1. Basic Concept and Introduction to Data Structure.

1. What is data structure?

→ ① Data structure is used to describe the way data is stored. and the term algorithm is used to describe the way data is processed.

② Algorithm + data structure = program.

2. What is Pointers in Data structure

→ ① A pointer in C is used to allocate memory dynamically i.e at run time.

② Normal variable stores the value, pointer variable stores the address of the variable.

③ C pointer initialized to null i.e int *p = null. *p = null.

④ Value of null pointer is 0.

⑤ Pointer allocate memory at run time.

⑥ Pointer is a variable which stores the address of another variable.

⑦ & Symbol is used to get the address of variable.

⑧ * Symbol is used to get value of the variable that the pointer is pointing to.

<stdlib.h>

3. What is Dynamic Memory Allocation.

→

① The Process of allocating memory at run time is known as dynamic memory allocation.

② There are 4 library functions provided by C defined Under <stdlib.h> header file.

③ Dynamic memory allocation functions -

	Function	description
i.	malloc()	it is Used to allocate a Specified number of bytes in memory. Returns a pointer to the beiginning of allocated block.
ii.	calloc()	it is Similar to malloc(), but initializes the allocated bytes to zero. This function also allows us to allocate memory for more than one Object at a time.
iii.	free()	it is Used to free up memory that was previously allocated with malloc(), calloc(), realloc();
iv.	realloc()	it is Used to change the Size of previously allocated block.

4. What is algorithm and write down its characteristics.

→

① An algorithm is a finite sequence of instruction each of which has a clear meaning and can be executed with a finite amount of effort in finite time.

② Following are the characteristics.

a. Input - An algorithm should accept zero or more inputs.
b. output - An algorithm must produce atleast 1 result
c. Definiteness - Each step is an algorithm must be clear and unambiguous.
d. finiteness - An algorithm must halt.
e. Effectiveness - An algorithm must be done exactly and in finite time.

③ Advantages of algorithm

a. it gives language independent layout to progra
b. Program can be developed in any desired laungaage
c. Representation is in simple english lang.
d. It is very easy to understand
e. Facilitase easy coding.

5. What is algorithm analysis

→

① Performance analysis of an algorithm is the process of calculating space required by that algorithm and time required by that algorithm.

* Space Complexity

② "Total amount of Computer memory required by an algorithm to complete its execution is called space complexity of that algorithm"

③ Program use Computer memory for 3 reasons
i. Instruction Space
ii. Enviornmental Stack
iii. Data Space;

* Time Complexity

① "The time Complexity of an algorithm or program is calculated is the total amount of the time required by an algorithm to complete its execution."

② Time required by each statement depends on.
i. Time required for executing it once
ii. Number of times the statement is executed.

Big-oh(O)
Big-omega(Ω)

③ Types of time Complexity
i. Best Case
ii. Worst Case
iii. Average Case.

6. What is Asymptotic Notation.
→

① Asymptotic Notation is of an algorithm is a mathematical representation of its Complexity.

② Asymptotic notation is Used to describe the running time of an algorithm. How much time an algorithm takes with given input.

③ There are two types of asymptotic notation.
i) Big-oh(O) Big-oh(O)
ii) Big-omega(Ω) Big-omega(Ω)

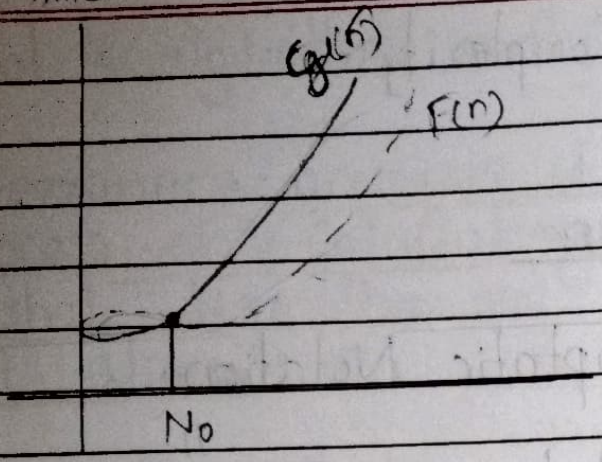
④ I] Big-oh(O) :-

① Big-oh notation is Used to define Upper bound of an algorithm in terms of time Complexity.

② Big-oh notation always indicates the maximum time required by an algorithm for all input values.

③ That means Big-oh notation describes the Worst Case of an algorithm time Complexity.

④



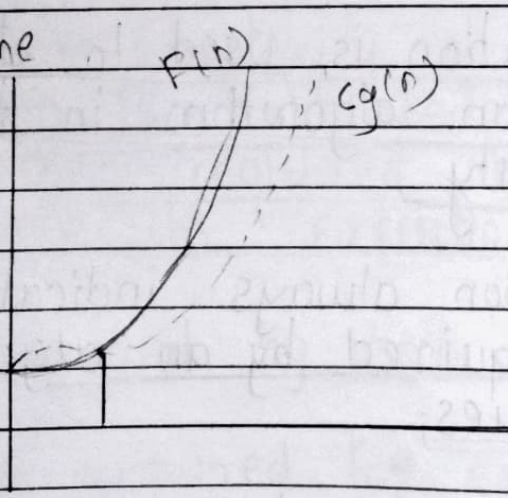
⑤ Big-Omega (Ω):-

① Big-omega notation is used to define the lower bound of an algorithm in terms of time Complexity.

② Big-omega always indicates the minimum time required by an algorithm for all input values.

③ That means Big-omega notation describes the best case of an algorithm time Complexity.

④



7. Introduction to data structure.

→

1. Data structure term describe how the way data is stored.
2. Data structure is a Specialized format for Organising, storing and retriving data.

* Needs of data structure.

1. data structure are used to study, how data are stored in Computer.
2. Data structure is Used to stored large amount of data.
3. Data structure helps us to Understand the relationship of one data element with other.
4. Conceptual and Concrete way to organize data is data structure.

* Advantages of data structure.

1. Allows easier Processing of data.
2. It is Secure way for storage of data.
3. it allows processing of data on Software System.
4. Graphs: model real life problems.
5. We Can access data any time anywhere
6. it design an efficient algorithm.

8. What is Data Object.

→

① A data object represents a Container for data Values, a place

② A data Object place where data Values may be stored and later retrieved.

③ A data Object is characterized by a set of attributes.

④ The data object "alphabets" Can be defined as D = { A, B, ... Z, a, b, ... z }.

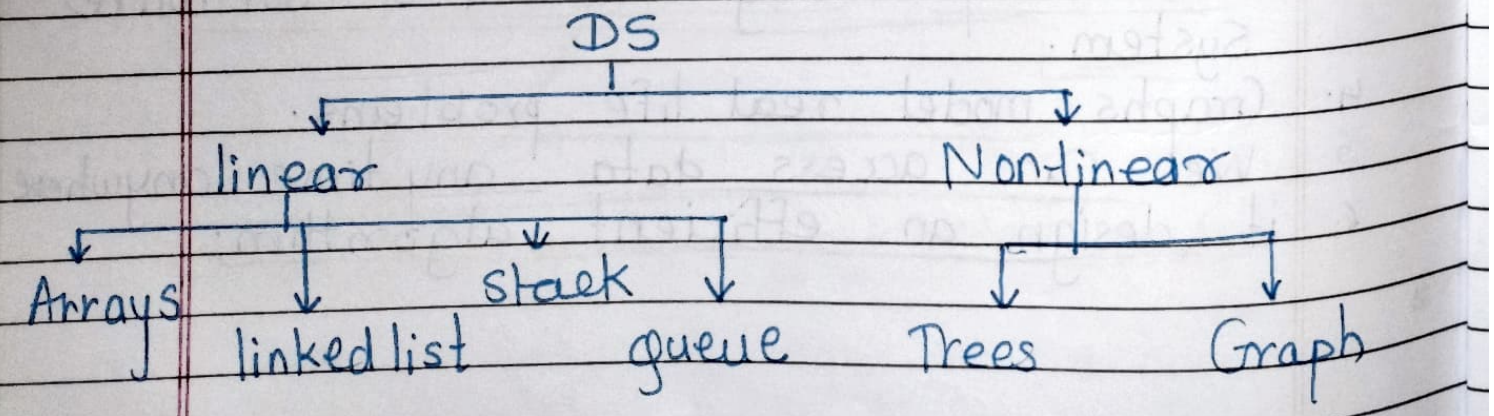
⑤ The data object "integer" Can be defined as D = { ..., -3, -2, -1, 0, 1, 2, 3 ... }

⑥ Data Object is a runtime instance of data Structure.

