1] Write short note on time complexity & space complexity.

i] Time Complexity:-

- Time complexity is the computational complexity describing the amount of time required for the execution of an algorithm.
- Time complexity measures the time taken by every statement of the algorithm.
- Hence it highly depends on the size of processed data.
- It helps to define the effectiveness of an algorithm and to evaluate its performance.
- It takes as notation the mathematical symbols O, Ω, Θ.
- There are different types of time complexity depends on time spent by each algorithm till it reaches the end of its execution. therefore it depends on statements in a program.

ii] Space complexity:-

- When an algorithm is executed on a computer it necessarily requires a specific amount of memory space.
- Space complexity represents the amount of memory one program uses in order to achieve its execution.
- Because a program needs memory to store input data and temporal values while being executed.
- It also helps to evaluate a solution.
- It's represented using same notations as time complexity.

- Similar to time complexity, there are different types of space complexity, depends on the memory consumed by each algorithm.

2] Explain recursive functions with an example.

= i] A function which calls itself directly or indirectly again and again until some specified condition is satisfied is known as Recursive function.

ii] It repeatedly invokes the mechanism, the consequently increases the overhead of function calls.

iii] This repetition can be expensive in terms of both processor time and memory space.

example

program to find factorial of a given number recursively

```
#include<iostream.h>
#include<conio.h>
int fact(int n);
void main()
{
  int num, result;
  clrscr();
  cout<<"Enter number";
  cin>>num;
  result = fact(num);
  cout<<"factorial is "<< result;
  getch();
}
```

```
int fact(int n)
{
  int f=1;
  if(n<=0)
  {
    return(1);
  }
  else
  {
    f=n* fact(n-1);
    return(f);
  }
}
```

3] Distinguish between Merge sort and Quick sort.

Merge sort	Quick sort
1] The array is split into two sub-arrays (n/2) where 'n' is the number of elements in the array	1] The array is split into a specific number of parts in a specific ratio.
2] It is based on divide and conquer strategy	2] It is based on divide and conquer strategy.
3] Merge sort is considered as an external sorting algorithm	3] It is also known as partition exchange sort. consider as an interna Sorting algorithm. 4] It i

5]

Merge Sort	Quick sort.
4] It's coorst case complexity is O(n Log n), where 'n' is number of element	4] It's coorst case complexity is O(n squarred).
5] It coorks coith any size of the array, be it small or large.	5] It coorks coell coith large number of elements in an array.
6] It is considered to be a stable sorting algorithm.	6] It is not considered a stable sorting method
7] It requires ~~less~~ additional space / memory	7] It uses less additional space / memory.
8] It uses three arrays, where two of them store two halves of the array and the third array store the final sorted list	8] It uses a key element to sort elements.

4] Explain Recursive Binary Search algorithm with suitable examples.

=

1] Compare x coith the middle element.
2] If x matches coith the middle element, coe return the mid index
3] Else if x is greater than the mid element, then x can only lie in the right half subarray after the mid element.

so we recur for the right half.
4] Else (x is smaller) recur for the left half.

Algo

```
binarySearch(arr, x, loco, high)
    if loco > high
    return false;

    else
    mid:= (loco + high) /2 ;
        if x == arr[mid]
            return mid ;

    else if x > arr[mid]
        return binarySearch(arr, x, mid+1, high);

    else
    return binarySearch(arr, x, loco, mid-1);
```

5] Explain Merge sort and trace this algorithm for n=8 elements: 24, 12, 35, 23, 45, 34, 20, 48.

6] What is heap and write a algorithm of Heap Sort.

= Heap: In computer science, a heap is a specialized tree based data structure which is essentially an almost complete tree that satisfies the heap property

Algorithm

```
Insert (a,n)
{
    i:=n;
    item := a[n];
    while ((i>1) and (a[L i/2] < item) do
     {
       a[i]:=a[i/2];
     }
      a[i] := item;
        return true;
}
```

7] What is heap and write a algorithm of Heap sort using adjust and heapify

= A heap is a complete binary tree, and the binary tree is a tree in which the node can have utmost two children

9] Define union, find and write a algorithm of collapsing find algorithm

i] Union:- Join two subsets into a single subset. Here first check if the two subsets belong to same set. If no, then we cannot perform union.

ii] Find:- Determine which subset a particular element is in. This can be used for determining if two elements are in the same subset.

Algorithm:-

```
CollapsingFind(element, set)
{
    temp := element;
    while set[element] >= 0 do
    {
        element := set[element];
    }
    while element != temp do
    {
        j = set[element]
        set[element] := temp;
        element = j;
    }
return temp;
}
```

10] Write a algorithm of recursive MaxMin
=

Algorithm

```
maxmin( max1 , min2)
{
   if(i==j)
     max1 = min1 = num[i];
   else if (i == j-1)
    {
      if (num[i] < num[i])
       {
          max1 = num[i];
          min1 = num[i];
       }
      else
       {
          max1 = num[i];
          min1 = num[i];
       }
}
```