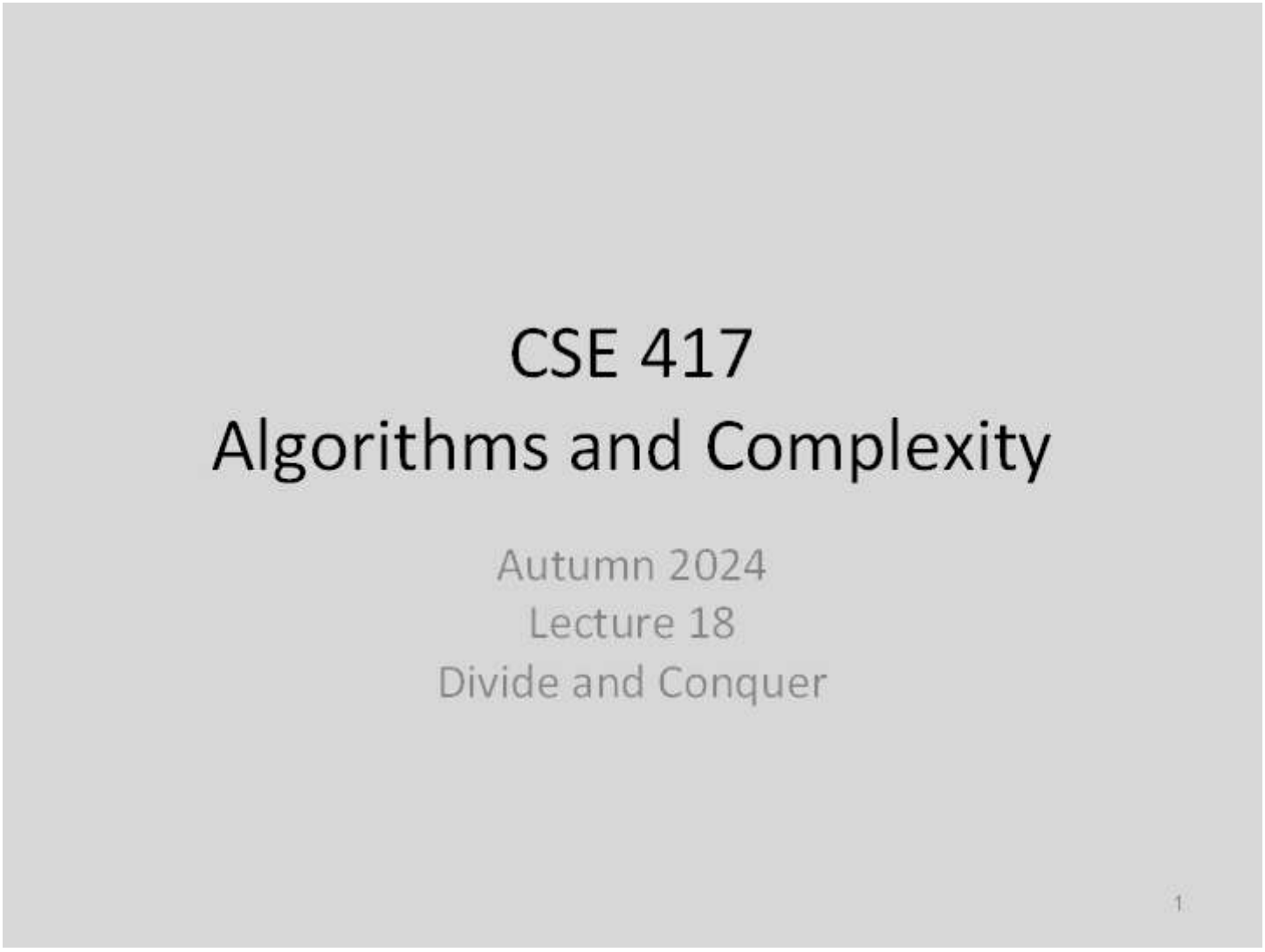
Lecture18


CSE 417
Algorithms and Complexity
Autumn 2024
Lecture 18
Divide and Conquer
1

Announcements

- No class on Monday
- Schedule
  - Lectures 19-23: Dynamic Programming
  - Lectures 24-26: Network Flow
  - Lectures 27-29: NP-Completeness

