Divide and Conquer

Closest Pair I

Design and Analysis of Algorithms I

The Closest Pair Problem

<u>Input</u> : a set $P = \{p_1, ..., p_n\}$ of n points in the plane R^2.

<u>Notation</u> : $d(p_i, p_j)$ = Euclidean distance

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

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

<u>Output</u> : a pair $p*, q* \in P$ of distinct points that minimize d(p,q) over p,q in the set 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 :

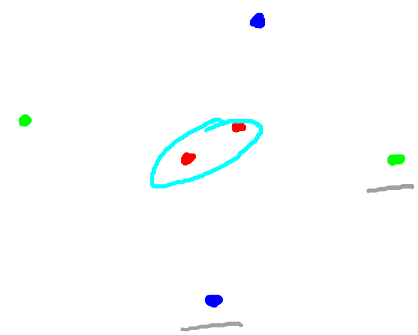
1. Sort points (O(nlog(n)) time)
2. Return closest pair of adjacent points (O(n) time)

Goal : O(nlog(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(nlog(n)) time]

   (but this is not enough!)



2. Use Divide+Conquer

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.

Tim Roughgarden

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)

Ter
ver

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

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

<u>Key Idea</u> : only need to bother computing the closest split pair in “unlucky case” where its distance is less than $d(p_1,q_1)$ and $d(p_2,q_2)$.

Result of 1st recursive call

Result of 2nd recursive call

ClosestPair(P_x, P_y)

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)

BASE CASE OMITTED

WILL DESCRIBE NEXT

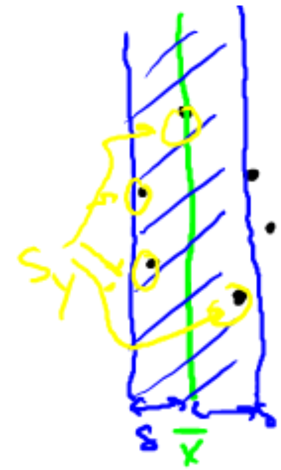
Requirements

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
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) < δ

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

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

Corollary1 : If the closest pair of P is a split pair, then the ClosestSplitPair finds it.

Corollary2 ClosestPair is correct, and runs in O(nlog(n)) time.

Assuming claim is true!