

# CS3301 DATA STRUCTURES

## UNIT- I

### PART A

#### 1. Define: data structure.

Data structure is a way of organizing data in a computer so that it can be used efficiently. A data structure is a collection of data values and the relationships among them.

#### 2. Give few examples for data structures?

Arrays, stacks, queue, list, tree, graph, set, map and table.

#### 3. What are the different types of data structures?

- i) Primitive
- ii) Composite
  - (a) Linear
  - (b) Nonlinear

#### 4. What are primitive data types?

The basic building blocks for all data structures are called primitive data types. They are single element. (e.g) int, float, char, double, Boolean

#### 5. What are composite data types?

The Data structures that are not atomic are called non-primitive or composite. They are collection of elements. (e.g) array, structure, union

#### 6. What is meant by an abstract data type?

An abstract data type (ADT) is a set of elements together with a set of operations. Implementation details of the operations are not mentioned. (e.g) list, stack, queue

#### 7. How can we categorize data structures based on data access?

Linear list, stack, queue Non-linear- heap, tree, graph

#### 8. State the difference between linear and non-linear data structures.

Linear	Nonlinear
Data elements are arranged in a sequential order	Data elements are not arranged in sequential order
Data elements are present at single level	Data elements are present at multiple levels
It is simple, easy to understand and implement	It is difficult to understand and implement
There is only one way of traversing a linear data structure	There are different ways of traversing a nonlinear data structure
Examples: arrays, linked lists, stacks, and queues	Examples: trees and graphs

#### 9. List a few real-time applications of data structures?

- Function call - stack
- Printing jobs - queue
- Compilers - hash table
- Directory structure - tree
- Communication networks - graph

### 10. Define List.

The general form of the list is  $A_0, A_1, A_2, \dots, A_{N-1}$ . Size of this list is  $N$ . For any list,  $A_i$  follows (or succeeds)  $A_{i-1}$  and that  $A_{i-1}$  precedes  $A_i$ . The position of element  $A_i$  in a list is  $i$ . List is associated with set of operations like search, insert, delete etc.

### 11. What are the different ways to implement list?

- Array implementation of list
- Linked list implementation of list
- Cursor implementation of list

### 12. What are the advantages in the array implementation of list?

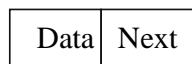
- Elements are stored in consecutive memory locations
- It allows random access of data

### 13. What are the disadvantages in the array implementation of list?

- Requires that the list size to be known in advance
- Insertion and deletion requires more data movement

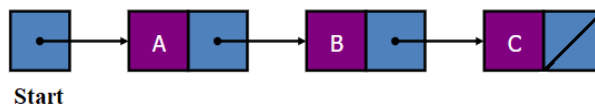
### 14. Define node.

A node consists of two fields namely an information field called DATA and a pointer field called NEXT. The DATA field is used to store the data and the NEXT field is used to store the address of the next node.



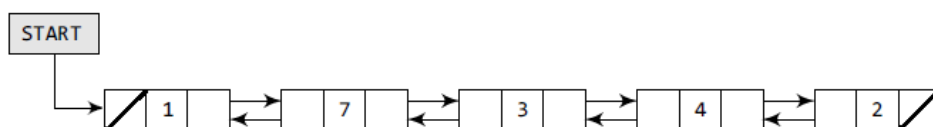
### 15. What is a linked list?

A linked list is a linear collection of data elements. These data elements are called nodes. A linked list does not store its elements in consecutive memory locations



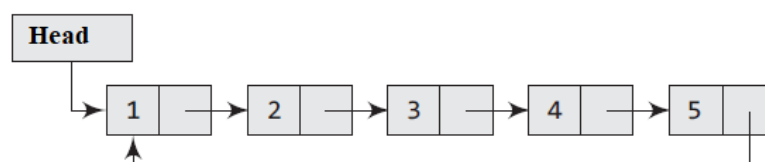
### 16. What is a doubly linked list?

A doubly linked list is a two-way linked list which contains a pointer to the next as well as a pointer to the previous node. Every node in a doubly linked list consists of three parts: data, a pointer to the next node, and a pointer to the previous node.



### 17. Define circularly linked list?

In a circular linked list, the last node contains a pointer to the first node of the list. We can have a circular singly linked list as well as a circular doubly linked list.



**18. Mention the advantages and disadvantage of using circular list.**

The advantage of using circular list is

- Traversing a circular linked list can begin at any node and traverse the list in any direction, forward or backward

The disadvantage of using circular list is

- Complexity of iteration. i It is possible to get into an infinite loop.

**19. What are the disadvantage of linked list over array? (Nov/Dec 2018)**

- It requires more memory space to store data
- Nodes are not stored in consecutive memory locations
- It does not allow random access of data

**20. State the advantages of ADT?(Nov/Dec 2018)**

- ADT is reusable, robust, and is based on principles of Object Oriented Programming (OOP) and Software Engineering (SE)
- An ADT can be re-used at several places and it reduces coding efforts
- Encapsulation ensures that data cannot be corrupted
- Working of various integrated operation cannot be tampered with by the application program
- ADT ensures a robust data structure

**21. List the applications of linked list**

- Polynomial Manipulation
- Arithmetic operations on long integers
- Representation of sparse matrices
- Symbol table creation.
- Dynamic memory management
- Linked allocation of files

**22. What is multilist?**

In a multi-linked list, each node can have n number of pointers to other nodes. Multi-linked lists are generally used to organize multiple ordered set of elements.