Thanksgiving

2

Divide and Conquer

- D&C Algorithms
  - MergeSort and QuickSort
  - $O(n^{2.80})$ Matrix Multiplication (Strassen)
  - Integer Multiplication
  - O(n) Median Algorithm
- Today's Algorithms – combining solutions
  - Counting Inversions
  - Closest Pair (in 2D)

3

Recurrences

- $T(n) = 2\,T(n/2) + n^2$

n^2

$\frac{n^2}{2} + \frac{n^2}{4}$

- $T(n) = 2\,T(n/2) + n$

4

n^2

$\frac{1}{2} n^2$

$\frac{1}{4} a^2$

Inversion Problem

- Let $a_1, \ldots, a_n$ be a permutation of 1 . . n
- (a_i, a_j) is an inversion if $i < j$ and $a_i > a_j$

4, 6, 1, 7, 3, 2, 5

3 + 4 + 3 + 1

17 Inversions

- Problem: given a permutation, count the number of inversions
- This can be done easily in $O(n^2)$ time
  - Can we do better?

5

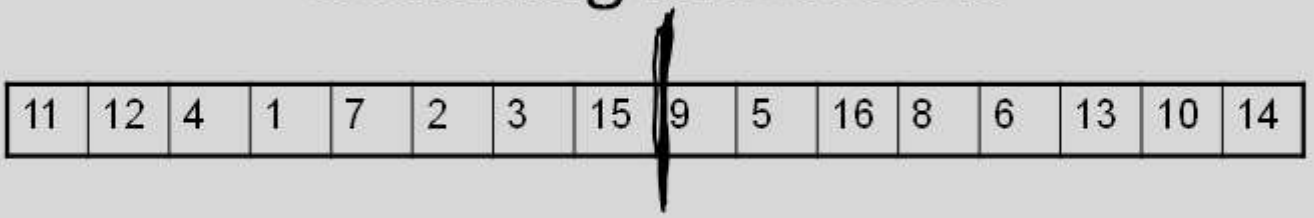
Application

- Counting inversions can be use to measure how close ranked preferences are
  - People rank 20 movies, based on their rankings you cluster people who like that same type of movie

