Jawahar Education Societys Annasaheb Chudaman Patil College of Engineering, Kharghar, Navi Mumbai

NAME: PRIYUSH BHIMRAO KHOBRAGADE

PRN NO: 211112018

Roll No: 52

SUBJECT: Analysis of Algorithms Lab

Quick Sort using Divide and Conquer Approach

EXPERMINT: 03

PAGE NO.:
DATE.: / / 20

Experiment No :- 03

• Aim :- 'C program for Quick Sort using Divied and conquer Approach.

• Hardware / software :- 'TURBO C'

• Theory :-

Like merge sort, quick sort is a Drivide and conquer algorithm it picks an element an pivot & paritions the given array arrand the picked pivot. There are many different versions of quicksort that pick pivot in different ways.

1.) Always pick first element as pivot (implement below)
2.) Always pick last element as pivot.
3) pick a random element as pivot.
4) pick a median an pivot.

The key process in quicksort is partition(). Target of partitions is, given an array and element x of array as pivot, put x at its correct position in sorted array and put all smaller element (smaller than x) before x, and put all greater element (greater than x) after x. All this should be done in linear time.

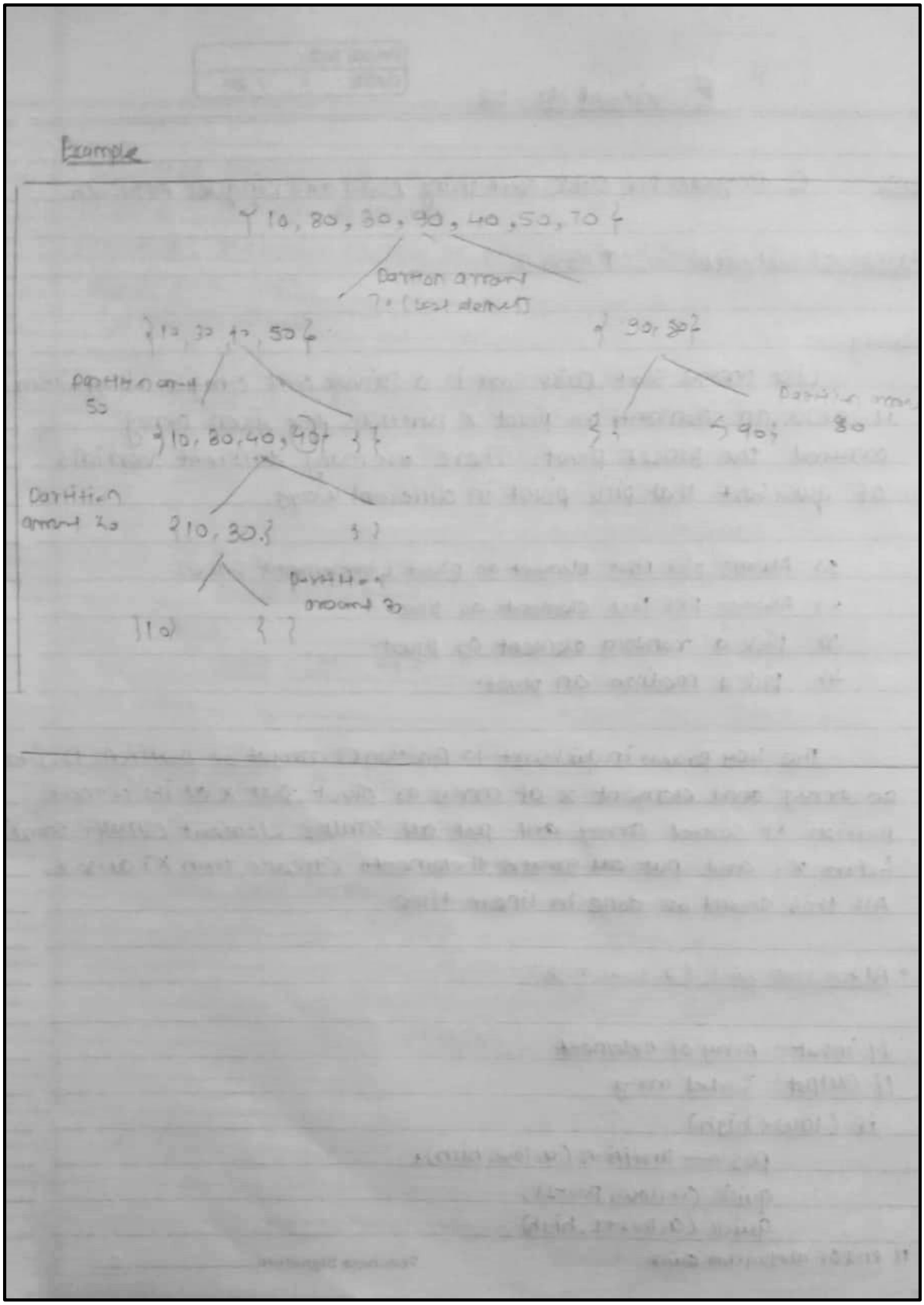
• Algorithm quick (a, low, high)

```
// input : Array of n element
// Output: Sorted array
if (low < high)
        pos ← partition (a, low, high);
        quick (a, low, pos-1);
        quick (a, pos+1, high);
// end of algorithm quick.
```

