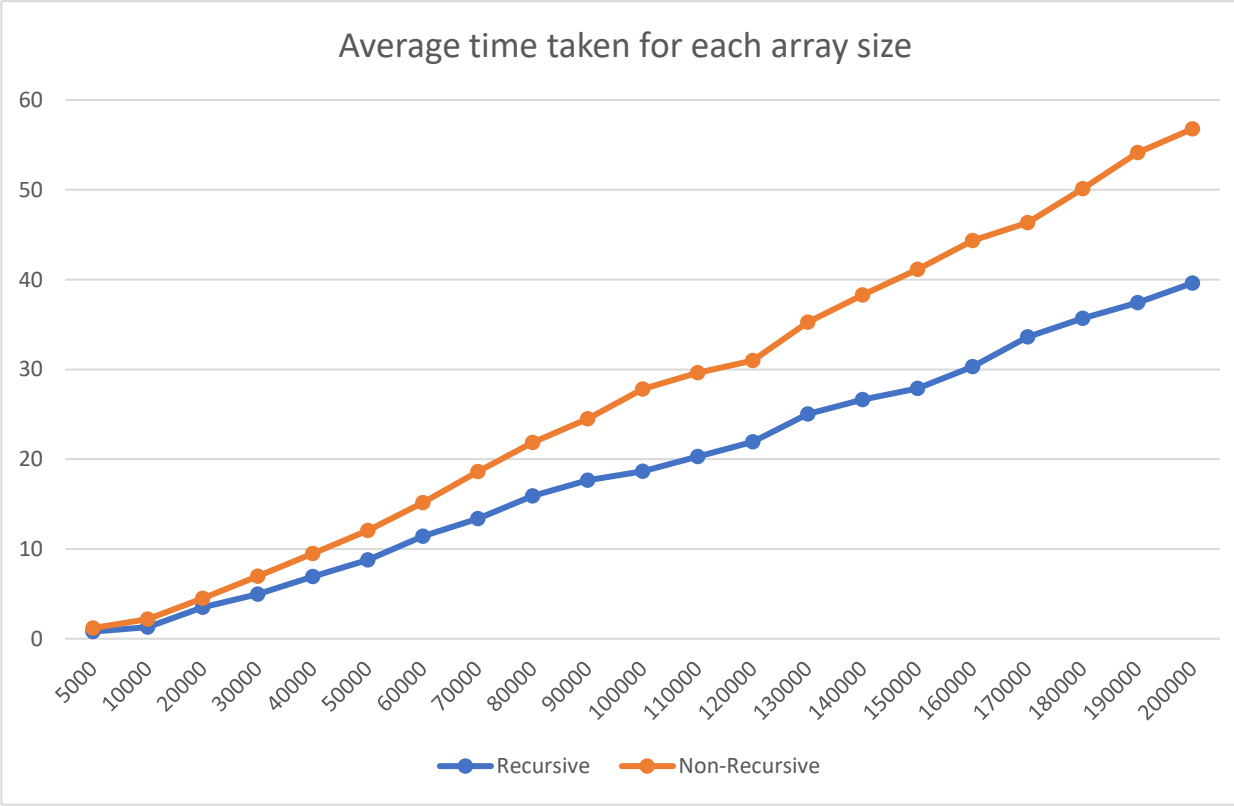
Lab Report – Week 5

CS2023 Data Structures and Algorithms

Dept. of Computer Science and Engineering, University of Moratuwa

Name: Tharindu Perera Index Number: 210472J

Question 1



Time in milliseconds in 10 consecutive runs (average in the last column)

	Recursive										
5000	0.969	0	0.998	0.998	0.998	1.013	1	0.997	0	0.995	0.7968
10000	0.997	0.978	1.994	0.998	1.994	1.979	0.999	0.998	0.997	0.998	1.2932
20000	3.989	1.987	2.991	2.99	3.989	2.993	3.989	3.99	3.989	3.997	3.4904
30000	5.012	3.989	3.989	4.988	5.984	5.988	5.808	5.001	4.99	3.981	4.973
40000	5.994	6.98	6.981	6.954	7.979	7.009	8.386	5.968	6.98	5.984	6.9215
50000	7.942	8.978	7.978	9.033	9.973	8.493	9.969	7.978	8.485	8.977	8.7806
60000	9.973	11.965	10.011	11.55	10.973	11.941	13.963	10.974	11.972	10.969	11.4291
70000	13.96	15.535	11.934	13.927	12.48	14.957	13.962	11.993	11.946	13.021	13.3715
80000	13.965	18.946	15.96	16.956	15.957	16.983	15.957	14.007	15.922	14.447	15.91
90000	17.473	19.947	18.46	16.955	14.96	18.857	15.011	14.932	17.952	21.941	17.6488
100000	16.953	18.946	21.944	17.979	17.952	17.95	16.971	17.952	18.948	20.973	18.6568
110000	18.976	18.989	21.939	18.979	23.938	19.461	20.493	18.977	21.191	19.807	20.275
120000	20.919	21.506	23.937	23.664	22.452	20.943	19.934	19.95	23.934	21.999	21.9238
130000	24.935	22.913	26.959	24.896	23.93	24.971	26.914	24.442	27.936	22.438	25.0334
140000	24.453	27.922	31.913	24.933	28.921	24.9	23.91	28.961	26.516	23.961	26.639
150000	25.932	28.923	30.918	25.884	26.948	26.44	27.45	26.929	28.462	30.891	27.8777
160000	30.942	30.487	29.923	27.438	28.41	27.925	29.945	31.118	36.879	29.956	30.3023
170000	31.471	29.894	36.417	27.926	33.907	36.901	34.865	39.893	32.912	31.919	33.6105
180000	34.907	35.903	35.902	30.969	37.97	33.509	35.454	32.913	35.479	43.88	35.6886
190000	35.904	36.48	34.555	37.406	36.471	34.884	41.933	43.369	37.905	35.555	37.4462
200000	40.435	37.624	42.866	43.882	39.867	36.929	37.883	39.922	37.9	38.895	39.6203

