Analysis of Divide-Conquer Algorithms

The typical recurrence describing the complexity of a divide and conquer algorithm would look like:

$$T(N) = aT(N/b) + f(N)$$

where

- N/b is the size of each subproblem
- a is the number of subproblems and
- $f(N)$ is the cost of dividing up the problem into subproblems and combining the solutions.

Usually, dividing into subproblems is quite direct and the ingenuity is in stitching together the solutions.

Closest Pair

Given a collection

$$P = \{(x_1, y_1), (x_2, y_2) \ldots (x_N, y_N)\}$$

of points on the plane, find the pair that is closest to each other.

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

Given a collection

$$P = \{(x_1, y_1), (x_2, y_2) \ldots (x_N, y_N)\}$$

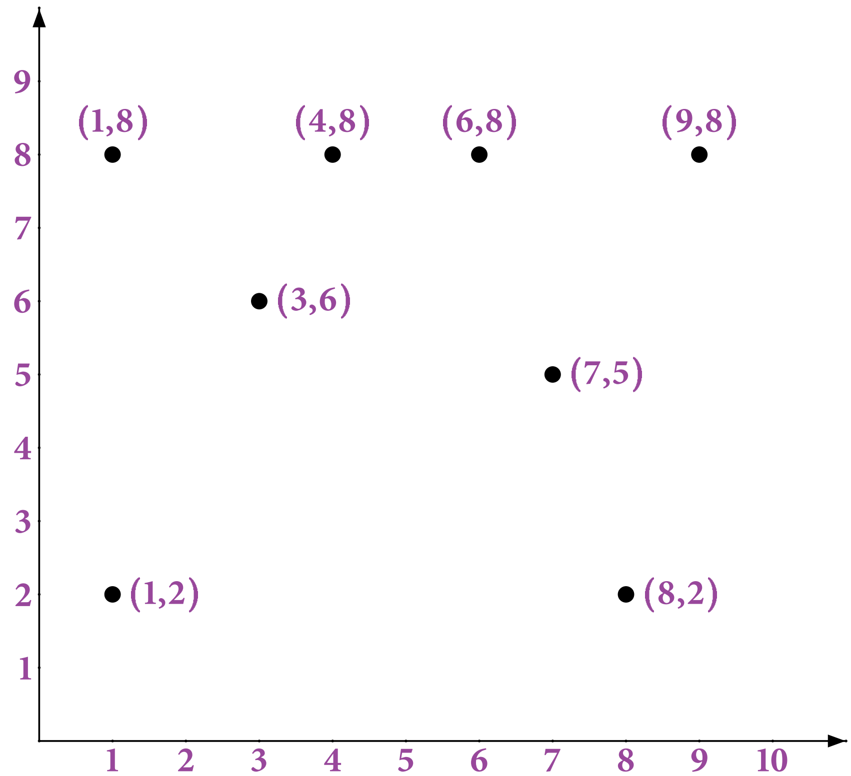
of points on the plane, find the pair that is closest to each other.

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

The obvious algorithm that considers all possible pairs and finds the minimum takes $O(N^2)$ time.

Consider the list of points

$$(1,2),\ (8,2),\ (7,5),\ (9,8),\ (6,8),\ (3,6),\ (4,8),\ (1,8)$$



Closest Pair: Divide and Conquer

Step 1: How do we divide the given problem into subproblems?

Closest Pair: Divide and Conquer

Step 1: How do we divide the given problem into subproblems?

Sort the points on their X-coordinates and divide them into to equal halves.

Step 1: How do we divide the given problem into subproblems?

