Lect 02. Divide and Conquer

Spring, 2020

School of Computer Science & Engineering

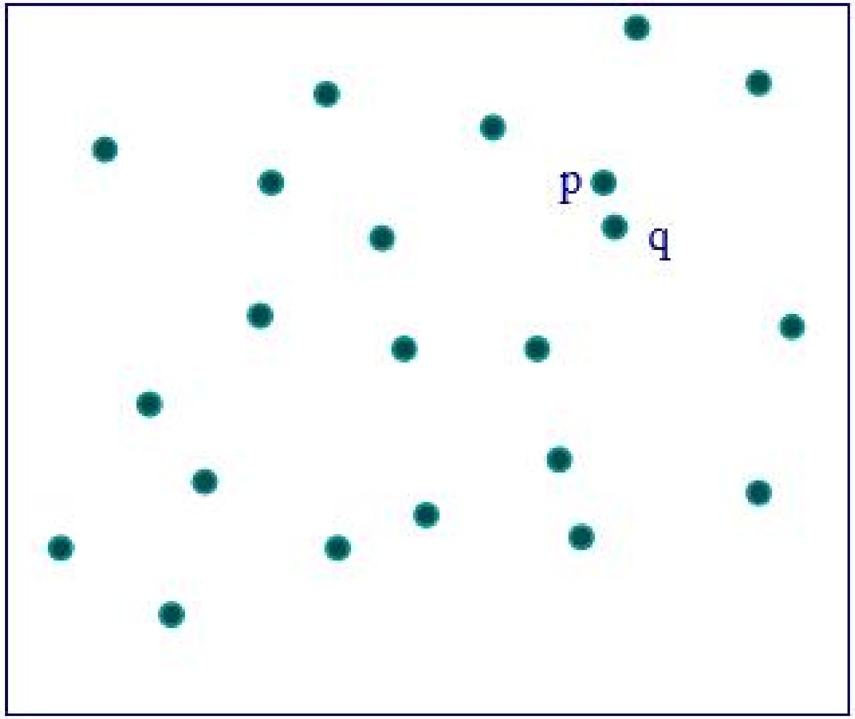
Pusan National University

2

Closest Pair Problem

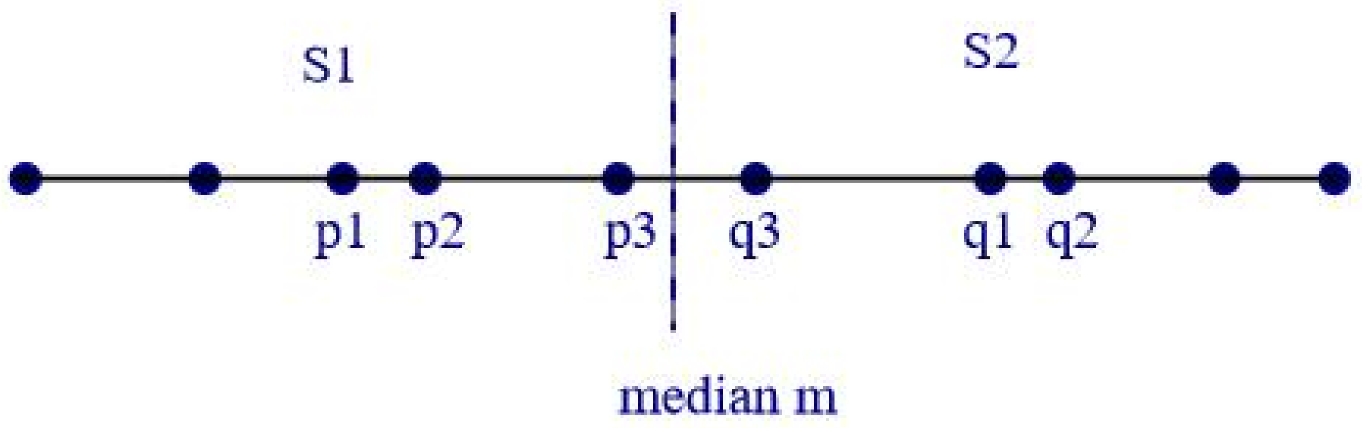
Closest Pair Problem

- Problem : Given n points in d-dimensions, find two whose mutual distance is smallest.
- Input : n points
- Output : closest pair (p, q)



