21BCP110

DOMS | Page No.
Date / /

TOPIC 1: Introduction to Data Structures

1) Discuss the difference between

(1) Primitive Data Types	Non-Primitive Data Types
i) Basic data types directly operated by machine instruct.	i) Group of homogeneous or heterogeneous primitive data types
ii) Cannot be NULL	ii) Can have NULL values
iii) Size depends on type of data structure (it is fixed)	iii) Size is not fixed
iv) Eg, integer, float, character, boolean, etc	iv) Eg, Array, String, Linked List, Stack, etc

(2) Linear Data Structures	Non-Linear Data Structures
i) Have data elements arranged in sequential manner	i) Have no set sequence of connecting all its element
ii) Each element is connected to its previous and next element	ii) Each element can have multiple paths to connect to other elements
iii) Elements present in single level and can be traversed in single run.	iii) Elements present in multi-level and often can't be traversed in single run
iv) Eg, Array, Stack, Queue, etc	iv) Eg, Trees, Graphs, etc

(3) Space complexity	Time complexity
i) The amount of computer memory that is required from start of its execution to its completion wrt input size.	i) The amount of time taken by an algorithm to run wrt input size.
ii) It is calculated as sum of: → Fixed Component Store instructions, constants, variables (like arrays) → Variable Component Store recursion stack and structured variables (like trees, graphs, etc)	ii) It is calculated by counting the number of elementary operations performed (considering that each elementary operation takes a fixed amount of time to perform.

(4) Big Oh (O):

→ Express upper bound of algorithm's running time.
→ Measures the worst case time complexity
→ Maximum amount of time algorithm can take.

Omega (Ω):

→ Express lower bound of algorithm's running time.
→ Measures the best case time complexity
→ Minimum amount of time an algorithm can take to complete.

Theta (θ):

→ It measures the average case time complexity.

21BCP110

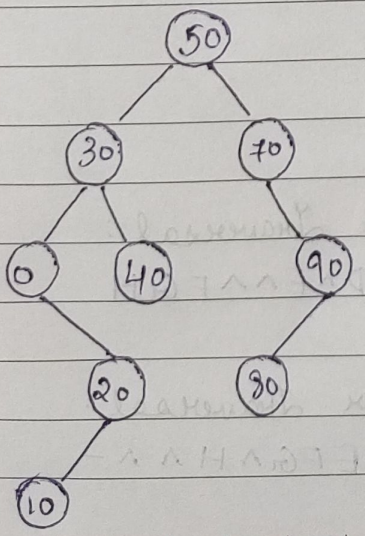
TOPIC 6 : Trees

1) Algorithm for evaluating an expression using tree

STEP 1 - START
STEP 2 - IF root is NULL return 0
STEP 3 - IF both Left and Right child of root are NULL return value of root as an integer
STEP 4 - DECLARE LeftVal and RightVal
STEP 5 - Call FUNCTION evaluateExpression for Left child of root and assign its value to LeftVal
STEP 6 - Call FUNCTION evaluateExpression for Right Child of root and assign its value to RightVal
STEP 7 - IF root's value is +, return LeftVal + RightVal
STEP 8 IF root's value is -, return LeftVal - RightVal
IF root's value is *, return LeftVal * RightVal
IF root's value is /, return LeftVal / RightVal

2) Draw Binary Search Trees and Traverse → In-order, Pre-order and Post-order:

(1) 50, 30, 0, 20, 70, 90, 80, 10, 40



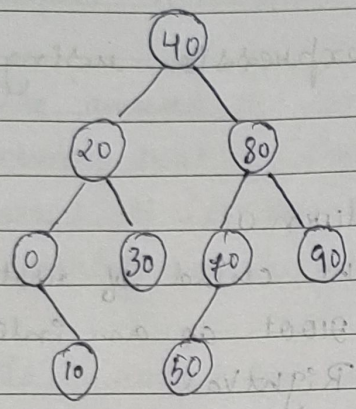
Traversal:

In-Order : 0, 10, 20, 30, 40, 50, 70, 80, 90

Pre-Order : 50, 30, 0, 20, 10, 70, 90, 80

Post-Order : 10, 20, 0, 40, 30, 80, 90, 70, 50

(2) 40, 20, 0, 30, 80, 90, 70, 10, 50



Traversal:

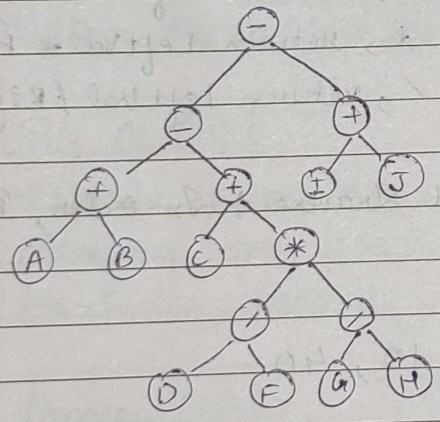
In-Order: 0, 10, 20, 30, 40, 50, 70, 80, 90

Pre-Order: 40, 20, 0, 10, 30, 80, 70, 50, 90

Post-Order: 10, 0, 30, 20, 50, 70, 90, 80, 40

3) Draw expression Trees and give Traverse: Pre-order and Post-order.

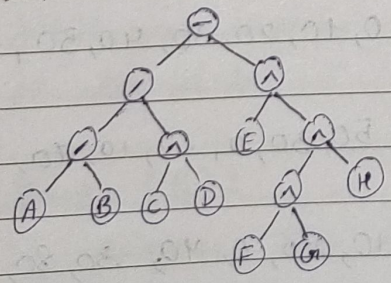
(1) A + B - C + D / F * G / H - I + J



Pre-Order Traversal:
- - + A B + C * / D F / G H + I J

Post-Order Traversal:
A B + C D F / G H / * + - I J + -

(2) A / B / C ^ D - E ^ (F ^ G) ^ H



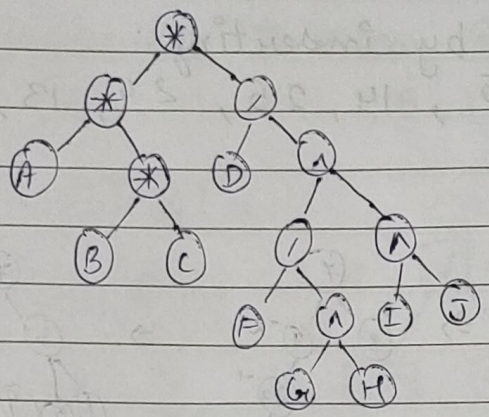
Pre-Order Traversal:
- / / A B ^ C D ^ E ^ ^ F G H

Post-Order Traversal:
A B / C D ^ / E F G ^ H ^ ^ -

21BCP110

DOMS | Page No.
Date / /

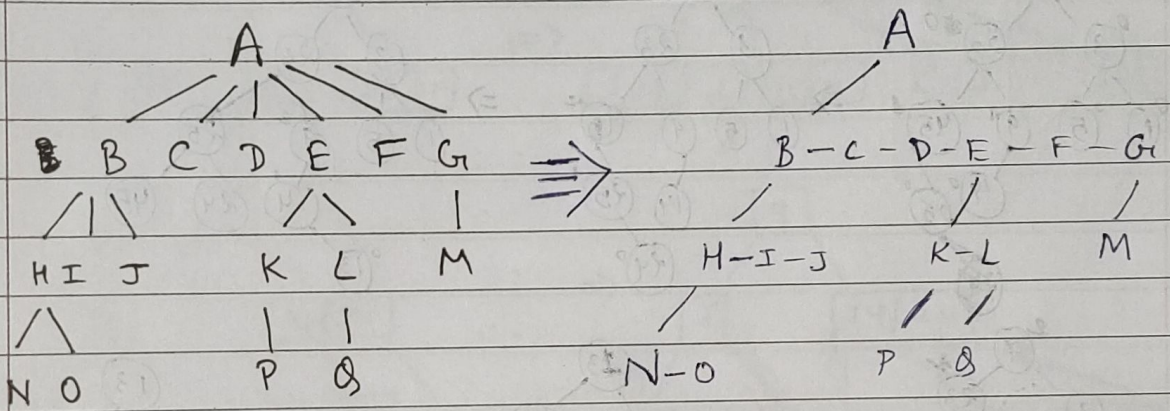
(3) A * (B * C) * D / (F / G ^ H) ^ I ^ J