6

$T(n) = 2T(\frac{n}{2}) + n^2$

Counting Inversions

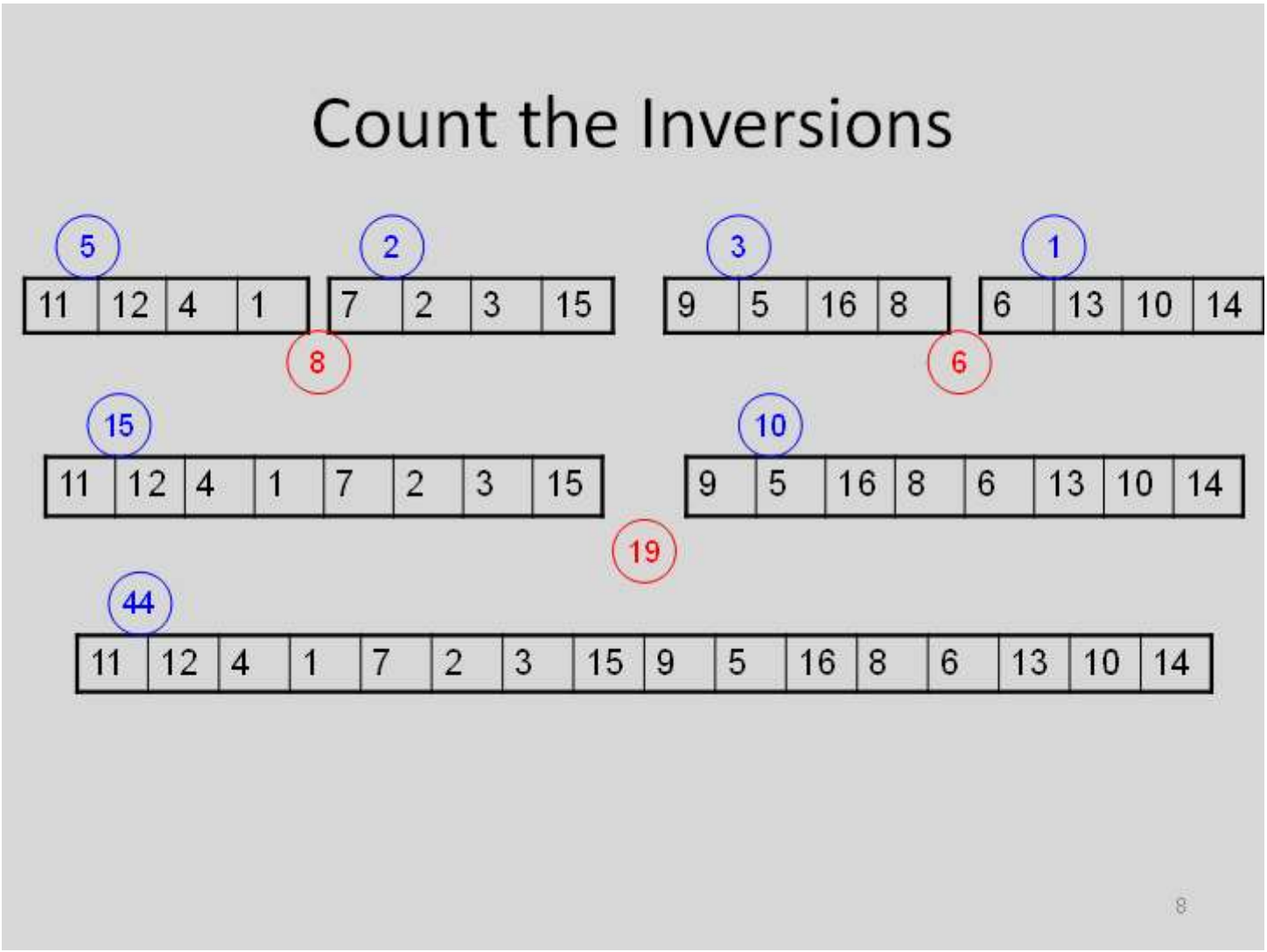


Count inversions on lower half

Count inversions on upper half

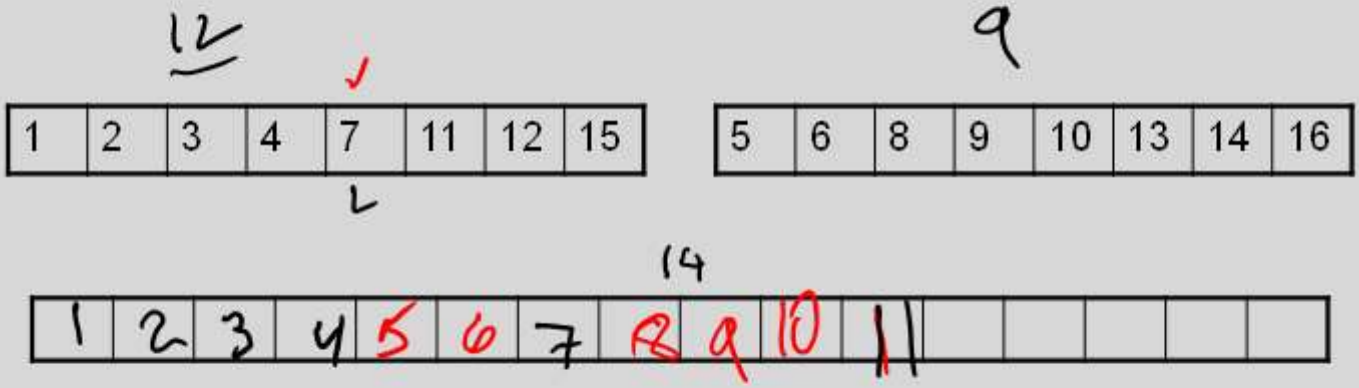
Count the inversions between the halves

7


Count the Inversions
5
11 12 4 1
2
7 2 3 15
8
3
9 5 16 8
1
6 13 10 14
6
15
11 12 4 1 7 2 3 15
10
9 5 16 8 6 13 10 14
19
44
11 12 4 1 7 2 3 15 9 5 16 8 6 13 10 14
8

Problem – how do we count inversions between sub problems in O(n) time?