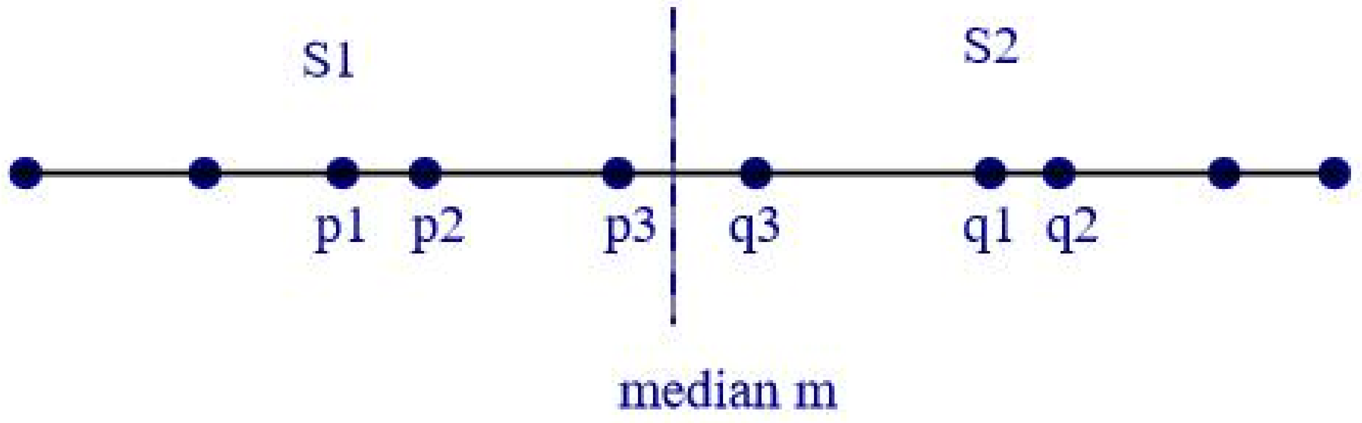
Closest Pair Problem : 1-Dimension Problem

- Problem : Given n points in 1-dimension, find two whose mutual distance is smallest.
- Solution : can be solved in $O(n\log n)$ using sorting – does not generalize to extended dimension



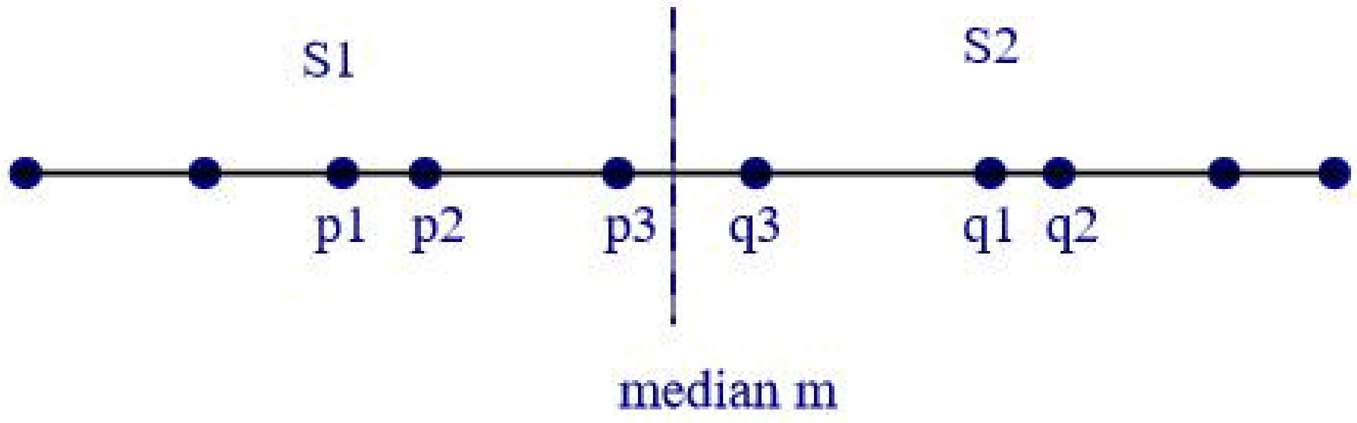
Closest Pair Problem : 1-Dimension Problem (2)

