

### **Experiment:2.1**

**Student Name: Virat Samdarshi**

**UID: 22BCS12648**

**Branch: B.E CSE**

**Section/Group: IOT-627-B**

**Semester: 5th**

**Date of Performance: 21-08-24**

**Subject Name: DAA lab**

**Subject Code: 22CSH-311**

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.
2. **Objective:** The objective of this experiment is to implement the Quick Sort algorithm to sort a given set of elements and measure the time required for the sorting process. These elements will either be read from a file or generated using a random number generator.

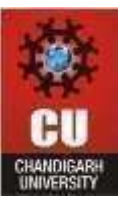
### **3. Implementation/Code:**

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <iomanip>
using namespace std;

void swapElements(int arr[], int index1, int index2) {
    int temp = arr[index1];
    arr[index1] = arr[index2];
    arr[index2] = temp;
}

int partitionArray(int arr[], int start, int end) {
    int pivot = arr[end];
    int i = start - 1;

    for (int j = start; j < end; j++) {
        if (arr[j] <= pivot) {
            i++;
            swapElements(arr, i, j);
        }
    }
}
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
swapElements(arr, i + 1, end);
return i + 1;
}

void quickSortArray(int arr[], int start, int end) {
    if (start < end) {
        int pivotIndex = partitionArray(arr, start, end);
        quickSortArray(arr, start, pivotIndex - 1);
        quickSortArray(arr, pivotIndex + 1, end);
    }
}

int main() {
    int numElements;
    cout << "Enter the number of elements: ";
    cin >> numElements;

    int arr[numElements];
    char inputMethod;

    cout << "Would you like to input from a file? (y/n): ";
    cin >> inputMethod;

    if (inputMethod == 'y' || inputMethod == 'Y') {
        string fileName;
        cout << "Enter the filename: ";
        cin >> fileName;

        ifstream inputFile(fileName);
        if (!inputFile) {
            cerr << "Error: Could not open file!" << endl;
            return 1;
        }

        for (int i = 0; i < numElements && inputFile >> arr[i]; i++);

        inputFile.close();
    } else {
        cout << "Generating random array...\n";
        srand(time(0));
        for (int i = 0; i < numElements; i++) {
            arr[i] = rand() % 10000;
            cout << arr[i] << " ";
        }
        cout << endl; }
}
```

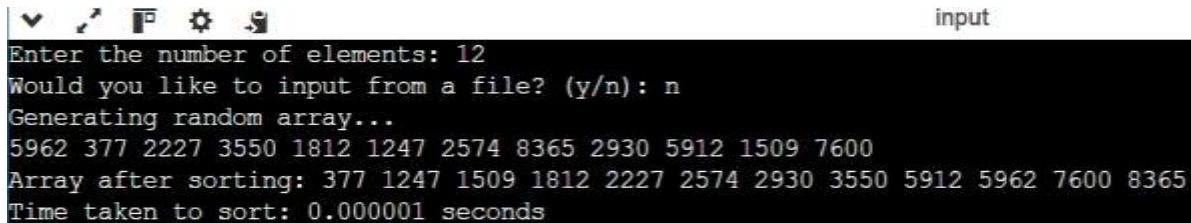
```
clock_t startTime = clock();
quickSortArray(arr, 0, numElements - 1);
clock_t endTime = clock();

cout << "Array after sorting: ";
for (int i = 0; i < numElements; i++) {
    cout << arr[i] << " ";
}
cout << endl;

double timeTaken = double(endTime - startTime) / CLOCKS_PER_SEC;
cout << fixed << setprecision(6);
cout << "Time taken to sort: " << timeTaken << " seconds" << endl;

return 0;
}
```

#### 4. Output:



```
Enter the number of elements: 12
Would you like to input from a file? (y/n): n
Generating random array...
5962 377 2227 3550 1812 1247 2574 8365 2930 5912 1509 7600
Array after sorting: 377 1247 1509 1812 2227 2574 2930 3550 5912 5962 7600 8365
Time taken to sort: 0.000001 seconds
```

#### 5. Time Complexity:

Best Case:  $O(n \log n)$

Average Case:  $O(n \log n)$

Worst Case:  $O(n^2)$