not prime
```

Let us consider the else case is same for both of them.

Now, when they run their programs for larger inputs RAM became sad while shyam is happy. Let's see why?

Let us consider computer takes 1 ms for a division operation. In worst case, ram's loop will run (n-2) times while shyam's loop will execute (√n-1) times.

n	(n-2) times	(√n-1) times
11	9 times = 9 ms	(√11-1) ≅ 3-1= 2 ms
101	101-2 = 99 ms	(√101-1) ≅ 10-1= 9 ms
1000003 = 10^6+3	10^6+3-2 ≅ 10^6 ms =10^3 sec =16.66 mins	(√10^6+3-1) ≅ 10^3 ms =1 sec
10000000019= 10^{10}+19	≅ 10^{10} ms =10^7 s=115 days	(√10^{10}+19-1) ≅ 10^5 ms =100 sec= 1.66 mins

So, we can see that for very large inputs RAM's program will take much longer time as compared to SHYAM's.
In general, we can say that Time Complexity says **"How fast the time taken by program increases in proportion of input n".**

Time complexity of an algorithm can be defined as follows...
The time complexity of an algorithm is the total amount of time required by an algorithm to complete its execution.
Generally, running time of an algorithm depends upon the following...

1. Whether it is running on **Single** processor machine or **Multi** processor machine.
2. Whether it is a **32 bit** machine or **64 bit** machine
3. **Read** and **Write** speed of the machine.
4. The time it takes to perform **Arithmetic** operations, **logical** operations, **return** value and **assignment** operations etc.,
5. **Input** data

Calculating Time Complexity of an algorithm based on the system configuration is a very difficult task because, the configuration changes from one system to another system. To solve this problem, we must assume a model machine with specific configuration. So that, we can able to calculate generalized time complexity according to that model machine.

To calculate time complexity of an algorithm, we need to define a model machine. Let us assume a machine with following configuration...

1. Single processor machine
2. 32-bit vs 64-bit Operating System machine
3. It performs sequential execution
4. It requires 1 unit of time for Arithmetic and Logical operations
5. It requires 1 unit of time for Assignment and Return value
6. It requires 1 unit of time for Read and Write operations

Now, we calculate the time complexity of following example code by using the above defined model machine...

Example 1
Consider the following piece of code...

```
int sum (int a, int b)
{
return a+b;
}
```

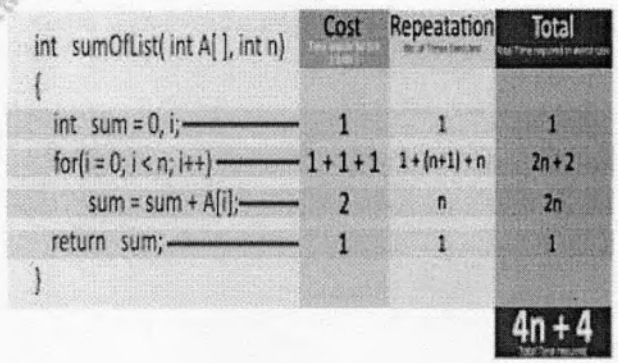
In above sample code:

- It requires 1 unit of time to calculate a+b and
- 1 unit of time to return the value.
- That means, totally it takes 2 units of time to complete its execution. And it does not change based on the input values of a and b. That means for all input values, it requires same amount of time i.e. 2 units.

If any program requires fixed amount of time for all input values then its time complexity is said to be Constant Time Complexity.

Example 2
Consider the following piece of code...

```
int sum(int A[], int n)
{
int sum = 0, i;
for(i = 0; i < n; i++)
sum = sum + A[i];
return sum;
}
```



For the above code, time complexity can be calculated as follows...

In above calculation

- **Cost** is the amount of computer time required for a single operation in each line.
- **Repetition** is the amount of computer time required by each operation for all its repetitions.
- **Total** is the amount of computer time required by each operation to execute.

So above code requires **'4n+4' Units** of computer time to complete the task. Here the exact time is not fixed and it changes based on the **n** value. If we increase the **n** value then the time required also increases linearly. Totally it takes '4n+4' units of time to complete its execution and it is *Linear Time Complexity*.

If the amount of time required by an algorithm is increased with the increase of input value then that time complexity is said to be Linear Time Complexity.

What is Asymptotic Notation?

Whenever we want to perform analysis of an algorithm, we need to calculate the complexity of that algorithm. But when we calculate complexity of an algorithm it does not provide exact amount of resource required. So instead of taking exact amount of resource we represent that complexity in a general form (Notation), which produces the basic nature of that algorithm. We use that general form (Notation) for analysis process.

Asymptotic notation of an algorithm is a mathematical representation of its complexity.

NOTE

In asymptotic notation, when we want to represent the complexity of an algorithm, we use only the most significant terms in the complexity of that algorithm and ignore least significant terms in the complexity of that algorithm (Here complexity may be Space Complexity or Time Complexity).

For example, consider the following time complexities of two algorithms...

- **Algorihtm 1 : $5n^2 + 2n + 1$**
- **Algorihtm 2 : $10n^2 + 8n + 3$**

Generally, when we analyze an algorithm, we consider the time complexity for larger values of input data (i.e. '**n**' value). In above two time complexities, for larger value of '**n**' the term in algorithm 1 '**2n + 1**' has least significance than the term '**$5n^2$**', and the term in algorithm 2 '**8n + 3**' has least significance than the term '**$10n^2$**'.

Here for larger value of '**n**' the value of most significant terms (**$5n^2$** and **$10n^2$**) is very larger than the value of least significant terms (**2n + 1** and **8n + 3**). So for larger value of '**n**' we ignore the least significant terms to represent overall time required by an algorithm. In asymptotic notation, we use only the most significant terms to represent the time complexity of an algorithm.

Majorly, we use THREE types of Asymptotic Notations and those are as follows...

1. **Big - Oh (O)**
2. **Big - Omega (Ω)**
3. **Big - Theta (Θ)**

Big - Oh Notation (O)

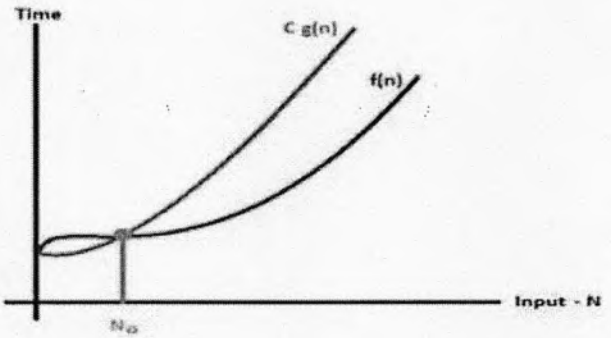
Big - Oh notation is used to define the **upper bound** of an algorithm in terms of Time Complexity. That means Big - Oh notation always indicates the maximum time required by an algorithm for all input values. That means Big - Oh notation describes the worst case of an algorithm time complexity.

Big - Oh Notation can be defined as follows...

Consider function **f(n)** the time complexity of an algorithm and **g(n)** is the most significant term. If **f(n) <= C g(n)** for all **n >= n_0**, **C > 0** and **n_0 >= 1**. Then we can represent **f(n)** as **O(g(n))**.

f(n) = O(g(n))

Consider the following graph drawn for the values of f(n) and C g(n) for input (n) value on X-Axis and time required is on Y-Axis



In above graph after a particular input value n_0, always C g(n) is greater than f(n) which indicates the algorithm's upper bound.

Example

Consider the following f(n) and g(n)...

f(n) = 3n + 2

g(n) = n

If we want to represent **f(n)** as **O(g(n))** then it must satisfy **f(n) <= C x g(n)** for all values of **C > 0** and **n_0>= 1**

f(n) <= C g(n)

⇒3n + 2 <= C n

Above condition is always TRUE for all values of **C = 4** and **n >= 2**.

By using Big - Oh notation we can represent the time complexity as follows...

3n + 2 = O(n)

Big - Omega Notation (Ω)

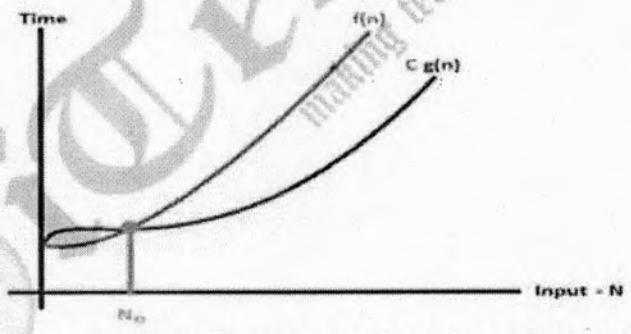
Big - Omega notation is used to define the **lower bound** of an algorithm in terms of Time Complexity. That means Big - Omega notation always indicates the minimum time required by an algorithm for all input values. That means Big - Omega notation describes the best case of an algorithm time complexity.

Big - Omega Notation can be defined as follows...

Consider function **f(n)** the time complexity of an algorithm and **g(n)** is the most significant term. If **f(n) >= C x g(n)** for all **n >= n_0, C > 0** and **n_0 >= 1**. Then we can represent f(n) as **Ω(g(n))**.

f(n) = Ω(g(n))

Consider the following graph drawn for the values of f(n) and C g(n) for input (n) value on X-Axis and time required is on Y-Axis



In above graph after a particular input value n_0, always C x g(n) is less than f(n) which indicates the algorithm's lower bound.

Example

Consider the following f(n) and g(n)...

f(n) = 3n + 2

g(n) = n

If we want to represent **f(n)** as **Ω(g(n))** then it must satisfy **f(n) >= C g(n)** for all values of **C > 0** and **n_0>= 1**

f(n) >= C g(n)

⇒3n + 2 <= C n

Above condition is always TRUE for all values of **C = 1** and **n >= 1**.

By using Big - Omega notation we can represent the time complexity as follows...

3n + 2 = Ω(n)

Big - Theta Notation (Θ)

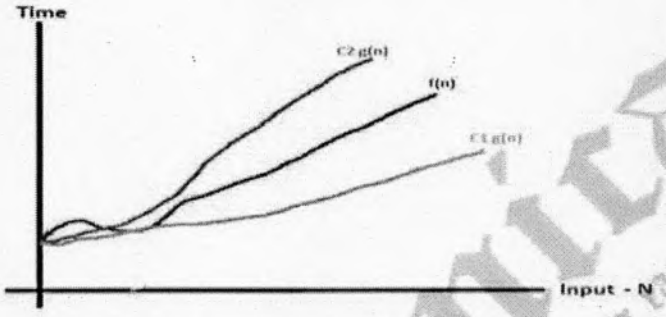
Big - Theta notation is used to define the **average bound** of an algorithm in terms of Time Complexity. That means Big - Theta notation always indicates the average time required by an algorithm for all input values. That means Big - Theta notation describes the average case of an algorithm time complexity.

Big - Theta Notation can be defined as follows...

Consider function **f(n)** the time complexity of an algorithm and **g(n)** is the most significant term. If **C_1 g(n) <= f(n) >= C_2 g(n)** for all **n >= n_0, C_1, C_2 > 0** and **n_0 >= 1**. Then we can represent **f(n)** as **Θ(g(n))**.

f(n) = Θ(g(n))

Consider the following graph drawn for the values of f(n) and C g(n) for input (n) value on X-Axis and time required is on Y-Axis



In above graph after a particular input value n_0, always **C_1 g(n)** is less than **f(n)** and **C_2 g(n)** is greater than **f(n)** which indicates the algorithm's average bound.

Example

Consider the following f(n) and g(n)...

f(n) = 3n + 2

g(n) = n

If we want to represent **f(n)** as **Θ(g(n))** then it must satisfy **C_1 g(n) <= f(n) >= C_2 g(n)** for all values of **C_1, C_2 > 0** and **n_0>= 1**

$C_1 g(n) <= f(n) >= \Rightarrow C_2 g(n)$

$C_1 n <= 3n + 2 >= C_2 n$

Above condition is always TRUE for all values of **C_1 = 1, C_2 = 4** and **n >= 1**.

By using Big - Theta notation we can represent the time complexity as follows...

3n + 2 = Θ(n)

Some General Rules:

To analyze time complexity, we follow some rules as we analyze any algorithm for very large inputs.

1. Drop lower order terms
2. Drop constant multipliers
3. Running time of a program = $\sum$ Running time of all fragments

Example:

1. **$T(n)=17n^4+3n^3+4n+8$**
   Then drop $3n^3+4n+8$ as they will grow very slow as compared to $17n^4$ for very large inputs.

   So, **$O(n^4)$**

2. **T(n)=16n+log(n)**
   Then drop log(n) as they will grow very slow as compared to 16n for very large inputs.

   So, **O(n)**

3. **int a;**
   a=5;

```
a++;
```

O(1) // Constant time

4.
```
for(i=0;i<n;i++)
{
//
//
}
```

O(n) //as it will grow with respect to n.

5.
```
for(i=0;i<n;i++)
{
        for(j=0;j<n;j++)
        {
        //
        //
        }
}
```

O(n^2) //as it will grow with respect to nxn.

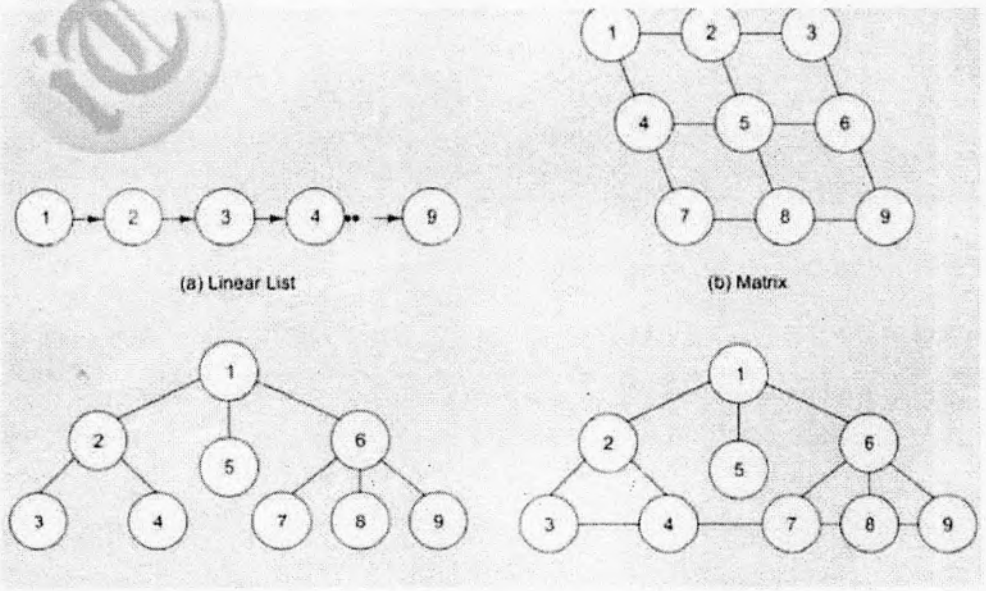
6.
```
int a;
a=5;
a++;

for(i=0;i<n;i++)
{
//
//
}

for(i=0;i<n;i++)
{
        for(j=0;j<n;j++)
        {
        //
        //
        }
}
```

Overall T(n) = O(1) + O(n) + O(n^2) = O(n^2)

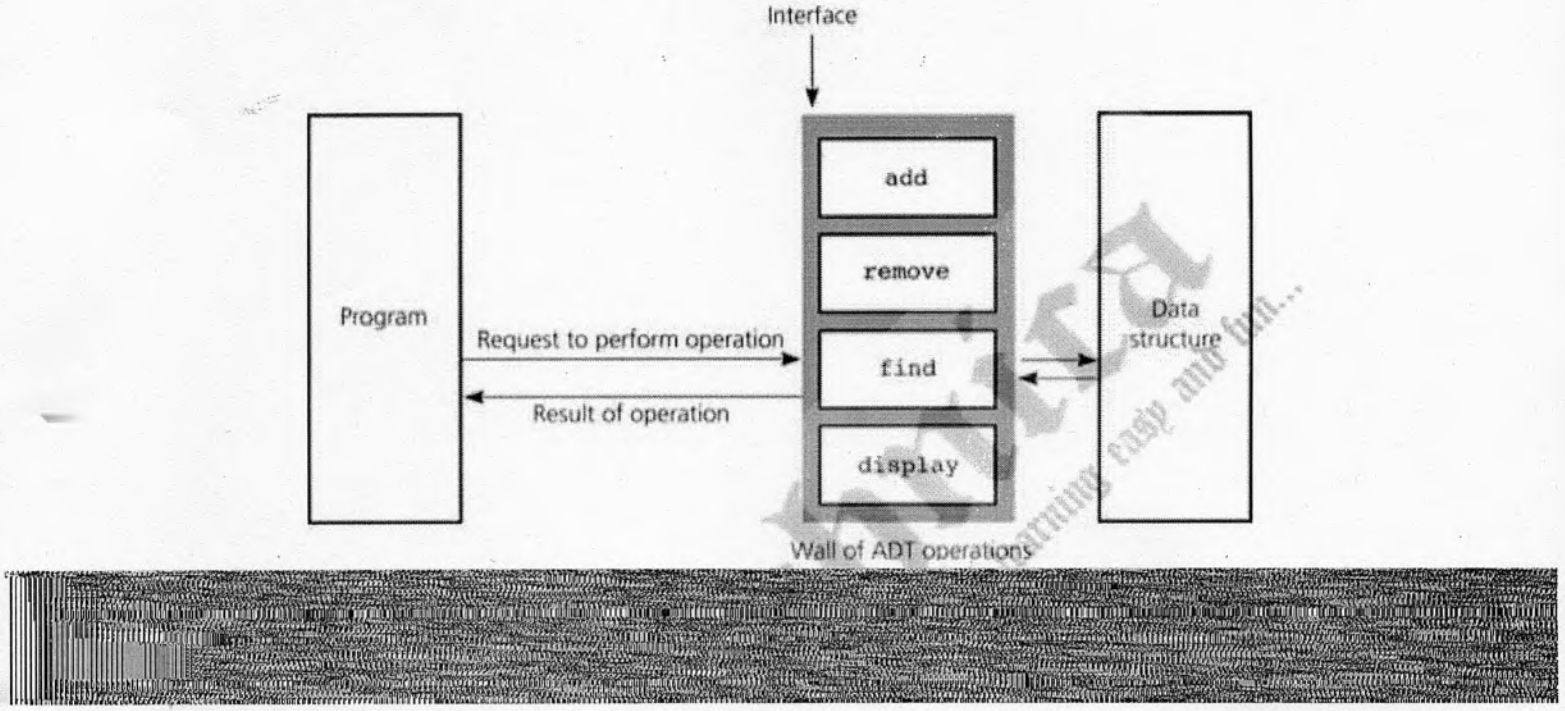
Abstract Data Types:

- An ADT refers to a set of data values and associated operations that are specified accurately, independent of any particular implementation.
- With an ADT, we only know what a specific data type can do, but how it actually does is hidden.

- **E.g.** Consider a list L of data items- 1,2,3,4,5,6,7,8,9 as shown if figure (a). We can use any data structure to support L -> a
    - linear list,
    - matrix,
    - tree and
    - graph.



- So, An ADT is composed of
  - A collection of data
  - A set of operations on that data
- Specifications of an ADT indicate
  - What the ADT operations do, not how to implement them
- Implementation of an ADT
  - Includes choosing a particular data structure



Where *datatype* declares the base type of array, which is the type of each element in the array. The *array-name* specifies the name with which the array will be referenced and *size* defines how many elements the array will hold. The size must be an integer value or integer constant without any sign.

For e.g.

int marks [10];

The above statement declared array **marks** with 10 elements, marks [0] to marks [9].

Initialization of array:

data_type array-name[size]= {element-1, element-2, ,.....,element-n};

or

data_type array-name [ ]={element-1,element-2,.....,element-n};

For example:

```
int marks[5]={50,25,72,45,30};
Marks[0]=50;
Marks[1]=25;
Marks[2]=72;
Marks[3]=45;
Marks[4]=30;
or
int marks[ ]={50,25,72,45,30};
```

Also float price[ ] = {300.50, 250.50, 500.50, 175.50, 900.50};
and char grade[ ] = {'D' , 'A' , 'C' , 'B' , 'A' , 'C' };

Accessing an element at a particular index for one dimensional arrays

Individual element of an array can be accessed using the following syntax:

Implementation of one-dimensional array in memory (Address Calculation):

First Method

The address of a particular element in one-dimensional array is given by the relation:

Address of element A[I] = B+ W * I

- Where **B** is the base address of the array,
- **W** is the size of each element in array, and
- **I** is the number of required element in the array (index of element) which should be a integer quantity.

For example:
Let the base address of first element of the array is 2000 (i.e. base address B is = 2000), and each element of the array occupies four bytes in the memory, then address of fifth element of a one-dimensional array a[4] will be given as :

Address of element a[4] = 2000 + 4 * 4
= 2000 + 16
= 2016

Second Method (When Upper & Lower Bounds are given)

Array of an element of an array say "A[I]" is calculated using the following formula:
Address of A [I] = B + W * (I – LB)
Where,
B = Base address
W = Storage Size of one element stored in the array (in byte)
I = Subscript of element whose address is to be found
LB = Lower limit / Lower Bound of subscript, if not specified assume 0 (zero)

Example:
Given the base address of an array **B[1300…..1900]** as 1020 and size of each element is 2 bytes in the memory. Find the address of **B[1700]**.
Solution:
The given values are: B = 1020, LB = 1300, W = 2, I = 1700
Address of A [I] = B + W * (I – LB)
= 1020 + 2 * (1700 – 1300)
= 1020 + 2 * 400
= 1020 + 800
= 1820 **[Ans]**

PROGRAMS (SINGLE DIMENSIONAL ARRAYS)

Program 1 : WAP for array initialization

```
#include<stdio.h>
#include<conio.h>
void main( )
{
// Array declaration and initialization
int marks[ ]={50,60,70,80,90},i;
// Array output
printf("\nMarks of 5 students are : \n");
for(i=0;i<5;i++)
printf("%d\t",marks[i]);
getch( );
}
```

OUTPUT :
Marks of 5 students are :
50 60 70 80 90

Program 2 : WAP for array input from user

```
#include<stdio.h>
#include<conio.h>
void main( )
{
// Array declaration
int marks[5],i;
// Array input
printf("Enter marks of 5 students : \n");
for(i=0;i<5;i++)
scanf("%d",&marks[i]);
// Array output
printf("\nMarks of 5 students are : \n");
for(i=0;i<5;i++)
printf("%d\t",marks[i]);
getch( );
}
```

OUTPUT :
Enter marks of 5 students :
50 60 70 80 90
Marks of 5 students are :
50 60 70 80 90

Program 3 : WAP to display the sum and average of elements of an array

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
int A[5],i;
float sum=0,avg;
printf("Enter 5 elements of an array : \n");
for(i=0;i<5;i++)
scanf("%d",&A[i]);
for(i=0;i<5;i++)
sum=sum+A[i];
avg=sum/5;
printf("\nSum : %.2f",sum);
printf("\nAverage : %.2f",avg);
getch( );
}
```

OUTPUT :
Enter 5 elements of an array :
2 4 6 8 10
Sum : 30.00
Average : 6.00

Program 4 : WAP to display the largest and smallest element of an array

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
int A[5],i,max,min;
printf("Enter 5 elements of an array : \n");
for(i=0;i<5;i++)
scanf("%d",&A[i]);
max=min=A[0];
for(i=0;i<5;i++)
{
if(max<A[i])
max=A[i];
if(min>A[i])
min=A[i];
}
printf("\nLargest element in array : %d",max);
printf("\nSmallest element in array : %d",min);
getch( );
}
```

OUTPUT :

```
Enter 5 elements of an array :
3 9 5 1 8
Largest element in array : 9
Smallest element in array : 1
```

Program 5 : WAP to insert a number in an array

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
int len;
printf("Enter size of array (max. 10) : ");
scanf("%d",&len);
int A[10],num,i,pos;
printf("\nEnter %d elements of array : \n",len);
for(i=0;i<len;i++)
scanf("%d",&A[i]);
printf("\nOriginal array is : \n");
for(i=0;i<len;i++)
printf("%d\t",A[i]);
printf("\n\nEnter the element to be inserted : ");
scanf("%d",&num);
printf("Enter the position of insertion : ");
scanf("%d",&pos);
pos--;
for(i=len-1;i>=pos;i--) // Shifts down one position
A[i+1]=A[i];
A[pos]=num;
if(pos>len)
```

```
printf("\nInsertion outside the array");
else
{
printf("\nNew array after insertion : \n");
len++;
for(i=0;i<len;i++)
printf("%d\t",A[i]);
}
getch( );
}
```

OUTPUT :

```
Enter size of array (max. 10) : 5
Enter 5 elements of array :
2 4 8 10 12
Original array is :
2 4 8 10 12
Enter the element to be inserted : 6
Enter the position of insertion : 3
New array after insertion :
2 4 6 8 10 12
```

Program 6 : WAP to delete a number from an array

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
int A[10],i,len,num,f=0;
printf("Enter the size of array (max. 10) : ");
scanf("%d",&len);
printf("\nEnter %d elements of an array : \n",len);
for(i=0;i<len;i++)
scanf("%d",&A[i]);
printf("\nOriginal array is : \n");
for(i=0;i<len;i++)
printf("%d\t",A[i]);
printf("\n\nEnter the element to delete : ");
scanf("%d",&num);
for(i=0;i<len;i++)
{
if(num==A[i])
{
f=1;
for(;i<len-1;i++)
A[i]=A[i+1];
len--;
break;
}
}
if(f==0)
printf("\nNumber not found in array");
else
{
```

```
printf("\nNew Array after deletion : \n");
for(i=0;i<len;i++)
printf("%d\t",A[i]);
}
getch( );
}
```

OUTPUT :

```
Enter the size of array (max. 10) : 5
Enter 5 elements of an array :
2 4 6 8 10
Original array is :
2 4 6 8 10
Enter the element to delete : 6
New Array after deletion :
2 4 8 10
```

DOUBLE-DIMENSIONAL ARRAYS

A **double dimensional array** is an array in which each element is itself an array. For example, an array A[R][C] is an R by C table with R rows and C columns containing R * C elements.

The number of elements in a two-dimensional array can be determined by multiplying number of rows with number of columns. For example, the number of element in an array A[4][3] is calculated as 4 * 3 = 12.

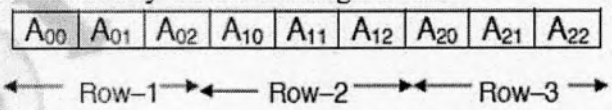
Implementation of Two-dimensional array in Memory

While storing the elements of a 2-D array in memory, these are allocated contiguous memory locations. A two-dimensional array can be implemented in a programming language in two ways:

1. Row-major implementation
2. Column-major implementation

Row-major implementation:

Row-major implementation is a linearization technique in which elements of array are read from the keyboard row-wise i.e. the complete first row is stored, then the complete second row is stored and so on. For example, an array **A [3] [3]** is stored in the memory as shown in Figure below :



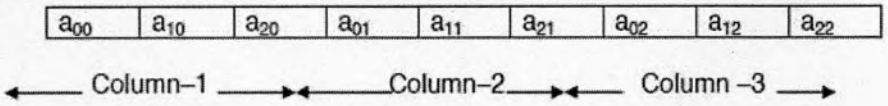
The storage can be clearly understood by arranging array as matrix as shown below:

$$a = \begin{vmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{vmatrix}$$

Row 1
Row 2
Row 3

Column-major implementation:

Column-major implementation is a linearization technique in which elements of array are read from the keyboard column-wise i.e. the complete first column is stored, then the complete second column is stored and so on. For example, an array a[3][3] is stored in the memory as shown in Figure below:



The storage can be clearly understood by arranging array as matrix as shown below :

$$a = \begin{vmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{vmatrix}$$

Column 1 Column 2 Column 3

Double Dimensional Array declaration:

Syntax:

datatype arrayvariablename[rowsize][col.size];

For e.g.

int A[4][3]; //where **int** is datatype, **A** is array variable_name, **4** is row size and **3** is columnsize.

Double Dimensional Array initialization:

Syntax:

datatype arrayname[rowsize][col. size]={{1st row elements},{2nd row elements},.....};

For e.g.

int A[4][3]={{1,2,3}, {4,5,6,}, {7,8,9}, {10,11,12}};

Array input from user :

Row major form

```
for(r=0;r<4;r++)
for(c=0;c<3;c++)
scanf("%d",&A[r][c]);
```

Column major form

```
for(c=0;c<3;c++)
for(r=0;r<4;r++)
scanf("%d",&A[r][c]);
```

Array output:

```
for(r=0;r<4;r++)
{
for(c=0;c<3;c++)
printf("%d\t",A[r][c]);
printf("\n");
}
```

PROGRAMS (DOUBLE DIMENSIONAL ARRAYS)

Program 1 : Double Dimensional array (Matrix) Initialization & Output

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
// Double Dim: Array declaration and initialization
int A[4][3]={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
int r,c;
// Double Dim: Array Output
printf("\nGiven 4 * 3 matrix is : \n");
for(r=0;r<4;r++)
{
for(c=0;c<3;c++)
printf("%d\t",A[r][c]);
printf("\n");
}
getch( );
}
```

OUTPUT :

```
Given 4 * 3 matrix is :
1 2 3
4 5 6
7 8 9
10 11 12
```

Program 2 : Double Dimensional array(Matrix) input & Output

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
// Matrix declaration
int A[4][3],r,c;
// Matrix input
printf("\nEnter elements of a 4 * 3 matrix : \n");
for(r=0;r<4;r++)
for(c=0;c<3;c++)
scanf("%d",&A[r][c]);
// Matrix output
printf("\nGiven 4 * 3 matrix is : \n");
for(r=0;r<4;r++)
{
for(c=0;c<3;c++)
printf("%d\t",A[r][c]);
printf("\n");
}
getch( );
}
```

OUTPUT :

```
Enter elements of a 4 * 3 matrix :
1 2 3
4 5 6
7 8 9
10 11 12
Given 4 * 3 matrix is :
1 2 3
4 5 6
7 8 9
10 11 12
```

Program 3: Addition of Two 3 * 3 Matrices

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
// Matrix declaration
int A[3][3],B[3][3],C[3][3],r,c;
// Matrix input
printf("\nEnter elements of first 3 * 3 matrix : \n");
for(r=0;r<3;r++)
```

```
for(c=0;c<3;c++)
scanf("%d",&A[r][c]);
printf("\nEnter elements of second 3 * 3 matrix : \n");
for(r=0;r<3;r++)
for(c=0;c<3;c++)
scanf("%d",&B[r][c]);
// Matrix addition
printf("\nAddition of first two matrices : \n");
for(r=0;r<3;r++)
{
for(c=0;c<3;c++)
{
C[r][c]=A[r][c]+B[r][c];
printf("%d\t",C[r][c]);
}
printf("\n");
}
getch( );
}
```

OUTPUT :

```
Enter elements of first 3 * 3 matrix :
1 2 3
4 5 6
7 8 9
Enter elements of second 3 * 3 matrix :
2 3 4
5 6 7
8 9 10
Addition of first two matrices:
3 5 7
9 11 13
15 17 19
```

Program 4 : WAP to display the transpose of a 3 * 3 matrix

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
int A[3][3],r,c;
printf("\nEnter elements of a 3 * 3 matrix :\n");
for(r=0;r<3;r++)
for(c=0;c<3;c++)
scanf("%d",&A[r][c]);
printf("\nOriginal matrix is :\n");
for(r=0;r<3;r++)
{
for(c=0;c<3;c++)
printf("%d\t",A[r][c]);
printf("\n");
}
printf("\nTranspose of given matrix is :\n");
for(r=0;r<3;r++)
```

```
{
for(c=0;c<3;c++)
printf("%d\t",A[c][r]);
printf("\n");
}
getch( );
}
```

OUTPUT :

```
Enter elements of a 3 * 3 matrix :
1 2 3
4 5 6
7 8 9
Original matrix is :
1 2 3
4 5 6
7 8 9
Transpose of given matrix is :
1 4 7
2 5 8
3 6 9
```

Program 5 : WAP to multiply two 3 * 3 matrices

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
int A[3][3],B[3][3],C[3][3],r,c,k;
printf("Enter elements of first 3 * 3 matrix : \n");
for(r=0;r<3;r++)
for(c=0;c<3;c++)
scanf("%d",&A[r][c]);
printf("\nEnter elements of second 3 * 3 matrix : \n");
for(r=0;r<3;r++)
for(c=0;c<3;c++)
scanf("%d",&B[r][c]);
printf("\nProduct of first two 3 * 3 matrices :\n");
for(r=0;r<3;r++)
{
for(c=0;c<3;c++)
{
C[r][c]=0;
for(k=0;k<3;k++)
C[r][c]=C[r][c]+(A[r][k]*B[k][c]);
printf("%d\t",C[r][c]);
}
printf("\n");
}
getch( );
}
```

OUTPUT :

```
Enter elements of first 3 * 3 matrix :
```

```
1 2 3
4 5 6
7 8 9
Enter elements of second 3 * 3 matrix :
1 2 3
4 5 6
7 8 9
Product of first two 3 * 3 matrices :
30 36 42
66 81 96
102 126 150
```

Some More Programs (functions):

```
//to count non zero elements in the array
int count(int a[10][10],int m,int n)
{
  int c=0,i,j;
  for(i=0;i<m;i++)
  {
    for(j=0;j<n;j++)
    {
      if(a[i][j]!=0)
      {
        c++;
      }
    }
  }
  return c;
}

//Find the sum of diagonal elements
//Hint: For any element a[i][j], if i==j
//i.e. row index==col index
void sum_diagonal(int a[10][10],int m)
{
  int i, sum=0;
  for(i=0;i<m;i++)
  {
      sum=sum+a[i][i];
  }
  printf("\nSum of diagonal elements is %d", sum);
}

//Find the sum of lower triangle
//Hint: For any element a[i][j], if i>j
//i.e. row index>col index
void sum_lowertri(int a[10][10],int m,int n)
{
  int i,j, sum=0;
  for(i=1;i<m;i++)
  {
    for(j=0;j<i;j++)
    {
      sum=sum+a[i][j];
```

```
    }
  }
  printf("\nSum of lower triangle elements is %d", sum);
}
```

```
//Find the sum of upper triangle
//Hint: For any element a[i][j], if i<j
//i.e. row index<col index
void sum_uppertri(int a[10][10],int m,int n)
{
  int i,j, sum=0;
  for(i=0;i<m-1;i++)
  {
    for(j=i+1;j<n;j++)
    {
      sum=sum+a[i][j];
    }
  }
  printf("\nSum of upper triangle elements is %d", sum);
}
```

Multidimensional Arrays:
The general syntax of a multidimensional array is:

datatype arrayname[size-1][size-2].....[size-n];

For example:
int A[5][2][3];
float B[2][5][3];
The simplest form of a multidimensional array is a two-dimensional array, which is also known as array of an array.

Finding the location (address) of particular element in 2D array:

Row Major Implementation

1. **If Array index starts from 0 (Follows Strictly in C/C++)**
   Consider an Array:
   int A[3][4];

A00	A01	A02	A03
A10	A11	A12	A13
A20	A21	A22	A23

Its arrangement in memory will be like:

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Element	A00	A01	A02	A03	A10	A11	A12	A13	A20	A21	A22	A23
Address (Assume)	**200**	**202**	**204**	**206**	**208**	**210**	**212**	**214**	**216**	**218**	**220**	**222**
	Row-0				**Row-1**				**Row-2**			

Address of A[1][2]:
= Base Address + [(No. of rows to skip to reach A[1][2] * No of elements in each row) + (No. of elements to move in current row to reach A[1][2])] * Size of each element
= 200 + [1 * 4 + 2] * 2

= 200 + 6 * 2
= 212

Address of A[2][3]:
= Base Address + [(No. of rows to skip to reach A[2][3] * No of elements in each row) + (No. of elements to move in current row to reach A[2][3])] * Size of each element
= 200 + [2 * 4 + 3] * 2
= 200 + 11 * 2
= 222

So, in general we can write this in formula as:

$$LOC(A[I][J]) = Base(A) + [I * N + J] * W$$

Here

LOC (A [I][J])	:	is the location of the element in the I^{th} row and J^{th} column.
Base (A)	:	is the base address of the array A.
W	:	is the number of bytes required to store single element of the array A.
N	:	is the total number of columns in the array.
I	:	is the row number of the element.
J	:	is the column number of the element.

2. If Array index starts from 1 (Older languages)

$$LOC(A[I][J]) = Base(A) + [(I-1) * N + (J-1)] * W$$

Here

LOC (A [I][J])	:	is the location of the element in the I^{th} row and J^{th} column.
Base (A)	:	is the base address of the array A.
W	:	is the number of bytes required to store single element of the array A.
N	:	is the total number of columns in the array.
I	:	is the row number of the element.
J	:	is the column number of the element.