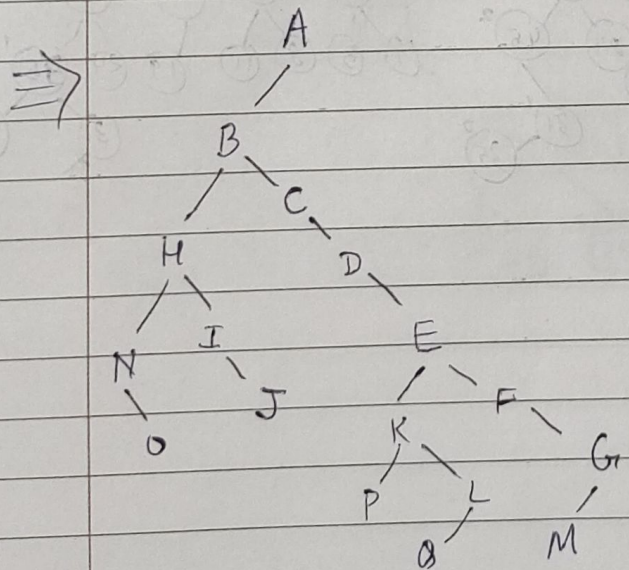


Pre-Order Traversal:
* * A * B C / D ^ / F ^ G H ^ I J

Post-Order Traversal:
A B C * * D F G H ^ / I J ^ ^ / *

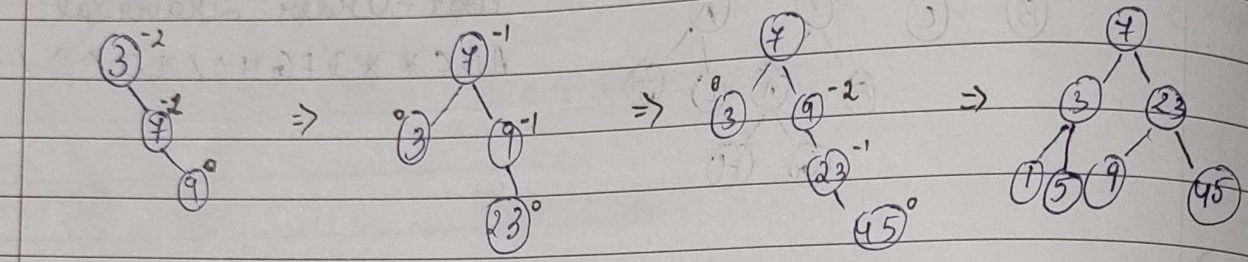
4) Convert the following tree to binary tree:

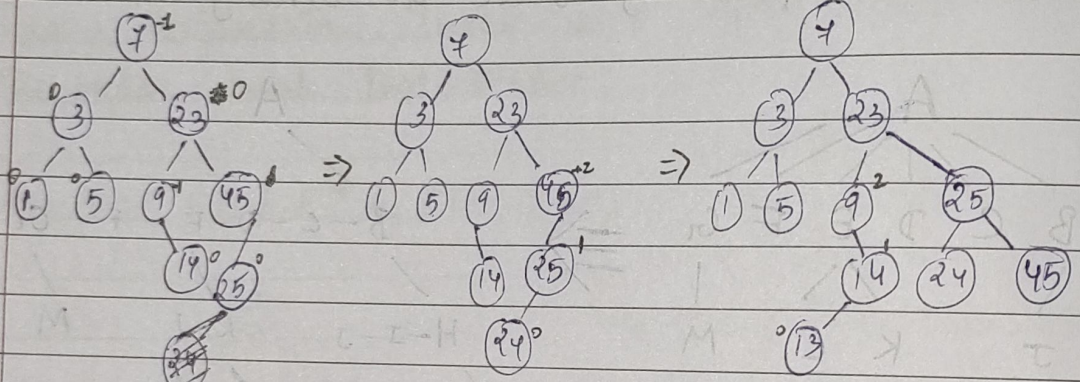


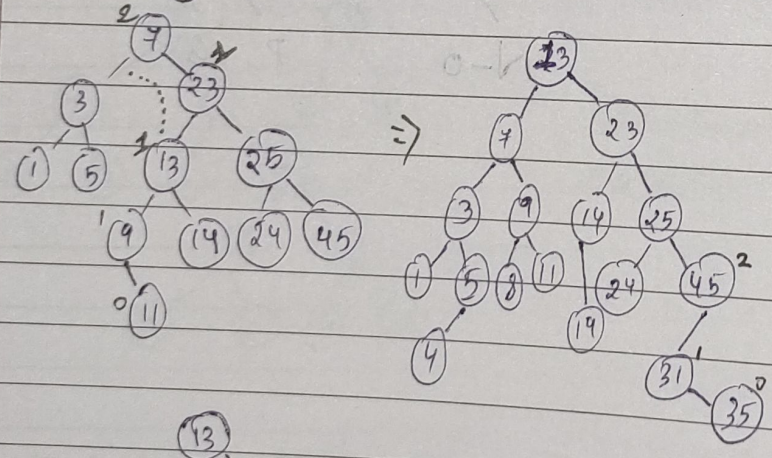


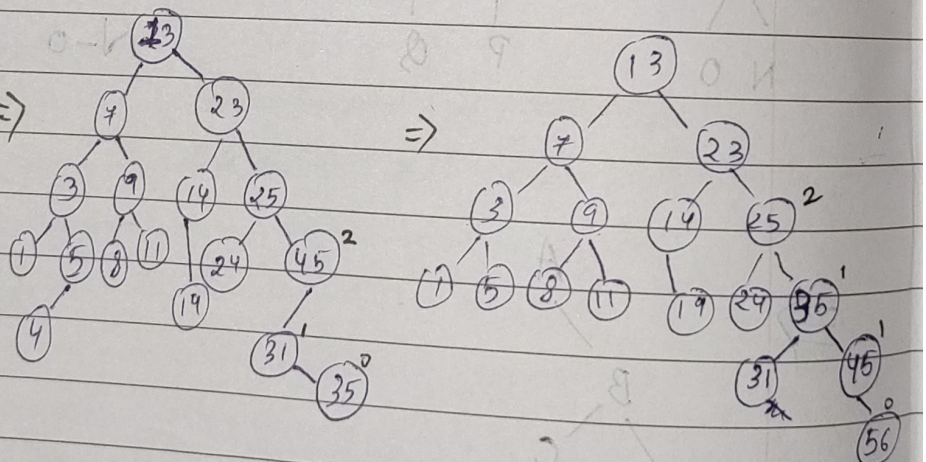
TOPIC 6.1 : Balanced Trees and More

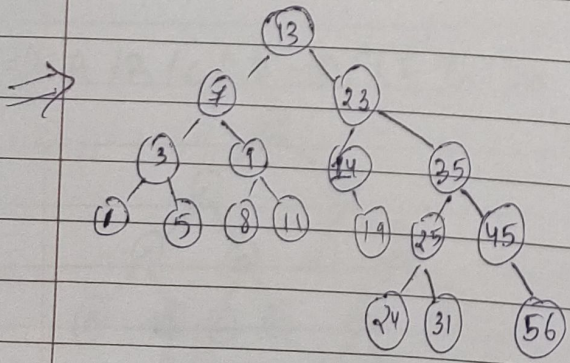
1) Construct AVL Tree by inserting :
3, 7, 9, 23, 45, 1, 5, 14, 25, 24, 13, 11, 8, 19, 4, 31, 35, 56









