ASHOKA
Date:
Page No.

Assignment

By- Yash Gupta
Branch- CSE
Roll No. - 11911024

1. Insertion Sort function

```
void insert (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] = x
    }
}
```

Time complexity (average case) $\simeq O(n^2)$

Analysing time complexity for best case →

Suppose the array is already sorted.

A =

1	3	5	7	8

In this array for every $j = i-1$; the condition A[j] > A[i] will be false. Therefore, the condition of while loop will never be true.

For n elements, number of comparisons = $O(n)$

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

$\therefore$ Time complexity of best case is $O(n)$

The time complexity of insertion sort can be reduced by compairing & 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(nlogn) or taking gaps as prime number less than size of array thus making time complexity as $O(n^{3/2})$

Eg:

0	1	2	3	4
2	7	4	3	6

Gap is 2 ∴ Insertion sort at i=0 & i=2
i=1 & i=3
i=2 & i=4

2	3	4	7	6

Gap = 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

BUBBLE SORT Algorithm

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

Eg:

7	5	6

Pass I Compairing element 7 with adjacent element.
7 > 5 → swap
7 > 6 → swap

Pass II

5	6	7

5 > 6 → No
6 > 7 → No

ASHOKA
Date:
Page No.

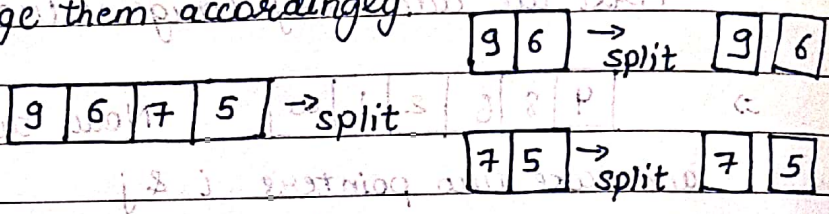
Time complexity of this algorithm is $O(n^2)$ as every element is compared to adjacent element for n times.

Space complexity : $O(n)$ This technique do not require any extra space other than the array to be sorted.
- Since, no extra space is required to sort elements, therefore, it is inplace algorithm.

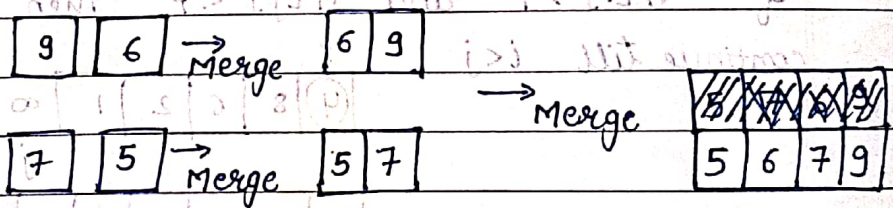
MERGE SORT Algorithm

In merge sort, we take two list or array having sorted elements and merge them accordingly.

Eg:



Now, single element array's are always sorted.