E.g. A 3 x 4 integer array A is as below:

Subscript	Elements	Address
(1,1)	10	1000
(1,2)	60	1002
(1,3)	30	1004
(1,4)	55	1006
(2,1)	20	1008
(2,2)	90	1010
(2,3)	80	1012
(2,4)	65	1014
(3,1)	50	1016
(3,2)	40	1018
(3,3)	75	1020
(3,4)	79	1022

Suppose we have to find the location of A [3, 2].
The required values are:

Base (A)	:	1000
w	:	2
		(int takes 2 bytes in memory)
N	:	4
I	:	3
J	:	2

Now put these values in the given formula as below:
LOC (A [3, 2]) = 1000 + 2 [(3-1) * 4 + (2-1)]
= 1000 + 2 [(2) * 4 + 1]
= 1000 + 2 [8 + 1]
= 1000 + 2 [9]
= 1000 + 18
= 1018

Column Major Implementation

1. **If Array index starts from 0 (Follows Strictly in C/C++)**
   Consider an Array:
   int A[3][4];

A00	A01	A02	A03
A10	A11	A12	A13
A20	A21	A22	A23

Its arrangement in memory will be like:

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Element	A00	A01	A20	A01	A11	A21	A02	A12	A22	A03	A13	A23
Address (Assume)	**200**	**202**	**204**	**206**	**208**	**210**	**212**	**214**	**216**	**218**	**220**	**222**
	Col-0			**Col-1**			**Col-2**			**Col-3**		

Address of A[1][2]:
= Base Address + [(No. of cols to skip to reach A[1][2] * No of elements in each col) + (No. of elements to move in current col to reach A[1][2])] * Size of each element
= 200 +[2 * 3 + 1] * 2
= 200 + 7 * 2
= 214

Address of A[2][3]:
= Base Address + [(No. of cols to skip to reach A[2][3] * No of elements in each col) + (No. of elements to move in current col to reach A[2][3])] * Size of each element
= 200 +[3 * 3 + 2] * 2
= 200 + 11 * 2
= 222

So, In general we can write this in formula as:

$$\mathbf{LOC(A[I][J]) = Base(A) + [J*M + I] * W}$$

Here

LOC (A [I][J])	:	is the location of the element in the I^{th} row and J^{th} column.
Base (A)	:	is the base address of the array A.
W	:	is the number of bytes required to store single element of the array A.
M	:	is the total number of rows in the array.
I	:	is the row number of the element.
J	:	is the column number of the element.

2. **If Array index starts from 1 (Older languages)**

$$\mathbf{LOC(A[I][J]) = Base(A) + [(J-1)*M + (I-1)] * W}$$

Here

LOC (A [I][J])	:	is the location of the element in the I^{th} row and J^{th} column.
Base (A)	:	is the base address of the array A.
W	:	is the number of bytes required to store single element of the array A.
M	:	is the total number of rows in the array.
I	:	is the row number of the element.
J	:	is the column number of the element.

E.g.

A 3 x 4 integer array A is as below:

Subscript	Elements	Address
(1,1)	10	1000
(2,1)	20	1002
(3,1)	50	1004
(1,2)	60	1006
(2,2)	90	1008
(3,2)	40	1010
(1,3)	30	1012
(2,3)	80	1014
(3,3)	75	1016
(1,4)	55	1018
(2,4)	65	1020
(3,4)	79	1022

Suppose we have to find the location of A [3, 2].
The required values are:

Base (A) : 1000
w : 2
(int takes 2 bytes in memory)
M : 3
J : 3
K : 2

Now put these values in the given formula as below:

LOC (A [3, 2]) = 1000 + 2 [(2-1) * 3 + (3-1)]
= 1000 + 2 [(1) * 3 + 2]
= 1000 + 2 [3 + 2]
= 1000 + 2 [5]
= 1000 + 10
= 1010

Special Case

If Array index starts from Any Other Number (Older languages) or,

Important: Usually number of rows and columns of a matrix are given (like A[20][30] or A[40][60]) but if it is given as A[Lr- - - - - Ur, Lc- - - - - Uc]. In this case number of rows and columns are calculated using the following methods:

Number of rows (M) will be calculated as = (Ur – Lr) + 1
Number of columns (N) will be calculated as = (Uc – Lc) + 1

And rest of the process will remain same as per requirement (Row Major Wise or Column Major Wise).

Address of an element of any array say "A[I][J]" is calculated in two forms as given:

Row Major System:

The address of a location in Row Major System is calculated using the following formula:

Address of A [I][J] = B + [N * (I – Lr) + (J – Lc)] * W

Column Major System:

The address of a location in Column Major System is calculated using the following formula:

Address of A [I][J] = B + [M * (J – Lc) + (I – Lr)] * W

Where,
B = Base address
I = Row subscript of element whose address is to be found
J = Column subscript of element whose address is to be found
W = Storage Size of one element stored in the array (in byte)
Lr = Lower limit of row/start row index of matrix, if not given assume 0 (zero)
Lc = Lower limit of column/start column index of matrix, if not given assume 0 (zero)
M = Number of row of the given matrix
N = Number of column of the given matrix

Examples:

Q 1. An array X [-15.....10, 15.....40] requires one byte of storage. If beginning location is 1500 determine the location of X [15][20].

Solution:

As you see here the number of rows and columns are not given in the question. So they are calculated as:

Number or rows say M = (Ur – Lr) + 1 = [10 – (- 15)] +1 = 26
Number or columns say N = (Uc – Lc) + 1 = [40 – 15)] +1 = 26

(i) Column Major Wise Calculation of above equation
The given values are: B = 1500, W = 1 byte, I = 15, J = 20, Lr = -15, Lc = 15, M = 26
Address of A [I][J] = B + W * [(I – Lr) + M * (J – Lc)]
= 1500 + 1 * [(15 – (-15)) + 26 * (20 – 15)] = 1500 + 1 * [30 + 26 * 5] = 1500 + 1 * [160] = 1660 [Ans]

(ii) Row Major Wise Calculation of above equation
The given values are: B = 1500, W = 1 byte, I = 15, J = 20, Lr = -15, Lc = 15, N = 26
Address of A [I][J] = B + W * [N * (I – Lr) + (J – Lc)]
= 1500 + 1* [26 * (15 – (-15))) + (20 – 15)] = 1500 + 1 * [26 * 30 + 5] = 1500 + 1 * [780 + 5] = 1500 + 785
= 2285 [Ans]

Q 2. An array VAL[1…15][1…10] is stored in the memory with each element requiring 4 bytes of storage. If the base address of the array VAL is 1500, determine the location of VAL[12][9] when the array VAL is stored (i) Row wise (ii) Column wise.
Solution:
Given Data:
VAL[1…15][1…10]
Word Length (W) = 4 Bytes
Base Address of VAL(B) = 1500
VAL[12][9] = ?
M= Total no of Rows= (15-1)+1=15
N=Total no of Cols= (10-1)+1=10
Lr = Least Row=1
Lc = Least Column=1

(i) Row Major:
Address of an element (I,J) in row major = B + W (C (I-Lr) + (J – Lc))
VAL [12][9] = 1500 + 4 (10 * (12-1) + (9-1))
= 1500 + 4 (10 * 11+8)
= 1500 + 4 (118)
= 1500 + 472
= 1972.

(i) Column Major:
Address of an element (I,J) in column major = B + W ((I-Lr) + R(J – Lc))
VAL [12][9] = 1500 + 4 ((12-1) +15 * (9-1))
= 1500 + 4 (11 + 15 * 8)
= 1500 + 4 (11+ 120)
= 1500 + 4 * 131
= 1500 + 524
= 2024.

Sparse Matrix:

In computer programming, a matrix can be defined with a 2-dimensional array. Any array with 'm' rows and 'n' columns represents a mxn matrix. There may be a situation in which a matrix contains more number of ZERO values than NON-ZERO values. Such matrix is known as sparse matrix.

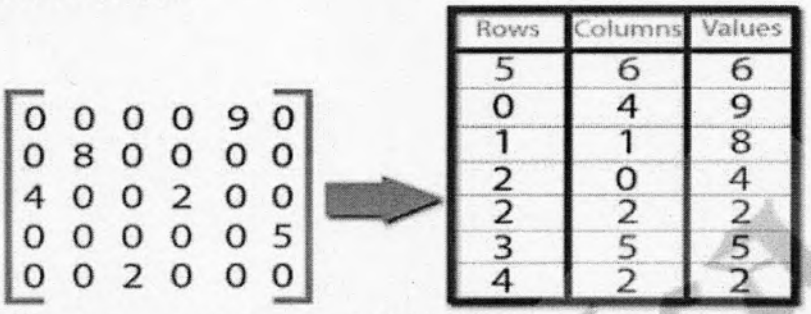
"Sparse matrix is a matrix which contains very few non-zero elements."

When a sparse matrix is represented with 2-dimensional array, we waste lot of space to represent that matrix. For example, consider a matrix of size 100 X 100 containing only 10 non-zero elements. In this matrix, only 10 spaces are filled with non-zero values and remaining spaces of matrix are filled with zero. That means,

totally we allocate 100 X 100 X 2 = 20000 bytes of space to store this integer matrix. And to access these 10 non-zero elements we have to make scanning for 10000 times.

Sparse Matrix Representations:

In this representation, we consider only non-zero values along with their row and column index values. In this representation, the 0^{th} row stores total rows, total columns and total non-zero values in the matrix. **For example,** consider a matrix of size 5 X 6 containing 6 number of non-zero values. This matrix can be represented as shown in the image...



Rows	Columns	Values
5	6	6
0	4	9
1	1	8
2	0	4
2	2	2
3	5	5
4	2	2

In above example matrix, there are only 6 non-zero elements (those are 9, 8, 4, 2, 5 & 2) and matrix size is: 5 X 6. We represent this matrix as shown in the above image.

- Here the first row in the right-side table is filled with values 5, 6 & 6 which indicates that it is a sparse matrix with 5 rows, 6 columns & 6 non-zero values.
- Second row is filled with 0, 4, & 9 which indicates the value in the matrix at 0th row, 4th column is 9.
- In the same way the remaining non-zero values also follows the similar pattern.

WAP to:

1. **Create a sparse matrix from 2D Array (Matrix).**
2. **Print the sparse matrix elements.**
3. **Check whether the matrix is sparse or not.**

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 20

//Function to take input from user
void read(int a[10][10], int m, int n)
{
   int i,j;
   for(i=0;i<m;i++)
   {
      printf("\n");
      for(j=0;j<n;j++)
      {
         scanf("%d",&a[i][j]);
      }
   }
}

//Function to create sparse matrix
int create_sparse(int a[10][10], int row, int column, int b[MAX][3])
{
   int i, j, k;
   k = 1;
   b[0][0] = row;
```

```
b[0][1] = column;
for (i = 0; i < row; i++)
{
  for (j = 0; j < column; j++)
  {
    if (a[i][j] != 0)
    {
      b[k][0] = i;
      b[k][1] = j;
      b[k][2] = a[i][j];
      k++;
    }
  }
  b[0][2] = k - 1;
}
return k; }
```

//Function to print sparse matrix

```
void print_sparse(int b[MAX][3],int k)
{
  int i,j;
  for(i=0;i<k;i++)
  {
      printf("\n%d \t%d \t%d",b[i][0], b[i][1], b[i][2]);
  }
}
```

//Function to check whether matrix is sparse or not

```
void chk_sparse(int a[10][10], int m, int n)
{
  int i, j, counter=0;
  for(i=0;i<m;i++)
  {
    for(j=0;j<m;j++)
    {
      if(a[i][j]==0)
      {
        counter++;
      }
    }
  }

  if(counter>((m*n)/2))
  {
    printf("\nThe given matrix is sparse matrix");
  }
  else
  {
    printf("\nThe given matrix is not a sparse matrix");
  }
}
```

//Main Function

```
int main()
```

```
{
  int a[10][10], i, j, m, n, sp[MAX][3];

  printf("Enter the size of matrix a:");
  scanf("%d%d",&m, &n);

  printf("\nEnter the elements of Matrix A:");
  read(a,m,n);

  int count=create_sparse(a,m,n,sp);
  print_sparse(sp,count);
  chk_sparse(a,m,n);
  return 0;
}
```

What is Search?

Search is a process of finding a value in a list of values. In other words, searching is the process of locating given value position in a list of values.

Linear Search Algorithm (Sequential Search Algorithm)

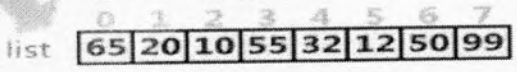
Linear search algorithm finds given element in a list of elements with **O(n)** time complexity where **n** is total number of elements in the list.

Linear search is implemented using following steps...

- **Step 1:** Read the search element from the user
- **Step 2:** Compare, the search element with the first element in the list.
- **Step 3:** If both are matching, then display "Given element found!!!" and terminate the function
- **Step 4:** If both are not matching, then compare search element with the next element in the list.
- **Step 5:** Repeat steps 3 and 4 until the search element is compared with the last element in the list.
- **Step 6:** If the last element in the list is also doesn't match, then display "Element not found!!!" and terminate the function.

Example

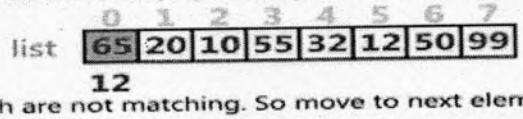
Consider the following list of element and search element...



search element 12

Step 1:

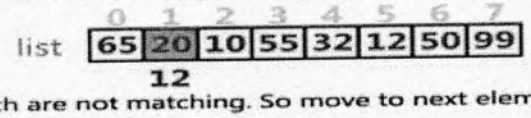
search element (12) is compared with first element (65)



Both are not matching. So move to next element

Step 2:

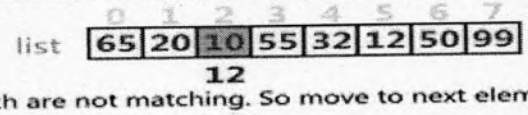
search element (12) is compared with next element (20)



Both are not matching. So move to next element

Step 3:

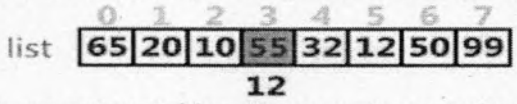
search element (12) is compared with next element (10)



Both are not matching. So move to next element

Step 4:

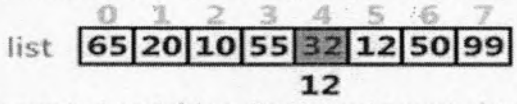
search element (12) is compared with next element (55)



Both are not matching. So move to next element

Step 5:

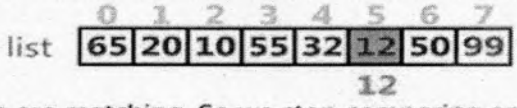
search element (12) is compared with next element (32)



Both are not matching. So move to next element

Step 6:

search element (12) is compared with next element (12)



Both are matching. So we stop comparing and display element found at index 5.

WAP to implement Linear Search algorithm in "C"

```
#include<stdio.h>
int main()
{
   int a[10]={1,2,4,6,7,8,9,10,11,13};
   int item,i,flag,c=0;;

   printf("Enter the item to be searched:");
   scanf("%d",&item);

   for(i=0;i<9;i++)
   {
      if(item==a[i])
      {
         flag=1;
         break;
      }
   }
   if(flag==1)
   {
      printf("\nItem found at index %d",i);
   }
   else
   {
      printf("Item not found");
   }
   return 0;
}
```

Binary Search Algorithm

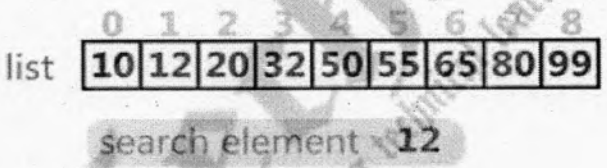
Binary search algorithm finds given element in a list of elements with **O(log n)** time complexity where **n** is total number of elements in the list. The binary search algorithm can be used with only sorted list of element. The binary search cannot be used for list of elements, which are in random order.

Binary search is implemented using following steps...

- **Step 1:** Read the search element from the user
- **Step 2:** Find the middle element in the sorted list
- **Step 3:** Compare, the search element with the middle element in the sorted list.
- **Step 4:** If both are matching, then display "Given element found!!!" and terminate the function
- **Step 5:** If both are not matching, then check whether the search element is smaller or larger than middle element.
- **Step 6:** If the search element is smaller than middle element, then repeat steps 2, 3, 4 and 5 for the left sublist of the middle element.
- **Step 7:** If the search element is larger than middle element, then repeat steps 2, 3, 4 and 5 for the right sublist of the middle element.
- **Step 8:** Repeat the same process until we find the search element in the list or until sublist contains only one element.
- **Step 9:** If that element also doesn't match with the search element, then display "Element not found in the list!!!" and terminate the function.

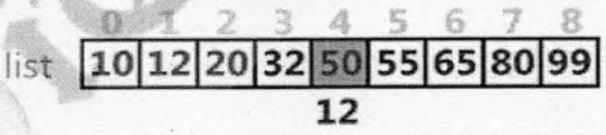
Example

Consider the following list of element and search element...

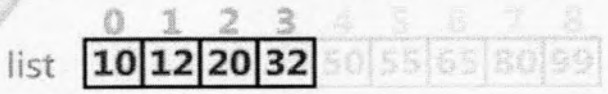


Step 1:

search element (12) is compared with middle element (50)

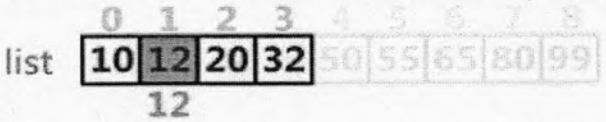


Both are not matching. And 12 is smaller than 50. So we search only in the left sublist (i.e. 10, 12, 20 & 32).



Step 2:

search element (12) is compared with middle element (12)



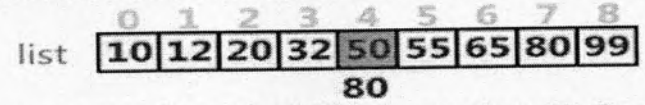
Both are matching. So the result is "Element found at index 1"

Now consider, we want to search another element in the same list of items.

search element **80**

Step 1:

search element (80) is compared with middle element (50)



Both are not matching. And 80 is larger than 50. So we search only in the right sublist (i.e. 55, 65, 80 & 99).

Step 2:

search element (80) is compared with middle element (65)

Both are not matching. And 80 is larger than 65. So we search only in the right sublist (i.e. 80 & 99).

Step 3:

search element (80) is compared with middle element (80)

Both are not matching. So the result is "Element found at index 7"

WAP to implement BINARY SEARCH algorithm in "C"

```
#include<stdio.h>
void main()
{
   int a[10]={1,2,4,6,7,8,9,10,11,13};
   int i,item, flag;
   int low=0,high=9,mid;

   printf("Enter the item to be searched:");
   scanf("%d",&item);

  for (i=0;i<10;i++)
   {
      mid=(low+high)/2;
      if(item==a[mid])
      {
         flag=1;
         break;
      }
      else if(item<a[mid])
      {
         high=mid-1;
      }
```

```
    else
    {
      low=mid+1;
    }
  }
  if(flag==1)
  {
    printf("Item found at index %d",mid);
  }
  else
  {
    printf("Item not found");
  }

