UNIT –I

LINEAR STRUCTURES

1. What are the main objectives of Data structure?

- To identify and create useful mathematical entities and operations to determine what classes of problems can be solved by using these entities and operations.
- To determine the representation of these abstract entities and to implement the abstract operations on these concrete representation.

2. Need of a Data structure?

- To understand the relationship of one data elements with the other and organize it within the memory.
- A data structures helps to analyze the data, store it and organize it in a logical or mathematical manner.

3. Define data structure with example.

A data structure is a way of organizing data that considers not only the items stored but also their relationship to each other.

e.g.: Arrays, Records etc.

e.g of complex data structure

Stacks, Queues, Linked list, Trees, Graphs.

4. Define abstract data type and list its advantages.

ADT is a set of operations.

An abstract data type is a data declaration packaged together with the operations that are meaning full on the data type.

Abstract Data Type

1. Declaration of data

2. Declaration of operations.

Advantages:

1. Easy to debug small routines than large ones.
2. Easy for several people to work on a modular program simultaneously.
3. A modular program places certain dependencies in only one routine, making changes easier.

5. What are the two basic types of Data structures?

1. Primitive Data structure

   Eg., int,char,float

2. Non Primitive Data Structure

   i. Linear Data structure

      Eg., Lists Stacks Queues

   ii. Non linear Data structure

      Eg., Trees Graphs

6. What are the four basic Operations of Data structures?

1. Traversing
2. Searching
3. Inserting
4. Deleting

7. List out the different ways to implement the list?

1. Array Based Implementation
2. Linked list Implementation

   i. Singly linked list

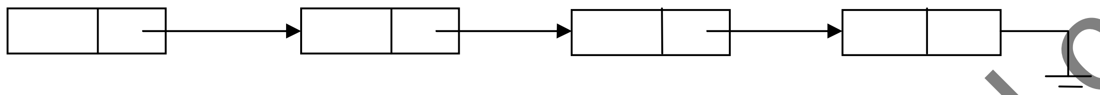
   ii. Doubly linked list

   iii. Cursor based linked list

8. Define singly linked list with neat diagram.

A singly linked list is a collection of nodes each node is a structure it consisting of an element and a pointer to a structure containing its successor, which is called a next pointer.

The last cell's next pointer points to NULL specified by zero.



9. List the operations of single linked list.

MakeEmpty

IsEmpty

IsLast

Find

Delete

FindPrevious

Insert

Deletelist

Header

First

Advance

Retrieve

10. Write the routine for insertion operation of singly linked list.

```
/* Insert (after legal position P)*/

/* Header implementation assumed*/

/* Parameter L is unused in this implementation*/


Void

Insert (ElementType X, List L, Position P)

{

        Position TmpCell;
        TmpCell=malloc(sizeof(struct Node));
        if(TmpCell==NULL)
                FatalError("Out of space!!!");
        TmpCell->Element =X;
        TmpCell->Next=P->Next;
        P->Next=TmpCell;
}
```

11. Write the routine for deletion operation of singly linked list.

```
/* Delete first occurrence of x from a list*/
/* Assume use of a header node*/
Void
Delete(ElementType X, List L)
{
        Position P, TmpCell;
        P=FindPrevious(X, L);
        if (! IsLast(P,L))
        {
                TmpCell=P->Next;
                P->Next=TmpCell->Next;
                Free(TmpCell);
        }
}
```

12. List out the advantages and disadvantages of singly linked list.

Advantages:

1. Easy insertion and deletion.
2. Less time consumption.

Disadvantages:

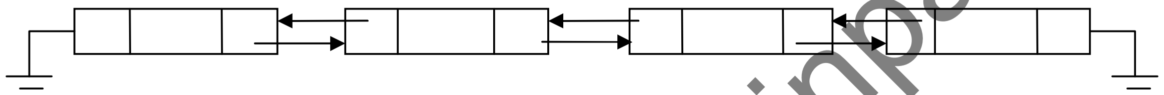
1. Data not present in a linear manner.

2. Insertion and deletion from the front of the list is difficult without the use of header node.

13. Define doubly linked linked list with neat diagram.

Doubly linked list is a collection of nodes where each node is a structure containing the following fields

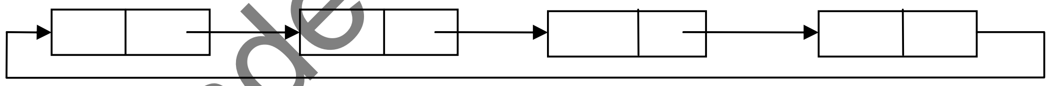
1. Pointer to the previous node.
2. Data.
3. Pointer to the next node.



14. Define circularly linked list with neat diagram.

Circularly linked list is a collection of nodes , where each node is a structure containing the element and a pointer to a structure containing its successor.

The pointer field of the last node points to the address of the first node. Thus the linked list becomes circular.



15. Write the difference between Singly and doubly linked list.

Singly	Doubly
It is a collection of nodes and each node is having one data field and next link field	It is a collection of nodes and each node is having one data field one previous link field and one next link field
The elements can be accessed	The elements can be accessed using

using next link	both previous link as well as next link
No extra field is required hence, Node takes less memory in SLL.	One field is required to store previous Link Hence, node takes memory in DLL.
Less efficient access to elements	More efficient access to elements.

16. Write the difference between doubly and circularly linked list.

Doubly	Circularly
If the pointer to next node is Null, it specifies the last node.	There is no first and last node.
Last node's next field is always Null	Last nodes next field points to the address of the first node.
Every node has three fields one is Pointer to the pervious node, next Is data and the third is pointer to the next node.	it can be a singly linked list and doubly linked list.

17. Write the difference between cursor and pointer implementation of singly linked list.

CURSOR IMPLEMENTATION	POINTER IMPLEMENTATION
Global array is maintained for storing the data elements .The corresponding index value for each data item represents its address.	The data are stored in a collection of structures. Each structure contains data and a pointer to the next structure.
Collection of empty cells is maintained in the form of a array from which cells	A new structure can be obtained from the systems global memory by a call to

can be allocated and also cells can be returned after use.	a *malloc()* and released by a call to *free()*.

18. Define stack ADT with example.

A stack is a list with a restriction that insertions and deletions can be performed in only one position namely the end of the list called the top.

e.g.: undo statement in text editor.

