ALL NOTES CLICK
CAT-2
(8 marks ques)
(TREES)
1. Write short notes on:-

i. Binomial heaps

A binomial heap can be defined as the collection of binomial trees that satisfies the heap properties, i.e., min-heap. The min-heap is a heap in which each node has a value lesser than the value of its child nodes. Mainly, Binomial heap is used to implement a priority queue. It is an extension of binary heap that gives faster merge or union operations along with other operations provided by binary heap.

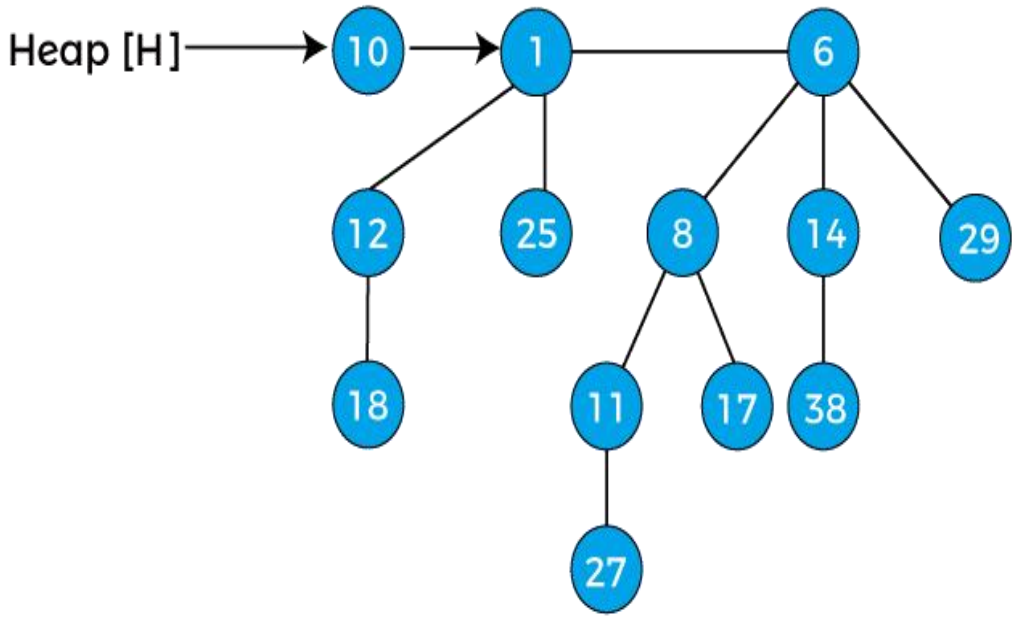
Properties of Binomial heap

There are following properties for a binomial heap with **n** nodes -

- Every binomial tree in the heap must follow the **min-heap** property, i.e., the key of a node is greater than or equal to the key of its parent.
- For any non-negative integer k, there should be atleast one binomial tree in a heap where root has degree k.

The first property of the heap ensures that the min-heap property is hold throughout the heap. Whereas the second property listed above ensures that a binary tree with **n** nodes should have at most **1 + $\log_2$ n** binomial trees, here **$\log_2$** is the binary logarithm.

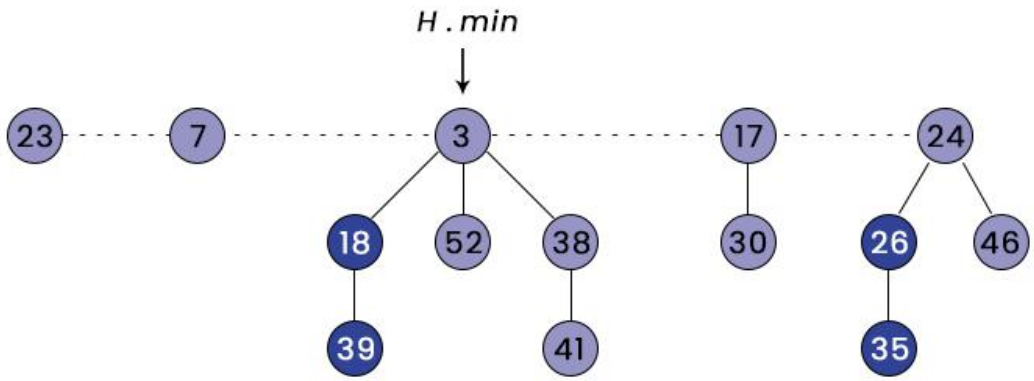
We can understand the properties listed above with the help of an example -



