Divide and Conquer Algorithms

Introduction

- Solves computational problem by
  - Dividing it into subproblems of smaller size
    - Solve each problem recursively
  - Merging solutions to sub-problems to produce solution
- e.g.,
  - Merge Sort
  - Quick Sort
- Complexity analyzed using recurrence relations

Divide and Conquer

- Divide Step:
  - If input size smaller than certain threshold solve by straightforward method
  - Divide input data into two or more disjoint subsets
- Recur:
  - Recursively solve subproblems associated with the subsets
- Conquer
  - Take the solutions to the subproblems and merge them into a solution to the original problem

Sorting

- Merge Sort
    - Divide: Trivial
    - Conquer: Recursively Sort each sub-array
    - Combine: Linear-time merge
- Quick Sort
    - Divide: Split array based on pivot
    - Conquer: Recursively split
    - Combine: Trivial

Powering a Number

- Compute a^n
- Naive algorithm: $\Theta(n)$
- Divide and Conquer Algorithm
  - $$a^n = \begin{cases} a^{n/2} . a^{n/2} \; if\ n\ is\ even \\ a^{(n-1)/2} . a^{n/2} \; if\ n\ is\ odd \end{cases}$$
  - $T(n) = T(n/2)+\Theta(1) \Rightarrow T(n) = \Theta(\lg n)$

Matrix Multiplication

- Input: $X = [x_{ij}]$, $Y = [y_{ij}]$, $i,j = 1,2,3,\ldots,n$
- Output: $Z = [z_{ij}] = X \mathrm{x} Y$
  - $$z_{ij} = \sum_{k=1}^{n} x_{ik} \cdot y_{kj}$$
  - $O(n^3)$
- *Divide and conquer can reduce cost*

Divide and Conquer Strategy

- Idea
  - n x n matrix = 2x2 matrix of (n/2)x(n/2) submatrices
  - Rewrite Z = X x Y as
  - $$\begin{pmatrix} i & j \\ k & l \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} x \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$
  - i= ae+bg
  - j= af+bh
  - k= ce+dg
  - l= cf+dh

Divide and Conquer Strategy

- Analysis
  - 8 multiplications of $(n/2)\times(n/2)$ submatrices
  - 4 additions of $(n/2)\times(n/2)$ submatrices
- $T(n) = 8T(n/2)+\Theta(n^2)$
- This is $\Theta(n^3)$ – not better than brute force

Strassen's Method

- Multiply 2×2matrices with 7 recursive multiplications
- P1= a·(f–h)
- P2= (a+ b) ·h
- P3= (c+ d) ·e
- P4= d·(g–e)
- P5= (a+ d) ·(e+ h)
- P6= (b–d) ·(g+ h)
- P7= (a–c) ·(e+ f)
- r =P5+ P4–P2+ P6
- s=P1+ P2
- t=P3+ P4
- u=P5+ P1–P3–P7
- 7 multiplications, 18 adds/subs
- No reliance on commutativity of multiplication

Strassen's Algorithm

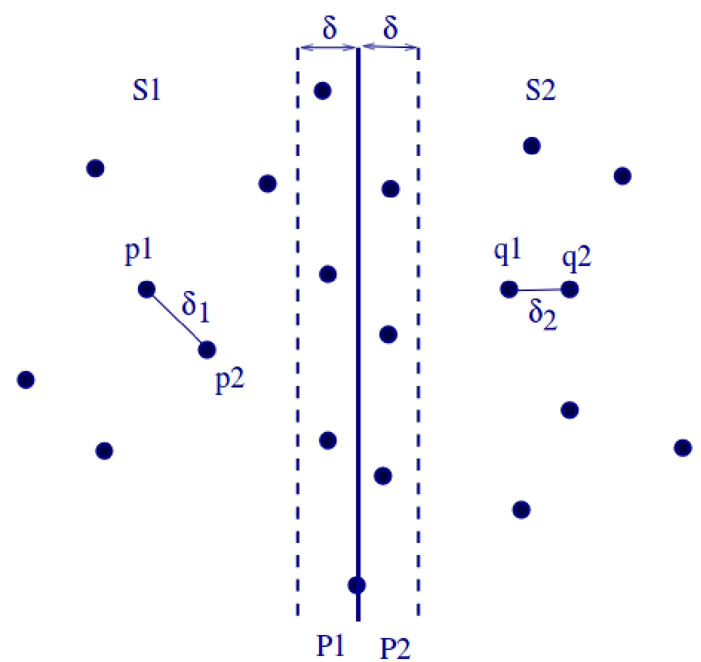
- Partition: A and B into $(n/2)\times(n/2)$ submatrices.
    - Form terms to be multiplied using + and –.
- Conquer: Perform 7 multiplications of $(n/2)\times(n/2)$ submatrices recursively
- Combine: Form C using +and –on $(n/2)\times(n/2)$ submatrices.
- $T(n) = 7T(n/2)+\Theta(n^2)$
- This is $\Theta(n^{2.81})$

Closest Pair Problem

- Input: n points in a plane, each given by a pair of real numbers
- Output: Pair of points with shortest distance between them
- Brute Force
  - Compute distance between every pair and find the minimum distance pair
  - □(n^2)
- Apply divide and conquer

