Computer Science 130B
Spring 2017
Programming Assignment #1

Due: **11:59pm**, April 23, Sunday

Consider a problem in computational geometry: Given a set of n 2D points, the goal is to find the closest pair of points in the set. Assume that coordinates are given:

$$P_i = (x_i, y_i), \qquad i = 0, \cdots, n-1.$$

a. Design a brute-force algorithm to accomplish $ClosestPair(n, X, Y)$, where n is the number of points, X and Y are arrays of size n which store the x and y coordinates, respectively, of the n points. Analyze the complexity of your algorithm.

b. Design a divide-and-conquer algorithm for $ClosestPair(n, X, Y)$. Your algorithm should have a better performance than the brute-force one. Analyze the complexity of your algorithm.

For parts **a.** and **b.**, electronically turn in a PDF file named README.pdf with your codes that contains your analysis. You can use any document generation software (MS Word, Linux OpenOffice Writer, Latex, etc.) to generate the PDF file.

c. Implement the brute-force algorithm and the divide-and-conquer algorithm in C++. Your program should accept inputs of the following format from standard input (`stdin` in C) (coordinates are real numbers):

n	/* number of points */
$x_0\ y_0$	/* coordinate of the 1st point */
$x_1\ y_1$	/* coordinate of the 2nd point */
$\cdots$	
$x_{n-1}\ y_{n-1}$	/* coordinate of the nth point */

Your program should output the following information (written to `stdout` in C):

$x_i\ y_i\ x_j\ y_j$	/*	the x and y coordinates of the closest pair computed from your brute-force algorithm */
k	/*	total number of distance comparisons in your brute-force algorithm */
$x_{i'}\ y_{i'}\ x_{j'}\ y_{j'}$	/*	the x and y coordinates of the closest pair computed from your divide-and-conquer algorithm */
k'	/*	total number of distance comparisons in your divide-and-conquer algorithm */

The basic operation in finding the closest point pair is to compute the distances between different pairs of points, and then *compare* the distance measurements to select the smallest one. This comparison operation should be used in both your brute-force algorithm and divide-and-conquer algorithm. The complexity of your algorithms is largely determined by how many such comparisons are made. Hence, your k' should be much smaller than k. Try your program on sample inputs

of progressively larger sizes (e.g,. 10, 100, 1,000, 10,000, 100,000, 1,000,000, $\cdots$). Compare the run times and numbers of comparisons of the two implementations by plotting k and k' as functions of n. Include your plots in README.pdf. What do you observe?