Closest Pair of Points

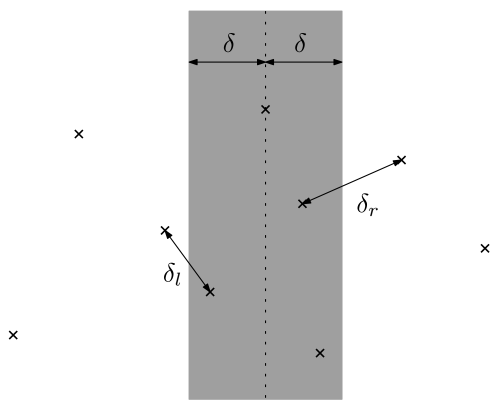
Reading:

- "Finding the closest pair of points" 33.4 CLRS

Problem: Given a list $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ of points in the plane, find the pair of points that are closest together.

Simple Solution: Test each of the $\binom{n}{2} \in \Theta(n^2)$ pairs of points to see if they are the closest pair.

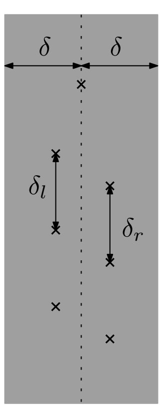
Sophisticated Solution: We can do better with a divide-and-conquer algorithm that exploits the geometry of distance. The idea is to split the point set into two halves with a vertical line, find the closest pair within each half, and then find the closest pair between the two halves. The result of finding the closest pair within each half speeds up finding the closest pair between the halves. The closest pair is the minimum of the closest pairs within each half and the closest pair between the two halves.



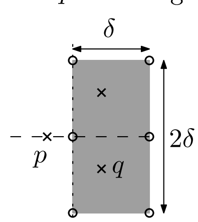
To split the point set in two, we find the x-median of the points and use that as a pivot. Finding the closest pair of points in each half is subproblem that is solved recursively. Let δ_l be the minimum distance in the left half, and let δ_r be the minimum distance in the right half. Then the minimum distance between every pair of points is less than or equal to $\delta = \min\{\delta_l, \delta_r\}$.

If the minimum distance between all pairs of points is equal to δ, we have found the pair and we are done. Otherwise, the minimum distance must be between a point on the left half and a point on the right half.

Both such points must be in the gray region shown because if one of the points were not, it would be greater than δ from the dotted dividing line and, hence, greater than δ from the other point.

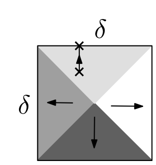


It is possible for all of the points lie within this gray strip. But do we really need to compare every point in the left half to every point in the right half? Absolutely not. A simple bound is that given a point p on the left half, it is sufficient to check every point q in the right half that is vertically within δ of p.

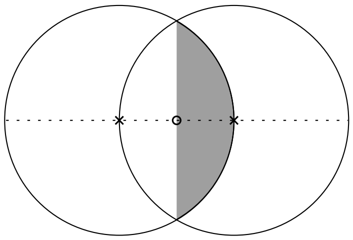


Fortunately, not many such q lie in this restricted region because those points are at least $\delta_r \geq \delta$ apart. The open dots in the picture show an arrangement of six points that are δ apart. We will now argue that this is the maximum number.

Consider a $\delta \times \delta$ square. We want to cram as many points into this square such that the points are at least δ apart. We first argue that we can put all of the points on the boundary of the square: For every point, we replace it with closest point on the boundary. The drawing on the left partitions the interior by the edge to which each point is closest.

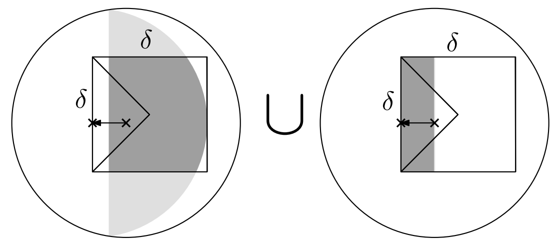


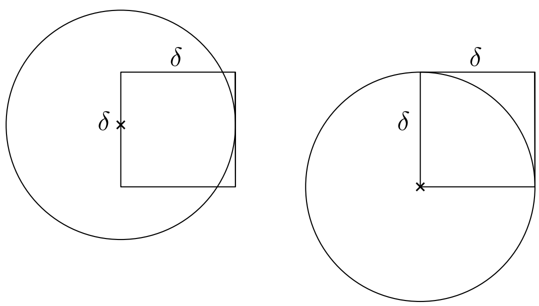
We push points to the edge this way because the area forbidden by the new point is strictly contained in the area forbidden by original point.



To see this, first consider two circles of the same radius δ (forbidden regions). Assume without loss of generality that the line through the circle centers is horizontal. The part of the left circle that is right of the midpoint of the two circle centers is contained in the right circle.

The region forbidden by the new point can represented as the union of two regions, both of which are forbidden by the old point. Hence, the region forbidden by the new point is contained in the old forbidden region. Therefore, given a set of k points at least δ apart that are contained in a $\delta \times \delta$ square, we push each point to boundary so that the k points are still at least δ apart.





Clearly we cannot place more than two points on a square edge. The only feasible arrangement of two points per edge is on the corners. If one point is on the interior of a square edge, the remainder of that edge is forbidden, and the best we can hope for the remaining edges is one per edge. If one point is on a corner, the interiors of two edges are forbidden, and the best we can do with the remaining two edges is three points.

So we only need to compare each point on the left to at most 6 points on the right. If the rights points in the gray band are sorted by y-coordinate, we can binary search for the topmost element on the right to compare to a given left point. The remaining interesting points on the right must follow consecutively because of the sorting.

```
Algorithm MinDist(P)
   if |P| ≤ 3 then
      try all pairs
   else
      find x median of the points
      split P into L and R of about the same size using x

      δ = min{MinDist(L), MinDist(R)}
      B_L =points in L within δ of x median
      B_R =points in R within δ of x median

      sort B_R by y-coordinates
      for p ∈ B_L do
         binary search for highest q ∈ B_R in a δ × 2δ box of p
         compare at most 6 points starting with q to p
```

The algorithm is dominated by sorting and recursing, so the run-time recurrence is $T(n) = 2T(n/2) + \Theta(n \log n)$. This is case 2 of the Master Theorem in the Goodrich and Tamassia textbook. Hence, the run-time is $\Theta(n \log^2 n)$. We note that the algorithm in CLRS does some clever bookkeeping to cut this down to $\Theta(n \log n)$.