ii. Fibonacci heaps

Fibonacci Heap - A Fibonacci heap is defined as the collection of rooted-tree in which all the trees must hold the property of Min-heap. That is, for all the nodes, the key value of the parent node should be greater than the key value of the parent node:

The given figure is an example of the Fibonacci tree:



2. Explain the tree traversal techniques with an example.

Traversal is a process to visit all the nodes of a tree and may print their values too. Because, all nodes are connected via edges (links) we always start from the root (head) node. That is, we cannot randomly access a node in a tree. There are three ways which we use to traverse a tree –

- In-order Traversal
- Pre-order Traversal

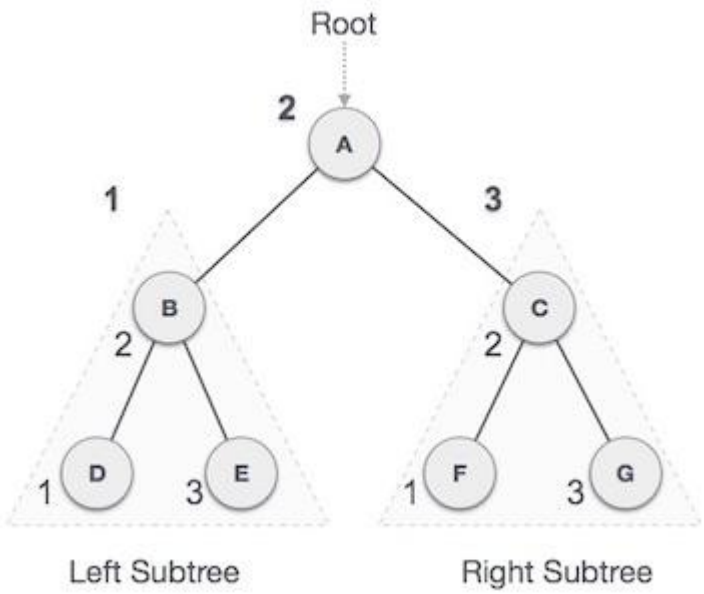
- Post-order Traversal

Generally, we traverse a tree to search or locate a given item or key in the tree or to print all the values it contains.

In-order Traversal

In this traversal method, the left subtree is visited first, then the root and later the right sub-tree. We should always remember that every node may represent a subtree itself.

If a binary tree is traversed **in-order**, the output will produce sorted key values in an ascending order.



We start from **A**, and following in-order traversal, we move to its left subtree **B**. **B** is also traversed in-order. The process goes on until all the nodes are visited. The output of inorder traversal of this tree will be –

$$D \rightarrow B \rightarrow E \rightarrow A \rightarrow F \rightarrow C \rightarrow G$$

