Data Structures Lab

Lab Task 1

Rayan Ahmed

23i-0018

Code:

```cpp
#include <iostream>
#include <chrono>
#include <fstream>

using namespace std;
using namespace std::chrono;

int findMax(int arr[], int size)
{
    int max = arr[0];
    for (int i = 1; i < size; i++)
        if (arr[i] > max)
            max = arr[i];

    return max;
}

void selectionSort(int arr[], int size)
{
    for (int i = 0; i < size - 1; i++)
    {
        int minIndex = i;
        for (int j = i + 1; j < size; j++)
            if (arr[j] < arr[minIndex])
                minIndex = j;

        int temp = arr[i];
        arr[i] = arr[minIndex];
        arr[minIndex] = temp;
    }
}

int countTriplets(int arr[], int size)
{
    int count = 0;

    for (int i = 0; i < size; i++)
        for (int j = i + 1; j < size; j++)
            for (int k = j + 1; k < size; k++)
                count++;

    return count;
}

void merge(int arr[], int left, int mid, int right)
{
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int arrL[n1];
    int arrR[n2];

    for (int i = 0; i < n1; i++)
        arrL[i] = arr[left + i];
    for (int i = 0; i < n2; i++)
        arrR[i] = arr[mid + 1 + i];

    int i = 0;
    int j = 0;
    int k = left;

    while (i < n1 && j < n2)
    {
        if (arrL[i] <= arrR[j])
        {
            arr[k] = arrL[i];
            i++;
        }
        else
        {
            arr[k] = arrR[j];
            j++;
        }
        k++;
    }
```

```c
    while (i < n1)
    {
        arr[k] = arrL[i];
        i++;
        k++;
    }

    while (j < n2)
    {
        arr[k] = arrR[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[], int left, int right)
{
    if (left < right)
    {
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

int binarySearch(int arr[], int n, int x)
{
    int left = 0;
    int right = n - 1;

    while (left <= right)
    {
        int mid = left = (right - left) / 2;

        if (arr[mid] == x)
            return mid;
        else if (arr[mid] < x)
            left = mid + 1;
        else
            right = mid - 1;
    }

    return -1;
}

int main()
{
```

```cpp
    fstream file;
    file.open("data_100.txt", ios::in);

    int size;
    file >> size;

    int *arr = new int[size];
    for (int i = 0; i < size; i++)
    {
        file >> arr[i];
    }

    file.close();

    for(int i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;

    auto start = high_resolution_clock::now();
    int max = findMax(arr, size);
    auto stop = high_resolution_clock::now();
    auto findMaxDuration = duration_cast<milliseconds>(stop - start);
    cout << "Max value in the array is: " << max << endl;

    start = high_resolution_clock::now();
    selectionSort(arr, size);
    stop = high_resolution_clock::now();
    auto selectionSortDuration = duration_cast<milliseconds>(stop -
start);

    cout << "The array after sorting is: " << endl;
    for(int i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;

    start = high_resolution_clock::now();
    int count = countTriplets(arr, size);
    stop = high_resolution_clock::now();
    auto countTripletsDuration = duration_cast<milliseconds>(stop -
start);
    cout << "Number of triplets in the array is: " << count << endl;

    start = high_resolution_clock::now();
    mergeSort(arr, 0, size - 1);
    stop = high_resolution_clock::now();
    auto mergeSortDuration = duration_cast<milliseconds>(stop - start);

    cout << "The array after sorting is: " << endl;
    for (int i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;

    start = high_resolution_clock::now();
    int num = binarySearch(arr, size, 1096);
    stop = high_resolution_clock::now();
    auto binarySearchDuration = duration_cast<milliseconds>(stop -
start);
    if(num == -1)
        cout << "Number not found" << endl;
    else
        cout << "Number found at index " << num << endl;

    ofstream result("result.txt");
    fstream resultFile;
    resultFile.open("result.txt", ios::app);

    resultFile << "Size O(logn) O(n) O(nlogn) O(n^2) O(n^3)\n";
    resultFile << size << " " << binarySearchDuration.count() << " " <<
findMaxDuration.count() << " " << mergeSortDuration.count() << " " <<
