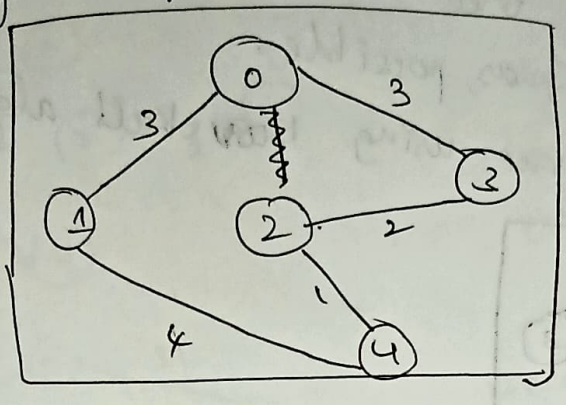
24/01/2023

Important Questions

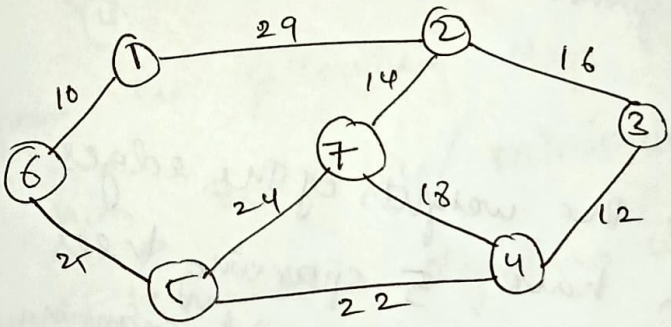
1) what is a minimum spanning Tree & find out MST by using Kruskals Algorithm



2) what is an AVL Tree ? construct an AVL tree having the following elements.

H, I, J, B, A, E, C, F, D, G, K, L

3) construct the minimum spanning tree (MST) for the given graph, using Prim's Algorithm.



4) Insertion sorting

5) BFA

6) DFA

7) Overflow handling Techniques.

8) what is Max heap ? Eg (min heap).

9) Explain adjacency matrix with help of suitable examp.

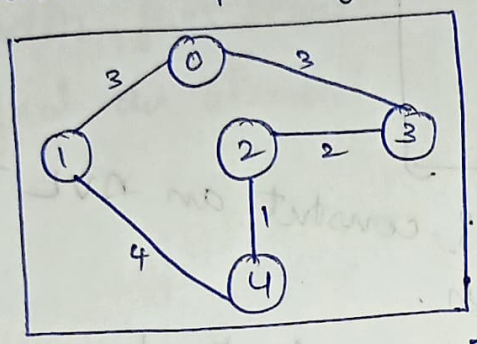
10) Hashing.

11) Diff b/w single Linked list & Double Linked list.

1Ans) A spanning Tree is a subset of a graph G which has all the vertices covered with minimum possible number of edges.

it is a spanning tree whose sum of edges weights is as small as possible.

Find Minimum spanning Tree using Kruskal algorithm



Here, no. of vertex = 5 = n

no. of edges = n - 1

= 5 - 1

edges = 4

from kruskal's algorithm we can define minimum spanning tree

Algorithm:

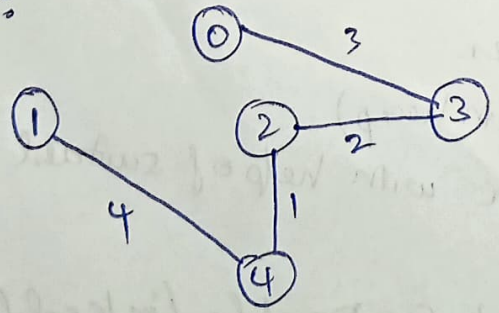
First we have see the weights of the edges.

from this we can have 5 spanning trees.

we have to choose minimum weight spanning tree.

STEPS:

1)

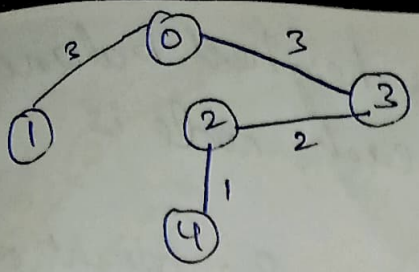


weight = 10

vertex = 4

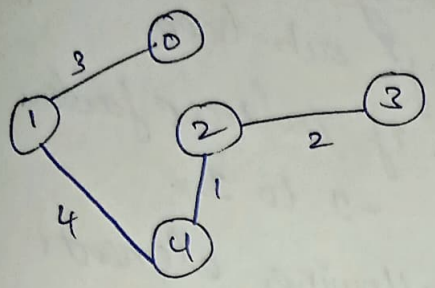
edge = 4

2)