Pile of bills in a hotel.

19. State the operations on stack. Define them and give the diagrammatic representation.

Push

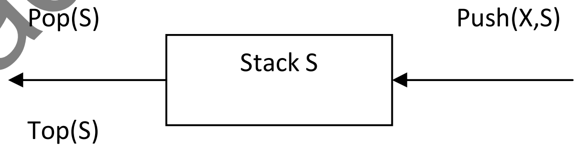
Pop

Top

Push: Push is performed by inserting at the top of stack.

Pop: pop deletes the most recently inserted element.

Top: top operation examines the element at the top of the stack and returns its value.



20. Write the routine for push and pop of linked list.

```
/*routine for push*/

Void

Push(ElementType X,Stack S)
```

```
{
        PtrToNode TmpCell;
        TmpCell=malloc(sizeof(struct Node));
        If(TmpCell==NULL)
                FatalError("out of space!!!");
        else
        {
                TmpCell->Element=x;
                TmpCell->Next=s->Next;
                S->Next=TmpCell;
        }
}

/*Routine for pop*/

Void
Pop(stack S)
{
        PtrToNode FirstCell;
        If(IsEmpty(S))
                Error("Empty stack");
        Else
        {
                FirstCell=S->Next;
                S->Next=S->Next->Next;
                free(firstCell);
        }
}
```

21. What is the purpose of top and pop?

Top operation examines the element in the top of the list and returns its value. Pop operation deletes the element at the top of the stack and decrements the top of the stack pointer by one.

22. State the disadvantages of linked list implementation of stack.

1. Calls to malloc and free functions are expensive.
2. Using pointers is expensive.

23. State the applications of stack.

1. Balancing parentheses.
2. Postfix Expression.
    i. Infix to postfix conversion
3. Function calls.

24. Write the algorithm for balancing symbols.

```
1. Make an empty stack.
2. Read characters until end of file.
3. If the character is an opening symbol, then push it onto the stack.
4. If it is a closing symbol
        Then
                If the stack is empty
                Report an error
        Otherwise pop the stack
5.If the symbol popped is not the corresponding openinig symbol
        Then
                Report an error
6. If the stack is not empty at the end of file
        Then
                Report an error
```

25. Give the Features of balancing symbols.

1. It is clearly linear.
2. Makes only one pass through the input.
3. It is online and quite fast.
4. It must be decided what to do when an error is reported.

26. Convert the given infix to postfix.

(j*k)+(x+y)

Ans:jk* xy++

27. Convert into postfix and evaluate the following expression.

(a+b*c)/d

a=2 b=4 c=6 d=2

Ans:

Post fix:

abc*+d/

Evaluation:

2 4 6 * +2/

=13

28. Write the features of representing calls in a stack.

1. When a function is called the register values and return address are saved.
2. After a function has been executed the register values are resumed on returning to the calling statement.
3. The stack is used for resuming the register values and for returning to the calling statement.

4. The stack overflow leads to fatal error causing loss of program and data.

5. Recursive call at the last line of the program is called tail recursion and also leads to error.

29. Define queue with examples.

Queue is a list in which insertion is done at one end called the rear and deletion is performed at another called front.

e.g: Ticket counter.

Phone calls waiting in a queue for the operator to receive.

30. List the operations of queue.

Two operations

1. Enqueue-inserts an element at the end of the list called the rear.
2. Dequeue-delets and returns the element at the start of the list called as the front.

31. What are the prerequisites for implementing the queue ADT using array?

For each queue data structure the following prerequisites must be satisfied

1. Keep an array queue[ ].
2. The positions front and rear represents the ends of the queue.
3. The number of elements that are actually in the queue is kept trasck of using 'size'.

32. How the enqueue and dequeue operations are performed in queue.

To enqueue an element X:

```
    increment size and rear
    set queue[rear]=x
```

To dequeue an element

```
        set the return value to queue[front].
        Decrement size
        Increment front
```

33. Write the routines for enqueue operation in queue.

```
    Void
    Enqueue(Element type X,Queue Q)
    {
        If(IsFull(Q))
            Error("Full queue");
        Else
        {
            Q->Size++;
            Q->Rear=Succ(Q->Rear,Q);
            Q->Array[Q->Rear]=X;
        }
}
```

34. Write the routines for dequeue operation in queue.

```
    Void Dequeue(Queue Q)
    {
        If(IsEmpty(Q))
            Error("Empty Queue");
        Else
            Q->front++;
    }
```

35. List the Applications of queue?

- Graph Algorithm
- Priority Algorithm

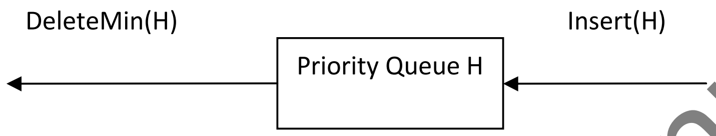
- Job scheduling
- Categorizing Data

36. Define priority queue with diagram and give the operations.

Priority queue is a data structure that allows at least the following two operations.

1. Insert-inserts an element at the end of the list called the rear.

2. DeleteMin-Finds, returns and removes the minimum element in the priority Queue.



Operations:

Insert

DeleteMin

37. Give the applications of priority queues.

There are three applications of priority queues

1. External sorting.

2. Greedy algorithm implementation.

3. Discrete even simulation.

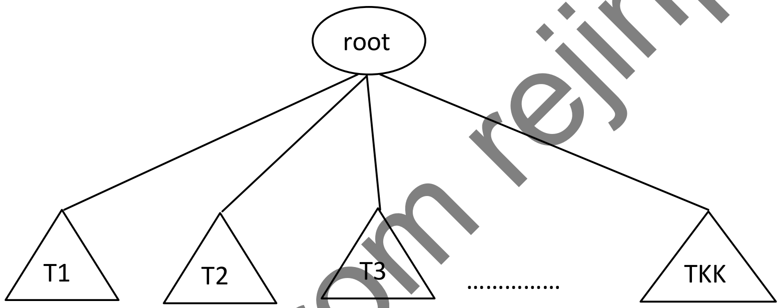
4. Operating systems.

