Advanced Algorithms & Data Structures

Department of CSE

ADVANCED ALGORITHMS AND DATA STRUCTURES
23CS03HF

Topic:

Closest Pair

AIM OF THE SESSION

To familiarize students with the concept of Closest Pair

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1.Demonstrate :- Closest Pair.

2.Describe :- Solving of Closest pair using Divide and Conquer

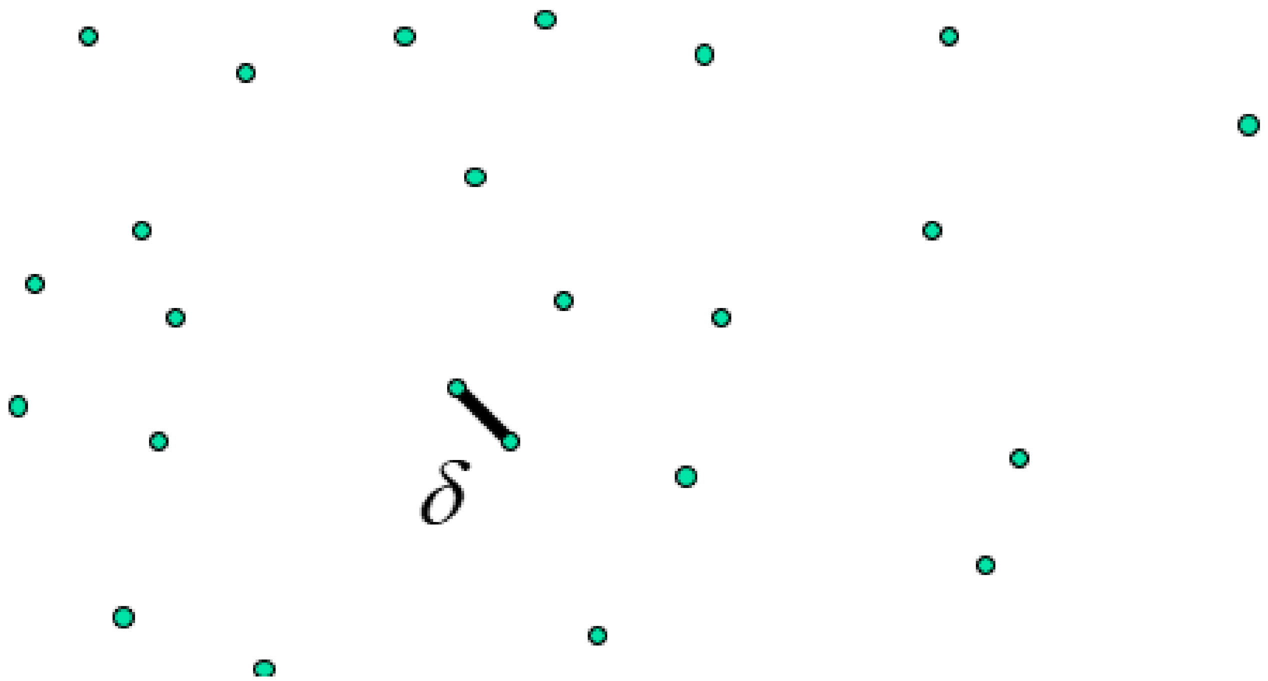
LEARNING OUTCOMES

At the end of this session, you should be able to:

1. Define :- Closest Pair.
2. Describe :- Identifying Closest pair using Divide and Conquer
3. Summarize:- Description about the closest pair and time complexity of closest pair

Closest Pair

Given a set S = {*p1*, *p2*, ..., *pn*} of *n* points in the plane find the two points of *S* whose distance is the smallest.



