Implementing and evaluating performance from linear and binary search algorithms
Homework #1

By
Eduardo Castro

CS 303 Algorithms and Data Structures

September 3, 2014

1. Problem Specification

The goal of this assignment was to write a program that will read a text file with a lot of different numbers and another file with some keys(or in other words numbers to be searched) that will serve as reference for two search algorithms that needs to be implemented in this homework: the linear search and the binary search.

After the implementation we need to evaluate the results and compare the performance from both algorithms.

2. Program Design

This program required two search algorithms: one for linear search and another one for binary search. The project also needs to have a driver class to test both search algorithms and should work for different and huge sets of data.

The following steps were required to develop this program:

a) write the LinearSearch class
b) test it for different sets of data and possibilites using JUnit
c) write the BinarySearch class
d) test it for different sets of data and possibilites using JUnit
e) write the driver class
f) use the driver class to display and measure the results/performance

The following methods were defined within the LinearSearch, BinarySearch and TestSearch classes:

a) lookForTheNumber (int[] arrayOfNumbers, int sizeOfArrayOfNumbers, int key)
    a. Receives the array with all the numbers, and integer that informs how many elements there are in that array and a key that represents the number that the software needs to find. It returns a message saying that the number was found or not.
b) makeALinearSearch()
    a. Sets the path for the files with the numbers and keys, creates an array that can receive all the numbers from this file and calls the method from LinearSearch class that iterates around this array looking for the key.
c) makeABinarySearch()
    a. Sets the path for the files with the numbers and keys, creates an array that can receive all the numbers from this file and calls the method from BinarySearch class that iterates around this array looking for the key.
d) getsHowManyElementsOnTheInputFile()
    a. Makes a search between all the elements from the text file to discover how many elements it has.
e) getsHowManyElementsOnTheKeyFile()
    a. Makes a search between all the elements from the text file to discover how many keys to be found it has.

The Scanner class provided by Java was used to read in the necessary values within the provided driver program. The println method of the System.out object was used to display the inputs and results for the provided driver program.

3. Testing Plan

The provided driver program, TestSearch.java, was used to test the LinearSearch and BinarySearch classes as required by the assignment. The following test cases were used for this assignment.

The inputs used came from the text files given for the assignment: inputs100.txt, inputs1000.txt and inputs10000.txt for the data structure with all the elements and keys.txt only with the elements that we want to find at this data structure. The keys are always the same, what we change in each test case is only the input file(and obviously its length).

The TestSearch.java class catches this file and test all this data using the LinearSearch or BinarySearch algorithms. For evaluating purposes the TestSearch also catches the execution time from each search.

4. Test Cases

The test cases are shown in the table below:

Test Case Number	Algorithm	Input Quantity	Expected Output
1	Linear	100	15 is not in the list 55 is not in the list 900 is not in the list 987 is in the list 1 is in the list 25 is not in the list 566 is in the list execution time for linear search with 100 elements: 25

2	Linear	1000	15 is not in the list 55 is not in the list 900 is not in the list 987 is not in the list 1 is not in the list 25 is not in the list 566 is not in the list execution time for linear search with 1000 elements: 26
3	Linear	10000	15 is not in the list 55 is not in the list 900 is not in the list 987 is not in the list 1 is not in the list 25 is not in the list 566 is not in the list execution time for linear search with 10000 elements: 25
4	Binary	100	15 is not in the list 55 is not in the list 900 is not in the list 987 is not in the list 1 is not in the list 25 is not in the list 566 is not in the list execution time for binary search with 100 elements: 26
5	Binary	1000	15 is not in the list 55 is not in the list 900 is not in the list 987 is not in the list 1 is not in the list 25 is not in the list 566 is not in the list execution time for binary search with 1000 elements: 25
6	Binary	10000	15 is not in the list 55 is not in the list 900 is not in the list 987 is not in the list 1 is not in the list 25 is not in the list 566 is not in the list execution time for binary search with 100 elements: 26

5. Analysis and Conclusions

LinearSearch seems to be better for small sets of data and BinarySearch for bigger ones. At the first time that I made the project the algorithms was correct but the code was not working because I was not manipulating the Scanner library correctly but at the second time everything seems to work fine. The output now is exactly the same as expected. The code was developed using TDD and all the tests passed.

According to the execution times using the 3 different sizes data sets with two different search algorithms the differences were not that clear. The LinearSearch even seems to be faster, or at least the same, but according to the studied theory I know that the BinarySearch is better for huge data sets and use less steps when searching and can find the element faster because it doesn't iterate into all the elements of a data set.

The program could be improved by designing it in a better way, I didn't have enough time to do it and the TestSearch class made most part of the work, preparing the data set and calling the LinearSearch or BinarySearch class. I did not followed the atomic pattern(as we call it in Portuguese, I'm not sure yet about how this pattern is called in English) and the main class has a lot of responsibilities. It would also be great to create a proper JUnit class with better tests and better comments.

6. References

The parameters used was from the homework assignment provided in class and I used as reference for learning how to manipulate text files in Java one post from Stack Overflow(http://bit.ly/1gS00Ne) and another post for learning how to measure time(http://bit.ly/Wck04t).