9. Explain data structures.

→

I] linear and Non-linear data structures:-



	linear data structure	Non-linear data structure
1.	In a linear data str. data elements are arranged in a linear order.	1. In a non-linear data str, data elements are attached in heirarchically manner
2.	In linear data structure, Single level is involved	2. In non-linear multiple levels are involved.
3.	These are easy to implement in the Computers memory.	3. These are difficult to implement in Computers memory
4.	Memory is not utilized in an efficient way	4. Memory is utilized in an efficient way.
5.	example. Array, linked list, Stack and queue	5. Example. tree Graph.

II] Primitive and Non-Primitive data structures.

→

- Primitive data structures defines a set of Primitive element which do not involve any other elements as its Subparts. example- int, char, float and double are primitive data structures.
- Non-primitive data structure are those which defines a set of derived elements Such as array example- array, structures.

III] static and dynamic Data structures.

→

- In static data structure the size of structure is fixed.
- In static data structure memory is allocated at compile time.
- example - Array

- In Dynamic data structure the size of structure is not fixed.
- In Dynamic data structure memory is allocated at run time.
- example - linked list, stack etc.

10. What is ADT

→

① ADT stands for abstract Data Types.
② An Abstract data type is a data declaration packaged together with operations that are meaningful for the data type.
③ ABstract data Type is a mathematical model with a Set of operations defined on that model.
④ ADT Data type includes.
1. Domain - Collection of data
2. Functions - A set of operations on data.
3. Axioms - A set of axioms are the rule of behavior.

Axioms

⑤ Advantages of ADT.

1. encapsulation
2. Information hiding
3. implementation
4. ADT is reusable.

11. What are array and structure; OR
→ diff^n bet^n Array and structure.

Array	Structure
1. An array is a collection of related data element of same type.	1. structure can have elements of different type of data
2. An array is derived data type	2. A structure is programmer defined data type.
3. Array allocates static memory	3. structure allocates dynamic memory
4. Array uses index for access elements of it.	4. structure uses (.) dot operator for access member of a structure.
5. Array works as a pointer to the first element of it.	5. But it is not in case of structure
6. The elements of the array are contiguous in memory Contiguous.	6. The elements of structure may not contiguous.

* Array

› An array is a linear data structure which is collection of data items having similar data type stored in memory.
› An array can be defined as an infinite collection of homogenous elements.
› Array form an important part of almost all programming language.
› Syntax -
data-type array-name [array-size];
› example -
int A[10];
data-type array-name [array-size].
› Type of Array.

I] One dimensional array - / Single dimensional :-
· Single dimensional or one dimensional array represents a linear collection of data;

· Syntax - data-type array-name si[array-size];
· example - int A[10];

II] Two dimensional array / multidimentional array :-
· Two dimensional are also called table or matrix.
· It has multiple dimensions & it has two subscript.
· Syntax - data-type array-name [rows][columns];

· example - int A[10][10];

$a_n x^n + \ldots\ldots a_2 x^2 + a_1 x + a_0$

$P(x) = 3x^2 + 3x - 16$

12. What is polynomial.

→

1. A polynomial is a mathematical expression involving sum of powers in one or more variables multipled by coeffieeients;
2. Polynomial is one variable with constant coefficients.
3. Representation of polynomial -

$$a_n x^n + \ldots\ldots + a_2 x^2 + a_1 x + a_0$$

4. example,

$$P(x) = 3x^2 + 3x - 16.$$

13. Explain self. referential structure?

→

1. Self. Referential structures are those structures that have one or more pointers which point to the same type of structure, as their member.
2. Structures pointing to the same type of structure are self referential structure.

```
struct node{
int data1;
char data2;
struct node * link;
};
```

3. Self refential structures are used to create linked lists, staeks. etc.

2. linear Data structure

Q. Introduction to Array

→

① An array is a finite ordered Collection of homogeneous data elements which.

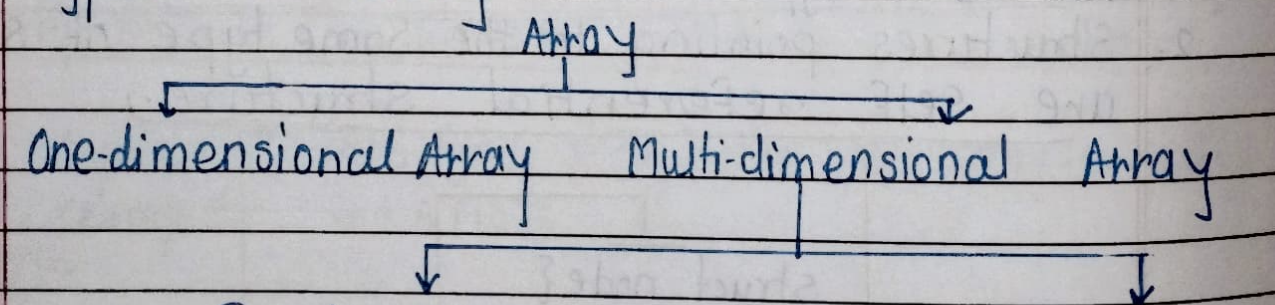
② Array initialization and representation :-

```
int a[5] = { 15, 25, 30, 32, 41}
float b[3] = { 5.1, 2.16, 3.3}
char c[3] = {'S', 'U', 'C'}
```

③ Representation :-

0	1	2	3	4	← index
15	25	30	32	41	← Value
100	102	104	106	108	← address

④ Types of Array



⑤ Application of Array :-

1. Array is Used to store list of values
2. Array is Used to implement search algorithm.
3. Array is Used to implement data structure
4. Array Used to Perform matrix operations
5. Array is Used to implemenet Sorting algorithm.

Q. Sorting. Explanation?

→

① Sorting is a technique to rearrange the elements in ascending or desending order.
② Order can be numerical or any user-defined order.
③ Sort order asending & desending.
④ Sorting algorithm are divided into two Categories.
   i. Internal Sort
   ii. External Sort.
⑤ i. Internal Sort - This Process use only the primary memory during Sorting process.

   ii. External Sort - This Process use external or Seeondary memory during Sorting process.

Q. Explain 5. Methods of Sorting.

1. Bubble Sort
2. Insertion Sort
3. Merge Sort
4. Quick Sort
5. Selection Sort.

Q. 1. Explain Bubble Sort.

1) Bubble sort arrange the N number of array element by placing the biggest element at last.

2) In Bubble Sort, Each element is Compared with its adjecant element if the first element is larger than the second one then the position of the elements are interchanged otherwise it is not changed.

3.) Bubble sort algorithm always arrange data in ascending order.

