

## Assignment 2

Algorithm:-

- Divide

The array is divided into subparts taking pivot as the partitioning point. The elements smaller than the pivot are placed to the left of the pivot and the elements greater than the pivot are placed to the right.

- Conquer

The left and the right subparts are again partitioned using the by selecting pivot elements for them. This can be achieved by recursively passing the subparts into the algorithm.

- Combine

This step does not play a significant role in quicksort. The array is already sorted at the end of the conquer step.

- Worst Case Complexity [Big-O]:  $O(n^2)$

It occurs when the pivot element picked is either the greatest or the smallest element.

This condition leads to the case in which the pivot element lies in an extreme end of the sorted array. One sub-array is always empty and another sub-array contains  $n - 1$  elements. Thus, quicksort is called only on this sub-array.

However, the quick sort algorithm has better performance for scattered pivots.

- **Best Case Complexity [Big-omega]:**  $O(n \log n)$

It occurs when the pivot element is always the middle element or near to the middle element.

- **Average Case Complexity [Big-theta]:**  $O(n \log n)$

It occurs when the above conditions do not occur.

## Space Complexity

The space complexity for quicksort is  $O(\log n)$ .

Code:-

```
#include<bits/stdc++.h>
using namespace std;
int partition(vector<int>& arr,int alt,int end)
{
    int pivot=arr[end];
    int i=alt-1;
    for(int j=alt;j<=end;j++)
    {
        if(arr[j]<pivot)
        {
            i++;
            int temp=arr[i];
            arr[i]=arr[j];
            arr[j]=temp;
        }
    }
    int temp=arr[i+1];
    arr[i+1]=arr[end];
    arr[end]=temp;
    return i+1;
}
```

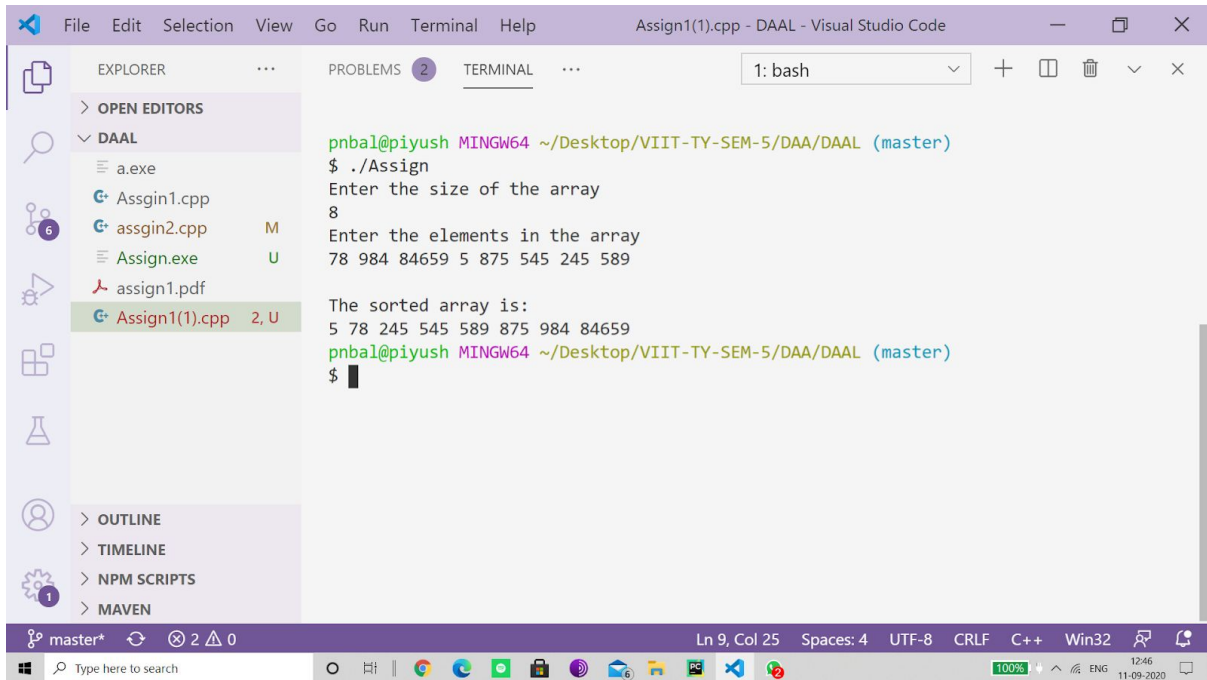
```

void quicksort(vector<int>& arr,int alt,int end)
{
    int q;
    if(alt<end)
    {
        q=partition(arr,alt,end);
        quicksort(arr,alt,q-1);
        quicksort(arr,q+1,end);
    }
}

int main()
{
    int size;
    vector<int> arr;
    cout<<"Enter the size of the array\n";
    cin>>size;
    arr.resize(size);
    cout<<"Enter the elements in the array\n";
    for(int i=0;i<size;i++)
    {
        cin>>arr[i];
    }
    quicksort(arr, 0, size-1);
    cout<<"\nThe sorted array is: \n";
    for(int i=0;i<size;i++)
    {
        cout<<arr[i]<<" ";
    }
    return 0;
}

```

## Output:



The screenshot shows the Visual Studio Code interface with the Explorer, Problems, and Terminal panels. The Explorer panel on the left shows the file structure of a project named 'DAAL', including files like 'a.exe', 'Assgin1.cpp', 'assgin2.cpp', 'Assign.exe', 'assign1.pdf', and 'Assign1(1).cpp'. The Terminal panel on the right shows the output of a C++ program. The program prompts the user to enter the size of the array (8) and the elements in the array (78 984 84659 5 875 545 245 589). It then displays the sorted array: 5 78 245 545 589 875 984 84659.

```
pnbal@piyush MINGW64 ~/Desktop/VIIT-TY-SEM-5/DAA/DAAL (master)
$ ./Assign
Enter the size of the array
8
Enter the elements in the array
78 984 84659 5 875 545 245 589

The sorted array is:
5 78 245 545 589 875 984 84659
pnbal@piyush MINGW64 ~/Desktop/VIIT-TY-SEM-5/DAA/DAAL (master)
$
```