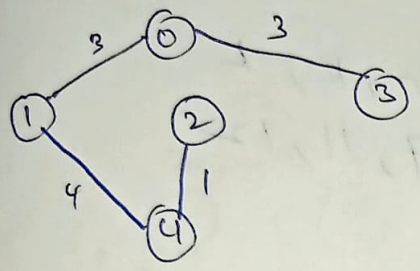
weight = 9
vertex = 5
edges = 4

3)



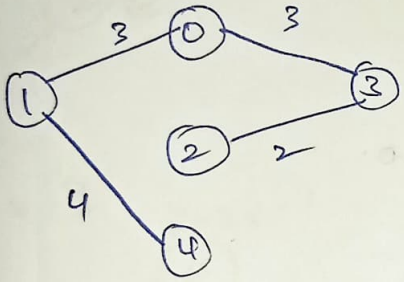
weight = 10.
vertex = 5
edges = 4

4)



weight = 11.
vertex = 5
edge = 4

5)



weight = 12
vertex = 5
edge = 4

Now from the following steps we can see the step 2 has the less/minimum weight = 9.

so, from Kruskal's Algorithm we can say.

STEP-2 is the minimum spanning Tree.

2Aw)

AVL tree can be defined as height balance Binary search tree in which each node is associated with balance factor. which is calculated by subtracting the height of its right sub tree from the top left sub-tree. Tree is said to be balanced if balance factor of each node is in between -1 to 1.

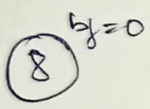
construct an AVL tree for the following elements

H, I, J, B, A, E, C, F, D, G, K, L

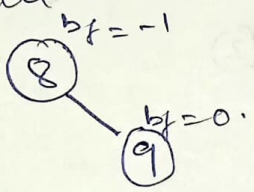
8, 9, 10, 2, 1, 5, 3, 6, 4, 7, 11, 12

STEPS :

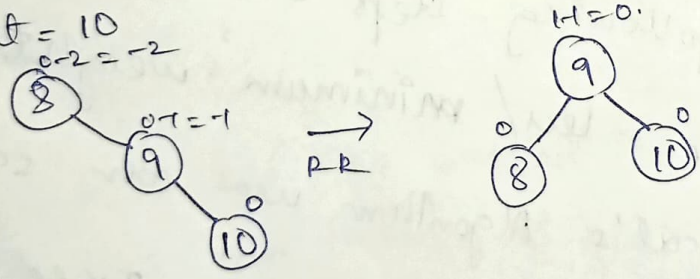
I) Insert = 8

1) 

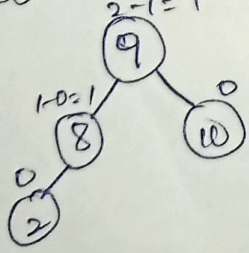
2) Insert = 9.



