Sardar Vallabhbhai National Institute of Technology, Surat

Subject: DESIGN AND ANALYSIS OF ALGORITHM.

- DAA Assignment-1.
- Name: SURU MANOJ
- Roll No.: B101
- Admission No.: U20CS101

1.1. Implement the above algorithms using the programming language of your choice.

I chose C++ to implement the above algorithms.

1.2. Provide the details of Hardware/Software you used to implement algorithms and to measure the time.

Hardware details is provided below and the ide I used is Visual Studio.

Device name	LAPTOP-LDLFM5HU
Processor	Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz 2.50 GHz
Installed RAM	8.00 GB (7.87 GB usable)
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Edition	Windows 11 Home Single Language
Version	21H2
Installed on	22-10-2021
OS build	22000.434
Serial number	PF1V7WF3
Experience	Windows Feature Experience Pack 1000.22000.434.0

1.3. Submit the code (complete programs).

Linear Search:

```
#include <bits/stdc++.h>
using namespace std;
#include <time.h>
```

```
void linearSearch(vector<int> arr, int n, int s)
{
    for (int i = 0; i < n; i++)
    {
        if (arr[i] == s)
        {
            cout << "The element found at the index : " << i << endl;
            return;
        }
    }
    cout << "The element is not found.." << endl;
}

int main()
{
    string filename("File6.txt");
    vector<int> values;
    int number;
    ifstream input_file(filename);
    if (!input_file.is_open())
    {
        cerr << "Could not open the file - '"
             << filename << "'" << endl;
        return EXIT_FAILURE;
    }
    int n = 0;

    while (input_file >> number)
    {
        values.push_back(number);
        n++;
    }
    cout << n << endl;
    //sort(values.begin(), values.end(), greater<int>());
    sort(values.begin(), values.end());
    /*int s;
    cout << "Enter the element to be searched : " << endl;
    cin >> s;*/

    clock_t t1Start = clock();
    linearSearch(values, n, values[(n / 2) - 1]);
    cout.precision(10000);
    cout << "Time taken by function: " << (double)(clock() - t1Start) /
CLOCKS_PER_SEC << " microseconds" << endl;

    /*clock_t t2Start = clock();
    linearSearch(values, n, values[n - 1]);
```

```
    cout.precision(10000);
    cout << "Time taken by function: " << (double)(clock() - t2Start) /
CLOCKS_PER_SEC << " microseconds" << endl;*/

    /*clock_t tStart = clock();
    linearSearch(values, n, values[0]);
    cout.precision(10000);
    cout << "Time taken by function: " << (double)(clock() - tStart) /
CLOCKS_PER_SEC << " microseconds" << endl;*/

    input_file.close();

    return EXIT_SUCCESS;
}
```

Bubble Sort:

```
#include <bits/stdc++.h>
using namespace std;
#include <time.h>

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void bubbleSort(vector<int> &values, int n)
{
    //bool swapped;
    for (int i = 0; i < n - 1; i++)
    {
        //swapped = false;
        for (int j = 0; j < n - i - 1; j++)
        {
            if (values[j] > values[j + 1])
            {
                swap(&values[j], &values[j + 1]);
                //swapped = true;
            }
        }
        //if (!swapped)
        //break;
    }
}

int main()
```

```
{
    string filename("File5.txt");
    vector<int> values;
    int number;
    ifstream input_file(filename);
    if (!input_file.is_open())
    {
        cerr << "Could not open the file - '"
             << filename << "'" << endl;
        return EXIT_FAILURE;
    }
    int n = 0;

    while (input_file >> number)
    {
        values.push_back(number);
        n++;
    }
    cout << n << endl;
    sort(values.begin(), values.end(), greater<int>());
    //sort(values.begin(), values.end());
    clock_t tStart = clock();
    bubbleSort(values, n);
    cout.precision(10);
    cout << "Time taken by function: " << (double)(clock() - tStart) /
CLOCKS_PER_SEC << " microseconds" << endl;
    cout << "The sorted array is : " << endl;
    /*for (int i = 0; i < n; i++)
    {
        cout << values[i] << " ";
    }*/
    cout << endl;
    input_file.close();

    return EXIT_SUCCESS;
}
```