UNIT II

TREE STRUCTURES

1. Define Tree .Give an example.

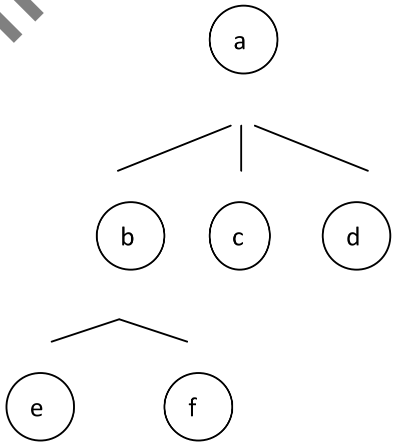
   A tree is a collection of nodes .The collection can be empty .Otherwise a tree consists of a distinguished node r called the root and 0 or more non empty sub-trees T_1,T_2,T_3……………T_k each of whose roots are connected by a directed edge from r.



Eg: directory structure hierarchy

2. Define depth of a node in a tree. Give example.

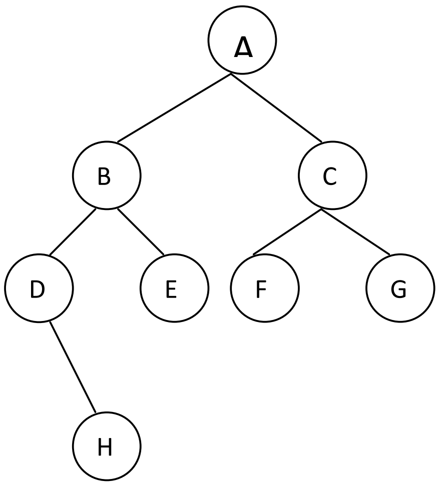
   For any node n_i the depth of n_i is the length of the unique path from the root to n_i
   eg:



The depth of e is 2.

3. Define length of the path in a tree with an example.

Length of a path is the number of edges on the path.

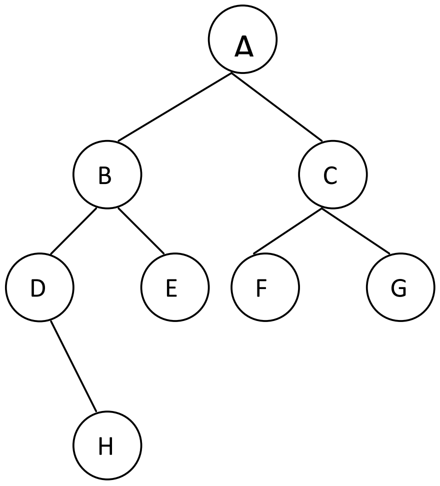


The length of the path from A-H is 3.

4. Define a path in a tree. Give example.

A path from a node n_1 to n_k is defined as the sequence of nodes n_1, n_2…..n_k such that n_i is the parent of n_{i+1} for $1 \le i < k$.

Eg:

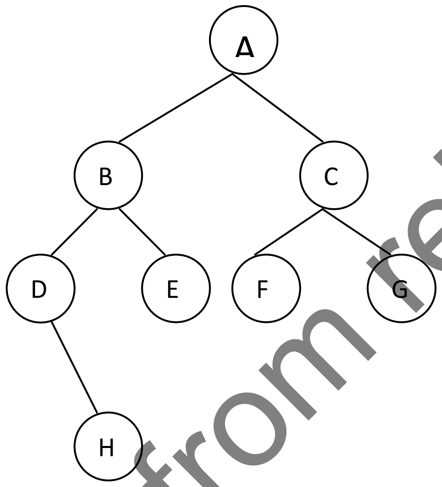


The path from A-H is A-B-D-H

5. Define height of the node in a tree. Give example.

The height of node n_i the length of the longest path from n_i to a leaf

Eg:



The height of node B is 2.

6. Write the routine for node declaration in trees.

```
typedef struct TreeNode *PtrToNode;
struct TreeNode
{
        ElementType Element;
        PtrToNode FirstChild;
        PtrToNode NextSibling;
};
```

7. List the applications of trees.

- Binary search trees
- Expression trees
- Threaded binary trees

8. Define Binary tree.

A Binary tree is a tree in which no node can have more than two children.

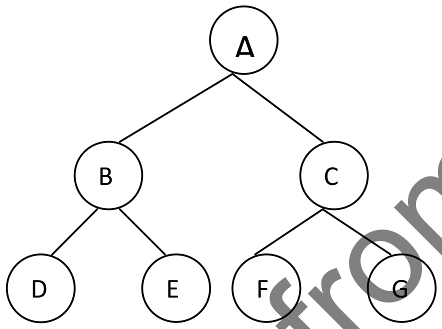
9. List the tree traversal applications.

1. Listing a directory in an hierarchal file system (preorder)

2. Calculating the size of a directory (post order)

10. Define binary tree ADT with an example.

A binary tree is a tree in which no node can have more than two children.



11. Define binary search tree?

Binary Search tree is a binary tree in which each internal node *x* stores an element such that the element stored in the left sub tree of *x* are less than or equal to *x* and elements stored in the right sub tree of *x* are greater than or equal to *x*. This is called binary-search-tree

12. List the Types of binary search trees

i) Performance comparisons

ii) Optimal binary search trees

13. List the Operations of binary search tree?
- Make Empty
- Find
- Insert
- Delete
- Search
- Display

14. Define Threaded Binary tree.

A Threaded Binary Tree is a binary tree in which every node that does not have a right child has a THREAD (in actual sense, a link) to its INORDER successor. By doing this threading we avoid the recursive method of traversing a Tree, which makes use of stacks and consumes a lot of memory and time.

