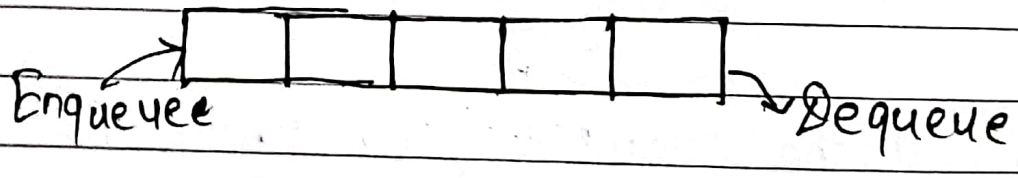
2076

4) Define queue. Write are different applications of queue? Explain queue operation with example.

→ A queue is an ordered collection of items from which items may be deleted at one end (called the front of the queue) and into which items may be inserted at the order end (the rear of the queue)

The stack as LIFO (last-in-first-out technique but the queue work as FIFO technique. (First-in-first out).



Application of queue:

→ Task waiting for the printing.
→ Time sharing system for use of cpu.
→ For access to disk storage
→ Task scheduling in operating system.

Operation on queue.

i) Make Empty (q): To make q as an empty queue.

ii) Enqueue (q, x) : To insert an item x at the rear of the queue.

iii) Dequeue(q): To delete an item from the first of the queue.

iv) Is full (q): To check whether the queue is full.

v) Is Empty (q): To check whether the queue q is empty.

vi) traverse (q): To read entire queue that is to display the context of the queue

Program to implement queue operations

```
#include <stdio.h>
#define Max_size 100
int arr_queue [max_size];
int rear = 0;
int front = 0;
int IsFull ()
{
return (rear == Max_SIZE);
}
int IsEmpty () {
return (front == rear);
}
```

```
void dequeue ()
    {
if (is empty' () )
    {
printf ("queue Ps empty :\n");
    }
    else {
printf (" Dequeued element =%.d \n", &
arr - queue [Front ]);
        front ++;
        }
        }
Void enqueue (int val)
        {
        if (is Full () ) {
printf ("queue is full . \n");
    } else
        {
arr - queue [rear] = val;
    rear ++;
    }}
Void traverse () {
if (isEmpty ()):
        {
```

```
printf ("queue is empty.\n");
    }
    else{
    int i;
    printf ("\n\t queue Data are:\n");
for(i= front; i <= rear; i++)
printf ("positions : %d , value : %d \n",i,
    arr-queue [i]);
        }
    }

int main () {
    int item, choice, i;
    int exit=1;
printf ("\tqueue main menu");
printf ("\n1. Insert \n2. Remove \n3.
    Display \n others to exist");
do
  {
printf (" enter your choice \n": );
scanf ("%d", & choice );
    switch (choice)
        {
    case 1:
printf ("enter element an item to insert:");
    scanf ("%d", & item);
    enqueue (item);
```

Date:
Page:

```
    break;
    case 2:
dequeue ();
    break;
case 3:
transverse ();
    break;
default:
    exit = 0;
    break;
    }
} while (exit);
    return 0;
    }
```

Date
Page

2. How do you find complexity of algorithms ? Explain.

→ Time complexity of an algorithm signifies the total time required by the program to run till its completion.

The time complexity of algorithms is most commonly expressed using the big O notation. It's an asymptotic notation to represent the time complexity.

Calculating Time complexity:

Now the most common metric for calculating time complexity is Big O notation. This removes all constant factors so that the running time can be estimated in relation to N, as N approaches infinity.

example:

```
int sum (int A[], int n){
    int sum = 0, i;
    for (i = 0; i < n; i++)
    sum = sum + A[i];
    return sum;
}
```

Date:

Page:

~~int~~ For the above code, time complexity can be calculated as follows.

int sumOfList (int A[], int n) {	Cost Time require for line unit	Repeated -tion	Total
int sum = 0, i;	1	1	1
for (i = 0; i < n; i++)	1+1+1	1+(n+1)+n	2n+1
sum = sum + A[i]	2	n	2n
return sum;	1	1	1
}			

The time complexity of above program is $4n + 4$.

Evaluate the expression ABCD-x+ using stack where A=5, B=4, C=3 & D=7

Given expression is

ABCD-x+ ; where A=5, B=4, C=3

D=7

final expression will be.

5437-x+

Scanned character	Stack	Evaluation
5	5	
4	5, 4	
3	5, 4, 3	
7	5, 4, 3, 7	
-		3-7=4
	5, 4, -4	
x		4x(-4) = -16
	5, -16	
+		5+(-16)=-11

so, the final answer will be -11

What is priority queue ? why do you need this type of queue ?

A priority queue is a collection of element such that each element has been assigned a priority & priority and the order in which elements are deleted and processed comes in following rules.

Date:
Page:

i) An element of higher priority is processed before any element of lower priority.
ii) if two elements has same priority then are processed according to the order in which they were added to the queue.