**PART B**

**1. What are the various operations on array? Write a procedure to insert and delete an element in the middle of an array.**

**Array Implementation of list:**

Array is a collection of specific number of same type of data stored in consecutive memory locations.

**The basic operations performed on an array are**

- a) Creation of List.
- b) Insertion of data in the List
- c) Deletion of data from the List
- d) Display all data's in the List
- e) Search data into the list.

**Routine to insert an element in the array:**

**Routine to delete an element from the array**

## 2. Outline the steps to perform insertion and deletion in a linked list with an example and relevant diagrams.

### Linked Lists:

A Linked list is an ordered collection of elements. Each element in the list is referred as a node. Each node contains two fields namely data and next

### Routine to insert an element in the linked list

### Routine to delete an element from the linked list

## 3. How can a polynomial be represented as a linked list? Outline the algorithm for addition of two polynomial using linked list with an example.

### Representation of polynomial using linked list

```
struct poly
{
    int coeff; int power;
    struct poly *next;
}*list1,*list2,*list3;
```

### Addition of two polynomials

Input – polynomial p1 and p2 represented as a linked list.

Step 1: loop around all values of linked list and follow step 2& 3.

Step 2: if the value of a node's exponent. is greater copy this node to result node and head towards the next node.

Step 3: if the values of both node's exponent is same add the coefficients and then copy the added value with node to the result.

Step 4: Print the resultant node.

## 4. Give an algorithm to perform insertion and deletion on doubly Linked list

### Routine to insert an element in a doubly linked list

Void insert(int X, List L)

```
{
    position Newnode;
    Newnode=(struct node*)malloc(sizeof(struct node));
    if(Newnode!=NULL)
    {
        Newnode->data=X;
        Newnode->next=L->next;
        L->next->prev=Newnode;
        L->next=Newnode;
        Newnode->prev=L;
    }
}
```

### Routine to delete an element from the doubly linked list

void delete(List L)

```
{
    position temp;
    if(L->next!=NULL)
    {
        temp=L->next;
        L->next=temp->next;
```

```

        temp->next->prev=L;
        free(temp);
    }
}

```

## 5. Write algorithm for circular Linked list insertion and deletion operation

### **Circular Linked list:**

A Singly linked circular list is a linked list in which the last node of the list points to the first node.

### **Routine to insert an element in a circular linked list**

```

void insert(int X, List L)
{
    position Newnode;
    Newnode=(struct node*)malloc(sizeof(struct node)); if(Newnode!=NULL)
    {
        Newnode->data=X;
        Newnode->next=L->next;
        L->next=Newnode;
    }
}

```

### **Routine to delete an element from a circular linked list**

```

void delete(List L)
{
    position temp;
    temp=L->next;
    L->next=temp->next;
    free(temp);
}

```

## UNIT-II

### PART A

#### 1. Define Stack

A stack is a set of elements with the restriction that insertion and deletion can be performed in only one position called as top. Stack is also called as "LIFO" or "Last In First Out" list.

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

The fundamental operations on a stack are push and pop. Push operation insert an element in the top of stack. Pop operation delete an element from the top of stack.

#### 3. How do you test an empty stack?

The condition for testing an empty stack is  $\text{top} = -1$ , where top is the pointer pointing to the topmost element of the stack.

#### 4. List the applications of stack?

- Balancing Symbols
- Infix to Postfix conversion
- Evaluation of Postfix expression
- Function call

#### 5. Define a postfix expression.

Postfix expression is an expression of the form  $a \ b \ op$ . An operator is followed for every pair of operands. Postfix notation is also called as Reverse Polish notation (e.g)  $a \ b \ + \ c \ *$

#### 6. Write the postfix and prefix forms of the expression $A+B*(C-D)/(P-R)$

Postfix form:  $ABCD-*PR-/+$

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

#### 7. Explain the usage of stack in recursive algorithm implementation?

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

#### 8. Define Queue

A queue is a set of elements with insertion is done at one end called rear end and deletion is performed at other end called front end. Queue is also called as "FIFO" or "First In First Out" list.

#### 9. What are the various operations performed on the Queue?

The fundamental operations on a queue are enqueue and dequeue. Enqueue operation inserts an element at one end of the queue (Called as rear end), Dequeue operation deletes an element at other end of the queue (Called as front end).

#### 10. What are the types of Queue?

- Simple Queue
- Circular Queue
- Priority Queue
- Dequeue (Double Ended Queue)

**11. Define Deque.**

Deque stands for Double ended queue. It is a variation of queue in which insertion and deletion operation can be performed at both ends that is front and rear of the queue.

**12. Define Circular Queue. Why we need a circular queue?**

It is a variation of queue in which last position is connected back to the first position to make a circle. In a normal Queue, we can insert elements until queue becomes full. But once queue becomes full, we cannot insert the next element even if there is a space in front of queue. This drawback is overcome in circular queue

**13. What are priority queue? What are the ways to implement priority queue?**

Priority queue is a variation of queue in which the elements will have some predefined priority. The data item with highest priority is removed first. Priority queue can be implemented using a data structure called a heap.