selectionSortDuration.count() << " " << countTripletsDuration.count()
<< endl;
    delete[] arr;

    file.open("data_1000.txt", ios::in);

    size;
    file >> size;

    arr = new int[size];
    for (int i = 0; i < size; i++)
    {
        file >> arr[i];
    }

    file.close();

    start = high_resolution_clock::now();
    max = findMax(arr, size);
    stop = high_resolution_clock::now();
    findMaxDuration = duration_cast<milliseconds>(stop - start);

    start = high_resolution_clock::now();
    selectionSort(arr, size);
    stop = high_resolution_clock::now();
    selectionSortDuration = duration_cast<milliseconds>(stop - start);
    start = high_resolution_clock::now();
    count = countTriplets(arr, size);
    stop = high_resolution_clock::now();
    countTripletsDuration = duration_cast<milliseconds>(stop - start);

    start = high_resolution_clock::now();
    mergeSort(arr, 0, size - 1);
    stop = high_resolution_clock::now();
    mergeSortDuration = duration_cast<milliseconds>(stop - start);

    start = high_resolution_clock::now();
    num = binarySearch(arr, size, 1096);
    stop = high_resolution_clock::now();
    binarySearchDuration = duration_cast<milliseconds>(stop - start);

    resultFile << size << " " << binarySearchDuration.count() << " " <<
findMaxDuration.count() << " " << mergeSortDuration.count() << " " <<
selectionSortDuration.count() << " " << countTripletsDuration.count()
<< endl;
    delete[] arr;

    file.open("data_5000.txt", ios::in);

    size;
    file >> size;

    arr = new int[size];
    for (int i = 0; i < size; i++)
    {
        file >> arr[i];
    }

    file.close();

    start = high_resolution_clock::now();
    max = findMax(arr, size);
    stop = high_resolution_clock::now();
    findMaxDuration = duration_cast<milliseconds>(stop - start);

    start = high_resolution_clock::now();
    selectionSort(arr, size);
    stop = high_resolution_clock::now();
    selectionSortDuration = duration_cast<milliseconds>(stop - start);

    start = high_resolution_clock::now();
    count = countTriplets(arr, size);
    stop = high_resolution_clock::now();
    countTripletsDuration = duration_cast<milliseconds>(stop - start);
    start = high_resolution_clock::now();
    mergeSort(arr, 0, size - 1);
    stop = high_resolution_clock::now();
    mergeSortDuration = duration_cast<milliseconds>(stop - start);

    start = high_resolution_clock::now();
    num = binarySearch(arr, size, 1096);
    stop = high_resolution_clock::now();
    binarySearchDuration = duration_cast<milliseconds>(stop - start);

    resultFile << size << " " << binarySearchDuration.count() << " " <<
findMaxDuration.count() << " " << mergeSortDuration.count() << " " <<
selectionSortDuration.count() << " " << countTripletsDuration.count()
<< endl;

    delete[] arr;
    file.open("data_10000.txt", ios::in);

    size;
    file >> size;

    arr = new int[size];
    for (int i = 0; i < size; i++)
    {
        file >> arr[i];
    }

    file.close();

    start = high_resolution_clock::now();
    max = findMax(arr, size);
    stop = high_resolution_clock::now();
    findMaxDuration = duration_cast<milliseconds>(stop - start);

    start = high_resolution_clock::now();
    selectionSort(arr, size);
    stop = high_resolution_clock::now();
    selectionSortDuration = duration_cast<milliseconds>(stop - start);

    start = high_resolution_clock::now();
    count = countTriplets(arr, size);
    stop = high_resolution_clock::now();
    countTripletsDuration = duration_cast<milliseconds>(stop - start);

    start = high_resolution_clock::now();
    mergeSort(arr, 0, size - 1);
    stop = high_resolution_clock::now();
    mergeSortDuration = duration_cast<milliseconds>(stop - start);

    start = high_resolution_clock::now();
    num = binarySearch(arr, size, 1096);
    stop = high_resolution_clock::now();
    binarySearchDuration = duration_cast<milliseconds>(stop - start);

    resultFile << size << " " << binarySearchDuration.count() << " " <<
findMaxDuration.count() << " " << mergeSortDuration.count() << " " <<
selectionSortDuration.count() << " " << countTripletsDuration.count()
<< endl;
    delete[] arr;

    return 0;
}
```

