University of Colombo School of Computing
SCS 1308 - Foundations of Algorithms
Take-home 05

Instructions

- Try the following questions and upload your code files and answer script as a zip file to the given link in the UGVLE on/before 10th January at 6 pm.
- Note: Rename your zip file with your index number and name. (i.e: indexNo_Name.zip).

01. Write a C program that compares the linear search algorithm to the binary search algorithm

Initialize an array named "numbers" with 50 integers using this code:

```
int numbers[] = {888, 552, 642, 191, 935, 584, 959, 495, 945, 27, 240, 637, 712, 608, 924, 631, 712, 234, 275, 258, 918, 15, 320, 144, 365, 305, 696, 783, 114, 434, 926, 467, 268, 989, 310, 721, 282, 73, 334, 115, 662, 798, 970, 433, 758, 213, 284, 154, 126, 543 };
```

The program should:

1. Display the array and ask the user to choose one of the numbers,
2. Call a function named linearSearch that uses the linear search algorithm to locate the user's choice,
3. Call a function named binarySearch that uses the binary search algorithm to locate the same value again,
4. Display the number of comparisons made by each function's attempt to find the value.
5. If the number is not found then display this message: "Sorry, that number is not in the array".

IMPORTANT: Each function should track, and then return, the array index of the target as well as the number of comparisons that it took to find the value. If the value is not found, then the function should return -1.

02. Write a C program that compares the performance of linear search and binary search for multiple values at once.

A. In main(), initialize an array numbers with 100 random integers between 1 and 1000.

B. The program should:

   a. Display the array to the user.
   b. Ask the user to input five different target numbers they want to search for.

C. Write a function named *linearSearch*:

   a. Perform a linear search to locate each target.
   b. Track and return the number of comparisons made for each target.
   c. Store the results (index and comparisons) for all five numbers.

D. Write a function named *binarySearch*:

   a. Sort the array (use any sorting method you like).
   b. Perform a binary search to locate each target.
   c. Track and return the number of comparisons made for each target.
   d. Store the results (index and comparisons) for all five numbers.

E. The program should:

   a. Display the indices and number of comparisons for each target number for both algorithms.
   b. If a number is not found, display: "Target X not found in the array."
   c. Calculate and display the average number of comparisons across the five targets for both algorithms.

F. Include a function that automatically calculates and displays each algorithm's time complexity (Big O) based on the array's number of elements.