**14. List the applications of queue.**

- CPU scheduling
- process synchronization
- printing
- Handling of interrupts

**15. Distinguish between stack and Queue**

<b>Stack</b>	<b>Queue</b>
A stack is a set of elements insertion and deletion can be performed in only one position called as top.	A queue is a set of elements with insertion is done at one end and deletion is performed at other end
Stack is a "LIFO" or "Last In First Out" list.	Queue is a "FIFO" or "First In First Out" list.
The fundamental operations on a stack are push and pop	The fundamental operations on a queue are enqueue and dequeue.
Only one pointer is used, named as Top	Two pointers are used, named as Rear and Front

**PART B****1. Explain the various operations on a stack**

**Stack is a Linear Data Structure that follows Last In First Out (LIFO) principle.**

**Operations on Stack (Stack ADT)**

Two fundamental operations performed on the stack are PUSH and POP.

**PUSH operation**

It is the process of inserting a new element at the Top of the stack.

For every push operation:

1. Check for Full stack ( overflow ).

2. Increment Top by 1. ( $\text{Top} = \text{Top} + 1$ )
3. Insert the element X in the Top of the stack.

### **POP operation**

It is the process of deleting the Top element of the stack.

For every pop operation:

1. Check for Empty stack ( underflow ).
2. Delete (pop) the Top element X from the stack
3. Decrement the Top by 1. ( $\text{Top} = \text{Top} - 1$  )

## **2. Briefly describe the operations of queue with an examples.**

**Queue is a Linear Data Structure that follows First in First out (FIFO) principle**

### **EnQueue operation**

It is the process of inserting a new element at the rear end of the Queue.

For every EnQueue operation

1. Check for Full Queue
2. If the Queue is full, Insertion is not possible.
3. Otherwise, increment the rear end by 1 and then insert the element in the rear end of the Queue.

### **DeQueue Operation**

It is the process of deleting the element from the front end of the queue.

For every DeQueue operation

1. Check for Empty queue
2. If the Queue is Empty, Deletion is not possible.
3. Otherwise, delete the first element inserted into the queue and then increment the front by 1.

## **3. Outline the algorithm for evaluating a postfix expression using stack data structure with an example.**

### **Algorithm to evaluate the obtained Postfix Expression**

Read the postfix expression one character at a time until it encounters the delimiter “#”

Step 1: If the character is an operand, push its associated value onto the stack.

Step 2: If the character is an operator, POP two values from the stack, apply the operator to them and push the result onto the stack.

### **Explanation with example**

## **4. Write the procedure to convert the infix to post fix expression and explain with example**

### **Algorithm to convert Infix Expression to Postfix Expression:**

Read the infix expression one character at a time until it encounters the delimiter “#”

Step 1: If the character is an operand, place it on the output.

Step 2: If the character is an operator, push it onto the stack. If the stack operator has a higher or equal priority than input operator then pop that operator from the stack and place it onto the output.

Step 3: If the character is left parenthesis, push it onto the stack

Step 4: If the character is a right parenthesis, pop all the operators from the stack till it encounters left parenthesis, discard both the parenthesis in the output.

### **Explanation with example**

## **5. What are circular Queue? Write the procedure to insert an element to circular Queue**

**In Circular Queue, the insertion of a new element is performed at the very first location of the queue if the last location of the queue is full, in which the first element comes just after the last element**

**Circular Queue Enqueue Operation**

It is same as Linear Queue EnQueue Operation (i.e) Inserting the element at the Rear end.

First check for full Queue.

If the circular queue is full, then insertion is not possible. Otherwise check for the rear end.

If the Rear end is full, the elements start getting inserted from the Front end.

**Circular Queue DeQueue Operation**

It is same as Linear Queue DeQueue operation (i.e) deleting the front element.

First check for Empty Queue.

If the Circular Queue is empty, then deletion is not possible.

If the Circular Queue has only one element, then the element is deleted and Front and Rear pointer is initialized to - 1 to represent Empty Queue.

Otherwise, Front element is deleted and the Front pointer is made to point to next element in the Circular Queue.

## UNIT- III

### PART A

#### 1. Define tree?

Tree is a non-linear data structure, which is a collection of nodes. A tree consists of a distinguished node called the root, and zero or more nonempty subtrees  $T_1, T_2, \dots, T_k$ . Nodes are connected as parent child relation

#### 2. Define Height of tree?

The height of a tree is the length of the longest path from root to a leaf. The length of this path is the number of edges on the path. The height of a tree is equal to a height of a root.

#### 3. Define sibling?

Nodes with the same parent are called siblings.

#### 4. Define binary tree?

A binary tree is a tree in which no node can have more than two children. A binary tree consists of a root and two subtrees, TL and TR

#### 5. What are the two methods of binary tree implementation?

Two methods to implement a binary tree are,

- a. Array representation.
- b. Linked list representation

#### 6. List out few Application of tree data-structure?

- Binary Search Tree allows fast search, insert, delete on a sorted data.
- Heap tree is used to implement priority queues.
- B-Tree and B+ Tree are used to implement indexing in databases.
- Syntax tree is used in Compilers.
- Spanning Trees and shortest path trees are used in computer networks

#### 7. Define expression tree?

Expression tree is a binary tree in which the leaf nodes are operands, such as constants or variable names, and the other nodes contain operators.