- Solution – Count inversions while merging

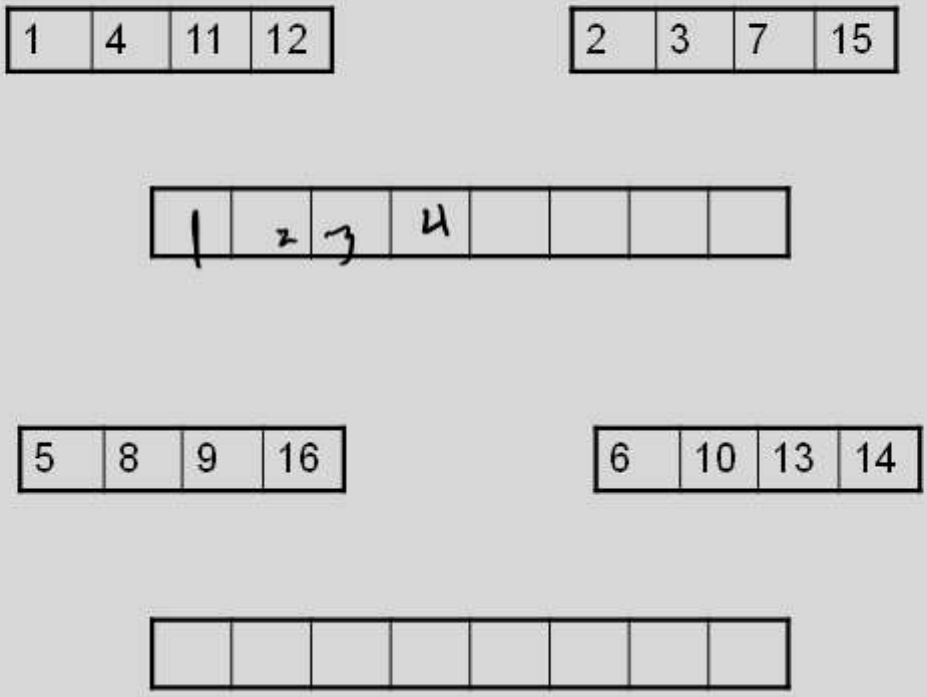


Standard merge algorithm – add to inversion count when an element is moved from the upper array to the solution

35

9

Use the merge algorithm to count inversions

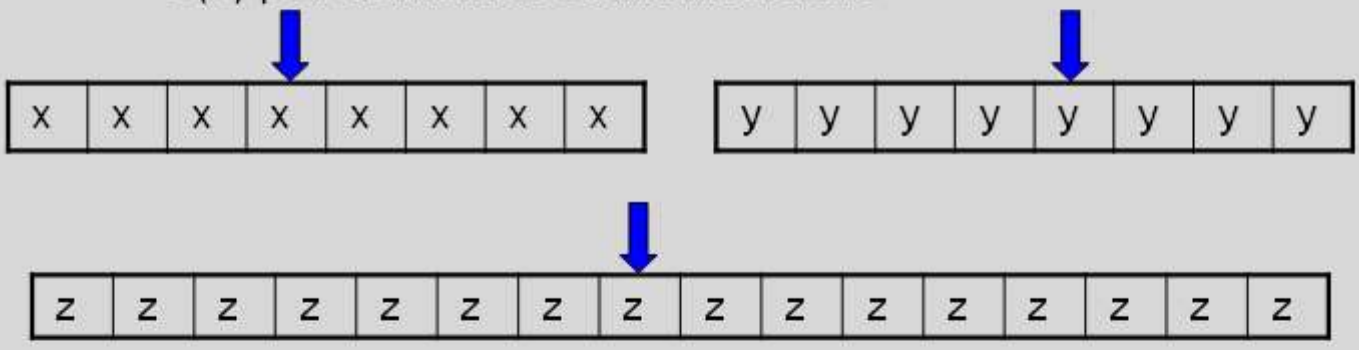


Indicate the number of inversions for each element detected when merging

10

Inversions

- Counting inversions between two sorted lists
  - O(1) per element to count inversions

