<u>Question 1</u>

total elements in array: 3

Recursive Quick Sort algorithm:

time taken is 0.572000 seconds

Non-recursive Quick Sort algorithm:

time taken is 5.672000 seconds

total elements in array: 5

Recursive Quick Sort algorithm:

time taken is 0.453000 seconds

Non-recursive Quick Sort algorithm:

time taken is 3.517000 seconds

total elements in array: 7

Recursive Quick Sort algorithm:

time taken is 0.639000 seconds

Non-recursive Quick Sort algorithm:

time taken is 4.510000 seconds

total elements in array: 15

Recursive Quick Sort algorithm:

time taken is 1.485000 seconds

Non-recursive Quick Sort algorithm:

time taken is 8.574000 seconds

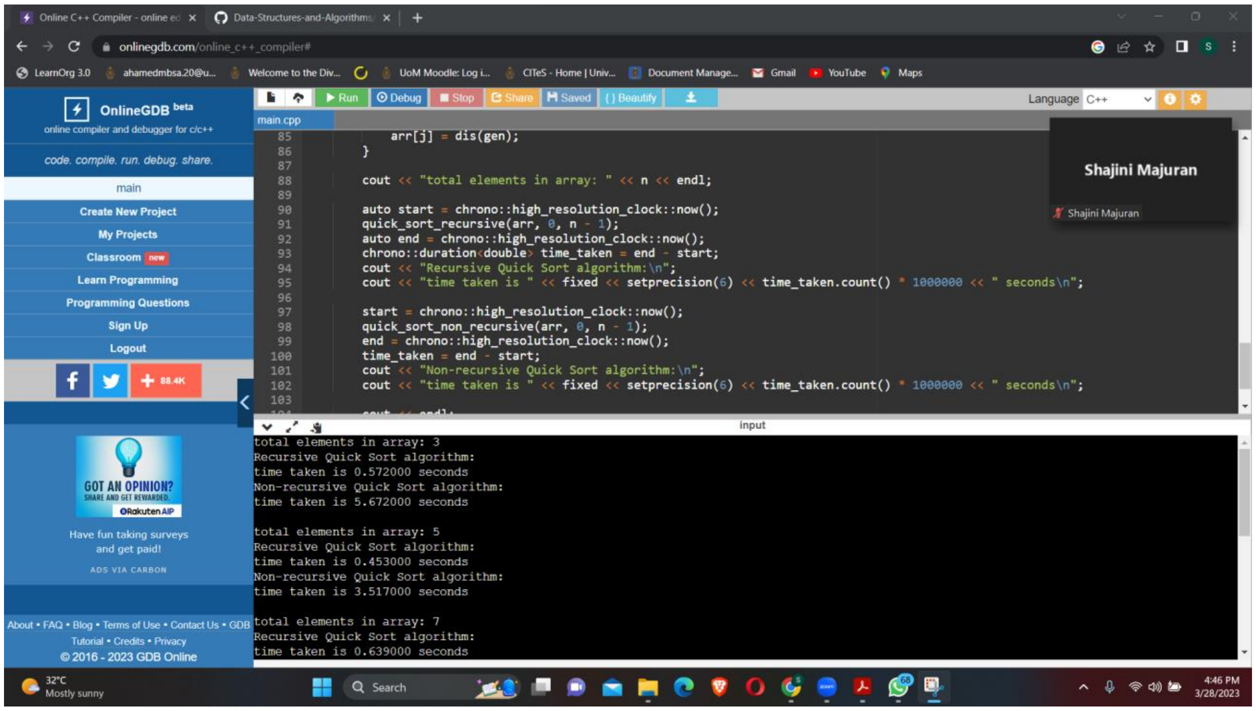
total elements in array: 20

Recursive Quick Sort algorithm:

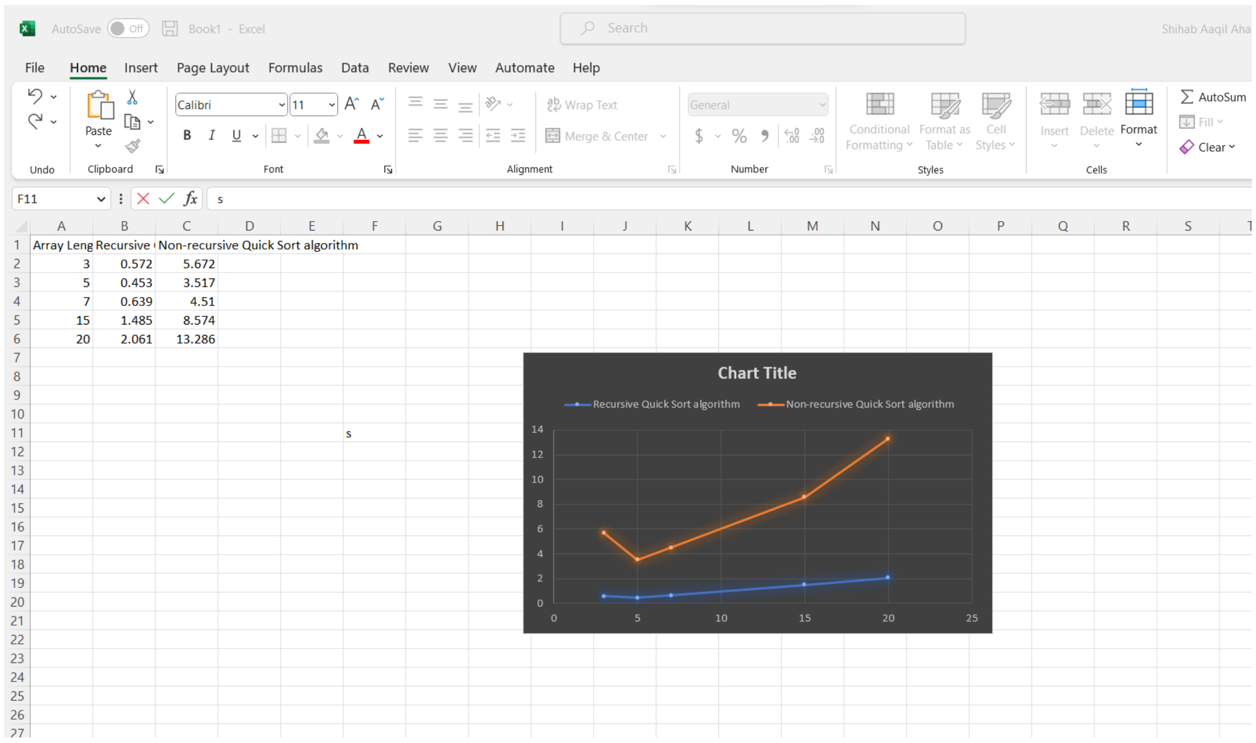
time taken is 2.061000 seconds

Non-recursive Quick Sort algorithm:

time taken is 13.286000 seconds


```
            arr[j] = dis(gen);
        }

        cout << "total elements in array: " << n << endl;

        auto start = chrono::high_resolution_clock::now();
        quick_sort_recursive(arr, 0, n - 1);
        auto end = chrono::high_resolution_clock::now();
        chrono::duration<double> time_taken = end - start;
        cout << "Recursive Quick Sort algorithm:\n";
        cout << "time taken is " << fixed << setprecision(6) << time_taken.count() * 1000000 << " seconds\n";

        start = chrono::high_resolution_clock::now();
        quick_sort_non_recursive(arr, 0, n - 1);
        end = chrono::high_resolution_clock::now();
        time_taken = end - start;
        cout << "Non-recursive Quick Sort algorithm:\n";
        cout << "time taken is " << fixed << setprecision(6) << time_taken.count() * 1000000 << " seconds\n";

```

```
total elements in array: 3
Recursive Quick Sort algorithm:
time taken is 0.572000 seconds
Non-recursive Quick Sort algorithm:
time taken is 5.672000 seconds

total elements in array: 5
Recursive Quick Sort algorithm:
time taken is 0.453000 seconds
Non-recursive Quick Sort algorithm:
time taken is 3.517000 seconds

total elements in array: 7
Recursive Quick Sort algorithm:
time taken is 0.639000 seconds
```

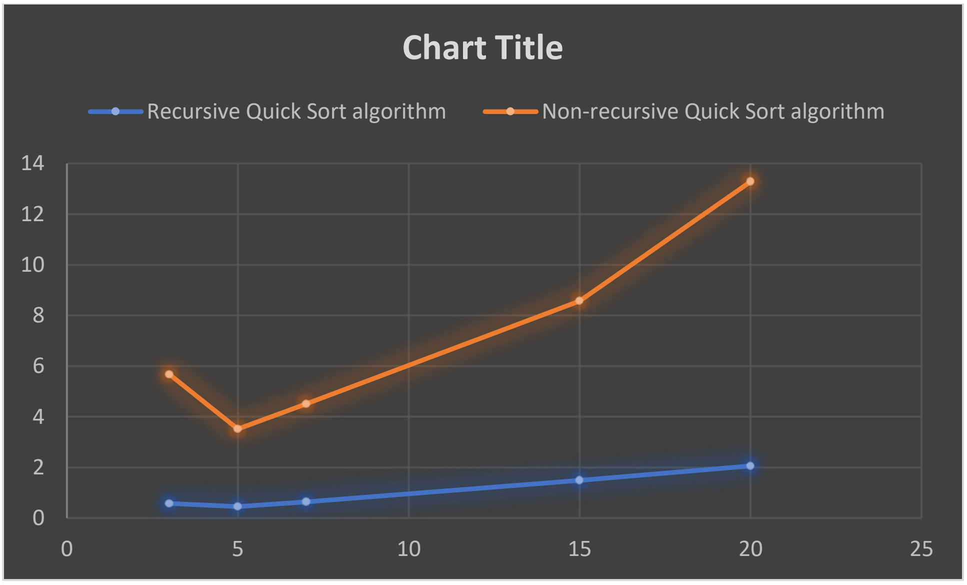


Array Leng	Recursive (	Non-recursive Quick Sort algorithm
3	0.572	5.672
5	0.453	3.517
7	0.639	4.51
15	1.485	8.574
20	2.061	13.286