Closest Pair-Divide & Conquer

- Divide the problem into two equal-sized sub problems
- Solve those sub problems recursively
- Merge the sub problem solutions into an overall solution

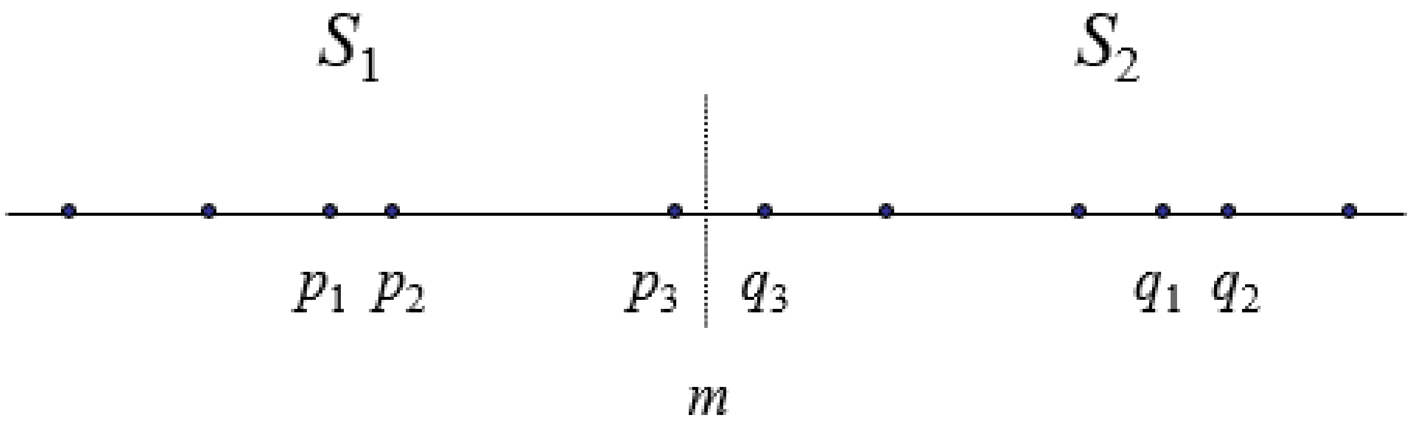
Closest Pair-Divide & Conquer

- Assume that we have solutions for sub problems S1, S2.

- How can we merge in a time-efficient way?
  - The closest pair can consist of one point from S_1 and another from S_2
  - Testing all posibilities requires: $O(n/2) \cdot O(n/2) \in O(n^2)$
  - Not good enough

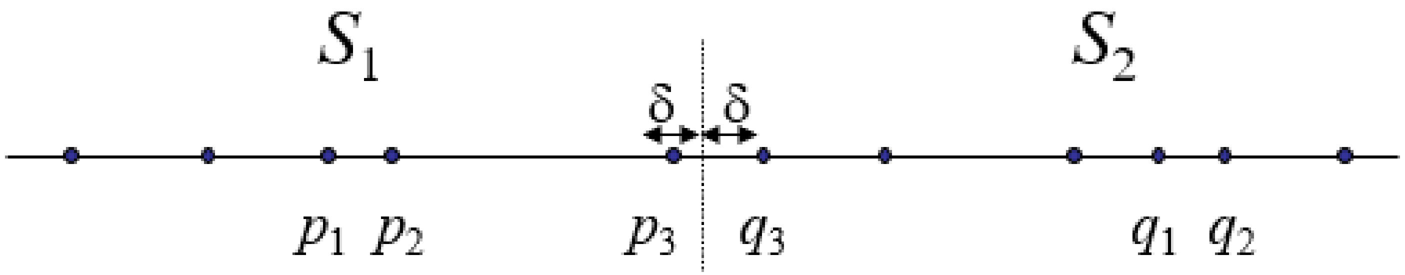
Closest Pair-Divide & Conquer dimension d=1

- Partition S, a set of points on a line, into two sets S_1 and S_2 at some point m such that for every point $p \in S_1$ and $q \in S_2$, $p < q$.
- Solve Closest Pair recursively on S_1 and S_2
- Let δ_1 be the smallest distance in S_1 and δ_2 in S_2
- Let δ be the smallest distance found so far: $\delta = \min(\delta_1, \delta_2)$
- What is the closest pair in S?
  - Either $\{p1, p2\}$ or $\{q1, q2\}$ or some $\{p3, q3\}$ with $p3 \in S_1$ and $q3 \in S_2$.