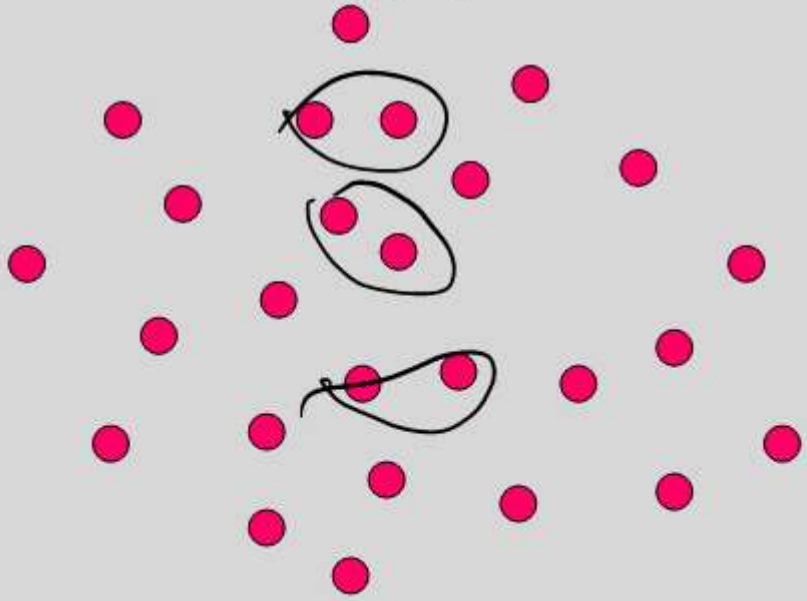


- Algorithm summary
  - Satisfies the "Standard recurrence"
  - $T(n) = 2\,T(n/2) + cn \leq O(n \log n)$

11

Closest Pair Problem (2D)

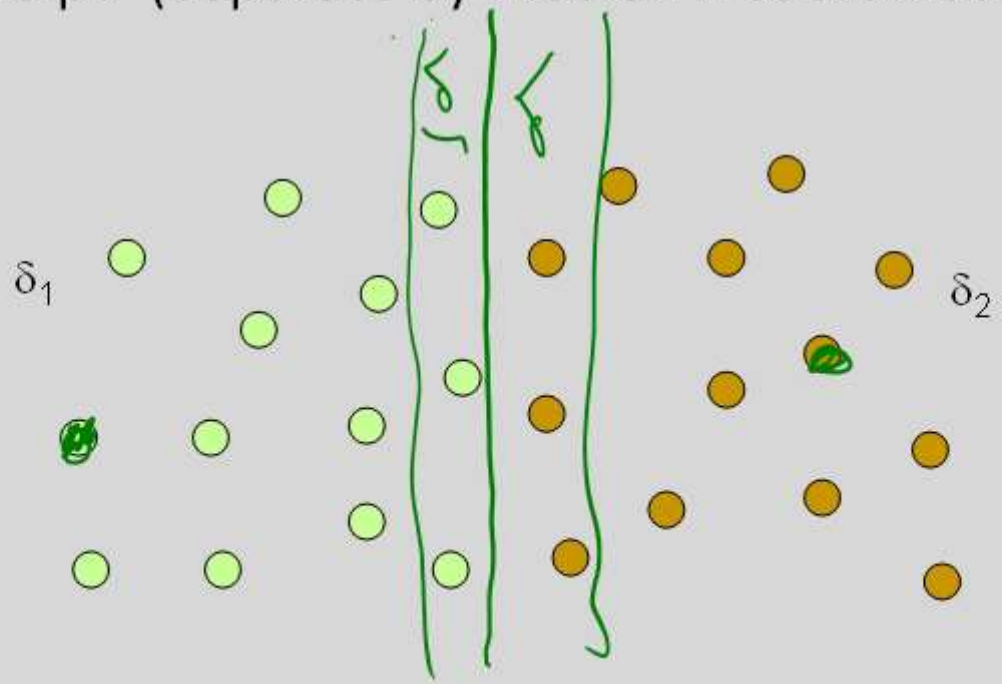
- Given a set of points find the pair of points p, q that minimizes dist(p, q)



12

Divide and conquer

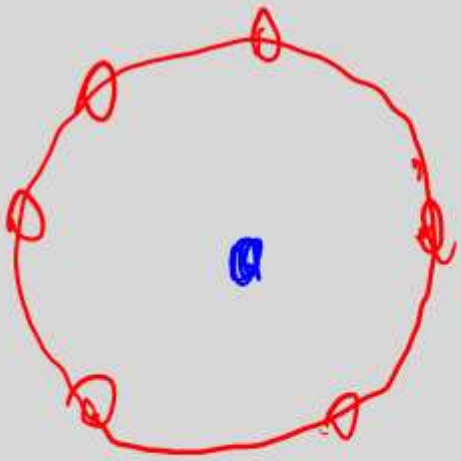
- If we solve the problem on two subsets, does it help? (Separate by median x coordinate)



13

Packing Lemma

Suppose that the minimum distance between points is at least δ, what is the maximum number of points that can be packed in a ball of radius δ?