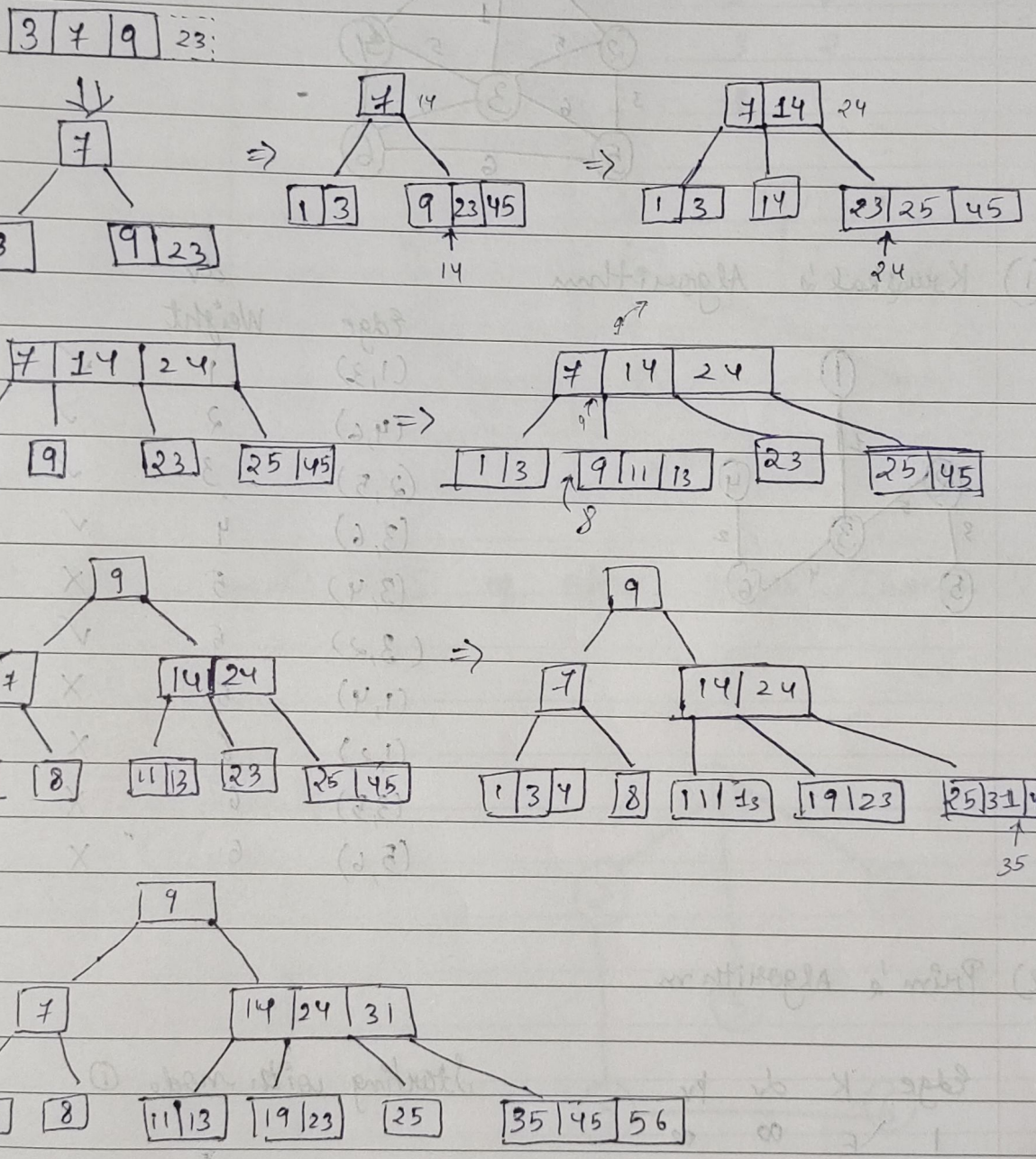


21BCP110

DOMS | Page No.
Date / /

(3) ~~A*(B*C)*D/(E/G^H)^I^J~~

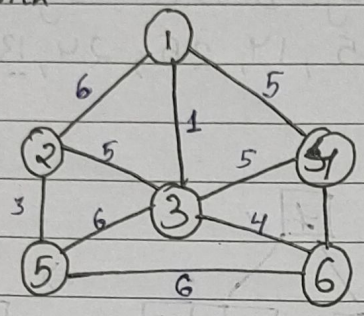
2) Construct B 4-way B-Tree by inserting
3, 7, 9, 23, 45, 1, 5, 14, 25, 24, 13, 11, 8, 19, 4, 31, 35, 56



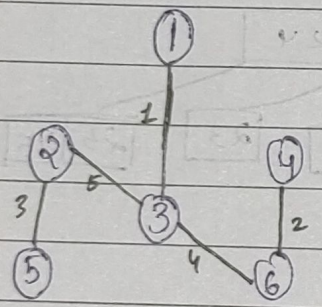
DOMS | Page No.
Date / /

TOPIC 7. : Graph

1) Find MST with



(1) Kruskal's Algorithm



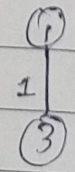
Edge	Weight	dv
(1,3)	1	✓