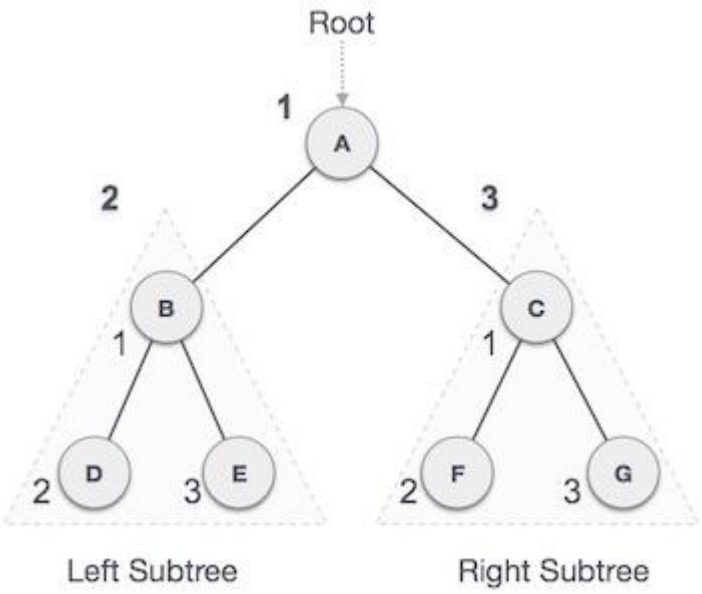
Algorithm

Until all nodes are traversed –
Step 1 – Recursively traverse left subtree.
Step 2 – Visit root node.
Step 3 – Recursively traverse right subtree.

Pre-order Traversal

In this traversal method, the root node is visited first, then the left subtree

and finally the right subtree.



We start from **A**, and following pre-order traversal, we first visit **A** itself and then move to its left subtree **B**. **B** is also traversed pre-order. The process goes on until all the nodes are visited. The output of pre-order traversal of this tree will be –

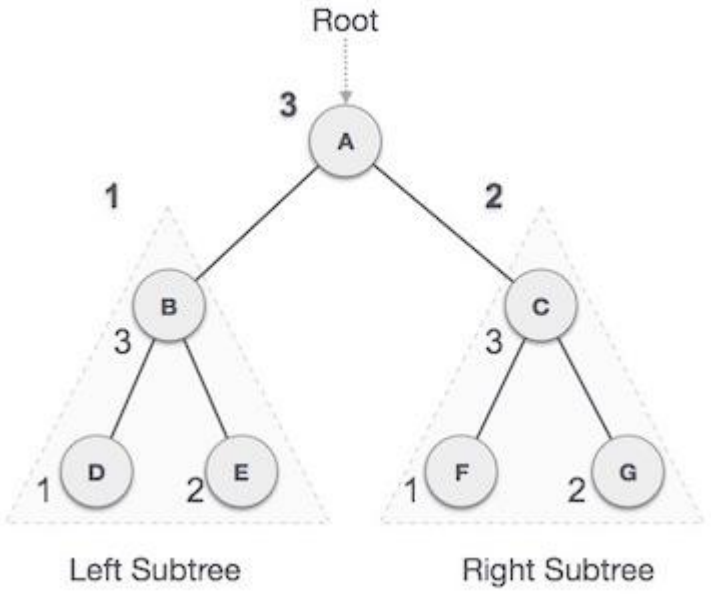
$$A \rightarrow B \rightarrow D \rightarrow E \rightarrow C \rightarrow F \rightarrow G$$

Algorithm

Until all nodes are traversed –
Step 1 – Visit root node.
Step 2 – Recursively traverse left subtree.
Step 3 – Recursively traverse right subtree.

Post-order Traversal

In this traversal method, the root node is visited last, hence the name. First we traverse the left subtree, then the right subtree and finally the root node.



We start from **A**, and following Post-order traversal, we first visit the left subtree **B**. **B** is also traversed post-order. The process goes on until all the nodes are visited. The output of post-order traversal of this tree will be –

$$D \rightarrow E \rightarrow B \rightarrow F \rightarrow G \rightarrow C \rightarrow A$$

Algorithm

Until all nodes are traversed –
Step 1 – Recursively traverse left subtree.
Step 2 – Recursively traverse right subtree.
Step 3 – Visit root node.

3 .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.

4. How to insert and delete an element into a binary search tree and write down the code for the insertion routine with an example.