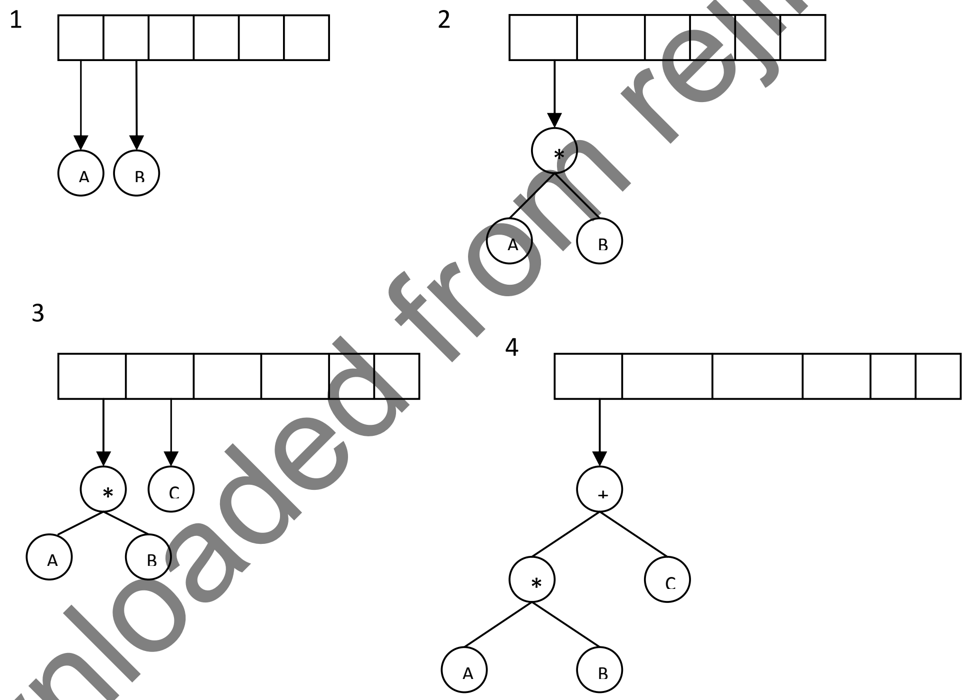
15. List the uses of binary tree.

1. Searching.

2. Compiler design.

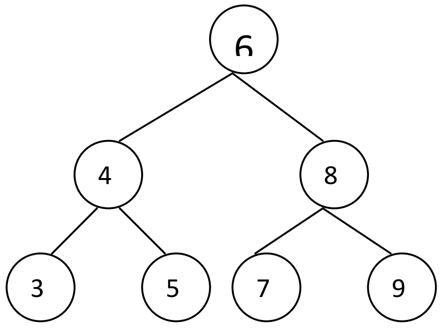
16. Draw the expression tree for the given postfix expression using stack.

AB*C+



17. Define binary search tree ADT with an example.

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



18. How deletion is performed in a binary search tree.

Once the node to be deleted is found there are three possibilities

1. If the node is a leaf, it can be deleted immediately.
2. If the node has one child the node can be deleted after its parent adjusts a pointer to bypass the node.
3. If the node has two children the general strategy is to replace the data of this node with the smallest data of the right sub tree and recursively delete the node which is empty.

19. Define internal path length.

It is the sum of the depths of all nodes in a tree.

20. What is the average depth of all nodes in an equally likely tree?

The average depth of all nodes in an equally likely tree is O (log N).

21. List out the disadvantages of Binary search tree.

1. Deletions in a binary search tree leads to trees which are not equally likely.
2. Absence of balanced search tree.
3. The average depth is not O (log N) in trees which are not equally likely.

22. Define tree traversal.

Traveling through all the nodes of the tree in such a way that each node is visited exactly once.

23. List out the types of Tree traversal?

There are three types of tree traversal

1. Preorder traversal
2. Inorder traversal
3. Postorder traversal

24. Write the steps and routine for the postorder traversal.

Steps:

1. Traverse the left sub tree.
2. Traverse the right sub tree.
3. Visit the root.

Routine:

```
void postorder(node *temp)
{
        if(temp!=NULL)
        {
                postorder(temp->left);
                postorder(temp->right);
                printf(“%d”,temp->data);
        }
}
```

25. Write the steps and routine for the preorder traversal.

Steps:

1. Visit the root
2. Traverse the left sub tree.
3. Traverse the right sub tree.

Routine:

```
void preorder(node *temp)
{
       if(temp!=NULL)
       {
               printf("%d",temp->data);
               preorder(temp->left);
               preorder(temp->right);

       }
}
```

26. Write the steps and routine for the inorder traversal.

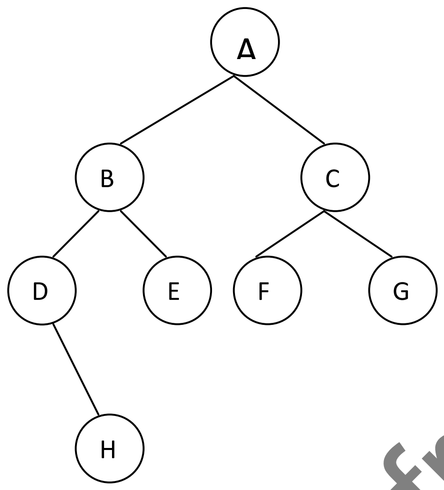
Steps:

1. Traverse the left sub tree
2. Visit the root
3. Traverse the right sub tree.

Routine:

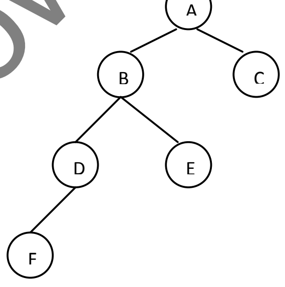
```
void inorder(node *temp)
{
        if (temp!=NULL)
        {
                inorder(temp->left);
                printf("%d",temp->data);
                inorder(temp->right);

        }
}
```

27. Perform preorder traversal for the given tree.



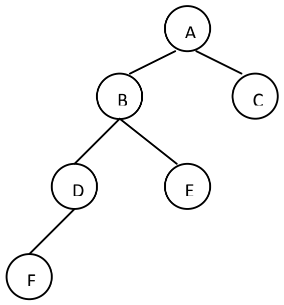
ABDHECFG

28. Perform inorder traversal for the given tree.



FDBEAC

29. Perform postorder traversal for the given tree.



FDEBCA

30. Define the following.

i) Leaf

Nodes at the bottommost level of the tree are called **leaf nodes**

ii) Sibling

The nodes with common parent are called Sibling

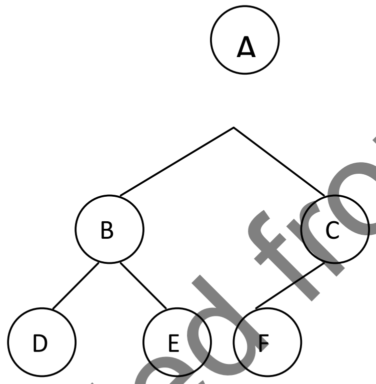
UNIT III

BALANCED TREE

1. Define AVL Tree. Give Example.

An AVL Tree is a binary search tree with a balance condition, which is easy to maintain and ensure that the depth of the tree is O(log N). Balance condition require that the left and the right sub trees have the same height.

