Homework #1 – COMP 250, Fall 2014 Mathieu Blanchette

Due date: Sept 22th 2014, before 23:59.
Turn-in your code for questions 1-4 electronically via myCourses.
Turn-in your answer for question 5 as a Word document, PDF document, or scan of a hand-written document, via myCourses.

This homework can be done in teams of two students.
Only one myCourses turn-in per team is necessary, but make sure to write both students' names at the very top of each file you turn in, using this exact syntax:
// STUDENT1_NAME: John Doe
// STUDENT1_ID: 260012345
// STUDENT2_NAME: Janet Doe
// STUDENT2_ID: 26054321

In this first assignment, you will implement in Java some of the algorithm we have seen in class to compute the size of the intersection between two unsorted list of students containing no duplicates. Start by downloading the code at http://www.cs.mcgill.ca/~blanchem/250/hw1/studentList.java .

Question 1. (10 pts)

Write the code of the intersectionSizeNestedLoops method, implementing the algorithm seen in class (Lecture 2; Algorithm 1).

Question 2. (20 pts)

Write the code of the intersectionSizeBinarySearch method, implementing the algorithm seen in class (Lecture 2; Algorithm 2). For this, you will need to be able to sort an array of integers. For an array of int called myArray, this is done by writing:
Arrays.sort(myArray);
The class Arrays also contains a method called binarySearch. I am asking you **not** to use that method, and to instead *implement your own version* of that algorithm, in the method called myBinarySeach.

Question 3. (20 pts)

Write the code of the intersectionSizeSortAndParallelPointers method, implementing the algorithm seen in class (Lecture 2; Algorithm 3).

Question 4. (20 pts)

Write the code of the intersectionSizeMergeAndSort method, implementing the algorithm seen in class (Lecture 2; Algorithm 4).

Question 5. (30 pts)

So, which of the methods implemented in questions 1-4 is fastest? This is for you to discover! For each intersection algorithm implemented, you will measure the average

running time needed to compute the intersection size of two lists of 2, 4, 8, 16, 32, 64, 128, 256, 512, 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000, 256000, 512000, and 1024000 students each.

Notes:

- For these measurements, the best is to use lists of randomly generated student IDs, as done in the code already.
- For small numbers, the execution will be too fast to be measured reliably, so you should measure the time to do, say, 1000 such intersections (each time on a different pair of randomly generated lists – see code for an example), and then divide the total running time by 1000.
- You can use
  long currentTime = System.nanoTime();
  to get the time (in nanoseconds) before and after the execution of the 1000 repetitions.
- The time taken to generate the two random lists is not negligible. You should first measure the running time of a program that generates the list but doesn't call the relevant intersectionSize method, and then run the program again, this time generating the lists and calling the intersectionSize method. The difference between the two running times is the amount of time taken by your method.
- When running your experiments, make sure that your computer isn't running too many other jobs; this may affect your measurements.

Questions:

a) (10 points) Report a table for the average running time for a single list intersection for each method and list sizes.

b) (10 points) Based on your observations, try to determine how the running time of each method increases as a function of the size of the two lists. For large values of n (e.g. n>=32000), the increase in running time should be quite predictable.
In general, what would be the average running time of each of the four methods for computing the intersection between two lists of n students (assuming n>=32000)? Give your answers as a function of n (e.g. $T(n) = 43\ n + 5 \log(n)$ milliseconds).

c) (10 points) Now, consider what happens if the two lists are of different sizes. For each of the four algorithms, report the running time for computing the intersection between a list L1 of 32000 students and a list L2 of 1024000 students, and the running time for computing the intersection between a list L1 of 1024000 students and a list of L2 of 32000 students. For each algorithm, explain (in 2-3 lines at most) why the running time changes or doesn't change.

Good luck!