  getch();
}
```

UNIT-2

QUEUES

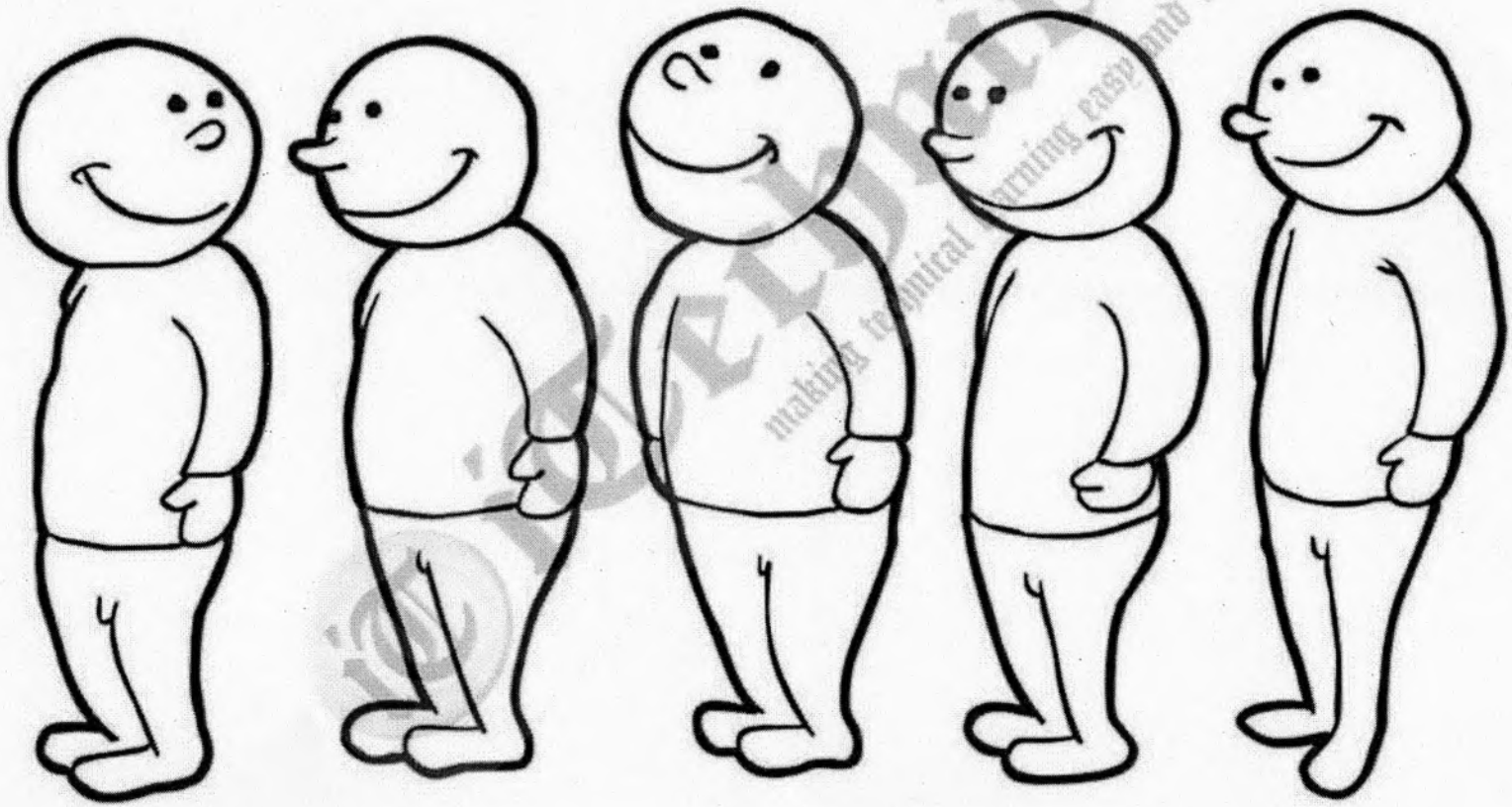
What is Queue?

- Queue is a linear data structure in which the insertion and deletion operations are performed at two different ends.
- In a queue data structure, the insertion operation is performed at a position which is known as **'rear'** and the deletion operation is performed at a position which is known as **'front'**.
- The first element that gets added into the queue is the first one to get removed from the list. Hence, Queue is also referred to as ***First-In-First-Out (FIFO)*** list.
- Consider a railway reservation booth, at which we have to get into the reservation queue.

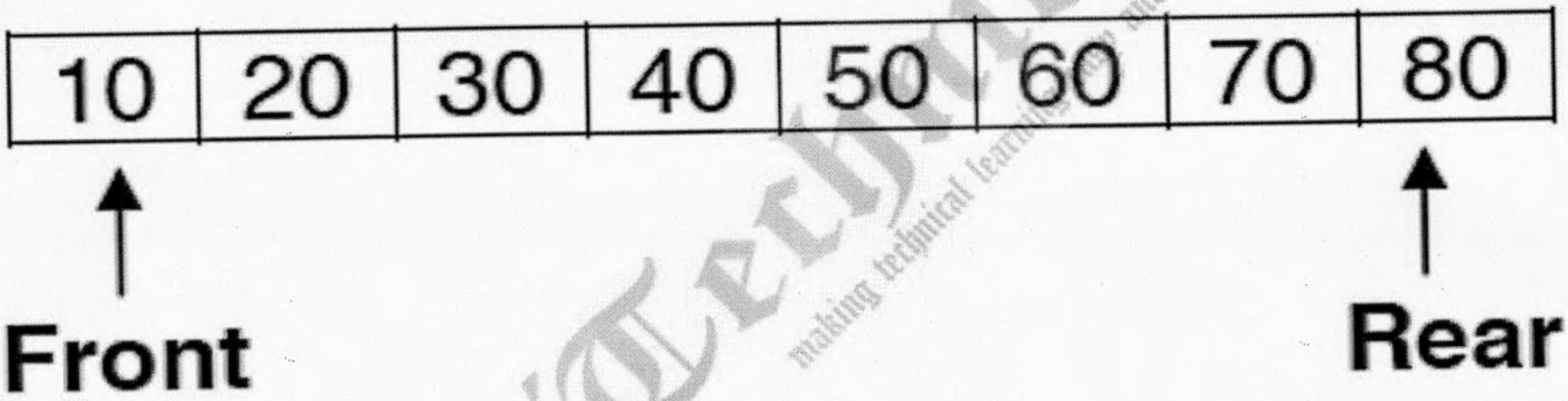
Queue as an ADT

- Queue is a list of items with following operation defined on it, but how they will be implemented is not known.
- **Enqueue(o):** Insert o at rear of queue
- **Dequeue():** Remove object at front; error if empty
- **Size():** Return number of objects in queue
- **isEmpty():** Return a boolean indicating queue empty
- **first():** Return object at front without removing; error if empty.

Queue



Queue (Pictorial Representation)



In fig (1), **10** is the first element and **80** is the last element added to the Queue.
Similarly, **10** would be the first element to get removed and **80** would be the last element to get removed.

Insertion in queue

- Figures 2(a) to 2(d) shows queue graphically during insertion operation :

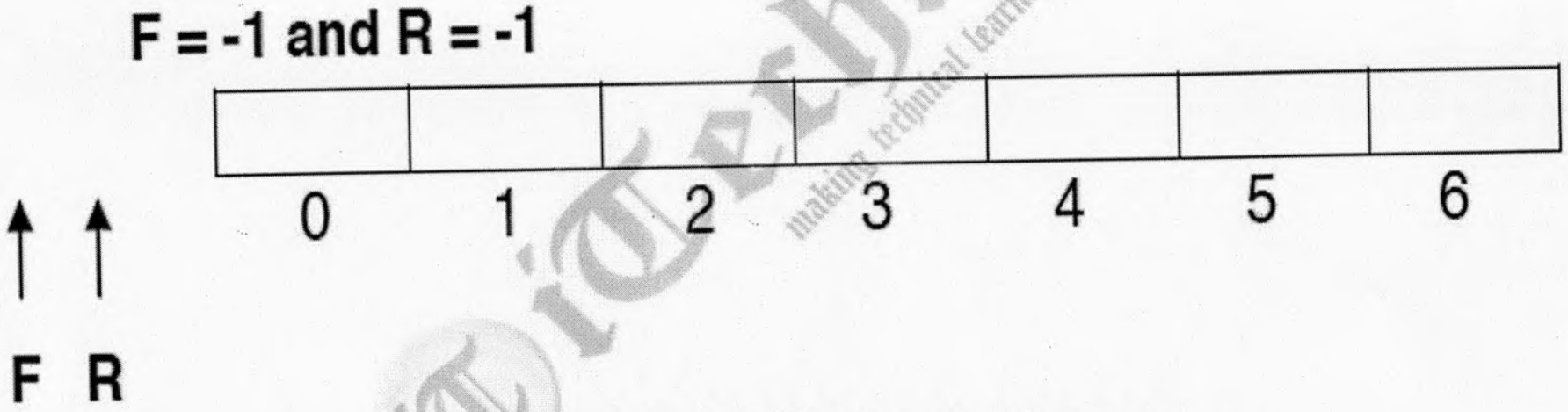


Fig. 2(a) Empty Queue

Insertion in queue contd..

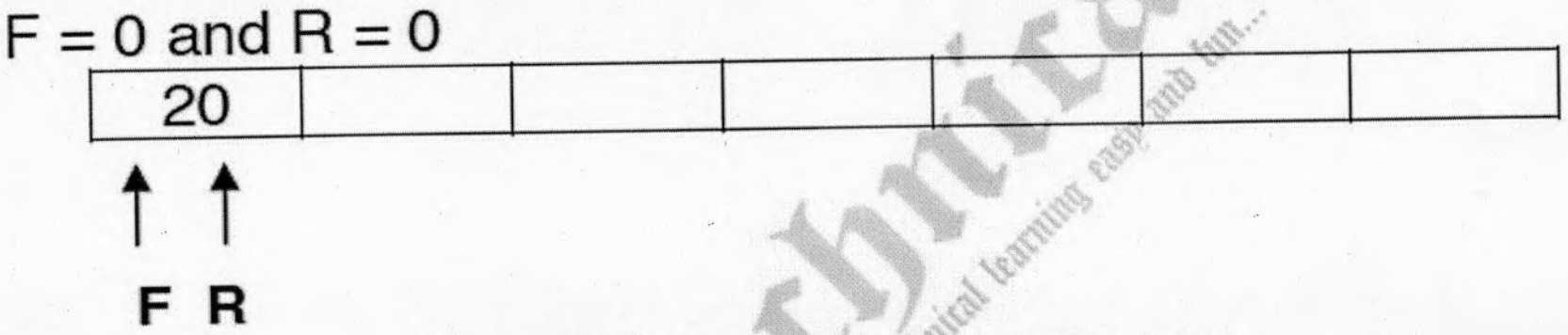


Fig. 2(b) One Element Queue

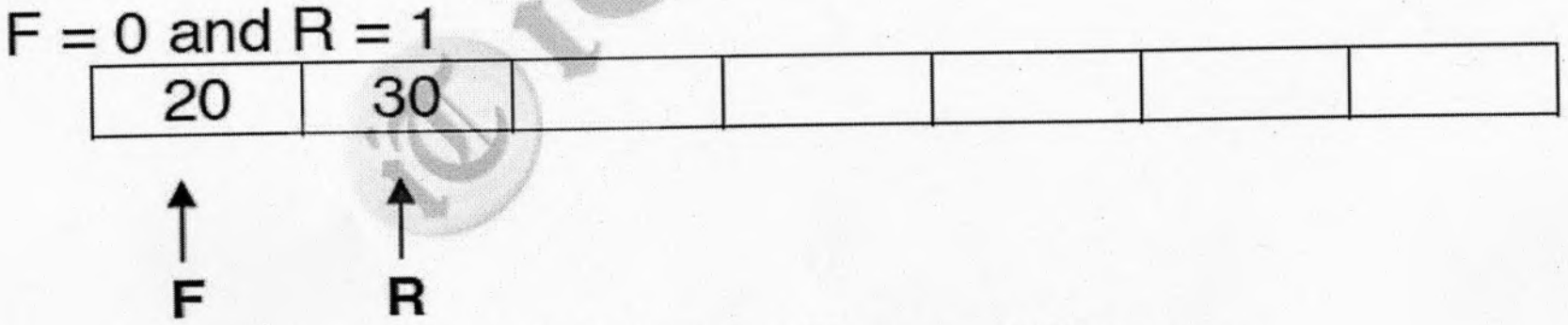


Fig. 2(c) Two Element Queue

Insertion in queue contd..

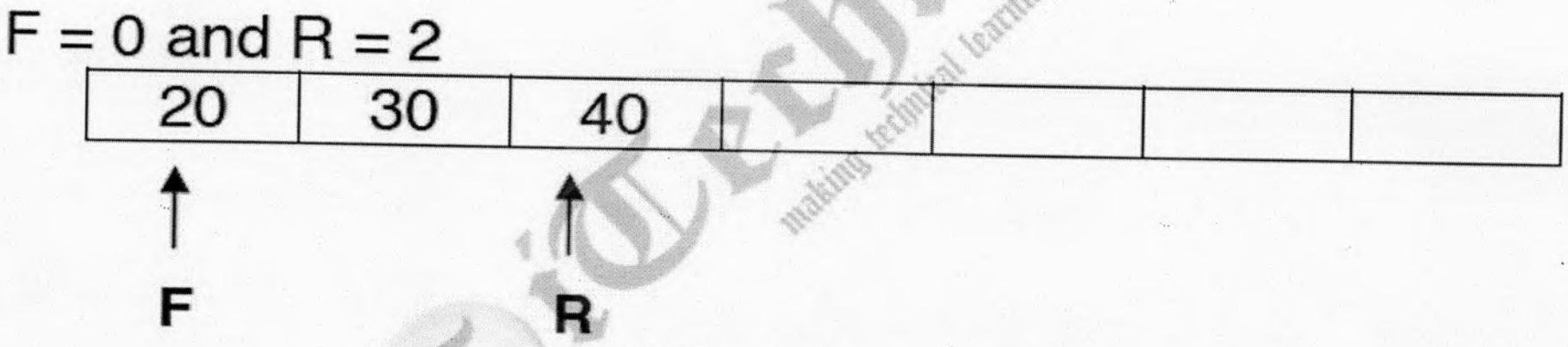


Fig. 2(d) Three Element Queue

Insertion in queue contd…

- It is clear from the above figures that whenever we insert an element in the queue, the value of Rear is incremented by one i.e.

Rear = Rear + 1

- Also, **during the insertion of the first element** in the queue we always incremented the Front by one i.e.

Front = Front + 1

- Afterwards the Front will not be changed during the entire operation.

Deletion from queue

- The following figures show Queue graphically during deletion operation :

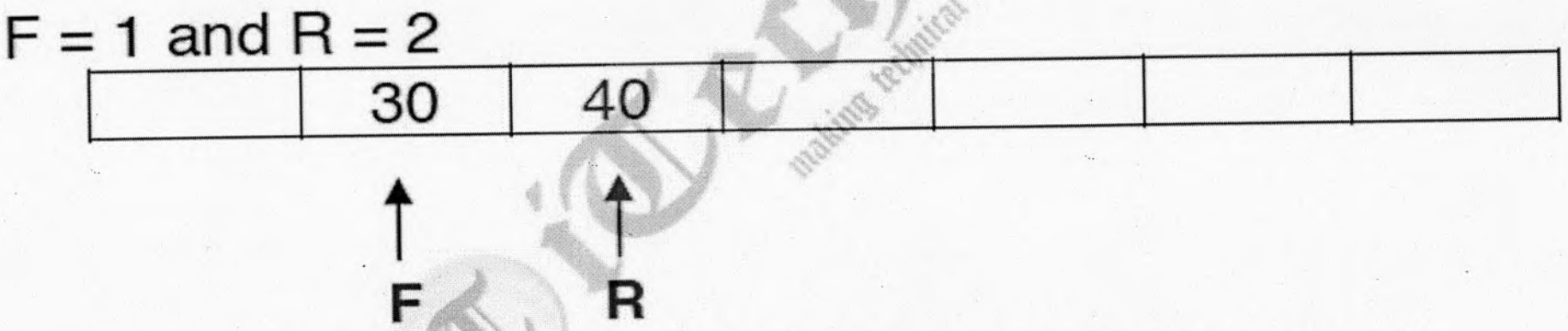


Fig. 2(e) One Element (20) Deleted from Front

Deletion from queue contd…

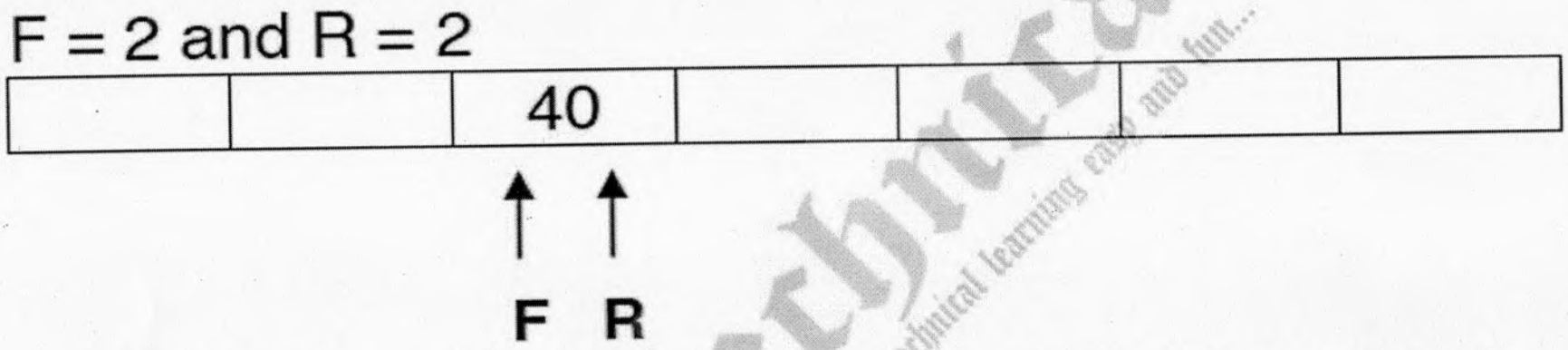


Fig. 2(f) Second Element (30) Deleted from Front

- This is clear from Fig. 2(e) and 2(f), that whenever an element is removed from the queue, the value of Front is incremented by one i.e.,

Front = Front + 1

Insertion after deletion

- Now, if we insert any element in the queue, the queue will look like :

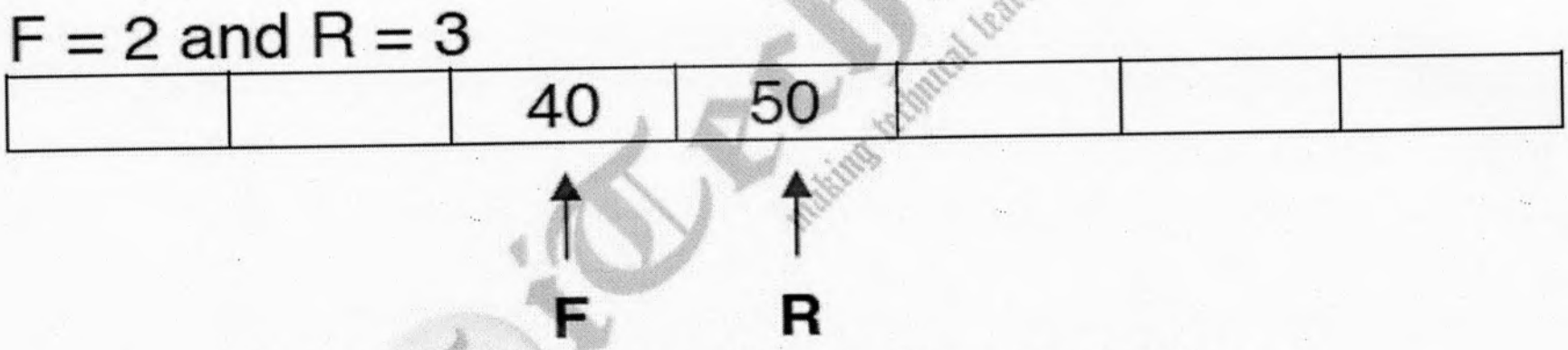


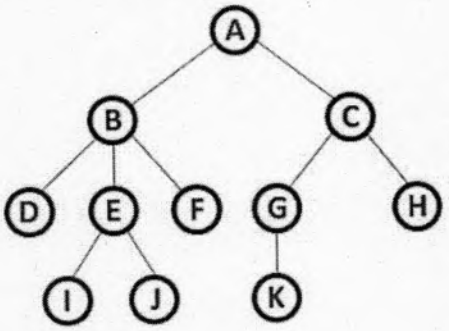
Fig. 2(g) Insertion after Deletion

UNIT-3

Unit-03
Trees

"A Tree is a non-linear data structure in which items are arranged in a sorted sequence. It is used to represent hierarchical relationship existing amongst several data items."

The graph theoretic definition of tree is: It is a finite set of one or more data items (nodes) such that
1. There is a special data item called the root of the tree.
2. And its remaining data items are partitioned into number of mutually exclusive (i.e. disjoint) subsets, each of which is itself a tree. And they are called **subtrees.**

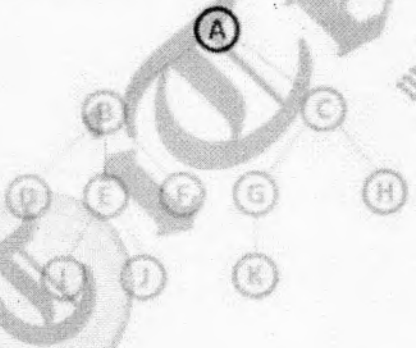


TREE with 11 nodes and 10 edges

- In any tree with 'N' nodes there will be maximum of 'N-1' edges
- In a tree every individual element is called as 'NODE'

Terminology
In a tree data structure, we use the following terminology...

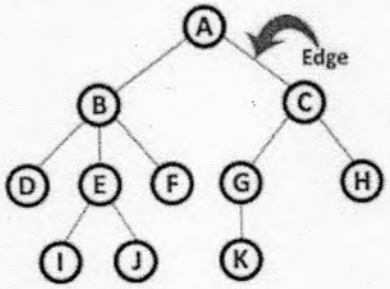
1. Root
In a tree data structure, the first node is called as **Root Node**. Every tree must have root node. We can say that root node is the origin of tree data structure. In any tree, there must be only one root node. We never have multiple root nodes in a tree.



Here 'A' is the 'root' node

- In any tree the first node is called as ROOT node

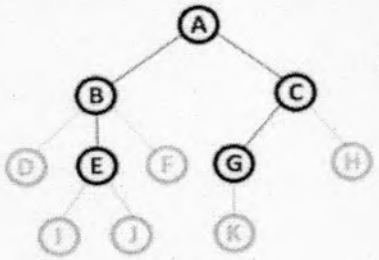
2. Edge
In a tree data structure, the connecting link between any two nodes is called as **EDGE**. In a tree with **'N'** number of nodes there will be a maximum of **'N-1'** number of edges.



- In any tree, 'Edge' is a connecting link between two nodes.

3. Parent

In a tree data structure, the node which is predecessor of any node is called as **PARENT NODE**. In simple words, the node which has branch from it to any other node is called as parent node. Parent node can also be defined as **"The node which has child / children"**.

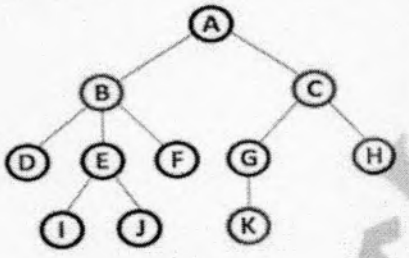


Here A, B, C, E & G are Parent nodes

- In any tree the node which has child / children is called 'Parent'
- A node which is predecessor of any other node is called 'Parent'

4. Child

In a tree data structure, the node which is descendant of any node is called as **CHILD Node**. In simple words, the node which has a link from its parent node is called as child node. In a tree, any parent node can have any number of child nodes. In a tree, all the nodes except root are child nodes.

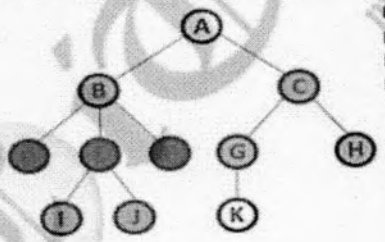


Here B & C are Children of A
Here G & H are Children of C
Here K is Child of G

- descendant of any node is called as CHILD Node

5. Siblings

In a tree data structure, nodes which belong to same Parent are called as **SIBLINGS**. In simple words, the nodes with same parent are called as Sibling nodes.

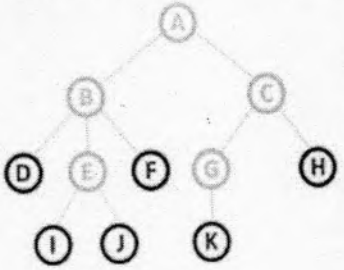


Here B & C are Siblings
Here D E & F are Siblings
Here G & H are Siblings
Here I & J are Siblings

- In any tree the nodes which has same Parent are called 'Siblings'
- The children of a Parent are called 'Siblings'

6. Leaf

In a tree data structure, the node which does not have a child is called as **LEAF Node**. In simple words, a leaf is a node with no child. In a tree data structure, the leaf nodes are also called as **External Nodes**. External node is also a node with no child. In a tree, leaf node is also called as **'Terminal'** node.

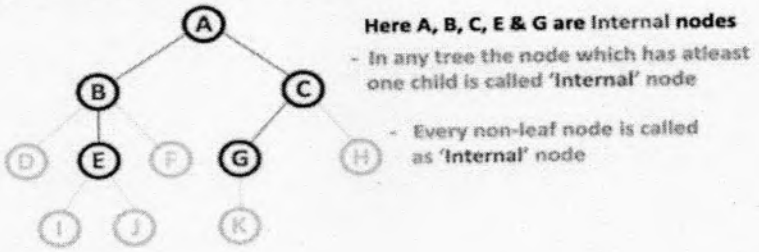


Here D, I, J, F, K & H are Leaf nodes

- In any tree the node which does not have children is called 'Leaf'
- A node without successors is called a 'leaf' node

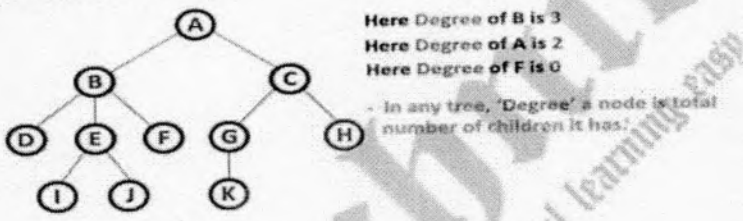
7. Internal Nodes

In a tree data structure, the node which has at least one child is called as **INTERNAL Node**. In simple words, an internal node is a node with at least one child. In a tree data structure, nodes other than leaf nodes are called as **Internal Nodes**. **The root node is also said to be Internal Node** if the tree has more than one node. Internal nodes are also called as **'Non-Terminal'** nodes.



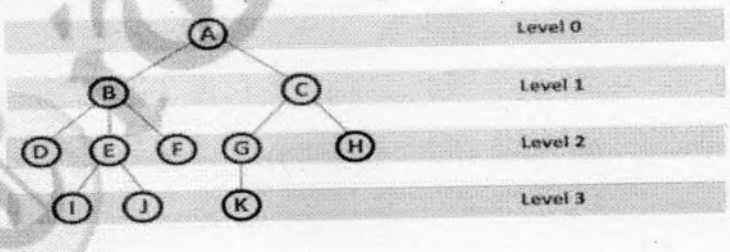
8. Degree

In a tree data structure, the total number of children of a node is called as **DEGREE** of that Node. In simple words, the Degree of a node is total number of children it has. The highest degree of a node among all the nodes in a tree is called as **'Degree of Tree'**



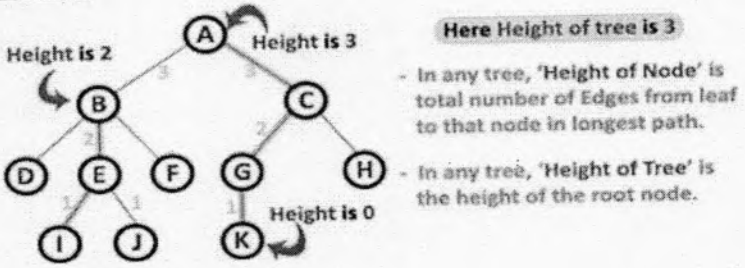
9. Level

In a tree data structure, the root node is said to be at Level 0 and the children of root node are at Level 1 and the children of the nodes which are at Level 1 will be at Level 2 and so on... In simple words, in a tree each step from top to bottom is called as a Level and the Level count starts with '0' and incremented by one at each level (Step).



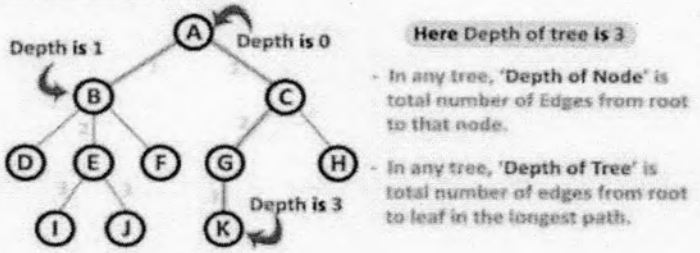
10. Height

In a tree data structure, the total number of edges from leaf node to a particular node in the longest path is called as **HEIGHT** of that Node. In a tree, height of the root node is said to be **height of the tree**. In a tree, **height of all leaf nodes is '0'.**



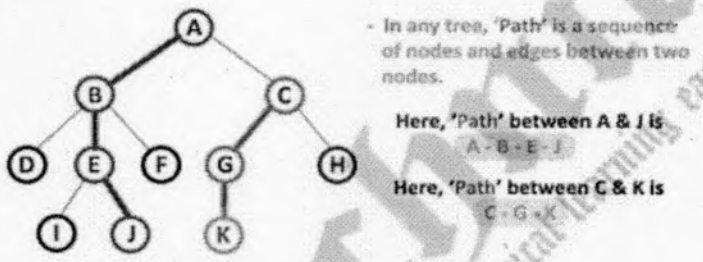
11. Depth

In a tree data structure, the total number of edges from root node to a particular node is called as **DEPTH** of that Node. In a tree, the total number of edges from root node to a leaf node in the longest path is said to be **Depth of the tree**. In simple words, the highest depth of any leaf node in a tree is said to be depth of that tree. In a tree, **depth of the root node is '0'.**



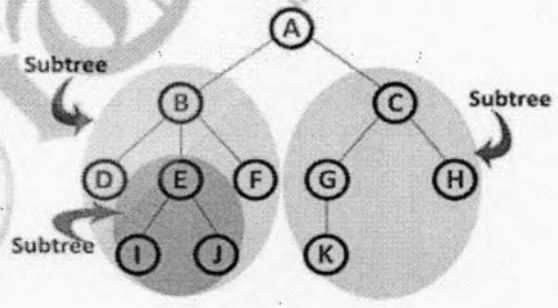
12. Path

In a tree data structure, the sequence of Nodes and Edges from one node to another node is called as **PATH** between that two Nodes. **Length of a Path** is total number of nodes in that path. In below example **the path A - B - E - J has length 4.**



13. Sub Tree

In a tree data structure, each child from a node forms a subtree recursively. Every child node will form a subtree on its parent node.



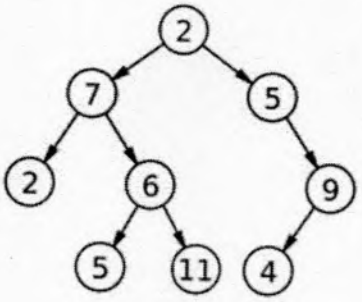
Binary Tree:

In a normal tree, every node can have any number of children. Binary tree is a special type of tree data structure in which every node can have a **maximum of 2 children**. One is known as left child and the other is known as right child. In a binary tree, every node can have either 0 children or 1 child or 2 children but not more than 2 children.

A binary tree consists of a finite set of elements that can be partitioned into three distinct sub-sets called the **root,** the **left sub-tree** and the **right sub-tree.** If there are no elements in the binary tree it is called an **empty binary tree.**

- **Node:** Each element present in a binary tree is called a **node** of that tree.
- **Root:** The element that represents the base node of the tree is called the **root** of the tree.

- **The left and right sub-tree:** Apart from the root, the other two sub-sets of a binary tree are binary trees. They are called the **left** and **right sub-trees** of the original tree. Any of these sub-sets can be empty.



Some Important Additional Terms:

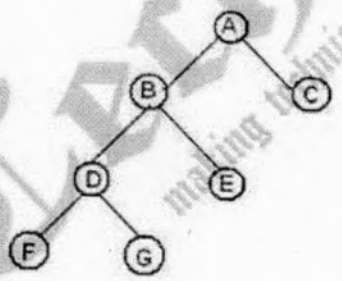
1. **Ancestor and descendant:** A node **A** is said to be an **ancestor** of node **B**, if **A** is either the father of **B** or the father of some **ancestor** of **B.**

2. **Climbing and descending:**
   When we are traversing the tree from the leaf node to the root node the operation is **climbing.** Similarly, traversing the tree from the root to the leaves is called **descending** the tree.

There are different types of binary trees and they are...

1. Strictly Binary Tree / Full Binary Tree /

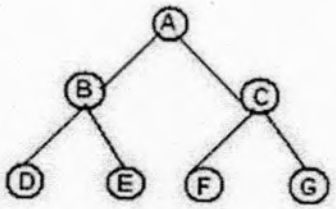
A binary tree in which every node has either two or zero number (i.e. leaf) of children is called Strictly Binary Tree



Strictly binary tree is also called as **Proper Binary Tree.**

2. Complete binary tree:

A complete binary tree in which all levels except possible the last level is completely filled and all nodes are as left as possible



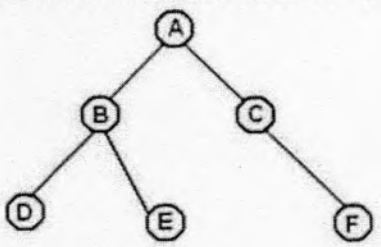
3. Extended Binary tree:

A binary tree can be converted to an extended binary tree by adding new nodes to its leaf nodes and to the nodes that have only one child. These new nodes are added in such a way that all the nodes in the resultant tree have either 0 or 2 children. The extended tree is also known as a 2-tree. The nodes of the original tree are called internal nodes and the new nodes that are added to binary tree, to make it an extended binary tree are called external nodes.

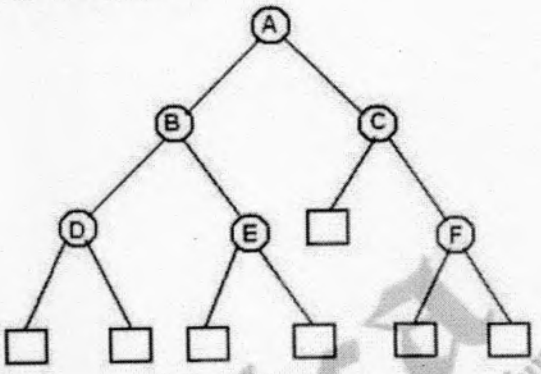
Fig. shows how extended binary tree can be obtained by adding new nodes to original tree. In Fig., all the nodes with circular shape are internal nodes and all the nodes with square shape are external nodes.

A few important points about Extended Binary tree are :

(a) If a tree has n nodes then the number of branches it has is (n-1).
(b) Except the root node every node in a tree has exactly one parent.
(c) Any two nodes of a tree are connected by only one single path.
(d) For a binary tree of height h the maximum number of nodes can be $2h+1 - 1$.
(e) Any binary tree with n internal nodes has (n + 1) external nodes.



(a) Binary Tree T

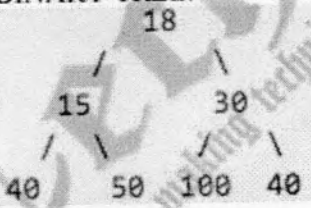


(b) Extended 2 - Tree

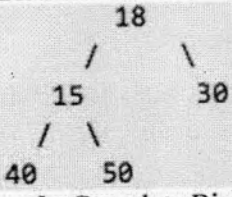
Difference between "Complete binary tree", "strict binary tree","full binary Tree"?

1) FULL BINARY TREE: A binary tree of height h that contains exactly (2^h)-1 elements is called a full **binary tree**. or in other words
In a FULL BINARY TREE each node has exactly 0 or 2 children and all leaf nodes are on the same level.

For Example: The following is a FULL BINARY TREE:



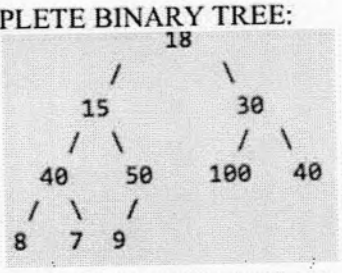
2) STRICT BINARY TREE: Each node has exactly 0 or 2 children.
For example: The following is a STRICT BINARY TREE:



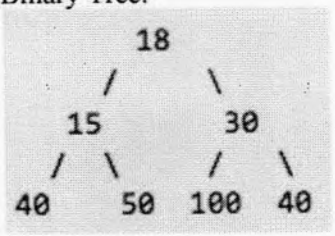
I think there's no confusion in the definition of a Complete Binary Tree, still for the completeness of the post I would like to tell what a Complete Binary Tree is.

3) COMPLETE BINARY TREE: A Binary Tree is complete Binary Tree if all levels are completely filled except possibly the last level and the last level has all keys as left as possible.

For Example: The following is a COMPLETE BINARY TREE:



Note: The following is also a Complete Binary Tree:

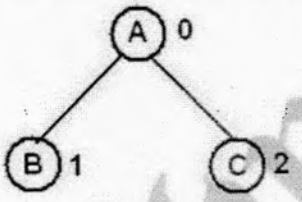


BINARY TREE REPRESENTATION

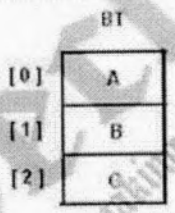
Array representation of a Binary Tree

An array can be used to store the nodes of a binary tree. The nodes stored in an array are accessible sequentially. The maximum number of nodes is specified by **MAXSIZE**.
In C, arrays start with index **0** to (**MAXSIZE – 1**). Here, numbering of binary tree nodes start from **0**. The root node is always at index **0**. Then, in successive memory locations the left child and right child are stored.

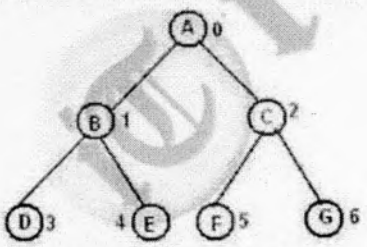
Consider a binary tree with only three nodes as shown. Let **BT** denote the binary tree.

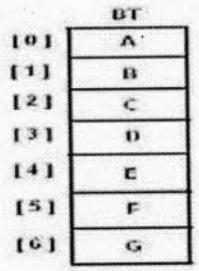


The array representation of this binary tree is as follows:



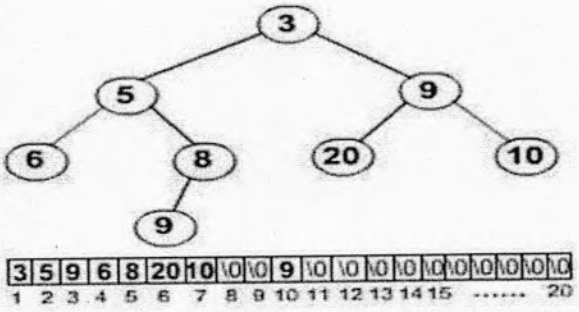
Here, A is the father of B and C, B is the left child of A and C is the right child of A. Let us extend the above tree by one more level as shown below :

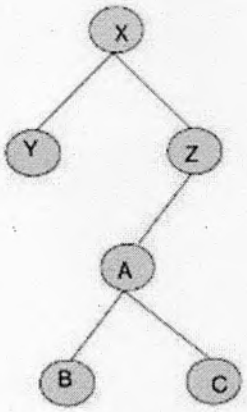


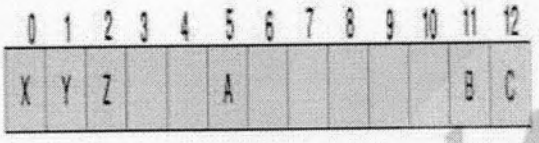


The array representation of this binary tree is as follows:

E.g. 2

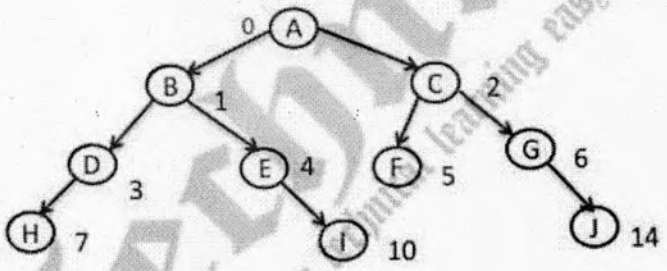




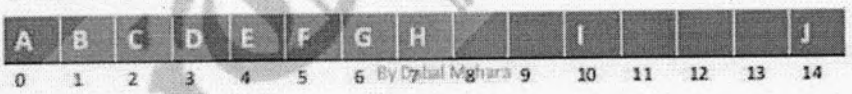


Array Representation

E.g. 4



Declare array of size $2^{3+1}-1 = 15$
Char a[15]



How to identify the father, the left child and the right child of an random node in such representation? It is very simple, to identify the father and the children of a node. For any node n, $0 <= n <= (MAXSIZE - 1)$, then we have

1. **father (n):** The father of node having index **n** is at floor ((n - 1) / 2), if n is not equal to 0. If n= 0, then it is the root node and has no father.
   Example : Consider a node numbered 3 (i.e. D). The father of D, no doubt, is B whose index is 1.
   This is obtained from
   floor ((3 – 1) / 2 = floor (2/2)
   = 1

2. **Lchild (n):** The left child of node numbered n is at $(2n + 1)$. For example, in the above binary tree
   (a) Lchild (A) = Lchild(0)
   $= 2 \times 0 + 1$
   $= 1$
   i.e., the node with index 1 which is nothing but B.
   (b) Lchild (C) = Lchild(2)
   $= 2 \times 2 + 1$
   $= 5$
   i.e., the node with index 5 which is nothing but F.

3. **Rchild (*n*):** The right child of node numbered *n* is index (2*n* + 2). For example, in the above binary tree,
   (a) Rchild(A) = Rchild(0)
   = 2 × 0 + 1
   = 2
   i.e., the node with index 2 which is nothing but C.
   (b) Rchild(B) = Rchild(1)
   = 2 × 1 + 2
   = 4
   i.e. the node with index 4 which is nothing but E.

4. **Siblings:** If the left child at index *n* is given then its right sibling (or brother) is at (*n* + 1). And, Similarly, if the right child at index *n* is given, then its left sibling is at (*n* - 1). For example, the right sibling is at (n – 1) of node indexed 4 is at index 5 in an array representation.

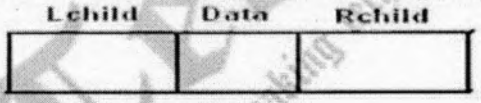
When a binary tree is represented using arrays, one array **A** stores the data fields of the trees. For this, numbers are given to each node starting from the root node – 0 to root node, 1 to the left node of the first level, then 2 to the second node from left of the first level and so on. In other words, the nodes are numbered from left to right level by level from top to bottom.

Linked representation of a Binary tree:

Binary tree can be represented either using array representation or using a linked list representation. The basic component to be represented in a binary tree is a node. The node consists of three fields such as

- Data
- Left child
- Right child

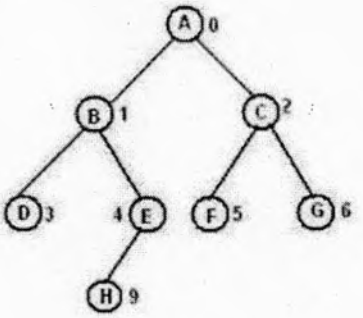
The data field holds the value to be given. The left child is a link field which contains the address of its left node and the right child contains the address of the right node. Fig. shows a structure of a node.



The logical representation of the node in C is given below :