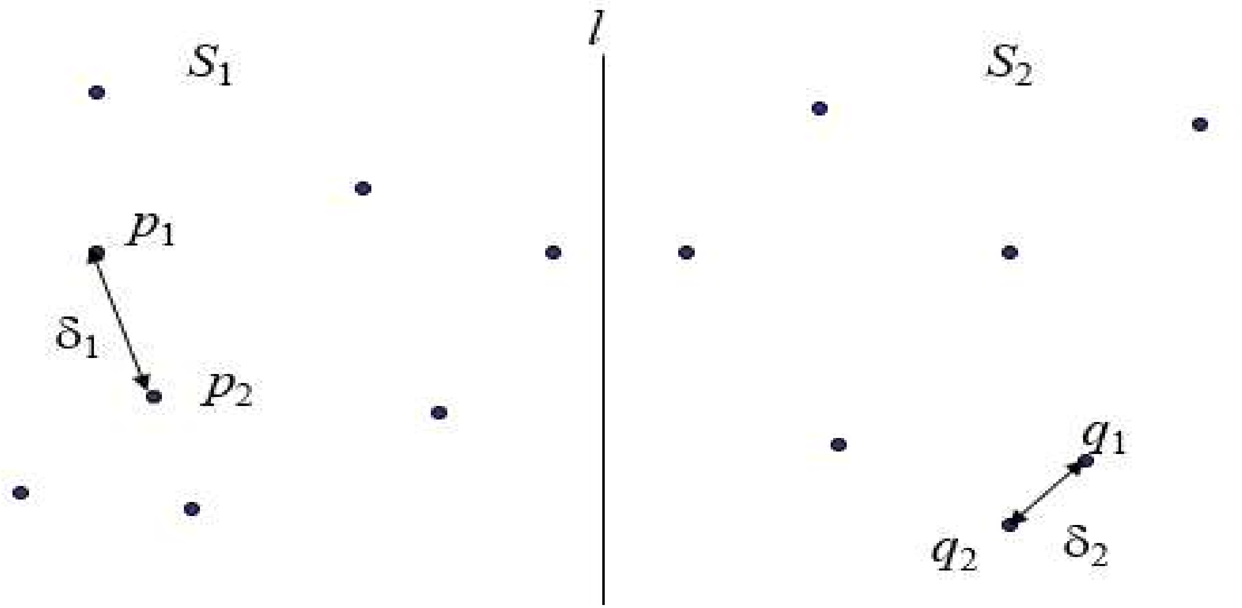
Closest Pair-Divide & Conquer dimension d=1

- To find a pair {$p3$, $q3$}, Is it necessary to check all possible pairs?
  No
- How many points of S1 or S2 can lie within δ of m?
  - Since δ is the distance between the closest pair in either S_1 or S_2, there can only be at most 1 point in each side.
  - Therefore, the number of distance computations required to check for a closest pair {$p3$, $q3$} with $p3 \in S_1$ and $q3 \in S_2$ is O(1).
- Thus, the time complexity is
  - O(n log n)



