Experiment 2.1

Aim: *Sort a given set of elements using the Quick sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted. The elements can be read from a file or can be generated using the random number generator.*

Objectives: *To understand quick sort.*

Input/Apparatus Used: *VS CODE*

Procedure/Algorithm:

1. *Quick Sort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quick Sort that pick pivot in different ways.*
2. *Always pick first element as pivot.*
3. *Always pick last element as pivot (implemented below)*
4. *Pick a random element as pivot.*
5. *Pick median as pivot.*
6. *The key process in quick Sort is partition (). Target of partition() is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x.*

Code:

```
#include <iostream>
#include <vector>
#include <chrono>
using namespace std;

int partition(vector<int>& arr, int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);
```

```
    for (int j = low; j <= high - 1; j++) {
        if (arr[j] <= pivot) {
            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[high]);
    return (i + 1);
}

void quickSort(vector<int>& arr, int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    vector<int> arr = {12, 11, 13, 5, 6, 7};
    int n = arr.size();

    cout << "Original array: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    auto start = chrono::high_resolution_clock::now();
```

```
    quickSort(arr, 0, n - 1);

    auto end = chrono::high_resolution_clock::now();
    chrono::duration<double> duration = end - start;

    cout << "Sorted array: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
    cout << "Time taken by Quick Sort: " << duration.count() << " seconds" << endl;
    return 0;
}
```

Observations/Outcome :

```
Original array: 12 11 13 5 6 7
Sorted array: 5 6 7 11 12 13
Time taken by Quick Sort: 1.044e-06 seconds
```

Time Complexity:

- *Best case : O(nlogn)*
- *Average case : O(nlogn)*
- *Worst case : $O(n^2)$*