Name: KAUSHAL VASHISTH Roll Number: 18ETCS002147

Laboratory 10

1. Questions

   Write a C program to implement sorting of numbers using bubble sort, selection sort and quick sort techniques. Calculate the time required for each approach.

2. Algorithm

Bubble Sort Algorithm:-

Following are the steps involved in bubble sort(for sorting a given array in ascending order):

1. Starting with the first element(index = 0), compare the current element with the next element of the array.
2. If the current element is greater than the next element of the array, swap them.
3. If the current element is less than the next element, move to the next element. Repeat Step 1.

Selection sort algorithm:-

1. Starting from the first element, we search the smallest element in the array, and replace it with the element in the first position.
2. We then move on to the second position, and look for smallest element present in the subarray, starting from index 1, till the last index.
3. We replace the element at the **second** position in the original array, or we can say at the first position in the subarray, with the second smallest element.
4. This is repeated, until the array is completely sorted.

Quick sort algorithm:-

Quicksort is a divide and conquer algorithm. The steps are:

1) Pick an element from the array, this element is called as pivot element.

2) Divide the unsorted array of elements in two arrays with values less than the pivot come in the first sub array, while all elements with values greater than the pivot come in the second sub-array (equal values can go either way). This step is called the partition operation.

3) Recursively repeat the step 2(until the sub-arrays are sorted) to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values. The same logic we have implemented in the following C program.

Time complexity calculation:-

Bubble Sort :-

In Bubble Sort, n-1 comparisons will be done in the 1st pass, n-2 in 2nd pass, n-3 in 3rd pass and so on. So the total number of comparisons will be,

(n-1) + (n-2) + (n-3) + ... + 3 + 2 + 1
Sum = n(n-1)/2
i.e O(n^2)

Hence the time complexity of Bubble Sort is O(n^2).

Selection sort :-

Selecting the lowest element requires scanning all n elements (this takes n - 1 comparisons) and then swapping it into the first position.

Finding the next lowest element requires scanning the remaining n - 1 elements and so on,

= (n - 1) + (n - 2) + ... + 2 + 1 = n(n - 1) / 2

= O(n^2) comparisons.

Best Case : O(n)^2

Worst Case : O(n)^2

Average Case : O(n)^2

Worst Case Space Complexity : O(1)

Quick sort :-

The worst-case time complexity of quicksort is Ω(n 2):-

Calculation:-

The partitioning step: at least, n − 1 comparisons.

• At each next step for n ≥ 1, the number of comparisons is one less, so that T(n) = T(n − 1) + (n − 1); T(1) = 0.

"Telescoping" T(n) − T(n − 1) = n − 1:

1. T(n)+T(n − 1)+T(n − 2)+. . .+T(3)+T(2) −T(n − 1)−T(n − 2)−. . .−T(3)−T(2)− T(1)

   = (n − 1) + (n − 2) +. . .+ 2 + 1 – 0

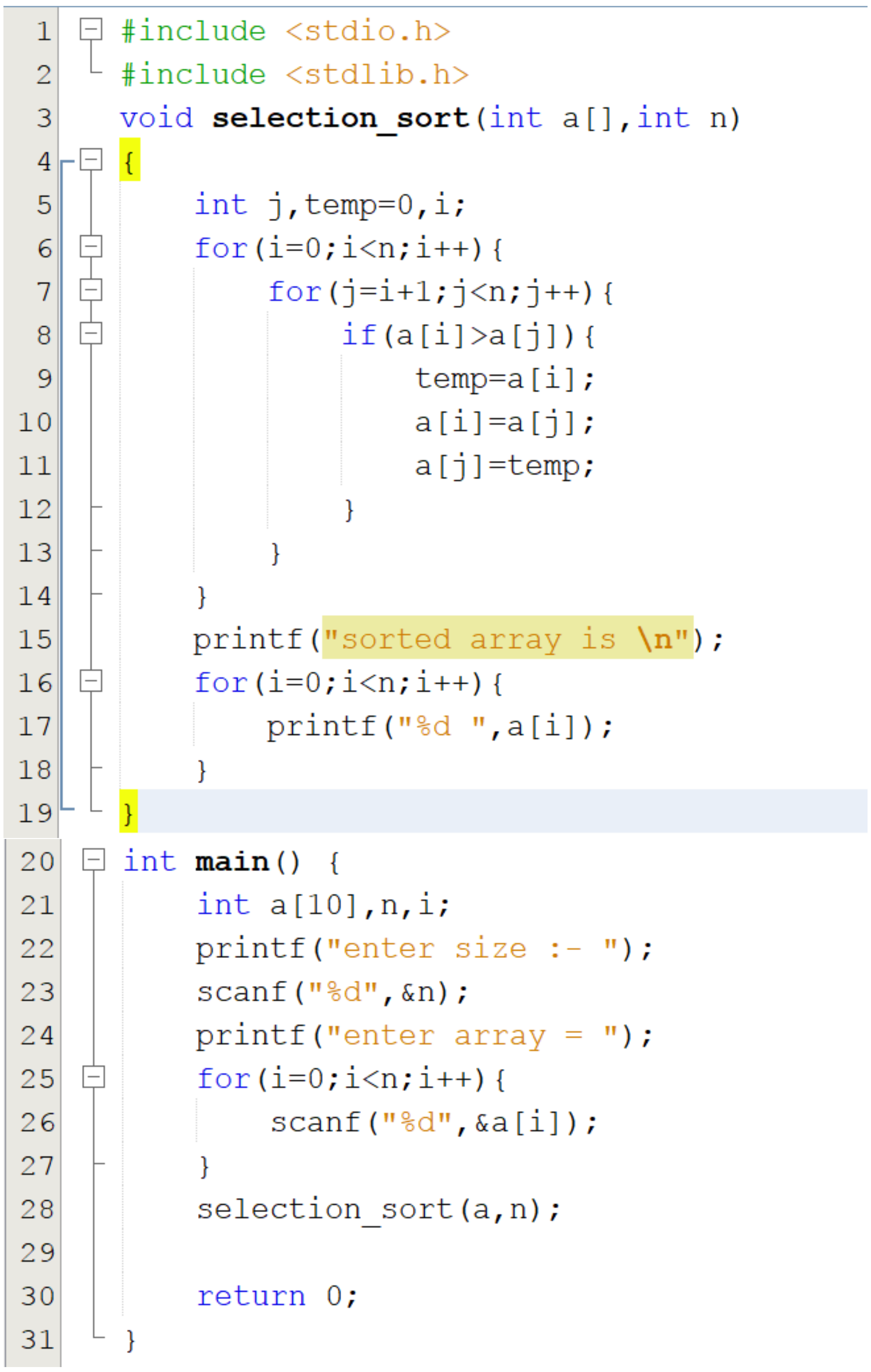
2. T(n)= (n − 1) + (n − 2) +. . .+ 2 + 1 =(n−1)n

This yields that T(n) ∈ Ω(n 2).

Where as, The Average Case T(n) ∈ Θ(n log n).