- Problem : Given n points in 1-dimension, find two whose mutual distance is smallest.
- Solution : Divide and Conquer
  - Divide the points S into two sets S_1, S_2 by some x-coordinate so that $p < q$ for all $p \in S_1$ and $q \in S_2$.
  - Recursively compute closest pair (p_1, p_2) in S_1 and (q_1, q_2) in S_2.



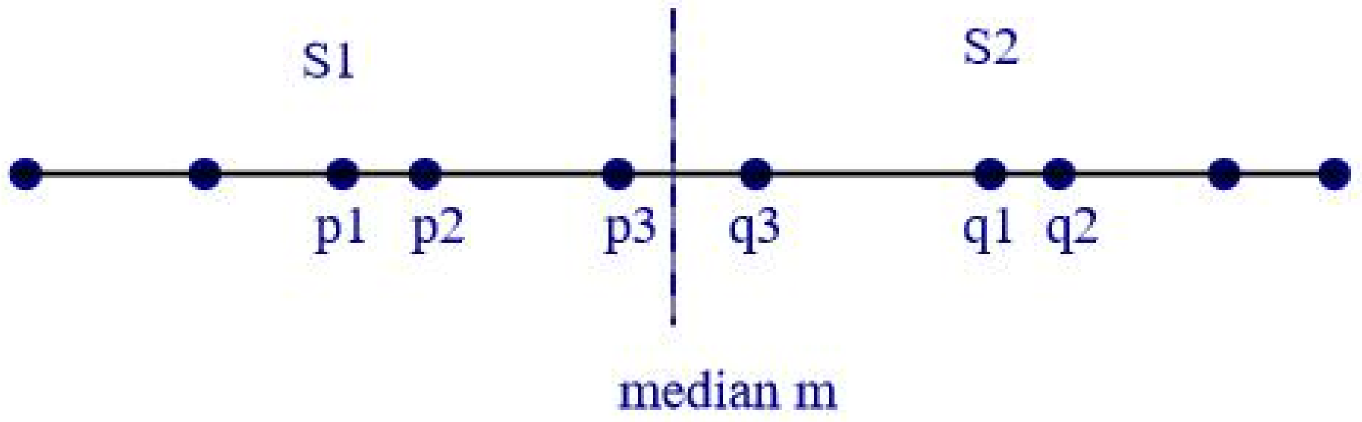
Closest Pair Problem : 1-Dimension Problem (3)

- Problem : Given n points in 1-dimension, find two whose mutual distance is smallest.
- Solution : Divide and Conquer
  - Let δ be the smallest separation found so far:
  - $\delta = \min(|p_2 - p_1|, |q_2 - q_1|)$
  - The closest pair is $\{p_1, p_2\}$, or $\{q_1, q_2\}$, or some $\{p_3, q_3\}$ where $p_3 \in S_1$ and $q_3 \in S_2$.



Closest Pair Problem: 1-Dimension Problem (4)