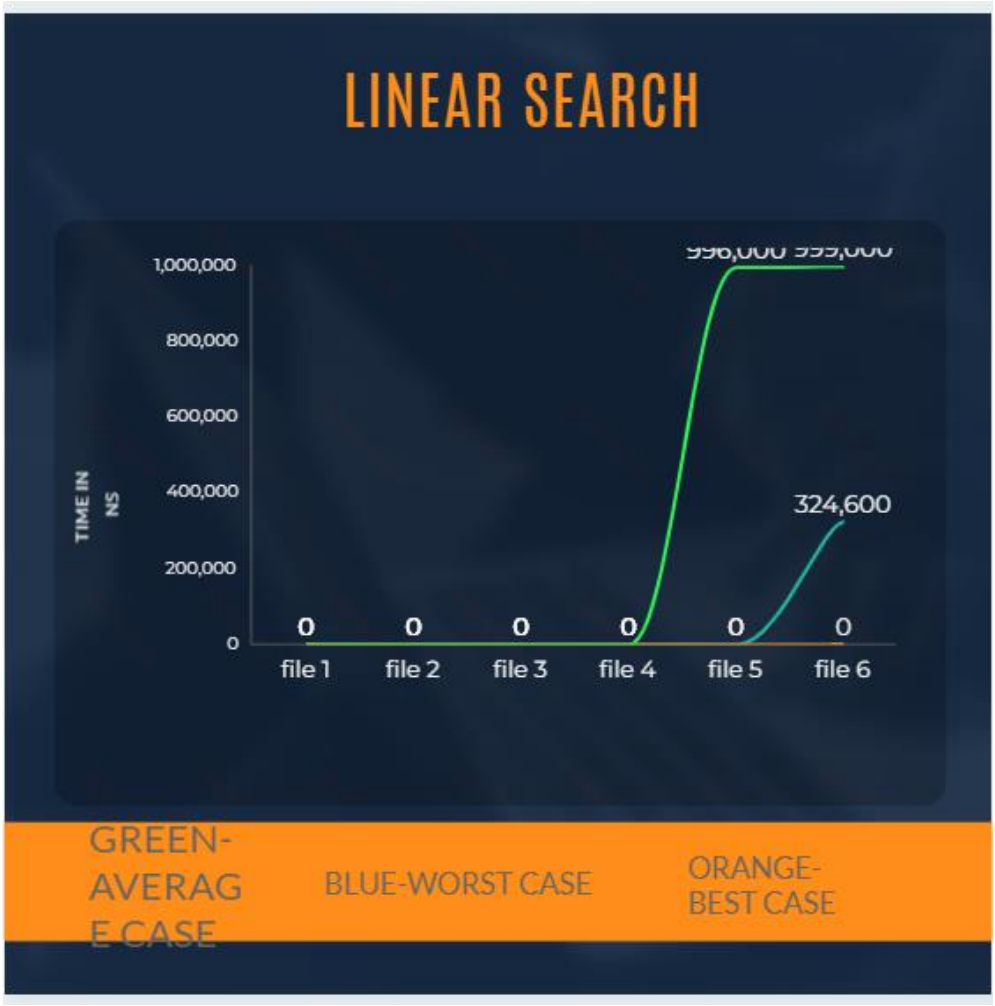
Selection Sort:

```
#include <bits/stdc++.h>
using namespace std;
#include <time.h>

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
```

```
}

void selectionSort(vector<int> &values, int n)
{
    int i, j, mini;
    for (int i = 0; i < n - 1; i++)
    {
        mini = i;
        for (int j = i + 1; j < n; j++)
        {
            if (values[j] < values[mini])
            {
                mini = j;
            }
        }
        swap(&values[i], &values[mini]);
    }
}

int main()
{
    string filename("File5.txt");
    vector<int> values;
    int number;
    ifstream input_file(filename);
    if (!input_file.is_open())
    {
        cerr << "Could not open the file - '"
             << filename << "'" << endl;
        return EXIT_FAILURE;
    }
    int n = 0;

    while (input_file >> number)
    {
        values.push_back(number);
        n++;
    }
    cout << n << endl;
    sort(values.begin(), values.end(), greater<int>());
    //sort(values.begin(), values.end());
    clock_t tStart = clock();
    selectionSort(values, n);
    cout.precision(10);
    cout << "Time taken by function: " << (double)(clock() - tStart) /
CLOCKS_PER_SEC << " seconds" << endl;
    cout << "The sorted array is : " << endl;
    /*for (int i = 0; i < n; i++)
```

```
    {
        cout << values[i] << " ";
    }*/
    cout << endl;
    input_file.close();

    return EXIT_SUCCESS;
}
```

1.4. Measure the best-case time and worst-case time of linear search for all six files. Plot a graph.



1.5. Measure the average-case time (considering current data of six files) of bubble sort, and selection sort for all six files. Plot a graph.

1.6. Measure the best-case time of bubble sort, and selection sort for all six files. Plot a graph.

1.7. Measure the worst-case time of bubble sort, and selection sort for all six files. Plot a graph.


SELECTION VS BUBBLE
350
300
250
200
150
100
50
0
SECONDS
323
214
33
11
0
0
2
1
FILE1
FILE2
FILE3
FILE4
FILE5
FILE6
FILE SIZE
SELECTION - ORANGE
WORST
BUBBLE - GREEN