CSCI 211 Lab 7. Sorting

Objectives

- To understand the efficiency of sorting algorithms by timing their execution
- Be able to call the sorting algorithms using an array of objects

This lab makes use of the Sorting class in the textbook. There are three parts. For the first part, you will fill in code to time sorting of randomly filled Integer arrays using the various algorithms as implemented in the Sorting class (NOTE: all of the sorting algorithms are static – to invoke, use the class name. For example, if you have a *numbers* array declared and populated, *Sorting.insertionSort(numbers)* will invoke the insertionSort method using *numbers*). The second part is to use the Sorting class to sort an array of objects. The third part will create a large array of objects and you will once again time sorting using the various algorithms.

Step 0. From Blackboard, download the SortingLab files and unzip. Copy and paste into an IntelliJ Maven project (main > java) **.**

Step 1. Fill in the TODO 1 section of the code. Note the method setRandom has been provided. It fills in the numbers array with random numbers to aid in performing the timing operations. A call to setRandom and a statement to output when the sorting algorithm is complete have been provided in the Driver. You will need to fill in the steps between. In addition, the statements to output the final results have been provided in the Driver as well.

The basic steps to do the timing for each sort are:

- Fill the numbers array with 50000 random values (using the setRandom method).
- Use the method System.currentTimeMillis() to get and store the start time for the sort. Note that currentTimeMillis() returns a long that represents "the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC".
- Execute the appropriate sorting algorithm on the numbers array (again, these are static methods in the Sorting class).
- Use the method System.currentTimeMillis() to get and store the end time.
- Calculate the duration of the sort in milliseconds by subtracting the start time from the end time.

Step 2. Fill in the TODO 2 section of the code. The second part of the lab uses the Person class to construct an array of Person objects, and then sort them.

- This part was commented out until you finished the first part of the lab. So, uncomment the code for Part 2 by removing the /* and */.
- Note that the code declaring, instantiating and populating the Person [] called people, peopleUnsorted and peopleToSort have been provided. The file uses

names.txt that contains 75 names, one per line with each line containing a first name, space, and last name.

- A printList method to print the array in the current order is provided. You should not need to make any additional changes to TODO 2 in the Driver.
- Fill in the missing code in the Person class, including a constructor, getters and setters, a compareTo method and a toString method. Note that the order the compareTo gives should be alphabetical order by last name and when the last names are the same, then by the first name.
- Verify that your array of objects correctly sorts in the Driver

Step 3. Fill in the TODO 3 section of the code. The third part of the lab uses a Person array 650 times larger than the original Person array. You will now determine the time it takes each of the sorting algorithms using an array of objects. Code to copy the unsorted array (peopleUnsorted) into the array to sort (peopleToSort) have been provided for you in the Driver. Similar to what you did in TODO 1

- Use the method System.currentTimeMillis() to get and store the start time for the sort. Note that currentTimeMillis() returns a long that represents "the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC".
- Execute the appropriate sorting algorithm on the peopleToSort array (again, these are static methods in the Sorting class).
- Use the method System.currentTimeMillis() to get and store the end time.
- Calculate the duration of the sort in milliseconds by subtracting the start time from the end time.

Step 4. You must do ALL of the following to get credit for the lab

1. Use the queue system to let the TA know that you are ready for your work to be checked.
2. Demonstrate to the lab TA that your code works correctly.
3. Export the project (or zip it in Explorer or equivalent).
4. Upload the zip file to Lab 7 on Blackboard.