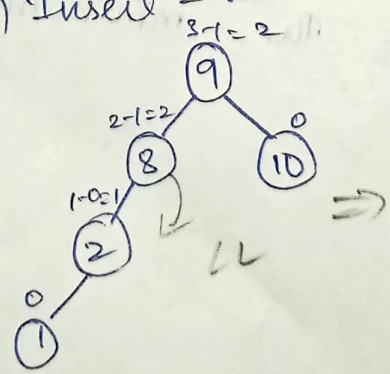
3) Insert = 10

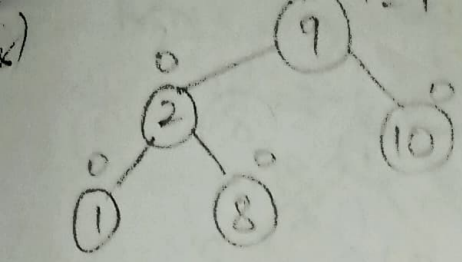


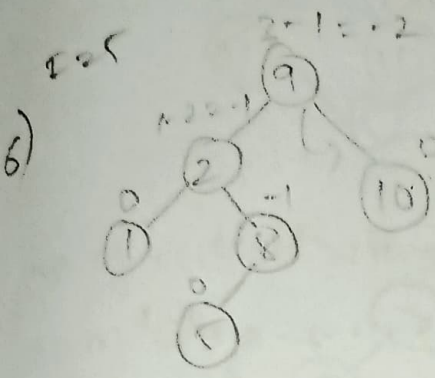
4) Insert = 2

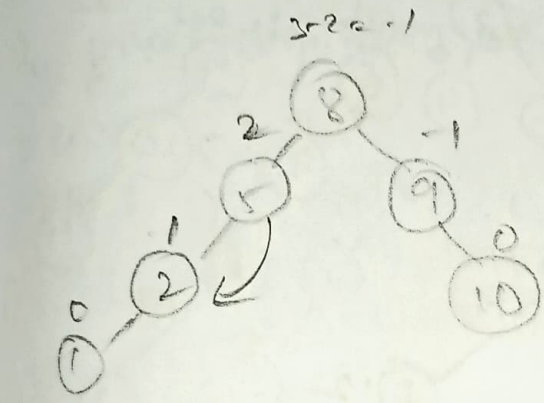


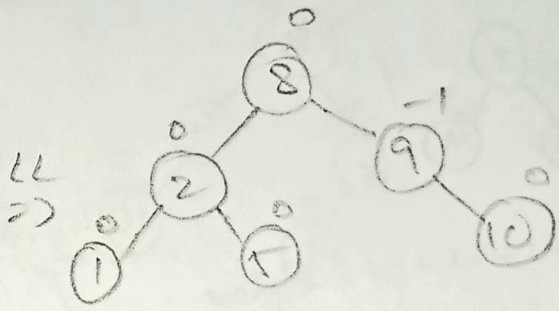
5) Insert = 1

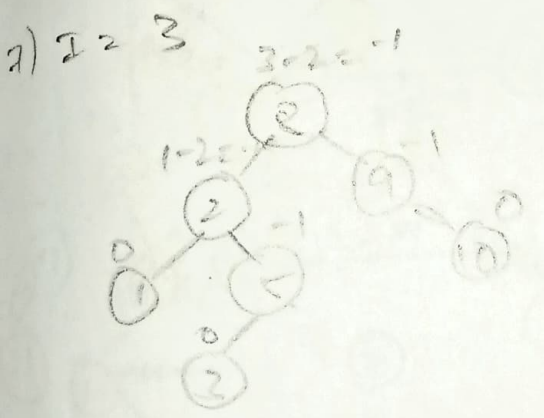


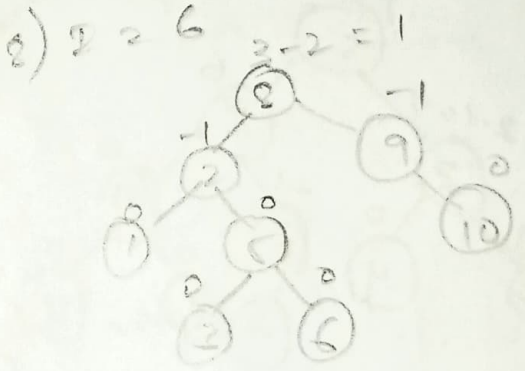


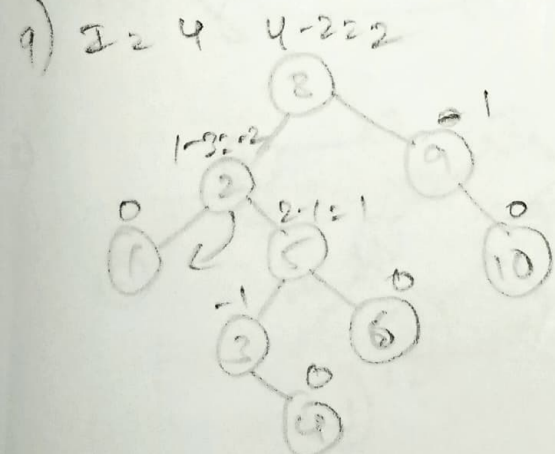


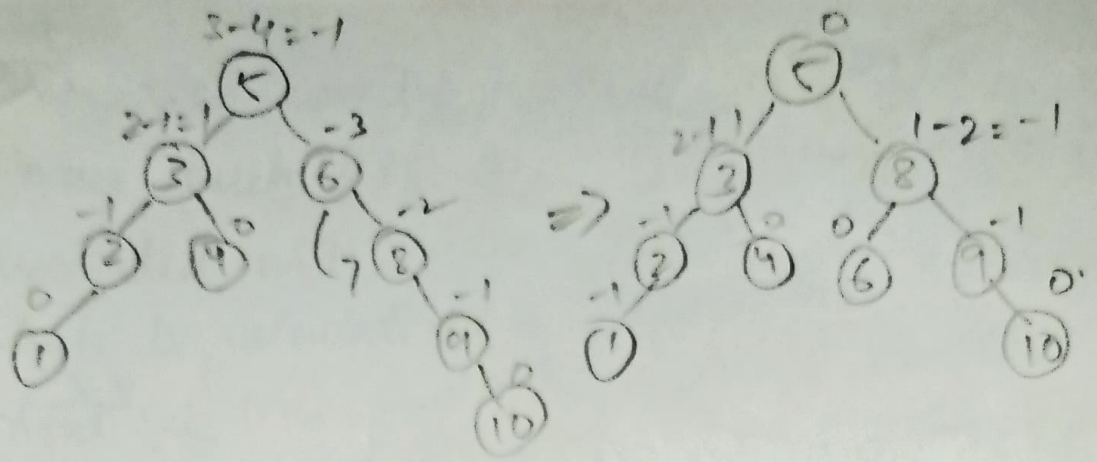




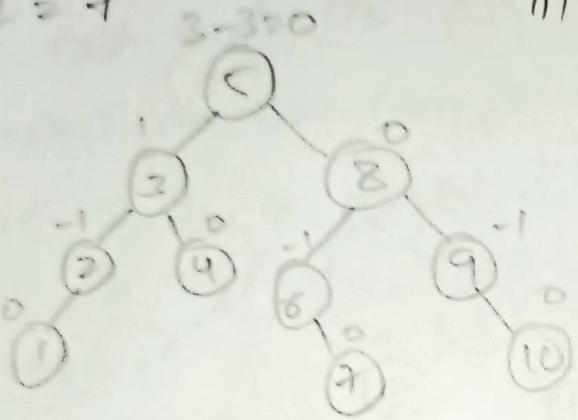




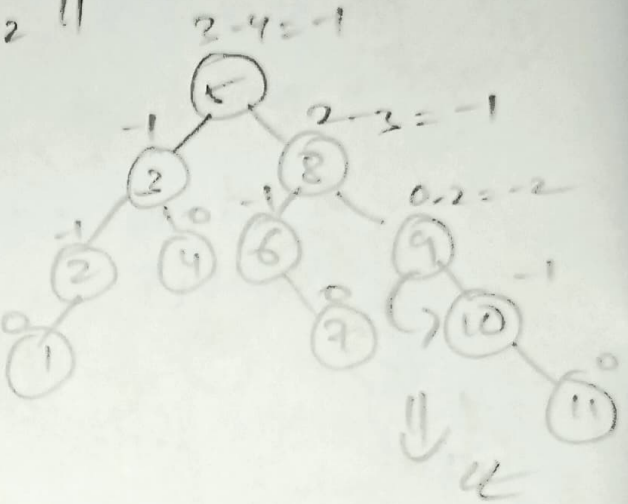




