DSA

Data types - It is a set of data with predefined values.

e.g.- integers, floating, character etc.

2- types of data types - 1) predefined dt.
2) user-defined dt.

Data structure - Data structure is a particular way of storing and organizing data in a Computer so that it can be used efficiently.

Depending on the organisation of elements, DS are classified into two types:

1) **Linear data structure**: Elements are accessed in a sequential order but it is not compulsory to store all elements sequentially.
Ex- Linked Lists, Stack, Queues.

2) **Non-linear data structure** - Elements of this ds are stored/accessed in a non-linear order.
ex- Trees and graphs.

Page No ....

Date..................

Abstract data types (ADTs) - To simplify the process of solving problems, we combine the data structure with their operations and we call this Abstract data types.

Commonly used ADT include: Linked lists, Stacks, Queues etc.

Recursion - Any function which calls itself is called recursive.

Recursion	Iteration
1) Terminates when a base case is reached.	1) Terminate when a condition is proven to be false
2) Each recursive call require extra space on the stack frame. (memory).	2) Each iteration does not require extra space.
3) If we get infinite recursion, the program may run out of memory and result in stack overflow.	3) An infinite loop could loop forever since there is no extra memory being created.
4) Solution to some problems are easier to formulate recursively.	4) Iterative solutions to a problem may not always be as obvious as a recursive solution.

Bactracking - Backtracking is an improvement of the brute force approach.

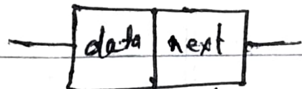
* It searches for a sol^n to a problem among all available options.
* In backtracking, we start with one possible option out of many available options and try to solve the problem if we are able to solve the problem with the selected move then we will print the sol^n else we will backtrack and select some other option and try to solve it.
* If none of the options work out we will claim that there is no sol^n for the problem.

Ex. algo. of Backtracking - N-Queens problem, Knapsack problem, Graph coloring problem, Hamiltonian graph

Linked Lists - A linked list is a data structure used for storing collection of data.

A Linked list has following properties -

* Successive elements are connected by pointers
* The last element points to NULL.
* Does not waste memory space (but takes some extra memory for pointers.)



Node

Disadvantage of Arrays - 1) Fixed size
2) preallocate memory waste memory space for indices in the array that are empty.

3) Complex position-based insertion - To insert an element at a given position, we may need to shift the existing elements.

Dynamic Arrays - Growable array / Resizable array / array li
Dynamic array automatically grows when we try to make an insertion and there is no more space left for the new item. Usually the area doubles in size.

* A dynamic array can be constructed by allocati an array of fixed size, typically larger than the number of elements immediately required

Dynamic array syntax ~~in C~~

1) In C - using malloc ()

```
int *ptr
    ptr = (int*) malloc(sizeof(int));
```

2) In C++ -

```
int *array = new int [size];
```

3) In java -

```
int array[] = new int [size];
```

* Basic operation on a list

1) Traversing 2) Insertion 3) Deleting.

Linked List Syntax

Page No .... Date ....

Linked list create in C →

C

```
struct Node {
    int data;
    struct Node *next;
};
```

C++

```
Class Node {
public: int data;
        Node *next;
};
```

java

```
Class LinkedList {
    Node head;
  class Node {
      int data;
      Node next;
      Node(int d)
        { data = d; }
  }
}
```

Array →

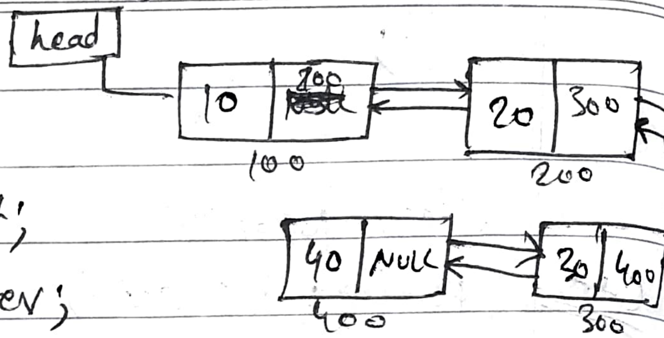
1)
2) continuous memory
3) index start from zero to n-1
4) char [arr] → String
5) char a[0]; X

Doubly Linked list - The advantage of a doubly (two-way linked list) linked list is that given a node in the list, we can navigate in both directn.