∵ at 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);
        }
}
```

ASHOKA
Date:
Page No.

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

∴ Total space complexity = $(2n + \log n)$

As, extra space is required, It is a outplace algorithm

Note: The algorithm of Insertion sort mentioned in 1st answer.

QUICK SORT Algorithm

4	8	6	2	1

First we add very large value at end of array.

⇒

4	8	6	2	1	∞

Now, take reference element as (4) and take two pointers i & j

i=0 and j = last element

If A[i] > P and A[j] < P then swap them. this process continue till i<j

P

(4)	8	6	2	1	∞

i j

⇒ Pass I →

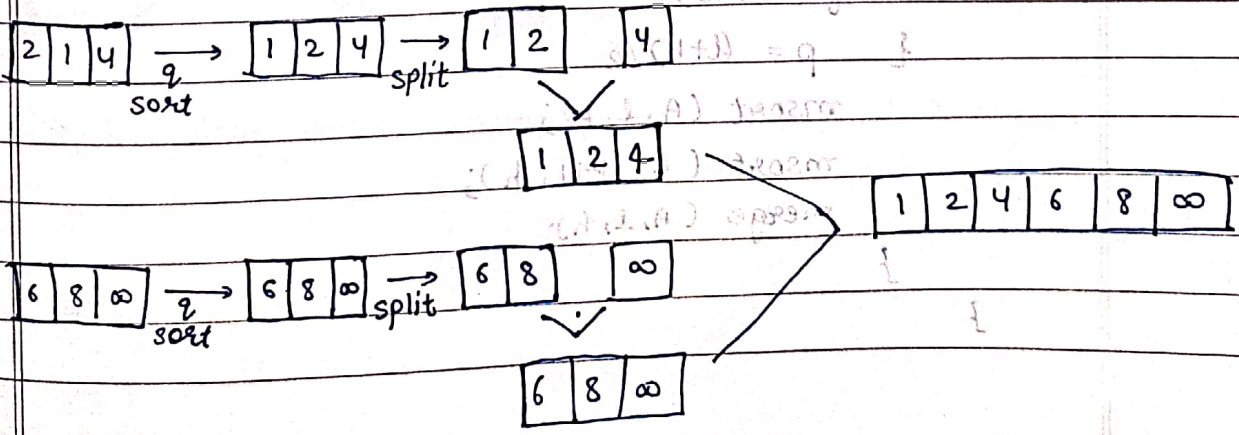
4	1	2	6	8	∞

j i

Now, swap A[j] & P element ⇒

2	1	4	6	8	∞

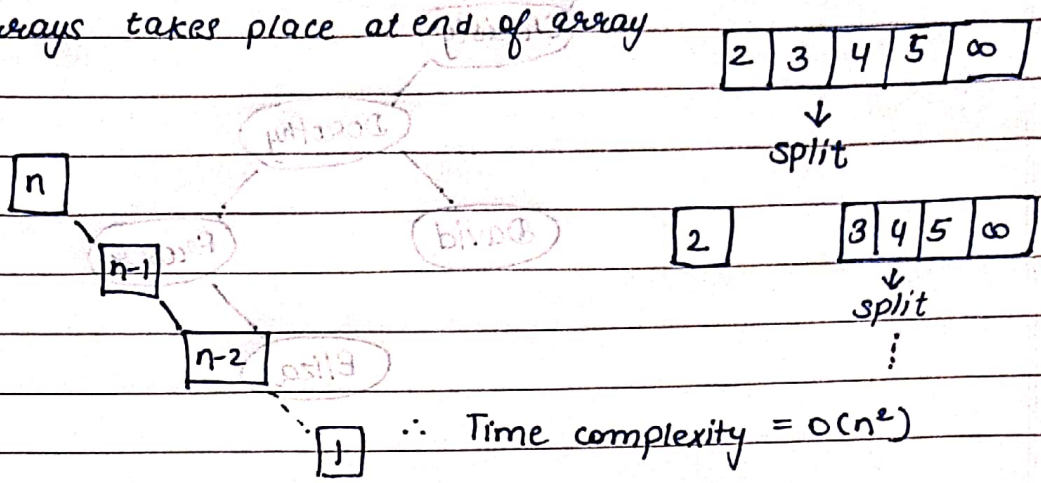
Now, the array is sorted before & after P so split the array and quick sort these arrays.



ASHOKA
Date:
Page No.

∴ Time complexity (average case) : splitting the array in half and comparing n elements = $O(n \log n)$

In worst case it is $O(n^2)$. In this case, positioning of arrays takes place at end of array



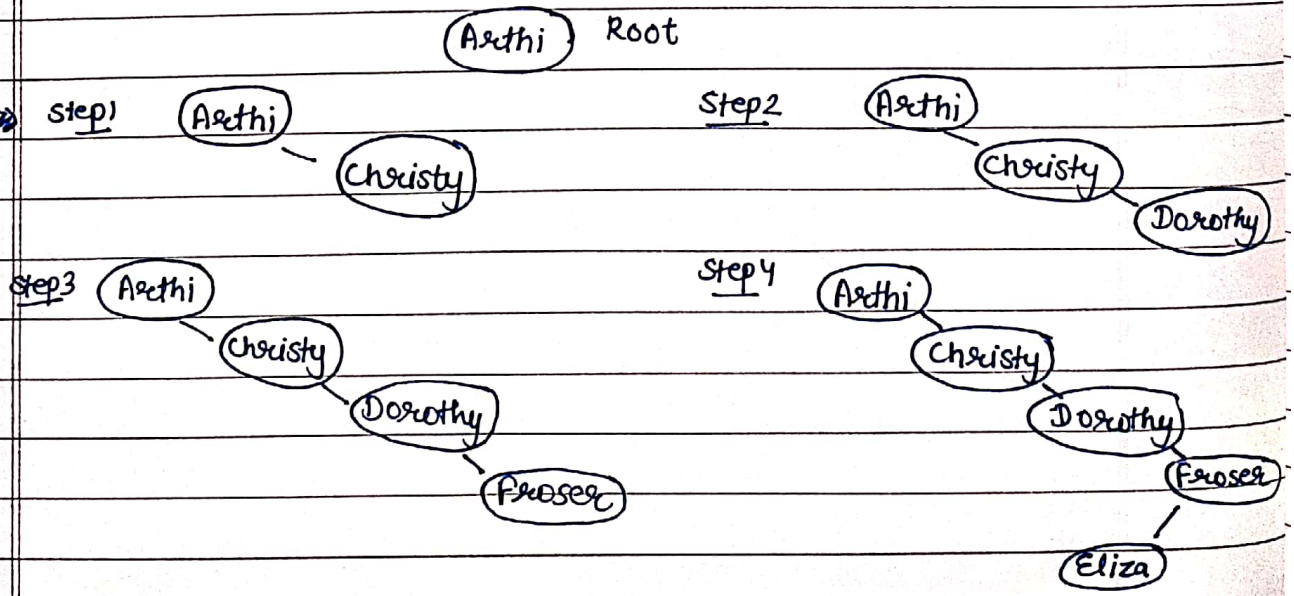
∴ Time complexity = $O(n^2)$

3. Approach

- The string is converted to lowercase and comparison is done on the basis of ASCII values And thus it is sorted using merge sort (Approach discussed in answer 2)

4. • First the names are taken in preorder format and comparison of names is done using strcmp(L) function.

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



ASHOKA

Date:

Page No.

Now, David is inserted.