#### 8. Define tree traversal and mention the type of traversals?

Tree traversal is a process to visit all the nodes of a tree and print their values. There are three types of tree traversal

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

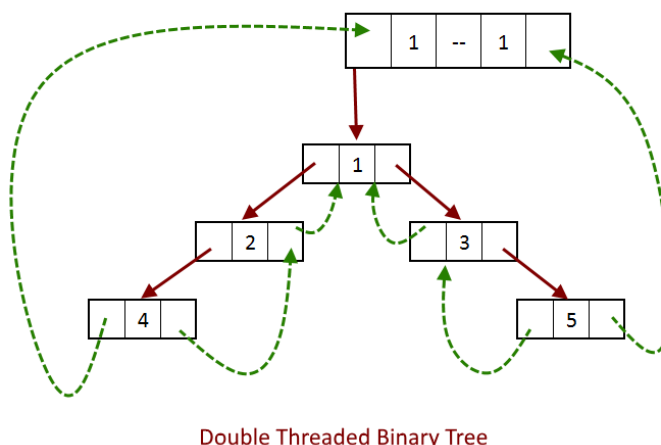
#### 9. Define in-order traversal?

In-order traversal involves the following steps:

- a. Traverse the left subtree
- b. Visit the root node
- c. Traverse the right subtree

### 10. Define threaded binary tree.

A binary tree is threaded by making all left child null pointers to point the inorder predecessor and all right child null pointers to point the inorder successor of the node. It is used to resolve null pointers. One example is given below



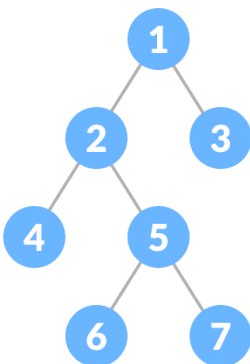
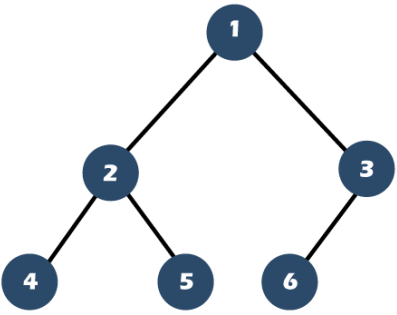
### 11. Define Binary Search Tree.

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

### 12. What is AVL Tree?

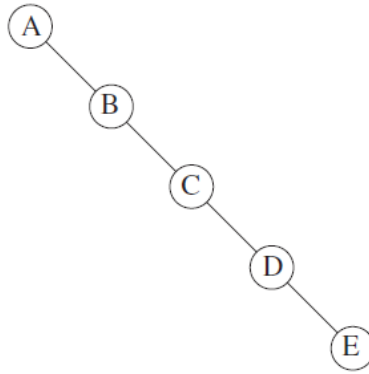
An AVL (Adelson Velskii and Landis) tree is a binary search tree with a balance condition. The balance condition is for every node in the tree, the height of the left and right subtrees can differ by at most 1.

### 13. Differentiate Full binary tree and Complete binary tree

Full binary tree	Complete binary tree
A full binary tree can be defined as a binary tree in which all the nodes have 0 or two children. In full binary tree all the nodes have two children except the leaf nodes.	A binary tree is said to be a complete binary tree when all the levels are completely filled except the last level, which is filled from the left.
Example: 	Example: 

### 14. What is skewed binary tree?

A skewed binary tree (worst case binary tree) is a type of binary tree in which all the nodes have only either one child or no child. Example



**15. The depth (height) of a binary tree is 8. Compute the number of nodes in leaf.**

Number of nodes in a binary tree of height  $h = 2^h - 1$

Number of nodes in a binary tree of height 8 =  $2^8 - 1 = 255$

Number of leaf nodes in a binary tree of height  $h = 2^{h-1}$

Number of nodes in a binary tree of height 8 =  $2^{8-1} = 2^7 = 128$

**16. How many NULL pointers does a binary tree with N nodes have?**

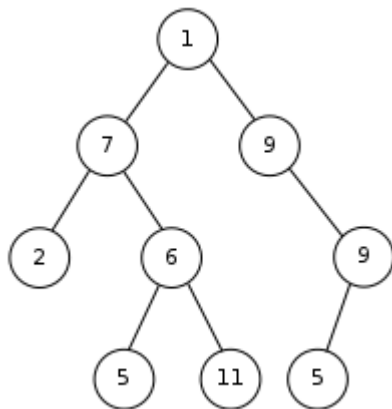
Binary tree with n nodes have  $N + 1$  null pointers.

For a binary tree with n nodes, there are  $2N$  pointers and  $N - 1$  incoming edges.

Therefore, we have  $2N - (N - 1) = N + 1$  NULL pointers.

**17. For the tree given below**

- (a) List the siblings for node 6
- (b) Compute the height



Sibling of node 6 is 2

Height of the tree is 3

**18. Define Heap**

Heap data structure is a complete binary tree that satisfies the heap property, where any given node is always greater than its child node/s and the key of the root node is the largest among all other nodes

**PART B**

**1. Outline preorder, inorder and post order traversal on a binary tree with an algorithm and an example.**

12

**Inorder**

1. Traverse left subtree

2. Process root node
3. Traverse right subtree

### **Preorder**

1. Process root node
2. Traverse left subtree
3. Traverse right subtree

### **Postorder**

1. Traverse left subtree
2. Traverse right subtree
3. process root node

## **2. State the binary search tree property and outline the algorithm to search an element in a binary search tree with an example.**

### **BST property**

- The left sub-tree of a node contains only nodes with keys less than the node's key.
- The right sub-tree of a node contains only nodes with keys greater than the node's key.
- Both the left and right sub-trees must also be binary search trees.

### **Search Operation**

```
Position find(elementtype X, searchtree T)
{
    If(T==NULL)
        return NULL;
    if(x< T-->element)
        return find(x,T-->left);
    else if(X> T-->element)
        return find(X,T-->right);
    else
        return T;
}
```

## **3. What are expression trees? Write the procedure for constructing an expression tree.**

### **Definition**

Expression tree is a binary tree in which the leaf nodes are operands, such as constants or variable names, and the other nodes contain operators.

