Design and Analysis of Algorithms I

Divide and Conquer

Closest Pair I

The Closest Pair Problem

Input: a set $P = \{p_1, \ldots, p_n\}$ of n points in the plane ($\mathbb{R}^2$).

Notation: $d(p_i, p_j)$ = Euclidean distance.

So if $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$,

$$d(p_i, p_j) = \cancel{\sqrt{(x_i - y_i)^2 + (x_j - y_j)^2}} \quad \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Output: a pair $p^*, q^* \in P$ of distinct points that minimize $d(p, q)$ over $p, q \in P$.

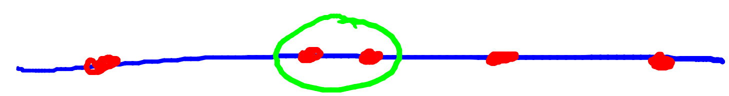
Tim Roughgarden

Initial Observations

Assumption: (for convenience) all points have distinct x-coordinates, distinct y-coordinates.

Brute-force search: takes $\Theta(n^2)$ time.

1-D version of Closest Pair:



① sort points ($O(n \log n)$ time)
② return closest pair of adjacent points ($O(n)$ time)

Goal: $O(n \log n)$ time algorithm for 2-D version.

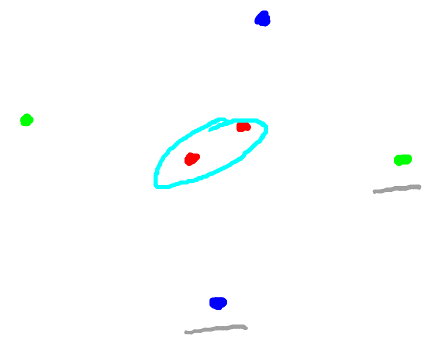
High-Level Approach

(1) make copies of points sorted by x-coordinate (P_x) and by y-coordinate (P_y).

[$O(n \log n)$ time]

(but this is not enough!)

(2) use Divide + Conquer



Tim Roughgarden

The Divide and Conquer Paradigm

1. DIVIDE into smaller subproblems.
2. CONQUER subproblems recursively.
3. COMBINE solutions of subproblems into one for the original problem.

ClosestPair(P_x, P_y)

base case omitted

1. let Q = left half of P, R = right half of P. form Q_x, Q_y, R_x, R_y [takes $O(n)$ time]
2. (p_1, q_1) = ClosestPair(Q_x, Q_y)
3. (p_2, q_2) = ClosestPair(R_x, R_y)
4. (p_3, q_3) = ClosestSplitPair(P_x, P_y)
5. return best of (p_1, q_1), (p_2, q_2), (p_3, q_3)

Suppose we can correctly implement the ClosestSplitPair subrouine in $O(n)$ time. What will be the overall running time of the Closest Pair algorithm? (Choose the smallest upper bound that applies.)

- $O(n)$
- $O(n \log n)$
- $O(n (\log n)^2)$
- $O(n^2)$

KEY IDEA: only need to bother computing the closest split pair in "unlucky case" where its distance is less than $d(p_1, q_1)$ — result of 1st recursive call and $d(p_2, q_2)$. — result of 2nd recursive call

ClosestPair(P_x, P_y)

base case omitted

1. Let Q = left half of P, R = right half of P. Form Q_x, Q_y, R_x, R_y [takes $O(n)$ time]
2. (p_1, q_1) = ClosestPair(Q_x, Q_y)
3. (p_2, q_2) = ClosestPair(R_x, R_y)
4. Let $\delta = \min\{d(p_1,q_1), d(p_2,q_2)\}$.
5. (p_3, q_3) = ClosestSplitPair(P_x, P_y, δ)
6. return best of $(p_1,q_1), (p_2,q_2), (p_3,q_3)$.

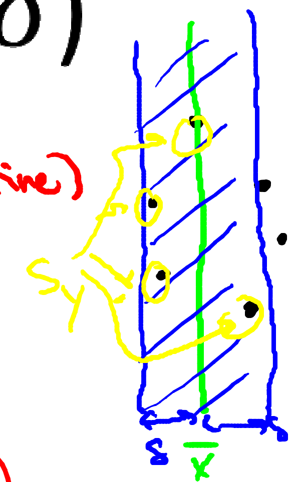
will describe next

Requirements

1. $O(n)$ time
2. correct whenever closest pair of P is a split pair

Tim Roughgarden

ClosestSplitPair(P_x, P_y, δ)



Let $\bar{x}$ = biggest x-coordinate in left of P. (O(1) time)

Let S_y = points of P with x-coordinate in $[\bar{x}-\delta, \bar{x}+\delta]$, sorted by y-coordinate. (O(n) time)

Initialize best = δ, best pair = NULL.

For i = 1 to $|S_y| - 7$

   For j = 1 to 7

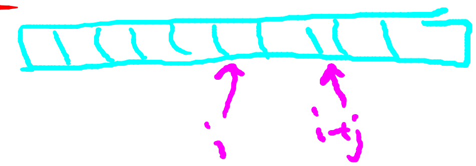
      Let p, q = i^{th}, $(i+j)^{th}$ points of S_y.

      If d(p,q) < best

         best pair = (p,q), best = d(p,q)

O(1) time

O(n) time



At end, return best pair

Tim Roughgarden

Correctness Claim

$\min\{d(p_1,q_1), d(p_2,q_2)\}$

Claim: Let $p \in Q$, $q \in R$ be a split pair with $d(p,q) < \delta$.

Then: (A) p and q are members of S_y.

(B) p and q are at most 7 positions apart in S_y.

Corollary 1: If the closest pair of P is a split pair, then ~~Count~~ Closest Split Pair finds it.

assuming Claim is true!

Corollary 2: ClosestPair is correct, and runs in $O(n \log n)$ time.