(4,6)	2	✓
(2,5)	3	✓
(3,6)	4	✓
(3,4)	5	X
(3,2)	5	✓
(1,4)	5	X
(1,2)	6	X
(3,5)	6	X
(5,6)	6	X

(2) Prim's Algorithm

Edge	K	dv	pv
1	F	∞	∞
2	F	∞	∞
3	F	∞	∞
4	F	∞	∞
5	F	∞	∞
6	F	∞	∞

Starting with node ①

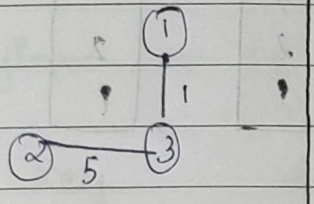
E	K	dv	pv
1	T	0	-
2		6	1
3	T	1	1
4		5	1
5			
6			



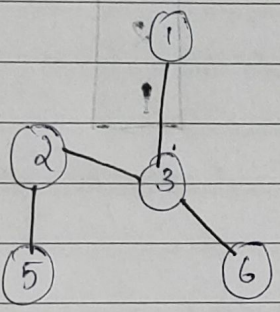
21BCP110

DOMS | Page No.
Date / /

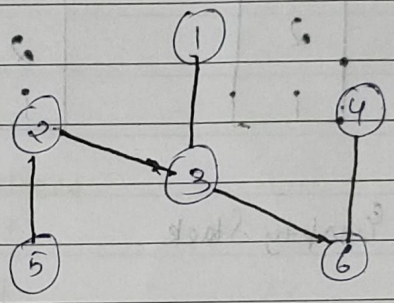
E	K	d_v	p_v
1	T	0	-
2	T	5	3
3	T	1	1
4			
5		3	2
6			