### **Procedure for constructing an expression tree**

1. Read one symbol at a time from the postfix expression.
2. Check whether the symbol is an operand or operator.
  - a. If the symbol is an operand, create a one node tree and push a pointer on to the stack.
  - b. If the symbol is an operator, pop two pointers from the stack namely, T1 and T2 and form a new tree with root as the operator, and T2 as the left child and T1 as the right child.
  - c. A pointer to this new tree is then pushed on to the stack.

## **4. What are AVL trees? Describe different Rotation defined for AVL Tree ?**

13

### **Definition**

AVL tree is a self-balancing binary search tree. In an AVL tree, the heights of the two child subtrees of any node differ by at most one; therefore, it is also said to be height-balanced.

## **Rotation**

1. Single Rotation – ( If insertion occurs on the outside)
  - LL (Left - Left rotation) - Do single Right.
  - RR (Right - Right rotation) - Do single Left.
2. Double Rotation - ( If insertion occurs on the inside)
  - RL ( Right -- Left rotation) --- Do single Right, then single Left.
  - LR ( Left -- Right rotation) --- Do single Left, then single Right.

## **5. Illustrate how delete operation is performed in the binary heap**

### **Priority queue is implemented by Binary heap**

- Mintree - Parent should have lesser value than children.
- Maxtree - Parent should have greater value than children.

### **Deletion Operation**

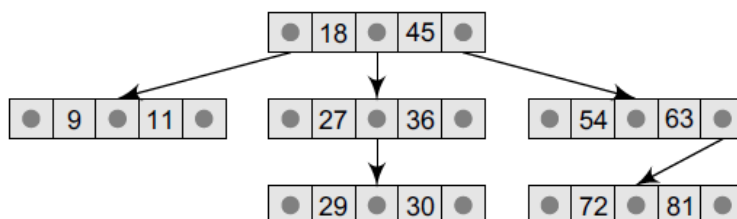
- Deletemin operation is deleting the minimum element from the heap.
- In min heap the minimum element is found in the root.
- When this minimum element is removed, a hole is created at the root.
- Since the heap becomes one smaller, make the last element X in the heap to move somewhere in the heap.
- If X can be placed in the hole, without violating heap order property, place it , otherwise slide the smaller of the hole's children into the hole, thus , pushing the hole down one level.
- Repeat this process until X can be placed in the hole.

## UNIT- IV

### PART A

#### 1. What is M-way tree?

Multiway Search Tree (M-way tree) is a search tree which has  $M - 1$  values per node and  $M$  subtrees. In such a tree,  $M$  is called the degree or order of the tree. Example:



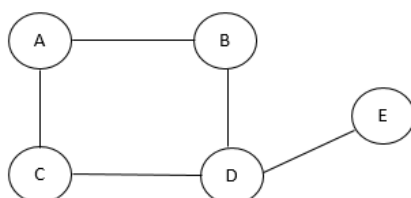
#### 2. Differentiate B tree and B+ Tree

B Tree	B+ Tree
B trees store data in leaf nodes and internal nodes.	B+ trees store data only in the leaf nodes. All other nodes (internal nodes) are called index nodes and store index values.
The leaf nodes of a B tree are not linked using linked list	The leaf nodes of a B+ tree are often linked to one another in a linked list.
Searching is less efficient than B+ tree	Searching is more efficient than B tree

#### 3. Define Graph?

A graph  $G = (V, E)$  consists of set of vertices  $V = \{v_1, v_2, \dots, v_n\}$  and set of edges that connect the vertices  $E = \{e_1, e_2, \dots, e_m\}$ .

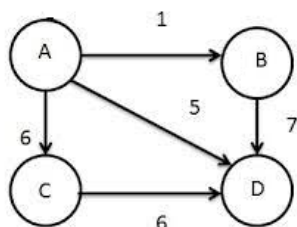
Example:



#### 4. What is weighted graph?

A graph in which weights are assigned to every edge is called a weighted graph.

Example:



#### 5. Define adjacency matrix?

Adjacency Matrix  $A$  is a two dimensional array of size  $n \times n$  where  $n$  is the number of vertices in a graph.  $A[i][j] = 1$  indicates that there is an edge from vertex  $i$  to vertex  $j$ . Otherwise it is 0.

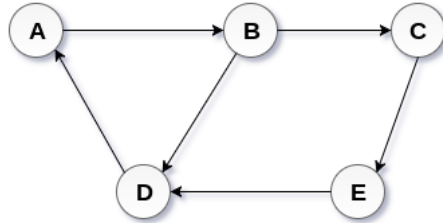
## 6. Define adjacent nodes?

Two vertices in a graph are adjacent if there is an edge connecting the vertices. For example, if an edge  $e$  is associated with a pair of nodes  $(u,v)$ , then we say that  $u$  and  $v$  are adjacent nodes.

## 7. What is a directed graph?

A directed graph or digraph is a graph in which each edge of the graph is associated with some direction and the traversing can be done only in the specified direction.

Example:



## 8. What do you mean by loop and cycle in a graph?

An edge that is associated with the same end points can be called as Loop. A cycle can be defined as the path that starts and ends on the same vertex. A path can be defined as the sequence of nodes that are followed in order to reach some terminal node  $U$  from the initial node  $V$ .

