1. In this problem we will develop a divide-and-conquer algorithm for the following geometric task.

**Closest Pair**

*Input:* A set of points in the plane, \( \{ p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_n = (x_n, y_n) \} \)

*Output:* The closest pair of points: that is, the pair \( p_i \neq p_j \) for which the distance between \( p_i \) and \( p_j \), that is,

\[
\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2},
\]

is minimized.

For simplicity, assume that \( n \) is a power of two, and that all the \( x \)-coordinates \( x_i \) are distinct, as are the \( y \)-coordinates.

Here’s a high-level overview of the algorithm:

- If we have two points we simply return those points. Otherwise, we:
- Find a value \( x_m \) for which exactly half the points have \( x_i \leq x_m \), and half have \( x_i > x_m \). On this basis, split the points into two groups, \( L \) and \( R \).
- Recursively find the closest pair in \( L \) and in \( R \). Say these pairs are \( \{ p_L, q_L \} \in L \) and \( \{ p_R, q_R \} \in R \), with distances \( d_L \) and \( d_R \) respectively. Let \( d \) be the smaller of these two distances.
- It remains to be seen whether there is a point in \( L \) and a point in \( R \) that are less than distance \( d \) apart from each other. To this end, discard all points with \( x_i < x_m - d \) or \( x_i > x_m + d \) and sort the remaining points by \( y \)-coordinate.
- Now go through this sorted list, and for each point, compute its distance to the seven subsequent points in the list. Let \( \{ p_M, q_M \} \) be the closest pair found in this way.
- The answer is one of the three pairs \( \{ p_L, q_L \}, \{ p_R, q_R \}, \{ p_M, q_M \} \), whichever is closest.
(a) (10 pts) In order to prove the correctness of this algorithm, start by showing the following property: any square of size $d \times d$ in the plane contains at most four points of $L$.

**Answer:** To show this property, we will try to create a square of size $d \times d$ with five points that are at least a distance of $d$ apart. Assume, without loss of generality, that the square covers the range $0 \leq x \leq d$, $0 \leq y \leq d$. We can start by placing one point in the corner $(d,d)$ of this square—as far away from the other points as possible. Now no other point can be placed within a circle of radius $d$ from this point. Thus we need to place four points in an area contained in the triangle formed by $(0,0)$, $(0,d)$, and $(d,0)$. So let us first try to place the four points into this triangle. Again, we put the next point into a corner as far away from the other points as possible—$(0,0)$. We again cannot place any point within the circle of radius $d$ of this point, so we are left with an area contained by the triangle $(0,0)$, $(d/2,d/2)$, and $(d,0)$. Continuing this process, we place a point at $(d,0)$ and are left with only the ability to place a fourth point at $(0,0)$. Therefore, any square of size $d \times d$ in a plane contains at most four points of distance at least $d$ apart (and thus at most four points of $L$).

(b) (10 pts) Now show that the algorithm is correct using induction. The only case which needs careful consideration is when the closest pair is split between $L$ and $R$.

**Answer:** Let our inductive hypothesis be that the algorithm correctly returns the closest pair of points from a set of points of size less than $n$. Our base case occurs when we have two points and the algorithm correctly returns them. By the inductive hypothesis, $(p_L,q_L)$ is the closest pair contained in $L$ and $(p_R,q_R)$ is the closest pair contained in $R$, so the algorithm is correct when the closest pair is not split between $L$ and $R$.

