CPSC 490 – Problem Solving in Computer Science

Lecture 20: Divide and Conquer

Jason Chiu and Raunak Kumar

2017/03/13

University of British Columbia

Review: Merge Sort

```
MERGE_SORT(A):
  if |A| <= 1: return A
  return MERGE(MERGE_SORT(first half of A),
               MERGE_SORT(second half of A))
MERGE(A, B):
  if |A| == 0 or |B| == 0: return A ++ B
  if A[0] < B[0]:
    return A[0] ++ MERGE(A[1..], B)
  else:
    return B[0] ++ MERGE(A, B[1..])
```

Time complexity: $O(n \log n)$

Problem 1: Counting Inversions

How many pairs of indices $i < j$ satisfy $A_i > A_j$?

$$[1, 5, 3, 4, 2]$$

Problem 1: Solution

Merge sort with some extra accounting!

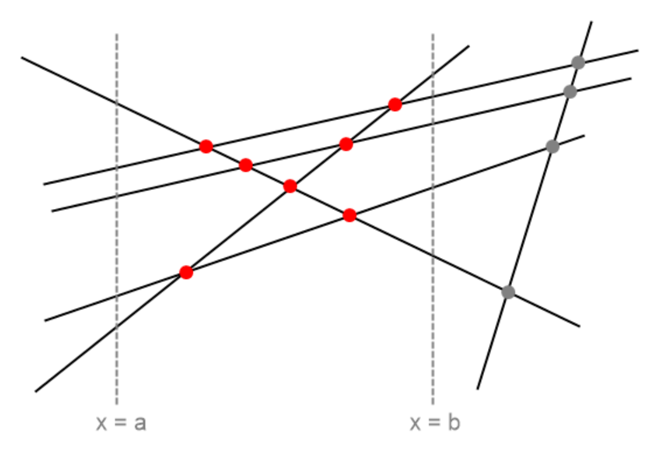
- Count inversions and merge sort first half
- Count inversions and merge sort second half
- Merge both halves – whenever element of second half is popped, add the current size of the first half of the list to answer

Time complexity: $O(n \log n)$

How would you deal with duplicates?

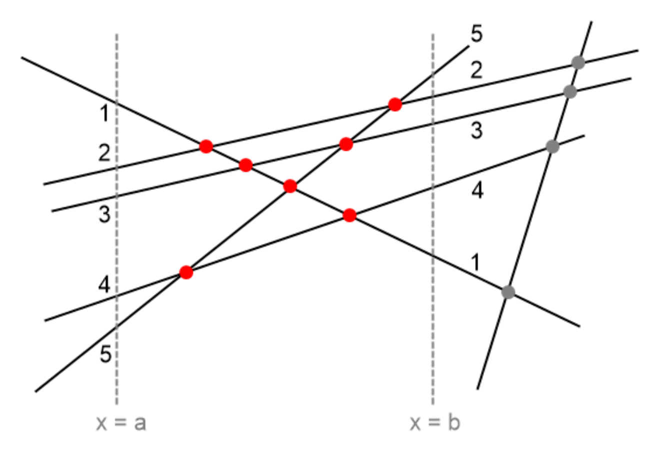
Problem 2: Counting Intersections

Given $n \leq 100,000$ lines how many intersections have x-coordinate satisfying $a \leq x \leq b$? No three lines intersect at the same point, and there are no vertical lines.



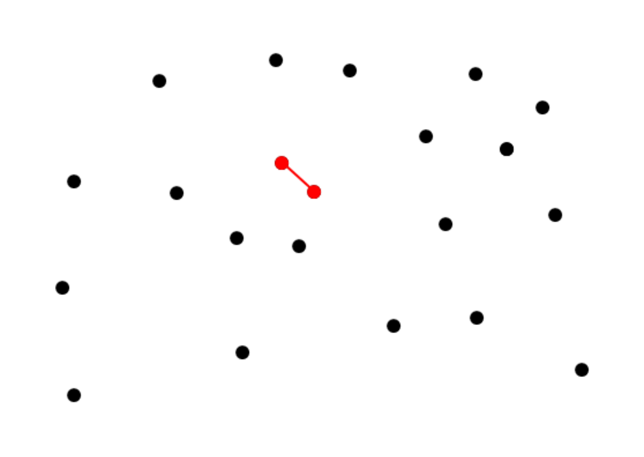
Problem 2: Solution

Recall that when two lines intersect, their order flip
$\Rightarrow$ label lines at $x = a$ by vertical order, count the number of inversions in the ordering at $x = b$, $O(n \log n)$