The best application of priority queue is observed in cpu scheduling.

There are two types of priority queue.
1) Ascending priority queue.
2. Descending priority queue.

Importance of priority queue-
→ priority queue are very important to system that juggle multiple programs and their execution (programs are chosen to run based on their priority) They are also very important to networking system. like the internet, because they can help prioritize important data to make sure it gets through faster

7) Write a recursive program to find nth fibonacci number.

Program:

```
#include <stdio.h>
int fibonacci (int);
int main ()
{
int numbers, i;
printf ("How many numbers?");
scanf ("%d", &number);
printf (" Fibonacci series : \n");
for (i= 0; i< numbers; i++)
{
printf ("%d \t", fibonacci (i));
}
return 0;
}
int fibonacci (int n)
{
if (n == 0)
return 0;
else if (n == 1)
return 1;
else
return (fibonacci (n-1) + fibonacci (n-2));
}
```

Date:

Page:

Output:

How many numbers ? : 10

Fibonacci Series :

0 1 1 2 3 5 8 13 21 34.

8) Explain array implementation of list

→ To implement a list, we need to define large to hold all the items of the list. The list item of the list is placed 0th position of array and successive items in successive positions of the array. The operation of insertion and deletion ~~ant~~ of the item can be done to / from anywhere within the last.

Implementation:

→ For array implementation, let first we make assumptions for some - user defined keywords that we are going to use further.

i) N be the maximum number of elements / items in an array.

ii) array [N] be an array to hold the items of lists.

→ last-index is ~~designed~~ denied to indicate index of the last element.
→ initially, last-index = -1.

Now, we perform array implementation for operations insertion and deletion as follows

1) insertion:
→ Read the position pos to insert.
→ if pos > last-index + 1, declare invalid and return.
→ let i = last-index.
→ while (i > pos)
    array [i+1] = array [i];
→ array [pos] = x;
→ increment last-index.

2. Deletion:
→ Read position pos to delete.
→ IF pos > last-index, declare invalid & return
→ x = array [pos]
→ i = pos;
→ while (i < last-index)
    array [i] = array [i+1];
    i = i+1;
→ Decrement last-index

Date:

Page:

9) Hand test selection sort with array of numbers. 4, 71, 32, 19, 61, 2, -5 in descer -ding order.

→ given data are.

4 71 32 19 61 2 -5

Now, For the First position in the corted array, the entire array is to be scanned sequentially.

At present, 4 is stored at the first position. After searching the entire array, it is Found that 71 is the largest value.

So, swap 4 with 71 After the First Pteration. 71 will appear at the first position in the sorted array.

71 4 32 19 61 2 -5

The same process is applied to the rest of the array elements.

71 4 32 19 61 2 -5

71 61 32 19 4 2 -5

Date:
Page:

10) WAP to implement sequential search algorithm

=> Program:

```
#include <stdio.h>
int main ()
{
int size, i, search, Found = 0;
printf (" Enter size of array :");
scanf ("%d", &size);
int data [size];
printf ("Enter %d numbers in Array :", size);
for (i=0; i<size; i++)
{
scanf ("%d", &data [i]);
}
printf (" Enter the number to be searched");
scanf ("%d", &search);
for (i=0; i<size; i++)
{
if (data [i] == search) {
printf ("%d Found in Array in index %d",
search, i);
Found = 1;
break;
}
```

```
if (!Found)
    {
printf("\n The number is not in Array.
                \n");
    }
return 0;
    }
```

Output:

```
Enter size of Array : 5
enter 5 numbers in Array : 10
  12
  13
  14
  15
Enter the number to be search : 12
  12 Found in Array in index 1.
```

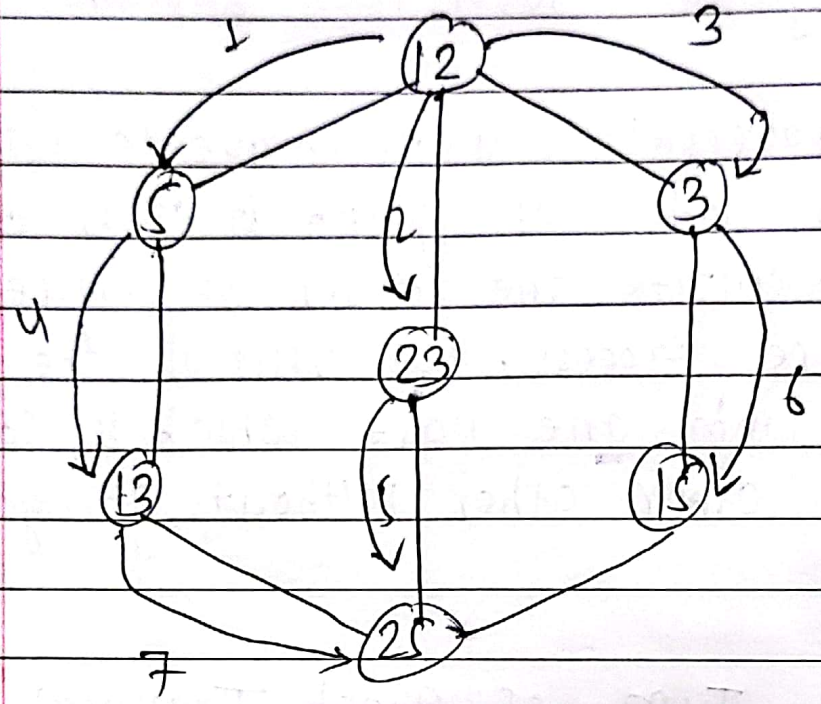
11) What is graph traversal ? explain.

