Experiment 2.1

Student Name: Pranav Aggarwal
UID: 21BCS3197
Branch: CSE
Section/Group: 619 B
Semester: 5th
Date of Performance: 02/09/23
Subject Name: Design and Analysis of Algorithms
Subject Code:21CSH-311

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.

Objective:

To understand quick sort.

Input/Apparatus Used:

Quick sort concept is used.

Procedure/Algorithm/Code:

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.

● Always pick first element as pivot.

● Always pick last element as pivot (implemented below)

● Pick a random element as pivot.

● Pick median as pivot.

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.

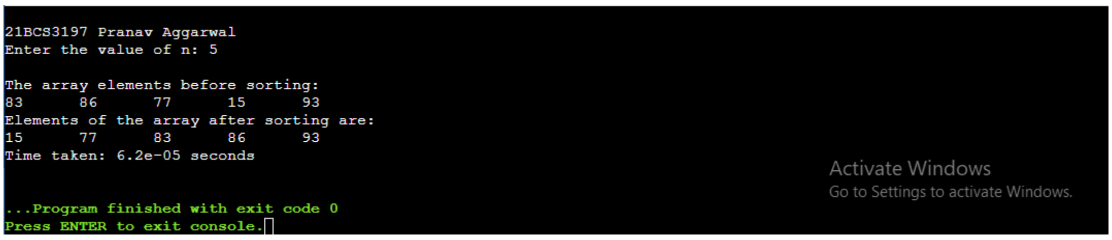
PROGRAM:

```
#include <iostream>

#include <ctime>

#include <cstdlib>
```

```
#define max 500


void qsort(int[], int, int);

int partition(int[], int, int);


using namespace std;


int main() {

    cout << "\n21BCS3197 Pranav Aggarwal\n";

    int a[max], i, n;

    clock_t s, e, z;

    s = clock();


    cout << "Enter the value of n: ";

    cin >> n;
    for (i = 0; i < n; i++)
        a[i] = rand() % 100;

    cout << "\nThe array elements before sorting:\n";
    for (i = 0; i < n; i++)
        cout << a[i] << "\t";

    qsort(a, 0, n - 1);

    cout << "\nElements of the array after sorting are:\n";
    for (i = 0; i < n; i++)
        cout << a[i] << "\t";

    e = clock();
    z = e - s;

    cout << "\nTime taken: " << static_cast<double>(z) / CLOCKS_PER_SEC << " seconds"
<< endl;
```

```
    return 0;
}

void qsort(int a[], int low, int high) {
    int j;
    if (low < high) {
        j = partition(a, low, high);
        qsort(a, low, j - 1);
        qsort(a, j + 1, high);
    }
}

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

Output:



```
21BCS3197 Pranav Aggarwal
Enter the value of n: 5

The array elements before sorting:
83      86      77      15      93
Elements of the array after sorting are:
15      77      83      86      93
Time taken: 6.2e-05 seconds


...Program finished with exit code 0
Press ENTER to exit console.
```

Complexity:

Best case - O (nlogn)

Average case - O (nlogn)

Worst case - O (n2)

Learning Outcomes:

1. Understood the concept of Quick Sort

2. Implemented quick sort on a list on randomly generated numbers

3. Calculated the time required to sort the list