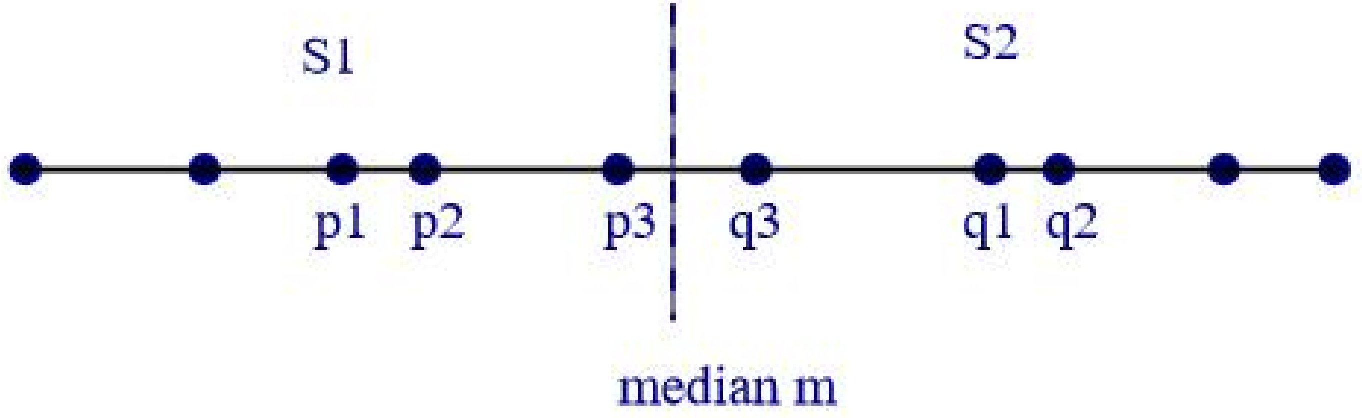
- Problem : Given n points in 1-dimensions, find two whose mutual distance is smallest.
- Key point : If m is the dividing coordinate, then p_3, q_3 must be within δ of m.



Closest Pair Problem: 1-Dimension Problem (5)

- Problem : Given n points in 1-dimensions, find two whose mutual distance is smallest.
- How many points of S_1 can lie in the interval $(m-\delta,\ m]$?

  By definition of δ, at most one. Same holds for S2.



Closest Pair Problem: 1-Dimension Problem (5)

Closest-Pair(S)

If |S| = 1, $\delta = \infty$

If |S| = 2, $\delta = |p_2 - p_1|$

else

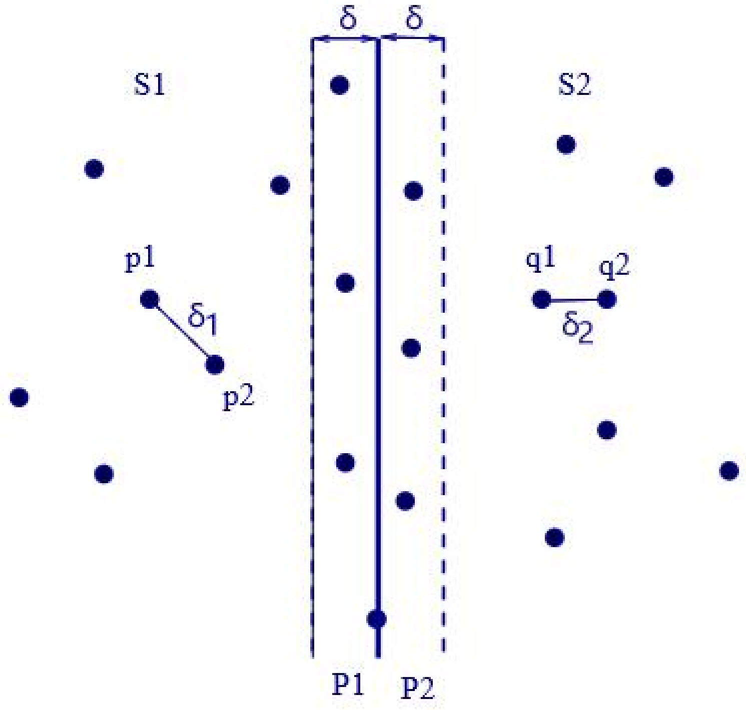
1. Let m = median(S)
2. Divide S into S1, S2 at m
3. δ_1 = Closest-Pair(S1)
4. δ_2 = Closest-Pair(S2)
5. δ_{12} is minimum distance across the cut
6. return $\delta = \min(\delta_1, \delta_2, \delta_{12})$

Time complexity

- Recurrence is $T(n) = 2T(n/2)+O(n)$, which solves to $T(n) = O(n\log n)$.