Sort the points on their X-coordinates and divide them into to equal halves.

$$(1,2),\ (1,8),\ (3,6),\ (4,8),\ (6,8),\ (7,5),\ (8,2),\ (9,8)$$

Closest Pair: Divide and Conquer

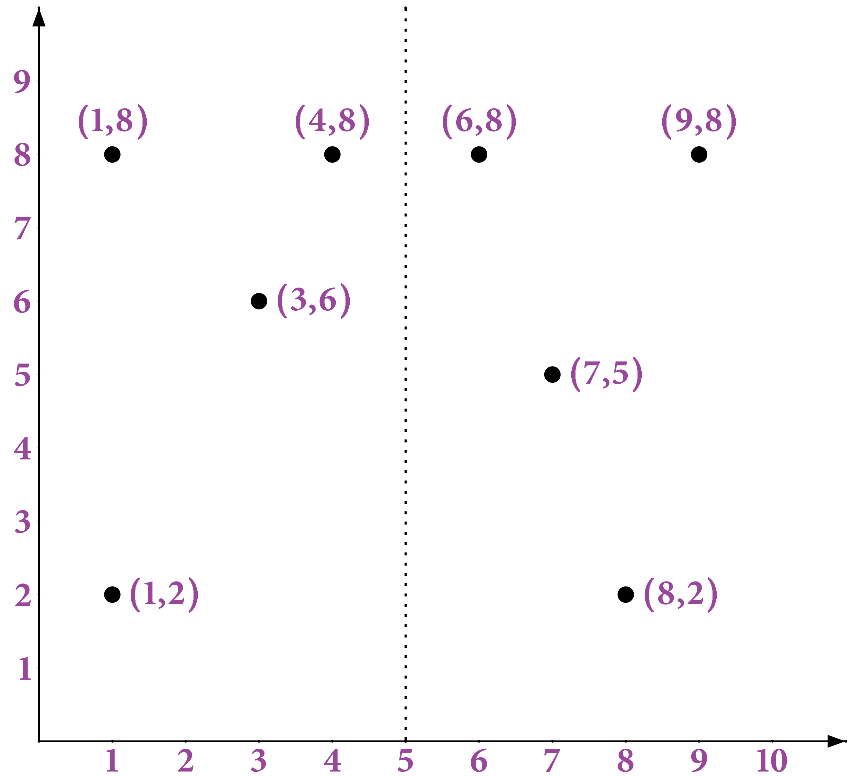
Step 1: How do we divide the given problem into subproblems?

Sort the points on their X-coordinates and divide them into to equal halves.

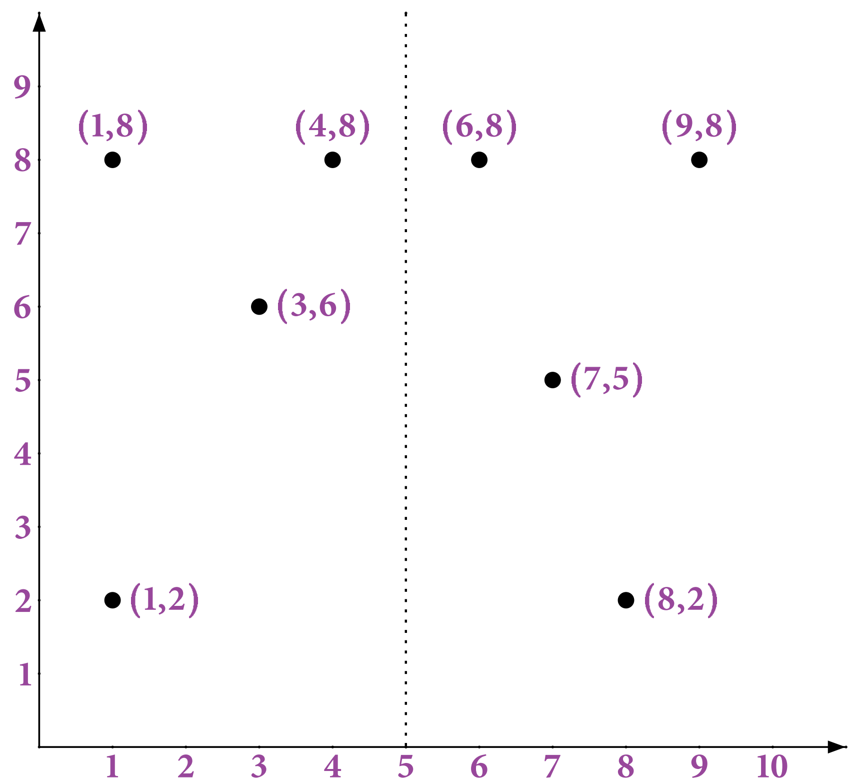
$$(1,2),\ (1,8),\ (3,6),\ (4,8),\ (6,8),\ (7,5),\ (8,2),\ (9,8)$$