* A node in a singly linked list cannot be removed unless we have the pointer of its previous node.
* But in a doubly linked list, we can delete a node even if we don't have the previous node's address.

d) Advantage of doubly linked list –
1) Each node requires an extra pointer (more space)
2) operations takes a bit longer.

Syntax –
```
Struct Node {
  int data;
  struct Node *next;
  struct Node *prev;
};
```

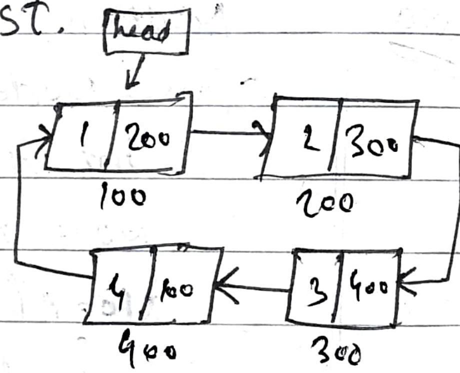


— Doubly linked list —

Circular Linked List – A linked list whose last node point the first node (or) endless linked list.

* There is no NULL pointer.

```
struct Node {
    int data;
    struct Node *next;
};
```



— Circular linked list —

Application – Circular linked list are used in managing the computing ~~process~~ resources of a computer. we can use circular lists for implementing stack and ques.

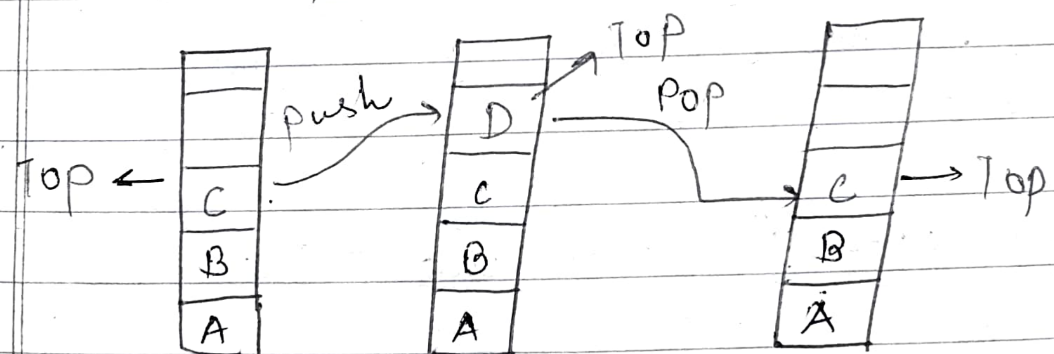
* One of the biggest advantage of linked lists over array is that inserting an element at any location takes only O(1) time.

Stack – A simple data structure used for storing data.

* In a stack, the order in which the data arrives is important.

Definatn – A stack is an ordered list in which insertion and deletion are done at one end called TOP. The last element inserted is the first one to be deleted. Hence, it is called the Last in First out (LIFO) or First in last out (FILO) list.

* **push** – When element is inserted in a stack.
* **pop** – When element is removed from a stack
* **Underflow** – Trying to pop out an empty stack
* **Overflow** – Trying to push an element in a full stack.

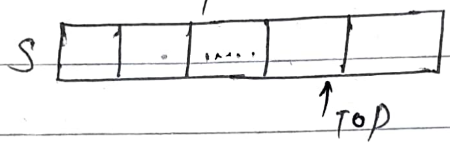


Applicatn – * Balancing of Symbol
* Infix-to-postfix Conversion.
* page-visited history in a web brower
* Undo sequence in text-editor.
* Matching tags in HTML and XML.

I Simple Array Implementation

Stack Implementation

This implementation of stack ADT uses an array. In this array, we add elements from left to right and use a variable to keep track of the index of the top element.



The array storing the stack elements may become full. A push operation will then throw a full stack exception.
Similarly, if we try deleting an element from an empty stack it will throw stack empty exception.

Limitation - The maximum size of the stack must first be defined and it cannot be changed. Trying to push a new element into a full stack causes an implementation-specific exception.

II Linked List Implementation

The other way of implementing stack is by using linked lists. push operation is implemented by inserting element at the begining of the list. POP operation is implemented by deleting the node from the beginning (head node).