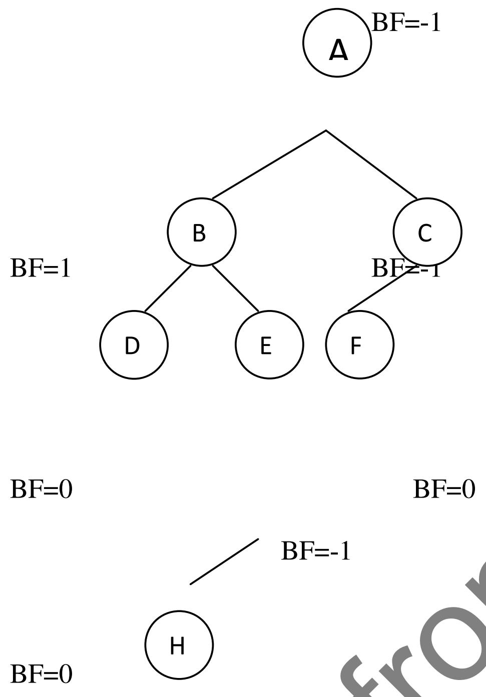
Example:



2. Define Balance factor.

The balance factor of a node in binary tree is defined to be h_R- h_L where h_L and h_R are heights of left and right subtrees of T. For any node in AVL tree the balance factor should be 1,0 or -1.

3. Give the balance factor of each node for the following tree.



4. When AVL tree property is violated and how to solve it?

After insertion of any node in an AVL tree if the balance factor of any node becomes other than -1,0, or 1 then it is said that AVL property is violated. So the node on the path from the inserted node to the root needs to be readjusted. Check the balance factor for each node in the path from inserted node to the root node and adjust the affected subtree such that the entire subtree should satisfy the AVL property.

5. Mention the four cases to rebalance the AVL tree.

- An insertion of new node into Left subtree of Left child(LL).
- An insertion of new node into Right subtree of Left child(LR).
- An insertion of new node into Left subtree of Right child(RL).
- An insertion of new node into Right subtree of Right child(RR).

6. Define Rotation in AVL tree. Mention the two types of rotations.

Some modifications done on AVL tree in order to rebalance it is called Rotation of AVL tree.

The two types of rotations are

- Single Rotation
- Left-Left Rotation
- Right-Right Rotation
- Double Rotation
- Left-Right Rotation
- Right-Left Rotation

7. Define Splay Tree.

A **splay tree** is a self-balancing binary search tree with the additional property that recently accessed elements are quick to access again. It performs basic operations such as insertion, look-up and removal in O (log(n)) amortized time. For many non-uniform sequences of operations, splay trees perform better than other search trees, even when the specific pattern of the sequence is unknown.

8. List the Operations of Splay tree.
   - Splaying
   - Insertion
   - Deleting

9. List the Operations on B-Trees.
   - Search
   - Create
   - Insert

10. List the B-Trees Applications.

- Databases
- Concurrent Access to B-Trees

11. Define B-Tree.

A search tree that is not a binary tree is called B-Tree. That satisfies the follpwing structural properties

- Root is either a leaf or has between 2 and M children
- All non leaf nodes except the root have between [M/2] and M children.
- All leafs are at the same depth.

12. Define binary heaps.

A **binary heap** is a heap data structure created using a binary tree. It can be seen as a binary tree with two additional constraints:

- The *shape property*: the tree is an *almost complete binary tree*; that is, all levels of the tree, except possibly the last one (deepest) are fully filled, and, if the last level of the tree is not complete, the nodes of that level are filled from left to right.
- The *heap property*: each node is greater than or equal to each of its children according to some comparison predicate which is fixed for the entire data structure.

13. List the Operations on Binary heap.
    - Adding to the heap
    - Deleting the root from the heap

14. List the Applications of Binary heap
    Heap sort

    Selection Algorithm

    Graph Algorithm

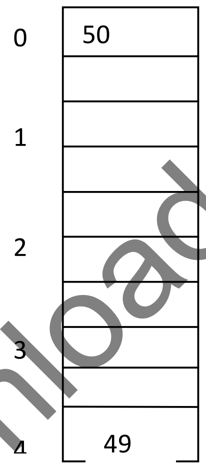
UNIT –IV

HASHING AND SET

1. Define hashing.
   It is the implementation of hash tables. Hashing is a technique used for performing insertions, deletions and finds in constant average time.

2. Define a Key in hashing.
   a. A key is a string with an associated value.
      e.g:Salary information.

3. Define Hash table. Give an example.
   The hash table data structure is an array of fixed size containing the keys.



4. Define table size of a hash table.
   Table size is the size of the table and it is part of the hash data structure. The table runs from 0 to Tablesize-1.

5. List the types of hash function.
   1. Division method
   2. Mid square

3. Multiplicative hash function
4. Digit folding
5. Digit analysis

6. Define a Hash function with an example.

Hash function maps each key into some number in the range 0 to Tableaize-1and places the key in the appropriate cell.

e.g:Key mod Tablesize is a basic hash function

7. Properties of a hash function.
   1. Hash function should be simple to compute.
   2. Should ensure that any two distinct keys get different cells.
   3. It should distribute the keys evenly among the cells.

8. What is collision in hashing?

   If an element is inserted and if it hashes to the same value as an already inserted element collision takes place.

9. What are the methods of resolving collision in hashing?
   1. Separate chaining hashing.
   2. Open addressing hashing.

10. Define separate chaining hashing

    Separate chaining means to keep a list of all elements that hash to the same value.

11. What are the operations of separate chaining?
    1. Find
    2. Insert

12. List out the applications of hashing.
    1. DBMS.
    2. Computer networks
    3. Storage of secret data
    4. Cryptography.
    5. Securing the database
    6. Database applications
    7. Storing data in a database

13. What are the advantages of separate chaining?
    1. Avoids collision by maintaining linked lists.
    2. The link lists will be short if the table is large and the hash function is good.

14. What are the disadvantages of separate chaining?
    1. Requires pointers.
    2. Slows down the algorithm
    3. Time is required to allocate new cells.

4. Requires the implementation of a second data structure.

15. What is open addressing?

In open addressing if a collision occurs alternative cells are tried until an empty cell is found. It is an alternative to resolving collisions with linked list.