$$(1,2),\ (1,8),\ (3,6),\ (4,8) \qquad (6,8),\ (7,5),\ (8,2),\ (9,8)$$

Two Subproblems

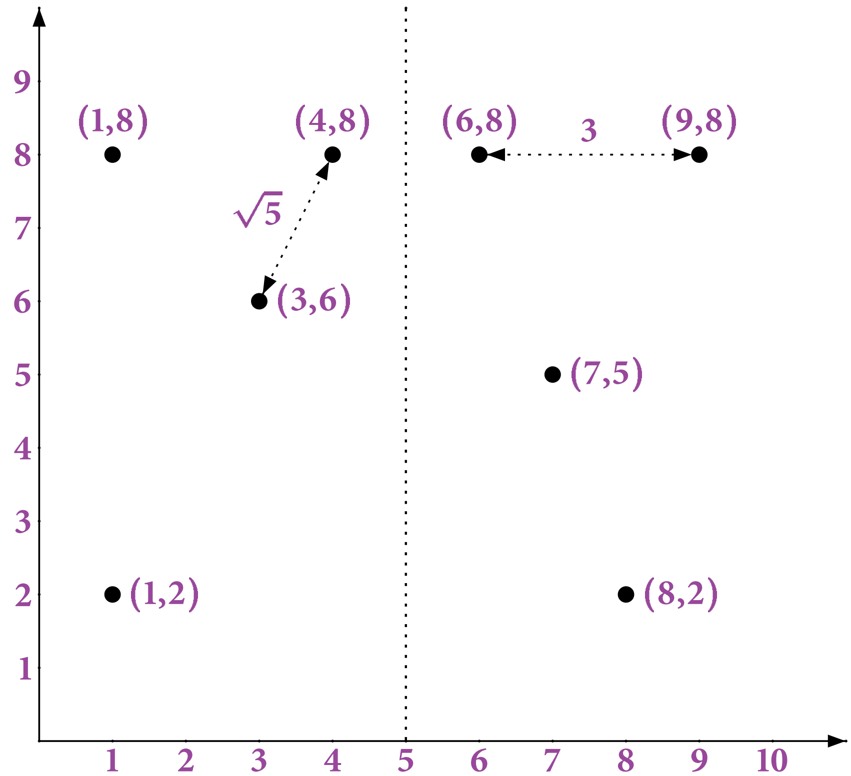


Two Subproblems

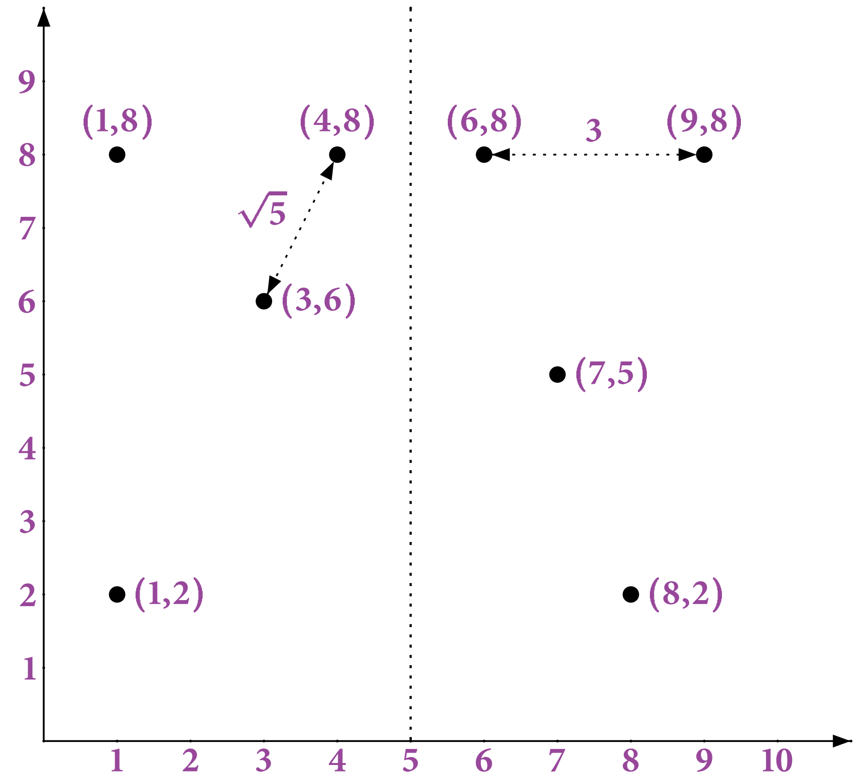


Solve each subproblem.

Solutions to Subproblems



Solutions to Subproblems



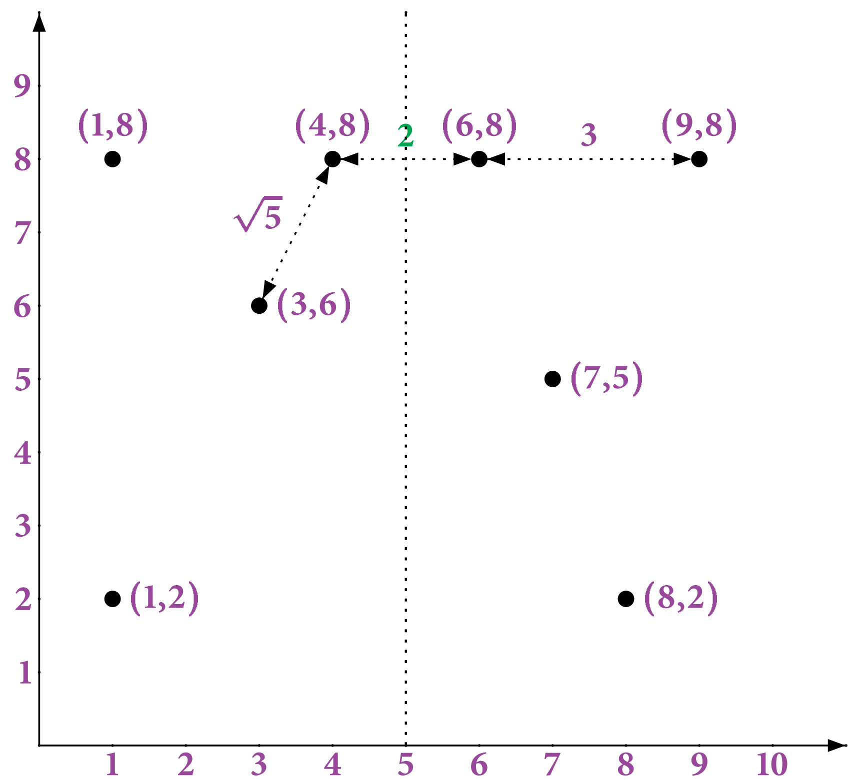
How do we put together the solutions?

Composing the Solutions

What is the difficulty?

What is the difficulty?

The closest pair of points can span across the division.



An observation

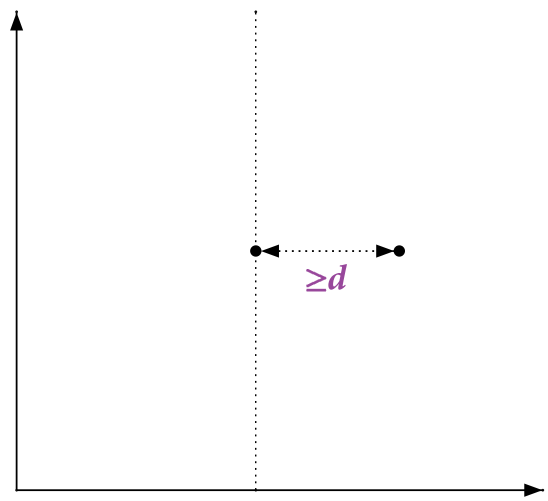
Let d_l and d_r be the answers to the two subproblems. Let d be the minimum of these two values.