E	K	d_v	p_v
1	T	0	-
2	T	5	3
3	T	1	1
4			
5	T	3	2
6		4	3



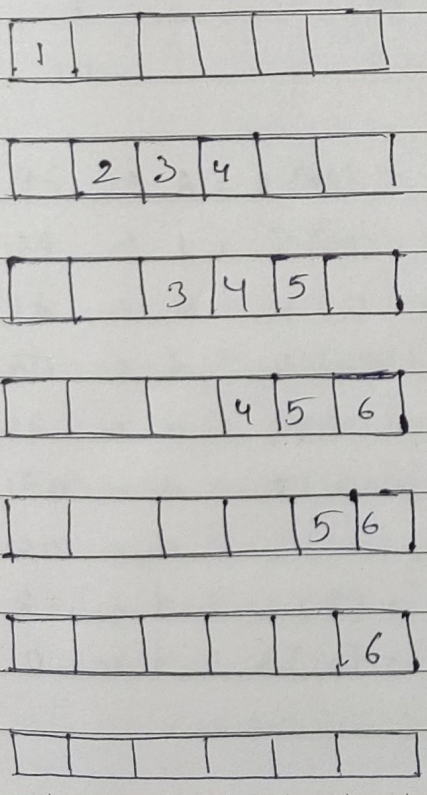
E	K	d_v	p_v
1	T	0	-
2	T	5	3
3	T	1	1
4	T	2	6
5	T	3	2
6	-	-	-

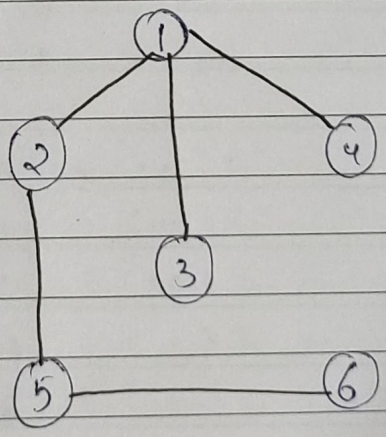


MST : 15

2) Find BFS and DFS of above graph (Choose ①)

