ASSIGNMENT

BY- AMAN GUPTA
BRANCH- CSE
ROLL NO- 11911023

1) Insertion Sort function

```
void insSort ( int A[], int n)
{
    for (int i=1; i<n; i++)  → O(n)
    {
        j = i-1
        x = A[i]
        while ( j>-1 && A[j]>x)  → O(n)
        {   A[j+1] = A[j];
            j--;
        }
        A[j+1] = x
    }
}
```

→ Time Complexity (Average Case) :: $O(n^2)$

→ Analysing Time Complexity for best case →

Suppose the array is already sorted

A = [2 | 3 | 4 | 5 | 6]

In this array for every $j = i-1$,

the condition $A[j] > A[i]$ will be false

∴ The condition of while loop will never be true

For n elements, No. of comparisons = $O(n)$

No. of swaps (ie while loop) = $O(1)$

∴ Time complexity of Best case is $O(n)$

- The time complexity of insertion sort can be reduced by comparing & swaping elements at certain indexes or gaps first, although this technique is called shell sort but it uses insertion sort technique to sort elements.

These gaps can be taken by successive division of size of elements by 2, thus making time complexity as $O(n \log n)$ or taking gaps as prime no. less than size of array thus making time complexity of $O(n^{3/2})$

Eg

0	1	2	3	4
2	7	4	3	6

Gap is $\frac{5}{2}$ ie 2

∴ Insertion sort at $i=0$ & $i=2$

$i=1$ & $i=3$

$i=2$ & $i=4$ & $i=0$ & $i=2$

2	3	4	7	6

Now gap = $\frac{2}{2} = 1$

∴ Compare and sort at 1

When gap reduces to 1 we perform normal insertion sort.

The advantage is that in last case we don't need to perform many swapings as we already swapped elements at gap

2	3	4	6	7

2) • Bubble Sort Algorithm

In this sorting technique every element is compared with every other adjacent element if it is in wrong order then it is swaped.

Eg

4	2	3

Pass I → Comparing element 4 with adjacent element

4 > 2 → swap

4 > 3 → swap

2	3	4

Pass II

2 > 3 → NO

3 > 4 → NO

Page No.:
Date : / /

→ Time complexity of this algorithm is $O(n^2)$ as every element is compared to adjacent element for n times

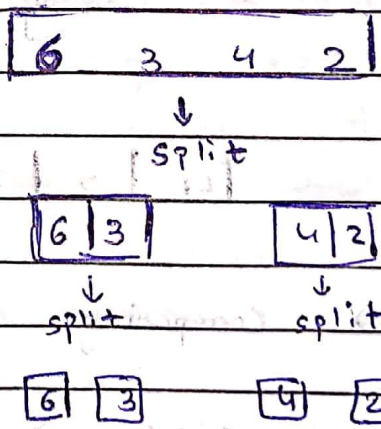
→ Space complexity → $O(n)$
This technique do not require any extra space other than the array to be sorted.

→ In place algorithm → Since no extra space is required to sort elements. ∴ It is in place algorithm.

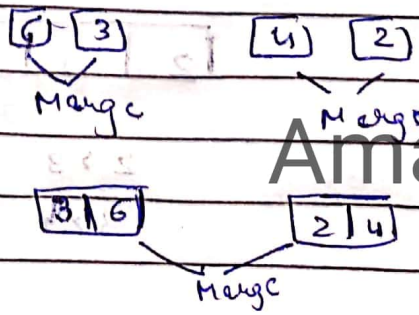
• Merge sort Algorithm

In merge sort we take two lists or arrays having sorted elements and merge them accordingly.

Eg



Now single element arrays are always sorted



2	3	4	6

Since at every step we divide the array is half and merge the n elements.

∴ Time complexity = $n \log n$

```
void msort ( int A[], int l, int h)
{
    if (l < h)
    {
        p = (l+h)/2
        msort (A, l, p);
        msort (A, p+1, h)
        merge (A, l, h);
    }
}
```

Space complexity → Since merge sort requires an extra array for sorting and the recursive msort also requires a stack having height of $\log n$

∴ Total space complexity = $(n + \log n)$

→ As extra space is required ∴ It is a out place algorithm.

Note → The algo. of insertion sort is mentioned in the 1st answer.

