Week 2: The closest pair of points in the plane

The problem: Given an array P of n points in the plane, find the closest pair. Assume that the distance between two points is given by the Euclidian distance.

Questions

1. Formulate the 1D version of the closest pair. How can you solve it, and how fast? Try to extend this solution to the 2D problem: does it work?

   For the remaining problems we consider the 2D version.

2. Describe how you can find a vertical line that splits P in half. How long does this take?

3. Consider the (refined) closest pair algorithm which takes as arguments the points in P sorted in two different ways. Let P_X and P_Y denote the points in P sorted by their x- and y-coordinates, respectively. Furthermore, Let L be the vertical line that splits P into two halves, and let P_1 and P_2 be the set of points in P to the left/right of this line, respectively.

   (a) Given P_X and P_Y, how can you find the x-coordinate of line L ?

   (b) Given P_X and P_Y, how can you find P_{1X} (the points in P_1 sorted by their x-coordinates) and P_{2X} (the points in P_2 sorted by their x-coordinates)?

   (c) Given P_X and P_Y, how can you find P_{1Y} (the points in P_1 sorted by their y-coordinates) and P_{2Y} (the points in P_2 sorted by their y-coordinates)?

4. The divide-and-conquer algorithm is guaranteed to run in $O(n \lg n)$ time for any set of points in the plane. In practice it is sometimes the case that data is nice; put differently we can make certain assumptions about the data, and exploit these assumptions to come up with simpler and more efficient algorithms.

   Assume that the set of points P is uniformly distributed. Can you come up with a different idea to find the closest pair? Can you get $O(n)$ time?

   Hint: throw a grid over the points. For the sake of the analysis, assume a grid of k-by-k cells. How many points do you expect to fall in each cell, on the average? What value of k would you pick ?