REG: 20202CA0829

NAME: SHISHU

NAME: SHISHU

Q2 Applying Divide and Conquer strategy, write a recursive algorithm for finding the maximum and minium elements from a list.

The divide-and-Conquer strategy divides a problem of a given size into one or more subproblems of the same type but smaller size. Then, supposing that the smaller size subproblems are solved recursively, the strategy is to try to obtain the solution to the original problem

Approach: To find the maximum and minimum element from a given array is an application for divide and conquer. In this problem, we will find the maximum and minimum elements in given array. In this problem, we are using a divide and conquer approch (DAC) which has three steps divide, conquer and combine.

For Maximum:- We are using the recursive approach to find maximum where we will see that only two elements are left and then we can easily using condition i.e. if(a[index] > a[index+1]) In a program line a[index] and a[index+1]) Condition will ensure only two elements in left.

NAME: SHISHU

NA-ME: SHISHU

SHISHU

```
    if (index >= l-2) {
        if (a[index] > a[index+1]) {
    // [a[index]
    // Now, we can say that the last element
    // will be maximum in a given array
    }
    else {
    // [a[index+1]
    // Now, we can say that last element
    will be maximum in a given array.
    }
}
```

In the above condition, we have checked the left side condition to find out the maximum. Now, we will see the right side condition to find the maximum.

Recursive function to check the right side at the current index of an array.

```
max = DAC-Max(a, index+1, l);
// Recursive call.
```

Now, we will compare the condition and check the right side at the current index of a given array.

```
// Right element will be maximum.
    if(a[index] > max)
        return a[index];
    // max will be maximum element
        in given array.
```


SHISHU

REG : 2020CA089

SHISHU

SHISHU

```
else
    return max;
}
```

For minimum :- In this problem, we are going to implement the recursive approach to find the minimum no. in a given array.

```
int DAC_Min(int a[], int index, int l) {
// Recursive call function to find the minimum no.
// in a given array.
  if(index >= l-2) {
// To check the condition that there will
//be two-element in the left then we
// can easly find the minimum element in a
// given array.
    // here we will check the condition
  if (a[index] < a[index+1])
    return a[index]);
  else
    return a[index+1];
}
```

Now, we will check the condition on the right side in given array.

```
// Recursive call for the right side in the
// given array.
    min = DAC_Min(a, index+1, l);
```

SHISHU

SHISHU

REG: 2020CA089

SHISHU

SHISHU

Now, we will check the condition to find the minimum on the right side.

// Right element will be minimum.

```
if (a[index] < min)
    return a[index]
else
    return min;
```

Q3 Show that the equation $33n^2 + 4n^2 = \Omega n^2$

given $33n^2 + 4n^2 = \Omega n^2$

$37n^2 = \Omega n^2$

Let $f(n) = 37n^2$

c = constant (Ω)

$g(n) = n^2$

$cg(n) \leq f(n)$

$c\,n^2 \leq 49\,n^2$

$c \leq 49$

or $c = 49$ proved

Q4 Given the complexity f(n) of various types, find the data size that can be solved in 1 sec and 1 min.

SHISHU

REG: 2020CA089

SHISHU

SHISHU

Q5 Which of the following is asymptotically smallest

(a) 2^n
(b) $n^{\lg n}$
(c) $n^{\sqrt{n}}$ ✓

Q6. State Master's Theorem and solve.

(a) $T(n) = 2T(n^{1/2}) + \lg n$

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$ & $b > 1$ possible sol^n upon 3 cases :-

Case -1 :-

If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$ then

$$T(n) = \Theta(n^{\log_b a})$$

Case -2 :-

If $f(n) = \Theta(n^{\log_b a})$ then

$$T(n) = \Theta(n^{\log_b a} \cdot \log n)$$

Case -3 :-

if $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and

If $af(\frac{n}{b}) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n then

$$T(n) = \Theta f(n)$$

REG: 2020CA089
NAME: SHISHU

Ques 8: Step 1 :- The main idea here is to walk intermittently right and left going each time exponentialy farther from the initial position.

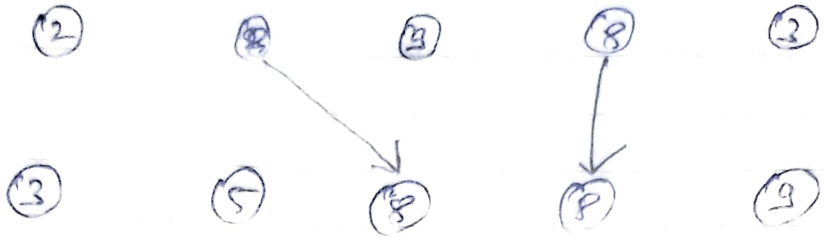
Step 2 :- A simple implementation of this idea is to do the following unil the door is reached :-
for i = 0, 1 .... enter 2i steps to the right. return to the initial position, make 2i steps to the left, and return to the initial position again.

Ques 11

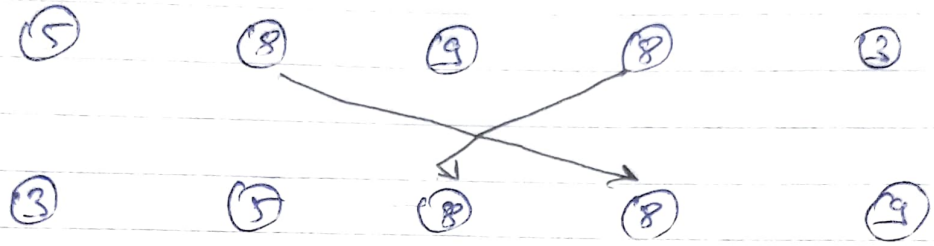
Stability in Sorting Algo :- The stability of sorting Algo is concerned with how the algo treats equals or repeated elements.

stable sorting algorithms preserve the native order of equal elements, while unstable sorting algorithms don't stable sorting maintains the position of two equals element relative one another.

REG: 2020CA089
NAME: SHISHU



Stable sorting Algo



unstable sorting.

According to implementation following will be stable sorting algo :-

- Insertion sort
- Bubble sort
- Radix sort
- merge sort

} Stable Sorting Algo

- Heap Sort
- Quick Sort

} unstable Sorting Algo.