(i.e.) $h_0(X)$,$h_1(X)$,………. Are tried in succession.

Where $h_i(X)=(Hash(X)+F(i)) \bmod Tablesize$ with $F(0)=0$ where F is the collision resolution strategy.

16. What are the types of open addressing hashing?

- Linear probing.
- Quadratic probing
- Double hashing

17. Differentiate between linear probing and quadratic probing

Linear probing	Quadratic probing
• Definition: • It amounts to trying cells sequentially with wraparound in search of an empty cell.	• Definition: • It is a collision resolution method that eliminates the primary clustering problem of linear probing.
• Function: • It is linear (i.e.)F(i)=i	• Function: • It is quadratic (i.e.) $F(i)=i^2$
• Advantages: • 1.Does not use pointers • No second data structure • 3. Time is not required for allocating new cells.	• Advantages: • 1.Does not use pointers • No second data structure • Time is not required for allocating new cells. • If the table is half empty and the table size is prime then inserting a new element is successful and guaranteed. • No primary clustering.
• Disadvantages: • Time for inserting is quite large. • Primary clustering. • Linear probing is a bad idea if the table is more than half full. • 4. Bigger table is needed than separate chaining.	• Disadvantages: • There is no guarantee of finding an empty cell if the table gets more than half full or if the table size is not prime. • Bigger table is needed than separate chaining. • The number of alternative locations is severely reduced if the table size is not prime.

	• 4. Standard deletion cannot be performed.

18. Define primary clustering.

    It is the forming of blocks of occupied cells. It means that any key that hashes into the cluster will require several attempts to resolve the collision and then it will add to the cluster.

19. Define double hashing.

    It is a collision resolution method .For double hashing F(i)=i.$hash_2$(X).

    (i.e) a second hash function is applied to X and probing is done at a distance $hash_2$(X), 2$hash_2$(x)………. and so on.

20. Define rehashing.

    Rehashing is the building of another table with an associated new hash function that is about twice as big as the original table and scanning down the entire original hash table computing the new hash value for each non deleted element and inserting it in the new table.

21. List out the advantages and disadvantages of rehashing.

    Advantage:

    - Table size is not a problem.
    - Hash tables cannot be made arbitrarily large.
    - Rehashing can be used in other data structures.

    Disadvantage:

    - It is a very expensive operation.
    - The running time is 0(N).
    - Slowing down of rehashing method.

22. What are the ways in which rehashing can be implemented.
    1. Rehash as soon as the table is half full.
    2. Rehash only when an insertion fails.
    3. Middle of the road strategy is to rehash when the table reaches a certain load factor.

23. Define Extendible hashing.

    Extendible hashing allows a find to be performed in two disk access and insertions also require few disk accesses.

24. What is heap order property?

The smallest element should be at the root .Any node should be smaller than all of its descendants.

25. What is meant by Expression Tree?

An expression tree is a binary tree in which the operands are attached as leaf nodes and operators become the internal nodes.

26. List the applications of set.

i. Disjoint-set data structures model the partitioning of a set, for example to keep track of the connected components of an undirected graph. This model can then be used to determine whether two vertices belong to the same component, or whether adding an edge between them would result in a cycle.

ii. This data structure is used by the Boost Graph Library to implement its Incremental Connected Components functionality. It is also used for implementing Kruskal's algorithm to find the minimum spanning tree of a graph.

27. Define Disjoint set.

A **disjoint-set data structure** is a data structure that keeps track of such a partitioning. A **union-find algorithm** is an algorithm that performs two useful operations on such a data structure:

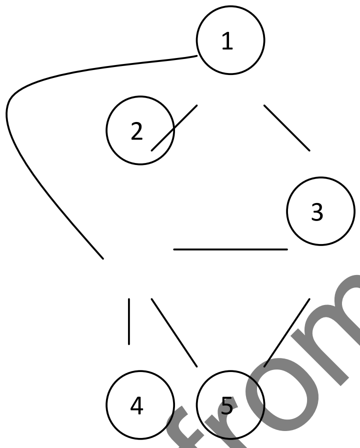
- *Find*: Determine which set a particular element is in. Also useful for determining if two elements are in the same set.
- *Union*: Combine or merge two sets into a single set.

UNIT V

GRAPHS

1. Define Graph.

A Graph is defined as G = (V, E) where V is the set of vertices (nodes) and E is the set of edges(arcs) connecting the vertices. An edge is represented as a pair of vertices (u,v).



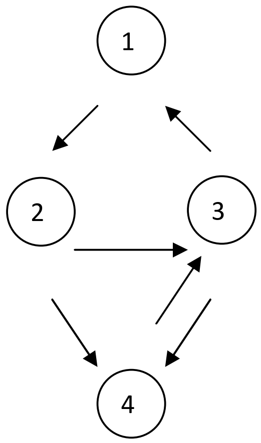
V = {1, 2, 3, 4, 5}

E = {(1, 2),(1, 3),(1,4),(2,3)

(2,4),(2,5),(3,5)}

2. Define Directed Graph.

A Directed graph is a graph in which the edges are directed. It is also called Digraph.



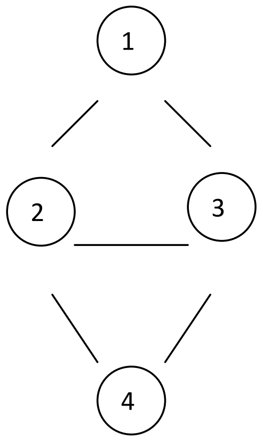
V={1,2,3,4}

E={(1,2),(2,3),(2,4)

(3,1),(3,4),(4,3)}

3. What do you mean by Undirected Graph?

Edges in the graph do not have directions marked. Such graphs are referred to as undirected graphs.



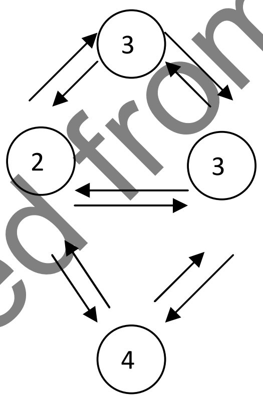
V={1,2,3,4}

E={(1,2),(2,1),(2,3),(3,2)

2,4),(4,2),(3,4),(4,3),(3,1),(1,3)}

Define Symmetric Digraph.

