Assignment 9: Search and Sort

CPSC 1150-006/007, Fall 2016
Instructor: Adina Goldberg
Langara College

Preparation

You are expected to be familiar with the following textbook sections and lectures *before* attempting the assignment. You will be required to submit the answers to some questions when you submit your code.

Textbook sections

- 7.10, 7.11

Most relevant lectures

- L15

Exercises

1. Read through the assignment.

Introduction

You may work in pairs. Download the two files `SearchAndSort.java` and `Lab9.java` from D2L, and store them in the same directory. You should not modify `Lab9.java`.

Comparing `Strings`

During this lab, you will need to compare `String`s to sort them into alphabetical order. You should ignore the distinction between upper- and lowercase letters. This means that the instance method `compareToIgnoreCase` from the `String` class will help you greatly. Look up this method in the Java API if you do not remember the type or meaning of its return value.

1 Implementing methods

In this section, you will implement four methods in the class called `SearchAndSort`. The method headers are already coded for you.

Before moving on to Section 2, you must test all of your methods with several different inputs to ensure that they are working properly. Use the `main` method to write some **test cases** for each method you implement.

You will notice a class variable called `numComparisons`. Do not use this variable until Section 1.4.

1.1 Linear Search

Implement and test the `linearSearch` method. The method should return an index of `key` in `arr`, and should use the Linear Search algorithm – see class slides. If `key` is not found, return `-1`.

1.2 Selection Sort

Implement and test the following two methods:

1. `selectionSort`: The `selectionSort` method should sort the array `arr` using the Selection Sort algorithm – see class slides. `selectionSort` must invoke the `swap` method.

2. `swap`: The `swap` method should swap the elements at positions `i` and `j` in the array `arr`.

1.3 Binary Search

Implement and test the `binarySearch` method. The method should return an index of `key` in the (already) sorted array `arr`, and should use the Binary Search algorithm – see class slides. If `key` is not found, return `-1`.

1.4 Counting comparisons

Counting the comparisons made by your search methods will help to demonstrate the speed (time complexity) of the two algorithms.

Note that the `SearchAndSort` class contains a class variable called `numComparisons`. Modify **both of your search methods** (`linearSearch` and `binarySearch`) so that they each do the following things:

1. Set `numComparisons` to `0` at the beginning of the method.

2. Increment `numComparisons` each time two elements are compared. Make sure you don't miss any comparisons!

2 Analysing the search methods

1. Once all your methods are working properly, delete the main method in the `SearchAndSort` class.

2. Ensure that `Lab9.java` from D2L is saved in the same directory as `SearchAndSort.java`.

3. Open `Lab9.java` and figure out what the main method does. Make sure you understand how `Lab9` uses your `SearchAndSort` class.

4. Compile `SearchAndSort`. Then, compile and run `Lab9`.

5. Use the output of `Lab9` to answer the following questions. The answers to these questions must be submitted with your lab.

   (a) Does Linear Search work more efficiently on a sorted list than an unsorted list? Explain.

   (b) Why doesn't Binary Search work on an unsorted list?

   (c) As the number of entries in the array (words in the sentence) increases, which of the following grows faster, and why?

      i. the number of comparisons made by Linear Search

ii. the number of comparisons made by Binary Search

(d) If the key is not in the array, how many comparisons are made by Linear Search? How many comparisons are made by Binary Search?

(e) Imagine you have an unsorted array of 10,000 Strings. You need to search the array, and if you'd like to use Binary Search, you certainly have to sort the array first.

i. If you had to search it once, which search algorithm would you use, and why?

ii. If you had to search it fifty thousand times (for fifty thousand different keys), which search algorithm would you use, and why?

Submission

Recall that submission instructions are in the **Lab Guide**. Your group is required to submit **one** `.zip` folder (in one person's D2L dropbox) containing the internally documented, and properly styled source code file:

- `SearchAndSort.java`
  - Final version should not have a main method
  - External documentation is not required

Also, you need to submit a text file (or .pdf) containing:

- the console output of `Lab9`
- the answers to the questions from Section 2

The files should contain both partners' names in the headers. *Make sure both partners save copies of the finished code to their personal H: drive.*