	Non-Recursive										
5000	0.998	0.999	1.011	1.995	0.997	0.953	0.999	1.952	0.998	0.986	1.1888
10000	1.995	1.993	1.981	1.994	1.994	2.992	2.992	1.995	1.508	2.51	2.1954
20000	4.071	5.985	3.99	3.989	3.99	4.986	4.986	3.989	5.017	3.988	4.4991
30000	5.884	6.982	6.98	6.98	8.968	6.981	6.981	5.978	6.997	6.981	6.9712
40000	10.014	8.976	8.976	8.977	9.949	9.974	9.974	9.976	8.934	9.004	9.4754
50000	12.454	11.97	11.968	9.972	13.476	11	11.97	11.943	11.968	13.974	12.0695
60000	15.958	16.47	12.969	13.009	16.956	16.911	14.95	12.967	15.564	15.919	15.1673
70000	18.981	18.935	15.475	15.537	16.954	19.951	21.512	16.344	19.506	22.941	18.6136
80000	19.94	20.944	20.944	18.972	25.96	22.485	19.975	19.944	22.941	26.477	21.8582
90000	28.934	23.935	24.961	20.872	28.95	25.901	23.932	21.984	21.489	23.914	24.4872
100000	27.452	26.933	27.901	30.887	28.437	27.925	29.918	28.953	24.932	24.933	27.8271
110000	25.929	30.433	33.429	30.428	26.925	29.488	31.469	31.414	28.877	27.954	29.6346
120000	32.912	32.912	30.92	29.919	29.948	30.889	32.918	28.917	27.929	32.426	30.969
130000	42.402	34.899	35.9	32.921	37.448	33.909	35.878	31.942	31.467	35.903	35.2669
140000	37.898	34.452	38.417	35.109	36.901	43.403	39.412	37.396	38.896	40.895	38.2779
150000	39.52	36.946	41.916	38.928	43.288	41.915	39.858	46.879	42.464	39.804	41.1518
160000	43.932	40.007	49.36	46.849	43.913	44.881	43.434	49.372	39.891	41.888	44.3527
170000	42.865	49.868	45.876	53.371	42.886	47.386	42.857	53.857	40.957	43.389	46.3312
180000	48.081	59.34	48.392	53.401	51.428	49.896	46.391	53.14	46.154	44.881	50.1104
190000	51.411	52.899	48.896	58.844	59.391	52.876	51.858	58.843	49.866	56.488	54.1372
200000	55.458	54.399	56.38	57.36	58.841	49.375	50.486	60.472	67.334	57.846	56.7951

Code

Helper functions

C++ helpers.cpp Week5\Q1\QuickSort-NonRecursive U

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <chrono>

using namespace std;

struct Algorythm
{
    string name;
    void (*algorythem)(int[], int, int);
};

void print_arr(int arr[], int length) {
    for (int i = 0; i < length; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int** generate_arrays(int N, int array_lengths[], unsigned int random_seed) {
    srand(random_seed);
    int** arrays = new int* [N];
    for (int i = 0; i < N; i++) {
        int length = array_lengths[i];
        int* array = new int[length];
        for (int j = 0; j < length; j++) {
            array[j] = rand() % 10000001;
        }
        arrays[i] = array;
    }
    return arrays;
}

float time_algorithm( Algorythm sorting_algorythm, int** arrays, int N, int array_lengths[]) {

    cout << sorting_algorythm.name << endl;
    for (int i = 0; i < N; i++) {
        int length = array_lengths[i];

        #ifdef PRINT_ARRAYS
            print_arr(arrays[i], length);
        #endif

        auto start = chrono::high_resolution_clock::now();
        sorting_algorythm.algorythem(arrays[i], 0, length-1);
        auto finish = chrono::high_resolution_clock::now();

        #ifdef PRINT_ARRAYS
            print_arr(arrays[i], length);
        #endif
        cout << chrono::duration_cast<chrono::microseconds>(finish - start).count()/1000.0 << endl;

    }
    cout << endl;
}
```

Recursive Quick Sort

main.cpp Week5\Q1\QuickSort-Recursive U