* [Rear + 1] % size == FRONT
↳ condition to check circular Queue.

Total 9 interface in collection

Page No ....

Date..........

1) **Collection** - Collection of heterogeneous (dissimilar type or similar types) objects ~~as~~ etc: - Integer, String Class, User defined

2)

	Array	Collection
1)	Similar kind of element	Similar and dissimilar
2)	Fixed in size	Growable in nature (increase and decrease)
3)	Not supported method	support methods
4)	faster	slower

3) **Collection framework** - classes and interface

Iterable (base interface)

↑

Collection (1.2 version)

→ List (1.2)
↳ duplicate
↳ insertion order preserved

→ set (1.2)
↳ no duplication
↳ insertion order is not preserved

→ Queue (1.5)
↳ follow FIFO
↳ duplicate
↳ preserve ins. order

2 way → Collection
→ map (Interface)

Page No ....

Date..................

Implementation class-

List → ArrayList (1.2) version (In)
→ Linkedlist (1.2) version
→ vector (1.0)] → lagacy classes
↓
→ stack (1.0)

Set → hashset (class) (1.2) ~~(interface)~~ → [LinkedHashset] (1.4)
→ SortedSet (1.2) (interface) → NavigableSet (1.6) (interface) → TreeSet (1.2) (class)

Queue → priorty Queue (1.5)
→ BlockingQueue → PBQ, LBQ (1.5)

(1.2) Map - come from mapping (key, value)

Map → HashMap (1.2) → LinkedHashMap (1.4)

Map → SortedMap (1.2) → NavigableMap (1.6) → Tree~~Set~~ Map (1.2)

Map → [HashTable] (1.0) → [properties] (1.0)

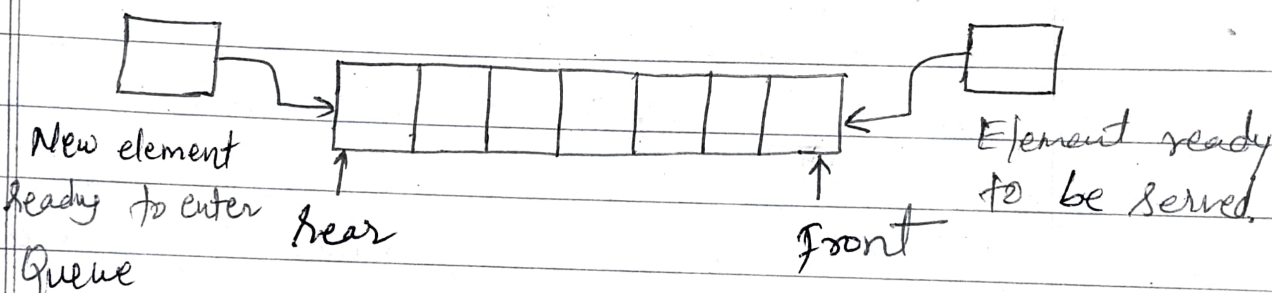
[Dictionary] Abstract class. (1.0)

Queue - A queue is a data structure used for storing data.

* In Queue, the order in which data arrives is important.

→ A queue is an ordered list in which insertion are done at one end (rear) and deletions are done at other end (front).

* The first element to be inserted is the first one to be deleted.
* Hence it is called first in first out (FIFO) or LIFO.