Every edge has an edge in the reverse direction i.e for every edge (U,V) there is an edge (V,U). Such type of graph is called Symmetric Digraph.



V={1,2,3,4}

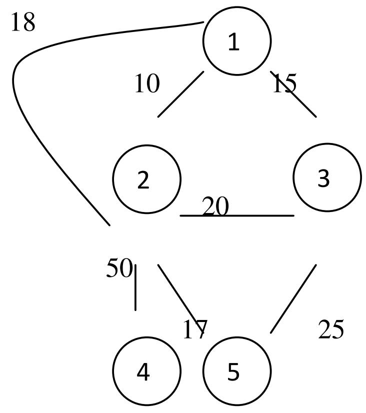
E={(1,2),(2,1),(2,3),(3,2)

(2,4),(4,2),(3,4),(4,3),(3,1),(1,3)}

4. What do you mean by weighted graph?

Weighted graphs are such graphs where the edges are associated with weights. These weights are used to mark the importance of edges in representing a problem.

Ex. Road map represented as graph where the weight is the distance between two places.



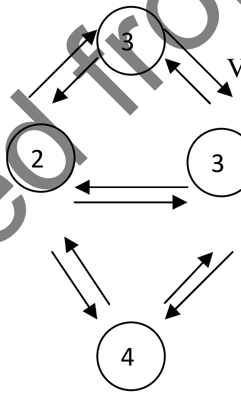
V={1,2,3,4,5}

E={(1,2),(1,3),(1,4),(2,3)

(2,4),(2,5),(3,5)}

5. Define adjacent vertices.

Two vertices are said to be adjacent vertices if there is an edge between them.

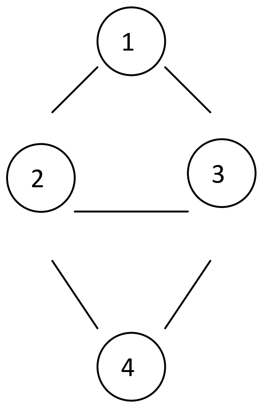


Vertices 1&2, 2&3, 2&4, 3&4 are

adjacent vertices.

6. Define path.

A Path between vertices (u,v) is a sequence of edges which connects vertices u & v.

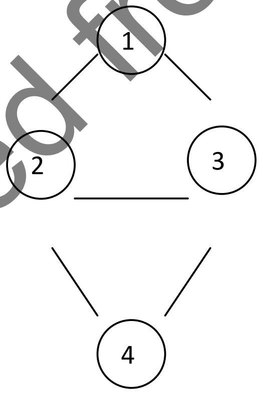


The path between the vertices 1 and 4 is 1-2-4 or 1-3-4

7. Define Length of the Path.

Length of the path is the number of edges in a path which is equal to N-1.

N represents the number of vertices.

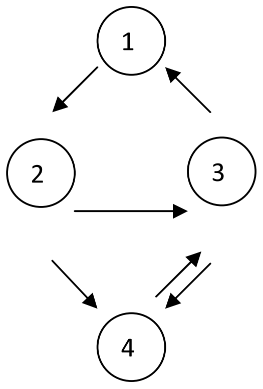


The Length of the path 1 -4 {(1,2),(2,3),(3,4) } is 3

{(1,3),(3,4)} is 2 and also {(1,2),(2,4)} is 2.

9. Define cycle.

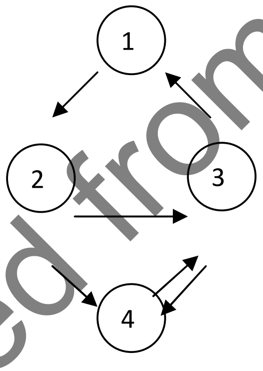
A Cycle in a graph is a path that starts and ends at the same vertex. i.e path in which the first and last vertices are same.



Ex: {(3, 1), (1, 2),(2,4),(4,3)} is a Cycle.

10. Define simple cycle.

Simple cycle means the vertex should not be repeated and the first and last vertex should be the same.

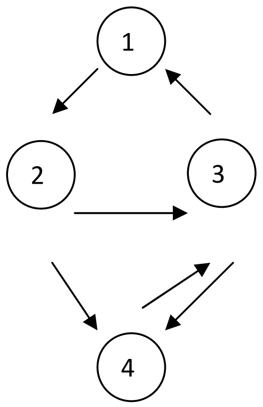


Example: 1-2-4-3-1 is a simple cycle

1-2-3-4-3-1 is not a simple Cycle

11. Define Simple Path.

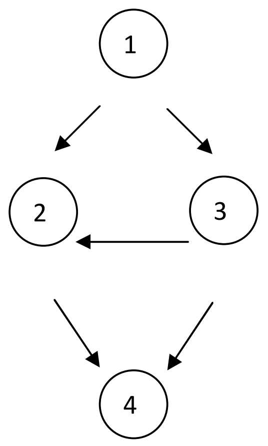
All vertices are distinct, except that the first and last may or may not be the same.

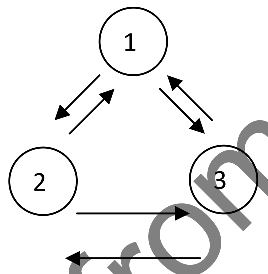


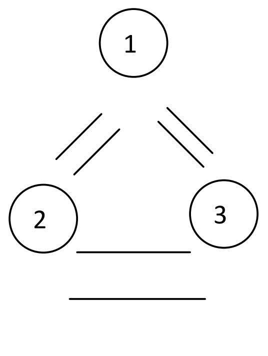
Example: 1-2-4-3-1 is a simple path

1-2-3-4-3-1 is not a simple path

1-2-4-3 is a simple path

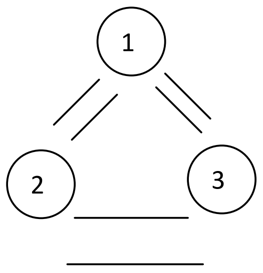

1
2
3
4


1
2
3


1
2
3

15. Define complete graph.

A graph in which there is an edge between every pair of vertices.



16. Mention the ways of representing a graph?

a) Adjacency Matrix representation
b) Adjacency List representation

17.What do you mean by Adjacency Matrix representation?