```
struct tree
{
char data;
struct node *left;
struct node *right;
};
struct tree *ptr;
```

Consider the following binary tree :



Its linked representation is shown in Fig. below. In the above binary tree all the data items are of type char.

UNIT-4

Unit-04
Graphs

Introduction to Graphs

Graph is a nonlinear data structure; it contains a set of points known as nodes (or vertices) and set of links known as edges (or Arcs) which connects the vertices. A graph is defined as follows...

"Graph is a collection of Vertices and Edges which connects Vertices in the graph"

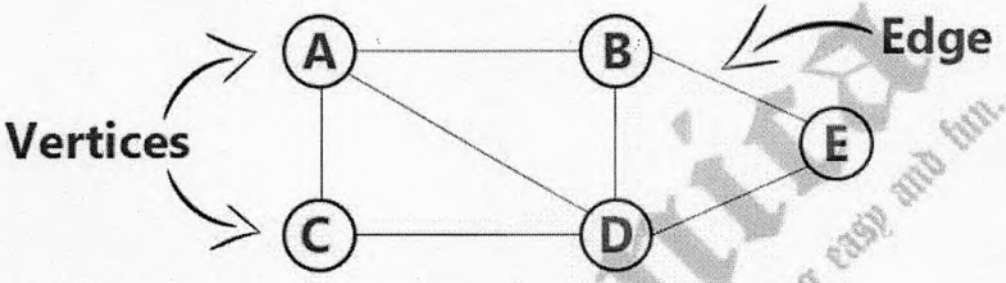
Generally, a graph **G** is represented as **G = (V, E)**, where **V is set of vertices** and **E is set of edges.**

Example

The following is a graph with 5 vertices and 6 edges.

This graph G can be defined as G = (V , E)

Where V = {A,B,C,D,E} and E = {(A,B),(A,C)(A,D),(B,D),(C,D),(B,E),(E,D)}.



Graph Terminology

We use the following terms in graph data structure...

Vertex

An individual data element of a graph is called as Vertex. **Vertex** is also known as **node.** In above example graph, A, B, C, D & E are known as vertices.

Edge

An edge is a connecting link between two vertices. **Edge** is also known as **Arc.** An edge is represented as (startingVertex, endingVertex). For example, in above graph, the link between vertices A and B is represented as (A, B). In above example graph, there are 7 edges (i.e., (A, B), (A, C), (A, D), (B, D), (B, E), (C, D), (D, E)).

Edges are of three types.

1. **Undirected Edge -** An undirected edge is a bidirectional edge. If there is a undirected edge between vertices A and B then edge (A , B) is equal to edge (B , A).
2. **Directed Edge -** A directed edge is a unidirectional edge. If there is a directed edge between vertices A and B then edge (A , B) is not equal to edge (B , A).
3. **Weighted Edge -** A weighted edge is an edge with cost on it.

- **Undirected Graph**
  A graph with only undirected edges is said to be undirected graph.
- **Directed Graph**
  A graph with only directed edges is said to be directed graph.
- **Mixed Graph**
  A graph with undirected and directed edges is said to be mixed graph.
- **End vertices or Endpoints**
  The two vertices joined by an edge are called the end vertices (or endpoints) of the edge.
- **Origin**
  If an edge is directed, its first endpoint is said to be origin of it.
- **Destination**
  If an edge is directed, its first endpoint is said to be origin of it and the other endpoint is said to be the destination of the edge.

- **Adjacent**
  If there is an edge between vertices A and B then both A and B are said to be adjacent. In other words, Two vertices A and B are said to be adjacent if there is an edge whose end vertices are A and B.
- **Incident**
  An edge is said to be incident on a vertex if the vertex is one of the endpoints of that edge.
- **Outgoing Edge**
  A directed edge is said to be outgoing edge on its origin vertex.
- **Incoming Edge**
  A directed edge is said to be incoming edge on its destination vertex.
- **Degree**
  Total number of edges connected to a vertex is said to be degree of that vertex.
- **Indegree**
  Total number of incoming edges connected to a vertex is said to be indegree of that vertex.
- **Outdegree**
  Total number of outgoing edges connected to a vertex is said to be outdegree of that vertex.
- **Parallel edges or Multiple edges**
  If there are two undirected edges to have the same end vertices, and for two directed edges to have the same origin and the same destination. Such edges are called parallel edges or multiple edges.
- **Self-loop**
  An edge (undirected or directed) is a self-loop if its two endpoints coincide.
- **Simple Graph**
  A graph is said to be simple if there are no parallel and self-loop edges.
- **Path**
  A path is a sequence of alternating vertices and edges that starts at a vertex and ends at a vertex such that each edge is incident to its predecessor and successor vertex.

Graph Representations

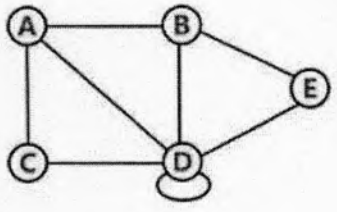
Graph data structure is represented using following representations...

1. **Adjacency Matrix**
2. **Incidence Matrix**
3. **Adjacency List**

Adjacency Matrix

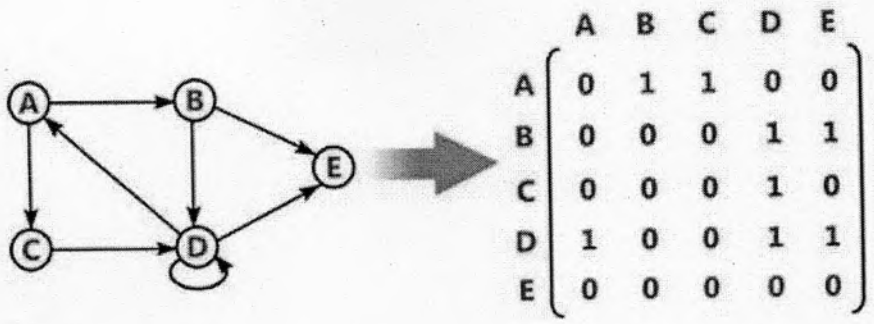
In this representation, graph can be represented using a matrix of size total number of vertices by total number of vertices. That means if a graph with 4 vertices can be represented using a matrix of 4X4 class. In this matrix, rows and columns both represents vertices. This matrix is filled with either 1 or 0. Here, 1 represents there is an edge from row vertex to column vertex and 0 represents there is no edge from row vertex to column vertex.

For example, consider the following undirected graph representation...



	A	B	C	D	E
A	0	1	1	1	0
B	1	0	0	1	1
C	1	0	0	1	0
D	1	1	1	1	1
E	0	1	0	1	0

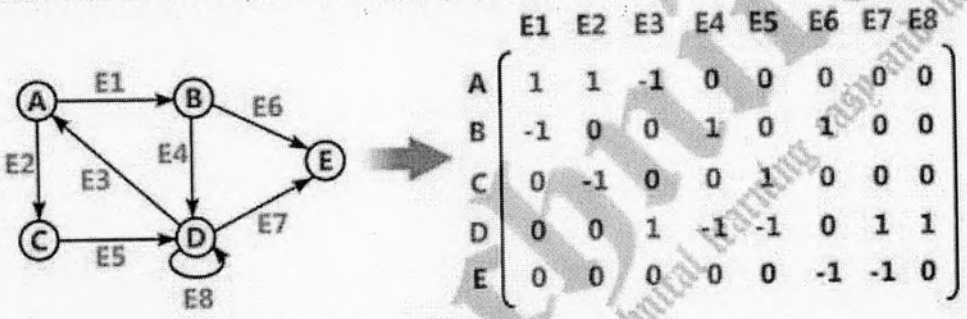
Directed graph representation...



Incidence Matrix

In this representation, graph can be represented using a matrix of size total number of vertices by total number of edges. That means if a graph with 4 vertices and 6 edges can be represented using a matrix of 4X6 class. In this matrix, rows represent vertices and columns represents edges. This matrix is filled with either 0 or 1 or -1. Here, 0 represents edge is not connected to row vertex, 1 represents edge is connected as outgoing edge to row vertex and -1 represents edge is connected as incoming edge to row vertex.

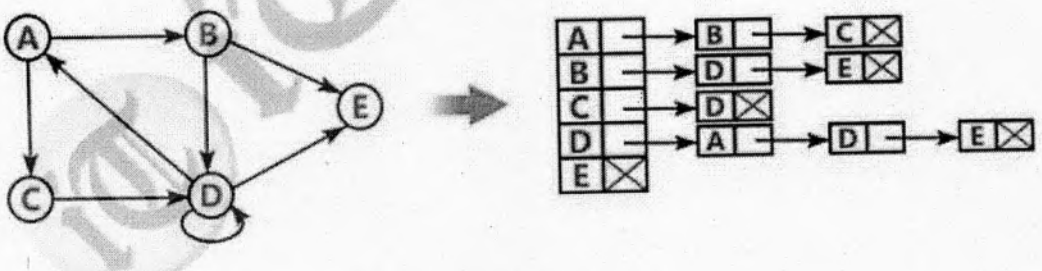
For example, consider the following directed graph representation...



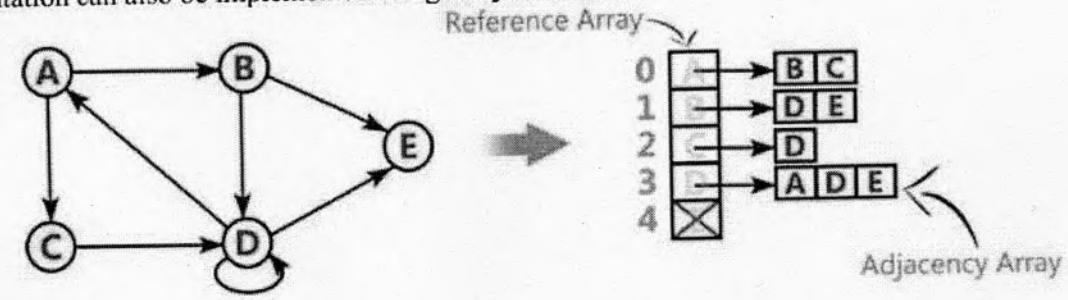
Adjacency List

In this representation, every vertex of graph contains list of its adjacent vertices.

For example, consider the following directed graph representation implemented using linked list...



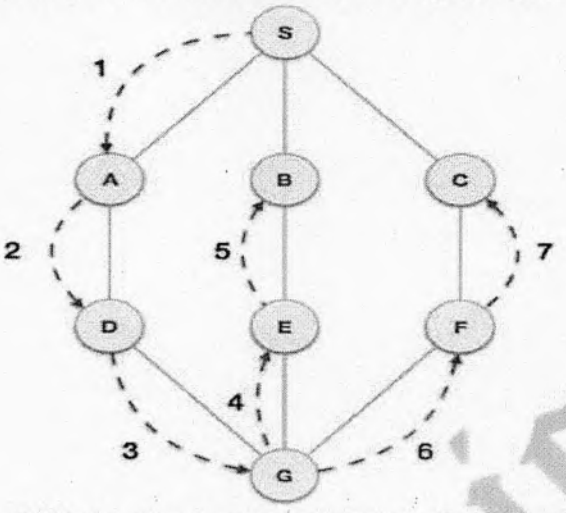
This representation can also be implemented using array as follows:-



Graph Traversals

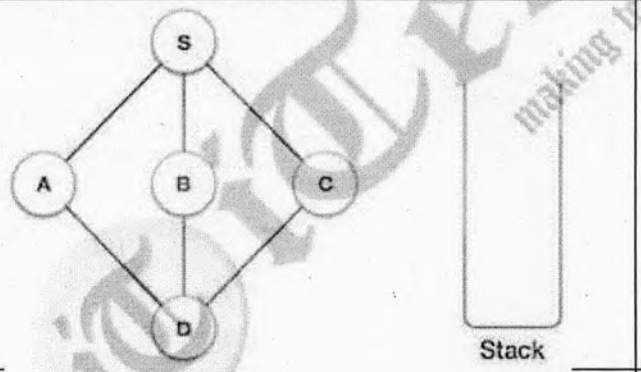
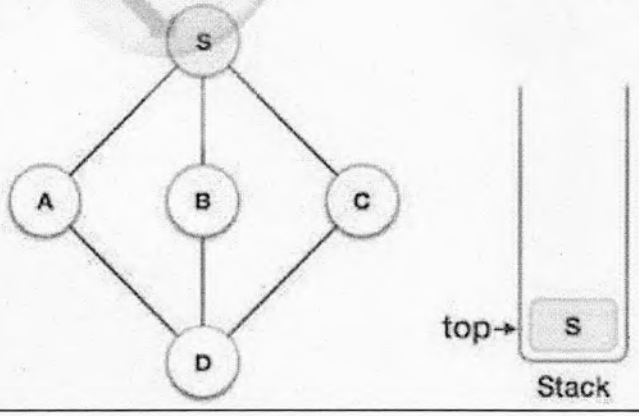
Depth First Search (DFS)

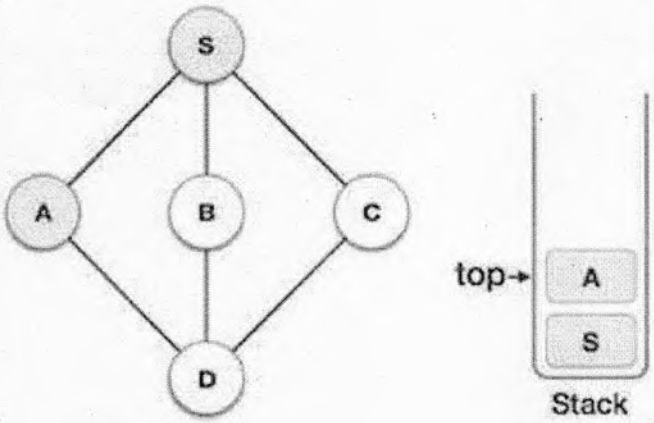
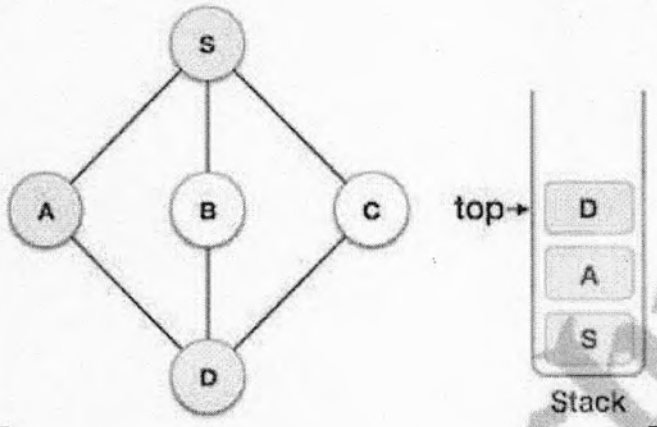
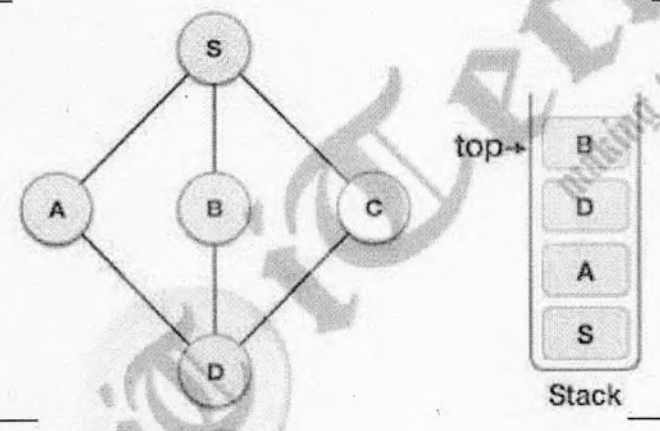
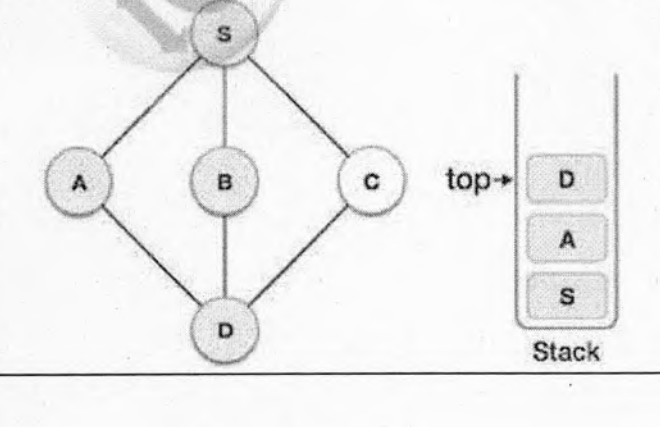
Depth First Search (DFS) algorithm traverses a graph in a depth ward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

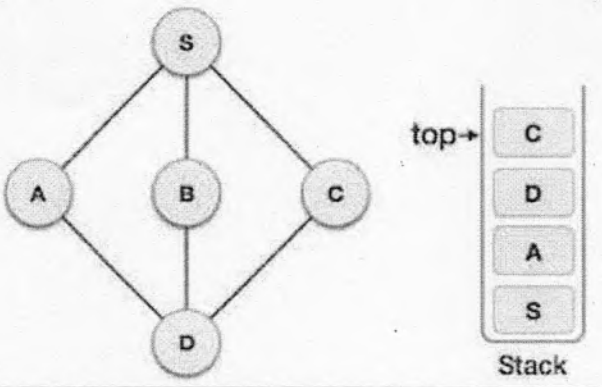


As in the example given above, DFS algorithm traverses from S to A to D to G to E to B first, then to F and lastly to C. It employs the following rules.

- **Rule 1** – Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack.
- **Rule 2** – If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.)
- **Rule 3** – Repeat Rule 1 and Rule 2 until the stack is empty.

Step	Traversal	Description
1.		Initialize the stack.
2.		Mark **S** as visited and put it onto the stack. Explore any unvisited adjacent node from **S**. We have three nodes and we can pick any of them. For this example, we shall take the node in an alphabetical order.

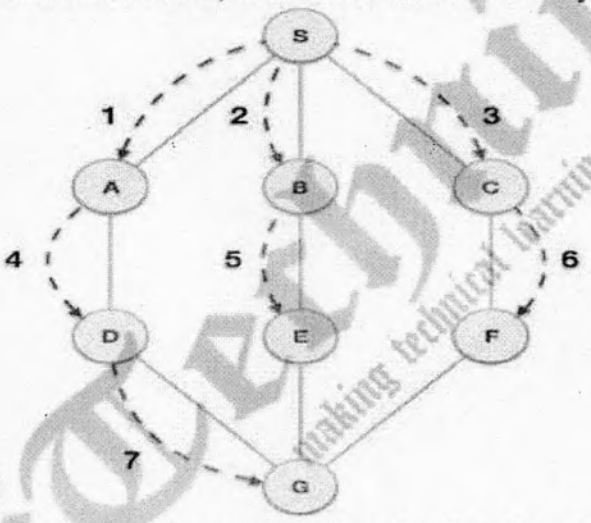
3.		Mark **A** as visited and put it onto the stack. Explore any unvisited adjacent node from A. Both **S** and **D** are adjacent to **A** but we are concerned for unvisited nodes only.
4.		Visit **D** and mark it as visited and put onto the stack. Here, we have **B** and **C** nodes, which are adjacent to **D** and both are unvisited. However, we shall again choose in an alphabetical order.
5.		We choose **B**, mark it as visited and put onto the stack. Here **B** does not have any unvisited adjacent node. So, we pop **B** from the stack.
6.		We check the stack top for return to the previous node and check if it has any unvisited nodes. Here, we find **D** to be on the top of the stack.

7.		Only unvisited adjacent node is from **D** is **C** now. So we visit **C**, mark it as visited and put it onto the stack.

As **C** does not have any unvisited adjacent node so we keep popping the stack until we find a node that has an unvisited adjacent node. In this case, there's none and we keep popping until the stack is empty.

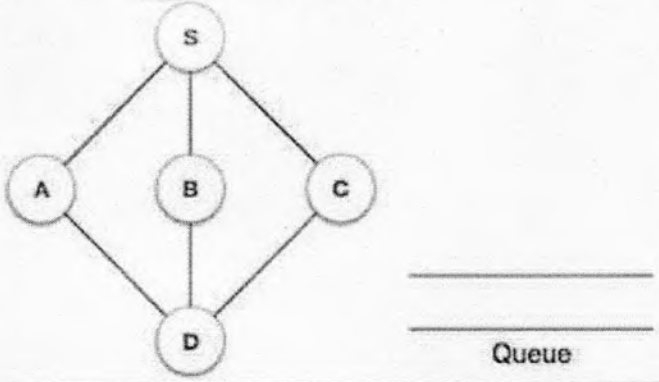
Breadth First Search (BFS)

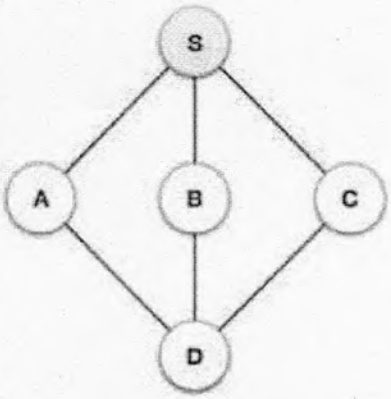
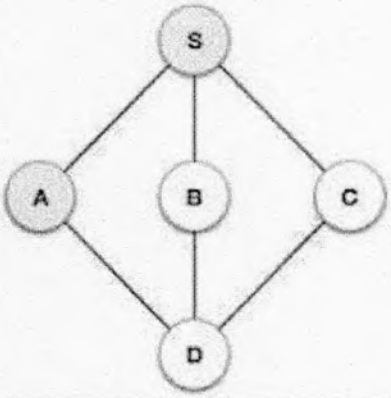
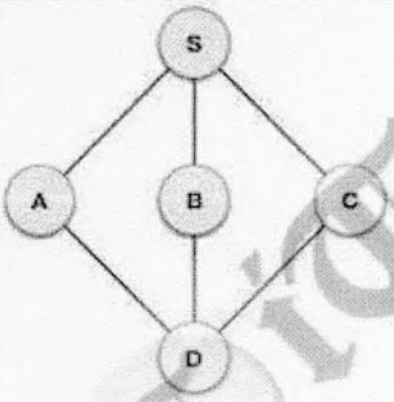
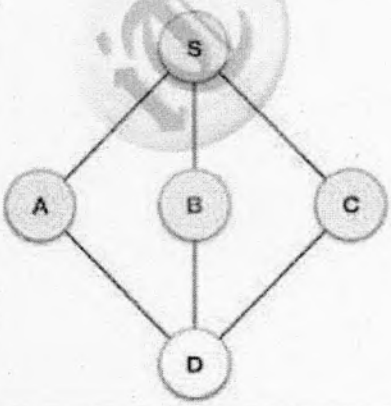
Breadth First Search (BFS) algorithm traverses a graph in a breadthward motion and uses a queue to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

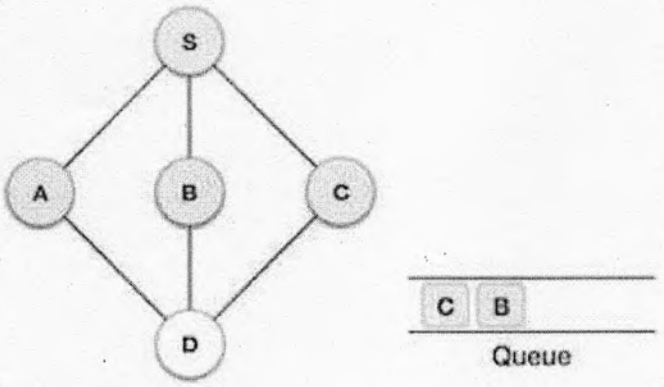
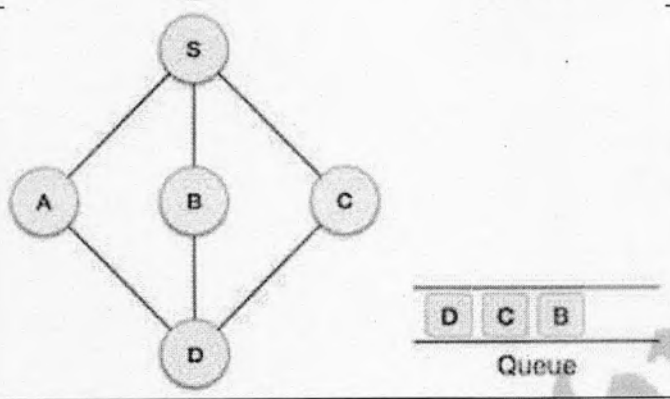


As in the example given above, BFS algorithm traverses from A to B to E to F first then to C and G lastly to D. It employs the following rules.

- **Rule 1** – Visit the adjacent unvisited vertex. Mark it as visited. Display it. Insert it in a queue.
- **Rule 2** – If no adjacent vertex is found, remove the first vertex from the queue.
- **Rule 3** – Repeat Rule 1 and Rule 2 until the queue is empty.

Step	Traversal	Description
1.		Initialize the queue.

2.	 Queue	We start from visiting **S** (starting node), and mark it as visited.
3.	 A Queue	We then see an unvisited adjacent node from **S**. In this example, we have three nodes but alphabetically we choose **A**, mark it as visited and enqueue it.
4.	 B A Queue	Next, the unvisited adjacent node from **S** is **B**. We mark it as visited and enqueue it.
5.	 C B A Queue	Next, the unvisited adjacent node from **S** is **C**. We mark it as visited and enqueue it.

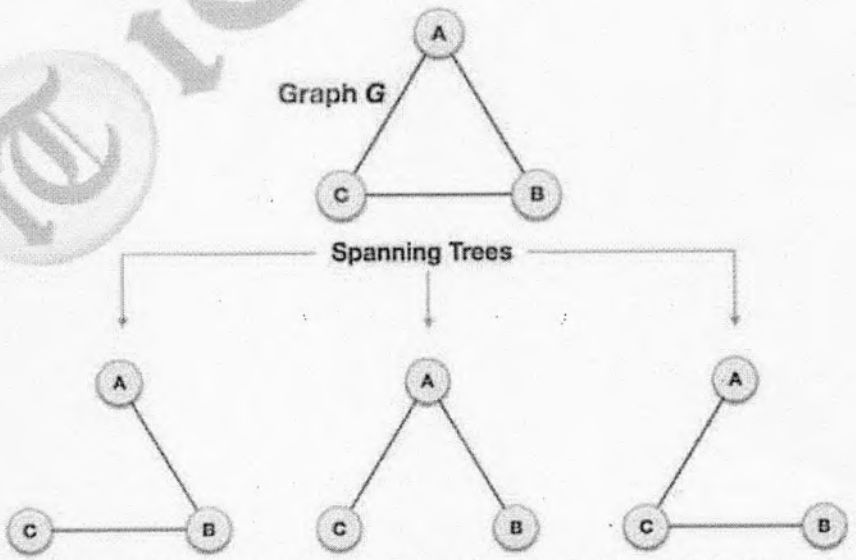
6.		Now, **S** is left with no unvisited adjacent nodes. So, we dequeue and find **A**.
7.		From **A** we have **D** as unvisited adjacent node. We mark it as visited and enqueue it.

At this stage, we are left with no unmarked (unvisited) nodes. But as per the algorithm we keep on dequeuing in order to get all unvisited nodes. When the queue gets emptied, the program is over.

Spanning Tree of a Graph

A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possible number of edges. Hence, a spanning tree does not have cycles and it cannot be disconnected.

By this definition, we can draw a conclusion that every connected and undirected Graph G has at least one spanning tree. A disconnected graph does not have any spanning tree, as it cannot be spanned to all its vertices.



We found 03 spanning trees off one complete graph. A complete undirected graph can have maximum $\mathbf{n^{n-2}}$ number of spanning trees, where **n** is the number of nodes. In the above addressed example, $\mathbf{3^{3-2} = 3}$ spanning trees are possible.

General Properties of Spanning Tree

We now understand that one graph can have more than one spanning tree. Following are a few properties of the spanning tree connected to graph G –

- A connected graph G can have more than one spanning tree.
- All possible spanning trees of graph G, have the same number of edges and vertices.
- The spanning tree does not have any cycle (loops).
- Removing one edge from the spanning tree will make the graph disconnected, i.e. the spanning tree is **minimally connected**.
- Adding one edge to the spanning tree will create a circuit or loop, i.e. the spanning tree is **maximally acyclic**.

Mathematical Properties of Spanning Tree

- Spanning tree has **n-1** edges, where **n** is the number of nodes (vertices).
- From a complete graph, by removing maximum **e - n + 1** edges, we can construct a spanning tree.
- A complete graph can have maximum $\mathbf{n^{n-2}}$ number of spanning trees.

Thus, we can conclude that spanning trees are a subset of connected Graph G and disconnected graphs do not have spanning tree.

Application of Spanning Tree

Spanning tree is basically used to find a minimum path to connect all nodes in a graph. Common application of spanning trees are –

- **Civil Network Planning**
- **Cluster Analysis**
- **Computer Network Routing Protocol**

Let us understand this through a small example. Consider, city network as a huge graph and now plans to deploy telephone lines in such a way that in minimum lines we can connect to all city nodes. This is where the spanning tree comes into picture.

Minimum Spanning Tree (MST)

In a weighted graph, a minimum spanning tree is a spanning tree that has minimum weight than all other spanning trees of the same graph. In real-world situations, this weight can be measured as distance, congestion, traffic load or any arbitrary value denoted to the edges.

Minimum Spanning-Tree Algorithm

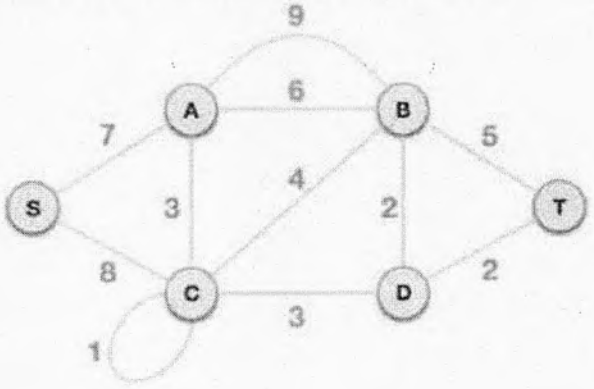
We shall learn about two most important spanning tree algorithms here –

- Kruskal's Algorithm
- Prim's Algorithm

Kruskal's algorithm

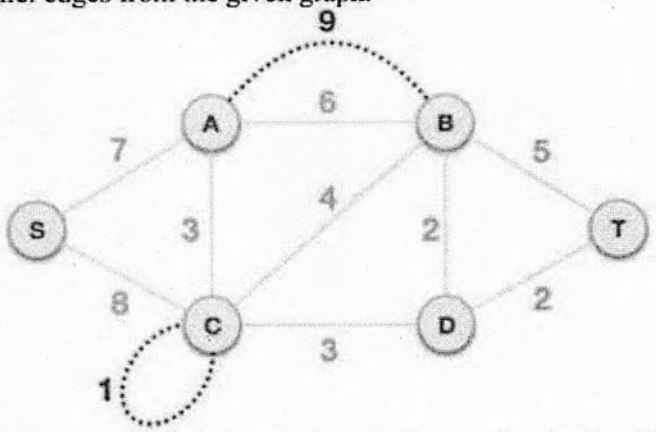
It is used to find the minimum cost spanning tree uses the greedy approach. This algorithm treats the graph as a forest and every node it has as an individual tree. A tree connects to another only and only if, it has the least cost among all available options and does not violate MST properties.

To understand Kruskal's algorithm let us consider the following example –

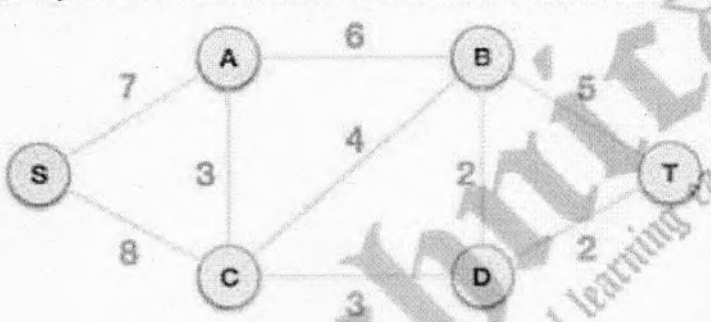


Step 1 - Remove all loops and Parallel Edges

Remove all loops and parallel edges from the given graph.



In case of parallel edges, keep the one which has the least cost associated and remove all others.



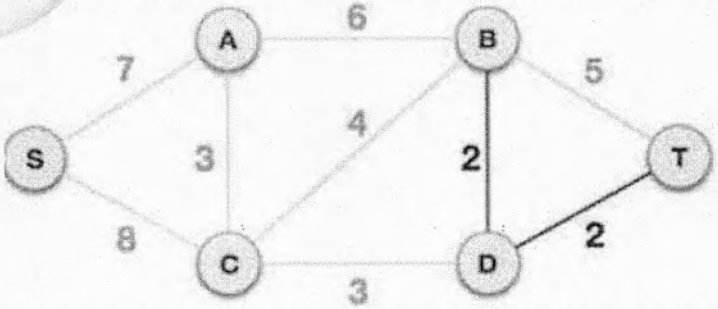
Step 2 - Arrange all edges in their increasing order of weight

The next step is to create a set of edges and weight, and arrange them in an ascending order of weightage (cost).

B, D	D, T	A, C	C, D	C, B	B, T	A, B	S, A	S, C
2	2	3	3	4	5	6	7	8

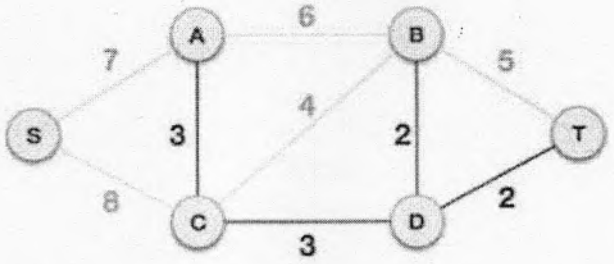
Step 3 - Add the edge which has the least weightage

Now we start adding edges to the graph beginning from the one which has the least weight. Throughout, we shall keep checking that the spanning properties remain intact. In case, by adding one edge, the spanning tree property does not hold then we shall consider not to include the edge in the graph.

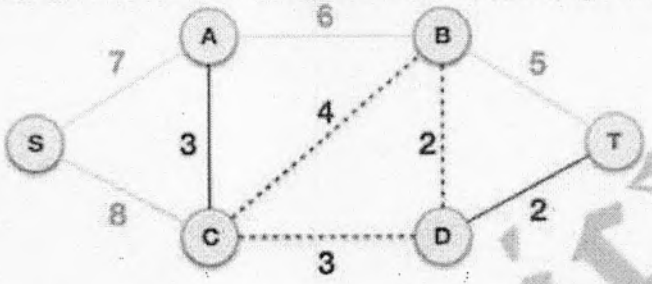


The least cost is 2 and edges involved are B, D and D, T. We add them. Adding them does not violate spanning tree properties, so we continue to our next edge selection.

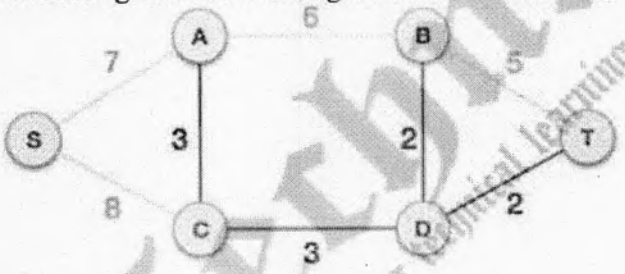
Next cost is 3, and associated edges are A, C and C,D. We add them again –



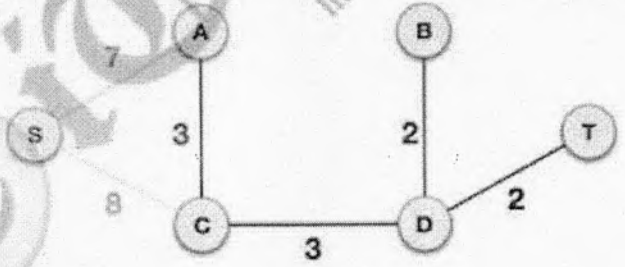
Next cost in the table is 4, and we observe that adding it will create a circuit in the graph. –



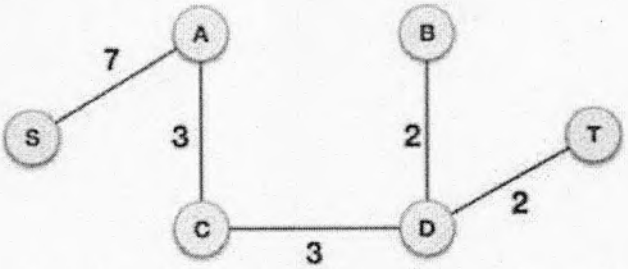
We ignore it. In the process we shall ignore/avoid all edges that create a circuit.



We observe that edges with cost 5 and 6 also create circuits. We ignore them and move on.



Now we are left with only one node to be added. Between the two least cost edges available 7 and 8, we shall add the edge with cost 7.



By adding edge S,A we have included all the nodes of the graph and we now have minimum cost spanning tree.

UNIT-5

Unit-05
Searching, Sorting and Search Trees

What is Searching?

Searching is a process of finding a value in a list of values. In other words, searching is the process of locating given value position in a list of values.

Linear Search Algorithm (Sequential Search Algorithm)

Linear search algorithm finds given element in a list of elements with **O(n)** time complexity where **n** is total number of elements in the list.

Linear search is implemented using following steps...

- **Step 1:** Read the search element from the user
- **Step 2:** Compare, the search element with the first element in the list.
- **Step 3:** If both are matching, then display "Given element found!!!" and terminate the function
- **Step 4:** If both are not matching, then compare search element with the next element in the list.
- **Step 5:** Repeat steps 3 and 4 until the search element is compared with the last element in the list.
- **Step 6:** If the last element in the list is also doesn't match, then display "Element not found!!!" and terminate the function.

Example

Consider the following list of element and search element...

WAP to implement Linear Search algorithm in "C"

```
#include<stdio.h>
int main()
{
  int a[10]={1,2,4,6,7,8,9,10,11,13};
  int item,i,flag,c=0;;

  printf("Enter the item to be searched:");
  scanf("%d",&item);

  for(i=0;i<9;i++)
  {
    if(item==a[i])
    {
      flag=1;
      break;
    }
  }
  if(flag==1)
  {
    printf("\nItem found at index %d",i);
  }
  else
  {
    printf("Item not found");
  }
  return 0;
}
```

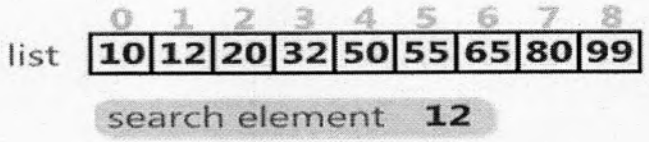
Binary Search Algorithm

Binary search algorithm finds given element in a list of elements with **O(log n)** time complexity where **n** is total number of elements in the list. The binary search algorithm can be used with only sorted list of element. The binary search cannot be used for list of elements, which are in random order.

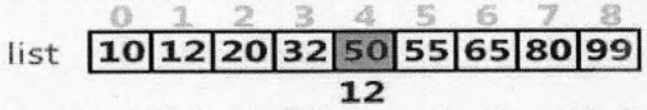
Binary search is implemented using following steps...

- **Step 1:** Read the search element from the user
- **Step 2:** Find the middle element in the sorted list
- **Step 3:** Compare, the search element with the middle element in the sorted list.
- **Step 4:** If both are matching, then display "Given element found!!!" and terminate the function
- **Step 5:** If both are not matching, then check whether the search element is smaller or larger than middle element.
- **Step 6:** If the search element is smaller than middle element, then repeat steps 2, 3, 4 and 5 for the left sublist of the middle element.
- **Step 7:** If the search element is larger than middle element, then repeat steps 2, 3, 4 and 5 for the right sublist of the middle element.
- **Step 8:** Repeat the same process until we find the search element in the list or until sublist contains only one element.
- **Step 9:** If that element also doesn't match with the search element, then display "Element not found in the list!!!" and terminate the function.

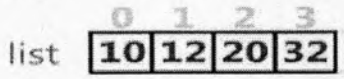
Example
Consider the following list of element and search element...



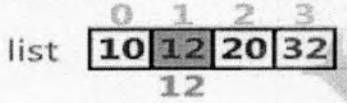
Step 1:
search element (12) is compared with middle element (50)



Both are not matching. And 12 is smaller than 50. So we search only in the left sublist (i.e. 10, 12, 20 & 32).



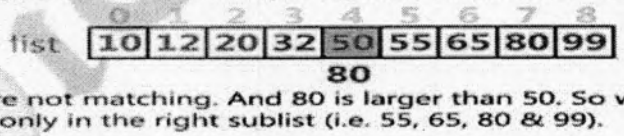
Step 2:
search element (12) is compared with middle element (12)



Both are matching. So the result is "Element found at index 1"

Now consider, we want to search another element in the same list of items.

Step 1:
search element (80) is compared with middle element (50)

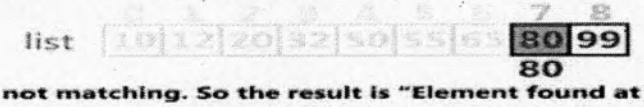


Both are not matching. And 80 is larger than 50. So we search only in the right sublist (i.e. 55, 65, 80 & 99).

Step 2:
search element (80) is compared with middle element (65)

Both are not matching. And 80 is larger than 65. So we search only in the right sublist (i.e. 80 & 99).

Step 3:
search element (80) is compared with middle element (80)



Both are not matching. So the result is "Element found at index 7"

WAP to implement BINARY SEARCH algorithm in "C"