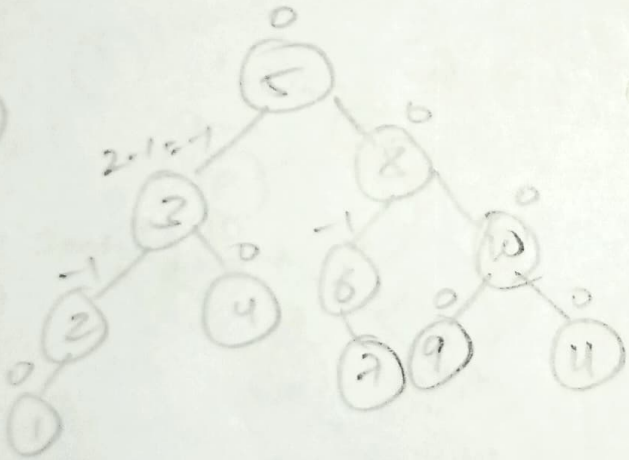
10) I = 7



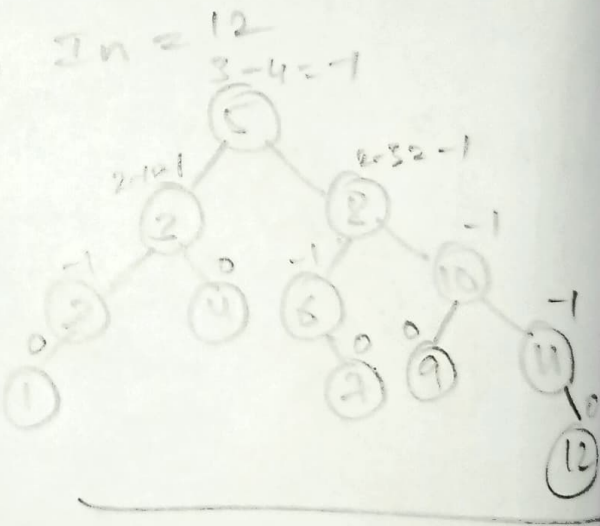
11) I = 11

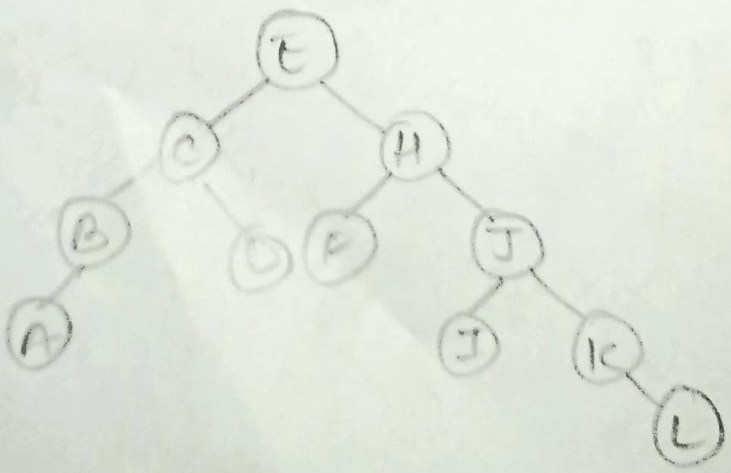


12)

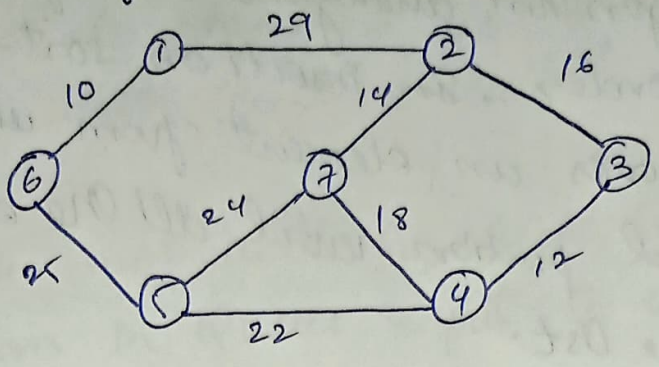


13) In = 12





2Ans) Prim's Algorithm.



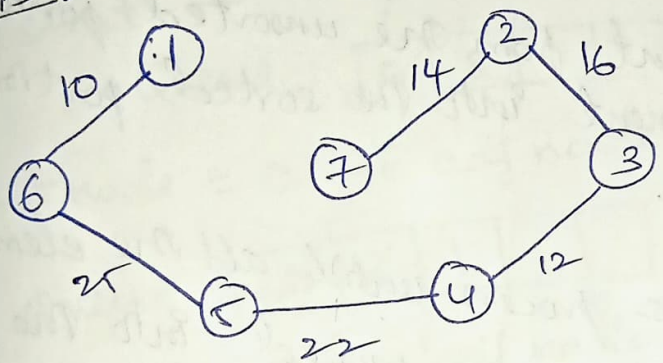
no of vertices = 7 = n

no of edges = n-1 = 7-1 = 6

By prim's algorithm we have to clear Minimum spanning tree.

STEPS:

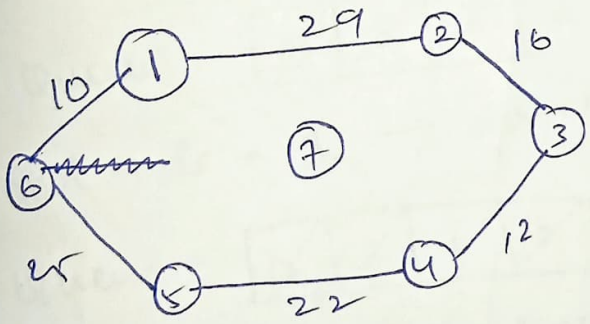
1)



weight = 10 + 25 + 14 + 16 + 12 + 22

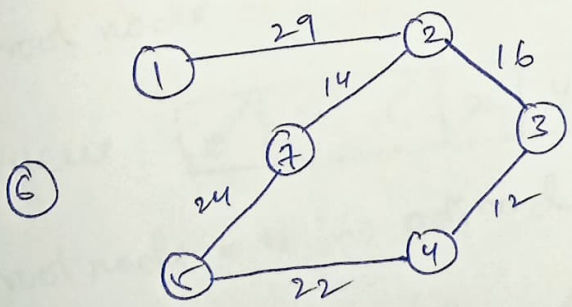
= 99.

2)



weight = 114.

3)



weight = 117

4Ans)

Insertion sort algorithm arranges a list of elements in a particular order. In insertion sort algorithm, every iteration moves an element from unsorted portion to sorted portion until all the elements are sorted in the list.

Algorithm:

Step1: Assume that first element in the list is in sorted portion and all the remaining elements are in unsorted portion.

step2: Take first element from the unsorted portion and insert that element into the sorted portion in the order specified.