4) logic of bubble sort

```
if (A[j] > A[j+1]) then
    temp = A[j]
    A[j] = A[j+1]
    A[j+1] = temp
```

5) Example · 13, 11, 14, 15, 19, 9

Pass-1 13 11 14 15 19 9

11 13 14 15 19 9

11 13 14 15 19 9

11 13 14 15 19 9

11 13 14 15 19 9

11 13 14 15 9 19

Pass-2 :-

11 13 14 15 9 19

11 13 14 15 9 19

11 13 14 15 9 19

11 13 14 15 9 19

11 13 14 9 15 19

Pass 3 - 11 13 14 9 15 19

11 13 14 9 15 19

11 13 14 9 15 19

11 13 9 14 15 19

Pass 4 - 11 13 9 14 15 19

11 13 9 14 15 19

11 9 13 14 15 19

Pass 5 - 11 9 13 14 15 19

9 11 13 14 15 19 ← Sorted Array

6) Program - of bubble Sort

Explain Insertion Sort.

```
# include < stdio.h>
# include < conio.h>
Void bubblesort (int [], int)
Void main()
{
 int a[100], i, n;
 clrscr();
printf("\n Enter Array size = ");
scanf("%d", &n);

for(i=0; i<n; i++)
{
  printf("/n Enter Array Element = ");
  scanF("%d", &a[i]);
}
bubblesort (a, n);
getch();
}
Void bubblesort (int a[], int n)
{
    int i, j, temp=0;
printf("\n Bubble Sort");

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

```
        temp = a[i];
        a[j] = a[j+1];
    a[j+1] = temp;
      }
    }
  }
  Printf("In After Sorting Elements ");
  For( i=0;  i<n;  i++)
  {
     Printf("\t  %d", a[i]);
  }
  }
```

Q. Explain Insertion Sort. (without example)

→

① Insertion Sort the element is inserted at appropriate place.

② In this algorithm array divided into 2 parts Sorted and Unsorted Sub-array

③ In each pass 1st element of Unsorted Sub-array is picked up & placed at Sorted Sub array by inserting it.

④ Advantage and disadvantage.

1. Relatively Simple and Easy to implement
2. Inefficient for large array as time Complexity is $O(n^2)$;

⑤ { Example and Program is remaining }

Q. Quick Sort (without example)

→

1) Quick Sort is most popular and faster Sorting algorithm.
2) In this method, we 1st define 1 Pivot element
3) Then divided the remaining element in two parts.
4) ~~As~~ And Then Sort the element.

* Algorithm divide given elements in 3 parts.
i. Element less than the Pivot element.
ii. Pivot Element ((entral element)
iii. Element greater than the pivot element.

5) Advantage.
1. Faster Sorting method
2. It required small amount of memory
3. It good efficiency.

6) Disadvantage.
1. It is Complex
2. It is hard to implement than other.

Q. Merge Sort- [with example]

1. Merge Sort divide list in two equal part.
2. And repeat this Process Until when 1 element is got.

3. Then Sort the element and arrang in list.
4. Example-

38 27 43 3 9 82 10

38 27 43 3 9 82 10
38 27 43 3 9 82 10
38 27 43 3 9 82 10

27 38 3 43 9 82 10

3 27 38 43 9 10 82

3 9 10 27 38 43 82 82 ← sorted array

Q. Selection Sort - [with Example]

1. In this Sorting algorithm we 1st Find min element.
2. Then check that element with 1st location and then swap them.
3. Repeat this process Until we get Sorted element.
4. example- 5, -1, 2, 0, 3

5 -1 2 0 3

-1 5 2 0 3

-1 0 2 5 3

-1 0 2 5 3

-1 0 2 3 5 ← sorted array

Q. What is Searching ?

→

A Searching is a techniques are Used to find the element in the list.

Their are two types of Searching techniques
1) linear Search
2) Binary Search

linear Search :-

① linear Search also Called as Orderly Search or Sequential Search.
② In this Search algorithm key element Search from 1st element to last element
③ In linear Search algorithm works by Comparing every element with key element
④ This Search algorithm is practical when few element in the list.
⑤ Steps :-

Step 1 = i=0, take 1st element in list a[i]

step 2 = if (a[i] == key) then return i

step 3 := if i<n then
i = i+1
Go to Step 2

Step 4 = if i=n then
return -1

Step 5 = stop

③ advantages :-

1) it is Simple, easy to Understand.
2) it does not require data in array
3) when key element is matched with 1st element the linear search is best,

④ disadvantages -

1) if array Size is large the linear Search is not Sufficient,

> Binary Search :-

① Binary Search is faster than linear search
② This Search Can not applied on Unstored data Structure:,
③ Binary Search approach is divide & Conquer
④ Binary Search start Searching from middle element:,
⑤ Advantages:

1) It is Very time efficient Searching
2) Complexity is O(log n),

⑥ disadvantage

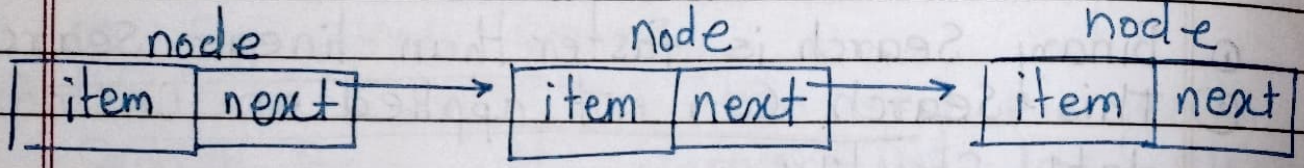
1) it required array are Sorted before applying this algorithm.

3. linked list.

Q. What is linked list?

→

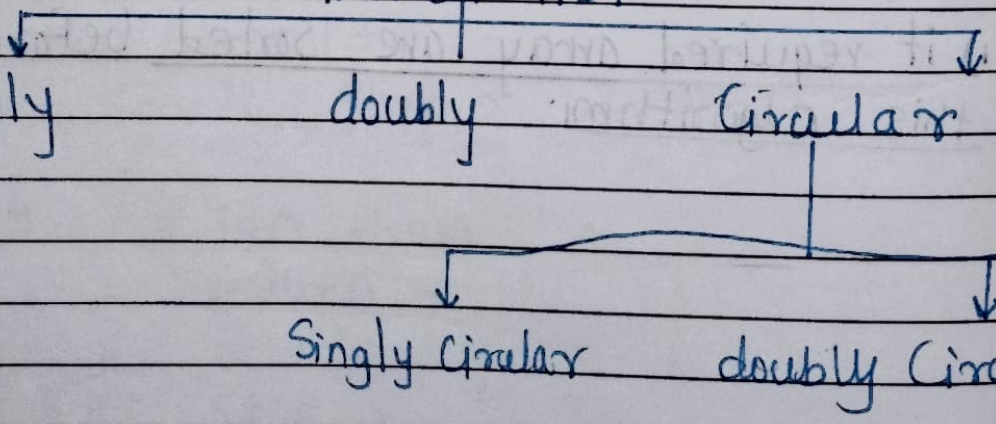
1. linked list is a linear data structure
2. It is one-way linear list.
3. linked list data element called as node
4. Each node Contain two part 1st part is data & 2nd part is link next address of element.
5. Structured of Linked list :-



6. characteristics-

1. Element Can be placed anywhere
2. Dynamic allocation
3. Each node is collection of data and address
4. it does not require data movement

7. Types of linked list-

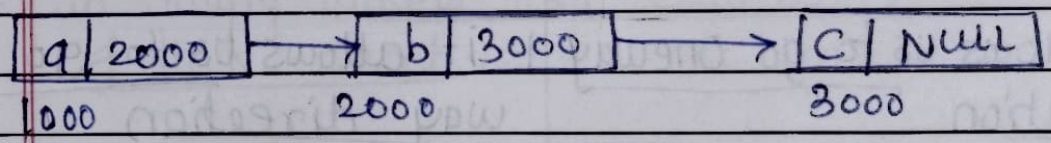


Singly linked list :-

1) In Singly linked list every node have 2 field. 1st is item and 2nd is store address of next node.
2) In Singly linked list we can tranverse only one direction.
3) Singly linked list is referred as ~~jst~~ just linked list.
4) Structur is :-

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

5) Representation of Singly linked list :-



doubly linked ~~ist~~ list :-

1) In doubly linked list every node have 3 field. 1st address of Previous node, 2nd item, 3rd address of next node.
2) In doubly linked list we can traverse two direction.
3) doubly linked list also called as two-way list.

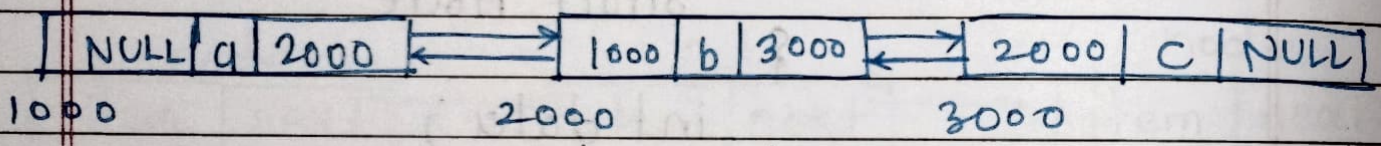
4) Structure of doubly linked list.

| Pre | Item | next |

- structure -

