Lab Assignment: Double Linear Search Analysis

In this assignment, we are going to create an improved version of the linear search algorithm and analyze its performance. Please study the following problem description and prepare your solution for the question.

Purpose:

Purpose of the assignment is to help the students understand how to write algorithm pseudo code. Further, the students would be able to practice evaluating the performance of the algorithm by determining the Big O. In order to verify the performance, the students would create simulation and verify with the obtained simulation results with the Big O. This would enhance their understanding of the importance of efficient code writing in their programming.

Requirements of the Assignment

The linear search algorithm accepts a vector variable and a search value. In turn, the algorithm must return the first two index positions at which the search value were found. If two search values are not found then the algorithm must return a sentinel value (e.g., -1).

The improved version of the algorithm must return a vector variable that will contain the first two index positions, where the search value has been found. For example, in case we have the following

```
vector<int> array {10, 50, 16, 1, 9, 15, 16, 20, 16, 2, 5};
int searchValue = 16;
vector<int> searchResult = linearSearch(array, searchValue);
```

Then in the searchResult variable, we should have the two index positions, {2, 6}; The algorithm will stop the search operation when two instances of the search value has been found.

To clarify, let us consider another scenario, where two instances of the searchValue would not be found. Please review the following example,

```
vector<int> array {10, 50, 16, 1, 9, 15, 16, 20, 16, 2, 5};
int searchValue = 10;
vector<int> searchResult = linearSearch(array, searchValue);
```

Then in the searchResult variable, we should have only the sentinel value {-1}. In order to reach to this NOT found decision, the algorithm will have to loop through all the values of the array.

Deliverable 1: Write down the steps by using an algorithm. Define the steps by using pseudo code instead of using programming code.

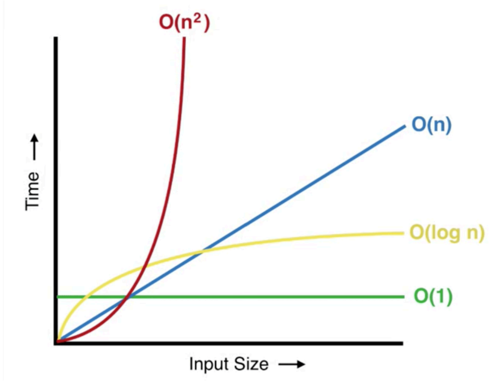
Deliverable 2: Convert the algorithm into a function and use the function (as shown above) in the main function. Pass necessary parameters and verify the accuracy of your program. Name your program, doubleLinearSearch.cpp and attach this program with your submission. Attach a screenshot depicting a sample run of the program

Deliverable 3: We know, Big O is an abstract function that describes how fast the cost of a function increases as the size of the problem becomes large. The order given by Big O is a least upper bound on the rate of growth.

We say that a function T(n) has order O(f(n)) if there exist positive constants c and n_0 such that:
$T(n) \leq c * f(n)$ when $n \geq n_0$.

Calculate the Big O of your algorithm by summing up the Big O of the individual statements used in your program. Find the equation T(n) of the program and furthermore, find the equation f(n), c, and n_0 in order to determine the Big O of your algorithm.

Deliverable 4: Modify the newly created linear search algorithm in order to manually count the number of steps (refer to the program provided in the content section on the Blackboard course website titled, "Algorithm analysis through simulation: a complete example") it takes to compute the searchResult. Name this program, simulDoubleLinearSearch.cpp and attach this program with your submission. Attach a screenshot depicting a sample simulation run of the program.



Use different input size as specified below and note down the average number of steps of your algorithm for each input size. Use the following table and report those numbers from your simulation.

Assign N = 10K, then N= 20K, N= 35K, N = 50K, N= 75K, and N =100K for each subsequent simulation. Here N is the number of elements in the vector variable.

For each value of N, the simulation should iterate 1000 number of times. In each iteration of the simulation, generate, N random values and store them in the vector variable. The random values should range between 0 and 15,000.

Input size	Hits	Misses	Min steps	Average steps
10K				
20K				
35K				
50K				
75K				
100K				

Deliverable 5: By using the tabular values, plot a graph similar to the graph shown above (you can use steps * 1ms to approximate the time required to run the algorithm). Compare your graph with the Big O plot that matches closely to your graph.

Use a document to include all the deliverables (**do not include the source code of the program in the document, rather attach them separately**) and export the document as a pdf file.

Please let me know if you have any questions. Thank you.

Rubrics
The Rubrics of the Assignment is given in the following:

Assignment rubrics | Total points: 30	Points
Deliverable 1: add this in the report	4 points
Deliverable 2: doubleLinearSearch.cpp, screenshot	6 points
Deliverable 3: add this in the report	5 points
Deliverable 4: simulDoubleLinearSearch.cpp, screenshot, table data for the provided values of N, add the values in the report	10 points
Deliverable 5: add your analysis in the report	5 points
Miscellaneous: Javadoc comments for all the methods Syntax error in the program (program does not run) Missing screenshot Missing chart and analysis of the data The report is not in pdf format	 -5 points -30 points -10 points -5 points -5 points