1.5em 0pt

Divide and Conquer Method

Dr. Raghunath Reddy M

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

Dept. of Computer Science Engineering,
Bennett University, Greater Noida

January 29, 2020

Closest pair of points

Problem Definition

Input: Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of points in the plane where each $p_i = (x_i, y_i)$ for real numbers x_i and y_i.
Output: Find a pair of points p_i and p_j such that the Euclidean distance between p and p_j, $d(p_i, p_j)$ is minimum.

Brute-force algorithm

Check all possible ${}^nC_2$ pairs and pick the best one.
Time : $O(n^2)$

Question

Can we do it in better time, like $O(n \log n)$-time ?

Divide and Conquer Approach

Divide and Conquer Approach

How to achieve?

- Find the closest pair among the points in the "left half " of P,
- Find the closest pair among the points in the "right half " of P, and

Divide and Conquer Approach

How to achieve?

- Find the closest pair among the points in the "left half" of P,
- Find the closest pair among the points in the "right half" of P, and
- use the information to get the overall solution in $O(n)$-time.

Divide and Conquer Approach

How to achieve?

- Find the closest pair among the points in the "left half " of P,
- Find the closest pair among the points in the "right half " of P, and
- use the information to get the overall solution in $O(n)$-time.

- As the sizes of left and right halves are almost the same, we have

$$T(n) = 2T(n/2) + O(n)$$

Thus, $T(n) = O(n \log n)$

Divide and Conquer Approach

How to achieve?

- Find the closest pair among the points in the "left half" of P,
- Find the closest pair among the points in the "right half" of P, and
- use the information to get the overall solution in $O(n)$-time.

- As the sizes of left and right halves are almost the same, we have

$$T(n) = 2T(n/2) + O(n)$$

Thus, $T(n) = O(n \log n)$

Main task

How to get the overall solution in $O(n)$-time when we have the solutions of left and right halves.

Notations

- For any set of points $P' \subseteq P$, let
  - P'_x be the points in P' sorted in the increasing x-coordinates, and
  - P'_y be the points in P' sorted in the increasing y-coordinates.

Notations

- For any set of points $P' \subseteq P$, let
  - P'_x be the points in P' sorted in the increasing x-coordinates, and
  - P'_y be the points in P' sorted in the increasing y-coordinates.

Lemma 0.1.

For any given set $P' \subseteq P$, one can obtain P'_x and P'_y in $O(n \log n)$ time.

Notations

- For any set of points $P' \subseteq P$, let
  - P'_x be the points in P' sorted in the increasing x-coordinates, and
  - P'_y be the points in P' sorted in the increasing y-coordinates.

Lemma 0.1.

For any given set $P' \subseteq P$, one can obtain P'_x and P'_y in $O(n \log n)$ time.

- Further, let P_x and P_y be the sets of points in P which are sorted with respect to x-coordinate and y-coordinate respectively.

Notations

- For any set of points $P' \subseteq P$, let
  - P'_x be the points in P' sorted in the increasing x-coordinates, and
  - P'_y be the points in P' sorted in the increasing y-coordinates.

Lemma 0.1.

For any given set $P' \subseteq P$, one can obtain P'_x and P'_y in $O(n \log n)$ time.

- Further, let P_x and P_y be the sets of points in P which are sorted with respect to x-coordinate and y-coordinate respectively.

- Closest-Pair(P)
  - Construct P_x and P_y ($O(n \log n)$-time)
  - $(p_0^*, p_1^*) =$ Closest-Pair-Rec(P_x, P_y)