```
struct node
{
    struct node *pre;
    int data;
    struct node *next;
};
```

5) Representation -



Q. Diff^n Bet^n Singly and Doubly linked list

Singly linked list	Doubly linked list
1. It allows us to go oneway direction	1. it allows us to go two way direction
2. It uses less memory per node	2. it uses more memory per node
3. Complexity is O(n)	3. Complexity is O(1)
4. Contain : data & link next	4. Contain - Pre link, data & link next.
5. It is Unidirectional ie. only one direction	5. it is bidirectional

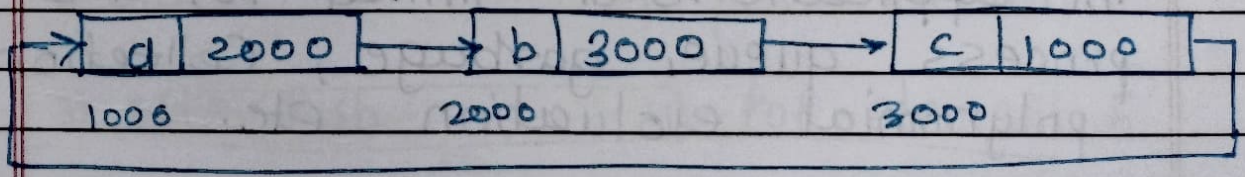
Circular linked list (CLL) :-

1. Circular linked list in which last node points to the head node.
2. Circuar linked list have niether beginning nor end.
3. From any node, it is possible to reach any other node.
4. Circular linked list could be either Singly or doubly linked list.
5. Types of Circular linked list :

I. Circular Singly linked list II. Circular doubly linked list

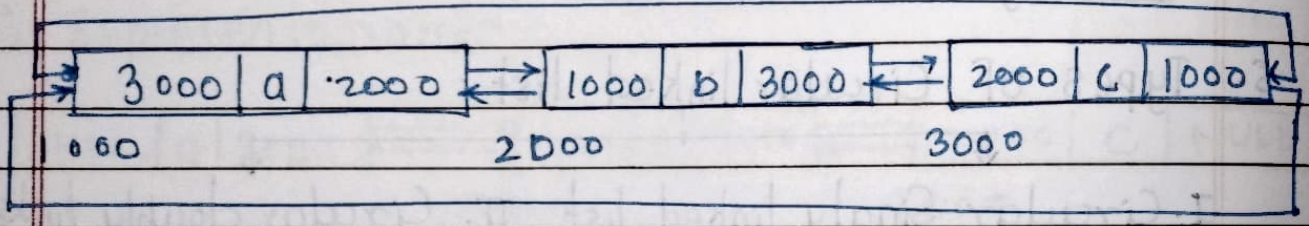
I. Circular Singly linked list :-

1. In Singly linked list Cantain 2 field data & link next.
2. The last node next is NULL is Singly
3. In Circular last node next is point 2nd node.



4. In Circular Singly linked list we Can access the first node.

II. Circular Doubly linked list :-

1. In doubly linked list Contain 3 field link Pre, data and link next.
2. The first node Pre and last node next is null in doubly
3. In Circular 1st node pre is point the last node and last node next is point 1st node.
4. 

Q. What are advantage of linked list over an array ?

1) Dynamic data structure
2) No Memory Wastage
3) Helpful for stack, queue, implementation
4) Insertion & deletion operation

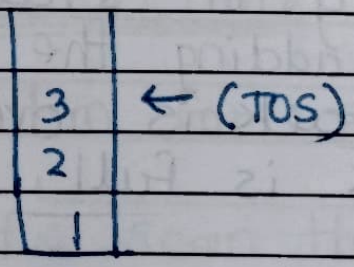
* The applications of linked list are process queue, garbage, Collection, polynomial evaluation etc.

4. Stacks

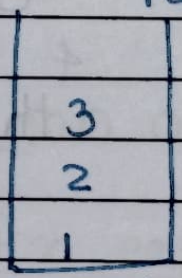
Q. stacks.

→

1) stacks is an ordered Collection of homogeneous data elements.
2) stack is most linear data Structure.
3) In stack insertion & deletion operations take place at only one end Called as top of the stack.
4) stack is set of elements in a (Last-In-Fisfst first-out technique (LIFO).
5) The inserting and or adding element into stack is called ~~push~~ push.
6) The deleting or removing element into stack is Called pop.
7) stack is also defined as Abstract data Type (ADT).
8) Ex -

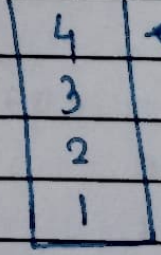


1) Stack

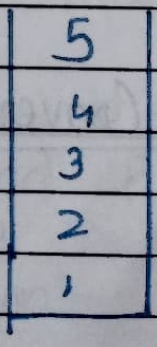


2) push element.

3) After Push



4) Pop element 5



5) After Pop

4	← TOS.
3	
2	
1	

Q Stack related term:-

1) stack - it is memory position, is used to store the elements.
2) Top - The top points the top element where element is add or delete.
3) Stack Under Flow - when no element in the stack & Perform delete then stack is Underflow or stack is empty
4) Stack OverFlow - when no. of element & Capacity of stack is equal & then adding the element then stack is overflow or stack is full.

Q. Applications of stack:-

→

stack data Structure is Used in many application.

1) Expression Conversion:-

> An expression can be represented in prefix, Postfix or infix notation.

> Stack is used to Convert one from of expression to another.

2) Expression Evaluation -

> A stack can be Used to evalute Prefix, Postfix & nitfix expression.

3) String Reversal -

> A stack is Used to reverse String.

4) Function call

> A stack is used to keep information about the active function.

Q. Algorithm for evaluation of Prefix notation.

→

* step1 : Scan the Prefix expression character by character.