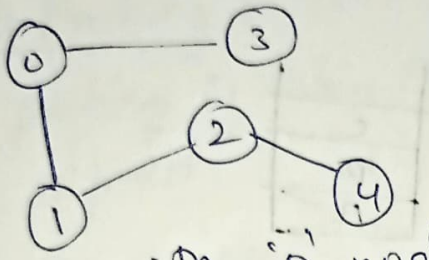
step3: Repeat the above process until all the elements from the unsorted portion are moved into the sorted portion.

5AM) **Breadth First Search (BFS)**

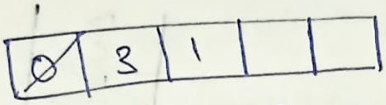
It is used to search a vertex in a graph
It uses data structure called Queue (FIFO)

Rules:
Take the root node & check its adjacent nodes
& put them in Queue & put it on result.

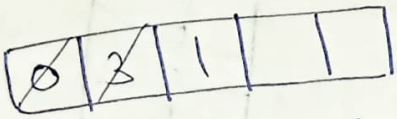
Example: BFS for the graph.



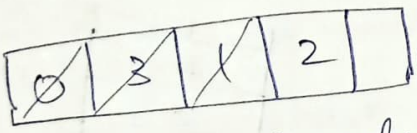
=>(i) start with '0' node.

Queue 

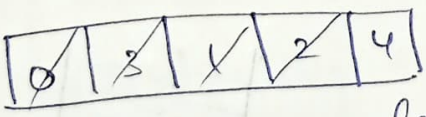
root node = 0 → adj nodes = 3, 1

(ii) Queue 

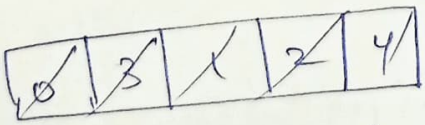
root node = 3 → adj nodes = 0

(iii) Queue 

root node = 1 → adj node = 2

(iv) Queue 

root node = 2 → adj node = 1, 4

(v) Queue 

root node = 4 → adj node = no nodes.

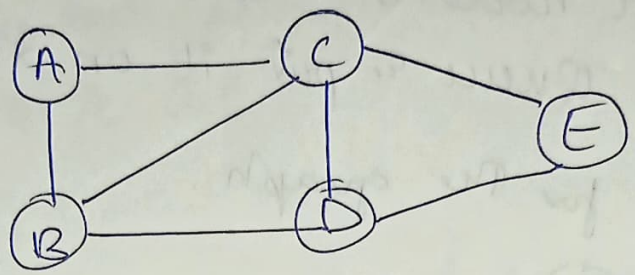
Traversal / Result = 0 3 1 2 4

Q Aw

Depth First Search – DFS

It uses Data Structure "stack" (LIFO).

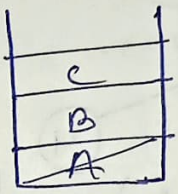
Example: DFS for the following graph.



Steps:

1) A as root node.

result : A

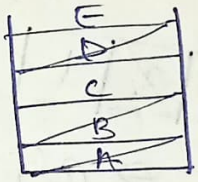


2) B as root node

result : AB

3) D as root node

result : ABD

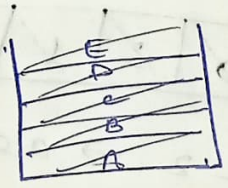


4) E as root node

result : ABDE

5) C as root node

result : ABDEC



Overflow Handling Techniques

1) **Chaining:**

Chaining is also called as closed addressing and open hashing. Separate chaining is one of the most commonly used collision resolution technique. It is usually implemented using linked lists.

2) **Linear Probing:**

Linear probing is when the interval b/w successive probes is fixed. Let's assume that the hashed index for a particular entry is index. The probing sequence for linear probing will be:

index = index % hash table size

index = (index + 1) % hash table size

index = (index + 2) % hash table size

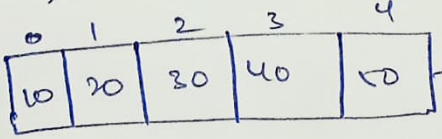
3) **Double Hashing:**

It is a technique in which 2 hash functions are used, the first hash function if is normal hash function if the collision occurs then second hash function is applied. The second hash function given the number of positions to move from point of collision.