1. Allocate the memory for tree.
2. Set the data part to the value and set the left and right pointer of tree, point to NULL.
3. If the item to be inserted, will be the first element of the tree, then the left and right of this node will point to NULL.
4. Else, check if the item is less than the root element of the tree, if this is true, then recursively perform this operation with the left of the root.
5. If this is false, then perform this operation recursively with the right sub-tree of the root.

```
#include<stdio.h>
#include<stdlib.h>
void insert(int);
struct node
{
    int data;
    struct node *left;
    struct node *right;
};
struct node *root;
void main ()
{
    int choice,item;
    do
    {
        printf("\nEnter the item which you want to insert?\n");
        scanf("%d",&item);
        insert(item);
        printf("\nPress 0 to insert more ?\n");
        scanf("%d",&choice);
    }while(choice == 0);
}
void insert(int item)
{
    struct node *ptr, *parentptr , *nodeptr;
    ptr = (struct node *) malloc(sizeof (struct node));
    if(ptr == NULL)
    {
        printf("can't insert");
    }
    else
    {
    ptr -> data = item;
    ptr -> left = NULL;
    ptr -> right = NULL;
    if(root == NULL)
    {
```

```
42.         root = ptr;
43.         root -> left = NULL;
44.         root -> right = NULL;
45.     }
46.     else
47.     {
48.         parentptr = NULL;
49.         nodeptr = root;
50.         while(nodeptr != NULL)
51.         {
52.             parentptr = nodeptr;
53.             if(item < nodeptr->data)
54.             {
55.                 nodeptr = nodeptr -> left;
56.             }
57.             else
58.             {
59.                 nodeptr = nodeptr -> right;
60.             }
61.         }
62.         if(item < parentptr -> data)
63.         {
64.             parentptr -> left = ptr;
65.         }
66.         else
67.         {
68.             parentptr -> right = ptr;
69.         }
70.     }
71.     printf("Node Inserted");
72.     }
73. }
```

Deletion

```
void deletion(Node*& root, int item)
{
    Node* parent = NULL;
```

```
Node* cur = root;

search(cur, item, parent);
if (cur == NULL)
   return;

if (cur->left == NULL && cur->right == NULL)
{
   if (cur != root)
   {
      if (parent->left == cur)
         parent->left = NULL;
      else
         parent->right = NULL;
   }
   else
      root = NULL;

   free(cur);
}
else if (cur->left && cur->right)
{
   Node* succ  = findMinimum(cur- >right);

   int val = succ->data;

   deletion(root, succ->data);

   cur->data = val;
}

else
{
   Node* child = (cur->left)? Cur- >left: cur->right;

   if (cur != root)
   {
```

```
            if (cur == parent->left)
                parent->left = child;
            else
                parent->right = child;
        }

        else
            root = child;
        free(cur);
    }
}

Node* findMinimum(Node* cur)
{
    while(cur->left != NULL) {
        cur = cur->left;
    }
    return cur;
}
```

5. What are threaded binary tree? Write an algorithm for inserting a node in a threaded binary tree.

The idea of threaded binary trees is to make inorder traversal faster and do it without stack and without recursion. A binary tree is made threaded by making all right child pointers that would normally be NULL point to the inorder successor of the node (if it exists).

There are two types of threaded binary trees.

Single Threaded: Where a NULL right pointers is made to point to the inorder successor (if successor exists)

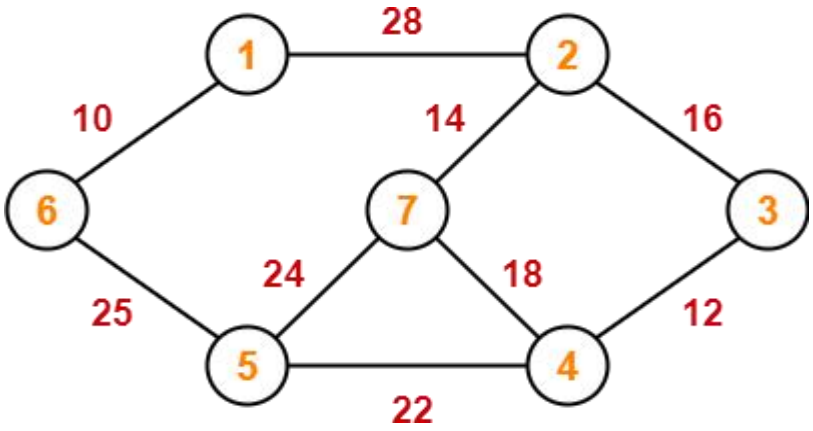
Double Threaded: Where both left and right NULL pointers are made to point to inorder predecessor and inorder successor respectively.

