

Question 1

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

The screenshot displays the OnlineGDB web interface. The browser address bar shows `onlinegdb.com/online_c++_compiler#`. The left sidebar contains navigation links: **OnlineGDB beta**, `code compile run debug share`, **main**, **Create New Project**, **My Projects**, **Classroom** (with a 'new' badge), **Learn Programming**, **Programming Questions**, **Sign Up**, and **Logout**. Below these are social media icons for Facebook, Twitter, and a '+ 88.4K' button. A 'GOT AN OPINION?' survey banner is also visible. The main editor area shows a C++ file named `main.cpp` with the following code:

```
85     arr[j] = dis(gen);
86   }
87
88   cout << "total elements in array: " << n << endl;
89
90   auto start = chrono::high_resolution_clock::now();
91   quick_sort_recursive(arr, 0, n - 1);
92   auto end = chrono::high_resolution_clock::now();
93   chrono::duration<double> time_taken = end - start;
94   cout << "Recursive Quick Sort algorithm:\n";
95   cout << "time taken is " << fixed << setprecision(6) << time_taken.count() * 1000000 << " seconds\n";
96
97   start = chrono::high_resolution_clock::now();
98   quick_sort_non_recursive(arr, 0, n - 1);
99   end = chrono::high_resolution_clock::now();
100  time_taken = end - start;
101  cout << "Non-recursive Quick Sort algorithm:\n";
102  cout << "time taken is " << fixed << setprecision(6) << time_taken.count() * 1000000 << " seconds\n";
103
104  cout << endl;
```

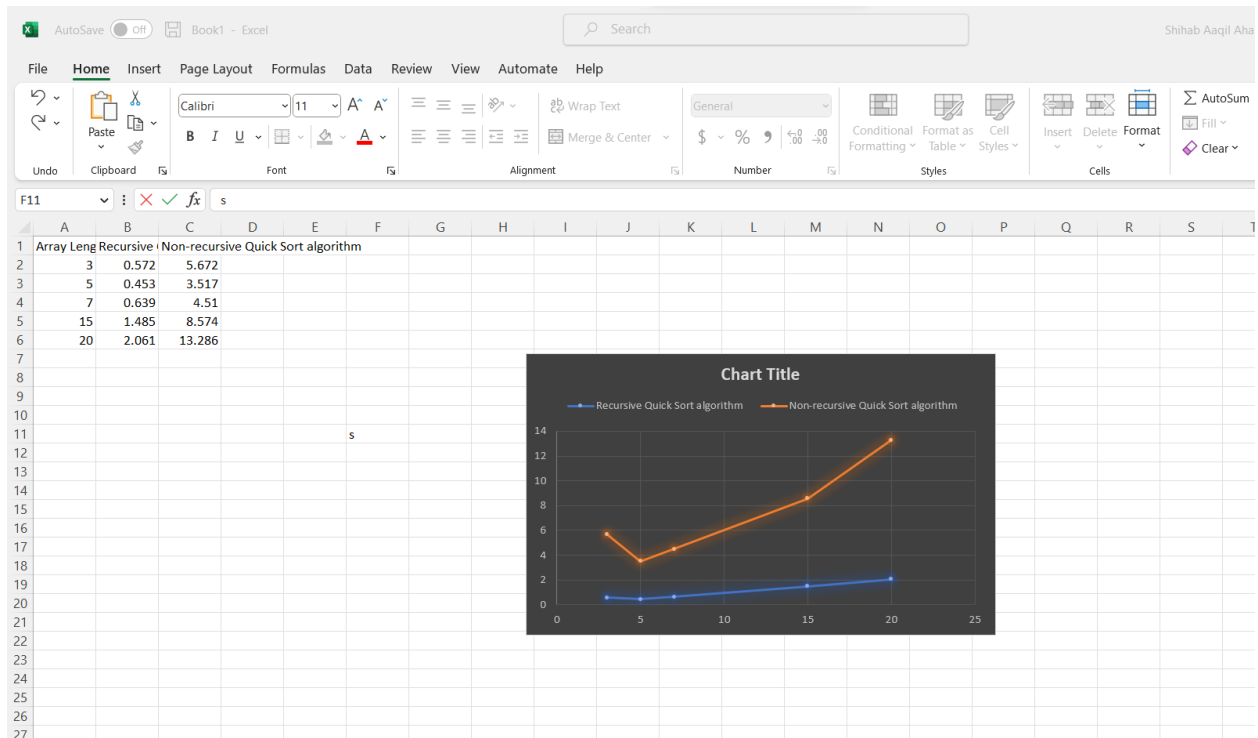
The output window at the bottom shows the results of the program execution for three different array sizes:

```
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
```

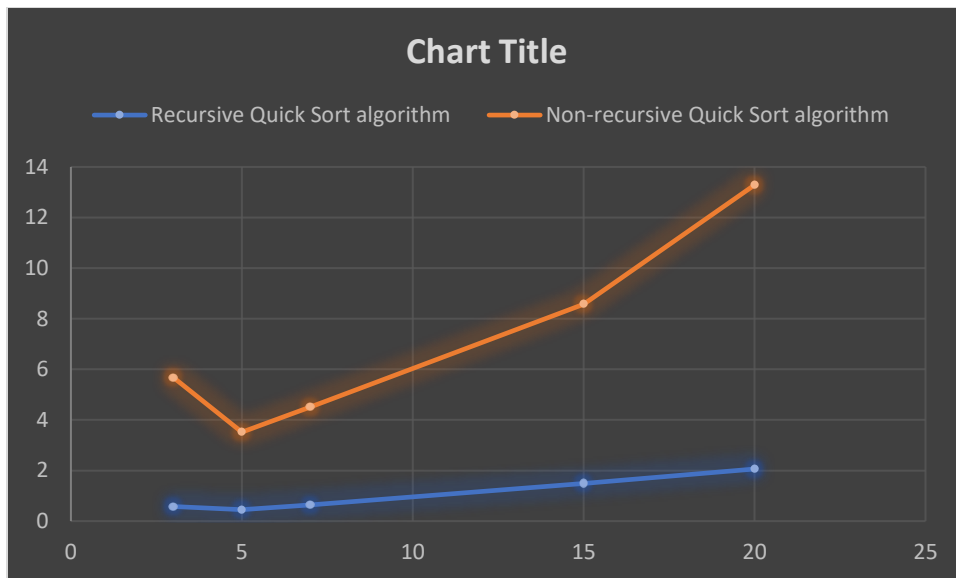
The Windows taskbar at the bottom indicates a temperature of 32°C, 'Mostly sunny' weather, and the time 4:46 PM on 3/28/2023.



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

```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <iomanip>
5
6 using namespace std;
7
8 void print_median(vector<int>& arr) {
9     int n = arr.size();
10    double median;
11    if (n % 2 == 0) {
12        median = (arr[n/2] + arr[n/2-1]) / 2.0;
13    } else {
14        median = arr[n/2];
15    }
16    cout << fixed << setprecision(1) << "Median    " << median << endl;
17 }
18
19 void update_median(int x, vector<int>& arr) {
20     arr.push_back(x);
21 }

```

Input

```

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>