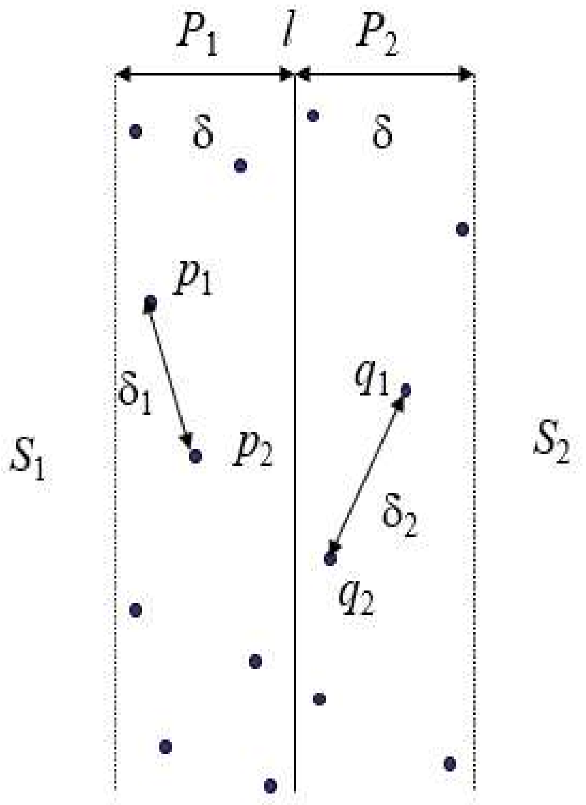
Closest Pair-Divide & Conquer dimension d=2

- Partition two dimensional set S into subsets S_1 and S_2 by a vertical line l at the median x coordinate of S.
- Solve the problem recursively on S_1 and S_2.
- Let $\{p1, p2\}$ be the closest pair in S_1 and $\{q1, q2\}$ in S_2.
- Let δ_1 = distance($p1$,$p2$) and δ_2 = distance($q1$,$q2$)
- Let $\delta = \min(\delta_1, \delta_2)$



Closest Pair-Divide & Conquer dimension d=2

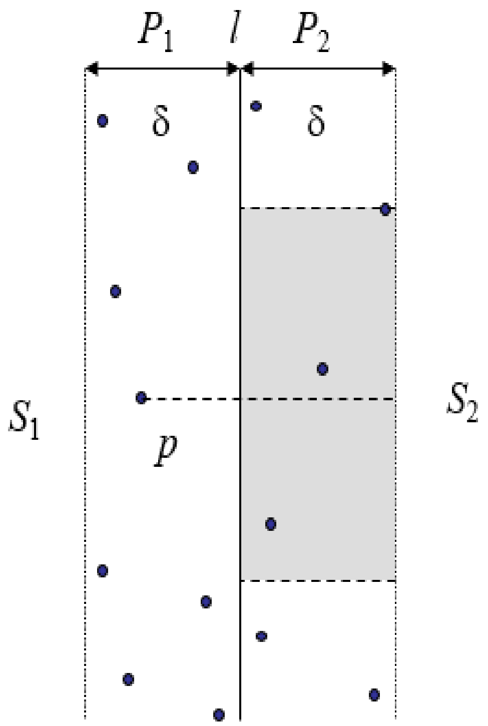
In order to merge we have to determine if exists a pair of points {*p*, *q*} where $p \in S_1$, $q \in S_2$ and distance(*p*, *q*) < δ.
If so, *p* and *q* must both be within δ of *l*.
Let P_1 and P_2 be vertical regions of the plane of width δ on either side of *l*.
If {*p*, *q*} exists, *p* must be within P_1 and *q* within P_2.
For $d = 1$, there was at most one candidate point for *p* and one for *q*.
For $d = 2$, every point in S_1 and S_2 may be a candidate, as long as each is within δ of *l*, which implies: $O(n/2) \cdot O(n/2) = O(n^2)$



<u>Can we do better ?</u>

Closest Pair-Divide & Conquer dimension d=2

- For a point *p in* P_1, which portion of P_2 should be checked?
- We only need to check the points that are within δ of p.
- Thus we can limit the portion of P2.
- The points to consider for a point p must lie within $\delta \times 2\delta$ rectangle R.
- At most, how many points are there in rectangle R?