Let $X = m$ be the line that is used to divide the problem into subproblems.

Let d_l and d_r be the answers to the two subproblems. Let d be the minimum of these two values.

Let $X = m$ be the line that is used to divide the problem into subproblems.

Observation: If (x, y) and (x', y') are from across the division and the distance between them is less than d then $|x - m| < d$ and $|x' - m| < d$.



The Algorithm: 1st Attempt

1. Sort the points by their X-coordinates. ($O(NlogN)$)

The Algorithm: 1st Attempt

1. Sort the points by their X-coordinates. ($O(N log N)$)
2. Pick $X = m$ that divides the set of points into two equal parts. ($O(N)$)

The Algorithm: 1st Attempt

1. Sort the points by their X-coordinates. ($O(N logN)$)
2. Pick $X = m$ that divides the set of points into two equal parts. ($O(N)$)
3. Find the shortest pair in each subproblem. ($T(N/2) + T(N/2)$).

The Algorithm: 1st Attempt

1. Sort the points by their X-coordinates. ($O(N log N)$)
2. Pick $X = m$ that divides the set of points into two equal parts. ($O(N)$)
3. Find the shortest pair in each subproblem. ($T(N/2) + T(N/2)$).

   Let d_l and d_r be the answers. Let d be the minimum of d_l and d_r.
4. Let S be the set of points whose X-coordinate is not farther than d from m. Find the shortest pair within this set. ($T(|S|)$ or $|S|^2$).

The Algorithm: 1st Attempt

1. Sort the points by their X-coordinates. ($O(N \log N)$)
2. Pick $X = m$ that divides the set of points into two equal parts. ($O(N)$)
3. Find the shortest pair in each subproblem. ($T(N/2) + T(N/2)$).

   Let d_l and d_r be the answers. Let d be the minimum of d_l and d_r.
4. Let S be the set of points whose X-coordinate is not farther than d from m. Find the shortest pair within this set. ($T(|S|)$ or $|S|^2$).

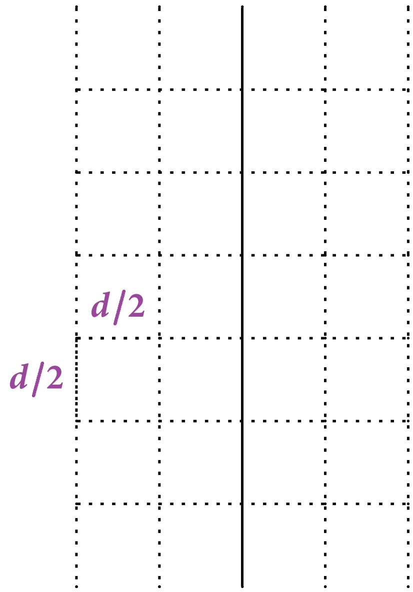
The Algorithm: 1st Attempt

1. Sort the points by their X-coordinates. ($O(N\log N)$)
2. Pick $X = m$ that divides the set of points into two equal parts. ($O(N)$)
3. Find the shortest pair in each subproblem. ($T(N/2) + T(N/2)$).

   Let d_l and d_r be the answers. Let d be the minimum of d_l and d_r.
4. Let S be the set of points whose X-coordinate is not farther than d from m. Find the shortest pair within this set. ($T(|S|)$ or $|S|^2$).

S might contain all the points!

Let us divide up the region between $X = m - d$ and $X = m + d$ into squares of size $d/2$



A Key observation

Observation: There can be at the most one point within any square region.

A Key observation

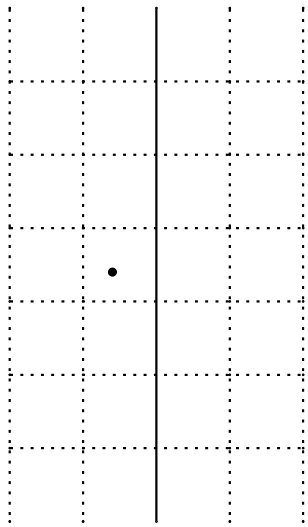
Observation: There can be at the most one point within any square region.

