Divide and Conquer Approach for the Closest Pair of Points Problem

1. Problem Statement

Given a set of n points in a two-dimensional plane, the **Closest Pair of Points** problem is to find the pair of points that are closest to each other in terms of Euclidean distance.

Formally, let the set of points be $P = \{p_1, p_2, \ldots, p_n\}$ where each $p_i = (x_i, y_i)$. The task is to compute:

$$\min_{1 \le i < j \le n} d(p_i, p_j)$$

where $d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ is the Euclidean distance.

2. Motivation

- A brute-force approach computes the distance between every pair of points, which requires $\mathcal{O}(n^2)$ time.
- The divide and conquer approach reduces the time complexity to $\mathcal{O}(n \log n)$.

3. Divide and Conquer Strategy

The algorithm follows the general paradigm of divide, conquer, and combine.

Step 1: Preprocessing

- Sort the points according to their x-coordinates. Let this array be P_x.
- Also, prepare a copy sorted by y-coordinates, denoted P_y. This will be useful in the combine step.

Step 2: Divide

- Divide the set of points into two halves by a vertical line passing through the median x-coordinate.
- Let the left subset be Q and the right subset be R.

Step 3: Conquer

- Recursively compute the closest pair distance δ_Q in Q.
- Recursively compute the closest pair distance δ_R in R.
- Let $\delta = \min(\delta_Q, \delta_R)$.

Step 4: Combine (Cross-Pair Check)

- The closest pair might lie across the two halves.
- Construct a strip S of width 2δ centered at the dividing line, containing all points within distance δ from it.
- Sort the points in S by their y-coordinates.
- For each point in S, compute its distance only with the next at most 7 points in the sorted y-order.

Step 5: Result

- The minimum distance among δ_Q, δ_R, and distances found in the strip S is the final answer.

4. Worked Example

Consider the set of points:

$$P = \{(2,3), (12,30), (40,50), (5,1), (12,10), (3,4)\}$$

1. Sort points by x: $[(2,3), (3,4), (5,1), (12,30), (12,10), (40,50)]$.
2. Divide into two halves:
   - Left $Q = \{(2,3), (3,4), (5,1)\}$
   - Right $R = \{(12,30), (12,10), (40,50)\}$
3. Recursively compute:
   - Closest in Q: distance between $(2,3)$ and $(3,4) = \sqrt{2}$
   - Closest in R: distance between $(12,30)$ and $(12,10) = 20$
   - Hence, $\delta = \min(\sqrt{2}, 20) = \sqrt{2}$
4. Construct strip around dividing line ($x = 12$). The strip includes points within δ of $x = 12$, i.e., $(12,30)$ and $(12,10)$.
5. Distances in strip: $d((12,30), (12,10)) = 20$, which is larger than $\sqrt{2}$.
6. Therefore, the closest pair overall is $(2,3)$ and $(3,4)$ with distance $\sqrt{2}$.

5. Correctness and Key Insight

- The key challenge is to ensure that we do not need to compare every point in the strip with every other point.
- Consider the strip S of width 2δ centered at the dividing line. We sort points in S by their y-coordinates.
- For any point p in S, we only need to compare it with the next at most 7 points in the y-order.

Why only 7 neighbors?

- The intuition comes from a **geometric packing argument**.
- Imagine drawing a $\delta \times 2\delta$ rectangle around a point p. If more than 8 points were inside this rectangle with mutual distances at least δ, then by the pigeonhole principle at least two of them must be closer than δ.
- More formally:
  (a) Partition the $\delta \times 2\delta$ rectangle into 8 sub-rectangles of size $\frac{\delta}{2} \times \frac{\delta}{2}$.
  (b) By construction, no two points inside the same sub-rectangle can be at least δ apart.
  (c) Hence, at most 8 points can exist in this region without violating the δ lower bound.
  (d) Therefore, when checking a point p, it suffices to check against the next at most 7 points in y-order.
- This ensures each point requires only $\mathcal{O}(1)$ comparisons, keeping the combine step linear.

6. Complexity Analysis

- Sorting the points initially: $\mathcal{O}(n \log n)$.
- Divide step: $\mathcal{O}(1)$.
- Conquer step: recurrence $T(n) = 2T(n/2) + \mathcal{O}(n)$.
- By the Master Theorem, $T(n) = \mathcal{O}(n \log n)$.
- Hence, the algorithm runs in $\mathcal{O}(n \log n)$ time with $\mathcal{O}(n)$ space.

7. Conclusion

The divide and conquer algorithm for the closest pair of points efficiently reduces the naive quadratic complexity to $\mathcal{O}(n \log n)$ by exploiting geometric properties and recursive decomposition.