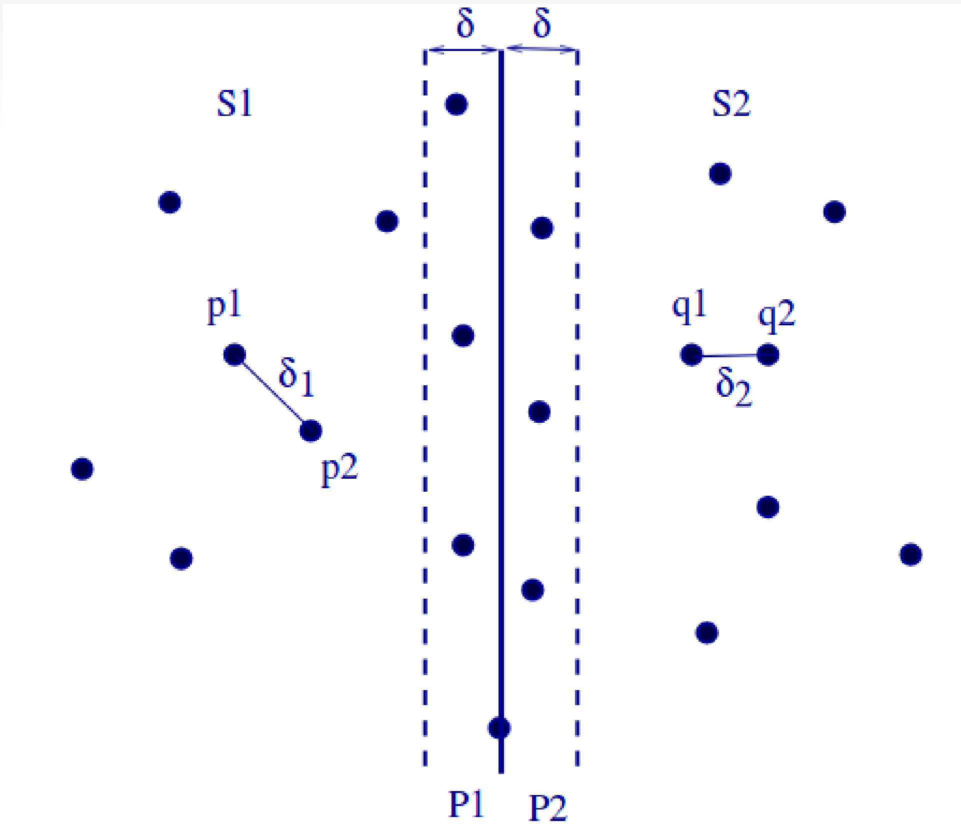
Strategy

- Partition S into S1, S2 by vertical line *l* defined by median x -coordinate in S
- Recursively compute closest pair distances δ1 and δ2. Set d = min(δ1, δ2).
- Now compute the closest pair with one point each in S1 and S2



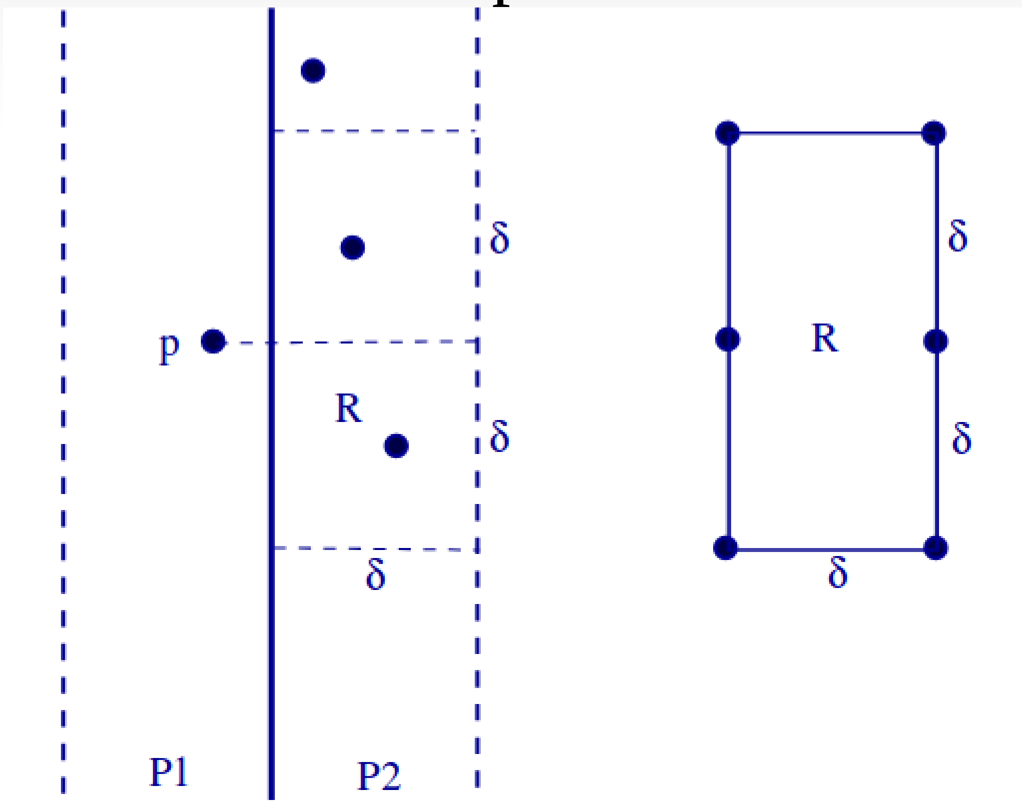
Divide and Conquer Strategy

- In each candidate pair (p, q), where p∈S1 and q∈ S2, the points p, q must both lie within δ of *l*
- It's possible that all n/2 points of S1 (and S2) lie within δ of *l*.