$H_1(key) = key \% D$

$H_2(key) = m - (key \% m)$

4) **Rehashing:**



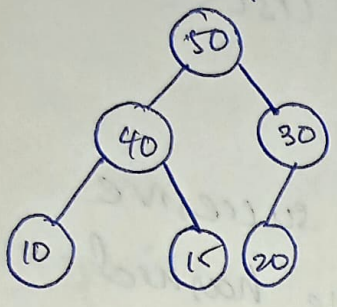
Max-Heap:

Always greater that's its child node i.e root node is the largest among all other nodes.

$$parent[i] > A[i]$$

Example:

Max Heap for the given:

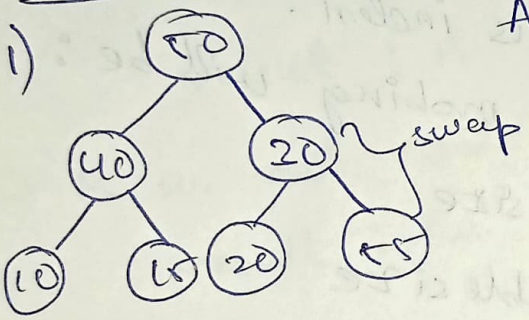


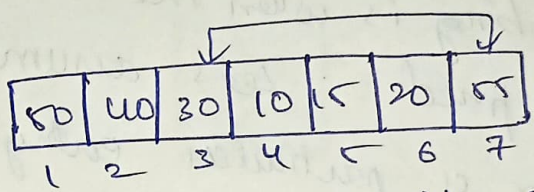
Now insert an greater number than the parent node.

Insert = 55

Steps:

1)



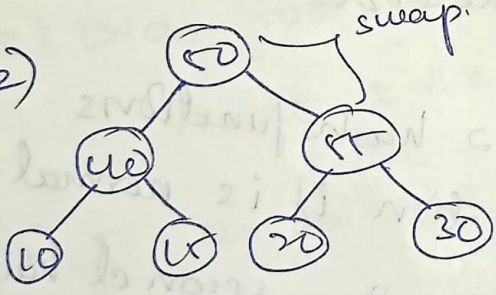


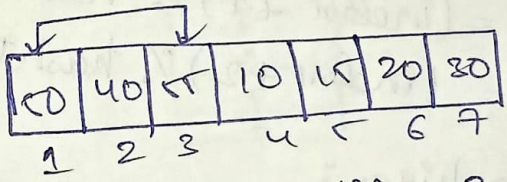
formula = $\left(\frac{i}{2}\right) = \frac{7}{2} = 3.5$

55 → root node.

30 parent node swap 30, 55

2)

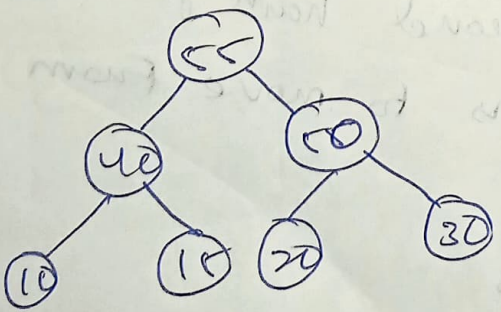


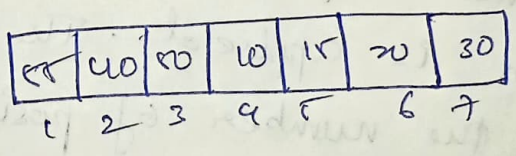


formula = $\left(\frac{i}{2}\right) = \frac{3}{2} = 1.5$

swap — 50, 55

3)



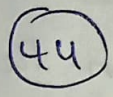


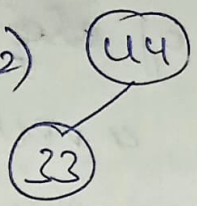
Example:

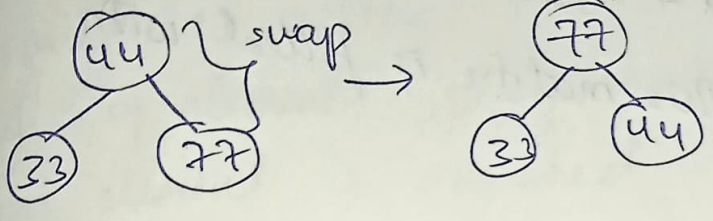
max Heap for the elements

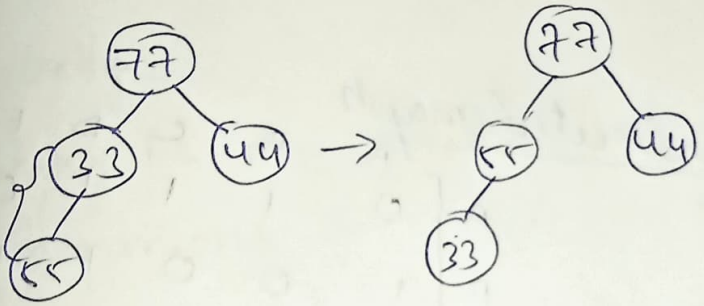
Insert the elements : 44, 33, 77, 55, 88, 66.