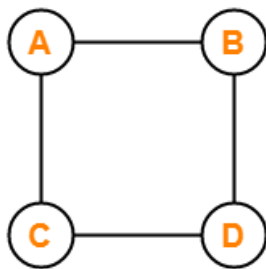
## 9. Define in-degree and out-degree in a graph?

A degree of a node is the number of edges that are connected with that node. A node with degree 0 is called as isolated node.

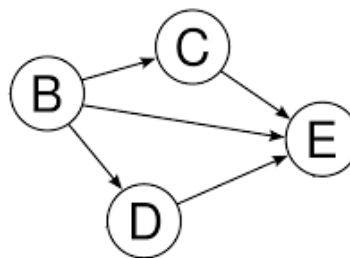
In a directed graph, for any node  $v$ , the number of edges coming towards  $v$  is called the in-degree of the node  $v$ . Number of edges having the node  $v$  as root node is called the out-degree of the node  $v$ .

## 10. Distinguish between cyclic and acyclic graph

A graph which have any cycle is called a cyclic graph. A cycle can be defined as the path that starts and ends on the same vertex. A graph which does not have any cycles is called an acyclic graph.



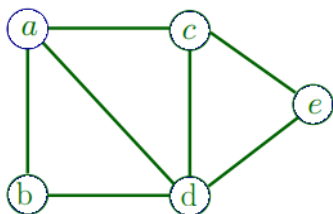
Cyclic Graph



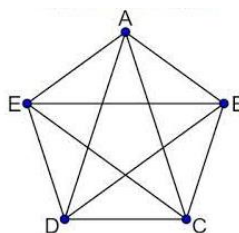
Acyclic Graph

## 11. What is meant by connected graph and complete graph?

A connected graph is the one in which some path exists between every two vertices in  $V$ . There are no isolated nodes in connected graph. A complete graph is the one in which every node is connected with all other nodes.



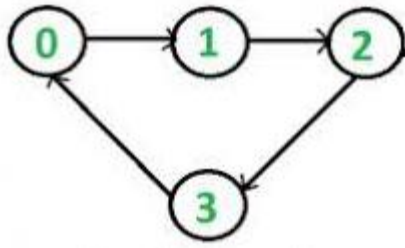
Connected Graph



Complete Graph

**12. When a graph said to be strongly connected?**

A directed graph is called strongly connected if there is a path between each pair of vertices of the graph. Example:



**13. Name the different ways of representing a graph?**

- a. Adjacency matrix
- b. Adjacency list

**14. What are the two graph traversal strategies?**

- a. Breadth first search (BFS)
- b. Depth first search (DFS)

**15. What is DAG?**

Directed acyclic graph (DAG) is a directed graph with no cycles. Example:

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

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

**17. What is topological sort?**

It is an ordering of the vertices in a directed acyclic graph (DAG), such that if there is a path from vertex  $u$  to  $v$ , then  $v$  appears after  $u$  in the ordering.

**18. Write an algorithm for DFS**

1. Visit the starting vertex  $V$  and explore the vertex
2. Exploration of a vertex  $V$  is suspended as soon as a new vertex  $W$  is reached
3. Then the exploration of this new vertex  $W$  begins.
4. After this new vertex has been explored, the exploration of  $V$  continues.
5. The search terminates when all reached vertices have been fully explored

**19. Define biconnected graph?**

A graph  $G$  is biconnected if and only if it contains no articulation points. A vertex  $v$  in a connected graph  $G$  is an **articulation point** if and only if the deletion of vertex  $v$  together with all edges disconnects the graph into two or more nonempty components. The articulation vertex is also called as **cut vertex**

**20. What is minimum spanning tree? Name the algorithms to find minimum spanning tree.**

A minimum spanning tree is tree constructed from graph with sum of the costs of the edges in that tree is minimum. The algorithms used to find minimum spanning tree are

- Prim's Algorithm
- Kruskal's Algorithm

**21. What is Euler circuit?**

Euler circuit (or Euler cycle) is an Euler path that starts and ends on the same vertex. Euler path (or Euler tour) is a path in a graph that visits every edge exactly once.

## 22. What is Hamiltonian graph?

Hamiltonian graph is a connected graph that contains a Hamiltonian Circuit. Hamiltonian circuit is a Hamiltonian path that starts and ends on the same vertex. Hamiltonian path is a path in a graph that visits each vertex exactly once.

## PART B

### 1. Outline Breadth First Traversal and Depth First Traversal with example.

#### Breadth First Search

The Breadth first search is one of the systematic approaches for exploring and searching the vertices/nodes in a given graph. Queue is used in the implementation of the breadth first search.

1. Select the start vertex/source vertex. Visit the vertex and mark it as one (1) (1 represents visited vertex).
2. Enqueue the vertex.
3. Dequeue the vertex.
4. Find the Adjacent vertices.
5. Visit the unvisited adjacent vertices and mark as 1.
6. Enqueue the adjacent vertices.
7. Repeat from Step – 3 to Step – 6 until the queue becomes empty.

#### Depth First Search

The Depth first search is another one systematic approach for exploring the vertices/nodes in a given graph. Stack is used in the implementation of the Depth first Search

1. Select the start vertex.
2. Visit the vertex.
3. Push the vertex on to the stack.
4. Pop the vertex.
5. Find the adjacent vertices, and select any 1 of them.
6. Repeat from Step – 4 to Step – 5 until the stack becomes empty.