Setting up the recursion

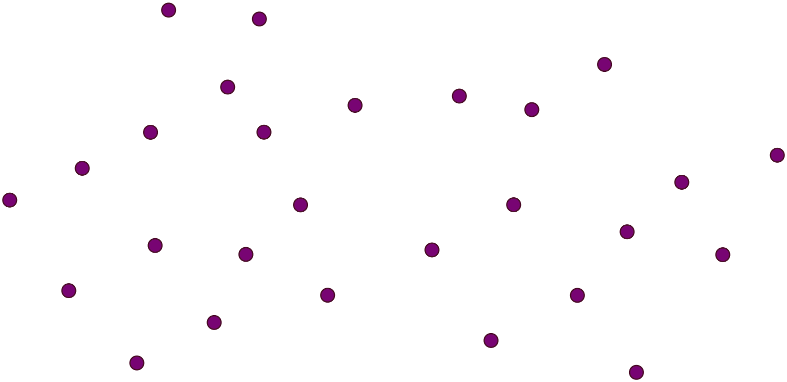


Figure: Points sorted with respect to x-coordinates.

Question

How to split the problem into two (almost) equal size sub-problems ?

How to split into two sub-problems

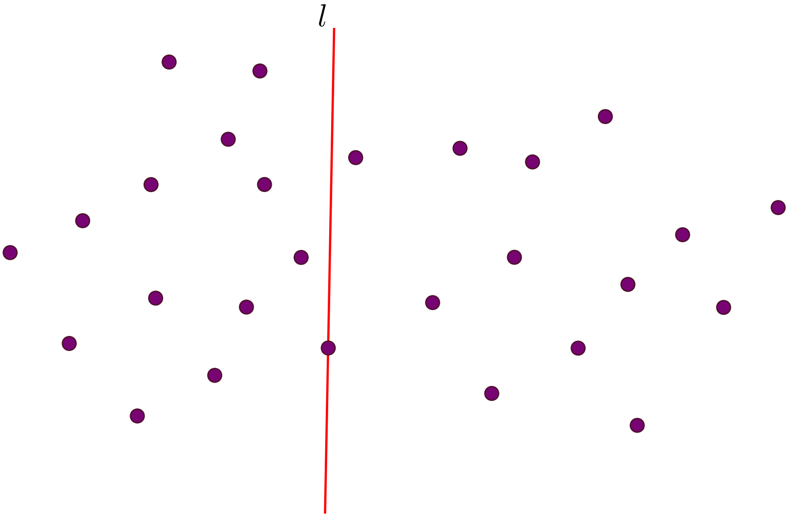


Figure: Vertical line l splits the input into two equal halves.

New sub-problems

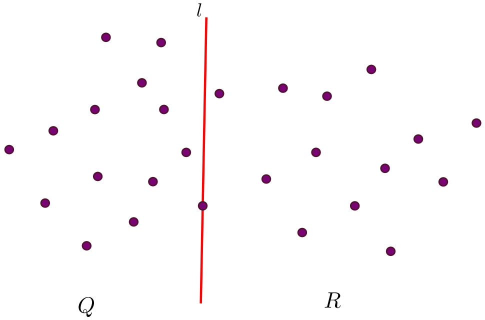


Figure: Two new set of points Q (left half) and R (right half).

New sub-problems

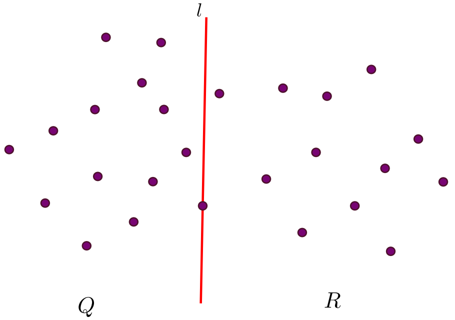


Figure: Two new set of points Q (left half) and R (right half).

Question ?

What is the time to compute Q and R.

New sub-problems

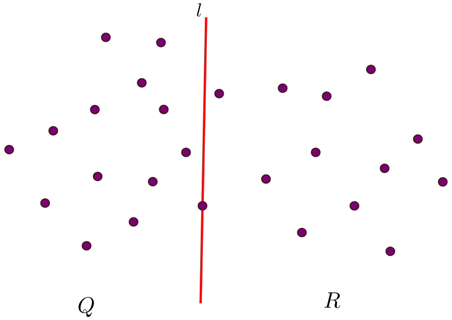


Figure: Two new set of points Q (left half) and R (right half).

