Lab 4 - Sorting

due September 22nd 2019 at 23:55

Objective

Today's lab will help you get familiar with

- Different Sort algorithms
- Writing a report
- Running experiments from the command line

Notice that this lab requires you to write a longer lab report than usual, with similar guidelines to the project reports. The project report guidelines are given as a separate document for this lab.

Binary Search

In this lab you will need to implement a binary search method. This method provides an efficient way to search an array for a value given that the array is sorted. If you have not gone over binary search in class the following provides an explanation of how it works: https://www.tutorialspoint.com/data_structures_algorithms/binary_search_algorithm.htm

---

Saving Experiment Results

When conducting experiments with different parameters it makes sense to save the results in a table. The easiest way to do that is to print your results in a CSV format. A CSV file is a text file where each line is a row, and columns are separated by commas. For example a table like this:

	1000	2000	3000	4000
Method a	2	5	8	10
Method b	1	1	1	1

Would be represented as a CSV in the following way:

```
, 1000, 2000, 3000, 4000
Method a, 2, 5, 8, 10
Method b, 1, 1, 1, 1
```

Therefore when running experiments from your experimentController it is easiest to simply create a file, write all the results in a CSV format and then open it with any spreadsheet application (such as excel). This will allow you to easily create graphs.

In addition, setting up command line arguments which will set the different parameters allows you to run the program with different parameters without needing to recompile it each time. For example, you might want to setup the program to run like this:

```
java ExperimentController 10 output.csv
```

This will run each method 10 times (to get an average) and output the results to output.csv. By not hard coding the parameters in code, you have more flexibility on how you would like to run your experiments.

Assignment:

This lab will require you to program and compare multiple sort and search algorithms

1. Construct a class called StringContainer that contains an ArrayList (private) of Strings that are just numbers.
    a. Create a constructor for the class to create an empty ArrayList.
    b. Create an add method which takes a number, converts it to a string and adds the string to the end of the ArrayList
    c. Create a method which performs insertion sort (Section 4.3 in your textbook) on the ArrayList.
    d. Create a method which performs selection sort (from Lab 2) on the ArrayList.
    e. Create a method which given a String performs linear search on the ArrayList.
    f. Create a method which given a String (number) performs binary search on the ArrayList.
2. Construct a class called RandomStringGenerator which generates random Strings of a specific length.
    a. Create a constructor which takes a seed and the length of the string to be generated.
    b. Create a method called nextString() which returns a random string with the correct length. The String should consist of numbers. For example: if the length of the string is 4 then it should create Strings of numbers (1000 - 9999). Create an ExperimentController with the following methods (similar to lab 2):
    c. timeToInsertionSortUnsorted(int numberOfItems, int seed)
    d. timeToInsertionSortsorted(int numberOfItems, int seed)
    e. timeToSelectionSortUnsorted(int numberOfItems, int seed)
    f. timeToSelectionSortsorted(int numberOfItems, int seed)

g. timeToSearchLinearFound(int numberOfItems, int seed) - Search for random strings which are in the array. You can choose how many (As a percentage of the array size. Should be distributed throughout the array)
h. timeToSearchLinearNotFound(int numberOfItems, int seed) - Search for a random string which is not in the array.
i. timeToSearchBinaryFound(int numberOfItems, int seed)
j. timeToSearchBinaryNotFound(int numberOfItems, int seed)
k. Your main method should be able to use the command line arguments to define the following:
    i. The output file name
    ii. The number of experiments to average over
    iii. A list of the number of items you would like to run the experiment with

    For example if I run:

    ```
    java ExperimentController output.csv 5 1000 2000 5000
    ```

    A table should be written to output.csv with 3 results columns (1000, 2000, 5000), each averaged over 5 runs. The table should perform the experiments for all 8 methods a-h. You can choose an arbitrary string length.

3. Run the code in the command line and make sure to include a screenshot of one run in your report.
4. Create graphs to include with the report:
    a. A graph which compares the different sorts as a function of the number of elements in the arrayList
    b. A graph which compares the different searches as a function of the number of elements in the arrayList.

Submission:

In addition to your code you must submit a report (in PDF format). Save your report in the project folder before you compress it and upload it. Details about the report can be found on the CS150 class moodle page.

Grading:

1. unit testing (for the first two classes) - 1 pt
2. RandomStringGenerator class - 1 pt
3. StringContainer class - 3 pts
4. ExperimentController Class - 1 pt
5. Running from command line and writing to file - 1pt
6. Style/commenting - 1 pt
7. project write up - 2 pts