### 2. Outline the steps in the Dijkstra's Shortest path algorithm with an example.

1. Create a set sptSet (shortest path tree set). Initially, this set is empty.
2. Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign the distance value as 0 for the source vertex so that it is picked first.
3. While sptSet doesn't include all vertices
  - Pick a vertex u that is not there in sptSet and has a minimum distance value.
  - Include u to sptSet.
  - Then update the distance value of all adjacent vertices of u.
  - To update the distance values, iterate through all adjacent vertices.
  - For every adjacent vertex v, if the sum of the distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

### 3. Illustrate Prim's and Kruskal algorithm for minimum spanning Tree with an example.

#### Prim's algorithm:

1. Select any vertex as root node
2. Choose the shortest edge connected to that vertex
3. Select the shortest edge connected to any vertex already connected. Inclusion of selected edge should not form a cycle.
4. Repeat step 3 until all vertices have been connected

### **Kruskal Algorithm**

1. Sort all edges in increasing order of their edges weights.
2. Pick the smallest edges
3. Check if the new edge creates a cycle or loop in spanning trees.
4. If it doesn't form the cycle, then include that edge in MST. otherwise discard it
5. Repeat from step 2 until all the vertices have been connected

### **4. Describe in detail about the various representation of a graph.**

#### **Adjacency Matrix**

- Create an  $n \times n$  matrix where  $n$  is the number of vertices in the graph.
- Initialize all elements to 0.
- For each edge  $(u, v)$  in the graph, if the graph is undirected mark  $a[u][v]$  and  $a[v][u]$  as 1,
- If the edge is directed from  $u$  to  $v$ , mark  $a[u][v]$  as the 1.
- If the graph is weighted, cells are filled with edge weight

#### **Adjacency List**

- In Adjacency List, we use an array of a list to represent the graph.
- The list size is equal to the number of vertices ( $n$ ).
- $Adjlist[0]$  will have all the nodes which are connected to vertex 0.
- $Adjlist[1]$  will have all the nodes which are connected to vertex 1 and so on.

### **5. Explain the Topological sorting with an example.**

#### **Definition**

It is an ordering of the vertices in a directed acyclic graph (DAG), such that if there is a path from vertex  $u$  to  $v$ , then  $v$  appears after  $u$  in the ordering.

#### **Procedure for Topological sorting**

1. Find a vertex that has indegree = 0 (no incoming edges)
2. Remove all the edges from that vertex that go outward (make its outdegree = 0, remove outgoing edges)
3. Add that vertex to the array representing the topological sorting of the graph
4. Repeat till there are no more vertices left.

### **6. Illustrate B tree and B+ tree with an example**

#### **B-Tree:**

B-Tree is known as a self-balancing tree as its nodes are sorted in the inorder traversal. In B-tree, a node can have more than two children.

In the B-tree data is sorted in a specific order, with the lowest value on the left and the highest value on the right. To insert the data or key in B-tree is more complicated than a binary tree. Some conditions must be held by the B-Tree:

- All the leaf nodes of the B-tree must be at the same level.
- Above the leaf nodes of the B-tree, there should be no empty sub-trees.
- B- tree's height should lie as low as possible.

**B+ Tree**

It eliminates the drawback B-tree used for indexing by storing data pointers only at the leaf nodes of the tree. Thus, the structure of leaf nodes of a B+ tree is quite different from the structure of internal nodes of the B tree. It may be noted here that, since data pointers are present only at the leaf nodes, the leaf nodes must necessarily store all the key values along with their corresponding data pointers to the disk file block, to access them.

Moreover, the leaf nodes are linked to providing ordered access to the records. The leaf nodes, therefore form the first level of the index, with the internal nodes forming the other levels of a multilevel index. Some of the key values of the leaf nodes also appear in the internal nodes, to simply act as a medium to control the searching of a record.

## UNIT – V

### PART A

#### 1. What is meant by Sorting?

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

#### 2. List the different sorting algorithms.

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

#### 2. Why bubble sort is called so?

The bubble sort gets its name because as array elements are sorted they gradually “bubble” to their proper positions, like bubbles rising in a glass of soda.

#### 3. State the logic of bubble sort algorithm.

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

#### 4. When does the Bubble Sort Algorithm stop?

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

#### 5. State the logic of selection sort algorithm.

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

#### 6. What is the output of selection sort after the 2<sup>nd</sup> iteration given the following sequence? 16 3 46 9 28 14

Ans: 3 9 46 16 28 14

#### 7. How does insertion sort algorithm work?

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

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

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

#### 9. How many key comparisons and assignments an insertion sort makes in its worst case 21

The worst case performance in insertion sort occurs when the elements of the input array are in descending order. In that case, the first pass requires one comparison, the second pass requires two comparisons, third pass three comparisons,...k<sup>th</sup> pass requires (k-1), and finally the last pass

requires  $(n-1)$  comparisons. Therefore, total numbers of comparisons are:  
 $f(n) = 1+2+3+\dots+(n-k)+\dots+(n-2)+(n-1) = n(n-1)/2 = O(n^2)$

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

Insertion sort as there is no movement of data if the list is already sorted and complexity is of the order  $O(N)$ .

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

Insertion sort is easily adaptable to singly linked list. In this method there is an array link of pointers, one for each of the original array elements. Thus the array can be thought of as a linear link list pointed to by an external pointer first initialized to 0. To insert the  $k^{\text{th}}$  element the linked list is traversed until the proper position for  $x[k]$  is found, or until the end of the list is reached. At that point  $x[k]$  can be inserted into the list by merely adjusting the pointers without shifting any elements in the array which reduces insertion time.