```
#include<stdio.h>
void main()
{
  int a[10]={1,2,4,6,7,8,9,10,11,13};
  int i,item, flag;
  int low=0,high=9,mid;

  printf("Enter the item to be searched:");
  scanf("%d",&item);

 for (i=0;i<10;i++)
  {
    mid=(low+high)/2;
    if(item==a[mid])
    {
      flag=1;
      break;
    }
    else if(item<a[mid])
    {
      high=mid-1;
    }
    else
    {
      low=mid+1;
    }
  }
  if(flag==1)
  {
    printf("Item found at index %d",mid);
  }
  else
  {
    printf("Item not found");
  }
  getch();
}
```

Sorting

Sorting refers to arranging data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order. Most common orders are in numerical or lexicographical order.
The importance of sorting lies in the fact that data searching can be optimized to a very high level, if data is stored in a sorted manner. Sorting is also used to represent data in more readable formats.

Following are some of the examples of sorting in real-life scenarios −

- **Telephone Directory** − The telephone directory stores the telephone numbers of people sorted by their names, so that the names can be searched easily.
- **Dictionary** − The dictionary stores words in an alphabetical order so that searching of any word becomes easy.

Types of Sorting

There are many types of Sorting techniques, differentiated by their efficiency and space requirements. Following are some sorting techniques:

1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Quick Sort
5. Merge Sort
6. Heap Sort

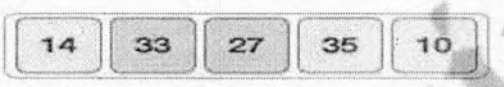
How Bubble Sort Works?

We take an unsorted array for our example. Bubble sort takes $O(n^2)$ time so we're keeping it short and precise.

Bubble sort starts with very first two elements, comparing them to check which one is greater.

In this case, value 33 is greater than 14, so it is already in sorted locations. Next, we compare 33 with 27.

We find that 27 is smaller than 33 and these two values must be swapped.



The new array should look like this –

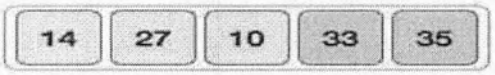
Next we compare 33 and 35. We find that both are in already sorted positions.

Then we move to the next two values, 35 and 10.

We know then that 10 is smaller 35. Hence they are not sorted.

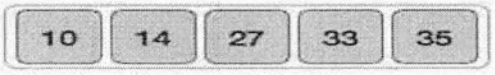
We swap these values. We find that we have reached the end of the array. After one iteration, the array should look like this –

To be precise, we are now showing how an array should look like after each iteration. After the second iteration, it should look like this –



Notice that after each iteration, at least one value moves at the end.

And when there's no swap required, bubble sorts learns that an array is completely sorted.



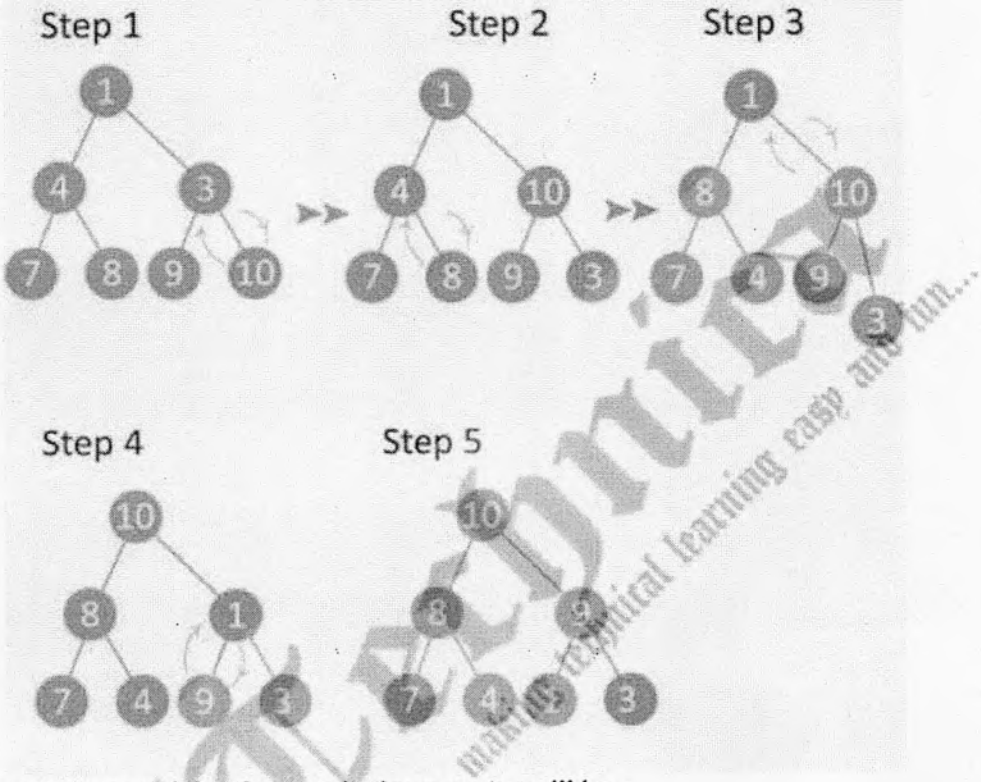
Algorithm

We assume **list** is an array of **n** elements. We further assume that **swap** function swaps the values of the given array elements.

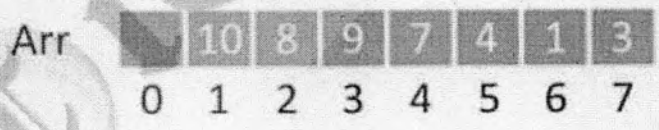
```
begin BubbleSort(list)

   for all elements of list
      if list[i] > list[i+1]
         swap(list[i], list[i+1])
      end if
   end for

   return list

end BubbleSort
```

Suppose you have 7 elements stored in array Arr

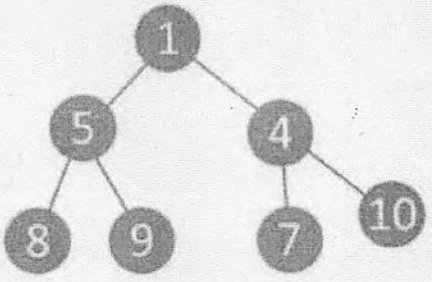
Converting into Max Heap



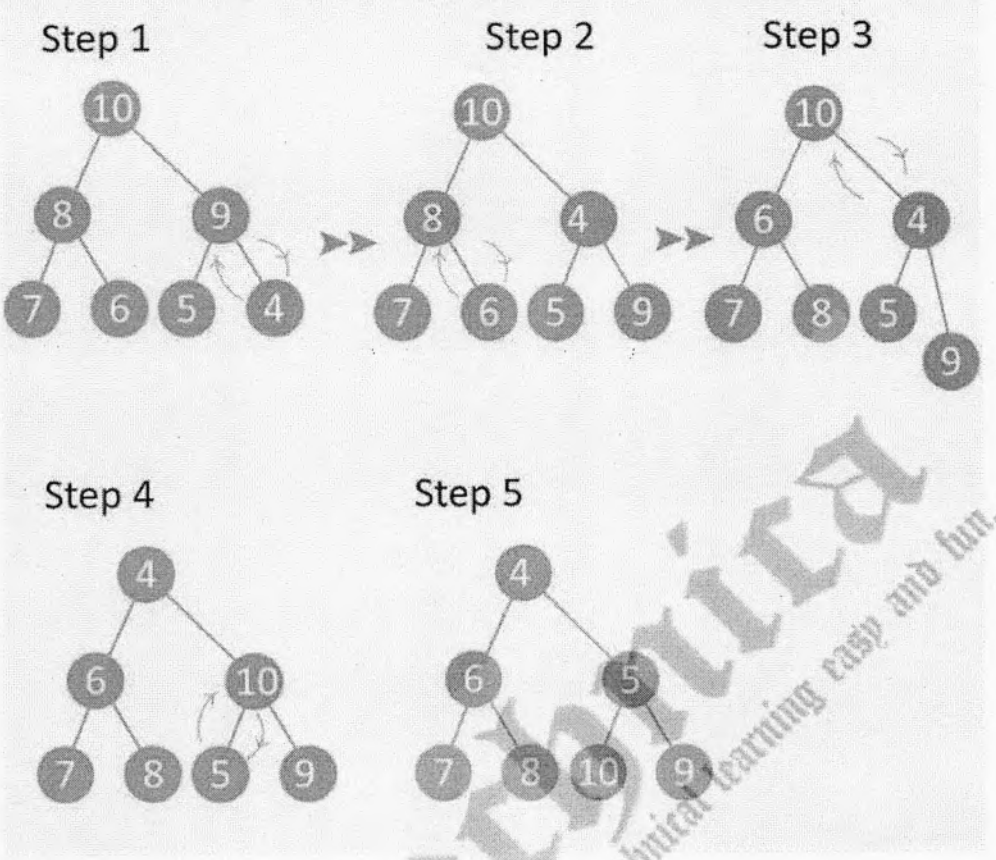
we finally get a max- heap and the elements in the array Arr will be :



Min Heap: In this type of heap, the value of parent node will always be less than or equal to the value of child node across the tree and the node with lowest value will be the root node of tree.



Consider elements in array {10, 8, 9, 7, 6, 5, 4} . We will run min_heapify on nodes to get a min_heap.



Heap Sort

Heaps can be used in sorting an array. In max-heaps, maximum element will always be at the root. Heap Sort uses this property of heap to sort the array.

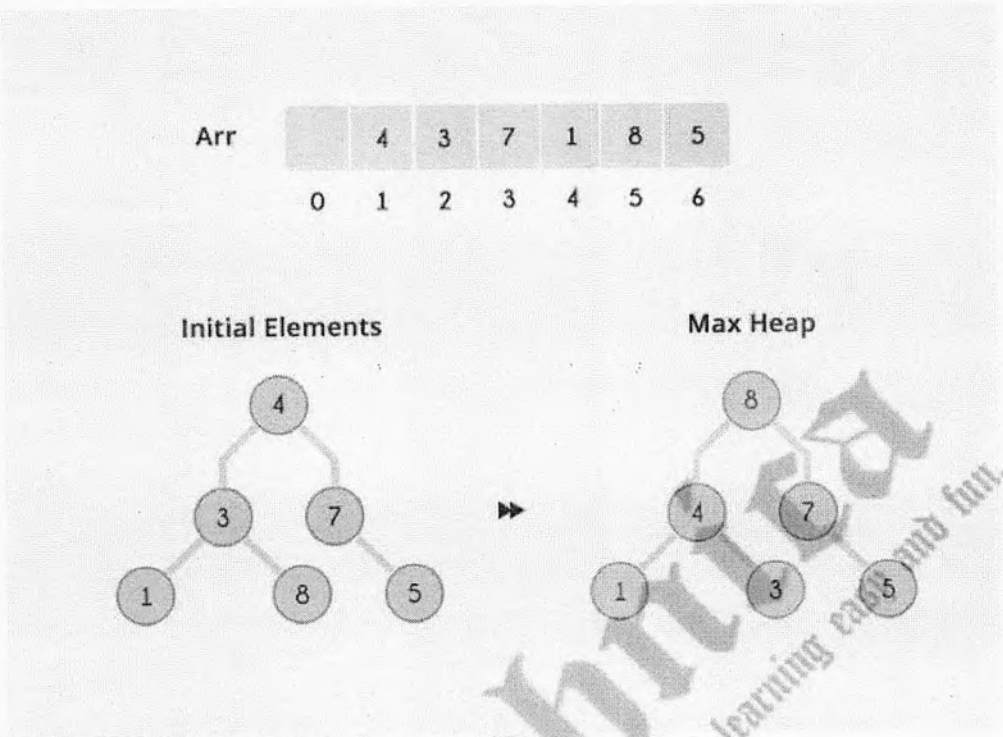
Consider an array Arr which is to be sorted using Heap Sort.

- Initially build a max heap of elements in Arr.
- The root element, that is $Arr[1]$, will contain maximum element of Arr. After that, swap this element with the last element of Arr and heapify the max heap excluding the last element which is already in its correct position and then decrease the length of heap by one.
- Repeat the step 2, until all the elements are in their correct position.

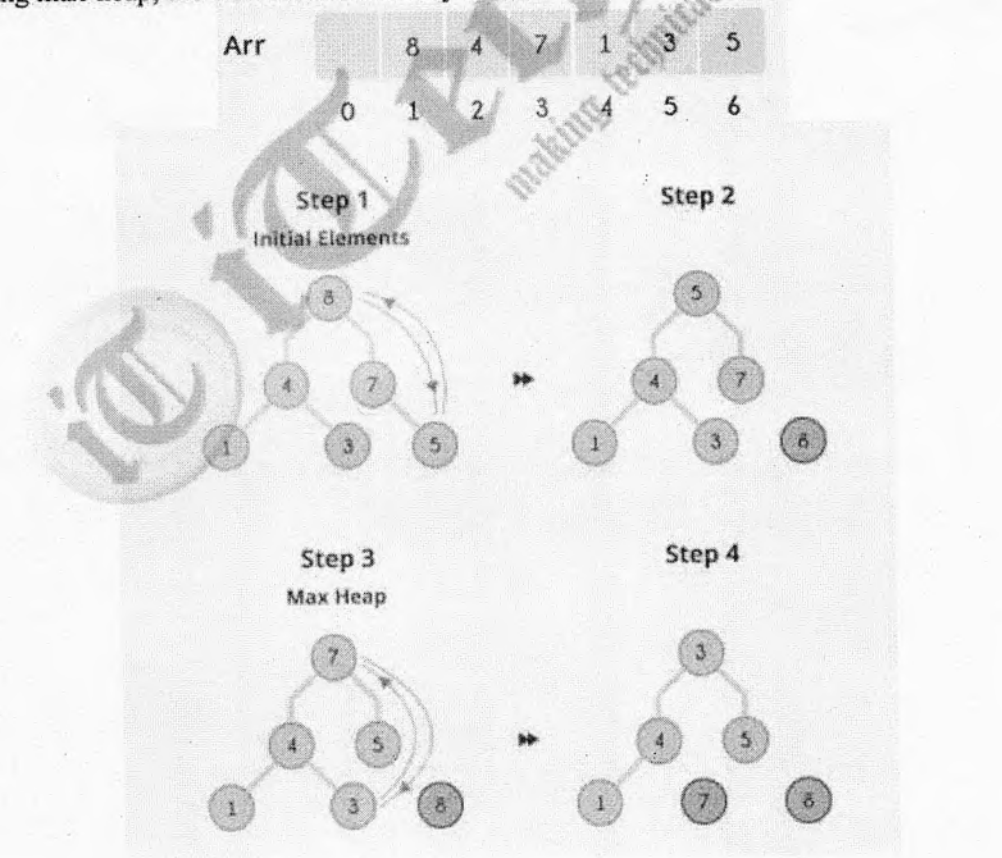
Implementation:

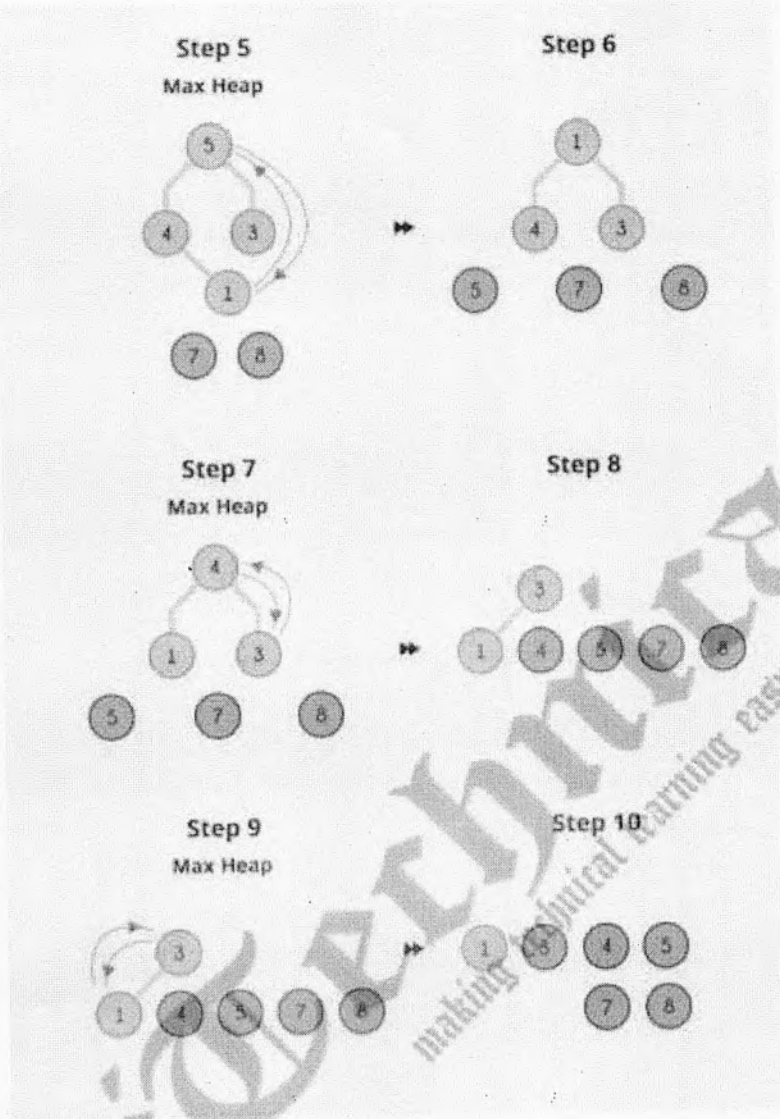
```
void heap_sort(int Arr[ ])

{
    int heap_size = N;