Question ?

What is the time to compute Q and R.

Ans: $O(n)$

Q is the collection of first $\lceil n/2 \rceil$ points in P_x and R is the collection of last $\lfloor n/2 \rfloor$ points in P_x.

New sub-problems

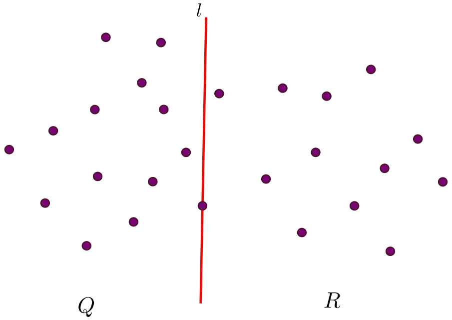


Figure: Two new set of points Q (left half) and R (right half).

Question ?

What is the time to compute Q and R.

Ans: $O(n)$

Q is the collection of first $\lceil n/2 \rceil$ points in P_x and R is the collection of last $\lfloor n/2 \rfloor$ points in P_x.

Four new sets of points

Define Q_x, Q_y, R_x, and R_y as before.

Next step

Suppose we have obtained the closest pairs in Q and R, by recursively solving the sub-problems.

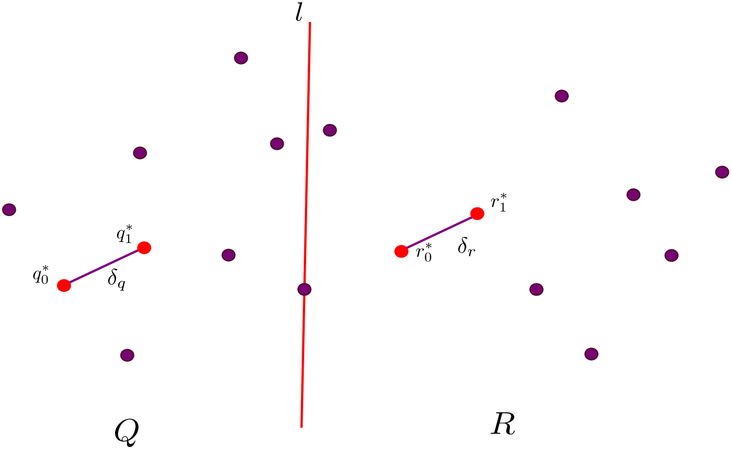


Figure: Solutions of Q and R given in red color.

Algorithm

- Closest-Pair-Rec(P_x, P_y)
  - If $|P_x| \leq 3$ then
    - find the closest pair by measuring all pairwise distances and return the pair.
  - Else

Algorithm

- Closest-Pair-Rec(P_x, P_y)
  - If $|P_x| \leq 3$ then
    - find the closest pair by measuring all pairwise distances and return the pair.
  - Else
    - Construct Q_x, Q_y, R_x, R_y ($O(n)$-time)
    - $(q_0^*, q_1^*) =$ Closest-Pair-Rec(Q_x, Q_y)
    - $(r_0^*, r_1^*) =$ Closest-Pair-Rec(R_x, R_y)

Algorithm

- Closest-Pair-Rec(P_x, P_y)
  - If $|P_x| \leq 3$ then
    - find the closest pair by measuring all pairwise distances and return the pair.
  - Else
    - Construct Q_x, Q_y, R_x, R_y ($O(n)$-time)
    - (q_0^*, q_1^*) = Closest-Pair-Rec(Q_x, Q_y)
    - (r_0^*, r_1^*) = Closest-Pair-Rec(R_x, R_y)
    - (p_0^*, p_1^*) =CombineQR($q_0^*, q_1^*, r_0^*, r_1^*$)
    - Return (p_0^*, p_1^*)

Time complexity ?