1

Teachers Signature _____

Quick Sort using Divide and Conquer Approach



PAGE NO.:
DATE.: / / 20

• Algorithm partition (a, low, high) :

// input : Array of element

// output : The exact position of pivot element is returned

```
pivot ← a [low];
i ← low ; j ← high;
while (i<j)
        while (a[i] <= pivot) && (i<j)
                i ← i+1
        while (a[j] > pivot)
                j ← j-1
        if (i<j)
            swap a[i] ↔ a[j]
        else
            swap a[low] ↔ a[j]
    return j
```

// end of algorithm partition.

• Analysis :-

Base case complexity :

$T(n) = 0 \qquad n = 1$

$T(n) = 0 \qquad '$

$T(n) = T(n/2) + T(n/2) + cn \qquad n > 1$

$= 2T(n/2) + cn$

Teachers Signature 2

PAGE NO.:
DATE.: / / 20

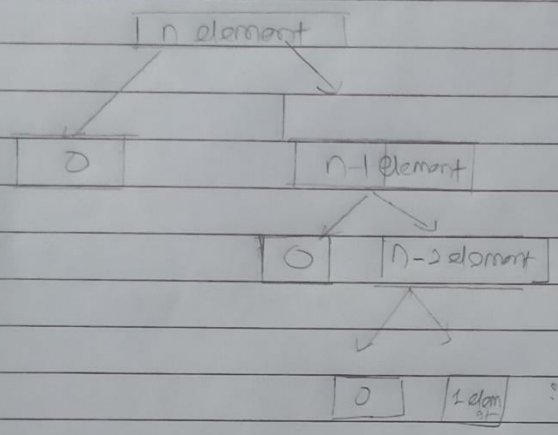
$$T(n) = 2T(n/2) + cn$$
$$= 2(2T(n/4) + cn/2) + cn$$
$$= 2^2 \cdot T(n/4) + cn + cn$$
$$= 2^2(2 \cdot T(n/8) + cn/4) + 2cn$$
$$= 2^3 \cdot T(n/8) + 3cn$$
$$= 2^i \cdot T(n/2^i) + icn$$