3. Program

Selection sort:-

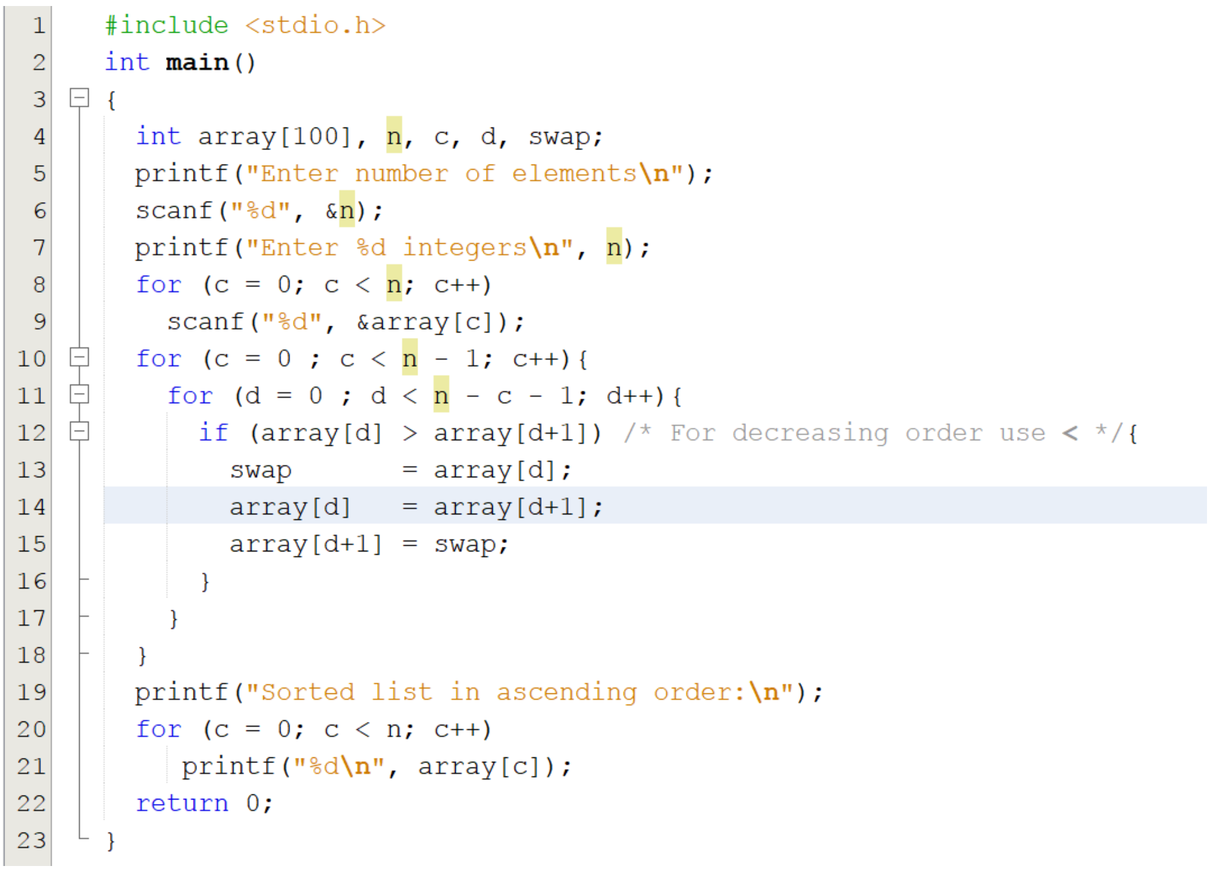


```c
#include <stdio.h>
#include <stdlib.h>
void selection_sort(int a[],int n)
{
    int j,temp=0,i;
    for(i=0;i<n;i++){
        for(j=i+1;j<n;j++){
            if(a[i]>a[j]){
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    }
    printf("sorted array is \n");
    for(i=0;i<n;i++){
        printf("%d ",a[i]);
    }
}
int main() {
    int a[10],n,i;
    printf("enter size :- ");
    scanf("%d",&n);
    printf("enter array = ");
    for(i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    selection_sort(a,n);

    return 0;
}
```

bubble sort:-

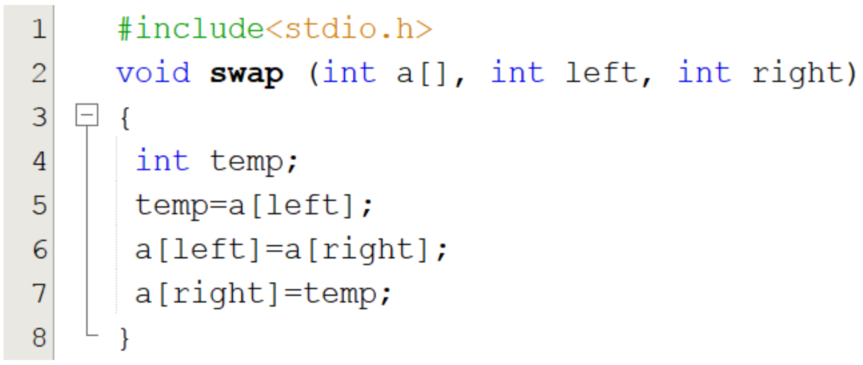
Name: KAUSHAL VASHISTH   Roll Number: 18ETCS002147