2nd part previous

6. Create a binary search tree for the following numbers start from an empty binary search tree.

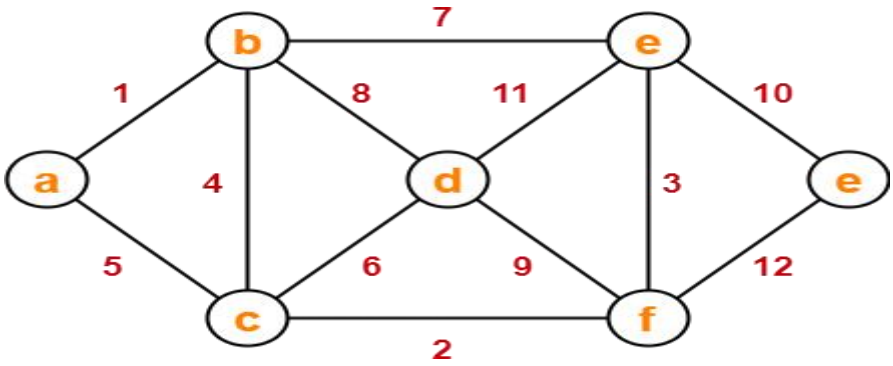
45,26,10,60,70,30,40 Delete keys 10,60 and 45 one after the other and show the trees at each stage

UNIT-2
(GRAPHS)
1. Explain Shortest Path algorithm: Dijikstra Algorithm.
2. a)Explain prims algorithm along with its implementation.
b) Construct the minimum spanning tree (MST) for the given graph using prims algorithm.



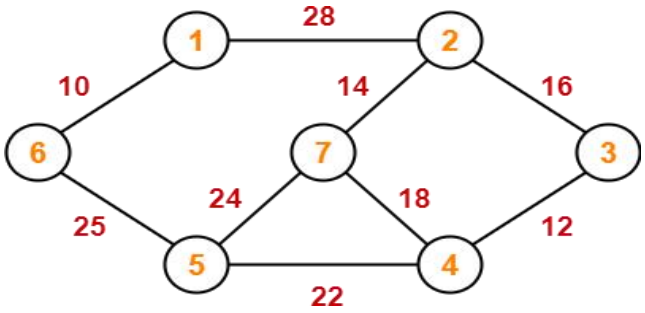
3.a) Explain Prims Algorithm along with its implementation.

b) Using Prim's Algorithm, find the cost of minimum spanning tree (MST) of the given graph-



4.a)Explain Kruskal algorithm along with its implementation.

b)Construct the minimum spanning tree (MST) for the given graph using Kruskal's Algorithm.



5 MARKS QUESTIONS

UNIT-3

1. Explain the logic behind using "Threaded binary tree" in Data Structures. Draw a labelled diagram to show working of threaded binary tree.