- Each square lies entirely on one side of the partition.
- The farthest points in a square are separated by the diagonal. The length of the diagonal is:

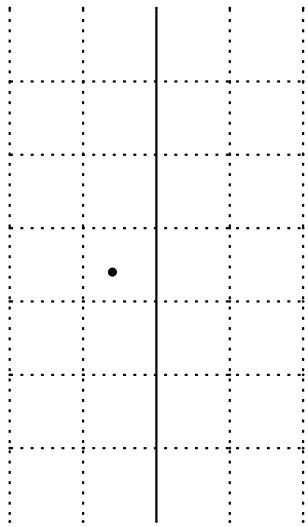
$$\sqrt{2}.d/2 = \frac{d}{\sqrt{2}} < d$$

If there are two in any one square then d cannot be the distance between the closest pair of points in the subproblems.

Candidate pairs

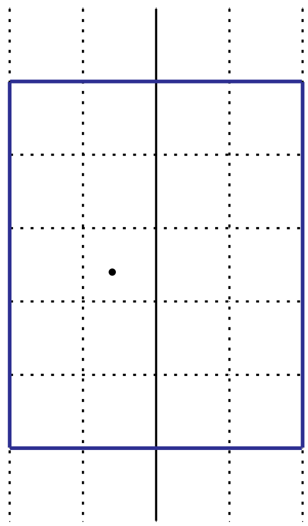


Candidate pairs

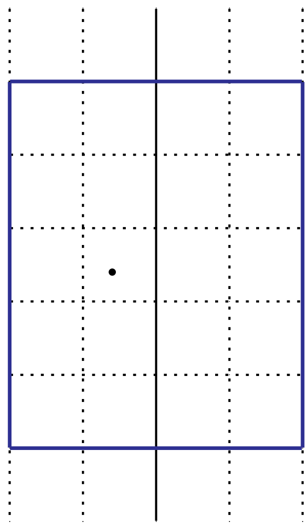


Where can points (in S) closer than d lie?

Candidate pairs



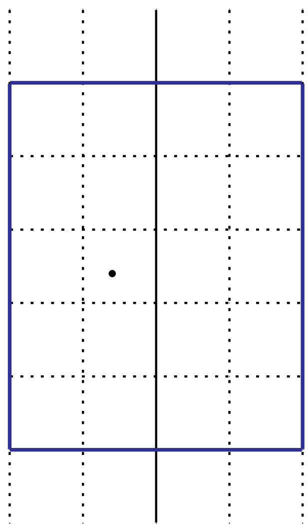
Only within the marked rectangle.



Only within the marked rectangle.

Thus, there are at the most **19** points whose distance from the given point is less than d

How to find the candidate pairs?



Sort the points in S based on the Y-coordinate. Then, a candidate pair cannot be separated by more than 11 in the list!

We can check all candidate pairs within S using at the most $\mathbf{11.|S|}$ calculations.

The Algorithm - 2nd Attempt

1. Sort the points by their X-coordinates. ($O(Nlog(N))$)

The Algorithm - 2nd Attempt

1. Sort the points by their X-coordinates. ($O(N log(N))$)
2. Pick a line $X = m$ that divides the set of points into two equal parts. ($O(N)$)

The Algorithm - 2nd Attempt

1. Sort the points by their X-coordinates. ($O(Nlog(N))$)
2. Pick a line $X = m$ that divides the set of points into two equal parts. ($O(N)$)
3. Find the shortest pair in each subproblem. ($T(N/2) + T(N/2)$).

The Algorithm - 2nd Attempt

1. Sort the points by their X-coordinates. ($O(N log(N))$)
2. Pick a line $X = m$ that divides the set of points into two equal parts. ($O(N)$)
3. Find the shortest pair in each subproblem. ($T(N/2) + T(N/2)$).