Now assume, for the purpose of obtaining a contradiction, that the closest pair is $\{(x_i,y_i),(x_j,y_j)\} \neq \{p_M,q_M\}$. Then the distance between $p_Z$ and $q_Z$ is less than $d$. This implies that $x_j - x_i < d$, so their $x$-coordinates do not differ from $x_m$ by more than $d$ and neither point can have been discarded. If there were fewer than 7 points with a $y$-coordinate between $y_i$ and $y_j$ that were not discarded then the algorithm would have considered the closest pair and returned it, so there must be 7 points with $y$ coordinates between $y_i$ and $y_j$. This means that there are 9 points in the rectangle covering $x_m - d \leq x \leq x_m + d$, $y_i \leq y \leq y_j$ which is a rectangle of size at most $2d \times d$. Then there must be 5 points in one half of this rectangle, a square of size $d \times d$, contradicting the property we proved in part (a).
(c) (10 pts) Write down the pseudocode for the algorithm, and show that its running time is given by the recurrence:

\[ T(n) = 2T(n/2) + O(n \log n). \]

**ANSWER:**

```plaintext
Closest-Pair(P)
1   if |P| = 2
2       then Return \{P[1], P[2]\}
3   else \( x_m \) ← the median \( x \)-coordinate
4       Partition \( P \) into two pieces:
5           \( L = \{(x_i, x_j) \in P \mid x_i \leq x_m\} \)
6           \( R = \{(x_i, x_j) \in P \mid x_i > x_m\} \)
7           \{p_L, q_L\} ← Closest-Pair(L)
8           \{p_R, q_R\} ← Closest-Pair(R)
9       \( P = L \cup R \)
10       if Distance(p_L, q_L) < Distance(p_R, q_R)
11           then \( c \leftarrow \{p_L, q_L\} \)
12               \( d \leftarrow \text{Distance}(p_L, q_L) \)
13           else \( c \leftarrow \{p_R, q_R\} \)
14               \( d \leftarrow \text{Distance}(p_R, q_R) \)
15       Discard points:
16           \( P = \{(x_i, x_j) \in P \mid x_m - d < x_i < x_m + d\} \)
17       Sort \( P \) by \( y \)-coordinate
18       for \( i \leftarrow 1 \) to |\( P \)|
19           do for \( j \leftarrow i + 1 \) to \( i + 7 \) and \( j < |P| \)
20               do if Distance(P[i], P[j]) < \( d \)
21                   then \( c \leftarrow \{P[i], P[j]\} \)
22                       \( d \leftarrow \text{Distance}(P[i], P[j]) \)
23       Return \( c \)
```

To find the recurrence we just add up all of the steps. The only recursion is on two subproblems with \( n/2 \) points, so we have \( 2T(n/2) \) in the recurrence. Finding the median \( x \)-coordinate, partitioning \( P \) into \( L \) and \( R \), discarding points, and comparing each point with its 7 successors takes linear time. Sorting takes \( O(n \log n) \) time. Thus we have the recurrence \( T(n) = 2T(n/2) + O(n \log n) \).
(d) (10 pts) Show that the solution to this recurrence is O \( (n \log^2 n) \).

**ANSWER:** We prove this using substitution. For our base case, we have that
\[
T(n) \leq c \leq cn \log^2 n \text{ for } n \leq 4 \text{ and a large enough value of } c.
\]
We then have
\[
T(n) = 2T(n/2) + O(n \log n)
\]
\[
\leq 2cn \frac{n}{2} \log^2 \frac{n}{2} + dn \log n
\]
\[
\leq cn \log^2 \frac{n}{2} + dn \log n
\]
\[
\leq cn \log n \log \frac{n}{2} + dn \log n
\]
\[
\leq cn \log n (\log n - \log 2) + dn \log n
\]
\[
\leq cn \log n (\log n - 1) + dn \log n
\]
\[
\leq cn \log^2 n - cn \log n + dn \log n
\]
\[
\leq cn \log^2 n - (c - d)n \log n \quad \text{for } c \geq d
\]

(e) (10 pts) How can the running time be reduced to \( O(n \log n) \)?

**ANSWER:** If we did not have to sort the points by \( y \)-coordinate then we would have the recurrence \( T(n) = 2T(n/2) + O(n) = O(n \log n) \). As with the recursion counting algorithm in class, we can simply keep the points sorted by \( y \)-coordinate and merge the two sorted lists of points \( L \) and \( R \) on Line 7 of the algorithm to get the list of points sorted by \( y \)-coordinate in \( O(n) \) time.