→ Graph traversal is a technique to visit the each nodes of graph & It is also use to calculate the order of vertices in traverse process. we visit all the nodes starting From one node which is connected to each other other without going into loop.

~~Basically~~ Type of graph Traversal
1) Breadth First search.
ii) Depth First search.

1) Breadth First search:
Breadth - first search graph traversal techniques use a queue data structure as an auxiliary data structure to store nodes for further processing.
The size of the queue will be maximum total number of vertices of graph.

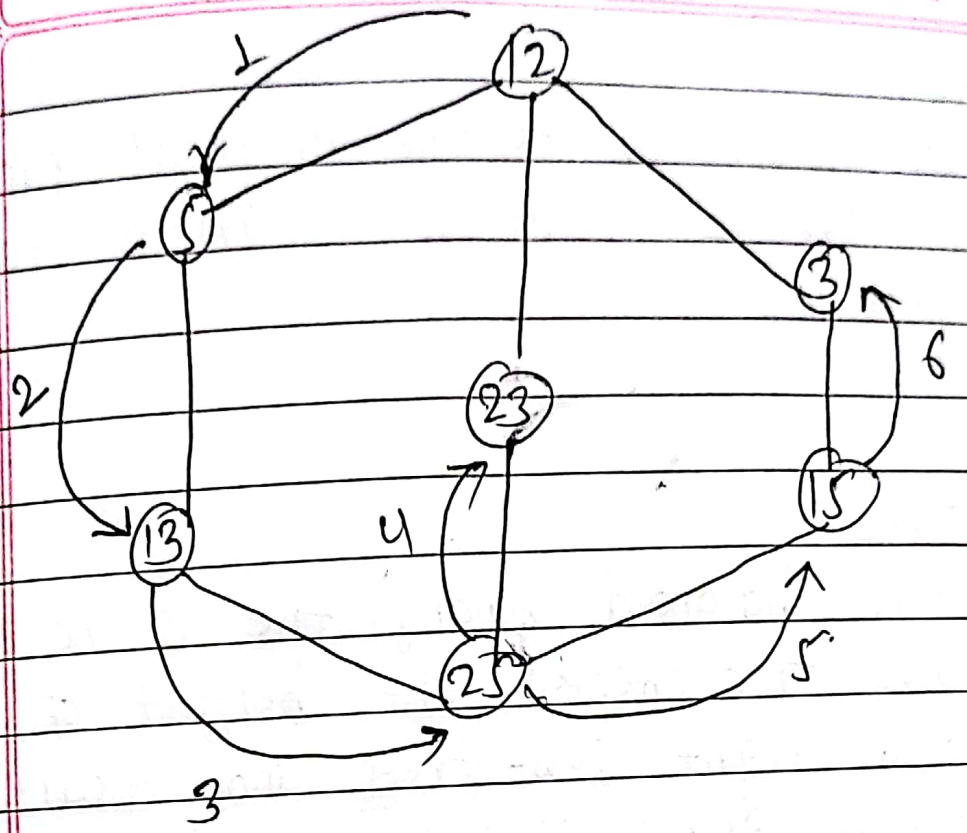
Date:
Page:



2. Depth first search:

→ DFS stands for depth first search, is one of the graph traversal algorithms that use stack data structure. In DFS Traversal go as deep as possible of the graph and then backtrack once reached a vertex that has all its adjacent vertices already visited.

Depth first search (DFS) algorithm traverse a graph in a depthward motion and uses a stack data structure to remember to get the next vertex to start a search when a dead end occurs in any iteration



12) Write short note on:
1) Divide and conquer sorting
2) AVL Tree.

a) Divide and conquer sorting:
→ Divide and conquer is an import proble solving technique the makes use of recursion. It is an effective recursive algorithm that consists of two part.

i) Divide:
→ in which smaller problem are solved recursively.

ii) conquer:
In which the solution to the original problem is then formed from the solutions to the sub-problems.

b) AVL Tree:
The First balanced binary tree is the AVL tree. AVL tree checks the height of the left and right sub-trees and assures that the difference is not more than 1. The difference is called the balanced Factor. An AVL tree is a binary search tree where the balance no. at each node is -1, 0, or 1. For an AVL tree of height H, we find that it must contain at least $F_{H+3} - 1$ nodes.