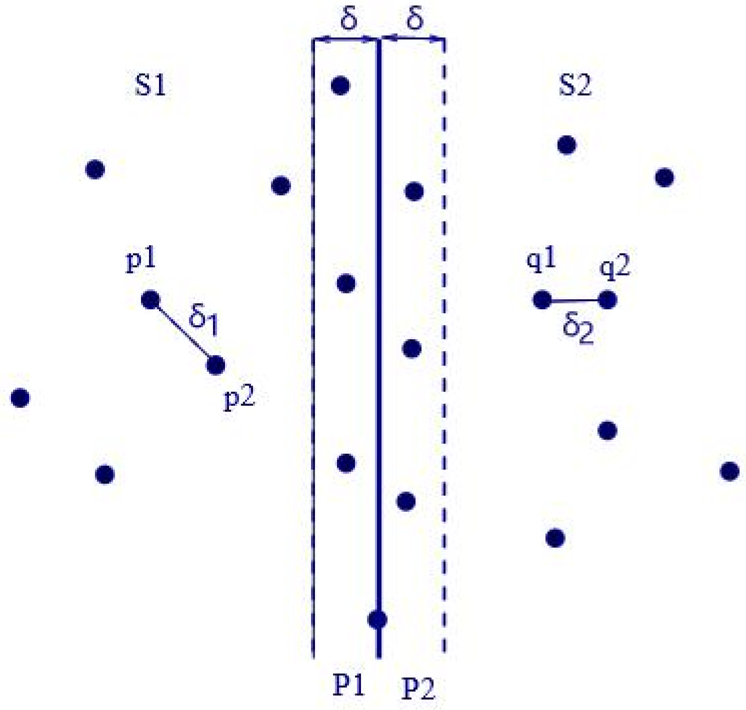
Closest Pair Problem: 2-Dimension Problem (1)

- Problem : Given n points in 2-dimension, find two whose mutual distance is smallest.
- (method)
  - Partition S into S_1, S_2 by vertical line l defined by median x-coordinate in S.
  - Recursively compute closest pair distance δ_1 and δ_2. Set $\delta = \min(\delta_1, \delta_2)$.
  - Compute the closest pair with one point each in S_1 and S_2.



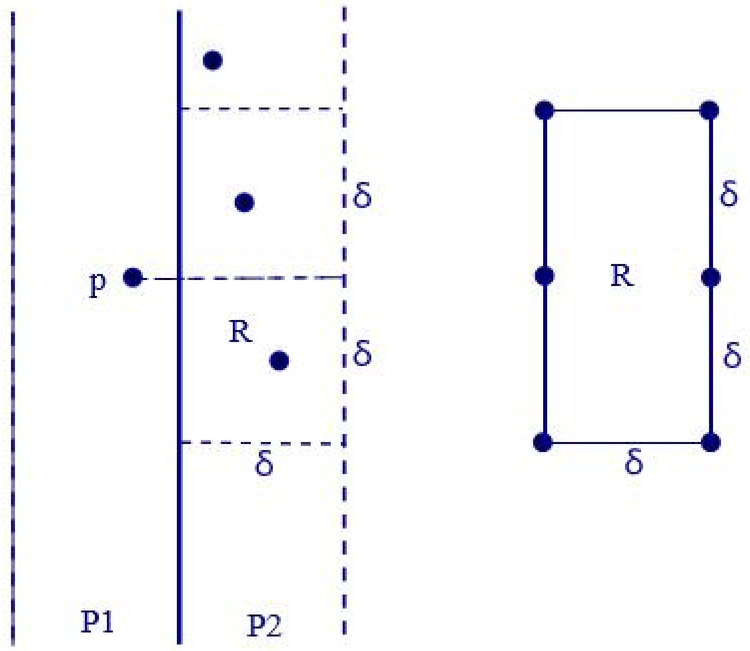
Closest Pair Problem: 2-Dimension Problem (2)

- Problem : Given n points in 2-dimension, find two whose mutual distance is smallest.
- In each candidate pair (p, q), where $p \in S_1$ and $q \in S_2$, the points p, q must both lie within δ of l.
- (complication case) It's entirely possible that all $n/2$ points of S_1 (and S_2) lie within δ of l. => It would be require $n^2/4$ cal…