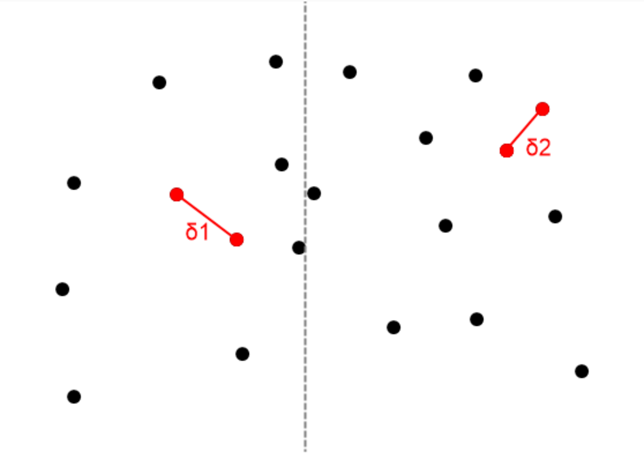
Problem 3: Closest Pair

Find the closest pair of points



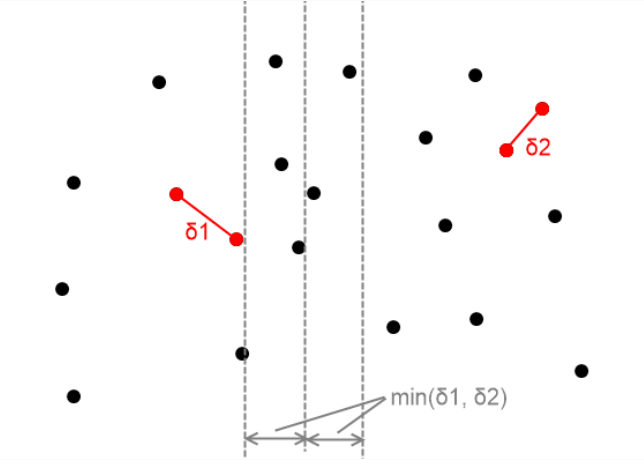
Problem 3: Solution

Strategy: like counting inversions – divide points in half, find closet pair of left side and right side, find closest pair going "across"



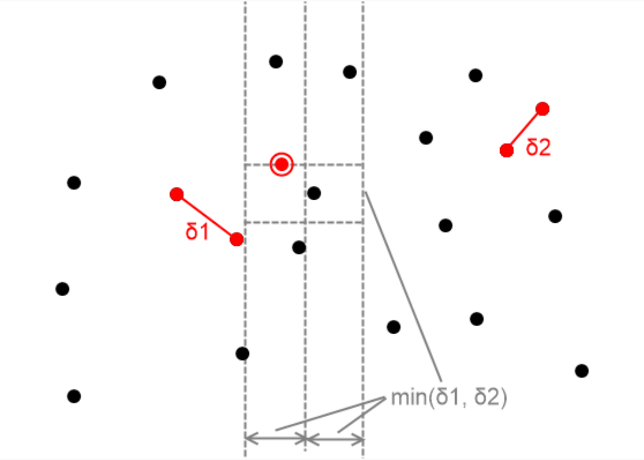
Problem 3: Solution

Observation 1: "closer" pair going "across" must be within $\delta =$ min distance so far from the split line $\Rightarrow$ take only points in $[x - \delta, x + \delta]$



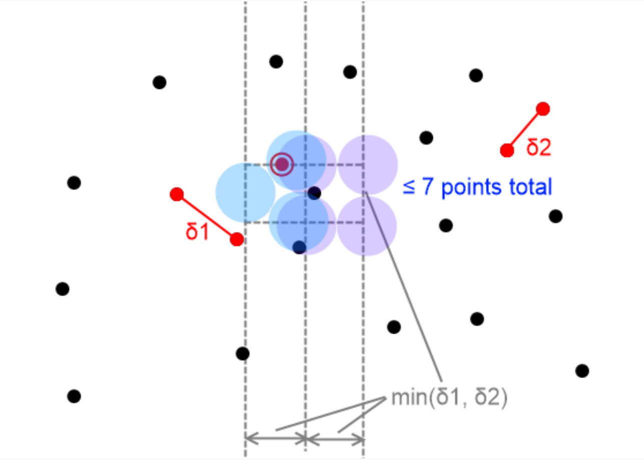
Problem 3: Solution

Observation 2: "closer" pair going "across" must have $\Delta y \leq \delta$
$\Rightarrow$ only compute distance from (x_i, y_i) to other points within $[y_i, y_i + \delta]$