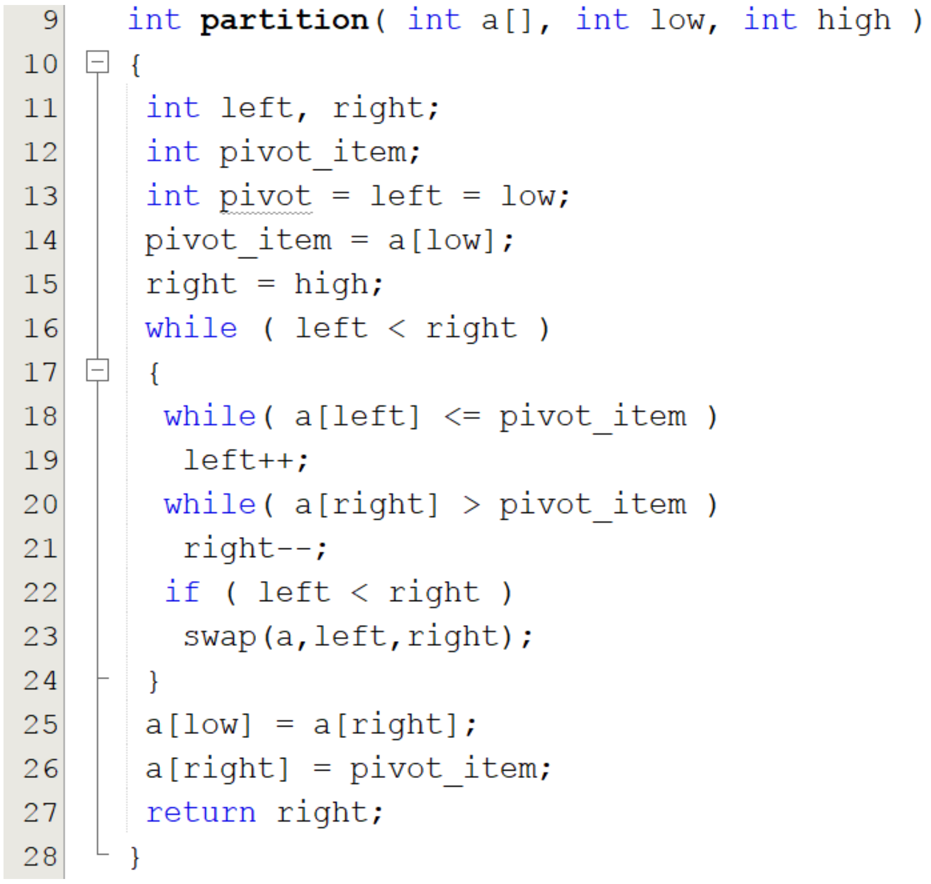
```c
#include <stdio.h>
int main()
{
  int array[100], n, c, d, swap;
  printf("Enter number of elements\n");
  scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);
  for (c = 0 ; c < n - 1; c++){
    for (d = 0 ; d < n - c - 1; d++){
      if (array[d] > array[d+1]) /* For decreasing order use < */{
        swap       = array[d];
        array[d]   = array[d+1];
        array[d+1] = swap;
      }
    }
  }
  printf("Sorted list in ascending order:\n");
  for (c = 0; c < n; c++)
     printf("%d\n", array[c]);
  return 0;
}
```

