Assignment-1 (DAA)

* Name: KRISHNA PANDEY
* Roll No.: B110
* Admission No.: U20CS110
  1. Implement the above algorithms using the programming language of your choice.

I chose C++ to implement the above algorithms.

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

Hardware details is provided below and the IDE used by me is Visual Studio.

Device name LAPTOP-LEHQUDIU

Processor 11th Gen Intel(R) Core (TM) i5-1135G7 @ 2.40GHz 2.42 GHz

Installed RAM 8.00 GB (7.75 GB usable)

Device ID 2C14C1C1-1420-41A4-B7F5-0D78ADA245F8

Product ID 00327-36277-89595-AAOEM

System type 64-bit operating system, x64-based processor

Pen and touch No pen or touch input is available for this display

Edition Windows 10 Home Single Language

Version 21H1

Installed on ‎29-‎07-‎2021

OS build 19043.1526

Experience Windows Feature Experience Pack 120.2212.4170.0

* 1. 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] << " ";

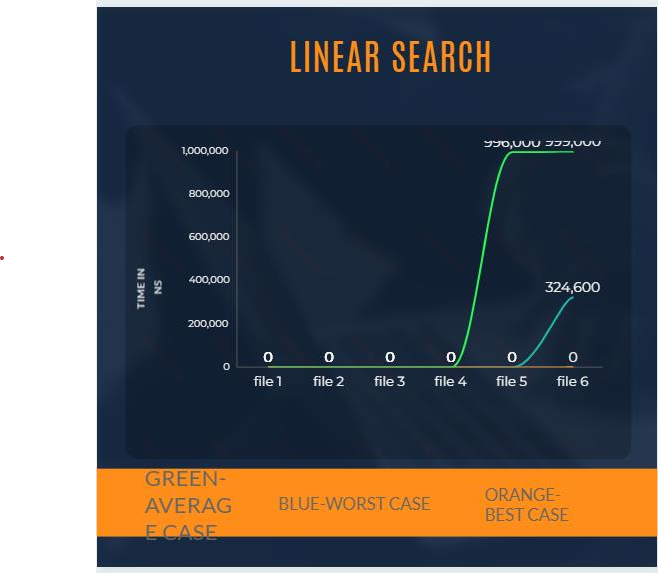
}\*/

cout << endl; input\_file.close();

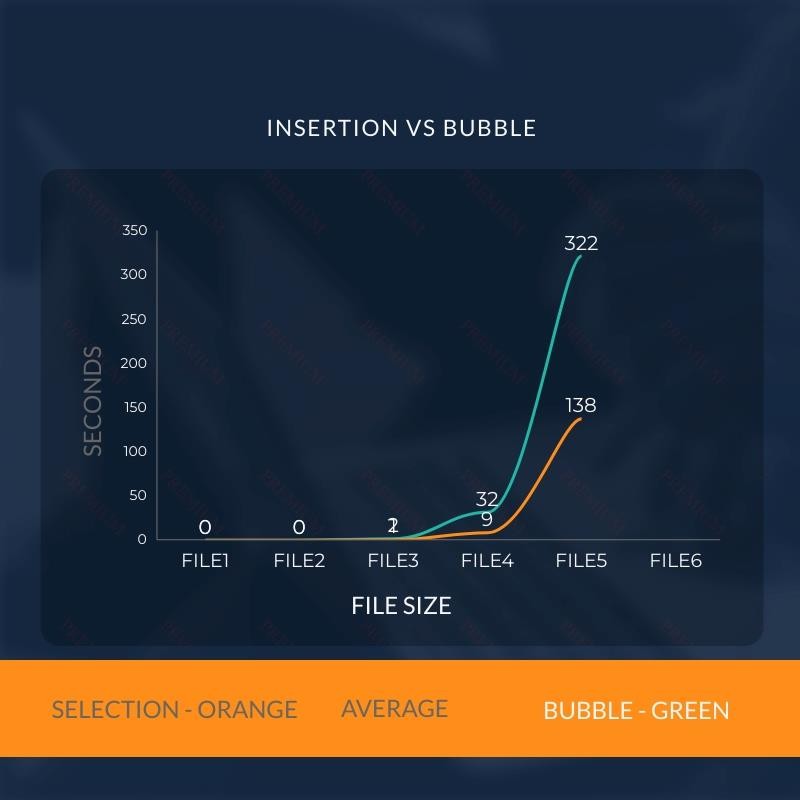
return EXIT\_SUCCESS;

}

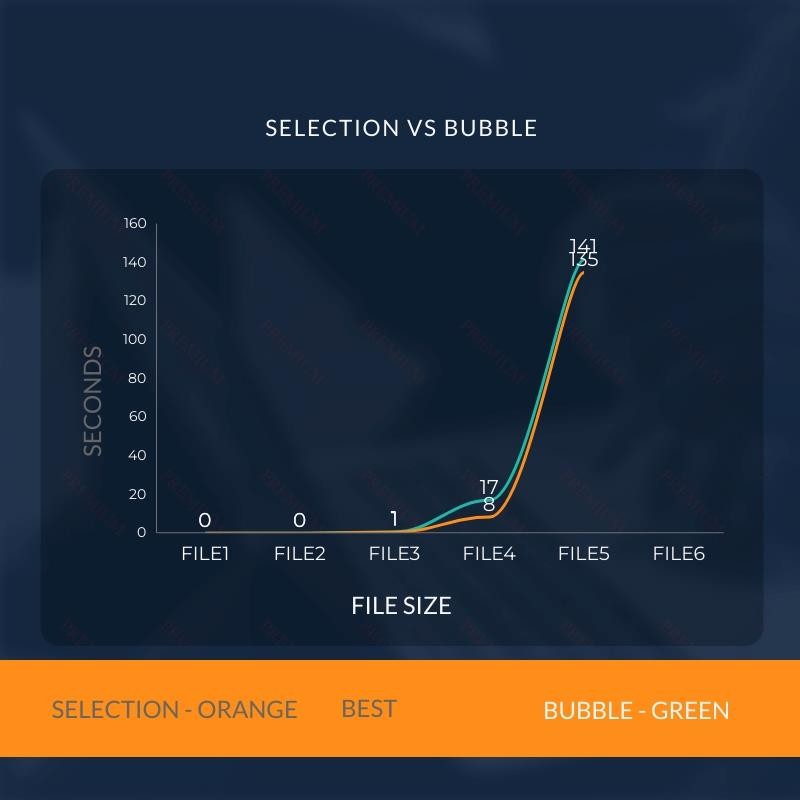
* 1. Measure the best-case time and worst-case time of linear search for all six files. Plot a graph.



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



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