Problem 3: Solution

Observation 3: at must 6 such other points, all adjacent when ordered by $y \Rightarrow$ try distance from point i to points $i+1, i+2, \ldots, i+6$



Problem 3: Solution

Summary

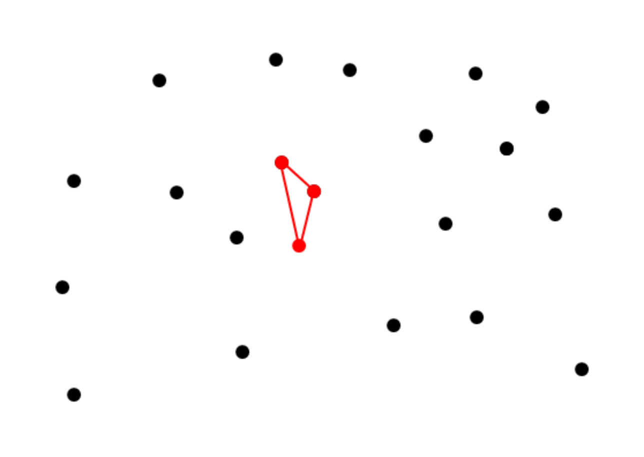
- Find x_0 that divides points evenly into left/right halves
- Solve each subproblem
- Let $\delta =$ smallest pairwise distance seen so far
- Get list of points having $x_0 - \delta \leq x \leq x_0 + \delta$ sorted by y
- For every point in the list, compute distance to subsequent points with $\Delta y \leq \delta$ (only have to look at the 6 points after)

Minor point: to avoid sorting by x, y in every recursive call, can sort once globally and then passed filtered lists down the recursion tree

Time complexity: $O(n \log n)$

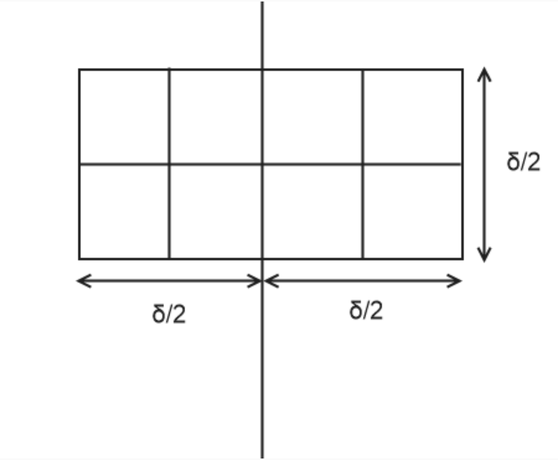
Problem 4: Smallest Perimeter Triangle

Find 3 points that make the smallest perimeter triangle



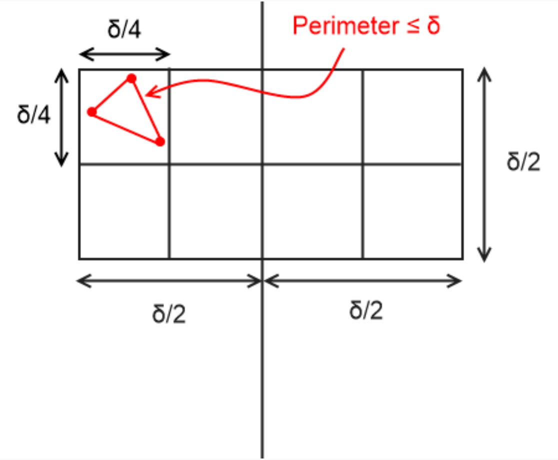
Problem 4: Solution

Apply same strategy! Triangle perimeter $\leq \delta \Rightarrow$ bounding box has width and height $\leq \delta/2$ so search in $[x_0 - \delta/2, x + \delta/2] \times [y, y + \delta/2]$



Problem 4: Solution

How many points? Each $\delta/4 \times \delta/4$ cell has ≤ 2 points, so ≤ 16 total



Problem 4: Solution

Apply same strategy!

- Find x_0 to divide points evenly into left/right halves
- Solve each subproblem
- Let $\delta =$ minimum perimeter so far
- Get list of points satisfying $x_0 - \delta/2 \leq x \leq x_0 + \delta/2$, sorted by y
- Scan through sorted list, make triangle for all triplets of points satisfying pairwise $\Delta y \leq \delta/2$ (i.e. indices within 16).

Time complexity: $O(n \log n)$ with large-ish constant factor

Other Applications of Divide and Conquer

- Convex hull in 2D, 3D
- Multiplication of large numbers / matrices
- Fast Fourier Transform
- DP Optimization

Next Class

Binary Search, Ternary Search