SEPTEMBER 1,2020

DESIGN AND ANALYSIS OF ALGORITHMS

(QUESTION: 3, QUICK SORT)
EXERCISE 3

SUBMITTED BY-
NAME - RAJ KRISHNA
ROLL NO - 1805233
BATCH - CSE-G1
GROUP - B

1. The objective of the Experiment

The objective of the experiment is to sort the numbers present in the given array using **Quick Sort**.

2. Solution Code

```
#include <bits/stdc++.h>
using namespace std;

int count=0;
int partition(vector<int> & values, int left, int right) {
    int pivotIndex = left + (right - left) / 2;
    int pivotValue = values[pivotIndex];
    int i = left, j = right;
    int temp;
    while(i <= j) {
        while(values[i] < pivotValue) {
            i++;
            count++;
        }
        while(values[j] > pivotValue) {
            j--;
            count++;
        }
        if(i <= j) {
            temp = values[i];
            values[i] = values[j];
            values[j] = temp;
            i++;
            j--;
        }
    }
```

```
        return i;
}

void quicksort(vector<int> & values, int left, int right) {
        if(left < right) {
        int pivotIndex = partition(values, left, right);
        quicksort(values, left, pivotIndex - 1);
        quicksort(values, pivotIndex, right);
        }
}

int main()
{
    vector<int> values { 7,9,2,11,19,17,12,5,7,12 };

    cout<<"Array Initialisation- "<<endl;

    for(vector<int>::iterator it = values.begin(); it != values.end(); it++){
          cout <<" "<< *it;
    }
    cout<<endl;

    quicksort(values, 0, values.size() - 1);

    cout<<"After QuickSort- "<<endl;

    for(vector<int>::iterator it = values.begin(); it != values.end(); it++){
          cout <<" "<< *it;
    }
    cout<<endl;
    cout<<"Number of comparison "<<count<<endl;
    return 0;
}
```

3. Summary of the program

Quicksort is an algorithm based on **Divide and Conquer** approach in which the array is split into subarrays and these sub-arrays are **recursively** called to sort the elements.
A **pivot** element is chosen from the array. We can choose any element from the array as the **pivot** element. Here, I have taken the approx. **middle** of the array as the **pivot** element.

Quicksort uses **recursion** for sorting the sub-parts.

On the basis of **Divide and Conquer** approach, quicksort algorithm can be explained as:

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.

Quick sort algorithms with the following time complexity.

Best Case - $O(n^*\log_2 n)$

Average Case - $O(n^*\log_2 n)$

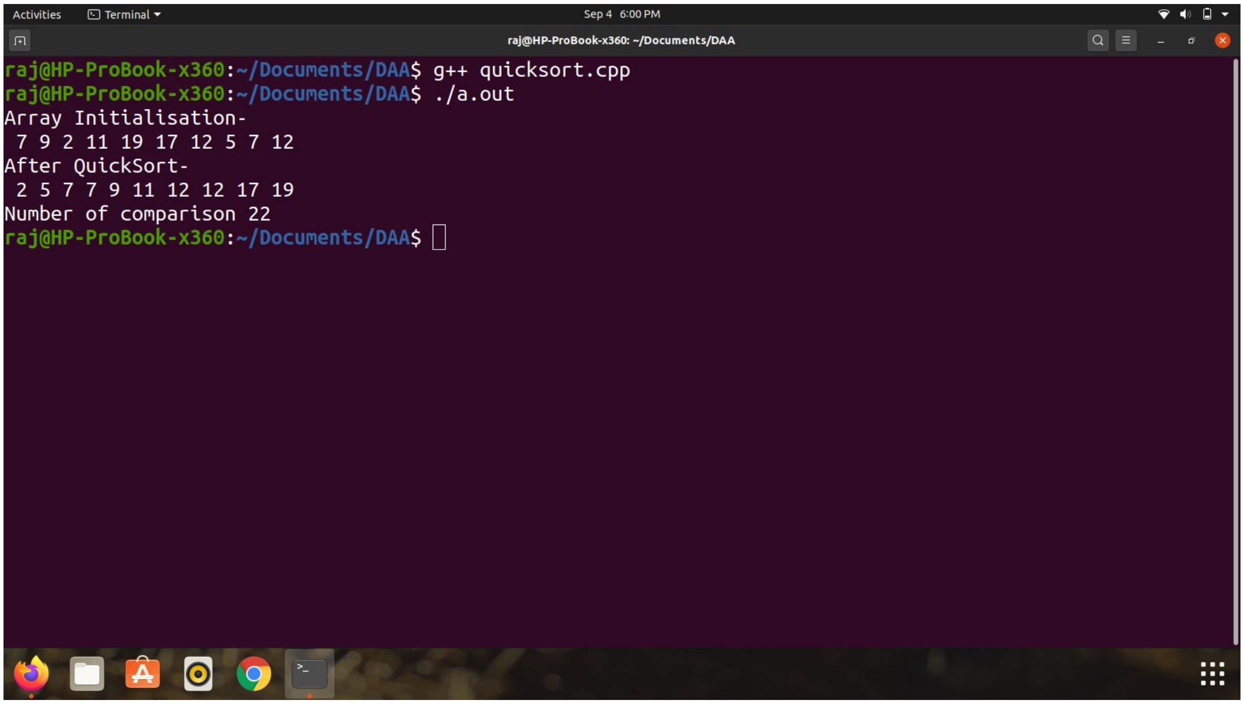
Worst Case - $O(n^2)$

4. **Sample Output**

Array Initialisation-
 7 9 2 11 19 17 12 5 7 12

After QuickSort-
 2 5 7 7 9 11 12 12 17 19

Number of comparison 22



```
raj@HP-ProBook-x360:~/Documents/DAA$ g++ quicksort.cpp
raj@HP-ProBook-x360:~/Documents/DAA$ ./a.out
Array Initialisation-
 7 9 2 11 19 17 12 5 7 12
After QuickSort-
 2 5 7 7 9 11 12 12 17 19
Number of comparison 22
raj@HP-ProBook-x360:~/Documents/DAA$ 
```