Closest Pair-Divide & Conquer dimension d=2

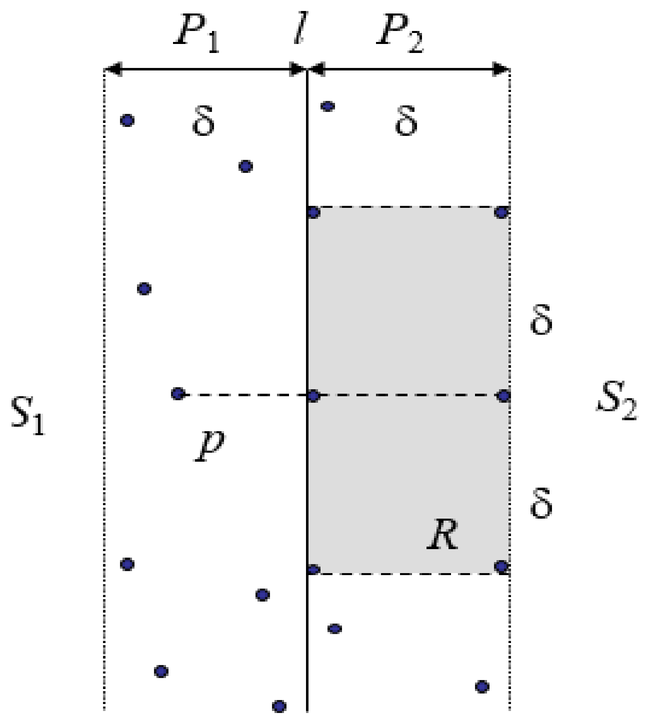
How many points are there in rectangle *R*?

Since no two points can be closer than δ, there can only be at most 6 points

Therefore, $6 \cdot O(n/2) \in O(n)$

Thus, the time complexity is

$O(n \log n)$



How do we know which 6 points to check?

Closest Pair-Divide & Conquer dimension d=2

How do we know which 6 points to check?

- Project *p* and all the points of S_2 within P_2 onto *l*.
- Only the points within δ of *p* in the *y* projection need to be considered (max of 6 points).
- After sorting the points on y coordinate we can find the points by scanning the sorted lists. Points are sorted by y coordinates only once.
- To prevent resorting in O(n log n) in each merge, two previously sorted lists are merged in O(n).

Time Complexity: O(n log n)

SUMMARY

Sort Points:
Sort all points based on their x-coordinates.

- **Divide**:
  Split the points into two halves, P_L (left) and P_R (right), based on the midpoint of the x-coordinates.p
- **Conquer**:
Recursively find the closest pair in P_L and P_R. Compute the smallest distance (δ) from these two halves.
- **Combine**:
  - Identify points in a vertical strip centered at the midpoint within a distance δ from the dividing line.
  - Sort the points in the strip by y-coordinates and calculate distances between points within the strip.
  - Update δ if a closer pair is found in the strip.
- **Return the Closest Pair**:
The minimum of the closest distances in P_L, P_R, and the strip is the result.

SELF-ASSESSMENT QUESTIONS

In the Divide and Conquer approach, which of the following is the first step?

(a) Divide the points into two halves
(b) Sort the points based on their x-coordinates
(c) Recursively find the closest pair in both halves
(d) Sort the points based on their y-coordinates

In the Closest Pair Problem, the Euclidean distance between two points (x_1,y_1) and (x_2,y_2) is calculated as:

(a) $(x_1 - x_2)^2 + (y_1 - y_2)^2$
(b) $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
(a) $(x_1 - x_2) + (y_1 - y_2)$
(b) None

1. What is the Closest Pair Problem in computational geometry? Explain the goal and significance of solving this problem.
2. In the Divide and Conquer approach, how do you combine the solutions from the subproblems (i.e., after splitting the set of points into two halves)?

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books :

1. Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein., 3rd, 2009, The MIT Press.
2 Algorithm Design Manual, Steven S. Skiena., 2nd, 2008, Springer.
3 Data Structures and Algorithms in Python, Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser., 2nd, 2013, Wiley.
4 The Art of Computer Programming, Donald E. Knuth, 3rd, 1997, Addison-Wesley Professiona.

MOOCS :

1. https://www.coursera.org/specializations/algorithms?=
2.https://www.coursera.org/learn/dynamic-programming-greedy-algorithms#modules

THANK YOU