step 2. if (ch = operand) then
push onto the stack.

Step3: if (ch = operator) then
a) Pop two operand from stack
b) Evaluate the expression
c) Push result onto the stack.

step 4: If no more element then pop the result else go to step 1.

Q. Algorithm For Evaluation To Postfix expression

→

step 1:- if (read character = operand) then push the element instead

step 2:- if (read character = operator) then

a) pop 2 operands from the stack.
y = pop() from the stack.
x = pop() from the stack.

b) evaluate them by applying operator i.e "x operator y".

c) push the result onto stack.

step 3:- if starting is ended, pop the result.

Q. Diff^n bet^n Stack and linked list.

→

stack	linked list.
1. Stack is abstract data type	A linked list is linear Collection of data element
2. Main Operation Poppush	main operation insert delete.
3. it is follows (LIFO) technique	3. it is elements Connect to each other.
4. Simplest than linked list	4. More Complex than Stack.

(Programm and functions are remainning)

5. Queue.

Q What / Explain queue?

→ Non-primitive data structure

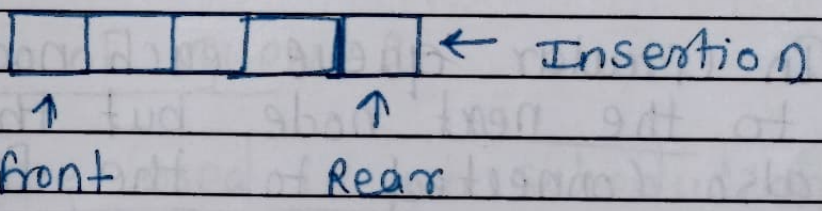
1) A queue is a linear list of elements.

2) In queue data can only be inserted at one end called rear and deleted from other end called as front.

3) The first element in queue will be removed first from queue.

4) A queue are called First-In-First-out (FIFO) technique.

5) A queue :-



Q. Operations on queue-

→

1) Create (Q) :- it creates an empty queue.

2) Insert (Q, x) :- it add element to the rear end of queue.

4) Delete(Q) :- it delete element from the front end of queue.

5) IsFull(Q) :- it return true if queue is full otherwise return false.

{ Remaining - static & dynamic implementation of queue) Program & function also.

5) IsEmpty(Q):- It return true if queue is empty other wise return False.

6) Peek(Q):- it is used to return value of first element without deleting it.

Q. Types of queue:-

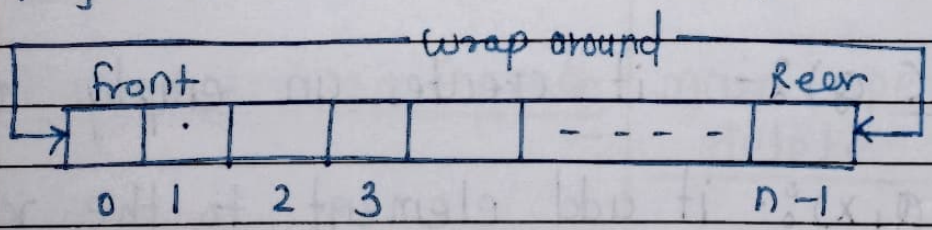
① Simple queue
② Priority queue
③ Circular queue.

* Circular Queue:-

1. A Circular queue is a wrap around from end to start is called Circular queue.

2. In Circular queue each node is Conneeted to the next node but the last node is also Conneeted to the first node.

3. Physical view of Circular queue.



4. Advantages:-
1. Circular view Consume less Spaee
2. It avoid shifting of queue elements.
3. Circular queue are Used in memory management, CPU Sheduling, traffic system etc.

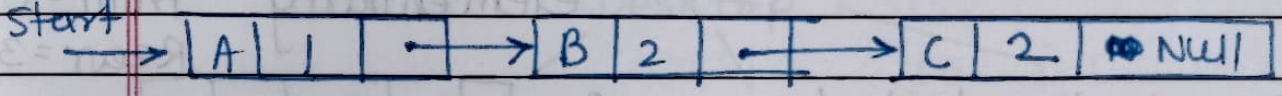
5) Enqueue, Dequeue, Rear, Front, isFull / isEmpty

are the Some Operation of Circular Queue.

* Priority Queue :-

1. A priority queue is the collection of items and items can be added at any time but the only one item can be removed is one with highest priority.

2. Representation of Priority queue.



3) IsEmpty, IsFull, Insert, Delete are the Some operation of priority queue.

Q. * Application of Queue :-

1. A queue are used in job scheduling by Operating System. in Example - printer.

2. Queue are used in Simulation Ex- Airport Simulation.

3. Handling the interrelu interrupts in real-time system.

4. To Process the Job in multiuser System.

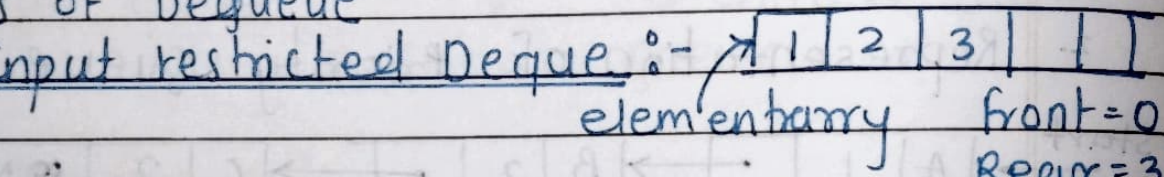
5. In batch Programming to Process the Job Sequentially.

* Define Dequeues.

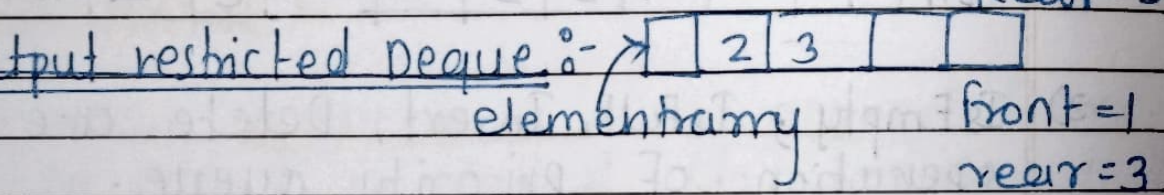
→

1. Dequeues stands for doubly ended queue.
2. A Dequeue is a linear list in which elements can be added or removed at either end but not in the middle. i.e Can be add or deleted at front or rear end.
3. Types of Dequeue

I. Input restricted Deque :-