Quick Sort Algorithm

4	8	6	2	1

→ First we add ∞ at large value at end of array

4	8	6	2	1	∞

Now take reference element as (4) → P

& Take two pointers i & j

i = 0 & j = last element

If A[i] > P & A[j] < P then swap them

this process continue till i < j

P

(4)	8	6	2	1	∞

i j

→ Pass I →

4	1	2	6	8	∞

j i

Now swap A[j] & P element

2	1	4	6	8	∞

Now the array is sorted before & after P

so split the array

& quick sort these arrays

2	1	4

↓ q sort

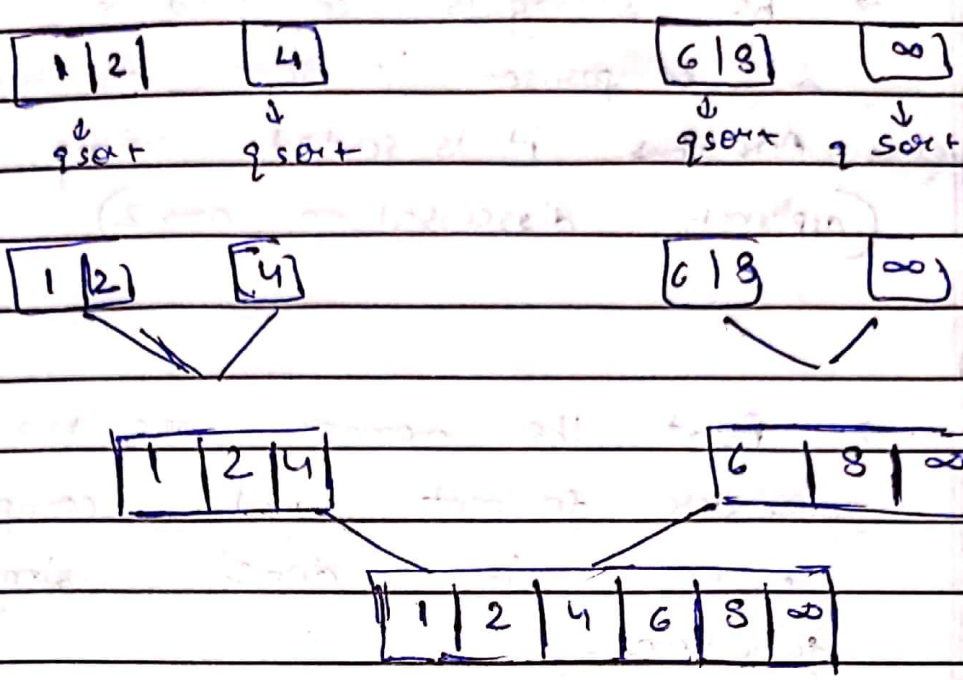
1	2	4

6	8	∞

↓ q sort

6	8	∞

split

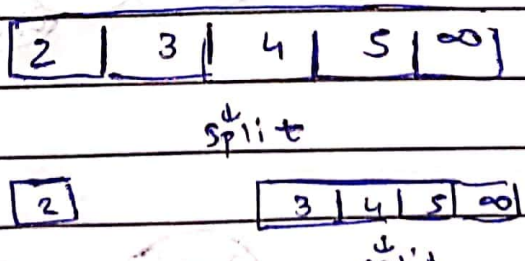


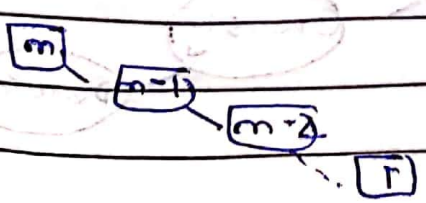
∴ Time complexity (average case)

Spliting the array in half & comparing n elements ∴ $(n \log n)$

→ Although the average time complexity is $n \log n$ but time complexity in worst case is $O(n^2)$

In worst case partitioning of array takes place at end of array





∴ Time complexity = $O(n^2)$

Page No.:
Date : / /

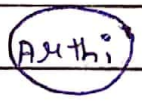
3 Approach

→ The string is converted to lowercase & comparison is done on basis of ASCII values And thus it is sorted using merge sort (approach disscussed in ans 2)

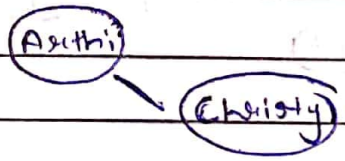
4)

→ First the names are taken in preorder format and compairson of names is done using strcmp() function.

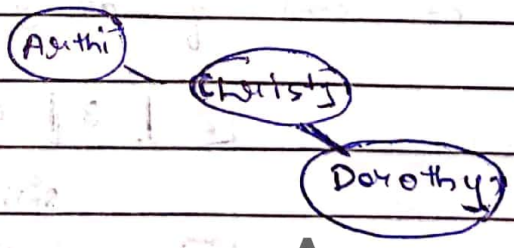
First element is made root then successive elements are made the left or right child on basis of strcmp <0 or >0



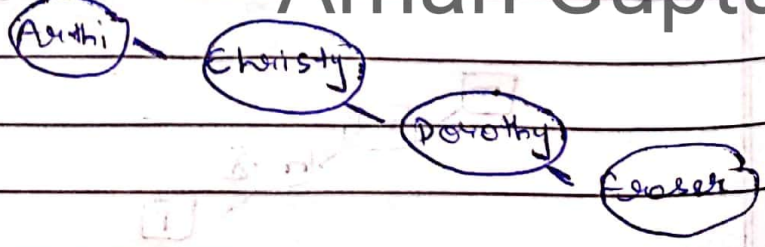
→



→

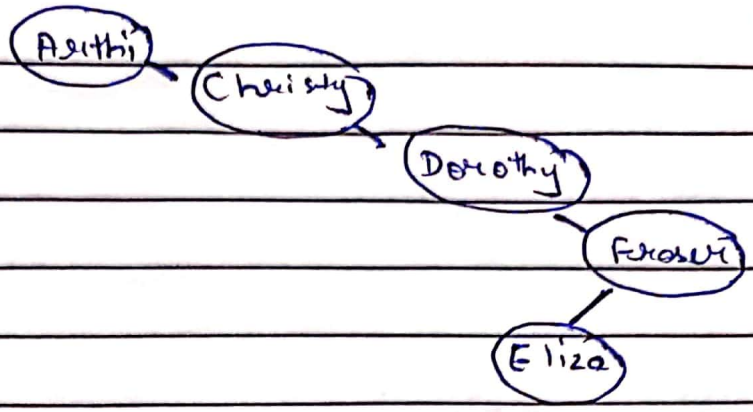


→



Date : / /

→



→ Now David is inserted