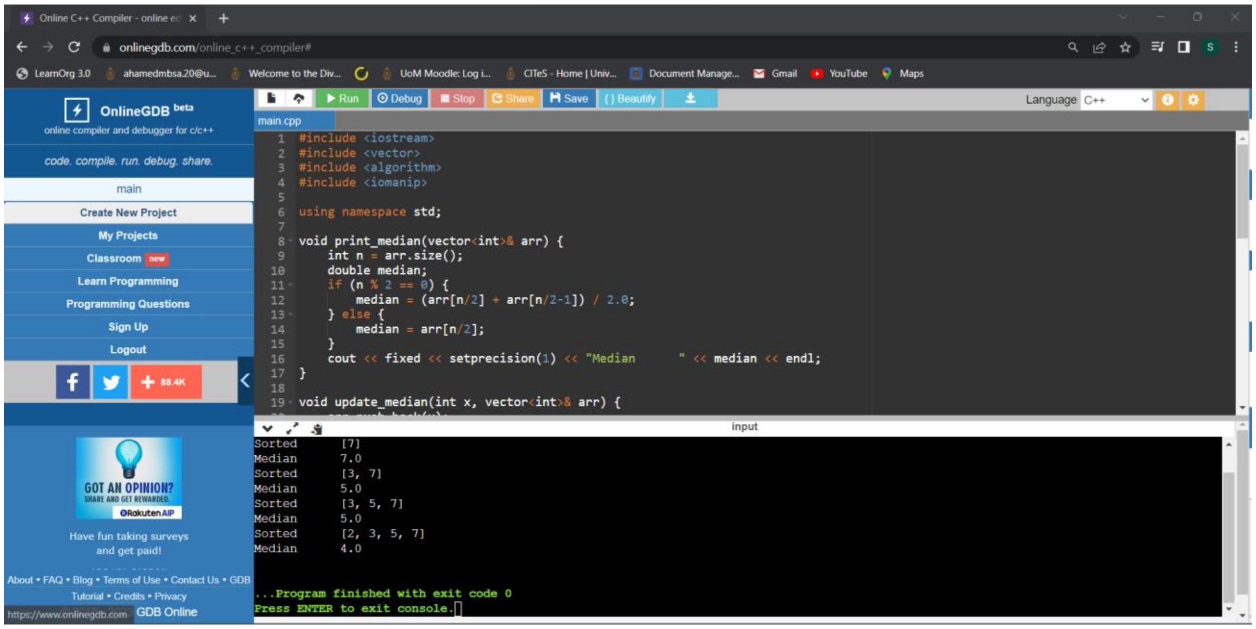
According to the provided data, the Non-recursive Quick Sort algorithm executes more quickly as the array size rises, whereas the Recursive Quick Sort method typically performs quicker for lower array sizes.

Recursive Quick Sort, for instance, takes 0.572000 seconds when the array size is 3, whereas Non-recursive Quick Sort, on the other hand, takes 5.672000 seconds. The Non-recursive Quick Sort technique, on the other hand, takes 13.286000 seconds when the array size is 20, whereas the Recursive Quick Sort algorithm takes 2.061000 seconds.

This is explained by the fact that recursion's cost rises as the size of the array grows. The Recursive Quick Sort technique might be more efficient for smaller arrays since it has less overhead.



Question 2



```
#include <iostream>
#include <vector>
#include <algorithm>
#include <iomanip>

using namespace std;

void print_median(vector<int>& arr) {
    int n = arr.size();
    double median;
    if (n % 2 == 0) {
        median = (arr[n/2] + arr[n/2-1]) / 2.0;
    } else {
        median = arr[n/2];
    }
    cout << fixed << setprecision(1) << "Median       " << median << endl;
}

void update_median(int x, vector<int>& arr) {
```

```
Sorted      [7]
Median      7.0
Sorted      [3, 7]
Median      5.0
Sorted      [3, 5, 7]
Median      5.0
Sorted      [2, 3, 5, 7]
Median      4.0


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

4

7 3 5 2

Sorted [7]

Median 7.0

Sorted [3, 7]

```
Median     5.0

Sorted     [3, 5, 7]

Median     5.0

Sorted     [2, 3, 5, 7]

Median     4.0
```

GitHub Link: https://github.com/MB-Shihab-Aaqil-Ahamed/Data-Structures-and-Algorithms