    build_maxheap(Arr);
    for(int i = N; i >= 2 ; i-- )
    {
        swap|(Arr[ 1 ], Arr[ i ]);
        heap_size = heap_size - 1;
        max_heapify(Arr, 1, heap_size);
    }
}
```

Example:
In the diagram below, initially there is an unsorted array Arr having 6 elements and then max-heap will be built.

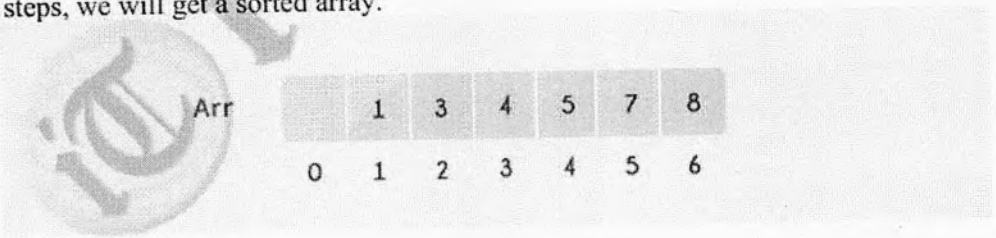


After building max-heap, the elements in the array will be:





After all the steps, we will get a sorted array.



To study further Sorting Techniques; Refer to Document named Unit-05 Sorting Techniques

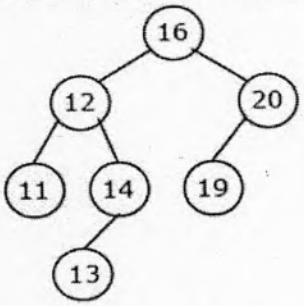
Search Trees

Binary Search Tree

A binary search tree is a binary tree. It may be empty. If it is not empty then it satisfies the following properties:

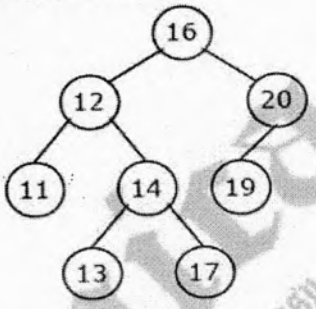
1. Every element has a key **and no two elements have the same key.**
2. The keys in the left subtree are smaller than the key in the root.
3. The keys in the right subtree are larger than the key in the root.
4. The left and right subtrees are also binary search trees.
5. Inorder Traversal gives sorted data.

Figure (a) is a binary search tree, whereas figure (b) is not a binary search tree.



Binary Search Tree

(a)



Not a Binary Search Tree

(b)

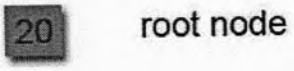
Common Operations on BST:

- **Insert:** Add a new node with a given key.
- **Delete:** Delete a node with a given key.
- **Search:** Search a node with a given key.
- **Maximum:** Find the largest key in a tree or subtree.
- **Minimum:** Find the smallest key in a tree or subtree.
- **Traverse:** Traverse the tree Inorder, Preorder or Postorder.

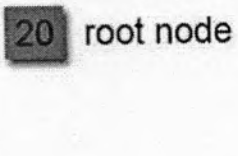
Creating a Binary Search Tree

A sequence of numbers is to formed into a binary search tree. These numbers are available in this order: 20, 17, 29, 22, 45, 9, 19. Task: form a sorted binary tree diagram
This is done step by step.

Sequence **20**, 17, 29, 22, 45, 9, 19
Step 1. The first item is 20 and this is the root node, so begin the diagram

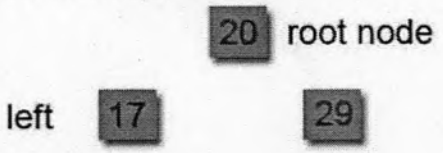


Step 2. Sequence 20, **17**, 29, 22, 45, 9, 19
This is a binary search tree, so there are two child nodes available, the LEFT and the RIGHT. The next number is 17, the rule is applied (left is less than parent node) and so it has to be the LEFT node, like this



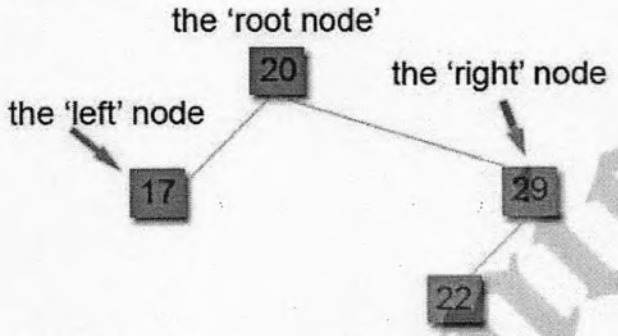
Step 3. Sequence 20, 17, **29**, 22, 45, 9, 19

The next number is 29, this is higher than the root node so it goes to the RIGHT sub-tree which happens to be empty at this stage, so the tree now looks like



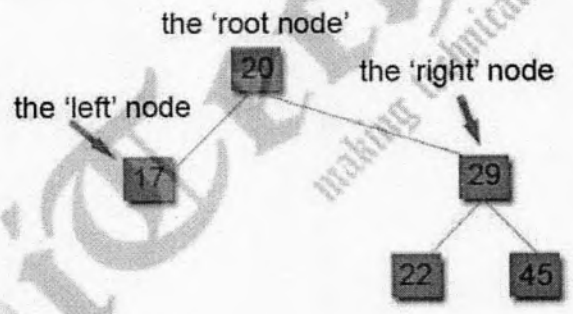
Step 4. Sequence 20, 17, 29, **22**, 45, 9, 19

The next number is 22. This is more that the root and so need so be on the RIGHT sub-tree. The first node is already occupied. So the rule is applied again to that node, 22 comes before 29 and so it needs to be on the LEFT sub-tree of that node, like this



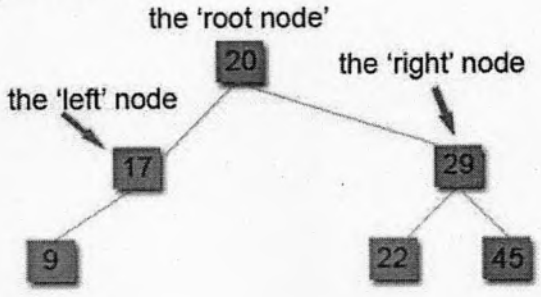
Step 5. Sequence 20, 17, 29, 22, **45**, 9, 19

The next number is 45, this is more than the root and more than the first right node, so it is placed on the right side of the tree like this



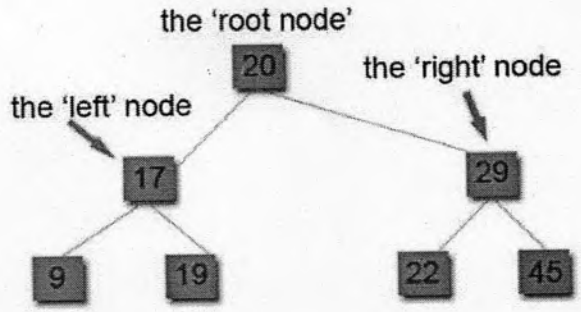
Step 6. Sequence 20, 17, 29, 22, 45, **9**, 19

The next number is 9 which is less than the root, the first left node is occupied and 9 is less than that node too, so it is placed on the left sub-tree, like this



Step 7 Sequence 20, 17, 29, 22, 45, 9, **19**

The next number is 19, which is less than the root, so it will need to be in the left sub-tree. It is greater than the occupied 17 node and so it is placed in the right sub-tree, like this



The tree is now complete.

Insertion in a Binary Search Tree

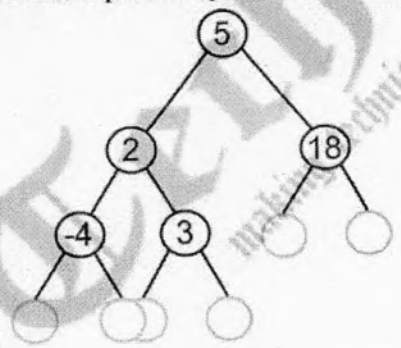
Adding a value to BST can be divided into two stages:

- search for a place to put a new element;
- insert the new element to this place.

Let us see these stages in more detail.

Search for a place

At this stage an algorithm should follow binary search tree property. If a new value is less, than the current node's value, go to the left subtree, else go to the right subtree. Following this simple rule, the algorithm reaches a node, which has no left or right subtree. By the moment a place for insertion is found, we can say for sure, that a new value has no duplicate in the tree. Initially, a new node has no children, so it is a leaf. Let us see it at the picture. Gray circles indicate possible places for a new node.

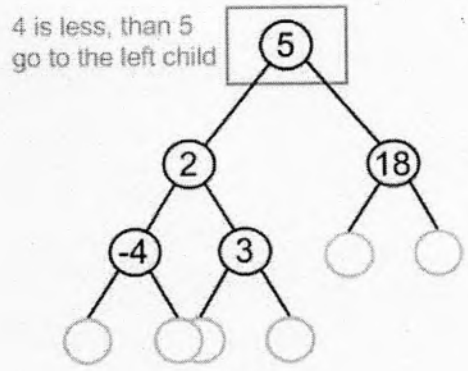


Now, let's go down to algorithm itself. Here and in almost every operation on BST recursion is utilized. Starting from the root,

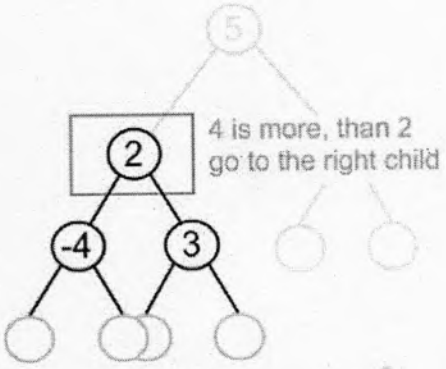
1. check, whether value in current node and a new value are equal. If so, duplicate is found. Otherwise,
2. if a new value is less, than the node's value:
    - if a current node has no left child, place for insertion has been found;
    - otherwise, handle the left child with the same algorithm.
3. if a new value is greater, than the node's value:
    - if a current node has no right child, place for insertion has been found;
    - otherwise, handle the right child with the same algorithm.

Just before code snippets, let us have a look on the example, demonstrating a case of insertion in the binary search tree.

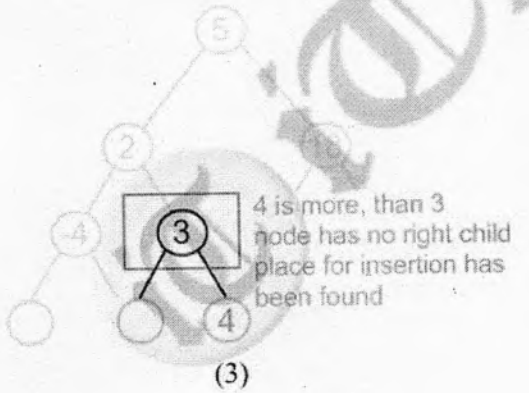
Example
Insert 4 to the tree, shown above.



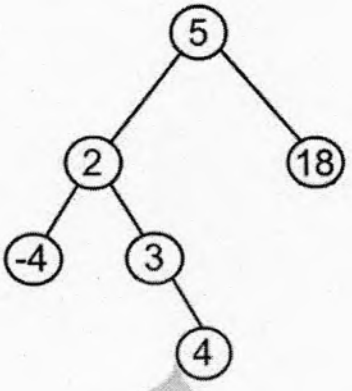
(1)



(2)



(3)



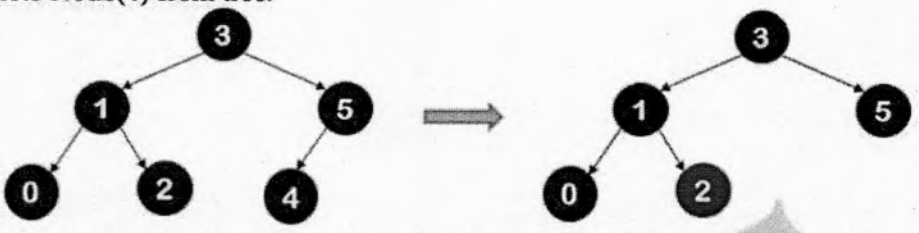
(4)

Deletion from a Binary Search Tree

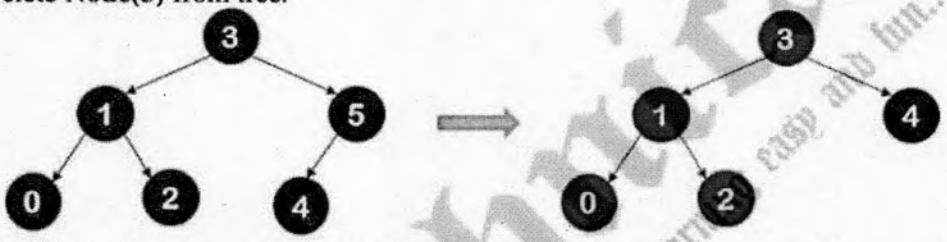
To delete a node/key from the given binary search tree, we need to consider three cases.

- **First case** - if the node to be deleted is a leaf node
- **Second Case** - if the node to be deleted has only one child.
- **Third case** - if the node to be deleted has both children.

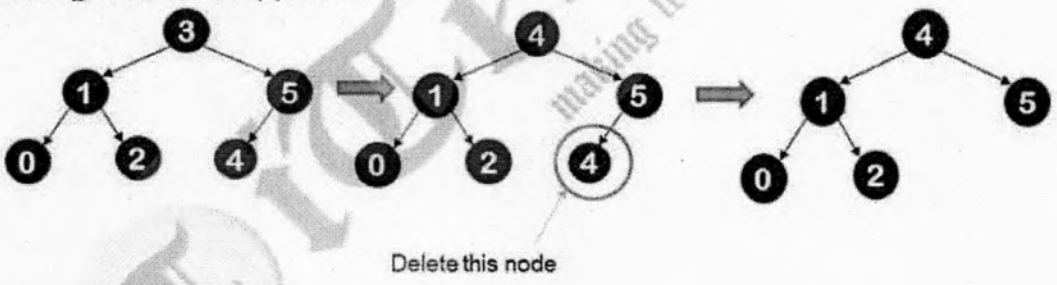
1) **Node to be deleted is leaf:** Simply remove from the tree.
For Eg: Delete **Node(4)** from tree.



2) **Node to be deleted has only one child:** Copy the child to the node and delete the child.
For Eg: Delete **Node(5)** from tree.



3) **Node to be deleted has two children:** Find inorder successor of the node. Copy contents of the inorder successor to the node and delete the inorder successor. Note that inorder predecessor can also be used.
For Eg: Delete **Node(3)** from tree.

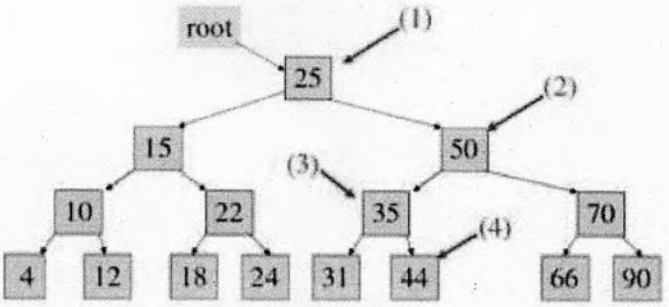


Searching in a Binary Search Tree

Example: search for 45 in the tree

(key fields are show in node rather than in separate obj ref to by data field):

1. start at the root, 45 is greater than 25, search in right subtree
2. 45 is less than 50, search in 50's left subtree
3. 45 is greater than 35, search in 35's right subtree
4. 45 is greater than 44, but 44 has no right subtree so 45 is not in the BST

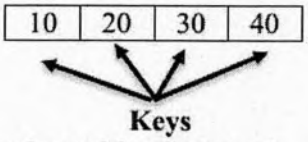


B Tree

It is a self-balanced search tree with multiple keys in every node and more than two children for every node. In this B doesn't represent any word however, it may stand for:

1. Balanced
2. Broad
3. Bayer (named after Inventor Bayer Rudolf)

Structure:



Properties: (B Tree of order m or branching factor m)

Order: defines max and min no of children for a particular node.

1. All leaf nodes must be at same level.
2. All nodes except root can have:
   a. Max no of keys: (m-1)
   b. Min no of keys: Ceil ((m/2)-1)
3. A non-leaf node with (n-1) keys must have n children
4. If the root is non-leaf node then it must have at least 2 children
5. All key values with in a node must be in ascending order.

(B Tree: If degree is given as min degree t)

Degree: defines max and min no of keys for a particular node.

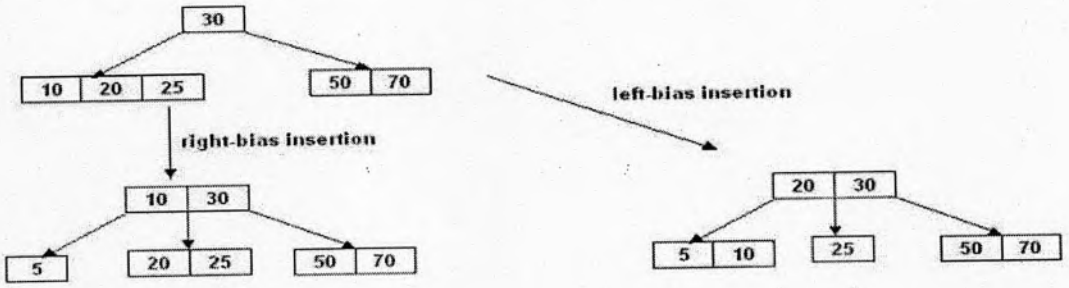
1. Each node has at most (2t-1) keys.
2. If node is not root node, it must have at least (t-1) keys.

For E.g. If order of B Tree is m=4
It can contain max (4-1) = 3 keys and min ceil ((4/2)-1) = 1 keys.
While constructing B Tree we may need to perform split operation, which can have two cases:

1. **B Tree's order is Odd Number**
   For example: 5 (max no of keys = 4)
   In such case split operation will be needed when tree node contains five nodes, then we find the median key for a node by following formula:
   Ceil ((m+1)/2) where m = order of tree
   Median key for node: ceil((5+1)/2) = ceil (6/2) = 3rd key

2. **B Tree's order is Even Number**
   For example: 4 (max no of keys = 3)
   In such case split operation will be needed when tree node contains four nodes, then we find the median key for a node by following formula:
   Ceil ((m+1)/2) where m = order of tree
   Median key for node: ceil((4+1)/2) = ceil (5/2) = 3rd key

- But sometimes, it can be done in following manner:
  - **Right-bias:** The node is split such that its right subtree has more keys than the left subtree.
  - **Left-bias:** The node is split such that its left subtree has more keys than the right subtree.
  - **Example**: Insert the key **5** in the following B-tree of order **4**:



Insertion in B Tree

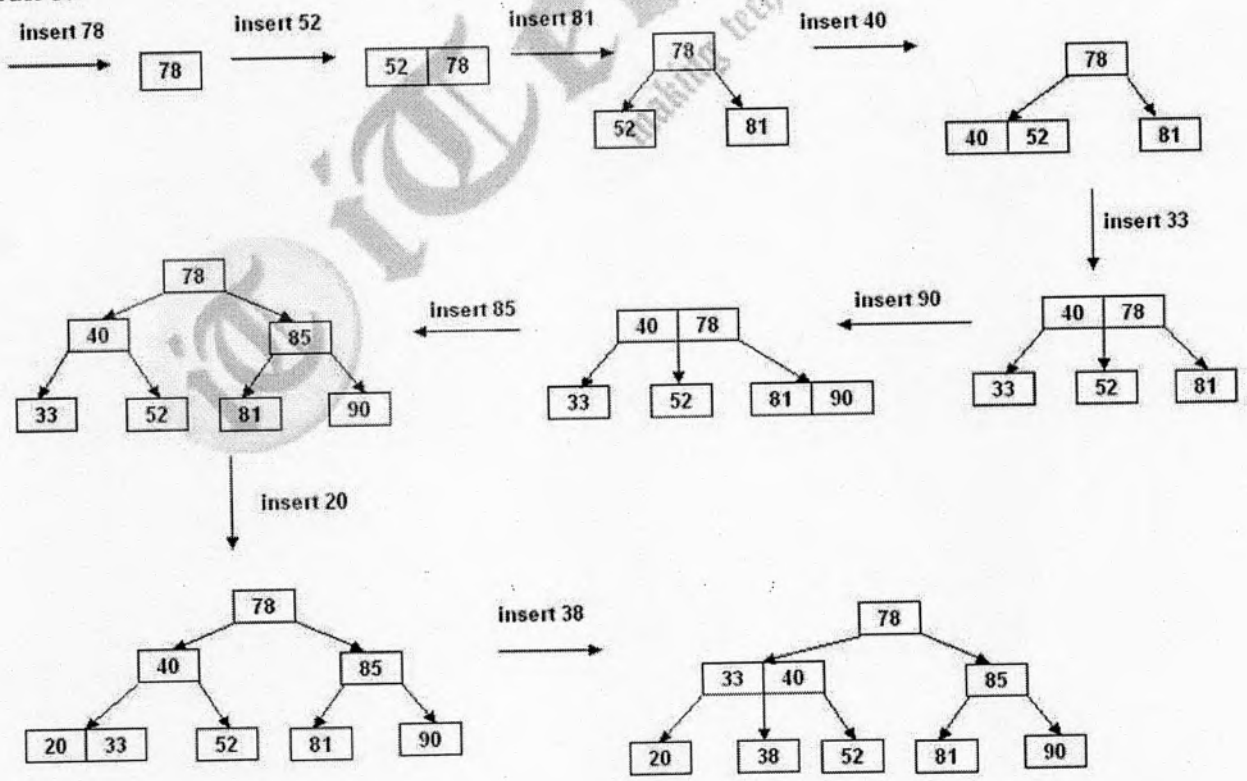
Insertion algorithm:

- If a node overflows, split it into two, propagate the "middle" key to the parent of the node.
- If the parent overflows the process propagates upward.
- If the node has no parent, create a new root node.

Overflow Condition: A root-node or a non-root node of a B-tree of order m overflows if, after a key insertion, it contains m keys.

Note: Insertion of a key always starts at a leaf node.

Example: Insert the keys 78, 52, 81, 40, 33, 90, 85, 20, and 38 in this order in an initially empty B-tree of order 3.

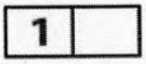


Example:

Construct a B-Tree of order 3 by inserting numbers from 1 to 10.

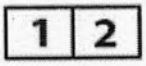
insert(1)

Since '1' is the first element into the tree that is inserted into a new node. It acts as the root node.



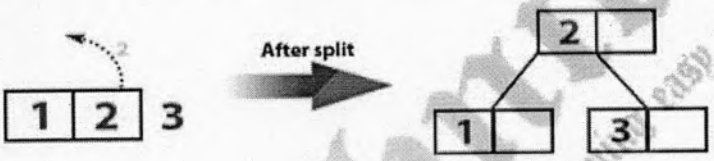
insert(2)

Element '2' is added to existing leaf node. Here, we have only one node and that node acts as root and also leaf. This leaf node has an empty position. So, new element (2) can be inserted at that empty position.



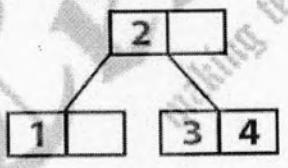
insert(3)

Element '3' is added to existing leaf node. Here, we have only one node and that node acts as root and also leaf. This leaf node doesn't has an empty position. So, we split that node by sending middle value (2) to its parent node. But here, this node doesn't has parent. So, this middle value becomes a new root node for the tree.



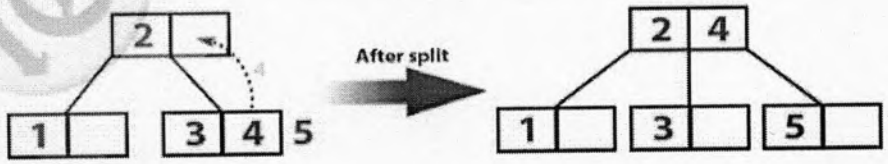
insert(4)

Element '4' is larger than root node '2' and it is not a leaf node. So, we move to the right of '2'. We reach to a leaf node with value '3' and it has an empty position. So, new element (4) can be inserted at that empty position.



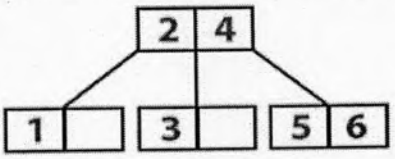
Insert(5)

Element '5' is larger than root node '2' and it is not a leaf node. So, we move to the right of '2'. We reach to a leaf node and it is already full. So, we split that node by sending middle value (4) to its parent node (2). There is an empty position in its parent node. So, value '4' is added to node with value '2' and new element '5' added as new leaf node.



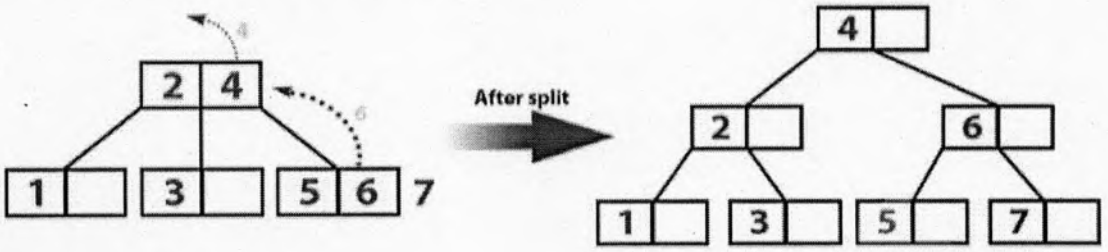
Insert(6)

Element '6' is larger than root node '2' & '4' and it is not a leaf node. So, we move to the right of '4'. We reach to a leaf node with value '5' and it has an empty position. So, new element (6) can be inserted at that empty position.



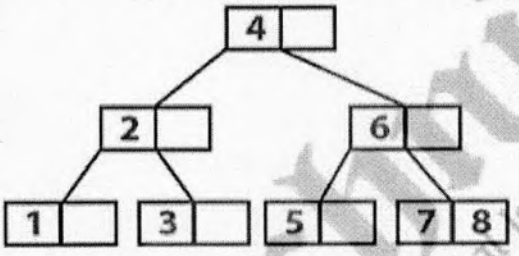
insert(7)

Element '7' is larger than root node '2' & '4' and it is not a leaf node. So, we move to the right of '4'. We reach to a leaf node and it is already full. So, we split that node by sending middle value (6) to its parent node (2&4). But the parent (2&4) is also full. So, again we split the node (2&4) by sending middle value '4' to its parent but this node doesn't have parent. So, the element '4' becomes new root node for the tree.



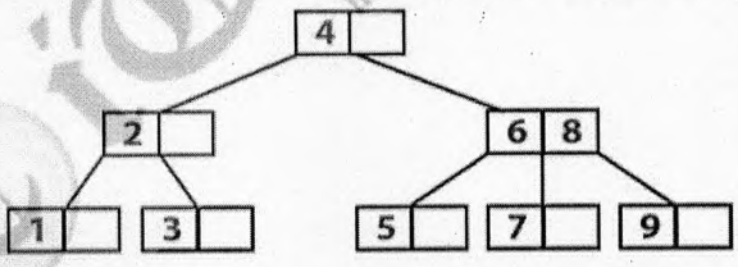
insert(8)

Element '8' is larger than root node '4' and it is not a leaf node. So, we move to the right of '4'. We reach to a node with value '6'. '8' is larger than '6' and it is also not a leaf node. So, we move to the right of '6'. We reach to a leaf node (7) and it has an empty position. So, new element (8) can be inserted at that empty position.



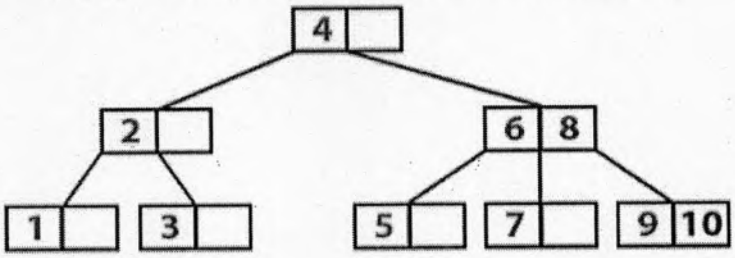
insert(9)

Element '9' is larger than root node '4' and it is not a leaf node. So, we move to the right of '4'. We reach to a node with value '6'. '9' is larger than '6' and it is also not a leaf node. So, we move to the right of '6'. We reach to a leaf node (7 & 8). This leaf node is already full. So, we split this node by sending middle value (8) to its parent node. The parent node (6) has an empty position. So, '8' is added at that position. And new element is added as a new leaf node.



insert(10)

Element '10' is larger than root node '4' and it is not a leaf node. So, we move to the right of '4'. We reach to a node with values '6 & 8 '. '10' is larger than '6 & 8' and it is also not a leaf node. So, we move to the right of '8'. We reach to a leaf node (9). This leaf node has an empty position. So, new element '10' is added at that empty position.



B+ Tree

- A B+ Tree is of order n.
- A B+ Tree contains index pages and data pages.
- The data pages always appear as leaf nodes in the tree.
- The root node and intermediate nodes are always index pages.
- Internal nodes can have upto n-1 keys and n pointers.

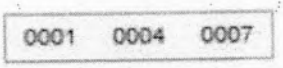
Example-

- Suppose we have a B+ tree with n=3. Then
  - It can have 2 keys max at each internal node.
  - It can have 3 pointers max at each internal node

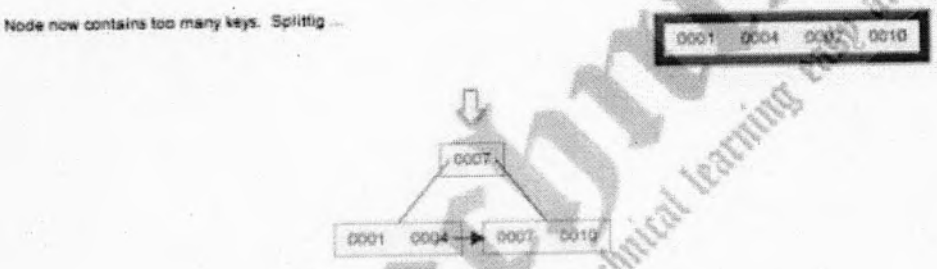
Example: Construct a B+ Tree for 1,4,7,10,17,21,31,25,19,20,28,42 with n=4 (Max no of keys in a node = 4-1 = 3).

Solution:

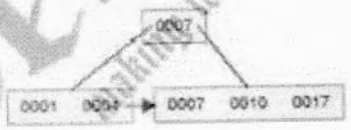
Insert 1,4,7



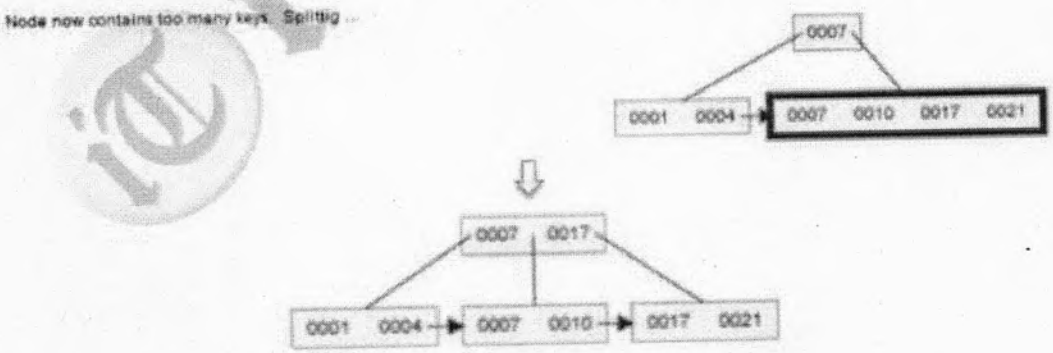
Insert 10



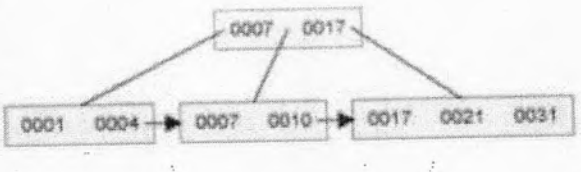
Insert 17



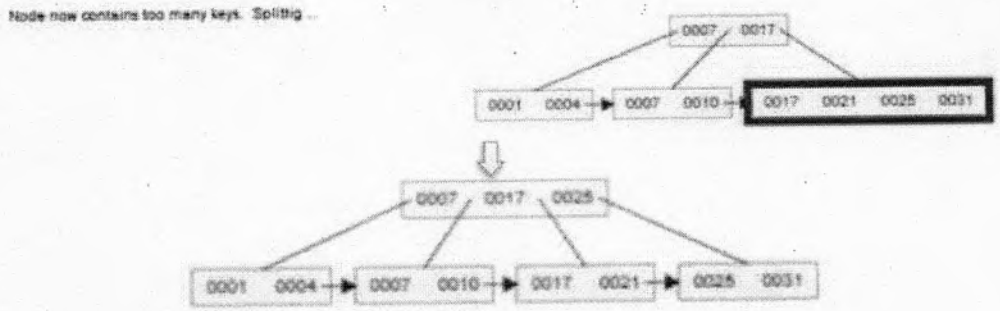
Insert 21



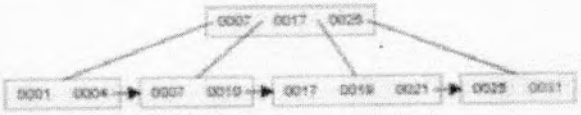
Insert 31



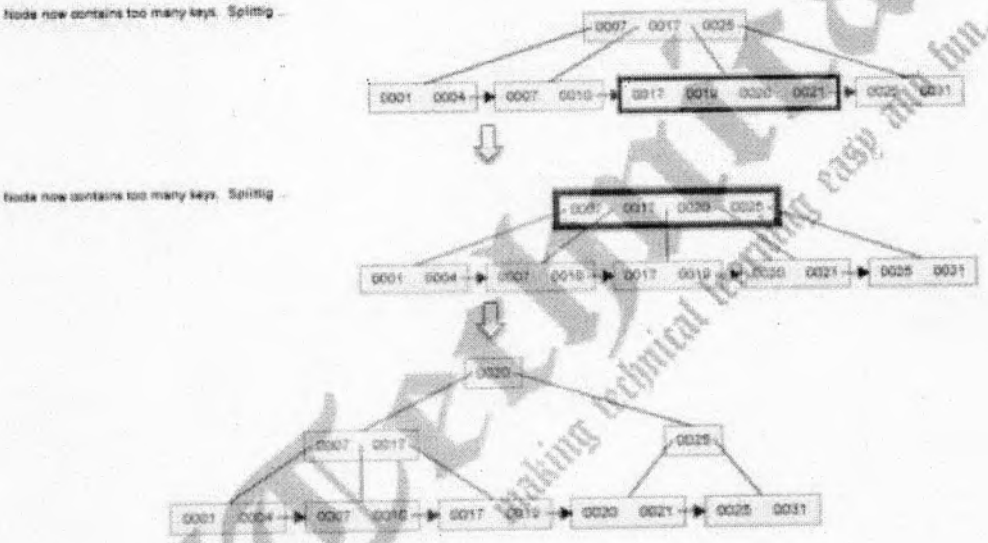
Insert 25



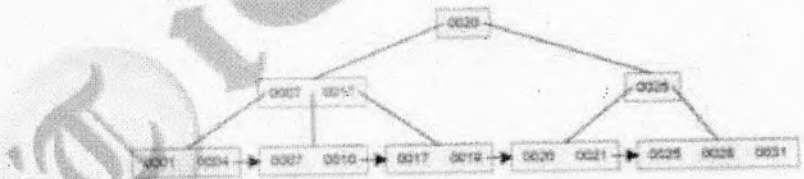
Insert 19



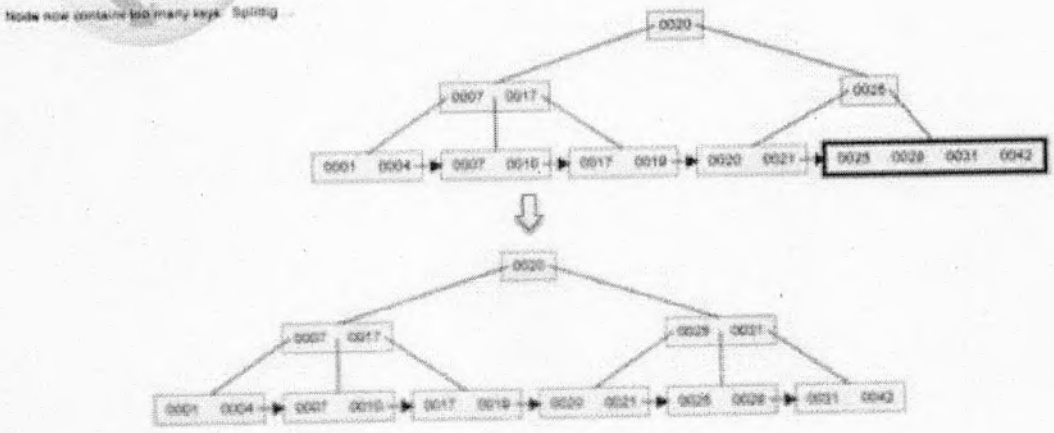
Insert 20



Insert 28

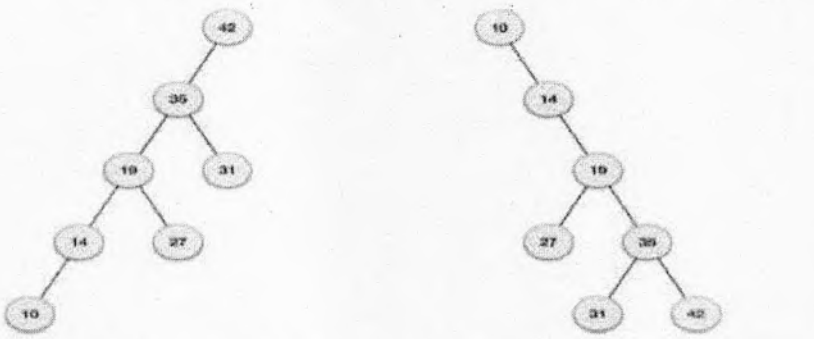


Insert 42



AVL Tree

What if the input to binary search tree comes in a sorted (ascending or descending) manner? It will then look like this –

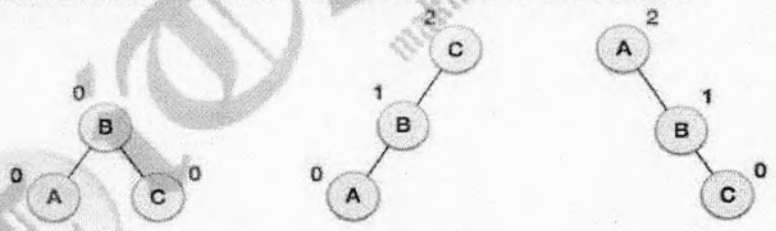


If input 'appears' non-increasing manner If input 'appears' in non-decreasing manner

It is observed that BST's worst-case performance is closest to linear search algorithms, that is O(n). In real-time data, we cannot predict data pattern and their frequencies. So, a need arises to balance out the existing BST.

- **AVL** named after their inventor **Adelson, Velski & Landis in 1962**, tree is a self-balanced binary search tree.
- That means, an AVL tree is also a binary search tree but it is a balanced tree.
- A binary tree is said to be balanced, if the difference between the heights of left and right subtrees of every node in the tree is either 0 or 1.
- In other words, a binary tree is said to be balanced if for every node, height of its children differs by at most one.
- In an AVL tree, every node maintains an extra information known as **Balance Factor**.
- AVL tree checks the height of the left and the right sub-trees and assures that the difference is not more than 1. This difference is called the **Balance Factor**.

Here we see that the first tree is balanced and the next two trees are not balanced –



In the second tree, the left subtree of **C** has height 2 and the right subtree has height 0, so the difference is 2. In the third tree, the right subtree of **A** has height 2 and the left is missing, so it is 0, and the difference is 2 again. AVL tree permits difference (balance factor) to be only 1.

BalanceFactor = Difference of [height(left-subtree) , height(right-subtree)]

If the difference in the height of left and right sub-trees is more than 1, the tree is balanced using some rotation techniques.

AVL Rotations

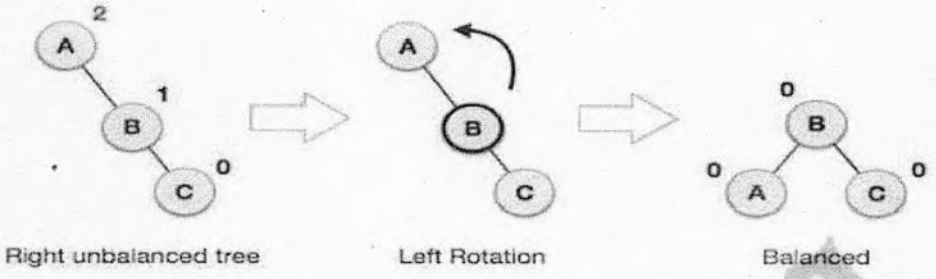
To balance itself, an AVL tree may perform the following four kinds of rotations –

- Left rotation
- Right rotation
- Left-Right rotation
- Right-Left rotation

The first two rotations are single rotations and the next two rotations are double rotations. To have an unbalanced tree, we at least need a tree of height 2. With this simple tree, let's understand them one by one.

Left Rotation

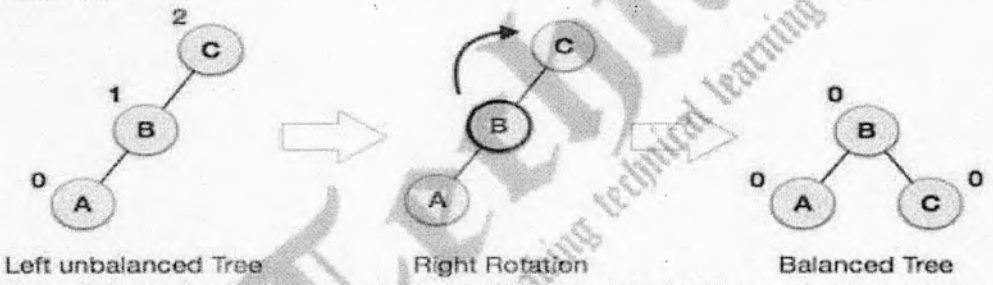
If a tree becomes unbalanced, when a node is inserted into the right subtree of the right subtree, then we perform a single left rotation –



In our example, node **A** has become unbalanced as a node is inserted in the right subtree of A's right subtree. We perform the left rotation by making **A** the left-subtree of B.

Right Rotation

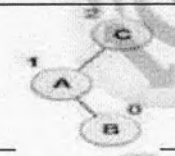
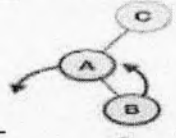
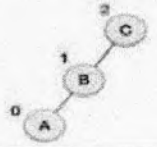
AVL tree may become unbalanced, if a node is inserted in the left subtree of the left subtree. The tree then needs a right rotation.

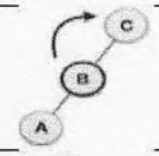
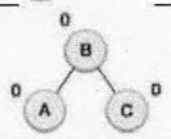


As depicted, the unbalanced node becomes the right child of its left child by performing a right rotation.

Left-Right Rotation

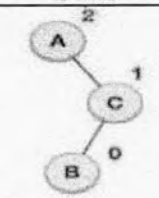
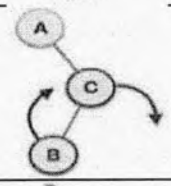
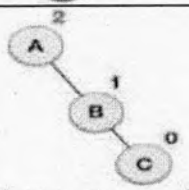
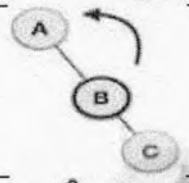
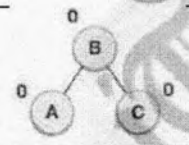
Double rotations are slightly complex version of already explained versions of rotations. To understand them better, we should take note of each action performed while rotation. Let's first check how to perform Left-Right rotation. A left-right rotation is a combination of left rotation followed by right rotation

State	Action
	A node has been inserted into the right subtree of the left subtree. This makes **C** an unbalanced node. These scenarios cause AVL tree to perform left-right rotation.
	We first perform the left rotation on the left subtree of **C**. This makes **A**, the left subtree of **B**.
	Node **C** is still unbalanced, however now, it is because of the left-subtree of the left-subtree.

	We shall now right-rotate the tree, making **B** the new root node of this subtree. **C** now becomes the right subtree of its own left subtree.
	The tree is now balanced.

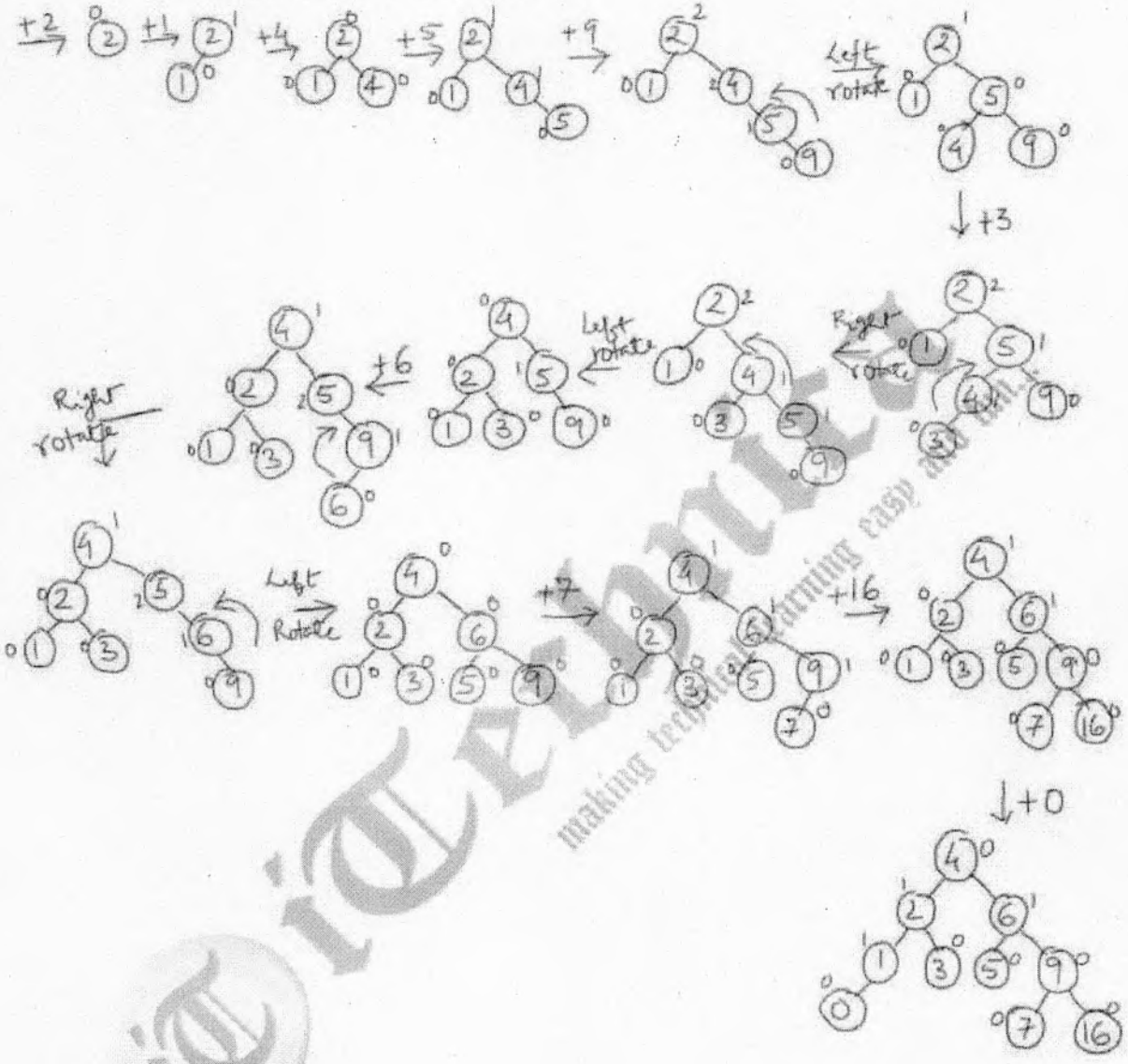
Right-Left Rotation

The second type of double rotation is Right-Left Rotation. It is a combination of right rotation followed by left rotation.

State	Action
	A node has been inserted into the left subtree of the right subtree. This makes **A**, an unbalanced node with balance factor 2.
	First, we perform the right rotation along **C** node, making **C** the right subtree of its own left subtree **B**. Now, **B** becomes the right subtree of **A**.
	Node **A** is still unbalanced because of the right subtree of its right subtree and requires a left rotation.
	A left rotation is performed by making **B** the new root node of the subtree. **A** becomes the left subtree of its right subtree **B**.
	The tree is now balanced.

Example:

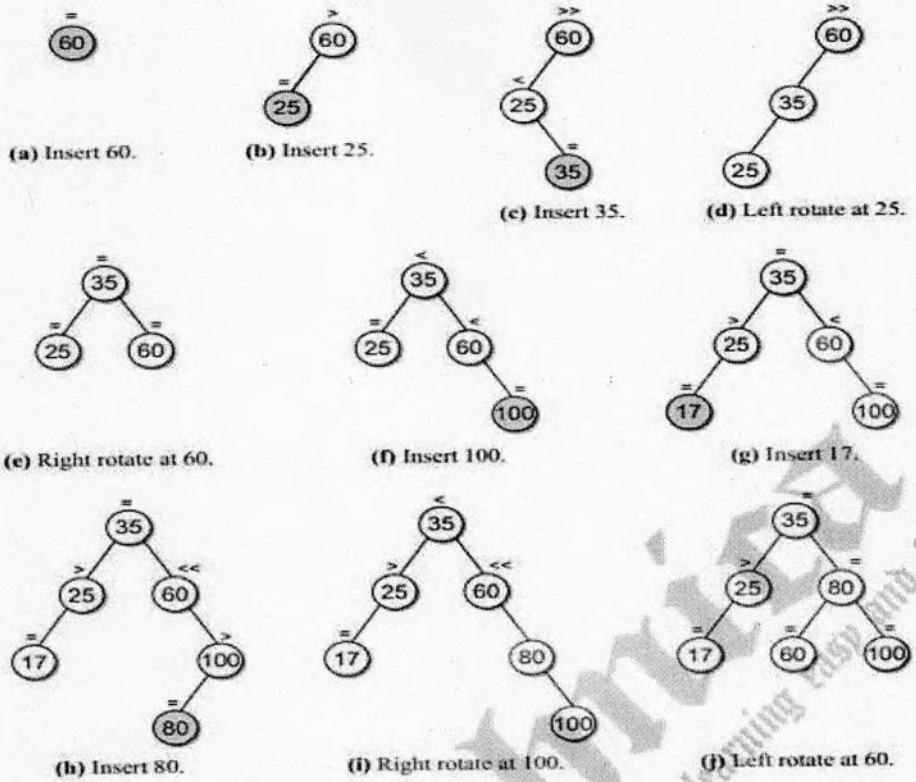
Que Construct an AVL Tree for the following sequence:-
2, 1, 4, 5, 9, 3, 6, 7, 16, 0



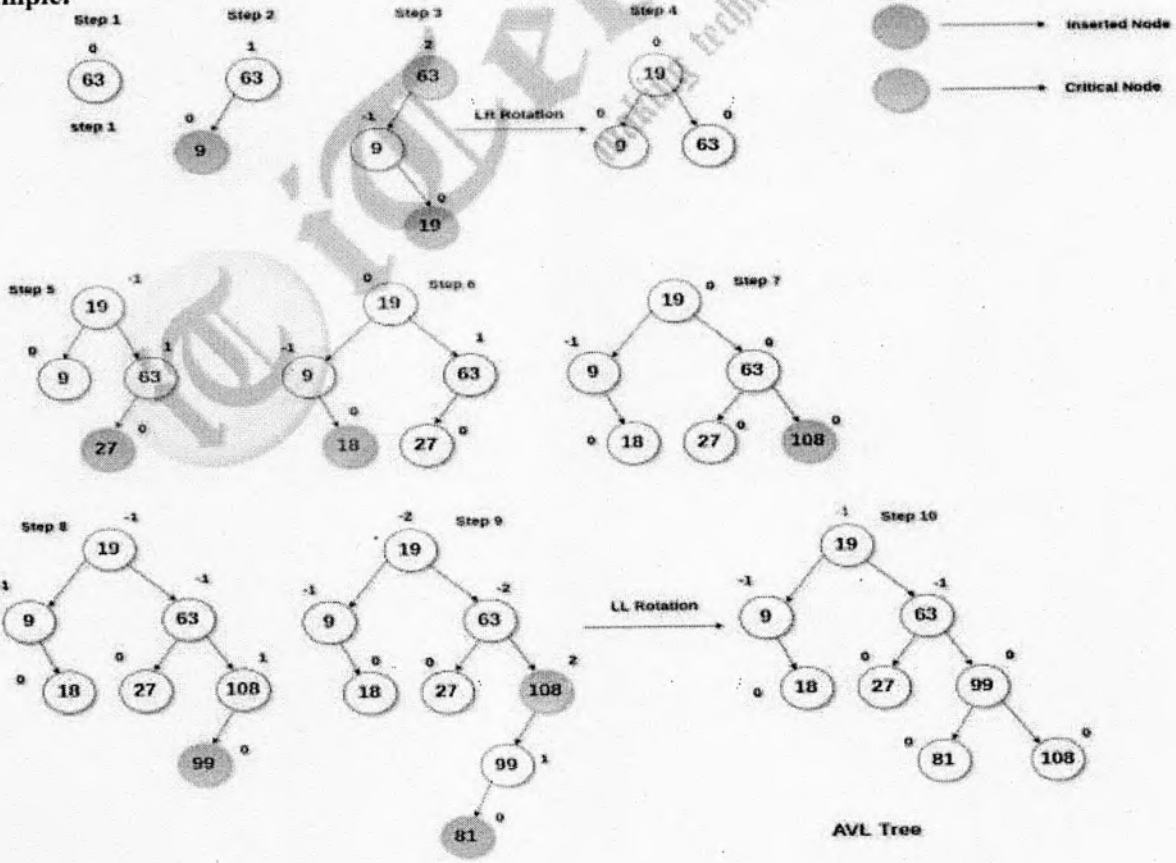
Example:

Construct an AVL Tree :- 50, 25, 10, 5, 7, 3, 30, 20, 8, 15.

Example:



Example:



Hashing

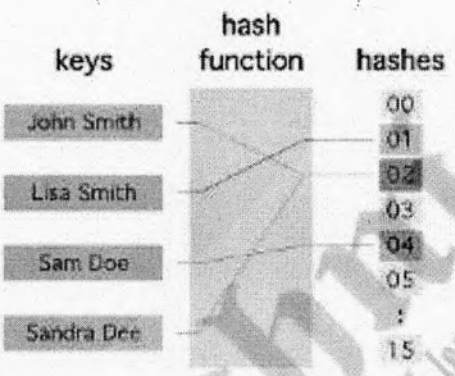
Hash Table:

It is a Data structure where the data elements are stored(inserted), searched, deleted based on the keys generated for each element, which is obtained from a hashing function. In a hashing system the keys are stored in an array which is called the Hash Table. A perfectly implemented hash table would always promise an average insert/delete/retrieval time of *O(1)*.

Hashing Function:

A function which employs some algorithm to computes the key *K* for all the data elements in the set *U*, such that the key *K* which is of a fixed size. The same key *K* can be used to map data to a hash table and all the operations like insertion,deletion and searching should be possible. The values returned by a **hash function** are also referred to as **hash** values, **hash** codes, **hash** sums, or **hashes**.

Hash Collision:

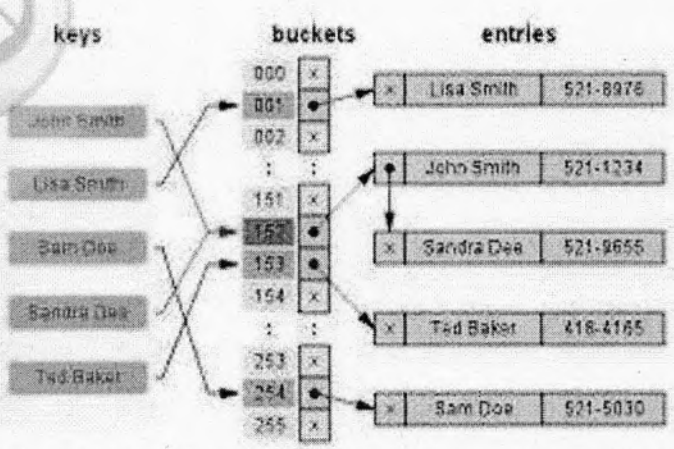


A situation when the resultant hashes for two or more data elements in the data set *U*, maps to the same location in the has table, is called a hash collision. In such a situation two or more data elements would qualify to be stored/mapped to the same location in the hash table.

Hash collision resolution techniques:

Open Hashing (Separate chaining)

Open Hashing, is a technique in which the data is not directly stored at the hash key index (*k*) of the Hash table. Rather the data at the key index (k) in the hash table is a pointer to the head of the data structure where the data is actually stored. In the most simple and common implementations the data structure adopted for storing the element is a linked-list.



In this technique when a data needs to be searched, it might become necessary (worst case) to traverse all the nodes in the linked list to retrieve the data.

Note that the order in which the data is stored in each of these linked lists (or other data structures) is completely based on implementation requirements. Some of the popular criteria are insertion order, frequency of access etc.

Closed hashing (open Addressing)
In this technique a hash table with pre-identified size is considered. All items are stored in the hash table itself. In addition to the data, each hash bucket also maintains the three states: EMPTY, OCCUPIED, DELETED. While inserting, if a collision occurs, alternative cells are tried until an empty bucket is found.

For which one of the following technique is adopted.

1. Liner Probing (this is prone to clustering of data + Some other constrains.)
2. Quadratic probing
3. Double hashing (in short in case of collision another hashing function is used with the key value as an input to identify where in the open addressing scheme the data should actually be stored.)

A comparative analysis of Closed Hashing vs Open Hashing

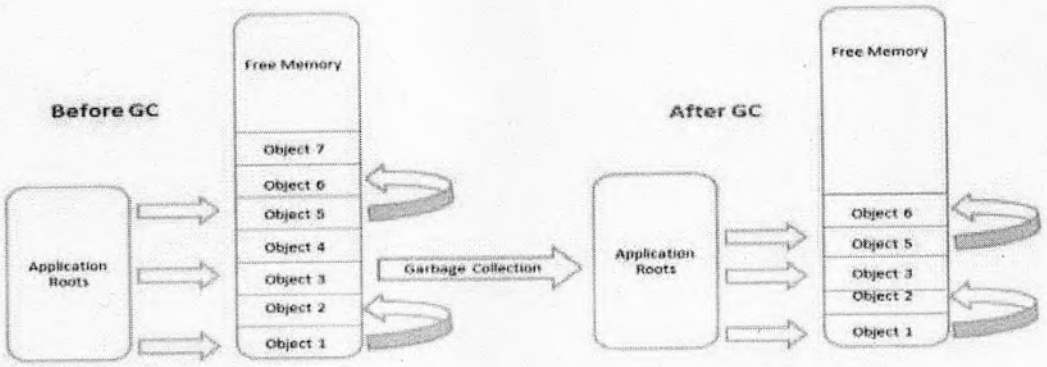
Open Addressing	Closed Addressing
All elements would be stored in the Hash table itself. No additional data structure is needed.	Additional Data structure needs to be used to accommodate collision data.
In cases of collisions, a unique hash key must be obtained.	Simple and effective approach to collision resolution. Key may or may not be unique.
Determining size of the hash table, adequate enough for storing all the data is difficult.	Performance deterioration of closed addressing much slower as compared to Open addressing.
State needs be maintained for the data (additional work)	No state data needs to be maintained (easier to maintain)
Uses space efficiently	Expensive on space

Storage Management

Garbage Collection:
Garbage collection (GC) is a dynamic approach to automatic memory management and heap allocation that processes and identifies dead memory blocks and reallocates storage for reuse. The primary purpose of garbage collection is to reduce memory leaks.

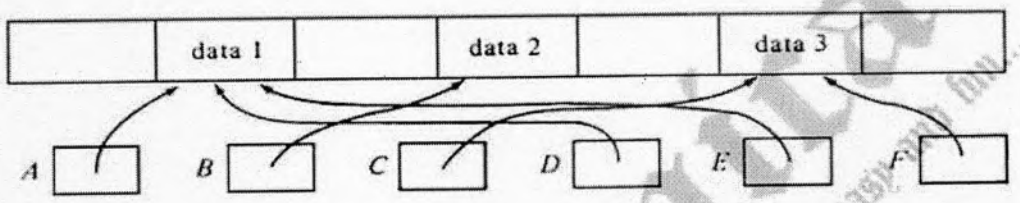
GC implementation requires three primary approaches, as follows:

- **Mark-and-sweep** - In process when memory runs out, the GC locates all accessible memory and then reclaims available memory.
- **Reference counting** - Allocated objects contain a reference count of the referencing number. When the memory count is zero, the object is garbage and is then destroyed. The freed memory returns to the memory heap.
- **Copy collection** - There are two memory partitions. If the first partition is full, the GC locates all accessible data structures and copies them to the second partition, compacting memory after GC process and allowing continuous free memory.

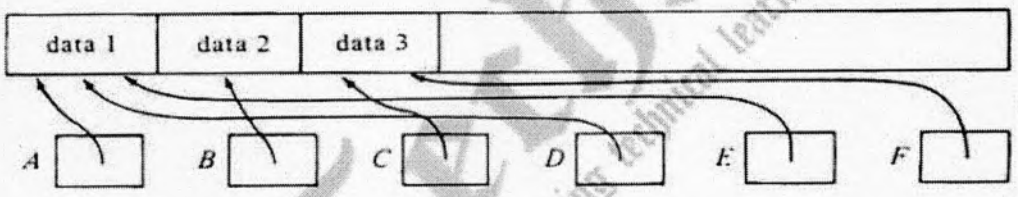


Compaction