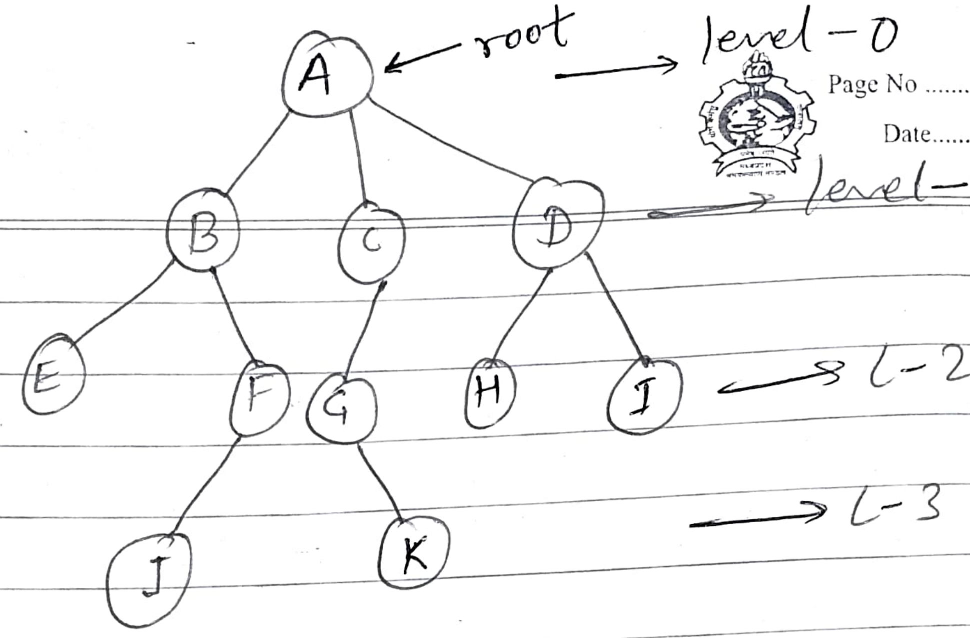
Direct application - 1) OS schedule jobs.
2) Multiprogramming.

Trees - A tree is a data structure similar to a linked list but each node points to a numbers of nodes.

* Tree is an example of a non-linear data structure.
* In tree ADT, the order of the element is not important. If we need odering information data structure liked linked list, stack, queue etc.

Page No ....

Date..................



→ The root of a tree is the node with no parents. There can be at most one root node in a tree (A)

→ An edge refers to the link from parent to child.

→ A node with no children is called leaf node (E, J, K, H and I)

→ Children of same parents are called siblings (B, C, D are siblings of A)

→ The set of all nodes at a given depth is called the level of the tree (B, C, D are the same level)

→ The depth of a node is the length of the path from the root to the node (G → 2)

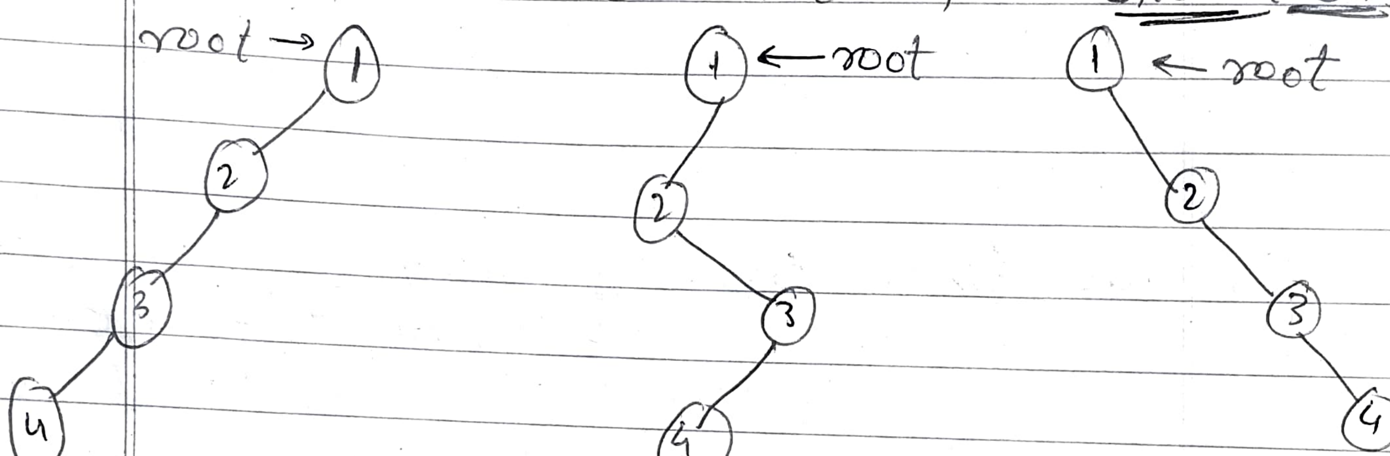
→ The height of a node is the length of the path from that node to the deepest node.

