Lab 7: Searching

Steven K. Andrianoff
Robert Harlan
David Levine
Computer Science Department
St. Bonaventure University

Objective:

This lab examines the linear (sequential) and binary search algorithms. In addition to testing the algorithms on both ordered and unordered data you will learn how to time the execution of code.

Instructions:

1. Create a Java project in Eclipse. Add the file `Dice.java` to your project. Create a main class named SearchingMain. Within the main method write code to generate 1000 random integer values in the range 1 to 1000. (Create a 1000-sided Dice to generate the numbers.) Store the values in an array of 1000 Integers (*Integer*, not *int*).

Add statements to display the first value in the array along with its index. Do the same for the last value in the array and the middle value in the array. Be sure to label the values and indices. Run the program and record the results.

2. Write a static method in your main class to perform a *linear search* of the array. The signature of the method should be:

```
private static int linearSearch(Object[] list, Object key)
```

Code the method to search the array *list* for the object *key*. If the object is found return the index where it is found. If the object is not found return the constant value NOT_FOUND. The constant NOT_FOUND should be defined (private static final) in the class to be –1. Be sure to properly Javadoc the method.

Add statements to the main method to use the method linearSearch() to search the array of random integers for:

a) the first value in the array
b) the last value in the array
c) the middle value in the array
d) a value not in the array

In each case display the value you are searching for and the index that results from linearSearch. As in Step 1, display the values and their indices prior to calling linearSearch then display the results from linearSearch. Explain any discrepancies.

Run the program and record your results.

3. Write another static method in your main class to perform a *binary search* of the array. The signature of the method should be:

```
private static int binarySearch(Comparable[] list, Comparable key)
```

Code the method to search the array *list* for the object *key* using the binary search algorithm.

In your lab write-up explain why the parameters are Comparable rather than Object for this method.

Run the program using the same set of test data described in Step 2 above. Record your results. Does the program run correctly?

4. Modify the main method to generate the numbers 1 through 1000 and store them in the array (in that order) as Integers.

Run the program using the same set of test data described in Step 2 above. Record your results. Does the program run correctly now?

Document the two methods and document the file with Javadoc comments. Print a copy of your source code, SearchingMain.java, and hand it in with your lab write-up.

5. Now modify the main method to time the execution of linear search and binary search. First modify the main method to generate the numbers 1 through 10,000 and store them in an array in that order. Call both linearSearch and binarySearch to search for the value 10,001. Display results that show that the methods worked correctly.

Now we want to time the linearSearch and the binarySearch independently. If you want to time a block of code you can do the following:

```
long start;
long stop;
long total;
start = System.currentTimeMillis();
      <block of code to time>
stop = System.currentTimeMillis();
total = stop - start;
```

The value of total will be the number of milliseconds (1 msec. = 0.001 sec.) that elapsed between the two calls to `System.currentTimeMillis()`. Time the call to linearSearch (searching for 10,001) and time the call to binarySearch (searching for 10,001). Note: Time only the statement that calls the search method – not the print statement. Record your results.

Time the two calls again using 100,000 values (searching for 100,001) and then 1,000,000 values (searching for 1,000,001) and record your results.

6. The problem with the timings in the previous step is that the binary search is so fast that it is difficult to measure. What we will do is build a 1 msec. delay into each of the two loops: the for loop in the linear search method and the while loop in the binary search method. Place the following statement as the first statement in the body of each loop:

```
Thread.sleep(1);
```

This introduces a 1 msec. delay for each iteration of the loop. (Fix the syntax error by selecting "Surround with try/catch" from the suggested fixes.)

Time the two calls using 10,000 values (searching for 10,001) and then 100,000 (searching for 100,001) and record your results. DO NOT try 1,000,000 values!

7. To better understand the behavior of binary search and the resulting speed up compared to linear search, consider an array storing the values 1 through 1000 in that order. Identify the sequence of probes that would be made in searching for the value 487. How many probes are there? How many probes would be made if linear search were used to search for the same value in the same set of data? How do the number of probes made using binary search compare with the number of probes made using linear search?

Include your observations and results in your lab write-up.

Extra Credit: Modify your program by removing the Thread.sleep(1) statements from each method. Modify the code in each method to count the number of "probes". Hint: establish a class (static) variable to do the counting. Each of the sorting methods as well as the main method should be able to access this class variable. Run your modified program on the data given in Step 7 and record your results. Hand in a copy of your modified program.

Hand in:

The write-up you hand in for this lab should include:

- the results requested in Steps 1-7.
- the explanation requested in Step 2.
- the explanation and observation requested in Step 3.
- the observation requested in Step 4.
- the printed copy of the source code requested in Step 4.
- the observations and results requested in Step 7.
- the results and the code requested for Extra Credit.

Help Policy:

Help Policy in Effect for This Assignment: Group Project with Limited Collaboration

In particular, you may discuss the assignment and concepts related to the assignment with the following persons, in addition to an instructor in this course: any member of your group; any St. Bonaventure Computer Science instructor; and any student enrolled in CS 132.

You may use the following materials produced by other students: materials produced by members of your group.

You may use the following materials produced by other students: NONE.