Computer Laboratory 7
CSCI 1913: Introduction to Algorithms,
Data Structures, and Program Development
October 22–23, 2019

0. Introduction.

Two algorithms for searching arrays, called *linear search* and *binary search,* were discussed in the lectures. The linear search algorithm did only one comparison per iteration, but it needed many iterations. The binary search algorithm did more than one comparison per iteration, but it needed few iterations. As a result, linear search should be more efficient than binary search for small arrays, and binary search should be more efficient than linear search for large arrays. How big must an array be before binary search becomes more efficient than linear search? You will determine this experimentally.

1. Implementation.

Here's what you must do for this laboratory exercise. **First,** write a method called `linearSearch` that takes an integer called `key` and an integer array called `keys` as its arguments. Your method must use linear search to find `key` in `keys`. It must then return the number of comparisons that were needed to find it.

Second, write a method called `binarySearch` that takes an integer called `key` and a sorted integer array called `keys` as its arguments. Your method must use binary search to find `key` in `keys`. It must then return the number of comparisons that were needed to find it.

Unlike the methods discussed in the lectures, your `linearSearch` and `binarySearch` methods must return the number of comparisons they use to find `key` in `keys`. This is not necessarily the same as returning the index of `key` in `keys`! Here are two things to know about counting comparisons.

- Count *only* comparisons made between `key` and elements of `keys`. Your methods may make other comparisons, but *do not* count those, because they are not relevant to this experiment.
- Count these comparisons whether or not they succeed. It doesn't matter if a comparison is true or false, you must still count it.

To write these methods, you can copy the linear search and binary search algorithms from the lectures, and then modify them. You can also copy algorithms from the textbook, or from the Internet, or from any other source. You can even write your own algorithms, but this is not recommended for binary search, because it is harder to get right than it looks.

Third, place your methods in the driver class **`BinaryVsLinear`,** as shown below. (Source code for this class is available on Canvas.) The driver class's `main` method constructs thirty arrays, each with sorted elements, and computes the average number of comparisons required to find an element in each array.

```
class BinaryVsLinear
{
  private static int linearSearch(int key, int [] keys)
  {
    :
  }

  private static int binarySearch(int key, int [] keys)
  {
    :
  }

  public static void main(String [] args)
  {
    for (int length = 1; length <= 30; length += 1)
    {
      int[] array = new int[length];
      for (int index = 0; index < length; index += 1)
      {
        array[index] = index;
      }

      double linearTotal = 0.0;
      double binaryTotal = 0.0;
      for (int element = 0; element < length; element += 1)
      {
        linearTotal += linearSearch(element, array);
        binaryTotal += binarySearch(element, array);
      }

      double linearAverage = linearTotal / length;
      double binaryAverage = binaryTotal / length;
      System.out.println(length + " " + linearAverage + " " + binaryAverage);
    }
```

```
        }
    }
```

Fourth, run the class `BinaryVsLinear`. You should obtain a table with three columns of numbers. The first column is an array size (from 1 to 30). The second column is the average number of comparisons required by `linearSearch` to find an element in an array of that size. The third column is the average number of comparisons required by `binarySearch` to find an element in an array of that size.

Fifth, draw a graph using the numbers in the table. You can draw it by hand, or you can use a program, like a spreadsheet. The *x*-axis of the graph must be the array size. The *y*-axis must be the average number of comparisons required to find an element in an array of that size. Draw two curves in the graph: one for linear search, and the other for binary search.

2. Deliverables.

You must submit all of the following, for a total of 30 possible points.

1. Your version of the class `BinaryVsLinear`, containing definitions for the methods `linearSearch` (5 points) and `binarySearch` (5 points).
2. The output generated by the `main` method of `BinaryVsLinear` (5 points).
3. A graph of the output (5 points).
4. A *short* written answer to this question: Based on your graph, for what array sizes is linear search more efficient than binary search? (5 points).
5. A *short* written answer to this question: Based on your graph, for what array sizes is binary search more efficient than linear search? (5 points).

If you do not know how to submit your work, then please ask your lab TA's. (It may be more complicated than usual because you are submitting both a file of Java code and a graph.) Your work must be turned in by **11:55 PM** on **Wednesday, October 30, 2019**.