→ The height of tree is the maximum heig

among all the nodes in the tree and
→ depth of tree is the maximum depth among all the nodes in the tree.

depth of tree = height of tree (always)

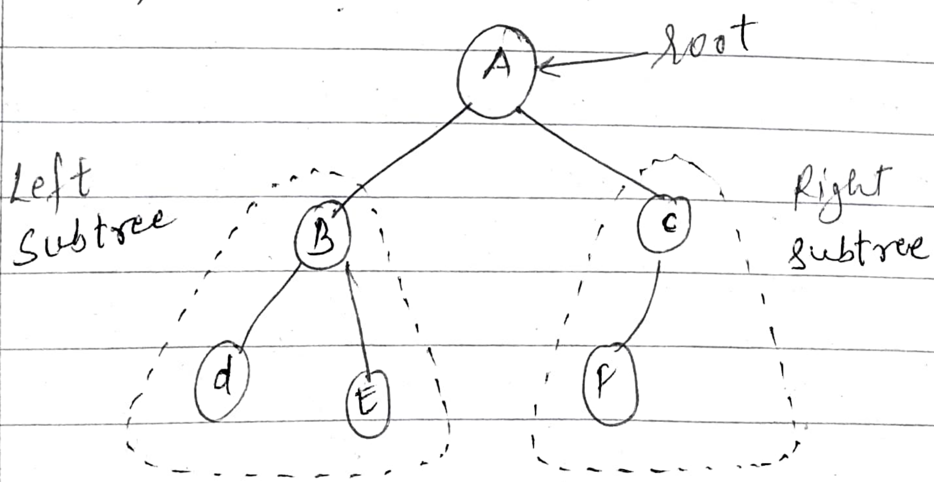
→ If every node in a tree has only one child then we call such trees skew trees



left Skew tree Skew tree Right skew tree

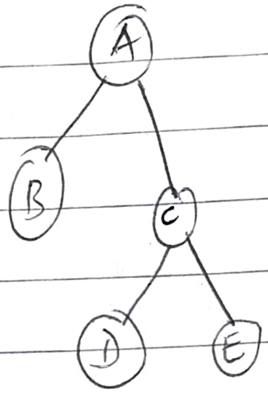
Binary trees: A tree is called binary tree if each node has zero child, one child or two children.

* Empty tree is also a valid binary tree.

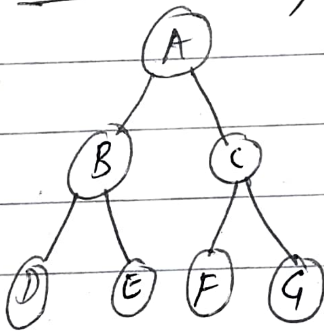


Types of Binary trees-

1) strict Binary tree - A binary tree is called strict binary tree if each node has exactly two children or no children.

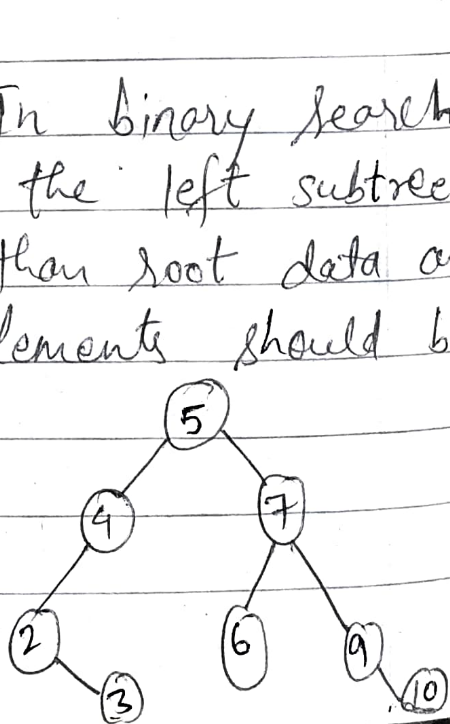


2) full binary tree :- A binary tree is called full binary tree if each node has exactly two children and all leaf nodes are at the same level.



In trees, the default flow is from parent to children.

Bineary Search tree - In binary search trees, all the left subtree elements should be less than root data and all the right subtree elements should be greater than root data.



find no. of nodes at any level.
$h \rightarrow$ height (level)
$h = 2 \rightarrow$ No. of nodes $= 2^h$

Important Notes – * Since root data is always between left subtree data and right subtree data, performing inorder traversal on binary search tree produces a sorted list.

