Finding Execution Times of Linear and Binary Searches

Homework #1

By

Tahseen Robbani

CS 303 Algorithms and Structures

January 15, 2019

1. Problem Specification

The goal of this assignment was to write a program that included creating a file of 1000 randomly generated integers between 1 and 2^20, creating exponentially increasing arrays between sizes 2^3 and 2^20, searching through them with the file of integers using a linear search and binary search algorithm, and then comparing the execution times.

2. Program Design

This program required a Search class that contained the binary and linear search methods, and a driver which included the file creation, array creations, and use of the searches.

The following steps were required to create the entire implementation of the program:

a) Write the Search class which included creating algorithms for the linear search and the binary search methods.
b) Create the .txt file which included 1000 randomly generated integers within the Java program.
c) Create a BufferReader object and a Search object.
d) Loop through the powers of 2 from 3 to 20 in order to create an array for each size and inserting random integers between 1 and whatever the size of the array is.
e) Loop through the random.txt file and search through it using the linear search method and time the execution.
f) Do the same as e) but with the binary search algorithm.
g) Print the results.

The following methods were defined in the Search and LargeSearchTester classes:

a) linear(arr: int[], ans: int)
    a. Method that implements an algorithm for a linear search of a given array with a key and returns the index.
b) binary(arr: int[], ans: int)
    a. Method that returns a different binary method with specified beginning and end.
c) binary(arr: int[], ans: int, beg: int, end: int)
    a. Recursive method that implements an algorithm for a binary search given the beginning of the array and the end.
d) main(args: String[])
    a. Main driver method that creates a file, creates random arrays, and uses the linear and binary search methods to provide execution times for each.

Large use of BufferedWriter and BufferedReader were used to create the random.txt file and make use of it in the implementation of searching through the file.

3. Testing Plan

Since this program required no use of user input, testing needed to be analyzed from the sizes of the array. Admittedly, there is an error in my output that was unfound and had forced my smallest array size to take the most execution time with the linear search. My plan was to analyze the output when running the program and justify why whatever execution time is associated with the array size is present.

When testing the array of size 16, the time was inconsistent when compared to the other linear search times. All the other linear search times were under 1 ms, which makes a search time being above 1 ms seem out of the ordinary. For the binary search times, the times were larger near the end of the list of arrays due to the amount of time that was added due to sorting such a large number of integers. Many of the times that were tested seemed incorrect because it seemed to be *too* fast for a linear search of an array sized 10^20. However, this may be attributed to the search finding whatever the key was early in the search which makes it end early.

4. Test Cases

The output is posted below as a table to represent each case of array.

```
size: 16        linear time (ms): 4.652245      binary time (ms): 0.372939
size: 32        linear time (ms): 0.034765      binary time (ms): 0.025679
size: 64        linear time (ms): 0.018963      binary time (ms): 0.045432
size: 128       linear time (ms): 0.025284      binary time (ms): 0.069926
size: 256       linear time (ms): 0.013432      binary time (ms): 0.101926
size: 512       linear time (ms): 0.034371      binary time (ms): 0.190814
size: 1024      linear time (ms): 0.015407      binary time (ms): 0.397827
size: 2048      linear time (ms): 0.014222      binary time (ms): 0.860049
size: 4096      linear time (ms): 0.045827      binary time (ms): 0.698074
size: 8192      linear time (ms): 0.110222      binary time (ms): 1.105382
size: 16384     linear time (ms): 0.033975      binary time (ms): 2.236444
size: 32768     linear time (ms): 0.042667      binary time (ms): 4.993973
size: 65536     linear time (ms): 0.058864      binary time (ms): 14.082365
size: 131072    linear time (ms): 0.058469      binary time (ms): 36.137467
size: 262144    linear time (ms): 0.062815      binary time (ms): 35.753863
size: 524288    linear time (ms): 0.041481      binary time (ms): 135.107108
size: 1048576   linear time (ms): 0.050963      binary time (ms): 95.307814
```

5. Analysis and Conclusions

As the sizes grew for the arrays, the linear search time did not show much pattern. It is inconclusive whether this is the intended effect, but the assumption can be made that the size does not have a definitive effect until it gets unreasonably large. The large time for the size-16 array for the linear search may be attributed to how the compiler works when the program begins to run since that is the first iteration.

For the binary search, the times become longer near the end because the Arrays.sort(arr) function takes much longer to do than searching. Therefore, the binary search is inefficient when the array is unsorted. Only at certain lower sizes is this binary search viable when it is unsorted. However, when sorted, it becomes much more useful.

I was unable to make conclusions off of the linear search's data. I think the expected output should have shown the time increasing as the size increased, but the data failed to show that.

6. References

Professor Mahmut's sample report was used as a reference along with the information provided on canvas.