II. output restricted Deque :-



4. insertFirst(), insertlast(), deleteFirst(), create(), are the Same Operation of Dequeues.

Q. Diff^n bet^n Stack and Queue.

→

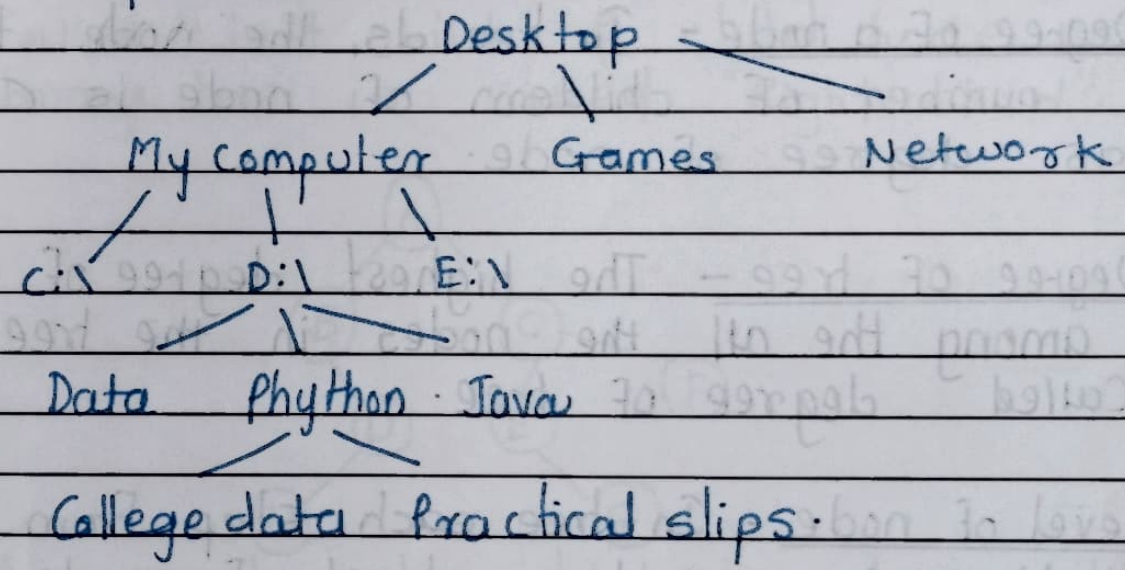
	Stack	Queue.
1.	Working principle is LIFO	1. working principle is F-I-FO.
2.	Stack does not have Variants	2. Queue has Vamiants like Cirular, Priority etc.
3.	In stack same end is used for insert and delete elements	3. In Queue two end are used for insert and is rear, & delete is front.
4.	Application of backt-racking.	4. Application of Serviceing requests.

6. Tree

Q.1. Explain/what is Tree?

→

1. Tree is a nonlinear data structure.
2. Tree is a hierarchy Consisting of a Collection of nodes and such that each node of tree stores a value.
3. A Tree is a finite set of one or more nodes.
4. Example of tree-



Q.2 What are the diff^n terminology of tree?

→

1. NULL Tree- A tree with no nodes is a null tree
2. Node- In tree every indiduval element is called node.
3. Root Node- First Node in tree is called rootnode.
4. Parent Node- In tree, the node which is predecessor of any node is called parent Node.

3. Binary Tree mainly Used for Searching and Sorting.

4. One of the most important application of binary tree is in the Searching algorithm.

· Operations of Binary tree.

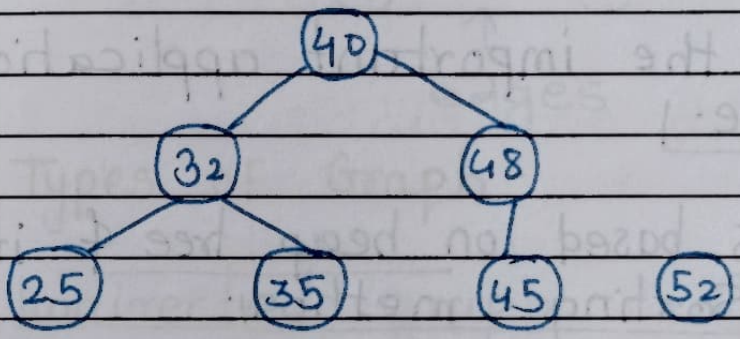
1. Create - Creating Binary Tree
2. Traversal - To Visit all the nodes in binary tree.
3. Insertion - Insertion Operation is Used to insert new node into binary tree
4. Deletion - Deletion operation is Used to delete any node into non empty binary tree.

Q. Binary Search Tree (BST)

1. A Binary Search Tree is a binary tree which is either empty or non-empty
2. If it is non-empty then every node Contains a key which is distinct and Satifies the 3 properties

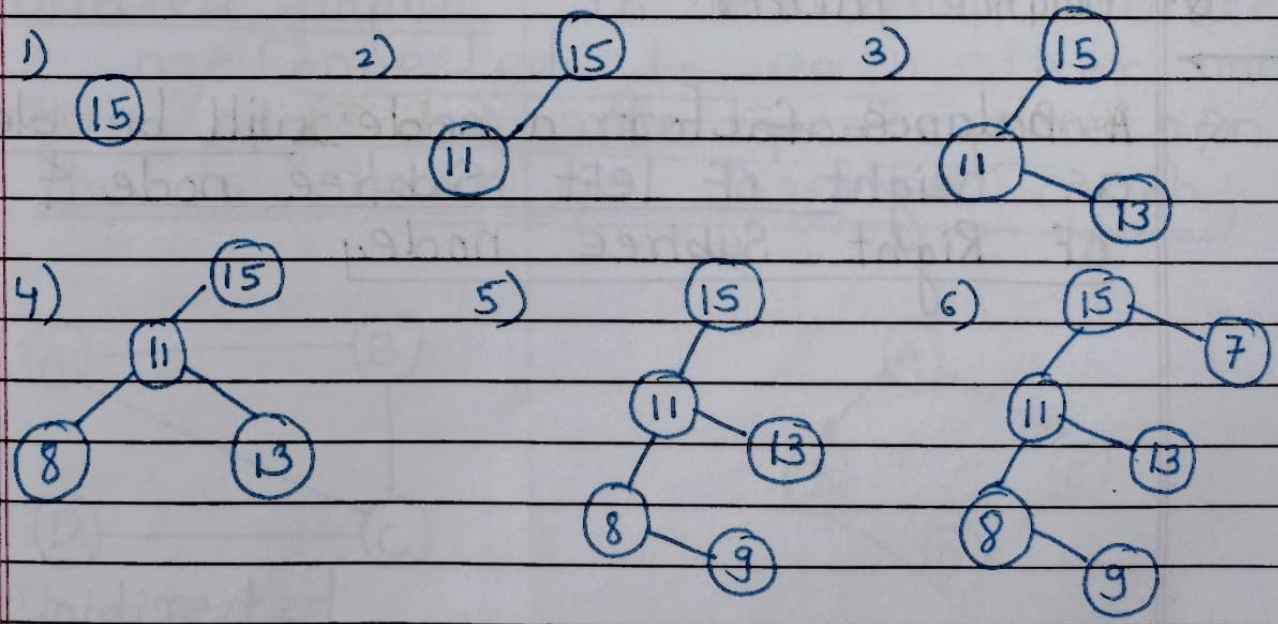
- Properties.

1. Value is less than its parent are placed at left Side of parent node.
2. Value is greater than its parent are placed at right side of parent node.
3. The left and right Substress are again binary Search tree.

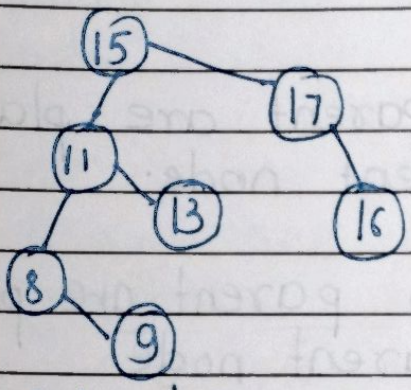


- Solve the example of binary Search tree.

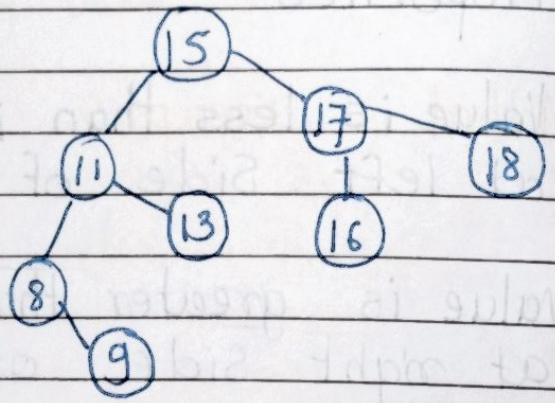
15, 11, 13, 8, 9, 17, 16, 18 key 15



7)



8)



Solved.

Q. Explain Heap Sort.

→

1. It is one of the important application of heap tree.
2. Heap Sort is based on heap tree & it is efficient Sorting method.
3. Using heap Sort we Sort accessding or decending orders.

Q. Balance Faetor.

→

A Balance factor a node will be defined as height of left Subtree node & height of Right Subtree node.

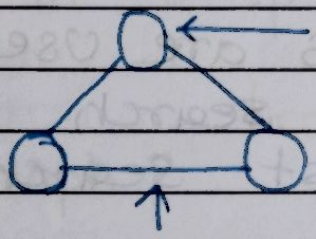
7 Graph

Q. What is Graph?

→

1. A Graph is a non-linear data structure that Consits the node or vertices and edges which Connects them.

2.

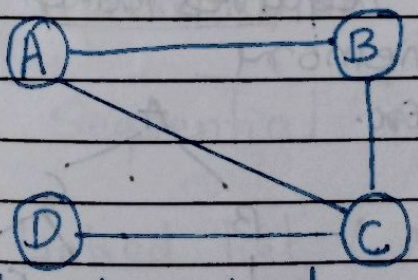


This Graph has 3 nodes.

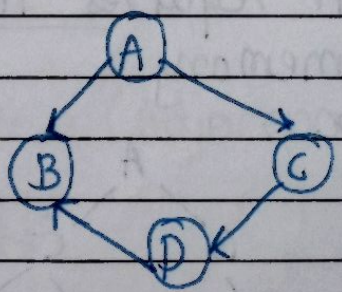
- Types of Graph -

1) Undirected graph - In this graph Vertices are Connected to each other and their edges have no direction this is known as Undirected graph.

2) Directed graph - In this graph Vertices are Connected to each other and their edges have one Specific direction this is known as Directed graph.



Unidirected Graph



Directed graph

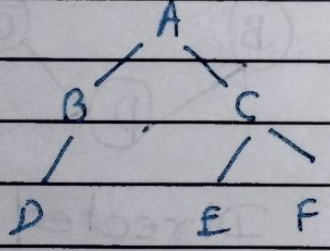
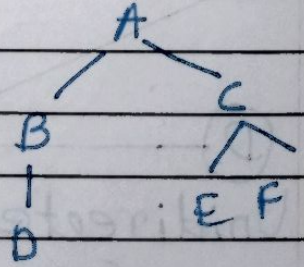
1. Representation of Graph -

1) Sequential Representation
2) linked Representation
3) Adjancency Multi-list Representation

Q. Graph traversal Method

Two techniques are used
1) Depth first search (DFS)
2) Breadth First search (BFS)

