Worksheet 8. A D&C method to find the closest pair

The problem. Given a list $P = [(x_1, y_1), \ldots, (x_n, y_n)]$ of n points in the plane, find the pair (i, j) $(i \neq j)$ minimizing the distance

$$d((x_i, y_i), (x_j, y_j)) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Problem 1. There aren't all that many pairs (i, j) to check. What's the (asymptotic worst-case) complexity of the brute-force solution?

Problem 2. How efficiently can you solve the problem in 1 dimension? (That is, you are given a list of numbers on the number line, not a list of points.)
(*Hint:* Sort the list first.)

Problem 3. Since the 1-dimensional version of the problem can be solved so efficiently, you might hope that you could simply examine the two points with closest x-coordinate (or closest y-coordinate) and look among those for the closest points. Show by giving an example or two that this will not work.

We make the following simplification:

Assume all x-coordinates $x_1, \ldots, x_n$ are distinct.

Problem 4. We can easily (and efficiently) eliminate this assumption, for example by applying a rotation to the points that makes it true. *Briefly* discuss with your groupmates how this would work.

This is the general strategy for how the algorithm begins:

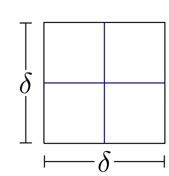
- Sort the input data by x-coordinate (in $O(n \log n)$ time).
- Find an x-value c such that $\lceil n/2 \rceil$ points lie in $L = (-\infty, c) \times \mathbb{R}$ and the remaining $\lfloor n/2 \rfloor$ points lie in $R = (c, \infty) \times \mathbb{R}$.
- Recursively find the closest pair p_L, q_L in L and the closest pair p_R, q_R in R.

Problem 5. Describe how to find c efficiently.

Let $\delta = \min(d(p_L, q_L), d(p_R, q_R))$. Of course, there might be points lying on opposite sides of the line $x = c$ that are closer than δ from each other. So we need to figure out how to deal with that.

For simplicity we discard all points with $x_i \notin (c - \delta, c + \delta)$ and sort the remainder by y-coordinate.

Problem 6. Prove that any $\delta \times \delta$ square in the plane contains at most 4 points of L.
(*Hint:* Divide the square into four $\delta/2 \times \delta/2$ squares.)

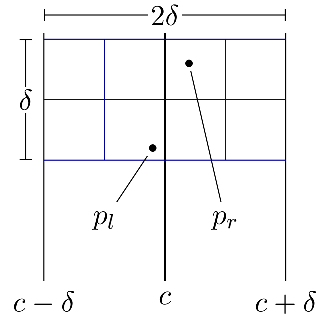


Problem 7. Suppose that $p_l = (x_l, y_l) \in L$ and $p_r = (x_r, y_r) \in R$ are the closest pair of points (among all the points)

(a) Explain why x_l and x_r must each lie in the interval $(c - \delta, c + \delta)$.

(b) Explain why p_l and p_r must lie in a $\delta \times 2\delta$ rectangle centered on the vertical line $x = c$.

(c) Explain why at most 8 points of $L \cup R$ lie in the $\delta \times 2\delta$ rectangle.



Problem 8. We can complete the recursive step of the algorithm as follows.

> Sort the points by y-coordinate. Scan through the list sorted by y-coordinate and, for each point, compute its distance to each of the subsequent 7 points in the list. Let p_M, q_M be the closest pair found in this way, and return whichever of (p_L, q_L), (p_M, q_M), (p_R, q_R) is closest.

Explain why this process produces the correct answer.

Problem 9. The worst-case running time $T(n)$ of this algorithm will satisfy a recurrence

$$T(n) = aT(n/b) + f(n).$$

(a) What is a? What is b?

(b) What is the complexity class of $f(n)$? Remember that we sorted by y-coordinate in the assembly step.

(c) What would $f(n)$ be if we somehow didn't have to sort by y-coordinate at each level of the recursive tree?

(d) Does the Master Theorem apply here?

Fear not: on the Problem Set, you will explain how to clean all this up to achieve $f(n) \in O(n)$ and therefore $T(n) \in O(n \log n)$.