= We know $T(1) = 0$

$\therefore$ substitute $n = 2^i \therefore i = \log_2 n$

$$= n \cdot T(1) + \log_2 n \cdot cn$$
$$= cn\log_2 n = n = O(n\log_2 n)$$

- worst case complexity :-



$T(n) = 0 \quad n = 1$

$T(0) + T(n-1) + cn \quad n > 1$

$T(n-1) + cn$

$T(1) = c(T(0)) + c1 = c$

$T(2) = T(1) + cn = c + 2c$

$T(3) = T(2) + cn = c + 2c + 3c$

$$T(n) = c + 2c + 3c + \ldots + nc$$
$$= c(1+2+3+4+5 \ldots + n)$$
$$= c\sum_{i=1}^{n} i = c\left[\frac{n(n+1)}{2}\right]$$
$$= c\left[\frac{n^2+n}{2}\right] = \underline{O(n^2)}$$

- Conclusion :- Thus, it is observed that in Quick Sort is $O(n\log n)$ and worst case complexity is $O(n^2)$

Teachers Signature _____ 3

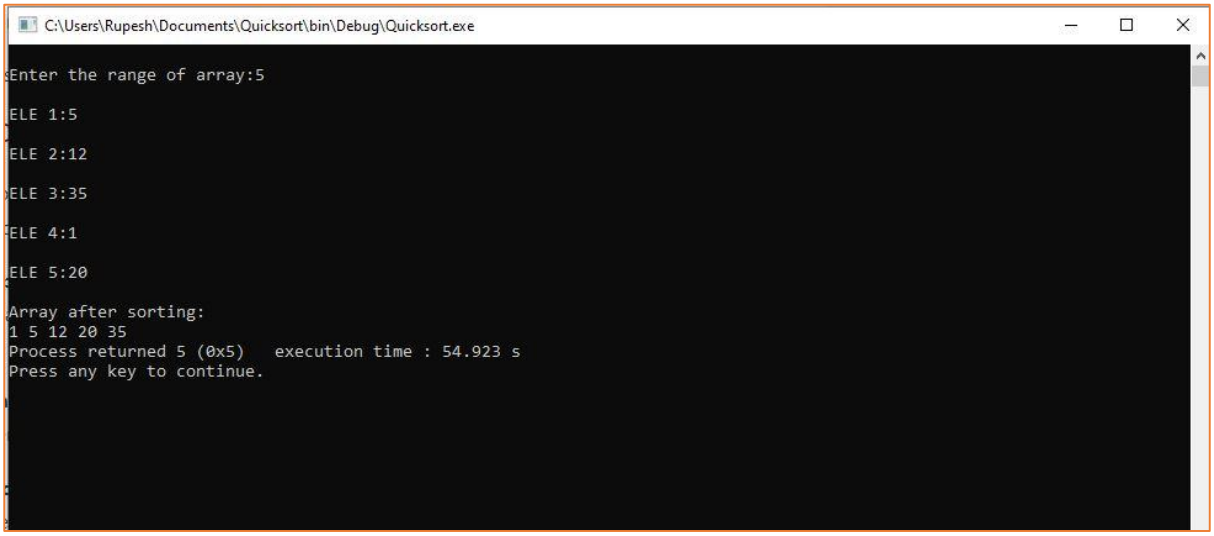
Quick Sort using Divide and Conquer Approach

Input:

```
#include<stdio.h>
void QS(int A[],int low,int high);
void main()
{
int A[100],n,i;
printf("\nEnter the range of array:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nELE %d:",i+1);
scanf("%d",&A[i]);
}
printf("\nArray after sorting:\n");
QS(A,0,n-1);
for(i=0;i<n;i++)
{
printf("%d ",A[i]);
}
}
void QS(int A[],int low,int high)
{
int i,j,p,temp;
if(low<high)
{
i=low;
p=low;
j=high;
while(i<j)
{
while(A[i]<=A[p]&&i<=high)
i++;
while(A[j]>A[p])
j--;

if(i<j)
{
temp=A[i];
A[i]=A[j];
A[j]=temp;
}
}
temp=A[j];
A[j]=A[p];
A[p]=temp;
QS(A,low,j-1);
QS(A,j+1,high);
}
}
```

Output:



Conclusion: Thus, it is observes that in Quick sort is Q (nlogn) and worst case complexity is $0n^2$.