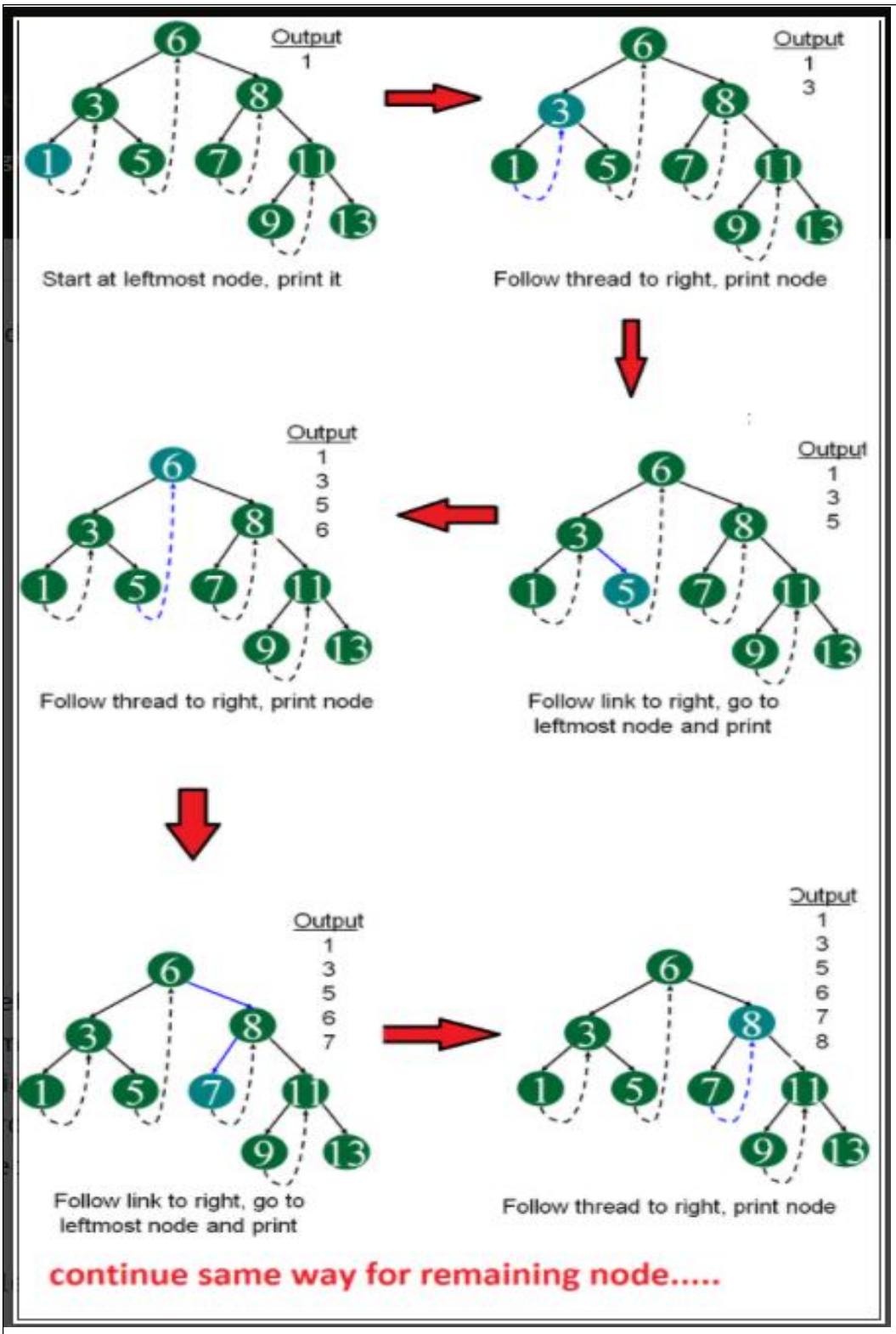
the idea of threaded binary trees is to make inorder traversal faster and do it without stack and without recursion. A binary tree is made threaded by making all right child pointers that would normally be NULL point to the inorder successor of the node (if it exists).

There are two types of threaded binary trees.

Single Threaded: Where a NULL right pointers is made to point to the inorder successor (if successor exists)

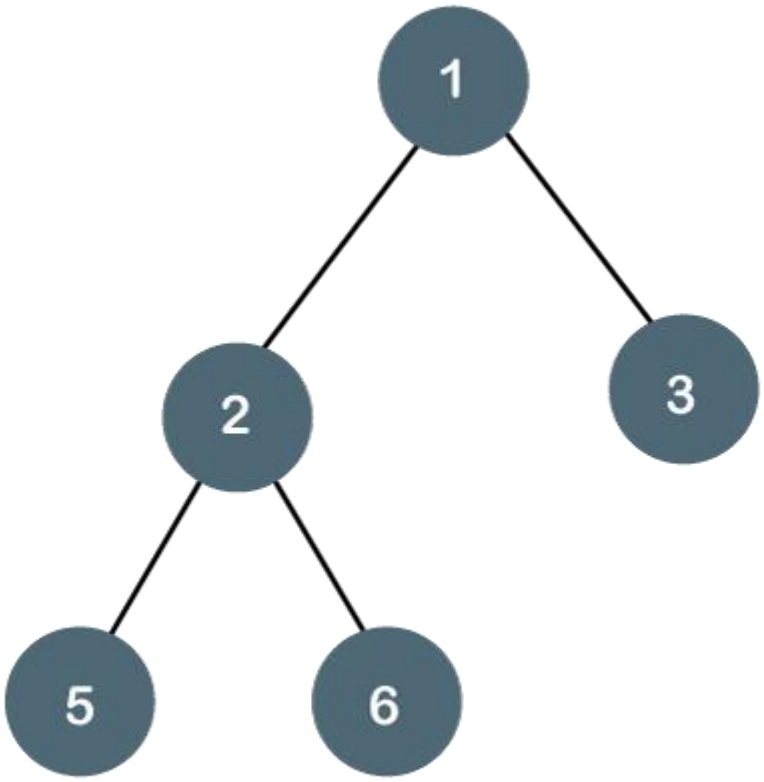
Double Threaded: Where both left and right NULL pointers are made to point to inorder predecessor and inorder successor respectively.