Binary Search tree declaration –

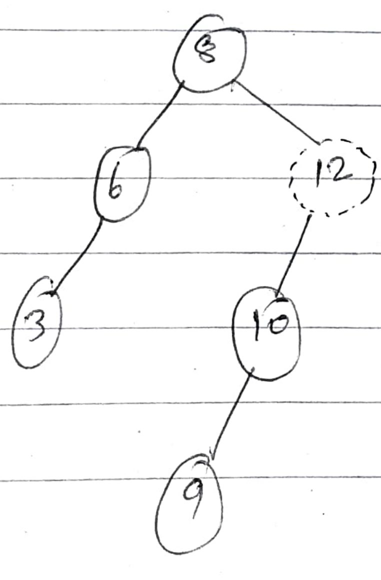
inc

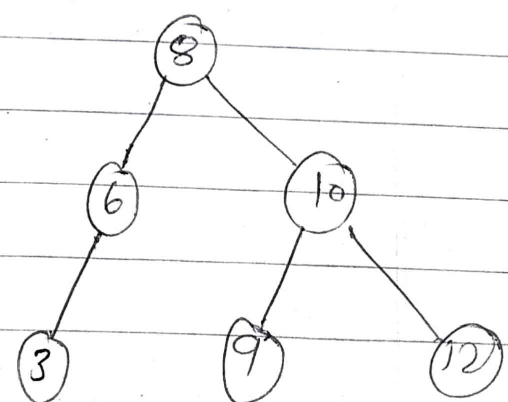
```
struct BinarySearchTreeNode {
  int data;
struct BinarySearchTreeNode *left;
struct BinarySearchTreeNode *right; };
```

AVL Tree (Adelson-Velskii and landis)

In HB(K), if K=1 (if balance factor is one), such a binary search tree is called an AVL tree.

left subtree height – right subtree = 1(max)





AVL tree declaration


```
Struct AVLTree {
    Struct AVLTree *left;
    int data;
    struct AVLTree *right;
    int right; };
```

AVL Rotation - Rotations is the technique used for restoring the AVL tree property.

* To restore the AVL tree property, we start at the insertion point and keep going to the root of the tree.

Red-black trees: A red black tree is a binary search tree that satisfies the following properties:

1) Root property :- the root is black
2) External property :- every leaf is black
3) Internal property :- children of a red node are black
4) Depth property :- all the leaves have the same black.

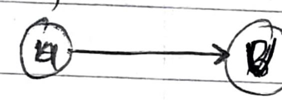
* Same as ~~Red~~ AVL tree, the Red-black tree becomes imbalanced, then we perform rotation to reinforce the balancing property.

Splay trees - Splay trees are BSTs with a self-adjusting property.

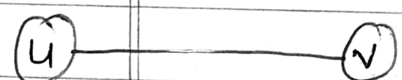
* Splay trees is starting with an empty tree.

Graph - A graph is a pair (V, E) where V is a set of nodes, called vertices, and E is a collection of pairs of vertices called edges.

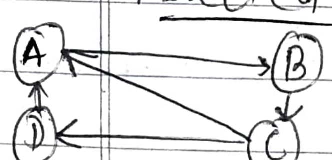
→ **Directed edge** - * ordered pair of vertices (u, v)
* first vertex u is the origin
* Second vertex v is destination
* Ex - One way road traffic



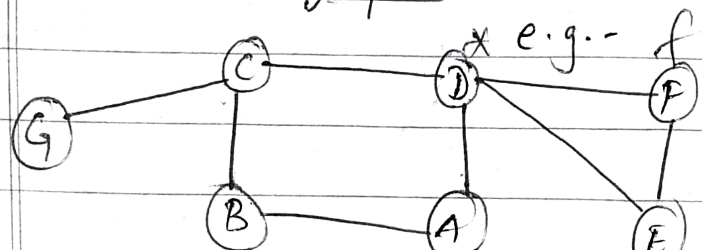
→ **Undirected edge** - * Unordered pair of vertices.
* e.g - railway lines.



→ **Directed graph** - * All the edges are directed
* e.g. - Route network.

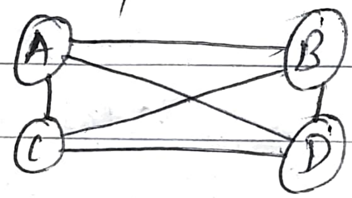


→ **Undirected graph** - * all the edges are undirected
* e.g. - flight network.