Q. Diff^n Bet^n DFS and BFS.

	DFS	BFS
1.	DFS stand For Depth First Search	1. BFS Stands for Breadth & First search
2.	DFS Visit node depth wise	2. BFS Visit node level by level.
3.	Uses stack data Structure	3. Uses Queue data structure.
4.	it requires less memory	4. it requires more memory
5.	Ex.  A, B, C, D, E, F	5. Ex.  A, B, C, D, E, F

Q. What is data structure.

→ DS is way to arrange data in main for efficient way
eg. arrays, linked links, stack queue.

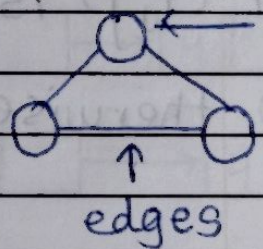
Q. What is algorithm

→ Sequence of steps to Solve a given problem

Q. What is Graph ?

→

1. A Graph is a non-linear data structure that Consits the ~~node or~~ vertices and edges which Connect them.

2. 

This Graph has 3 nodes or vertices

3. Graph is representation of relation. Vertices represent elements and edges represent relationships.

Q. List and explation various different Graph representation techniques.

→

1. Sequential Representation (Adjacency matrix)
2. Linked list (Adjacency lists)
3. Adajacency multi-list. representation

$A = [a_{ij}] = \{ 1 \text{ if } (i,j) \text{ is edge of } G \}$
$\{ 0 \text{ otherwise} \}$

I Sequential Representation

Adjacency Matrix :-

1. The graph when represented Using Sequential representation Using matrix known as Adjacency matrix.

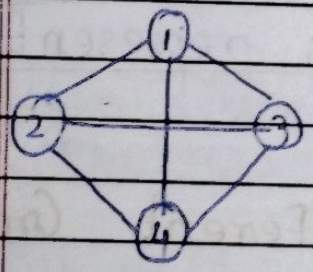
2. The adjacency matrix for an Undirected graph is Symmetric and for an directed graph need not to be Symmetric

3. $A(A_G)$ is a $n \times n$ zero-one matrix with 1 as its (i,j) th entry when i and j are adjacent and 0 otherwise

$$A = [a_{ij}] = \{ 1 \text{ if } (i,j) \text{ is edge of } G \}$$
$$\{ 0 \text{ therwise} \}$$

4. Example

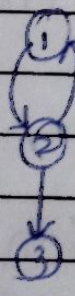
i.



adjacency matrix

	1	2	3	4
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	0

ii.



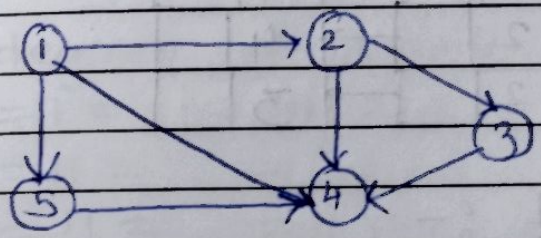
	1	2	3
1	0	1	0
2	1	0	1
3	0	0	0

II. Adjacency lists :-

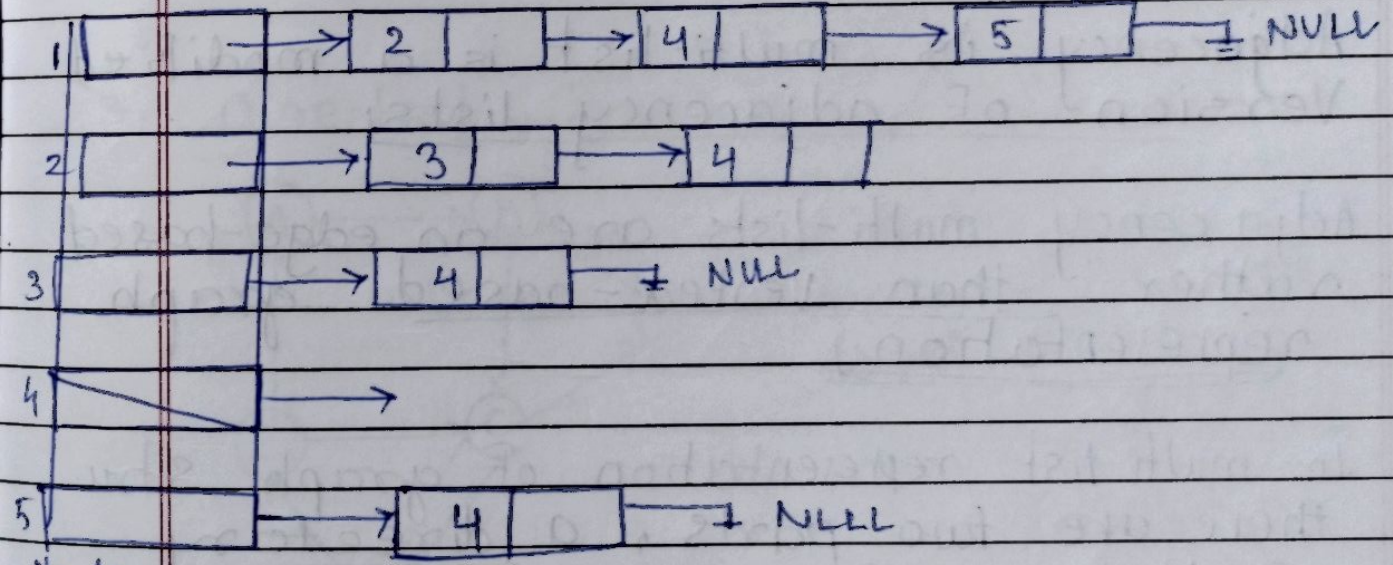
1. It is the way to represent a Graph w/o multiple edges it Specifices all the Vertices that are adjacent to each Vertex of a Graph.

2. Example.

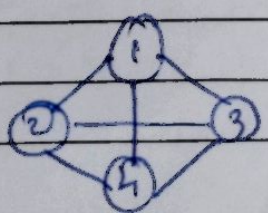
I. Adjacency list for directed graph.



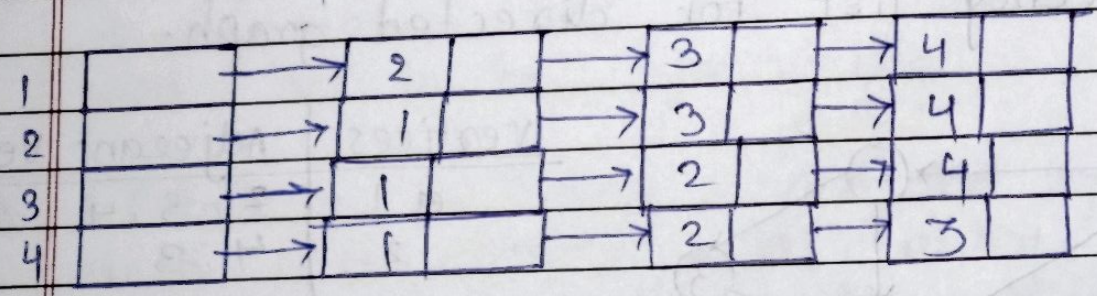
vertices	Adjeeant Vertices
1	2, 5, 4
2	4, 3
3	4
4	-
5	4



II. Adjaeenty list For Undirected Graph.



Vertices	Adjacent Vertices
1	2, 3, 4
2	1, 3, 4
3	1, 2, 4
4	1, 2, 3



III. Adajeency multi-list :-

1. Adjacency is multi-list is a modified version of adjacency lists.

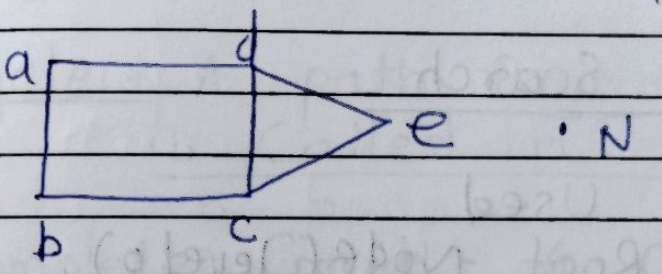
2. Adjacency multi-lists are an edge-based rather than vertex-based graph representation.

3. In multi-list representation of graph stru. there are two parts, a directory of Node's information and a set of linked list of edge information.

* Degree of a Vertex in Graph

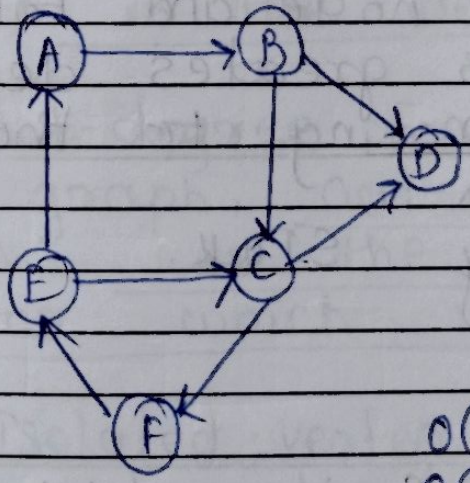
1. Def^n:- Number of edges incident on a vertex.

2. For Undirected Graph:-



degree of vertex a → 2
〃 〃 b → 2
〃 〃 d → 3
〃 〃 N → 0 called as isolated vertex

3. Directed Graph:-



~~I(D)~~ I(a) = 1
I(B) = 1
I(C) = 2
I(D) = 2
I(E) = 1
I(F) = 1

O(A) = 1
O(B) = 2
O(C) = 2
O(D) = 0
O(E) = 1
O(F) = 1

* Graph traversal

1. Graph traversal means visiting all the nodes of the Graph.

2. ⊖ BFS

* Breadth First Search

- Queue is Used
- Begins at Root Node (level 0), then visit all vertices at level 1, then all vertices of level 2 and so on

* Depth First Search

- Recursive algorithm
- Starts from root node and follows each path to its greates depth node before moving to the next path.
- Implement Using STAK.

* AOV Network :-