(1) BFS

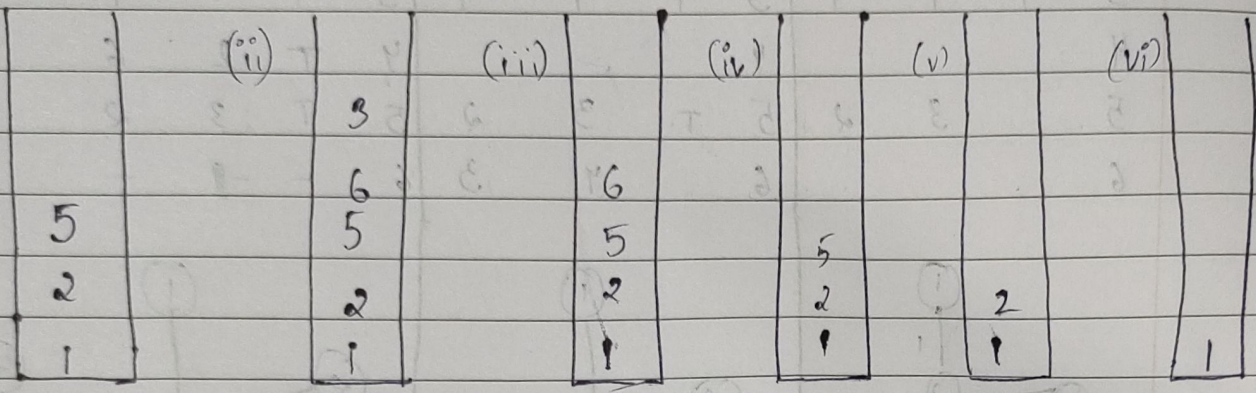




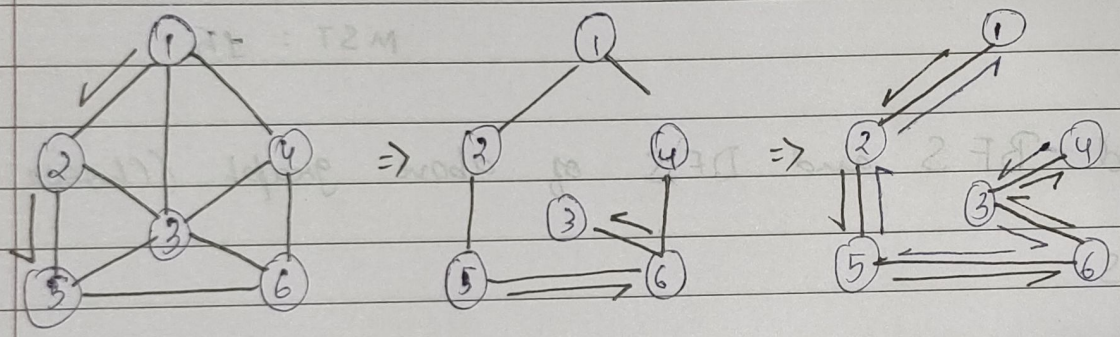
(2) DFS

Stack

(i) (ii) (iii) (iv) (v) (vi)



(vii) Empty Stack



21BCP110

TOPIC 8 : Hashing

1) Insert following keys by creating hash-table of size $m = 15$ and write the no. of probes for each key. Consider $h(K) = (2K+3) \bmod m$, where quadratic probing is used for collision resolution.
Keys : 45, 29, 16, 50, 25, 150, 120, 8, 10

Given $m = 15$, $h(K) = (2K+3) \bmod m$

Key	Index (Location)	Probes
45	3	1
29	1	1
16	5	1
50	13	1
25	8	1
150	4	2
120	7	3
8	9	2
10	10	1

i) $45 \rightarrow h = 2(45) + 3 \% 15 = 3$
ii) $29 \rightarrow h = 2(29) + 3 \% 15 = 1$
iii) $16 \rightarrow h = 2(16) + 3 \% 15 = 5$
iv) $50 \rightarrow h = 2(50) + 3 \% 15 = 13$
v) $25 \rightarrow h = 2(25) + 3 \% 15 = 8$
vi) $150 \rightarrow h = 2(150) + 3 \% 15 = 3$
vii) $120 \rightarrow h = 2(120) + 3 \% 15 = 3$
viii) $8 \rightarrow h = 2(8) + 3 \% 15 = 8$
ix) $10 \rightarrow h = 2(10) + 3 \% 15 = 10$

DOMS | Page No.
Date / /

2) Insert following keys, size $m = 24$ and no. of probes for each key. Consider $h_1(K) = (2K+1) \bmod m$ and $h_2 = (3+K) \bmod m$.
Keys: 122, 24, 99, 36, 100, 50, 144, 120, 88, 10

Given, $m = 24$; $h_1(k) = (2K+1) \bmod m$
$h_2(K) = (3+K) \bmod m$

Key :

Key	Index (Location)	Probe
122	5	1
24	1	1
99	7	1
36	16	2
100	8	1
50	10	2
144	0	1
120	3	2
88	19	2
10	23	3

i) $122 \to 2(122) + 1 \% 24 = 5$
ii) $24 \to 2(24) + 1 \% 24 = 1$
iii) $99 \to 2(99) + 1 \% 24 = 7$
iv) $36 \to 2(36) + 1 \% 24 = 1 \Rightarrow (3+36) \% 24 = 15$ 16
v) $100 \to 2(100) + 1 \% 24 = 8+1 = 9$
vi) $50 \to 2(50) + 1 \% 24 = 5 \Rightarrow (3+50) \% 24 = 5$ 10
vii) $144 \to 2(144) + 1 \% 24 = 0$
viii) $120 \to 2(120) + 1 \% 24 = 0 \Rightarrow (3+120 \% 24) = 3$ 3
ix) $88 \to 2(88) + 1 \% 24 = 9 \Rightarrow (3+88 \% 24) = 10$ 19
x) $10 \to 2(10) + 1 \% 24 = 10 \Rightarrow (3+10 \% 24) = 13$ 23