The process of moving all marked nodes to one end of memory and all available memory to other end is called compaction. Algorithm which performs compaction is called compacting algorithm.



(a) Before compaction



(b) After compaction

ASSIGNMENTS

Unit-1
Introduction to Arrays and Linked Lists

Assignment - 01

1. Consider a two-dimensional array arr[10][10] which has base address = 1000 and the number of bytes per element of the array = 2. Now, compute the address of the element arr[8][5] assuming that the elements are stored in row-major and column-major order.
2. Consider an array MARKS [20][5] which stores the marks obtained by 20 students in 5 subjects. Now write a program to
   a. find the average marks obtained in each subject.
   b. find the average marks obtained by every student.
   c. find the number of students who have scored below 50 on their average.
   d. display the scores obtained by every student
3. Write a program that reads a square matrix of size n ¥ n. Write a function int isSymmetric (int a[][], int n) that returns 1 if the matrix is symmetric. (Hint: Array A is symmetric if Aij = A ji for all values of i and j)
4. Given a linked list that contains the English alphabet. The characters may be in upper case or in lower case. Create two linked lists—one which stores upper case characters and the other that stores lower case characters.
5. Write a program to count the number of occurrences of a given value in a linked list.
6. Write a program to create a linked list from an already given list. The new linked list must contain every alternate element of the existing linked list.
7. Write a program to reverse a linked list.
8. Write a program to delete the kth node from a linked list

Ajay Parashar
IT @ HCST

Unit-2
Stack and Queue

Assignment - 02

1. Convert the following infix expressions to their prefix and postfix equivalents:
   a. (A * B) + (C / D) – (D + E)
   b. ((A – B) + D / ((E + F) * G))
   c. (A – 2 * (B + C) / D * E) + F
   d. 14 / 7 * 3 – 4 + 9 / 2
2. Write a function that accepts two stacks. Copy the contents of first stack in the second stack. Note that the order of elements must be preserved. (Hint: use a temporary stack)
3. Differentiate between peek() and pop() functions. Why are parentheses not required in postfix/prefix expressions?
4. Consider the queue given below which has FRONT = 1 and REAR = 5.

	A	B	C	D	E				

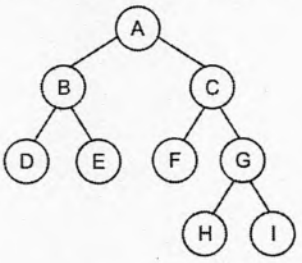
Now perform the following operations on the queue:
   a. Add F
   b. Delete two letters
   c. Add G
   d. Add H
   e. Delete four letters
   f. Add I
5. Write a program to input two queues and compare their contents.
6. Write a program to implement a circular queue with the help of array.
7. Write a program to create a stack from a queue.
8. The circular queue will be full only when
   (a) FRONT = MAX –1 and REAR = Max –1
   (b) FRONT = 0 and REAR = Max –1
   (c) FRONT = MAX –1 and REAR = 0
   (d) FRONT = 0 and REAR = 0

Ajay Parashar
IT @ HCST

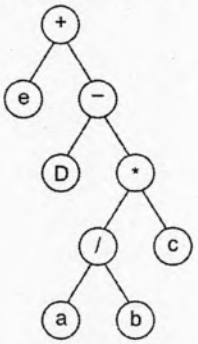
Unit-3
Trees

Assignment - 03

1. Find the in-order, pre-order, post-order, and level order traversal.

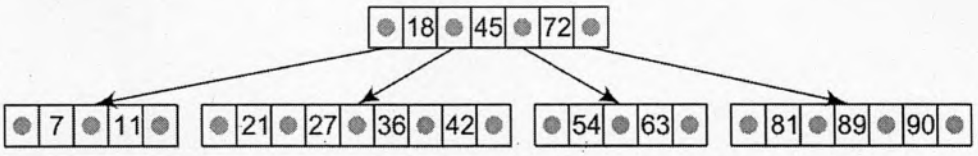


2. Evaluate the infix expression, given a = 30, b = 10, c = 2, d = 30, e = 10



3. Create a B tree of order 5 by inserting the following elements: 3, 14, 7, 1, 8, 5, 11, 17, 13, 6, 23, 12, 20, 26, 4, 16, 18, 24, 25, and 19.

4. Look at the B tree of order 5 given below and insert 8, 9, 39, and 4 into it.



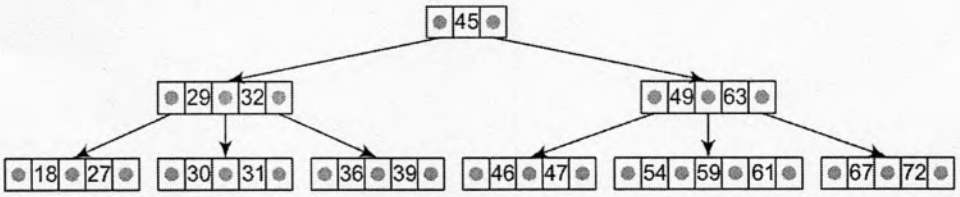
5. Create a B+ tree of order 5 for the following data arriving in sequence: 90, 27, 7, 9, 18, 21, 3, 4, 16, 11, 21, 72

6. Compare B trees with B+ trees

7. Consider the B tree given below

(a) Insert 1, 5, 7, 11, 13, 15, 17, and 19 in the tree.

(b) Delete 30, 59, and 67 from the tree.



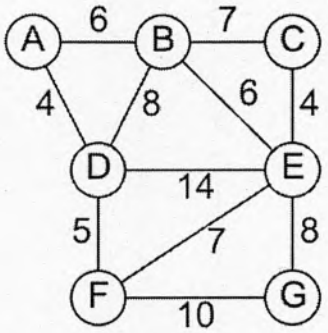
8. Construct an AVL tree by inserting the following elements in the given order. 63, 9, 19, 27, 18, 108, 99, 81.

Ajay Parashar
IT @ HCST

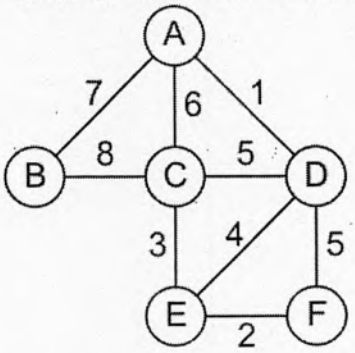
Unit-4
Graphs

Assignment - 04

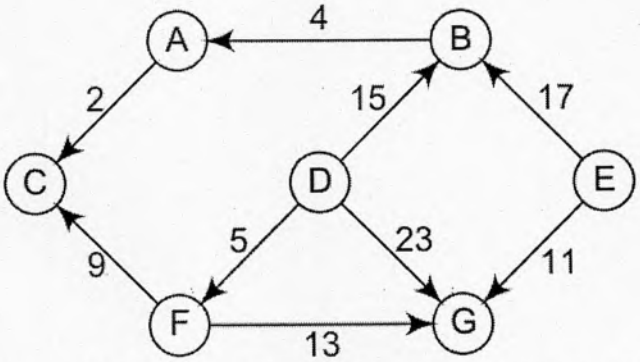
1. Construct a minimum spanning tree of the graph given in Fig. Start the Prim's algorithm from vertex D.



2. Apply Kruskal's algorithm on the graph given below to find the minimum spanning tree:

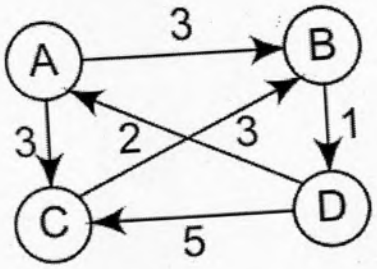


3. Consider the graph G given in Fig below, taking D as the initial node, execute the Dijkstra's algorithm on it.

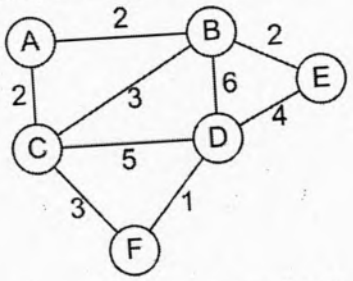


4. Consider a weighted graph G given in Fig. 13.42 and apply Warshall's shortest path algorithm to it.

Ajay Parashar
IT @ HCST



5. What are the applications of Graphs?
6. Consider the graph given below. Find the minimum spanning tree of this graph using (a) Prim's algorithm, (b) Kruskal's algorithm, and (c) Dijkstra's algorithm.



7. Given the following adjacency matrix, draw the weighted graph.

$$\begin{pmatrix} 0 & 4 & 0 & 2 & 0 \\ 0 & 0 & 0 & 7 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Ajay Parashar
IT @ HCST

Unit-5
Searching and Sorting

Assignment - 05

1. Which technique of searching an element in an array would you prefer to use and in which situation?
2. Define sorting. What is the importance of sorting?
3. What are the different types of sorting techniques?
4. Which sorting technique has the least worst case?
5. Explain the difference between bubble sort and quick sort. Which one is more efficient?
6. Sort the elements 77, 49, 25, 12, 9, 33, 56, 81 using
   (a) insertion sort (b) selection sort
   (c) bubble sort (d) merge sort
   (e) quick sort
7. Sort the following sequence of numbers in descending order using heap sort. 42, 34, 75, 23, 21, 18, 90, 67, 78

Ajay Parashar
IT @ HCST

TUTORIAL SHEETS

Unit-1
Introduction to Arrays and Linked Lists

Tutorial - 01

1. What are arrays and why are they needed? How is an array represented in the memory? For an array declared as int arr[50], calculate the address of arr[35], if Base(arr)=1000 and w=2.
2. Consider a two-dimensional array Marks[10][5] having its base address as 2000 and the number of bytes per element of the array is 2. Now, compute the address of the element, Marks[8][5], assuming that the elements are stored in row-major order and column-major order.
3. Why does storing sparse matrices need extra consideration? How are sparse matrices stored efficiently in the computer's memory?
4. Consider the array given below:

Name[0]	Adam
Name[1]	Charles
Name[2]	Dicken
Name[3]	Esha
Name[4]	Georgia
Name[5]	Hillary
Name[6]	Mishael

a. How many elements would be moved if the name Andrew has to be added in it?
(i) 7 (ii) 4 (iii) 5 (iv) 6

b. How many elements would be moved if the name Esha has to be deleted from it?
(i) 3 (ii) 4 (iii) 5 (iv) 6

Ajay Parashar
IT @ HCST

Unit-1
Introduction to Arrays and Linked Lists

Tutorial - 02

1. Make a comparison between a linked list and a linear array. Which one will you prefer to use and when?
2. Why is a doubly linked list more useful than a singly linked list? Give the advantages and uses of a circular linked list.
3. Give the linked representation of the following polynomial:

$$7x^3y^2 - 8x^2y + 3xy + 11x - 4$$

4. Explain the difference between a circular linked list and a singly linked list. Form a linked list to store students' details.
   a. Use the linked list of the above question to insert the record of a new student in the list.
   b. Delete the record of a student with a specified roll number from the list maintained.

Ajay Parashar
IT @ HCST

Unit-2
Stack and Queue

Tutorial - 01

1. What do you understand by stack overflow and underflow? How does a stack implement using a linked list differ from a stack implemented using an array?
2. Convert the following infix expressions to their postfix equivalents:
   a. A – B + C
   b. A * B + C / D
   c. (A – B) + C * D / E – C
3. Convert the following infix expressions to their postfix equivalents:
   a. A – B + C
   b. A * B + C / D
   c. (A – B) + C * D / E – C
4. Evaluate a postfix expression: 10 + ((7 – 5) + 10)/2

Ajay Parashar
IT @ HCST

Unit-2
Stack and Queue

Tutorial - 02

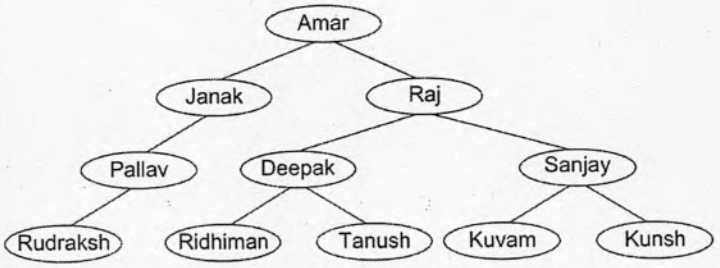
1. Draw the queue structure in each case when the following operations are performed on an empty queue.
    a. Add A, B, C, D, E, F
    b. Delete two letters
    c. Add G
    d. Add H
    e. Delete four letters
    f. Add I
2. What is a priority queue? Give its applications.
3. Explain the concept of a circular queue with an example? How is it better than a linear queue?
4. Write a program to create a linear queue of 10 values using linked lists.

Ajay Parashar
IT @ HCST

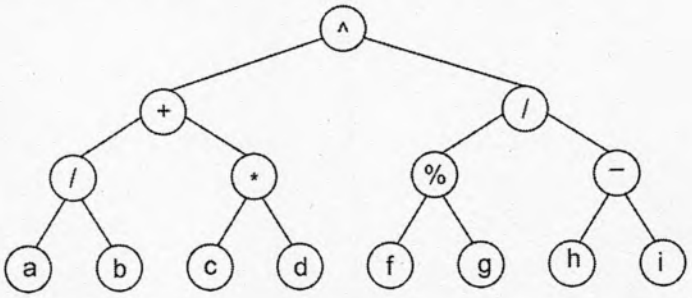
Unit-3
Trees

Tutorial - 01

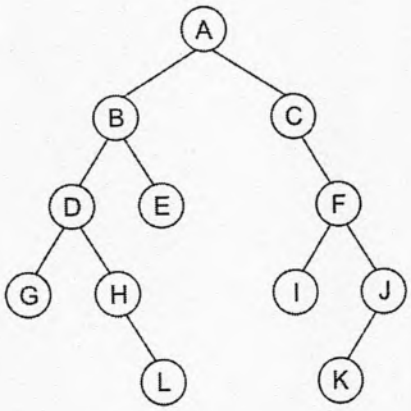
1. Give the memory representation of the following tree using arrays:



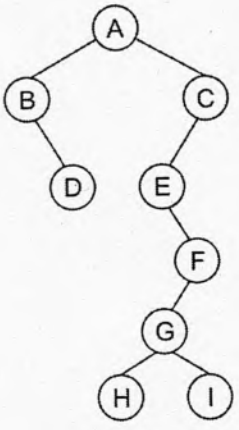
2. Given an expression, Exp = ((a + b) - (c * d)) % ((e ^f) / (g - h)), construct the corresponding binary tree.
3. Given the binary tree, write down the expression that it represents.



4. In Figs (a) and (b), find the sequence of nodes that will be visited using pre-order traversal algorithm.



(a)



(b)

Ajay Parashar
IT @ HCST

Unit-3
Trees

Tutorial - 02

1. Consider the traversal results given below, and construct a binary tree
   In–order Traversal: D B E A F C G
   Pre-order Traversal: A B D E C F G
2. Consider the traversal results given below, and construct a binary tree
   In–order Traversal: D B H E I A F J C G
   Post-order Traversal: D H I E B J F G C A
3. Create a Huffman tree with the following nodes arranged in a priority queue and the optimal Huffman codes for the alphabets.

A	B	C	D	E	F	G	H	I	J
7	9	11	14	18	21	27	29	35	40

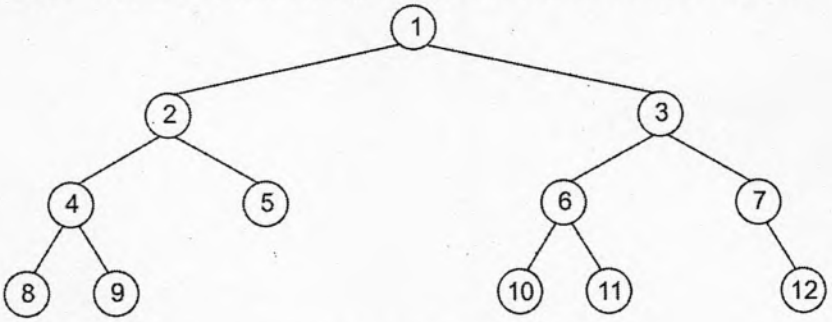
4. What are the applications of Trees?

Ajay Parashar
IT @ HCST

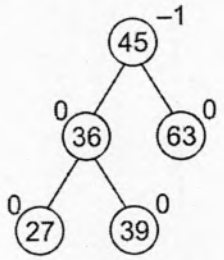
Unit-3
Trees

Tutorial - 03

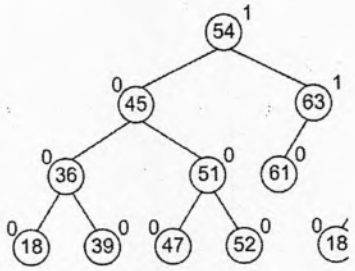
1. Create a binary search tree using the following data elements: 45, 39, 56, 12, 34, 78, 32, 10, 89, 54, 67, 81
2. Consider the below tree and create its equivalent double threaded binary tree



3. Consider the AVL tree given in Fig. below and insert 18 into it:



4. Delete nodes 52, 36, and 61 from the AVL tree given below:

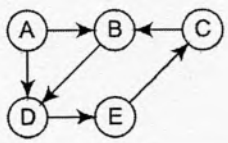
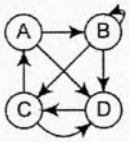
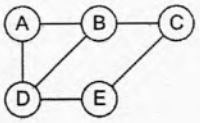
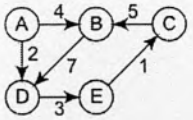


Ajay Parashar
IT @ HCST

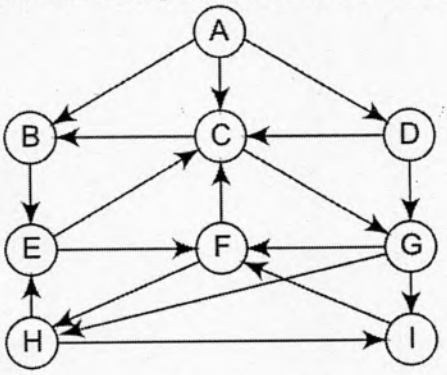
Unit-4
Graphs

Tutorial - 01

1. Consider the below graphs and generate their corresponding adjacency matrices

			
A	B	C	D

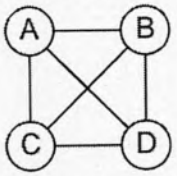
2. Consider the graph G given in Figure below. The adjacency list of G is also given. Suppose we want to print all the nodes that can be reached from the node H (including H itself). One alternative is to use a depth-first search of G starting at node H. Explain the procedure



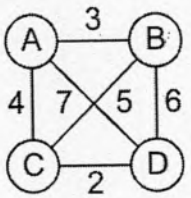
Adjacency lists

A: B, C, D
B: E
C: B, G
D: C, G
E: C, F
F: C, H
G: F, H, I
H: E, I
I: F

3. Consider an unweighted graph G given below, and find out it's possible spanning trees.



4. Consider a weighted graph G given below, find out its minimum spanning tree.

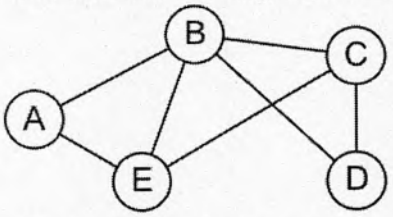


Ajay Parashar
IT @ HCST

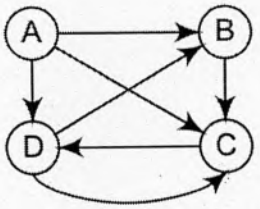
Unit-4
Graphs

Tutorial - 02

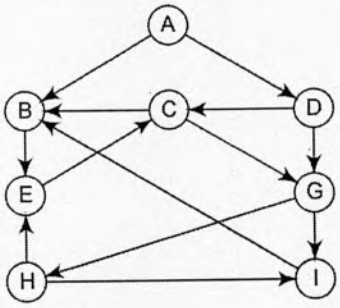
1. Consider the graph given below and find out the degree of each node.



2. Consider the graph given below. State all the simple paths from A to D, B to D, and C to D. Also, find out the in-degree and out-degree of each node. Is there any source or sink in the Graph?



3. When is a spanning tree called a minimum spanning tree? Take a weighted graph of your choice and find out its minimum spanning tree.
4. Consider the graph given below. Find out its depth-first and breadth-first traversal scheme.



Ajay Parashar
IT @ HCST

Unit-5
Searching and Sorting

Tutorial - 01

1. Sort the following array A[] that has the following elements: A[] = {30, 52, 29, 87, 63, 27, 19, 54}
2. Consider an array of integers given below. Sort the values in the array using insertion sort.

39	9	45	63	18	81	108	54	72	36

3. List the advantages of Insertion sort.
4. Sort the array given below using selection sort.

39	9	81	45	90	27	72	18

Ajay Parashar
IT @ HCST

Unit-5
Searching and Sorting

Tutorial - 02

1. Sort the array given below using merge sort.

39	9	81	45	90	27	72	18

2. Explain the procedure of quick sort and sort the following:

27	10	36	18	25	45

3. Sort the array given below using heap sort.

39	9	81	45	90	27	72	18

Ajay Parashar
IT @ HCST

RESULTS

(LAST 03 YEARS)

PARFORMA-1

Result Analysis For ODD Semester Examination Session 2021-22
B.Tech. III Semester

Branch - INFORMATION TECHNOLOGY

S.N.	Roll No.	Group	Name of the Student	Total Marks 950	%	3rd semester SGPA	End Semester Exam Theory Marks						Please Tick (√) for Carry Paper							Fail/Pass Status in odd sem Result	Odd Sem Internal Back with Subject Code
							Maths IV	UHV	DS	COA	DSTL	CSS	CP	CP	CP	CP	CP	CP	CP		
							KAS302	KVE301	KCS301	KCS302	KCS303	KNC301									
							100	100	100	100	100	50	1	2	3	4	5	6	0		
1	2000640130001		ABHAY GOYAL	462	48.63	4.6	36	46	49	12	39	35	√							COP	KCS301, KCS303
2	2000640130003		AMAN AHMED	631	66.42	7.2	41	43	54	48	47	19							√	PASS	
3	2000640130004		AYUSH MITTAL	439	46.21	3.5	12	43	45	15	39	26		√						COP	KCS301, KCS303
4	2000640130005		PANKAJ SAKHRANI	489	51.47	5.4	30	31	50	35	51	21							√	PASS	KCS301, KCS303
5	2000640130006		PIYUSHA VARSHNEY	731	76.95	8.0	57	64	66	39	64	32							√	PASS	
6	2000640130007		SANDEEP SINGH	452	47.58	3.7	17	32	33	31	24	27		√						COP	KAS302
7	2000640130008		TUSHANT SHASTRI	506	53.26	5.1	47	44	31	0	48	27	√							COP	
8	2000640130009		UDIT UPADHYAY	415	43.68	3.2	13	35	36	19	30	39		√						COP	KAS302, KCS301
9	2000640130010		UMESH SINGH	648	68.21	7.1	57	59	64	32	57	32							√	PASS	
10	2000640130011		VANSH UPADHYAY	405	42.63	2.9	30	46	12	16	10	27			√					COP	
GRAND TOTAL							340	443	440	247	409	285	2	3	1				4		
AVERAGE SCORE IN SUBJECTS:							34.0	44.3	44.0	24.7	40.9	57.0									
NO. OF STUDENTS PASS IN SUBJECTS (SCORE MORE THAN 30/100):							7	10	9	5	8	10									
NO. OF CARRY PAPERS IN THE SUBJECT (SCORE LESS THAN 30/100):							3	0	1	5	2	0									
PERCENTAGE OF PASS STUDENTS IN THE SUBJECT:							70.00	100.00	90.00	50.00	80.00	100.00									
PERCENTAGE OF FAIL STUDENTS IN THE SUBJECT:							30.00	0.00	10.00	50.00	20.00	0.00									

NOTE: * SUM UP THE RESULT IN SHADED AREA.

* STUDENTS SCORED MORE THAN 30/100 MARKS BUT FAILS BECAUSE OF POOR SESSIONAL MARKS SHOULD ALSO BE CONSIDERED AS CARRY PAPERS AND INCLUDED IN THIS CATEGORY.

Code No.	Subject's Name	Teacher's Name
KAS302	MATHS IV	DR. M P SINGH
KVE301	UHV	DR. ARCHANA GAUTAM
KCS301	DS	MR. AJAY PARASHAR
KCS302	COA	MRS. RUPALI MAHAJAN
KCS303	DSTL	MRS. DEEPTI MITTAL
KNC301	CSS	MR. SHIV PRATAP

Compiled by Teacher:
MR. AJAY PARASHAR

Checked by Teacher:
MRS. DEEPTI MITTAL

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:24:26+05'30'
Foxit PhantomPDF Version: 10.1.1

Head
Department of Information Technology
HOD Hindustan College of Science & Technology
DR. SHANKAR THAWKAR

PARFORMA-2

Result Analysis for ODD Semester Examination Session 2021-22

B. Tech III Semester

Branch: INFORMATION TECHNOLOGY

Semester /Branch	Total No. of Students (Enrolled)	Total No. of Students Detained	Total No. of Students Appeared	Result Declared	Result Incomplete	No. of Students Pass CP (0)	Total No. of Students with CP status	Subject Name with Subject Code	Sec.	Subject Wise Faculty Performance							
										Total No. of Students Result Declared	No. of Pass Students	% of Pass Students	Total Marks in Subject	% Avg. Marks	Highest Score	No. of Students score more than 75%	Name of the Subject Teacher
III SEM	10	NIL	10	10	NIL	4	6	KAS302 (MATHS-IV)	A	10	7	70	340	34.00	57	0	DR. M P SINGH
								KVE301 (UHV)	A	10	10	100	443	44.30	64	0	DR. ARCHANA GAUTAM
								KCS301 (DS)	A	10	9	90	440	44.00	66	0	MR. AJAY PARASHAR
								KCS302 (COA)	A	10	5	50	247	24.70	48	0	MRS. RUPALI MAHAJAN
								KCS303 (DSTL)	A	10	8	80	409	40.90	64	0	MRS. DEEPTI MITTAL
								KNC301 (CSS)	A	10	10	100	570	57.00	78	1	MR. SHIV PRATAP

A. Carry Paper Detail:

No. of Carry Papers	CP (1)	CP (2)	CP(3)	CP(4)	CP(5)	CP(6)	CP(0)	Aggregate Marks (Overall Marks)		
								75% and above	Between 60-75%	Less than 60%
No. of Students	2	3	1				4	1	2	7
Percentage	20	30	10	0	0	0	40	10	20	70

B. Top Five Rankers in the Branch: (On the basis of SGPA of 3rd Semester)

S.N.	University Roll. No.	Name of the student	Marks Obtained (950)	Aggregate Percentage of Marks Obtained	SGPA	Remark
1	2000640130006	PIYUSHA VARSHNEY	731	76.9	8	
2	2000640130003	AMAN AHMED	631	66.4	7.2	
3	2000640130010	UMESH SINGH	648	68.2	7.1	

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:24:14+05'30'
Foxit PhantomPDF Version: 10.1.1

Compiled by Teacher:
(Name & Dated Signature)

Checked by Teacher:
(Name & Dated Signature)

HOD
(Name & Dated Signature)

Head
Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

Result Analysis For ODD Semester Examination Session 2020-21

B.Tech. III Semester

Branch – Information Technology

S.N.	Roll No.	Name of the Student	Total Marks (950)	% Total Marks	End Semester Exam Theory Marks in Odd Sem						Please Tick (√) for Carry Paper in Odd Sem					Status of student in Result (Fail/Pass/RND/INC/PWG/PCP/PCP-A)	No. of Odd Sem Back Paper with Subject Code	Internal Back
					Name of Subjects & Codes						CP	CP	CP	CP	CP			
					MATHS IV (100)	TC (100)	DS (100)	COA (100)	DSTL (100)	CSS (50)								
					KAS302	KAS301	KCS301	KCS302	KCS303	KNC301	1	2	3	4	0			
1	1900640130001	AMAN KUMAR SONI	618	65.05%	37	43	41	30	40	9					√			
2	1900640130002	HIMANSHU MOOLCHANDANI	644	67.79%	49	53	30	55	50	24					√			
3	1900640130004	KUNAL	569	59.89%	39	36	37	34	41	35					√			
4	1900640130005	SEJAL JAIN	715	75.26%	70	60	48	62	37	35					√			
5	1900640130006	URAVASHI VERMA	654	68.84%	37	35	40	57	66	17					√			
6	1900640130007	VIPLAV KANT RAI	582	61.26%	18	48	70	8	75	29		√				CP(2)		
7	1900640130008	VIVEK SHARMA	611	64.32%	42	33	54	44	58	16					√			
GRAND TOTAL					292	308	320	290	367	165								
AVERAGE SCORE IN SUBJECTS:					41.71	44.00	45.71	41.43	52.43	47.14								
NO. OF STUDENTS PASS IN SUBJECTS (SCORE MORE THAN 30/100 or 15/50):					6	7	7	6	7	6								
NO. OF CARRY PAPERS IN THE SUBJECT (SCORE LESS THAN 30/100 or 15/50):					1	0	0	1	0	1								
PERCENTAGE OF PASS STUDENTS IN THE SUBJECT:					85.71	100.00	100.00	85.71	100.00	85.71								
PERCENTAGE OF FAIL STUDENTS IN THE SUBJECT:					14.28	0.00	0.00	14.28	0.00	14.28								
NO. OF STUDENTS WITH CARRY PAPERS [Σ(CP)]:											0	1	0	0	6			

NOTE: * SUM UP THE RESULT IN SHADED AREA.

*** STUDENTS SCORED MORE THAN 30/100 or 15/50 MARKS BUT FAILS BECAUSE OF POOR SESSIONAL MARKS SHOULD ALSO BE CONSIDERED AS CARRY PAPERS AND INCLUDED IN THIS CATEGORY.**

Code No.	Subjects Name
KAS302	MATHS IV
KAS301	TECHNICAL COMMUNICATION
KCS301	DATA STRUCTURE
KCS302	COMPUTER ORGANISATION & ARCHITECTURE
KCS303	DISCRETE STRUCTURE & THEORY OF LOGIC
KNC301	COMPUTER SYSTEM SECURITY

Compiled by Teacher:
(Name & Dated Signature)

Checked by Teacher: MRS. DEEPTI MITTAL

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:24:02+05'30'
Foxit PhantomPDF Version: 10.1.1

HOD
(Name & Dated Signature)

Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

PARFORMA-2

Result Analysis for Odd Semester Examination Session 2020-21

B. Tech III Semester

Branch: Information Technology

Semester /Branch	Total No. of Students (Enrolled)	Total No. of Students Detained	Total No. of Students Appeared	Result Declared	Result Incomplete	No. of Students Pass CP (0)	Total No. of Students with CP status	No. of Students Fail	Subject Name & Code	Sec.	Subject Wise Faculty Performance									
											Total No. of Students Result Declared	No. of Pass Students	% of Pass Students	% of Pass Students in Sec. (A)	Total Marks in Subject	% Avg. Marks	% Avg. Marks in Sec. (A)	Highest Score	No. of Students score more than 75%	Name of the Subject Teacher
III	7	0	7	7	0	6	1	0	MATHSIV (KAS302)	A	7	6	85.71	85.71	292	41.71	41.71	70	0	DR. M.P SINGH
									TC (KAS301)	A	7	7	100.00	100.00	308	44.00	44.00	60	0	DR.KESHAV DEV
									DS(KCS301)	A	7	7	100.00	100.00	320	45.71	45.71	70	0	MR.AJAY PARASHAR
									COA (KCS302)	A	7	6	85.71	85.71	290	41.43	41.43	62	0	MRS. RUPALI MAHAJAN
									DSTL (KCS303)	A	7	7	100.00	100.00	367	52.43	52.43	75	0	MRS. SUMITA LAMBA
									CSS (KNC301)	A	7	6	85.71	85.71	165	47.14	47.14	35	0	MRS. BHUPRABHA BHARTI

A. Carry Paper Detail:

No. of Carry Papers	Odd Sem	Odd Sem	Odd Sem	Odd Sem	Odd Sem	Odd Sem
	CP (1)	CP (2)	CP(3)	CP(4)	CP(5)	CP(6)
No. of Students	0	1	0	0	0	0
Percentage		14.3				

Aggregate Marks		
75% and above	Between 60-75%	Less than 60%
1	5	1
14.28	71.40	14.28

B. Top Five Rankers in the Branch:

S.N.	University Roll. No.	Name of the student	Marks Obtained (2000)	Aggregate Percentage of Marks Obtained	Remark

B. Top Five Rankers in the Branch:

S.N.	University Roll. No.	Name of the student	Marks Obtained (950)	Aggregate Percentage of Marks	Remark
1	1900640130005	SEJAL JAIN	715	75.26%	
2	1900640130006	URAVASHI VERMA	654	68.84%	
3	1900640130002	HIMANSHU MOOLCHANDANI	644	67.79%	
4	1900640130001	AMAN KUMAR SONI	618	65.05%	
5	1900640130008	VIVEK SHARMA	611	64.32%	

Compiled by Teacher:
(Name & Dated Signature)

Checked by Teacher:
(Name & Dated Signature)

Head
Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:23:47+05'30'
Foxit PhantomPDF Version: 10.1.1

Result Analysis For Even Semester Examination Session 2019-20

B.Tech. III Semester

Branch – IT

S.N.	Roll No.	Group	Name of the Student	Total Marks 1000	% Total Marks	SGPA	End Semester Exam Theory Marks in Even Sem						Please Tick (√) for Carry Paper in Even Sem					Status of student in Result (Fail/Pass/RND /INC/PWG/PC P/PCP-A)	No. of Odd Sem Back Paper with Subject Code	Internal Back
							MATH-IV	TC	DS	COA	DSTL	CSS	CP	CP	CP	CP	CP			
							KAS302	KAS301	KCS301	KCS302	KCS303	KNC301								
							100	100	100	100	100	50	1	2	3	4	0			
1	1806413001	IT-A1	ADARSH KUMAR JADON	651	68.526	7.23	50	53	58	40	49	20						PASS		
2	1806413002	IT-A1	AMAN NAGAR	471	49.579	4.5	16	51	44	36	35	18	√					PCP	math	
3	1806413004	IT-A1	Aparna Kulshrestha	733	77.158	8.23	66	67	60	56	40	25						PASS		
4	1806413005	IT-A1	ARIF ALAM	671	70.632	7.55	51	57	53	66	33	27						PASS		
5	1806413006	IT-A1	ASEEM MITHWANI	604	63.579	6.82	51	55	67	32	39	27						PASS		
6	1806413007	IT-A1	AYUSHI GUPTA	577	60.737	6.18	40	47	38	41	37	26						PASS		
7	1806413008	IT-A1	BAHART SINGH	531	55.895	4.55	23	40	17	69	30	16		√				PCP	math, ds	
8	1806413010	IT-A1	DIVYANSH JOHARI	559	58.842	5.73	31	48	56	46	18	17	√					PCP	dstl	
9	1806413012	IT-A1	HARSH AGARWAL	587	61.789	6.73	40	38	52	41	42	22						PASS		
10	1806413015	IT-A1	NITIN SINGH RANA	424	44.632	2.36	19	58	35	16	13	15				√		PCP	math, ds,coa, dstl	
11	1806413018	IT-A1	PRANSHI CHAURASIA	614	64.632	6.73	54	58	41	36	32	20						PASS		
12	1806413019	IT-A1	PRIYANSHU SAXENA	476	50.105	4.5	36	54	38	39	30	18	√					PCP	dstl	
13	1806413020	IT-A1	RAJAT SHARMA	752	79.158	8.59	74	70	59	74	44	34						PASS		
14	1806413021	IT-A1	ROHIT PATHAK	690	72.632	7.73	48	71	71	53	46	43						PASS		
15	1806413022	IT-A1	SATYAM SINGH RATHORE	670	70.526	7.55	66	43	51	40	45	20						PASS		
16	1806413023	IT-A1	SAURABH SHUKLA	702	73.895	7.77	72	53	45	54	39	22						PASS		
17	1806413024	IT-A1	SHARAD SHAKYA	620	65.263	7.23	48	58	46	58	37	25						PASS		
18	1806413025	IT-A1	SHIVAM CHOUDHARY	550	57.895	6.32	45	50	46	55	30	20						PASS		
19	1806413026	IT-A1	SHIVAM CHAHAR	690	72.632	7.86	93	48	63	47	36	24						PASS		
20	1806413027	IT-A1	YASH MANGHNANI	517	54.421	6.36	34	44	51	38	8	17	√					PCP	dstl	
GRAND TOTAL							957	1063	991	937	683	506								
AVERAGE SCORE IN SUBJECTS:							47.85	53.15	49.55	46.85	34.15	50.60								
NO. OF STUDENTS PASS IN SUBJECTS (SCORE MORE THAN							17	20	18	19	16	20								
NO. OF CARRY PAPERS IN THE SUBJECT (SCORE LESS THAN							3	0	2	1	4	0								
PERCENTAGE OF PASS STUDENTS IN THE SUBJECT:							85.00	100.00	90.00	95.00	80.00	100.00								
PERCENTAGE OF FAIL STUDENTS IN THE SUBJECT:							15.00	0.00	10.00	5.00	20.00	0.00								
NO. OF STUDENTS WITH CARRY PAPERS [Σ(CP)]:													4	1		1				

NOTE: * SUM UP THE RESULT IN SHADED AREA.

BE CONSIDERED AS CARRY PAPERS AND INCLUDED IN THIS CATEGORY.