Algorithm

- Closest-Pair-Rec(P_x, P_y)
  - If $|P_x| \leq 3$ then
    - find the closest pair by measuring all pairwise distances and return the pair.
  - Else
    - Construct Q_x, Q_y, R_x, R_y ($O(n)$-time)
    - $(q_0^*, q_1^*) =$ Closest-Pair-Rec(Q_x, Q_y)
    - $(r_0^*, r_1^*) =$ Closest-Pair-Rec(R_x, R_y)
    - $(p_0^*, p_1^*) =$CombineQR($q_0^*, q_1^*, r_0^*, r_1^*$)
    - Return (p_0^*, p_1^*)

Time complexity ?

$T(n) = 2T(n/2) + f(n)$ where $f(n)$ depends on the time taken by the function CombineQR(......).

What should be the goal

Algorithm

- Closest-Pair-Rec(P_x, P_y)
  - If $|P_x| \leq 3$ then
    - find the closest pair by measuring all pairwise distances and return the pair.
  - Else
    - Construct Q_x, Q_y, R_x, R_y ($O(n)$-time)
    - $(q_0^*, q_1^*) =$ Closest-Pair-Rec(Q_x, Q_y)
    - $(r_0^*, r_1^*) =$ Closest-Pair-Rec(R_x, R_y)
    - $(p_0^*, p_1^*) =$CombineQR($q_0^*, q_1^*, r_0^*, r_1^*$)
    - Return (p_0^*, p_1^*)

Time complexity ?

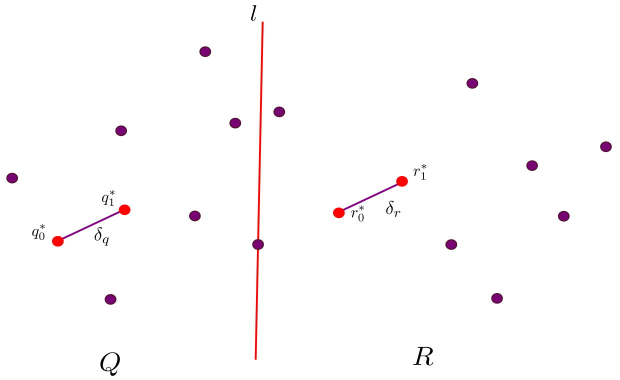
$T(n) = 2T(n/2) + f(n)$ where $f(n)$ depends on the time taken by the function CombineQR(......).

What should be the goal

$f(n)$ should be $O(n)$ to get $T(n) = O(n \log n)$.

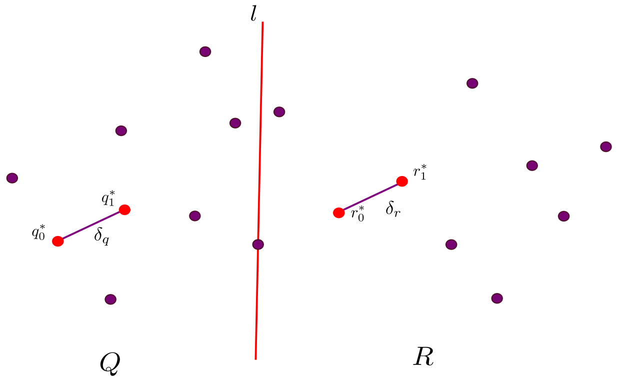
Combined solution

- Let $\delta = \min\{\delta_q, \delta_r\}$.



Combined solution

- Let $\delta = \min\{\delta_q, \delta_r\}$.



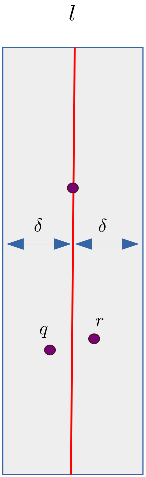
Question

Are there points $q \in Q$ and $r \in R$ such that $d(q, r) < \delta$?

Some observations about closest pair

Some observations about closest pair

- Suppose that there are two points $q \in Q$ and $r \in R$ such that $d(q, r) < \delta$. Then,



The fact about the vertical strip

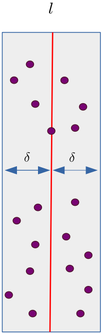


Figure: The strip may contain many points.