Name: KAUSHAL VASHISTH Roll Number: 18ETCS002147

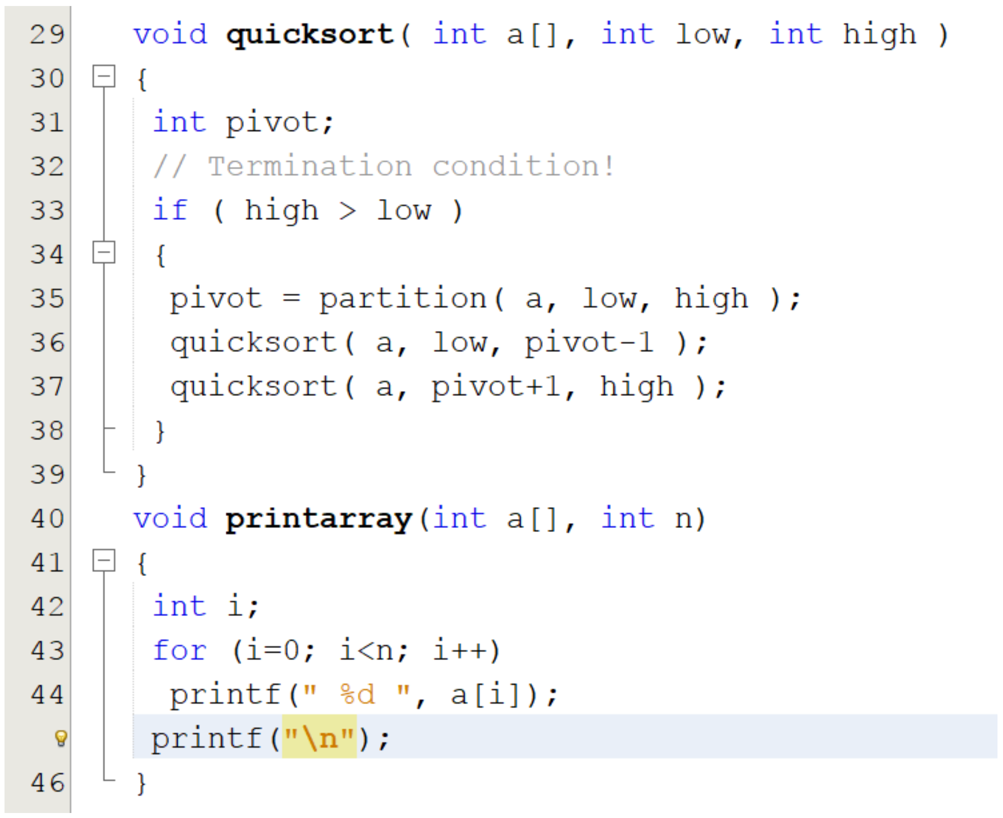
Quick sort:-



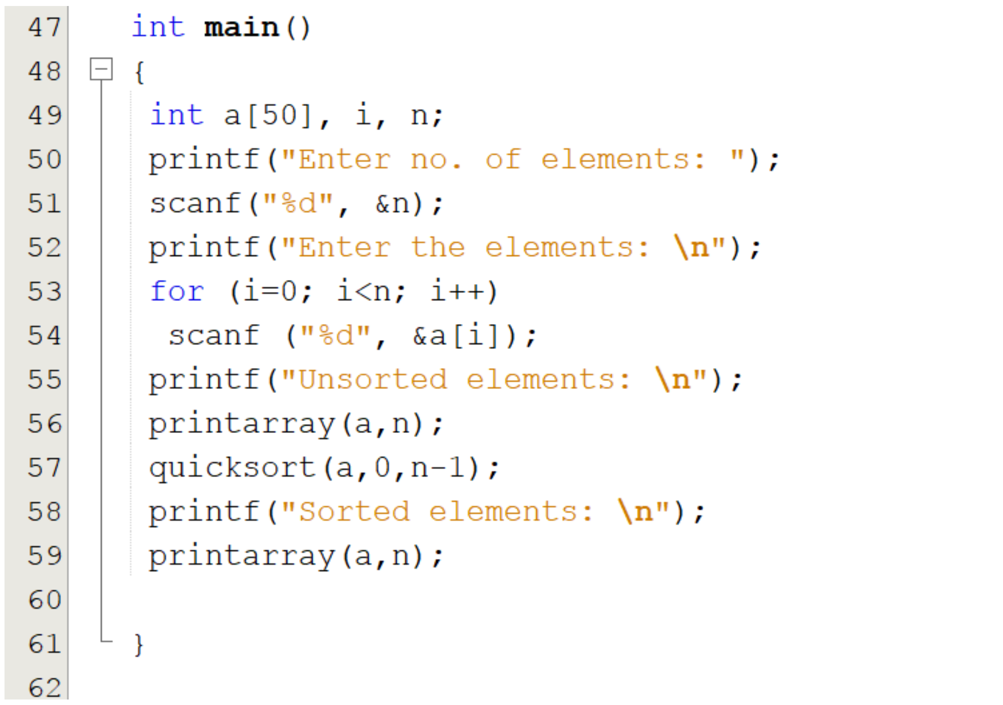
```c
#include<stdio.h>
void swap (int a[], int left, int right)
{
 int temp;
 temp=a[left];
 a[left]=a[right];
 a[right]=temp;
}
```



```c
int partition( int a[], int low, int high )
{
 int left, right;
 int pivot_item;
 int pivot = left = low;
 pivot_item = a[low];
 right = high;
 while ( left < right )
 {
  while( a[left] <= pivot_item )
   left++;
  while( a[right] > pivot_item )
   right--;
  if ( left < right )
   swap(a,left,right);
 }
 a[low] = a[right];
 a[right] = pivot_item;
 return right;
}
```



```c
void quicksort( int a[], int low, int high )
{
 int pivot;
 // Termination condition!
 if ( high > low )
 {
  pivot = partition( a, low, high );
  quicksort( a, low, pivot-1 );
  quicksort( a, pivot+1, high );
 }
}
void printarray(int a[], int n)
{
 int i;
 for (i=0; i<n; i++)
  printf(" %d ", a[i]);
 printf("\n");
}
```