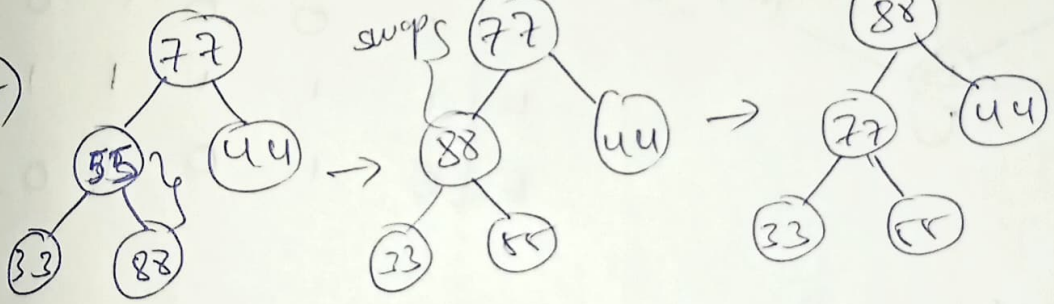
STEPS:

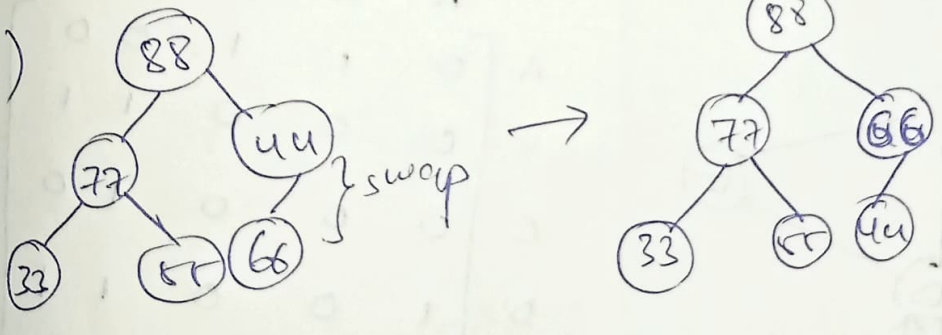
1) 

2) 

3) 

4) 

5) 

6) 

Ans)

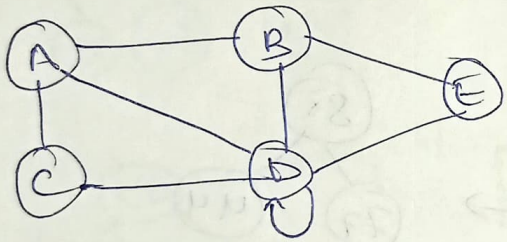
Adjacency Matrix:

In this representation, the graph is represented using a matrix of size i.e total number of vertices by total number of vertices.

That means a graph has 4 vertices then the matrix size is 4×4. In this both row & column represent vertices. This matrix is filled with either 1 or 0.

Example:

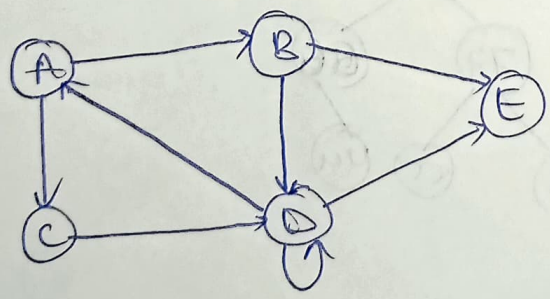
consider the following undirected graph



⇒

	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

Now, Directed graph.



	A	B	C	D	E
A	0	1	1	0	0
B	0	0	0	1	1
C	0	0	0	1	0
D	1	0	0	1	1
E	0	0	0	0	0

10/Aug

Hashing is the process of indexing & retrieving element (data) in a data structure to provide a faster way of finding the element using a hash key.

→ Hashing is another approach in which time required to search an element doesn't depend on the total number of elements. Using hashing data structure, a given element is searched with constant time complexity.

→ Hashing is an effective way to reduce the number of comparisons to search an element in a DS.

IIAns)

SINGLE LINKED LIST	DOUBLE LINKED LIST
1) A Linked list that contains nodes which have a data field and a next field which points to the next node in the line of nodes.	1) A linked list that contains the data field, next field that points to the next node & a previous field that points to the previous node in the sequence.
2) Allows traversing in one direction through the elements	2) Allows traversing in both directions (back & forward)
3) Requires less memory as it stores only one address.	3) Requires more memory as it stores two address.
4) complexity of insertion and deletion at a known position is O(n).	4) complexity of insertion & deletion at a known position is O(1)
5) SLL is prefered when we have memory limitation & searching is not required.	5) DLL is prefered when we don't have memory limitation & searching is required.