The adjacency matrix M of a graph G = (V, E) is a matrix of order V_i xV_j and the elements of the unweighted graph M are defined as

M [i] [j] = 1, if (V_i, V_j) ∈ E

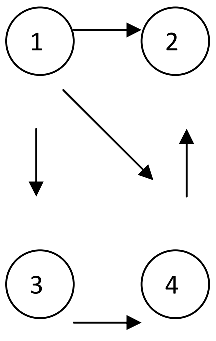
= 0 Otherwise

For a weighted graph the elements of M are defined as

M [i] [j] = W_{ij}, if (V_i, V_j) ∈ E and W_{ij} is the weight of edge (V_i, V_j)

= 0 Otherwise

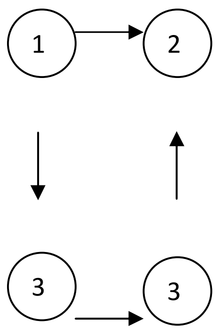
Example:

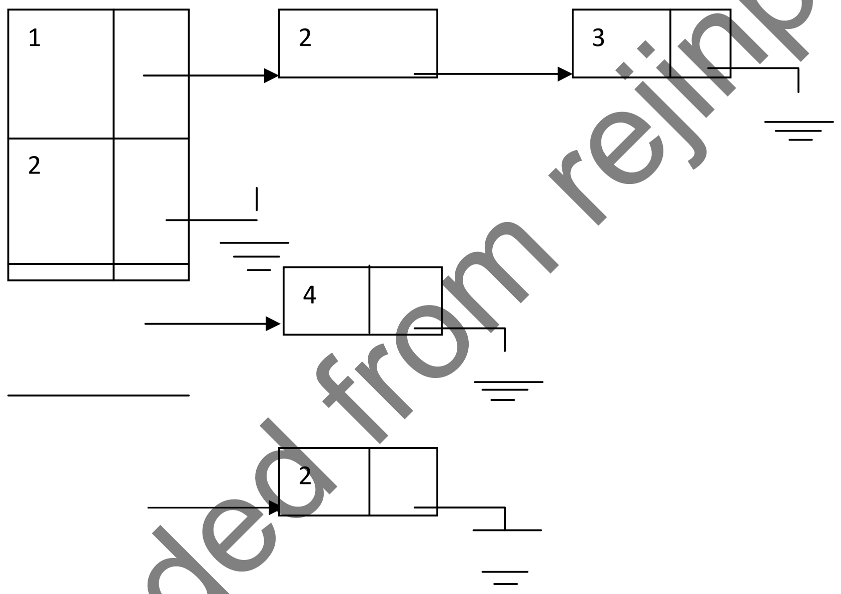


$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 1 & 1 \\ 2 & 0 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 & 1 \\ 4 & 0 & 1 & 0 & 0 \end{array}$$

18. What do you mean by adjacency List representation?

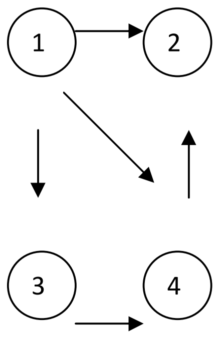
It is an array of linked list ,for each vertex a linked list of all adjacent vertices is maintained.





19. What do you mean by indegree and outdegree of a graph?

- Indegree of a vertex in a graph is the number of incoming edges.
- Outdegree of a vertex is the number of edges that leaves the vertex.
  Example:



Indegree for the vertex 1 is 0, vertex 2 is 2, vertex 3 is 1, and vertex 4 is 2

Outdegree for the vertex 1 is 3, vertex 2 is 0, vertex 3 is 1, and vertex 4 is 1.

20. Define Topological sort.

Topological sort is defined as an ordering of vertices in a directed acyclic graph. such that if there is a path from V_i to V_j, then V_j appears after V_i in the ordering.

21. Explain the principle of topological sort.

- Find the vertex with no incoming edge.
- Print the vertex and remove it along with its edges from the graph.
- Apply the same strategy to the rest of the graph.
- Finally all recorded vertices give topological sorted list.

22. What is the disadvantage of topological sort?

Ordering of vertices is not possible if the graph is a cyclic graph. Because if there are two vertices v and w on the cycle, v proceeds w and w proceeds v, so ordering not unique.

23. What is the running time for topological sort?

The running time of the algorithm for topological sort is $O(|V|^2)$ For the algorithm using Queue, the running time is $O(|E| + |V|)$ if adjacency list are used.

24. State the shortest path problem OR Single source shortest path problem.

Given as input a weighted graph or an unweighted graph G= (V, E) and a distinguished vertex s , the shortest path problem is to find the shortest weighted or unweighted path from s to every other vertex.

25. Explain weighted path length.

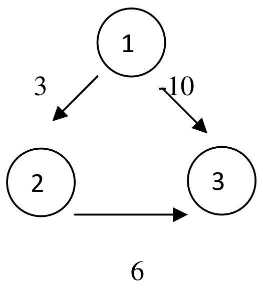
The cost of the path $V_1, V_2 \ldots V_N$ is $\sum_{i=1}^{N-1} C_{i,\, i+1}$. This is referred to as the weighted path length.

26. Explain unweighted path length

Unweighted path length is the number of edges on the path namely, N-1(where N is the number of vertices).

27. What do you mean by Negative edge?

Negative edge means a graph having atleast one edge with a negative weight.



28. What do you mean by negative cost cycle?

A graph having the shortest path with negative weight is known Negative cost cycle.

29. Give examples for problems solved by shortest path algorithm.

- Cheapest way of sending electronic news from one computer to another.
- To compute the best route

30. What is the running time for the weighted and unweighted shortest path?

- The running time for the weighted shortest path is $O(|E| + |V|)$
- The running time for the Unweighted shortest path is $O(|E| \log |V|)$

31. What is the advantage of unweighted shortest path algorithm?

There is no calculation with weights. Only the number of edges on the shortest path is found.

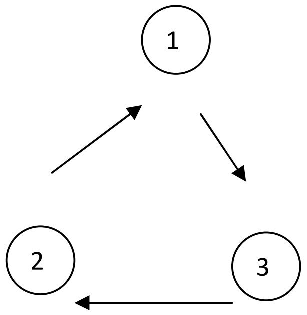
35. What are the applications of graphs?

1. Airport system

2. Street traffic

36. What is a cyclic graph?

A graph which has atleast one cycle is called a cyclic graph.



37. What is a connected graph?

An undirected graph is connected is connected if there is a path frog every vertex to every other vertex.