Lab:

Searching algorithms

Prelab Task:

1. Fix yourLab_01 (if you did something wrong.) Make sure your deSelsort function work correctly (since it will be used in this lab.)

2. Please review the following searching algorithms you have learned in COSC 120 & 220. Make sure you understand the algorithms, their implementation, and worst-case time complexity of each algorithm.

- Linear search
- Binary search

Lab Task:

In this lab, we will experiment the worst-case time complexity of binary search by finding the maximum number of comparisons needed for searching values in an array.

- For a random list of integers, what is the maximum number of comparisons required to find a target value by binary search? Please elaborate your answer.

- Declare and define the function binSearch in a file named **binSearch.h**. Your function binSearch should be able to record the total number of comparisons the algorithm executes for an unsuccessful search
- Write a program in a file named **lab02.cpp**
    - The program declares an integer array with ARRSIZE (=10000) random integers from 0 to RANDOMLIMIT (=99999).
    - The program performs a total of RANDOMVALUES (=10000) times binary search. During each search, a random target value (from 0 to RANDOMLIMIT) will be generated to perform search by calling the function of binSearch. [Note: The deSelsort function you create in lab 1 will be use to sort the numbers before you perform binary search.]
    - The integer **sumFailCom** will be used to record the total numbers of comparison in all unsuccessful search after the total of RANDOMVALUES times to run binary search.
    - The integer **sumSucCom** will be used to record the total numbers of comparison in all successful search after the total of RANDOMVALUES times to run binary search.
    - The integer **successTotal** will be used to record the total number of successful search during the total of RANDOMVALUES times to perform binary search.
    - The program will output the empirical result for the worst-case comparison used for an unsuccessful binary search, which can be calculated by

      sumFailCom/(RANDOMVALUES-successTotal)

- ◦ The program will output the empirical result for the worst-case comparison used for a successful binary search, which can be calculated

  sumSucCom/successTotal

- Run your program and print out the outputs.
- Does your empirical results verify your answer for the maximum number of comparisons required to find a target value by the binary search?

What to Turn In

- Hand in your printouts.
  - **binSearch.h**
  - **lab02.cpp**
  - Sample outputs
  - Answers to questions