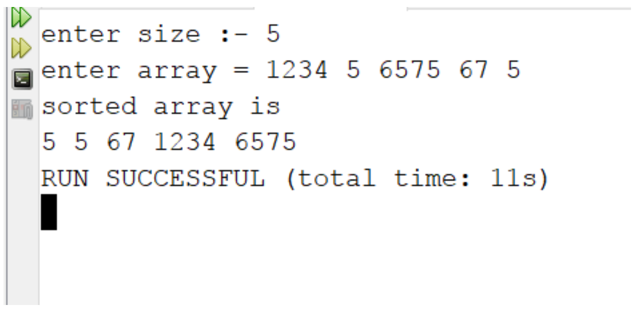


```c
int main()
{
 int a[50], i, n;
 printf("Enter no. of elements: ");
 scanf("%d", &n);
 printf("Enter the elements: \n");
 for (i=0; i<n; i++)
  scanf ("%d", &a[i]);
 printf("Unsorted elements: \n");
 printarray(a,n);
 quicksort(a,0,n-1);
 printf("Sorted elements: \n");
 printarray(a,n);

}
```

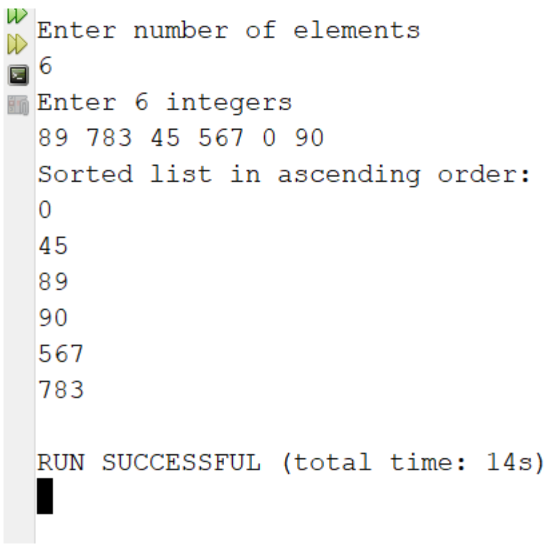
4. Presentation of Results

Selection sort output:-



```
enter size :- 5
enter array = 1234 5 6575 67 5
sorted array is
5 5 67 1234 6575
RUN SUCCESSFUL (total time: 11s)
```

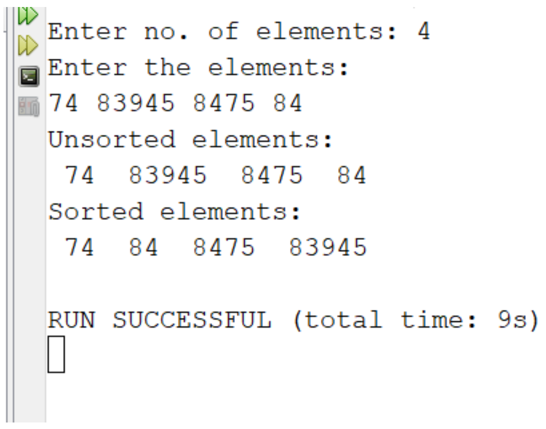
bubble sort output:-



```
Enter number of elements
6
Enter 6 integers
89 783 45 567 0 90
Sorted list in ascending order:
0
45
89
90
567
783

RUN SUCCESSFUL (total time: 14s)
```

Quick sort algorithm:-



```
Enter no. of elements: 4
Enter the elements:
74 83945 8475 84
Unsorted elements:
 74  83945  8475  84
Sorted elements:
 74  84  8475  83945

RUN SUCCESSFUL (total time: 9s)
```

5. Conclusions

All sorting programs have been successfully executed.