* A directed graph G in which the vertices represent tasks or activities and the edges represent precedence relations between activities is called Activity on Vertex Network or AOV Network.

* Important terminology or definations:-

1. Adjacent Vertex - When there is an edge from one vertex to another then these vertices are called as adjacent vertices.

2. Cycle - A path from a vertex to itself is called a cycle.

3. Complete Graph - A Graph G is said to be complete if every vertex in a graph is adjacent to every other vertex.

4. In-degree of a Vertex - In directed graph, in-degree of a vertex 'v' is the number of edges for which 'v' is the head.

5. Out-degree of a Vertex - In directed graph, out-degree of a vertex 'v' is the total number of edges for which 'v' is the tail.

6. Isolated vertex - If any vertex does not belong to any edge then is called isolated vertex.

7. Acyclic graph: A graph without cycle is called acyclic Graph.

8. Source vertex - A Vertex with in-degree zero is Called a Source Vertex.

9. Sink Vertex - A Vertex with out-degree zero is Called Sink Vertex.

10. Weighted graph - A weighted graph is a graph in which every edge is assigned a weight.

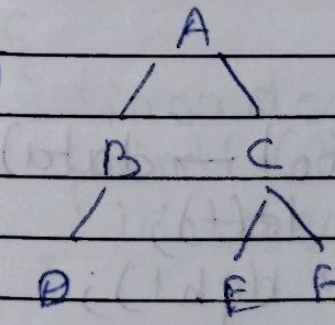
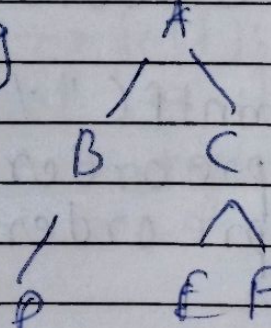
11. Pendant Vertex - When in-degree of vertex is one and out-degree is zero then Such Vertex is Called pendant Vertex.

12. Spanning Tree - A Spanning tree of a graph is a Subgraph of G having all Vertices of G and

12. Path - A path is a Sequence of distinct Vertices.

* Application of Graph :-

1. Used for Colouring of map
2. Topological Sorting of Graph
3. Spanning trees.
4. Critical path finding
5. To route from one location to another
6. Reliability analysis.

	DFS	BFS
1.	DFS Stands for Depth First Search	1. Breadth St First Search
2.	DFS Visit node depth wise	2. BFS visit node level by level
3.	DFS is faster Compare to BFS	3. BFS is Slower compare to DFS
4.	DFS requires less memory	4. BFS requires more memory
5.	Backtraking is allowed	5. Backtraking is not allowed
6	DFS is not optimal for finding shortest Path	6 BFS optimal for finding shortest path
7.	gets trapped into finite loop	7. can never get trapped into finite loop.
8.	Eg 	Eg 

Q. Explain Methods of traversing binary tree.

1. Preorder traversal :- / deft first

→ Process all nodes of tree by processing the root, then recursively processing all subtrees

→ Algorithm

Step 1 begin

Step 2 if tree not empty
visit the root
preorder (left child)
preorder (right child)

Step 3 end.

→ C program.

```
Void preorder (struct treenode * root)
{
   if(root)
   {
     printf("%d\t", root→data);
     preorder (root→left);
     preorder (root→right);
   }
}
```

2. Inorder Traversal / Symmetrical traver

→ Process all nodes of tree by recursively processing ~~out~~ the left Subtree, then processing root, and finally right Subtree

→ Algorith

Step 1 begin

step 2 if tree is not empty
inorder (left child)
visit the root
inorder (right child)

Step 3 end

→ program

```
void inorder ( tnode * root)
{
  if (root)
  {
    inorder (root → left);
    printf ("%d", root → data);
    inorder (root → right);
  }
}
```

3. Postorder - R

- Process all nodes of a tree by recursively processing all subtrees then finally processing the root.
- Algorith.

Step 1 begin

Step 2 if tree not empty
postorder (left child)
postorder (right)
visit the root

Step 3 end.

- Program

```
Void postorder (tnode *root)
{
  if (root)
  {
    postorder (root → left);
    postorder (root → right);
    printf ("%d", root → data);
  }
}
```