CS325: Analysis of Algorithms, Fall 2016

Group Assignment 1*

Due: Tue, 10/11/16

Homework Policy:

1. Students should work on group assignments in groups of preferably three people. Each group submits to TEACH a zip file that includes their source code and their report. Each group, also, hands in a printed hard copy of the report in class or slides the hard copy under my door before the midnight of the due day. The hard copy will be graded, and the codes submitted to teach will be tested.
2. The goal of the homework assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.
3. You are allowed to discuss the problems with other groups, and you are allowed to use other resources, but you *must* cite them. Also, you must write everything in your own words, copying verbatim is plagiarism.
4. *I don't know policy:* you may write "I don't know" *and nothing else* to answer a question and receive 25 percent of the total points for that problem whereas a completely wrong answer will receive zero.
5. Algorithms should be explained in plain english. You can use pseudocodes if it helps your explanation, but the grader will not try to understand a complicated pseudocode.
6. More items might be added to this list. ☺

Suppose you are given two sets of n points, one set $\{p_1, p_2, \ldots, p_n\}$ on the line $y = 0$ and the other set $\{q_1, q_2, \ldots, q_n\}$ on the line $y = 1$. Suppose, $p_1 \cdot x < p_2 \cdot x < \ldots < p_n \cdot x$ (where $p_i \cdot x$ is the X-coordinate of p_i). Create a set of n line segments by connecting each point p_i to the corresponding point q_i. Suppose no three line segments intersect in a point.

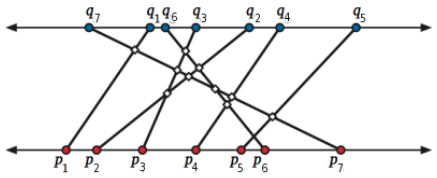

Figure 1: Eleven intersecting pairs of segments with endpoints on parallel lines.

In this assignment you design two algorithms to compute how many pairs of these line segments intersect.

*The problem is from Jeff Erickson's lecture notes. Looking into similar problems from his lecture notes on recursion is recommended.

Step 1. Design a $\Theta(n^2)$ time recursive algorithm to count the number of intersections.

Step 2. Modify your algorithm of Step 1 to work in $\Theta(n \log n)$ time. (Hint: Use divide and conquer.)

Input/Output Your program reads from the input file "input.txt". Line 1 of the input is a single integer $1 \leq n \leq 10^5$. Line 2 is a list of n numbers separated by commas that are X coordinates of $p_1, \ldots, p_n$. Line 3 is a list of n numbers separated by commas that are X coordinates of $q_1, \ldots, q_n$.

Your output must be written in "output.txt". The output is a single number, the number of intersections. Note that the number of intersections is at most $\binom{10^5}{2}$.

Sample Input (1):
2
2,15
4,10

Sample Output (1):
0

Sample Input (2):
3
5,6,8
3,2,1

Sample Output (2):
3

Sample Input (3):
7
2,4,6,7,8,9,10
2,5,4,6,7,8,1

Sample Output (3):
7

Code (30%). Write a program to find the number of intersections. Your program will be tested against several test cases, for correctness and efficiency. For each test case, the program will be automatically stopped after 20 seconds if it is not done in that time, and the group miss the points of that test case. Your program must be written in one of the following languages: Python, C++, or Java.

Report (70%). In your report, include descriptions for both algorithms. Prove that your algorithms are correct, and provide running time analysis. Algorithms should be explained in plain english. You can use pseudocodes if it helps your explanation, but the grader will not try to understand a complicated pseudocode.

Submission Each group submits to TEACH a zip file that includes their source code (*which must be just one file with name "intersectCount.cpp", "intersectCount.java", or "intersectCount.py"*) and their report. This file can be submitted by any member of the group, but all names must be listed in the submitted report. Each group, also, hands in a printed hard copy of the report in class or slides the hard copy under my door before the midnight of the due day. The hard copy will be graded, and the codes submitted to teach will be tested.

Test your code with the sample test files (`http://web.engr.oregonstate.edu/~nayyeria/CS325/Fall16/hws/test1.zip`) before submitting them, to make sure there is no formatting error.