Conquer Step

- Points in P1, P2 (δ strip around *l*) have a special structure,that helps solve the conquer step faster
- Consider a point $p \in S1$.
  - All points of S2 within distance δ of p must lie in a $\delta \times 2\delta$ rectangle R .

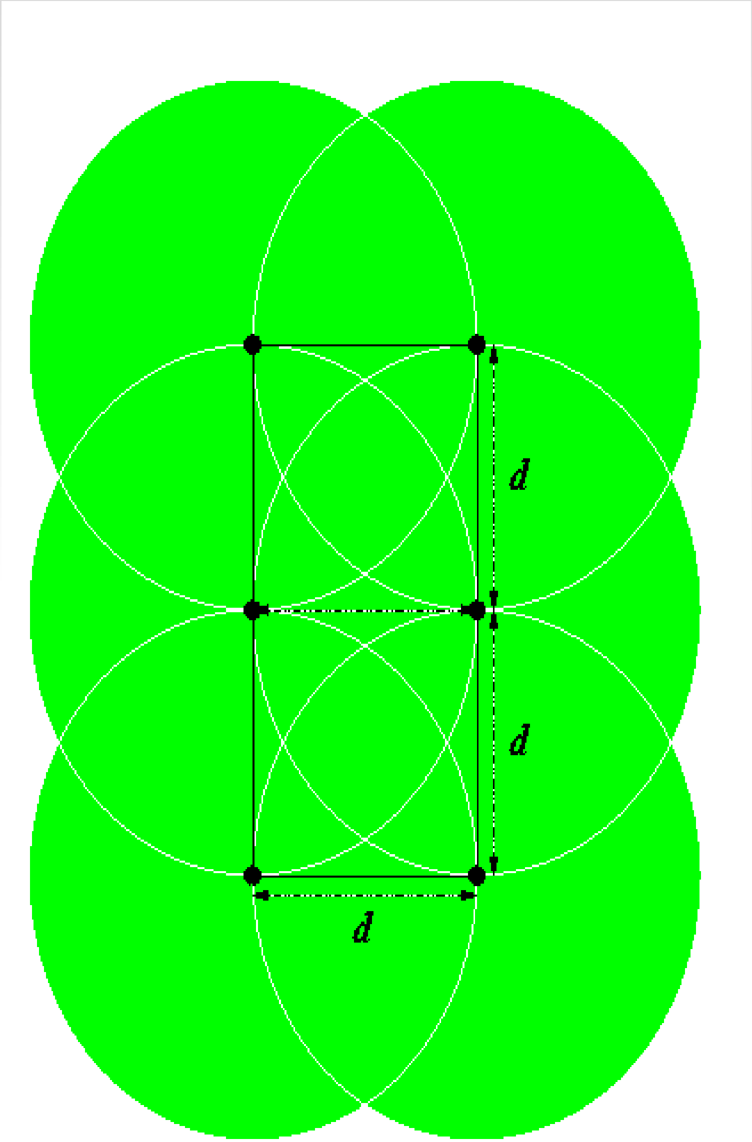


Conquer Step

- How many points can be inside R if each pair is at least δ apart?
  - In 2D, this number is at most 6! therefore 6x n/2 distance comparisons
  - This is still costly
  - In order to determine at most 6 potential counterpoints of p, project p and all points of P2 onto line l
  - Pick out points whose projection is within δ of p; at most six.
  - Presorting by 'y' axis can help make this faster
- Complexity: $T(n) = 2T(n/2) + \Theta(n)$

Why does this work

- Idea:
  - A rectangle of width d and height 2d can contain at most six points such that any two points are at distance at least d
- Proof
  - Place points into the box until it is impossible to add any more.
  - Imagine a circle around each point of radius d, which cannot contain any other point inside it
  - If you try to move any one of these points in any direction within the boundaries of the rectangle, then you would be moving two points too close together.



https://www.cs.mcgill.ca/~cs251/ClosestPair/proofbox.html

Problems

- Given an array having first n ints and next n chars

  A= *i*1 *i*2 *i*3 ... *i*n, *c*1 *c*2 *c*3 ... *c*n

  Write an in-place algorithm to rearrange the elements of the array as: A = *i*1 *c*1 *i*2 *c*2 ... *i*n *c*n
- Give an algorithm to divide an integer array into 2 sub-arrays s.t their averages are equal i.e average of values of left array must be same as average of right array