**12. Why Shell Sort is known diminishing increment sort?**

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

**13. Which of the following sorting methods would be especially suitable to sort a list L consisting of a sorted list followed by a few “random” elements?**

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

**14. Which sorting algorithm follows the divide-and-conquer strategy?**

Quick sort and Merge sort

**15. Mention the different ways to select a pivot element in quick sort.**

The different ways to select a pivot element are

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

**16. What is divide-and-conquer strategy?**

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

**17. Compare quick sort and merge sort.**

Quicksort has a best-case linear performance when the input is sorted, or nearly sorted. It has a worst-case quadratic performance when the input is sorted in reverse, or nearly sorted in reverse. Merge sort performance is much more constrained and predictable than the performance of quicksort. The price for that reliability is that the average case of merge sort is slower than the average case of quicksort because the constant factor of merge sort is larger.

**18. What is the key idea of radix sort?**

Sort the keys digit by digit, starting with the least significant digit to the most significant digit.

**19. Define Searching.**

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

**20. What is linear search?**

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

**21. What is Binary search?**

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

**22. Define hash function?**

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

**23. What are the important factors to be considered in designing the hash function? (Nov 10)**

- To avoid lot of collision the table size should be prime
- For string data if keys are very long, the hash function will take long to compute.

**24. What are the problems in hashing?**

- Collision
- Overflow

**25. What do you mean by hash table?**

The hash table data structure is merely an array of some fixed size, containing the keys. A key is a string with an associated value. Each key is mapped into some number in the range 0 to  $\text{tablesize}-1$  and placed in the appropriate cell.

**27. What do you mean by hash function?**

A hash function is a key to address transformation which acts upon a given key to compute the relative position of the key in an array. The choice of hash function should be simple and it must distribute the data evenly. A simple hash function is  $\text{hash} = \text{key} \bmod \text{tablesize}$ .

**28. What do you mean by separate chaining?**

Separate chaining is a collision resolution technique to keep the list of all elements that hash to the same value. This is called separate chaining because each hash table element is a separate chain (linked list). Each linked list contains all the elements whose keys hash to the same index.<sup>2,3</sup>

## PART B

### 1. Distinguish between linear search and binary search. State and explain algorithm for both search with an example.

#### Linear Search

It is also called sequential search. In linear search input data need not to be in sorted. The time complexity of linear search  $O(n)$ . It is simple and less complex

#### Linear Search Algorithm,

- Every element is considered as a potential match for the key and checked for the same.
- If any element is found equal to the key, the search is successful and the index of that element is returned.
- If no element is found equal to the key, the search yields “No match found”.

#### Binary Search

In binary search input data need to be in sorted order. The time complexity of binary search  $O(\log n)$ . It is very fast search algorithm, but it is more complex.

#### Binary Search algorithm:

1. The data structure must be sorted.
2. Compare the middle element of the search space with the key.
3. If the key is found at middle element, the process is terminated.
4. If the key is not found at middle element, choose which half will be used as the next search space.
5. If the key is smaller than the middle element, then the left side is used for next search
6. If the key is larger than the middle element, then the right side is used for next search.
7. This process is continued until the key is found or the total search space is exhausted.

### 2. Outline the algorithm to Sort an array of ‘N’ numbers using bubble sort. Illustrate each step of the algorithm with an example.

1. Start with the first element.
2. Compare the current element with the next element.
3. If the current element is greater than the next element, then swap both the elements.
4. If not, move to the next element.
5. Repeat steps 1 – 4 until we get the sorted list.

### 3. Outline the steps in the insertion sort algorithm with an example.

1. Add the first element in sorted sub-list.
2. Pick next element
3. Compare it with all elements in the sorted sub-list
4. Shift all the elements in the sorted sub-list that is greater than the value to be sorted
5. Insert the value
6. Repeat step-2 to 5 until list is sorted

### 4. Write an algorithm for Merge sort with suitable example

Merge sort is a sorting algorithm that uses the divide, conquer, and combine algorithmic paradigm.

- Divide means partitioning the  $n$ -element array to be sorted into two sub-arrays of  $n/2$  elements.
- Conquer means sorting the two sub-arrays recursively using merge sort.
- Combine means merging the two sorted sub-arrays of size  $n/2$  to produce the sorted array of  $n$  elements.

## 5. Explain Hashing and Extensible hashing

### Hashing

A hash function is a key to address transformation which acts upon a given key to compute the relative position of the key in an array. The choice of hash function should be simple and it must distribute the data evenly. A simple hash function is  $\text{hash} = \text{key} \bmod \text{tablesize}$ .

This process of mapping the keys to appropriate locations (or cell) in a hash table is called hashing. If, when an element is inserted, it hashes to the same value as an already inserted element, then we have a collision and need to resolve it. Collision resolution methods are

1. Separate Chaining
2. Open Addressing
  - a) Linear Probing
  - b) Quadratic Probing
  - c) Double hashing
3. Rehashing

### Extensible Hashing

Extensible Hashing is a mechanism for altering the size of the hash table to accommodate new entries when buckets overflow. Common strategy in internal hashing is to double the hash table and rehash each entry. However, this technique is slow, because writing all pages to disk is too expensive.

Therefore, instead of doubling the whole hash table, we use a directory of pointers to buckets, and double the number of buckets by doubling the directory, splitting just the bucket that overflows. Since the directory is much smaller than the file, doubling it is much cheaper. Only one page of keys and pointers is split.