The predecessor threads are useful for reverse inorder traversal and postorder traversal.



2. Describe Binary Tree along with its representation. How will you search an element in Binary Tree. Explain.

The Binary tree means that the node can have maximum two children. Here, binary name itself suggests that 'two'; therefore, each node can have either 0, 1 or 2 children.



The above tree is a binary tree because each node contains the utmost two children. The logical representation of the above tree is given below:

1. Compare the element with the root of the tree.
2. If the item is matched then return the location of the node.
3. Otherwise check if item is less than the element present on root, if so then move to the left sub-tree.
4. If not, then move to the right sub-tree.
5. Repeat this procedure recursively until match found.
6. If element is not found then return NULL.

Algorithm:

Search (ROOT, ITEM)

- **Step 1:** IF ROOT -> DATA = ITEM OR ROOT = NULL
  Return ROOT
  ELSE
  IF ROOT < ROOT -> DATA
  Return search(ROOT -> LEFT, ITEM)

ELSE

Return search(ROOT -> RIGHT,ITEM)

[END OF IF]

[END OF IF]

- **Step 2:** END

3. The Inorder and Preorder Traversal of a Tree are given below:-

INORDER: DBMINEAFCJGK

PREORDER: ABDEIMNCGJK

i) Construct the corresponding Binary Tree.

ii) Determine Post order Traversal of the Tree drawn.

4. Explain the following:-

i) Binary Tree & Binary Search Tree

ii) Complete Binary Tree

UNIT-4

1. Write Kruskal's Algorithm for finding Minimum Spanning Tree.

2. Outline the distinguishing features of Depth First Search (DFS) and Breadth First Search (BFS)

in context of graphs.

3. Explain Adjacency Matrix with the help of suitable diagram.

4. How will you detect a cycle in a Directed as well as Undirected graph. Explain with help of an example.

2 MARKS QUESTIONS

UNIT 3

1. What are various representation of Binary Tree.

A binary tree data structure is represented using two methods. Those methods are as follows...

1. Array Representation
2. Linked List Representation

1. Array Representation of Binary Tree

In array representation of a binary tree, we use one-dimensional array (1-D Array) to represent a binary tree.

Consider the above example of a binary tree and it is represented as follows...

To represent a binary tree of depth **'n'** using array representation, we need one dimensional array with a maximum size of **2n + 1**.

2. Linked List Representation of Binary Tree

We use a double linked list to represent a binary tree. In a double linked list, every node consists of three fields. First field for storing left child address, second for storing actual data and third for storing right child address.

In this linked list representation, a node has the following structure...

3. Write Advantages of Threaded Binary Tree.

Advantages of Threaded Binary Tree:

- In threaded binary tree, linear and fast traversal of nodes in the tree so there is no requirement of stack. If the stack is used then it consumes a lot of memory and

time.