The fact about the vertical strip

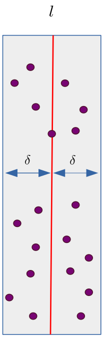


Figure: The strip may contain many points.

Set of points inside the strip

Let $S \subseteq P$ be the set of all points inside the strip of width 2δ around line l.

The fact about the vertical strip

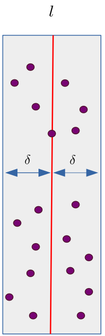


Figure: The strip may contain many points.

Set of points inside the strip

Let $S \subseteq P$ be the set of all points inside the strip of width 2δ around line l.

- Compute S_y, the points in S sorted in increasing order with respect to the y-coordinates.

The fact about the vertical strip

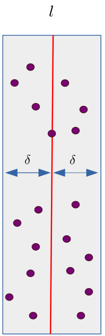


Figure: The strip may contain many points.

Set of points inside the strip

Let $S \subseteq P$ be the set of all points inside the strip of width 2δ around line l.

- Compute S_y, the points in S sorted in increasing order with respect to the y-coordinates.

Lemma 0.2.

There exists $q \in Q$ and $r \in R$ such that $d(q, r) < \delta$ if and only if there exist $s, s' \in S$ for which $d(s, s') < \delta$.

Some interesting result about s and s'

Lemma 0.3.

If $s, s' \in S$ have the property that $d(s, s') < \delta$, then s and s' are within 15 *positions of each other in the sorted list S_y.*

Algo for CombineQR(.....)

- CombineQR($q_0^*, q_1^*, r_0^*, r_1^*$)

Algo for CombineQR(.....)

- CombineQR($q_0^*, q_1^*, r_0^*, r_1^*$)
  - $\delta_q = d(q_0^*, q_1^*)$ and $\delta_r = d(r_0^*, r_1^*)$
  - $\delta = \min\{\delta_q, \delta_r\}$
  - $x^* =$ the x-coordinate of the rightmost point in Q
  - $S =$ points in P within δ distance from $x = x^*$ ($O(n)$-time).

Algo for CombineQR(.....)

- CombineQR($q_0^*, q_1^*, r_0^*, r_1^*$)
  - $\delta_q = d(q_0^*, q_1^*)$ and $\delta_r = d(r_0^*, r_1^*)$
  - $\delta = \min\{\delta_q, \delta_r\}$
  - x^* = the x-coordinate of the rightmost point in Q
  - S = points in P within δ distance from $x = x^*$ ($O(n)$-time).
  - Construct S_y ($O(n)$-time)
  - For each point $s \in S_y$, compute the distance from s to each of next 15 points in S_y ($O(n)$-time)
  - Let s, s' be the pair achieving minimum of these distances ($O(n)$-time)

Algo for CombineQR(.....)

- CombineQR($q_0^*, q_1^*, r_0^*, r_1^*$)
  - $\delta_q = d(q_0^*, q_1^*)$ and $\delta_r = d(r_0^*, r_1^*)$
  - $\delta = \min\{\delta_q, \delta_r\}$
  - x^* = the x-coordinate of the rightmost point in Q
  - S = points in P within δ distance from $x = x^*$ ($O(n)$-time).
  - Construct S_y ($O(n)$-time)
  - For each point $s \in S_y$, compute the distance from s to each of next 15 points in S_y ($O(n)$-time)
  - Let s, s' be the pair achieving minimum of these distances ($O(n)$-time)
  - If $d(s, s') < \delta$ then
    - Return (s, s')
  - Else if $\delta_q < \delta_r$ then
    - Return (q_0^*, q_1^*)
  - Else Return (r_0^*, r_1^*)

Lemma 0.4.

CombineQR($q_0^*, q_1^*, r_0^*, r_1^*$) takes $O(n)$ time to return the solution.