```cpp
// #define PRINT_ARRAYS
#include "helpers.cpp"

using namespace std;

int partition(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(int arr[], int low = 0, int high = -1) {
    if (low < high) {
        int pivot_index = partition(arr, low, high);
        quicksort(arr, low, pivot_index - 1);
        quicksort(arr, pivot_index + 1, high);
    }
}

int main() {
    const int number_of_arrays = 21;
    int array_lengths[number_of_arrays] = { 5000, 10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000, 110000, 120000, 130000, 140000, 150000, 160000, 170000, 180000, 190000, 200000 };
    const Algorythm SortingAlgorythem = {"Quick Sort", &quicksort};
    unsigned int random_seed = time(NULL);

    int** arrays = generate_arrays(number_of_arrays, array_lengths, random_seed);
    time_algorithm(SortingAlgorythem, arrays, number_of_arrays, array_lengths);
}
```

Non-Recursive Quick Sort

main.cpp Week5\Q1\QuickSort-NonRecursive U

```cpp
// #define PRINT_ARRAYS
#include "helpers.cpp"
#include <stack>
using namespace std;

int partition(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(int arr[], int low, int high) {
    stack<int> stk;
    stk.push(low);
    stk.push(high);
    while (!stk.empty()) {
        high = stk.top();
        stk.pop();
        low = stk.top();
        stk.pop();
        int pivot_index = partition(arr, low, high);
        if (pivot_index - 1 > low) {
            stk.push(low);
            stk.push(pivot_index - 1);
        }
        if (pivot_index + 1 < high) {
            stk.push(pivot_index + 1);
            stk.push(high);
        }
    }
}

int main() {

    const int number_of_arrays = 21;
    int array_lengths[number_of_arrays] = { 5000, 10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000, 110000, 120000, 130000, 140000, 150000, 160000, 170000, 180000, 190000, 200000 };
    const Algorythm SortingAlgorythem = {"Quick Sort", &quicksort};
    unsigned int random_seed = time(NULL);

    int** arrays = generate_arrays(number_of_arrays, array_lengths, random_seed);
    time_algorithm(SortingAlgorythem, arrays, number_of_arrays, array_lengths);
}
```

Question 2

main.cpp

```cpp
#include <iostream>
#include <vector>
```

```cpp
#include <string.h>
#include <sstream>

using namespace std;

void insertIntoSortedArray(vector<int>& arr, int element) {
    int i = 0;
    while (i < arr.size() && arr[i] < element) {
        i++;
    }
    arr.insert(arr.begin() + i, element);
}

bool stringIsInt(string s) {
    for (int i = 0; i < s.length(); i++)
    {
        if(isdigit(s[i])) continue;
        if (i == 0 && s[i]=='-') continue;
        return false;
    }
    return true;
}

void print_arr(vector<int> arr) {
    cout << "{";
    for (int i = 0; i < arr.size()-1; i++) {
        cout << arr[i] << ", ";
    }
    cout << arr[arr.size()-1] << "}";
}

int main() {

    vector<int> arr = {};
    string input = "";

    cout << "Enter integers one by one. Enter \"end\" to stop stream.\n\n";

    while (true)
    {
        cout << ">";
        cin >> input;
        if (input == "end") break;
```

```
46          if (!stringIsInt(input)) throw invalid_argument("Integer value
    expected. Enter end to stop stream.");
47
48          stringstream ss;
49          int next;
50          ss << input;
51          ss >> next;
52
53          insertIntoSortedArray(arr, next);
54
55          float median = (arr[(arr.size() - 1) /2] + arr[arr.size() / 2])
    /2.0;
56
57          cout << "Array: ";
58          print_arr(arr);
59          cout << "     | Median: " << median << endl << endl;
60
61      }
62
63      return 0;
64  }
65
```

```
PS C:\Users\thari\UoM-DSA-S2-Labs\Week5\Q2> ./bin/app.exe
Enter integers one by one. Enter "end" to stop stream.

>7
Array: {7}      | Median: 7

>3
Array: {3, 7}      | Median: 5

>5
Array: {3, 5, 7}      | Median: 5

>2
Array: {2, 3, 5, 7}      | Median: 4

>end
PS C:\Users\thari\UoM-DSA-S2-Labs\Week5\Q2> ./bin/app.exe
Enter integers one by one. Enter "end" to stop stream.

>23
Array: {23}      | Median: 23

>-3
Array: {-3, 23}      | Median: 10

>423
Array: {-3, 23, 423}      | Median: 23

>-72
Array: {-72, -3, 23, 423}      | Median: 10

>33
Array: {-72, -3, 23, 33, 423}      | Median: 23

>4234
Array: {-72, -3, 23, 33, 423, 4234}      | Median: 28

>3
Array: {-72, -3, 3, 23, 33, 423, 4234}      | Median: 23

>32
Array: {-72, -3, 3, 23, 32, 33, 423, 4234}      | Median: 27.5

>end
PS C:\Users\thari\UoM-DSA-S2-Labs\Week5\Q2>
```

Github repo: [Tharindu6516/UoM-DSA-S2-Labs: Lab tika (github.com)]