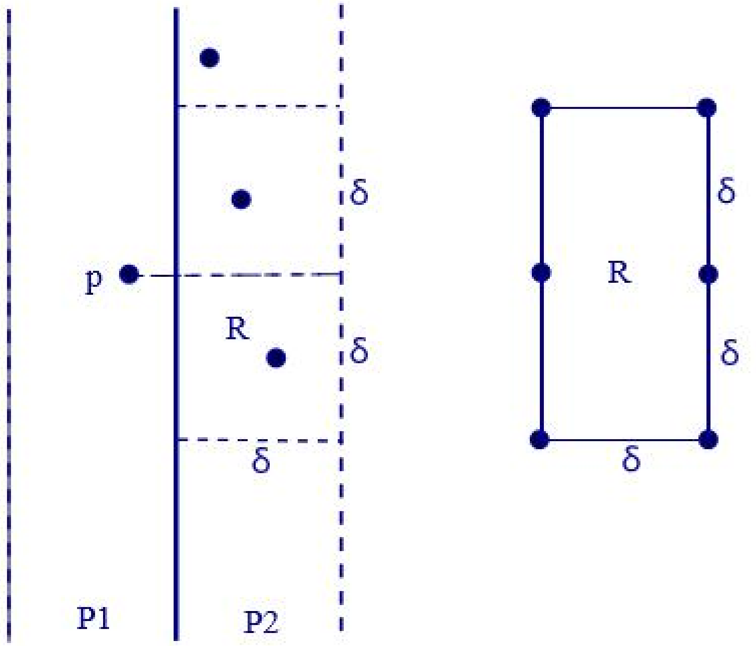
Closest Pair Problem: 2-Dimension Problem (3)

- Conquer step : Consider a point p∈S1. All points of S2 within distance δ of p must lie in a δ×2δ rectangle R.
  - How many points can be inside R if each pair is at least δ apart?
  - In 2D, this number is at most 6.
  - Only need to perform 6×n/2 distance comparisons.



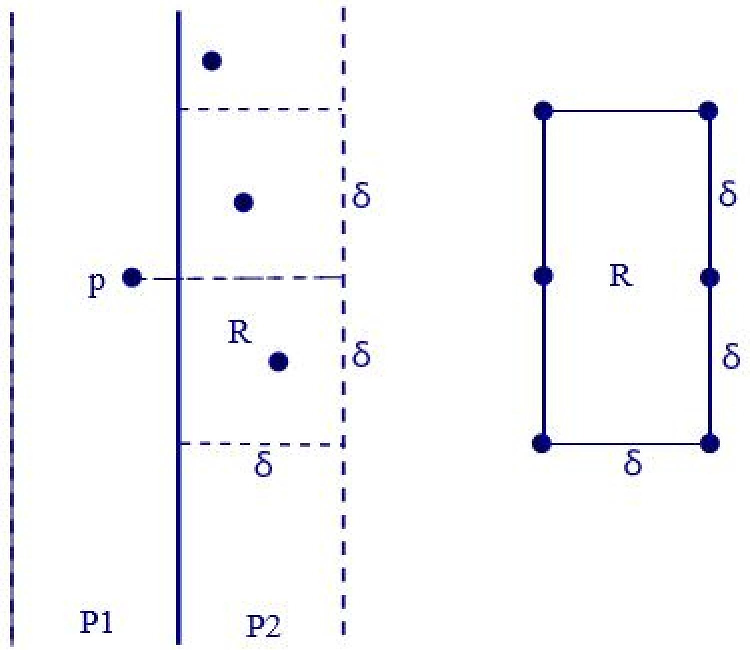
Closest Pair Problem: 2-Dimension Problem (4)

- In order to determine at most 6 potential mates of p, project p and all point s of p_2 onto line l.
- Pick out points whose projection is within δ of p; at most six.
- We can do this for all p, by walking sorted lists of p_1 and p_2, in total $O(n)$ time.
- The sorted lists for P_1, P_2 can be obtained from pre-sorting of S_1, S_2.



Closest Pair Problem: 2-Dimension Problem (5)

- Final recurrence is $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$, which solves to $T(n) = O(n\log n)$.



16

Thank you for your atten tion !