* A graph with no cycles is called tree.
A tree is an acyclic connected graph.

* Degree of a vertex is the number of edges incident on it.
* A directed acyclic graph [DAG] is a directed graph with no cycles.
* A forest is a disjoint set of trees.
* Graphs with all edges present are called complete graphs.



Graph Traversals - Two algorithms for traversing the graphs.
1) Depth First search (DFS)
2) Breadth First search (BFS)

→ Depth first search - DFS algo works in a manner similar to preorder traversal of the trees.
* Like preorder traversal, this algorithm also uses stack.

→ Breadth first search - BFS algorithm works similar to level-order traversal of the trees.
* BFS uses queue.

* Shortest path algorithm also known as Dijkstra
* Dijkstra's algo is a generalization of the BFS algorithm.

Page No ....

Date..................

* If the graph has negative edges costs, then Dijkstra's algorithm does not work. then here we use Bellman-Ford Algorithm

DFS	BFS
1) Backtracking is possible from a dead end.	1) Backtracking is not possible.
2) Vertices processed in a LIFO order.	2) Vertices processed in a LILO order.
3) Search is done in a particular direction.	3) vertices at the same level are maintained in parallel.

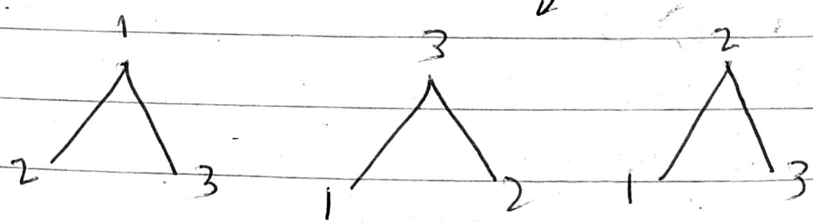
* formula - to find how many trees we can construct for given graph

$n \rightarrow$ vertices

$$n^{n-2} \text{ trees.}$$

e.g. ① $n = 2 \Rightarrow 2^0 = 1$ A —— B

② $n = 3 \Rightarrow 3^1 = 3$



Page No ....

Date..................

Sorting - Sorting is an algorithm that arranges the elements of a list in a certain order (ascending or descending)

Need of sorting - sorting can significantly reduce the complexity of a problem, and is often used for database algorithms and searches.

(1) **Bubble sort** - Bubble sort is the simplest sorting algorithm.

* Insertion sort has better performance than bubble sort.
* works by repeatedly swapping the adjacent elements if they are in wrong order.

e.g. - → 5 4 1 10 3

→ 4 5 1 10 3

Step-1 → 4 1 5 10 3

→ 4 1 5 10 3

→ 4 1 5 3 10

repeat till n×n

Code :-

```
for (i = 0; i<n-1; i++){
    for (j=0; j<n; j++){
        if (A[j] > A[j+1]){
            temp = A[j];
            A[j] = A[j+1];
            A[i+1] = temp;   }
        }
}
```

2) Selection Sort - Selection sort is an in-place sorting algorithm.

Page No .... Date........

* It is used for sorting the files with very large values and small keys.

* The selection sort algorithm sort an array by repeatedly finding the minimum element from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

1) The subarray which is already sorted.
2) Remaining subarray which is unsorted.

Code-

* Bring the smallest value at front of array.

```
for (i=0;i<n-1;i++)
{   min =i;
    for (j=i+1;j<n;j++)
    {  if(A[j]< A[min])
       {
          min =j;
       }
    }
    temp = A[min];
    A[min]= A[i];
    A[i] = temp;
}
}
```

(3) Insertion sort - It is a simple and efficient comparison sort.

* Each iteration removes an element from the input data and inserts it into the correct position in the list being sorted.

Code.-

```
for (i=1; i<=n-1 ;i++)
{   v= A[i];
    j = i;
    While (A[j-1]>v && j>=1) {
        A[j] = A[j-1];
        j--;
    }
    A[j] = v;
  }
}
```

(4) Shell sort

(5) Merge sort - Merge sort is an Example of divide and conquer Strategy.

* Selection → splits a list into two lists.
* Merging → joins two files to make one file.
* Merge sort is Quick sort's Compliment.
* This algo is used for