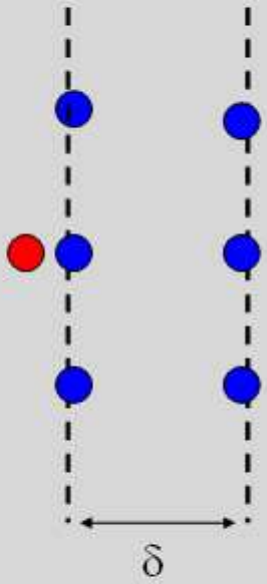
14

Combining Solutions

- Suppose the minimum separation from the sub problems is δ
- In looking for cross set closest pairs, we only need to consider points with δ of the boundary
- How many cross border interactions do we need to test?

15

A packing lemma bounds the number of distances to check



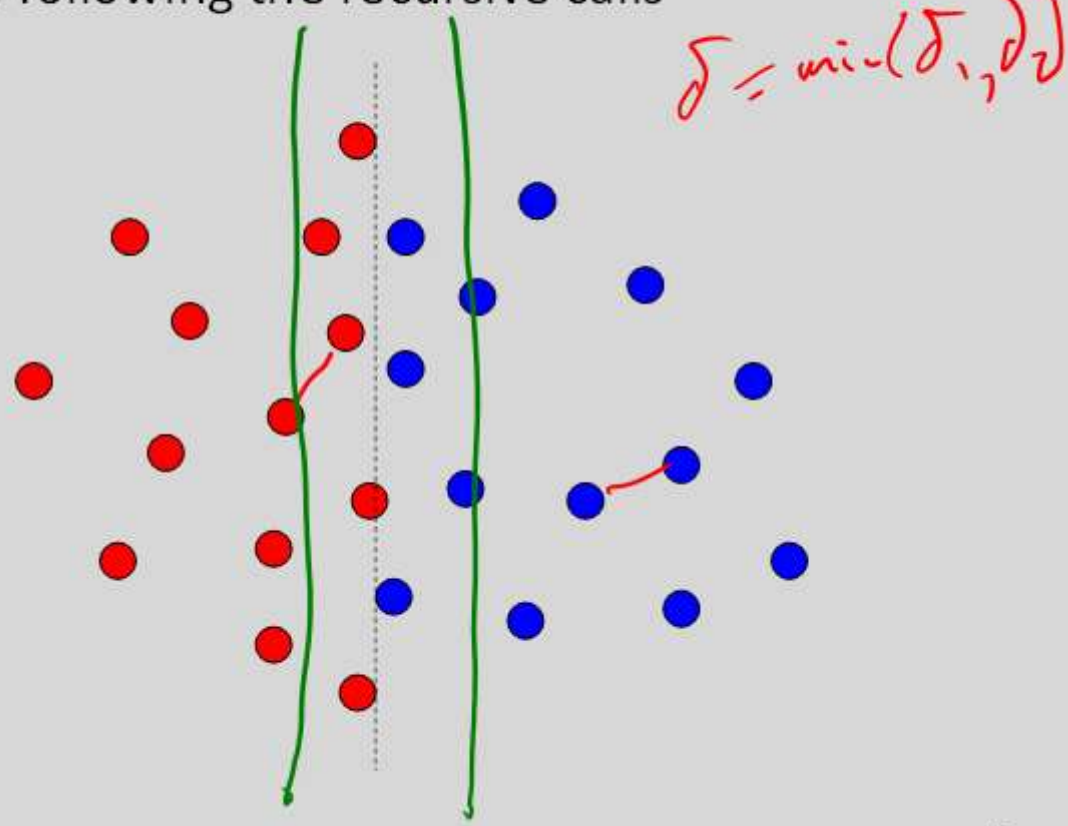
16

Details

- Preprocessing: sort points by y
- Merge step
  - Select points in boundary zone
  - For each point in the boundary
    - Find highest point on the other side that is at most δ above
    - Find lowest point on the other side that is at most δ below
    - Compare with the points in this interval (there are at most 6)

17

Identify the pairs of points that are compared in the merge step following the recursive calls



18

Algorithm run time

- After preprocessing:
  - $T(n) = cn + 2\,T(n/2)$

19

Divide and Conquer Summary

- Performance of Divide and Conquer
  - Reduce the number or size of subproblems
  - Reduce the amount of work in combining solutions

20