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:23:32+05'30'
Foxit PhantomPDF Version: 10.1.1

Compiled by Teacher:
(Name & Dated Signature)

Checked By Teacher:
(Name & Dated Signature)

HOD
(Name & Dated Signature)

Head
Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

PARFORMA-2

Result Analysis for Even Semester Examination Session 2019-20

B. Tech III Semester

Branch: Information Technology

Semester /Branch	Total No. of Students (Enrolled)	Total No. of Students Detained	Total No. of Students Appeared	Result Declared	Result Incomplete	No. of Students Pass CP (0)	Total No. of Students with CP status	No. of Students Fail	Subject Name	Sec.	Subject Wise Faculty Performance								
											Total No. of Students Result Declared	No. of Pass Students	% of Pass Students	% of Pass Students in Both Sec. (A+B)	Total Marks in Subject	% Avg. Marks	% Avg. Marks in Both Sec. (A+B)	Highest Score	No. of Students score more than 75%	Name of the Subject Teacher
III	20	0	20	20	0	14	6	0	MATH-IV	A	20	17	85.00	85.00	957	47.85	47.85	93	1	Dr. M. P. Singh
									TC	A	20	20	100.00	100.00	1063	53.15	53.15	71	0	Dr. Mamta Sharma
									DS	A	20	18	90.00	90.00	991	49.55	49.55	71	0	Mr. Ajay Parashar
									CO & A	A	20	19	95.00	95.00	937	46.85	46.85	74	0	Ms. Rupali Mahajan
									DS&TL	A	20	16	80.00	80.00	683	34.15	34.15	49	3	Ms. Deepti Mittal
									CSS	A	20	20	100.00	100.00	506	50.60	50.60	43	2	NPTEL

A. Carry Paper Detail:

No. of Carry Papers	Even Sem CP (1)	Even Sem CP (2)	Even Sem CP(3)	Even Sem CP(4)	Even Sem CP(5)	Even Sem CP(6)	Aggregate Marks: 75% and above	Between 60-75%	Less than 60%
No. of Students	4	1	0	1	0	0	2	11	7
Percentage	20.0	0.0	0.0	5.0	0.0	0	10.0	55.0	35.0

B. Top Five Rankers in the Branch:

S.N.	University Roll. No.	Name of the student	Marks Obtained (950)	Aggregate Percentage of Marks Obtained	Remark
1	1806413020	RAJAT SHARMA	752	79.16	VERY GOOD
2	1806413004	Aparna Kulshrestha	733	77.16	GOOD
3	1806413023	SAURABH SHUKLA	702	73.89	OK
4	1806413021	ROHIT PATHAK	690	72.63	OK
5	1806413026	SHIVAM CHAHAR	690	72.63	OK

Compiled by Teacher:
(Name & Dated Signature)

Checked by Teacher:
(Name & Dated Signature)

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA90987 85ACF2B07E25E09D7F5B87A4DCA3012 47D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:23:16+05'30'
Foxit PhantomPDF Version: 10.1.1

HOD
(Name & Dated Signature)

Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

HINDUSTAN COLLEGE OF SCIEN

B. TECH. / M.B.A. BRANCH ......IT......

S. No.	Roll No.	Name	Date →	1/9	2/9
			No.	1	2
1.	2100640130001	Abhishek Nohwar		A	A
2.	2	Adarsh Gautam		A	A
3.	3	Aditya Pratap Singh		A	A
4.	4	Aditya Rajpal		A	A
5.	5	Akash Thakur		A	A
6.	6	Akshit		1	2
7.	7	Aman Chaudhary		1	A
8.	8	Ananya Dubey		A	1
9.	9	Anjali Chaturvedi		A	A
10.	10	Ankit Arora		A	A
11.	11	Ankit Gautam		A	A
12.	12	Ashish Garg		A	A
13.	13	Aslan		A	A
14.	14	Astha		1	2
15.	15	Atharv Jadon		A	A
16.	16	Bhuvnesh Sharma		A	A
17.	17	Bishakha Kumari Shaw		A	A
18.	18	Chetan Chaudhary		A	A
19.	19	Devesh Kumar		1	2
20.	20	Dheeraj Yadav		A	A
21.	21	Dhwani Agarwal		1	A
22.	22	Diwakar Goswami		A	A
23.	23	Diya Gupta		1	2
24.	24	Gaurav Raj		A	A
25.	25	Govind Tyagi		1	2
26.	26	Harsh Sharma		1	2
27.	27	Harsh Tripathi		A	1
28.	28	Harshul Chawla		1	2
29.	29	Himanshu Singh		A	A
30.	30	Krati Maheshwari		1	A
	Total Presents				

Sign. of Faculty

THEORY / TUTORIAL ATTENDANCE & SESSIONAL MARKS RECORD

Max. Period Engaged..................

THEORY ATTENDANCE

Date	01/12	02/12	07/12	08/12	09/12	12/12	14/12	15/12	28/12	20/12	21/12	22/12	23/12	26/12	28/12	02/01	04/01	04/01	04/01		TOTAL
No.	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	
1	.	.	.	.	.	.	.	A	/	.	A	.	.	.	A	.	.	.	.		
2	.	.	15	.	.	16	.	A	.	.	.	17	.	.	18	.	.	.	.		
3	.	31	32	.	.	.	33	A	.	34	.	35	.	.	36	37	38	39	40		
4	12	.	13	14	.	.	15	A	.	16	.	.	.	.	.	.	17	18	19		
5	.	.	.	.	.	.	.	A	.	.	.	.	.	.	.	.	.	.	.		
6	23	.	.	24	.	.	25	A	.	.	.	.	26	.	27	.	28	29	30		
7	31	32	.	.	.	33	34	35	36	37	.	38	39	40	41	.	42	43	44		
8	34	.	35	36	.	37	38	39	40	41	42	43	44	45	46	.	.	.	.		
9	.	.	.	.	.	.	.	A	.	.	.	.	.	.	.	.	.	.	.		
10	20	21	22	23	.	.	24	25	26	27	28	29	.	30	31	32	33	34	35		
11	.	.	.	.	.	.	.	A	.	.	19	20	.	21	.	22	.	.	.		
12	.	.	.	.	.	.	.	A	.	.	.	.	.	.	.	.	.	.	.		
13	.	.	.	.	.	.	.	A	.	.	.	.	.	.	.	.	.	.	.		
14	42	43	44	45	.	46	.	47	.	.	48	49	.	50	51	.	.	.	.		
15	.	.	.	.	.	.	.	A	.	.	.	.	.	.	.	.	.	.	.		
16	.	.	.	.	.	.	.	A	.	.	.	.	.	.	.	.	.	.	.		
17	27	28	29	30	31	32	33	34	35	36	37	38	39	.	40	.	.	.	.		
18	.	.	.	.	.	.	.	A	.	.	.	.	.	.	.	.	.	.	.		
19	38	39	.	40	41	42	43	A	44	45	46	47	48	49	50	51	52	53	54		
20	.	.	.	.	.	.	.	A	.	.	.	.	.	.	.	.	.	.	.		
21	32	.	33	34	35	36	37	A	38	39	40	41	.	.	.	.	.	.	.		
22	.	.	.	.	.	.	.	A	.	.	.	.	.	.	.	.	.	.	.		
23	46	47	.	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63		
24	27	.	.	28	.	.	29	A	.	30	.	.	31	.	.	.	.	.	.		
25	.	.	.	.	.	.	.	29	30	31	.	.	.	.	.	32	33	34	35		
26	.	21	.	.	.	.	22	A	23	24	25	.	.	.	.	.	.	.	.		
27	.	.	27	28	.	29	30	.	31	32	33	34	35	36	37	38	39	40	41		
28	42	43	.	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59		
29	.	.	.	.	.	.	.	A	.	.	.	.	.	.	.	.	.	.	.		
30	33	34	.	35	.	36	37	38	39	40	41	42	.	43	44	45	46	47	48		

Sign. of Faculty

HINDUSTAN COLLEGE OF SCIENCE & TECHNOLOGY, MATHURA

B. TECH. / M.B.A. BRANCH ......IT...... SEMESTER ...3th... YEAR 20.22. - 20.23.

TUTORIAL ATTENDANCE

Date	16/09	23/9	30/9	07/10	14/10	20/10	04/11	07/11	14/11	30/11	07/12	14/12	21/12	28/12												TOTAL	Assignments	A-1	A-2	A-3	A-4	A-5	T-1	T-2	T-3	Tests	Atten.	Assign.	Total
No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25			10	10	10	10	10	30	30	30	30	10	10	50
1	.	A	.	A	A	.	A	.	A	.	A	.	A	.																									
2	1	A	A	A	A	.	2	3	A	A	A	.	A	4																									
3	1	.	A	2	3	A	4	5	A	6	7	8	A	9																									
4	.	A	A	.	A	1	2	3	A	.	4	5	A	.																									
5	A	.	.	.	A	.	A	A	A	.	A	.	A	.																									
6	A	A	1	.	2	3	4	5	A	.	A	6	A	7																									
7	1	.	2	3	4	5	6	.	A	7	A	8	A	9																									
8	1	.	2	.	.	.	3	4	5	6	7	8	9	10																									
9	.	A	.	A	.	.	A	.	A	.	A	.	A	.																									
10	1	.	2	.	.	.	3	.	A	4	5	6	A	7																									
11	1	.	2	3	.	4	5	.	A	.	A	.	6	.																									
12	.	.	.	.	.	.	A	.	A	.	A	.	A	.																									
13	.	A	.	A	.	A	A	.	A	.	A	.	A	.																									
14	1	2	3	4	5	.	6	.	7	.	8	.	9	10																									
15	.	.	.	.	.	.	A	.	A	.	A	.	A	.																									
16	1	.	2	3	.	A	A	.	A	.	A	.	A	.																									
17	1	2	3	4	5	6	7	.	A	.	8	9	10	11																									
18	.	.	.	.	.	.	A	.	A	.	A	.	A	.																									
19	.	A	.	1	2	3	4	5	6	7	A	8	9	10																									
20	1	.	A	.	A	.	A	.	A	.	A	.	A	.																									
21	1	.	2	3	4	5	6	7	8	.	9	10	11	.																									
22	.	A	A	.	.	.	A	.	A	.	A	.	A	.																									
23	1	.	2	3	4	5	6	7	8	9	A	10	11	12																									
24	A	.	1	2	3	4	A	5	6	7	A	8	A	.																									
25	1	A	.	.	2	.	3	4	5	6	A	.	A	.																									
26	1	.	.	.	A	.	A	.	A	.	A	1	A	.																									
27	1	2	3	.	4	.	5	6	A	.	7	8	9	10																									
28	1	2	3	4	5	6	7	8	9	10	11	12	13	14																									
29	1	A	.	.	.	.	A	.	A	.	A	.	A	.																									
30	1	2	3	4	5	6	7	8	9	10	11	12	13	14																									
	17	5	14	12	13	11	13	13	9	10	10	15	10	13																									

Sign. of H.O.D.

Hindustan C... Technology
Farah, Mathura

Sign. of Dean Academics

HINDUSTAN COLLEGE OF SCIENCE & TECHNOLOGY, MATHURA

B. TECH. / M.B.A. BRANCH..... IT ..... SEMESTER ..... 3rd ..... YEAR 20 22 - 20 23

S. No.	Roll No.	Name	Date	THEORY ATTENDANCE
			No.	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
31.	2100640130031	Kunal		A A A A A A A A A A A A A A A A A A 1 2 A A A A
32.	32	Manjeet Singh		A A A A A A A A A A A A A A A A A A A A A A A A
33.	33	Mani Sikarwar		A A A A A A A A A A A A A A A A A A A A A A A A
34.	34	Mikhil Rathore		1 2 3 4 5 6 7 8 9 A 10 11 12 13 14 15 16 17 A 18 19 20 21 22 23
35.	35	Mikunj Maheshwari		1 2 3 A A 4 5 6 7 8 9 10 11 A 12 13 A A A 16 17 18 A A 19
36.	36	Mishant Dixit +3		A A A A A 1 2 A 3 A 4 5 6 7 8 9 10 11 12 13 A 14 15 16 17
37.	37	Mishkarsh Rathore		A 1 2 A A A A 3 4 5 6 7 8 9 A 10 11 A 12 13 A A A A A
38.	38	Mitish Kumar Sharma		A A A A A A 1 2 A 3 A 4 A A A A A A A A A A A A
39.	39	Palak Sharma		1 2 3 A A 4 5 6 7 8 9 10 11 A 12 13 A A A 15 16 17 18 19 A
40	41	Parth Singh Sikarwar		A A A A A A A A A 1 2 3 4 A 5 A A 6 A A 7 A A A A
41.	42	Prashant Pal Singh		A A A A A A A A A A A A A A 1 A A A A A A A 3 A
42.	43	Prashant Upreti		1 2 3 A A 4 A 5 6 A 7 8 9 A 10 11 A A A A 14 A 15 16 17
43	44	Prateek Chaudhary		A A A A A 1 2 3 4 5 6 7 8 9 10 11 A A A A A A 12 A 13
44.	45	Pratham Yadav		A A A A A A A A 1 A 2 A 3 A A A A A A A A A A A
45	46	Priyanshi Gupta		A 1 2 A A 3 4 5 6 7 8 9 A 10 11 12 A A A A A A 17 A A
46.	48	Priyanshu Verma		A A A A A A A 1 2 A A A 3 A 4 A 5 A A A A 6 A 7
47.	47	Rohit Kumar Sharma		A A A A A A A A A 1 2 A 3 4 5 6 A A A A A A 11 A A
48.	50	Sachin Gautam		A A A A A A 1 2 A 3 4 5 6 7 A 8 9 A A A A A 14 15 A A
49.	51	Sahil Ali		1 A A A A 2 3 A 4 5 6 7 8 A 9 A A A A A A 15 16 A A A
50	52	Sakar Talwar		1 2 3 A A 4 A 5 6 7 8 9 10 11 12 A A A A A A 13 A A A
51	53	Saljam Singh		A 1 2 3 4 5 A 6 7 8 9 10 11 A 12 A A A A A A 18 A A A
52.	54	Shivani		1 A A A A 2 A 3 4 5 6 7 8 9 10 A A A A A A A A A A
53.	55	Siddharth Jadon		A A A A A A A 1 A A A A A A A 2 A A A A A A A A
54.	56	Suryansh Singh		A A A A A A A A 1 2 3 4 5 6 7 A A A A A A a A A A
55	57	Udit Kukreti		1 2 3 A A 4 5 6 7 8 9 10 11 A 12 A A A A A A 18 A A A
56.	58	Vikash Solanki		1 2 3 A A 4 5 6 7 8 9 10 A A 11 A A A A A A 15 A 16 A
57.	60	Vishesh Sharma		A A A A A A A A A A 1 2 3 4 5 A A A A A A A A A A
58.		Pratik Mudgal		A A A A A A A 1 2 3 4 5 6 7 A 8 9 A A A A A 14 15 A 16 17
59.		Priyanshi Chaudhary		A A A A A A A 1 2 3 4 5 6 7 8 9 A A A A A A 15 16 A A A
60.		Shivam Sonet		A A A A A A A A A 1 2 3 A A 4 5 A A A A 3 4 A A 16
61.		Devrishi		A A A A A A A A A A 1 2 3 4 5 6 A A A A A A A A A
62		Kushagra Kulshreshtha		A A A A A A A A A A A A A A A A A A A A A 1 A 2 3
		Mohit Sharma		A A A A A A A A A A A A A A A A A A A A A A A A
		Gagandeep Baghel		A A A A A A A A A A A A A A A A A A A A A 1 2 A A 3
		Sudhanshu Vaishal		A A A A A A A A A A A A A A A A A A A A A A A A
	Total Presents			41 24 38 44 22 23 24 31 29 35

Sign. of Faculty

Sign. of H.O.D.

(16)

THEORY / TUTORIAL ATTENDANCE & SESSIONAL MARKS RECORD

Max. Period Engaged.....

THEORY ATTENDANCE																																																TOTAL
18/10		19/10		20/10		2/11		3/11		4/11		7/11		9/11		10/11		11/11		14/11		16/11		17/11		18/11		21/11		23/11		25/11		30/11																
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45																													
1	•	•	•	•	•	•	•	•	•	•	•	•	A	A	A	A	A	A	A																													
2	•	•	•	•	•	•	•	•	•	•	•	•	A	A	A	A	A	A	A																													
3	•	•	•	•	•	•	•	•	•	•	•	•	A	A	A	A	A	A	A																													
4	24	25	26	27	•	28	29	30	31	32	•	•	A	•	•	A	•	33	34																													
5	20	21	•	•	22	•	•	23	24	25	26	27	•	28	29	30	31	32	•	33																													
6	18	19	20	21	22	23	24	25	•	•	26	27	28	29	30	31	32	33	•	34																													
7	15	16	•	17	•	18	19	20	•	•	•	•	•	21	22	23	24	25	26	•																													
8	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•																													
9	•	20	21	22	23	•	24	25	•	•	26	27	28	29	30	•	•	31	32	33																													
10	•	8	•	9	•	10	•	•	11	12	13	14	•	•	15	•	16	17	•	18																													
11	4	•	•	•	•	5	6	7	•	•	•	•	•	8	9	•	10	11	•	•																													
12	18	19	20	21	22	23	24	25	26	27	28	29	•	30	•	•	•	31	•	32																													
13	19	20	21	22	•	23	24	•	25	26	27	28	•	29	30	31	32	33	•	34																													
14	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•																													
15	22	•	23	24	25	26	27	28	29	30	31	32	33	•	34	35	36	37	38	39																													
16	10	•	11	12	13	•	•	14	15	16	•	17	•	•	•	•	•	18	•	19																													
17	15	16	17	18	•	•	•	19	20	21	22	23	24	25	26	27	28	29	•	30																													
18	17	18	19	20	•	21	22	23	24	25	26	27	•	28	29	30	•	31	32	33																													
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	•	•	34	35	36	37																													
20	•	21	22	23	24	25	26	27	28	29	30	31	•	32	•	•	33	34	•	35																													
21	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42																													
22	•	4	5	6	•	•	•	7	8	9	10	11	12	13	14	15	16	17	•	18																													
23	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•																													
24	15	16	17	18	19	20	21	22	23	24	25	26	•	27	28	29	30	31	•	32																													
25	27	24	25	26	27	•	28	29	30	31	32	33	34	35	36	•	37	38	39	40																													
26	18	19	•	•	20	•	•	•	21	22	•	•	•	23	24	25	26	•	27	28																													
27	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•																													
28	18	19	•	20	•	21	22	23	24	25	•	26	27	28	29	•	30	31	•	•																													
29	20	21	22	23	•	24	25	26	•	•	27	28	29	30	31	32	33	34	•	35																													
30	11	12	•	•	•	•	•	13	14	15	16	•	17	18	•	•	19	•	20																													
31	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•																													
32	4	5	6	7	•	•	•	8	9	10	11	•	•	12	13	14	15	16	17	18																													
33	5	6	7	8	9	10	•	•	11	12	13	14	•	15	16	17	•	18	19	•																													
34	4	5	6	7	•	•	•	8	9	•	10	•	•	11	•	•	•	•	•	•																													
																					1	2	3	4	5	6	7	8																					
35	35	30	30	21	28	33	•	34	33	32	35	19	40	37	26	33	38	19	33																													

Sign. of Dean Academics

(17)

ICE & TECHNOLOGY, MATHURA

..... SEMESTER ..... 3rd ..... YEAR 20 22 - 20 23

				TOTAL	Assignments	A-1	A-2	A-3	A-4	A-5	TESTS			Sessional Marks - 50			
											T-1	T-2	T-3	Tests	Atten.	Assign.	Total
2	23	24	25			10	10	10	10	10	30	30	30	30	10	10	50

Sign. of Dean Academics

(19)

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:22:46+05'30'
Foxit PhantomPDF Version: 10.1.1

HINDUSTAN COLLEGE OF SCIEN

B. TECH. / M.B.A. BRANCH..... IT

S. No.	Roll No.	Name	Date	1/9	2/9
			No.	1	2
31.	2100640130081	Kunal		A	A
32.	32	Manjeet Singh		A	A
33.	33	Manu Sikarwar		A	A
34.	34	Mikhil Rathore		1	2
35.	35	Mikunj Maheshwari		1	2
36.	36	Nishant Dixit +3		A	A
37.	37	Nishkarsh Rathore		A	1
38.	38	Nitish Kumar Sharma		A	A
39.	39	Palak Sharma		1	2
40.	41	Parth Singh Sikarwar		A	A
41.	42	Prashant Pal Singh		A	A
42.	43	Prashant Upreti		1	2
43.	44	Pratik Chaudhary		A	A
44.	45	Pratiom Yadav		A	A
45.	46	Priyanshi Gupta		A	1
46.	48	Priyanshu Verma		A	A
47.	49	Rohit Kumar Sharma		A	A
48.	50	Sachin Gautam		A	A
49.	51	Sahil Ali		1	A
50.	52	Sakar Talwar		1	2
51.	53	Satyam Singh		A	1
52.	54	Shivani		1	A
53.	55	Siddharth Jadon		A	A
54.	56	Suryansh Singh		A	A
55.	57	Udit Kukreti		1	2
56.	58	Vikash Solanki		1	2
57.	60	Vishesh Sharma		A	A
58.		Pratik Mudgal		A	A
59.		Priyanshi Chaudhary		A	A
60.		Shivam Soret		A	A
61.		Devrishi		A	A
62.		Kushagra Kulshreshtha			
		Mohit Sharma			
		Gagandeep Pradhan			
		Sudhanshu Vaishav			
	Total Presents				

Sign. of Faculty

(16)

THEORY / TUTORIAL ATTENDANCE & SESSIONAL MARKS RECORD

Max. Period Engaged.....

THEORY ATTENDANCE

No.	01/12	02/12	04/12	08/12	09/12	12/12	14/12	15/12	20/12	20/12	21/12	22/12	23/12	26/12	28/12	02/01	04/01	04/01	TOTAL
	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	
31	A	A	A	·	·	·	·	A	·	·	·	·	·	·	·	·	·	·	·	·	
32	·	·	·	·	·	·	A	·	·	·	·	·	·	·	·	·	·	·	·	·	
33	A	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	
34	35	36	37	38	·	39	40	41	42	43	·	44	·	45	·	46	47	48	49		
35	34	·	35	36	·	·	·	·	·	37	38	·	39	40	41	·	42	43	44		
36	35	36	·	·	·	37	38	39	40	41	42	43	44	45	46	47	48	49	50		
37	·	27	28	29	30	31	·	32	33	34	·	35	·	36	·	37	38	39	40		
38	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·		
39	34	·	35	36	37	38	39	40	·	·	41	42	43	·	·	·	·	·	·		
41	19	·	·	20	21	·	22	·	·	23	·	·	·	·	·	·	·	·	·		
42	12	·	13	14	·	·	15	·	·	16	·	·	·	·	·	·	17	18	19		
43	33	34	35	36	37	38	39	40	41	42	·	43	44	45	46	47	48	49	50		
44	35	36	·	37	·	38	39	·	·	·	·	·	·	·	40	41	42	43	44		
45	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·		
46	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	·	·	·	·		
48	20	21	·	22	23	·	24	·	25	26	·	·	27	28	29	·	·	·	·		
49	31	32	33	34	35	36	37	38	·	·	·	39	40	41	42	·	43	44	45		
50	34	35	·	36	37	38	39	40	41	42	43	·	44	45	46	47	48	49	50		
51	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	·	53	54	55		
52	36	37	·	38	39	40	41	42	43	44	·	45	46	47	·	·	48	49	50		
53	43	44	45	46	47	48	49	50	·	51	52	53	54	55	56	57	58	59	60		
54	·	19	·	·	20	·	21	·	·	·	·	·	22	·	·	·	·	·	·		
55	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·		
56	33	34	35	36	37	38	39	40	41	42	43	·	44	45	46	·	47	48	49		
57	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	·	56	57	58		
58	29	30	31	·	·	32	33	34	35	36	37	38	39	40	41	42	43	44	45		
60	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·		
P	32	·	33	34	35	36	·	37	38	39	40	·	41	42	·	·	43	44	45		
P	36	37	38	39	40	41	42	·	43	44	45	46	47	48	49	·	50	51	52		
S	21	·	22	·	·	·	23	24	·	·	25	26	27	·	28	·	29	30	31		
D	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·		
K	19	20	21	22	23	24	25	26	·	27	28	29	30	31	32	·	33	34	35		
M	20	21	22	·	24	25	·	26	·	27	28	29	·	30	31	32	33	34	35		
G	·	·	·	·	·	·	·	·	·	12	·	·	·	13	14	·	·	·	·		
S	9	·	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26		
	37	29	28	35	24	31	37	28	27	37	20	32	28		31	19	32	32	32		

Sign. of Faculty

(18)

HINDUSTAN COLLEGE OF SCIENCE & TECHNOLOGY, MATHURA

B. TECH. / M.B.A. BRANCH..... IT ..... SEMESTER ..... 3rd ..... YEAR 2022 - 2023

Date	TUTORIAL ATTENDANCE																									TOTAL	Assignments	A-1	A-2	A-3	A-4	A-5	TESTS			Sessional Marks - 50			
	7/9	14/9	12/10	26/10	30/11	07/12	14/12	21/12	25/12																								T-1	T-2	T-3	Tests	Atten.	Assign.	Total
	10	10	10	10	10																												30	30	30	30	10	10	50
No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25													
31	A	A	1	·	·	A	A	A	·																													
32	A	A	1	·	·	A	A	A	·																													
33	A	A	1	·	·	A	·	A	·																													
34	A	1	2	3	4	5	A	·																														
35	A	1	·	·	2	·	3	4																														
36	A	1	2	3	A	4	5	6																														
37	A	1	·	·	2	·	·	·																														
38	A	1	·	·	A	·	·	·																														
39	A	1	2	3	4	5	6	·																														
41	A	1	2	3	A	4	5	·																														
42	A	·	·	·	1	2	·																															
43	1	2	3	4	5	6	A	7																														
46	A	·	1	2	A	3	A	4																														
45	A	1	·	·	A	·	A	·																														
46	A	·	1	2	3	4	5	6																														
48	1	2	3	·	4	5	6	7																														
49	A	1	2	3	4	5	A	6																														
50	A	1	2	3	A	4	5	6																														
51	A	1	2	3	4	5	6	7	8																													
52	1	2	3	4	5	A	6	·																														
53	1	2	3	4	5	6	7	8	9																													
54	A	1	2	3	4	5	A	·																														
55	A	·	·	·	A	·	A	·																														
56	A	1	2	3	4	5	6	7																														
57	1	2	3	4	5	6	·	7	8																													
58	A	·	1	2	3	4	5	6	7																													
60	A	1	A	·	·	A	·	·	·																													
Pratik	A	1	2	3	4	·	5	·																														
Priyanshi	A	1	2	3	4	5	6	7																														
Shivam	A	·	1	2	3	4	5	6																														
Devrishi	A	1	·	·	·	·	·	·																														
Kushagra		1	2	3	4	5	6	7	8																													
Mohit				1	·	2	·	3	4																														
Gagan		1	2	3	·		·	4	5																													
Sudhanshu					1	2	3	4	5																														
	5	22	7	23	21	21	36	19	19																													

Sign. of Faculty

Head
Department of Information Technology
Hindustan College of Science & Technology
Mathura

Sign. of Dean Academics

(19)

RAJEEV KUMAR UPADHYAY

Digitally signed by RAJEEV KUMAR UPADHYAY
DN: C=IN, O=Personal, PostalCode=282001, S=Uttar Pradesh, SERIALNUMBER=AA3E8C12CFAA9098785ACF2B07E25E09D7F5B87A4DCA301247D08CBAEE03B9A3, CN=RAJEEV KUMAR UPADHYAY
Reason: I am the author of this document
Location: your signing location here
Date: 2023.09.06 18:22:30+05'30'
Foxit PhantomPDF Version: 10.1.1

HINDUSTAN COLLEGE OF SCIENCE & TECHNOLOGY, MATHURA

B. TECH. / M.B.A. BRANCH IT SEMESTER 3th YEAR 2022 - 2023

THEORY ATTENDANCE

Roll No.	Name	Date / No.	1/9 (1)	2/9 (2)	3/9 (3)	3/9 (4)	5/9 (5)	6/9 (6)	7/9 (7)	8/9 (8)	9/9 (9)	12/9 (10)	12/9 (11)	14/9 (12)	15/9 (13)	16/9 (14)	19/9 (15)	21 (16)	22 (17)	29 (18)	6/10 (19)	7/10 (20)	10/10 (21)	11/10 (22)	13/10 (23)	18/10 (24)	19/10 (25)
2100640130(00)	Abhishek Nohwar		A	A	A	A	A	A	A	A	A	A	A	A													
2	Adarsh Gautam		A	A	A	A	A	A	A	A	A	A	A	A	2	3	4	5	6						7	8	
3	Aditya Pratap Singh		A	A	A	A	A	1	2	3	4	A	5	6	7		8	9	10	11	12			13	14	15	16
4	Aditya Roypal		A	A	A	A	A	A	A	A	A	A	A	A				1	2							3	
5	Akash Thakur		A	A	A	A	A	A	A	A	A	A	A	A													
6	Akshit		1	2	3	A	A	4	5	6	7	8	A	A	9			10	11			12	13				14
7	Aman Chaudhary		1	A	A	A	A	A	2	3	4	5	6	7			8	9	10			11		12	13	14	15	16
8	Ananya Dubey		A	1	2	A	A	3	4	5	A	A	6	7	8	9	10	11	12			13			14	15	16	17
9	Anjali Chaturvedi		A	A	A	A	A	A	A	A	A	A	A	A													
10	Ankit Arora		A	A	A	A	A	A	1	2	A	A	A	3			4			5							
11	Ankit Gautam		A	A	A	A	A	1	A	2	A	3	4	5	6		7					8	9	10	11	12	
12	Ashish Garg		A	A	A	A	A	A	A	A	A	A	A	A				1									
13	Aslam		A	A	A	A	A	A	A	A	A	A	A	A	1	2	3										
14	Astha		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
15	Athar Jadan		A	A	A	A	A	A	A	A	A	A	A	A													
16	Bhuvnesh Sharma		A	A	A	A	A	A	1	2	A	3	4	5	6		7		8		9	10	11	12			
17	Bishakha Kumari Shaw		A	A	A	A	A	A	A	A	A	A	A	A	1		2	3									
18	Chetan Chaudhary		A	A	A	A	A	A	A	A	A	A	A	A				1									
19	Dinesh Kumar		1	2	3	4	5	6	7	8	9	10	11	12	13	14		15	16	17	18	19	20	21	22	23	24
20	Dinraj Yadav		A	A	A	A	A	A	A	A	A	A	A	A	A			1							2		
21	Dhwani Agarwal		1	A	A	A	A	2	3	A	4	A	5	6	7		8	9	10			11	12	13	14	15	16
22	Diwakar Goswami		A	A	A	A	A	1	A	2	A	A	A	A	A												
23	Diya Gupta		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
24	Gaurav Raj		A	A	A	A	A	A	A	1	2	3	4	5	6	7		8	9	10	11	12		13	14		15
25	Govind Tyagi		1	2	3	A	A	4	5	6	A	A	7	A	8	9	10		11	12	13		14		15		16
26	Harsh Sharma		1	2	3	A	A	4	5	A	A	6	7	8	9	10		11	12								
27	Harsh Tripathi		A	1	2	3	4	A	A	5	A	A	6	7	8		9	10	11	12	A	13	14				
28	Harshul Chawla		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		16	17	18	19	20	21	22	23	24
29	Himanshu Singh		A	A	A	A	A	A	A	1	A	A	A	A	2	3	4	5									
30	Krati Maheshwari		1	A	A	A	A	A	2	3	4	5	6	7	8		9	10	11	12	13	14	15	16	17	18	19
	Total Presents																										

Sign. of Faculty

Sign. of H.O.D.

Read
Department of Information Technology
Hindustan College of Science & Technology
Farah, Mathura

THEORY / TUTORIAL ATTENDANCE & SESSIONAL MARKS RECORD

Max. Period Engaged.....

THEORY ATTENDANCE

13/10 (26)	13/10 (27)	19/10 (28)	20/10 (29)	2/11 (30)	3/11 (31)	4/11 (32)	7/11 (33)	9/11 (34)	9/11 (35)	10/11 (36)	11/11 (37)	14/11 (38)	16/11 (39)	17/11 (40)	18/11 (41)	21/11 (42)	23/11 (43)	25/11 (44)	30/11 (45)	TOTAL
		9			10	11							12	13			14			
17	18	19			20	21	22	23	24	25		26		27		28	29	30		
4	5			6	7	8							9		10	11				
15	16				17				18			19		20	21				22	
		17	18		19	20	21	22	23		24	25	26		27	28	29		30	
19		20			21	22	23	24	25	26	27	28	29	30		31	32		33	
7	8	9			10	11		12	13	14	15		16		17	18			19	
						13				14	15		16		17	18				
12	13		14	15	16	17		18	19	20	21		22	23	24	25	26			
	25	26	27		28	29	30		31	32	33			34		35	36		37	
3													4	5						
	17		18		19	20	21	22	23	24	25	26	27		28		29	30	31	
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	
16	17	18	19		20		21				22		23	24	25				26	
17				18	19	20		21	22			23	24	25			26	27	28	
							14	15	16	17			18	19		20				
	15			16	17	18	19	20	21		22			23			24	25	26	
25		26	27	28		29	30	31		32	33	34	35	36	37	38	39	40	41	
20	21	22	23	24	25	26	27	28		29	30	31	32	33	34	35	36	37	38	

Sign. of Dean Academics

CE & TECHNOLOGY, MATHURA

.. SEMESTER 3th YEAR 2022 - 2023

23	24	25	TOTAL	Assignments	A-1	A-2	A-3	A-4	A-5	TESTS T-1	TESTS T-2	TESTS T-3	Sessional Marks - 50 Tests	Atten.	Assign.	Total
					10	10	10	10	10	30	30	30	30	10	10	50

Sign. of Dean Academics