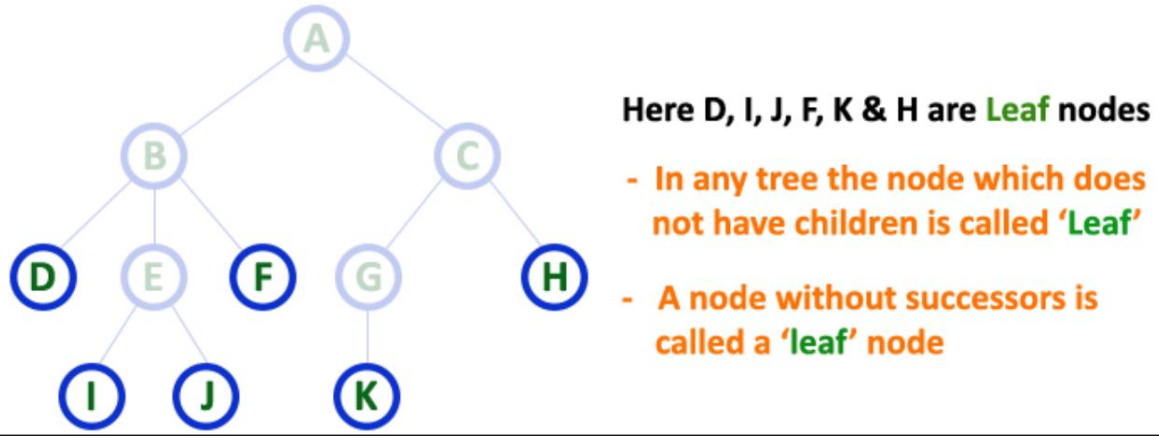
- It is more general as one can efficiently determine the successor and predecessor of any node by simply following the thread and links. It almost behaves like a circular linked list

Advantages of Threaded Binary Tree

- In this Tree it enables linear traversal of elements.
- It eliminates the use of stack as it perform linear traversal.
- Enables to find parent node without explicit use of parent pointer
- Threaded tree give forward and backward traversal of nodes by in-order fashion
- Nodes contain pointers to in-order predecessor and successor

4. Define 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.



5. What is Ordered Tree.

An **ordered tree** is an oriented tree in which the children of a node are somehow ordered. It is a rooted tree in which an ordering is specified for the children of each vertex. This is called a "plane tree" because the ordering of the children is equivalent to an embedding of the tree in the plane, with the root at the top and the children of each vertex lower than that vertex.

The ordered trees can be further specified as labelled ordered trees and

unlabelled ordered trees.

2 MARKS QUESTIONS

UNIT-4

1. Define Graph. List Any 3 application area of Graph.

A graph is a non-linear data structure, which consists of vertices(or nodes) connected by edges(or arcs) where edges may be directed or undirected.

In **computer science** graph theory is used for the **study of algorithms** like:

- Dijkstra's Algorithm
- Prims's Algorithm
- Kruskal's Algorithm

2. Electrical Engineering
3.Computer Network

2. Define OutDegree of Graph.

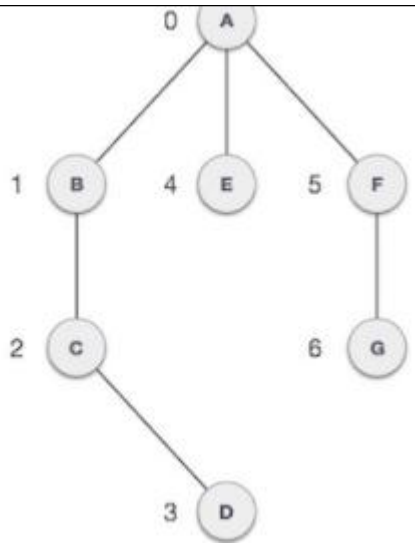
The number of edges going out of a vertex in a directed graph.

3. Define Undirected Graph.

An undirected graph is graph, i.e., a set of objects (called vertices or nodes) that are connected together, where all the edges are bidirectional. An undirected graph is sometimes called an undirected network.

4. Define Adjacent Nodes.

- **Adjacency** - Two node or vertices are adjacent if they are connected to each other through an edge. In the following example, B is adjacent to A, C is adjacent to B, and so on.


0
A
1
B
4
E
5
F
2
C
6
G
3
D

-