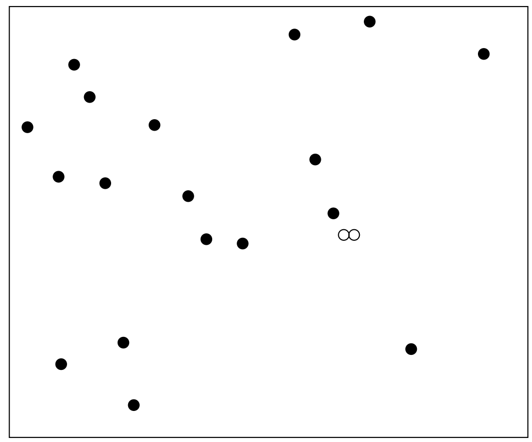
Closest Pair of Points

Reading: Goodrich/Tamassia §22.4. Suppose we have n points, each of which has an x-coordinate x_i and a y-coordinate y_i. Our goal is to find the pair of points p_i and p_j that are closest together. The distance between two points is $d(p_i, p_j)$.

Here is a Brute-Force approach to this problem:



`Closest-Pair`

Input: n points in $2D$-space

Output: The closest pair of points.

  min $= \infty$

  for $i = 2 \to n$ **do**

    for $j = 1 \to i-1$ **do**

      if $(x_j - x_i)^2 + (y_j - y_i)^2 <$ min **then**

        min $= (x_j - x_i)^2 + (y_j - y_i)^2$

        closestPair $= ((x_i, y_i), (x_j, y_j))$

  return closestPair

What is the running time of this algorithm?

To improve on the running time of the brute-force algorithm, we can try to set up our usual start for divide and conquer. For convenience, let's assume the points are sorted by y-coordinate before we first call this algorithm. We can do this in $\mathcal{O}(n \log n)$ time first; if the eventual running time is $\Omega(n \log n)$, this won't matter, and if we achieve $o(n \log n)$ for the rest of the algorithm, this will dominate the running time.

`Closest-Pair`

Input: n points in $2D$-space

Output: The closest pair of points.

  If P is sufficiently small, use brute force. // $\mathcal{O}(1)$

  $x_m \leftarrow$ median x-value from P

  $L \leftarrow$ any points from P with x-coordinate $\leq x_m$

  $R \leftarrow$ any points from P with x-coordinate x_m

  Let l_1 and l_2 be the closest pair of points in L, found recursively.

  Let r_1 and r_2 be the closest pair of points in R, found recursively.

  return whichever pair is closer together // Incorrect but good starting point.

The above algorithm is clearly incorrect; why?

How do we fix it?

How do we fix it while having a better running time than the brute force algorithm?