Recursive Binary Search Homework
CSS 143: Programming Methodology
Instructor Lesley Kalmin

Assignment by Rob Nash

Summary

Build two classes (LinearSearch ,BinarySearch) that inherit from the provided parent class SearchAlgorithm, and a third exception class. Each of the subclasses will implement only one public method that facilitates the appropriate search over an array of Strings. Also, you will need to build one exception class (10 lines of code or less) that services these two searches in the event the item isn't found in the String array.

Introduction

The best introduction to this homework is probably to start in the BinSearchDriver code that will be used to test your two search classes. Walking through this code, you can see that we will create two objects for use in searching through a large array of words. This array is populated from a sequential text file (longwords.txt), and all of this code is written for you in the driver. Your job is to construct two *subclasses* that derive from the superclass SearchAlgorithm; these two classes will each need to provide a version of the abstract search method declared in SearchAlgorithm. Note that you will not need to change any of the code in main for the driver, and in fact, should only need to modify the FILE_AND_PATH class constant to point at your local version of the input file (longwords.txt). Beyond that, the driver is self-contained, and should compile and execute without modification once your two classes {BinarySearch, LinearSearch} are built and in the same working directory. Lets discuss the two subclasses more in detail next, starting with an item-by-item (linear) search.

The LinearSearch Class

This class should loop through the words from beginning to end, comparing the current string with the target string at each step. Note that you must call the utility function incrementCount() each time you make a comparison (ie, each time in the loop) so that when your search is complete, you have an accurate count of the comparisons required by your search strategy. You must implement this iteratively and then attempt to implement this recursively.

Note that we say "attempt". What happens when you run the recursive linear version? Report on the results in your submission. You may put this in the comments of the recursive method.

The BinarySearch Class

The binary search algorithm can be accomplished in a number of ways, but the basic algorithm is outlined below. You must implement this **recursively** and **iteratively.**

An outline of the iterative algorithm:

LowIndex = 0
HighIndex = arraySize – 1
While LowIndex is less than or equal to HighIndex
    Set MidIndex to be equal to half way between the low and high index
    If the target word matches the word at MidIndex, return MidIndex (first case)
    If the target word is before the word at MidIndex, then set HighIndex to MidIndex - 1
    If the target word is after the word at MidIndex, then set LowIndex to MidIndex + 1
If the target word was not found, throw an ItemNotFoundException (you create this class)

The ItemNotFoundException Class

This class should be under 15 lines of code, and is an exercise in inheriting from classes provided to you by the Java API. Your class should have only two methods (both constructors) and no data items; see the slides on exceptions for an example of such a class.

Recursion

Provide the iterative implementation for the linear & binary search in the search() methods that you override, and put the recursive versions of these same methods in the recSearch() methods, respectively. Note that you must extend the driver to call your recursive functions!

Notes

- Your classes ***must*** inherit from the SearchAlgorithm class to compile and run.
- Your recursive search method may be a thin layer that calls an internal method with that take the initial low and high index values as parameters.
- Use the extra utility functions in SearchAlgorithm to track the number of comparisons that have been executed in each search.
- Interested in determining if string1 comes before string2 alphabetically? This ordering can be determined using the compareTo() function; if string1.compareTo(string2) returns a positive int, then string 1 > string2. If negative, then string1 < string2.
- If you fail to find the target word in the set, be sure to throw an ItemNotFoundException
- Your three classes will be graded by compiling them with the provided SearchAlgorithm and BinSearchDriver classes *without modification*.