Output:

Original Array:

16766 449 5982 15432 8571 9854 10320 9271 20638 16000 24905 8849 21760 16717 32324 23479 30858 27645 14221 6165 19976 505 3957 11012 3199 2819 10314 16623 13505 12919 13958 22709 13630 11216 25415 27727 29 16493 13055 3251 18316 27286 17517 29944 21588 8634 31692 22714 10928 15354 28234 29120 29023 2284 8635 4316 20216 2456 29839 13843 12670 27079 16897 29694 13364 7806 5778 8816 6066 19909 2230 9306 57 25516 2509 17643 28228 30864 17236 28667 9567 27803 21597 18857 30012 23690 828 26947 6404 21427 14212 15207 8427 24390 20530 20328 6663 26323 10791 5107

Max value in the array is: 32324

The array after sorting is:

29 57 449 505 828 2230 2284 2456 2509 2819 3199 3251 3957 4316 5107 5778 5982 6066 6165 6404 6663 7806 8427 8571 8634 8635 8816 8849 9271 9306 9567 9854 10314 10320 10791 10928 11012 11216 12670 12919 13055 13364 13505 13630 13843 13958 14212 14221 15207 15354 15432 16000 16493 16623 16717 16766 16897 17236 17517 17643 18316 18857 19909 19976 20216 20328 20530 20638 21427 21588 21597 21760 22709 22714 23479 23690 24390 24905

```
25415 25516 26323 26947 27079 27286 27645 27727 27803 28228 28234 28667
29023 29120 29694 29839 29944 30012 30858 30864 31692 32324


Number of triplets in the array is: 161700


The array after sorting is:

29 57 449 505 828 2230 2284 2456 2509 2819 3199 3251 3957 4316 5107 5778
5982 6066 6165 6404 6663 7806 8427 8571 8634 8635 8816 8849 9271 9306
9567 9854 10314 10320 10791 10928 11012 11216 12670 12919 13055 13364
13505 13630 13843 13958 14212 14221 15207 15354 15432 16000 16493 16623
16717 16766 16897 17236 17517 17643 18316 18857 19909 19976 20216 20328
20530 20638 21427 21588 21597 21760 22709 22714 23479 23690 24390 24905
25415 25516 26323 26947 27079 27286 27645 27727 27803 28228 28234 28667
29023 29120 29694 29839 29944 30012 30858 30864 31692 32324


Number not found
```

Text File:

Size O(logn) O(n) O(nlogn) O(n^2) O(n^3)

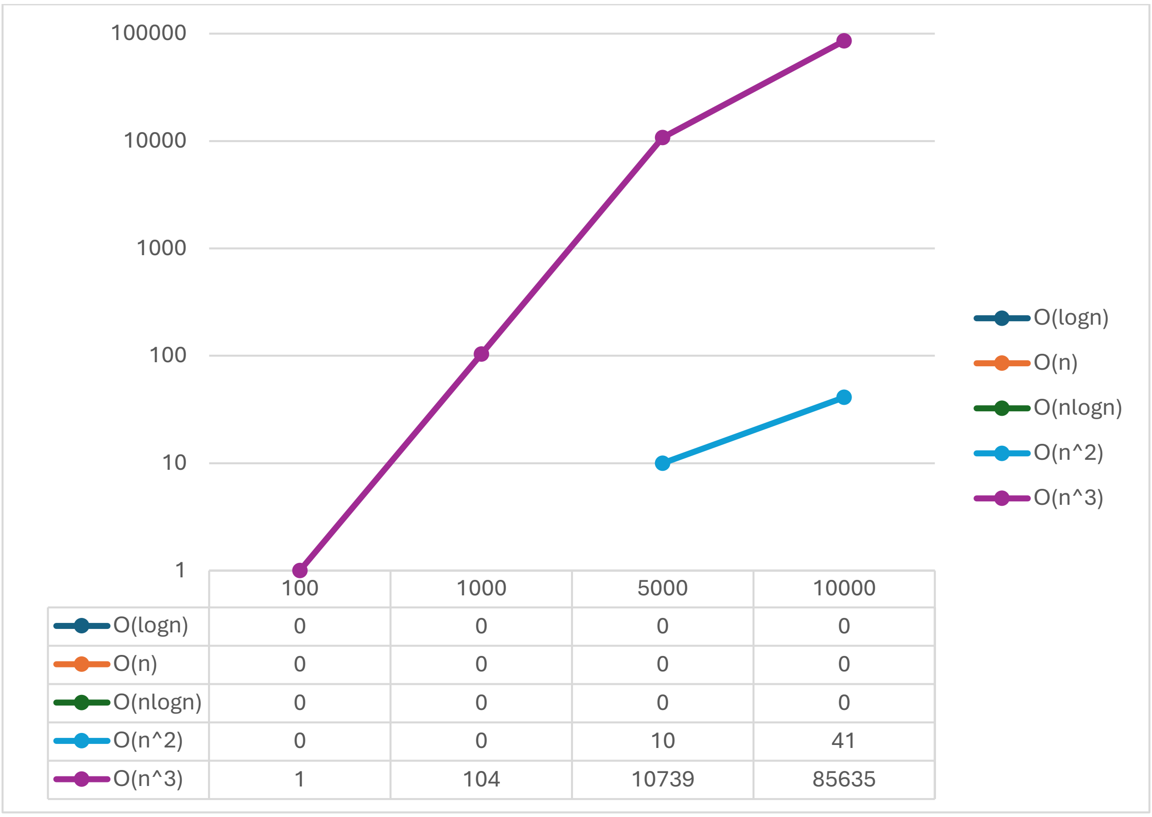
100 0 0 0 0 0

1000 0 0 0 1 73

5000 0 0 0 10 7946

10000 0 0 0 37 62756

Graph:



	100	1000	5000	10000
O(logn)	0	0	0	0
O(n)	0	0	0	0
O(nlogn)	0	0	0	0
O(n^2)	0	0	10	41
O(n^3)	1	104	10739	85635