   Let d_l and d_r be the answers. Let d be the minimum of d_l and d_r.
4. Let S be the set of points whose X-coordinate is not farther than d from m. ($O(N)$)

The Algorithm - 2nd Attempt

1. Sort the points by their X-coordinates. ($O(N log(N))$)
2. Pick a line $X = m$ that divides the set of points into two equal parts. ($O(N)$)
3. Find the shortest pair in each subproblem. ($T(N/2) + T(N/2)$).

   Let d_l and d_r be the answers. Let d be the minimum of d_l and d_r.
4. Let S be the set of points whose X-coordinate is not farther than d from m. ($O(N)$)
5. Sort the points in S along the Y-axis. ($O(|S| log(|S|))$)

The Algorithm - 2nd Attempt

1. Sort the points by their X-coordinates. ($O(Nlog(N))$)
2. Pick a line $X = m$ that divides the set of points into two equal parts. ($O(N)$)
3. Find the shortest pair in each subproblem. ($T(N/2) + T(N/2)$).

   Let d_l and d_r be the answers. Let d be the minimum of d_l and d_r.
4. Let S be the set of points whose X-coordinate is not farther than d from m. ($O(N)$)
5. Sort the points in S along the Y-axis. ($O(|S|log(|S|))$)
6. Consider all pairs within distance 11 in this list and compute the minimum. ($O(|S|)$)

$$T(N) = 2T(N/2) + c.Nlog(N)$$

Reducing the complexity

Start with two lists P_x and P_y listing the points sorted along X-axis and along Y-axis respectively.

1. Pick $X = m$ that divides the set of points into two equal parts. Use that to construct two subproblems (Q_x, Q_y) and (R_x, R_y). ($O(N)$)

Reducing the complexity

Start with two lists P_x and P_y listing the points sorted along X-axis and along Y-axis respectively.

1. Pick $X = m$ that divides the set of points into two equal parts. Use that to construct two subproblems (Q_x, Q_y) and (R_x, R_y). ($O(N)$)
2. Find the shortest pair in each subproblem. ($T(N/2) + T(N/2)$).

   Let d_l and d_r be the answers. Let d be the minimum of d_l and d_r.

Reducing the complexity

Start with two lists P_x and P_y listing the points sorted along X-axis and along Y-axis respectively.

1. Pick $X = m$ that divides the set of points into two equal parts. Use that to construct two subproblems (Q_x, Q_y) and (R_x, R_y). ($O(N)$)
2. Find the shortest pair in each subproblem. ($T(N/2) + T(N/2)$).

   Let d_l and d_r be the answers. Let d be the minimum of d_l and d_r.
3. Let S be the set of points whose X-coordinate is not farther than d from m. Construct S by scanning the list P_y so that S is sorted along Y-axis by construction. ($O(N)$)

Reducing the complexity

Start with two lists P_x and P_y listing the points sorted along X-axis and along Y-axis respectively.

1. Pick $X = m$ that divides the set of points into two equal parts. Use that to construct two subproblems (Q_x, Q_y) and (R_x, R_y). ($O(N)$)
2. Find the shortest pair in each subproblem. ($T(N/2) + T(N/2)$).

   Let d_l and d_r be the answers. Let d be the minimum of d_l and d_r.
3. Let S be the set of points whose X-coordinate is not farther than d from m. Construct S by scanning the list P_y so that S is sorted along Y-axis by construction. ($O(N)$)
4. Consider all pairs within distance 11 in this list and compute the minimum. ($O(|S|)$)

Reducing the complexity

Start with two lists P_x and P_y listing the points sorted along X-axis and along Y-axis respectively.

1. Pick $X = m$ that divides the set of points into two equal parts. Use that to construct two subproblems (Q_x, Q_y) and (R_x, R_y). ($O(N)$)
2. Find the shortest pair in each subproblem. ($T(N/2) + T(N/2)$).

   Let d_l and d_r be the answers. Let d be the minimum of d_l and d_r.
3. Let S be the set of points whose X-coordinate is not farther than d from m. Construct S by scanning the list P_y so that S is sorted along Y-axis by construction. ($O(N)$)
4. Consider all pairs within distance 11 in this list and compute the minimum. ($O(|S|)$)