

# Lecture18

---

# CSE 417

## Algorithms and Complexity

Autumn 2023  
Lecture 18  
Divide and Conquer

11/8/2023

CSE 417

1

# Announcements

- No class on Friday

# Divide and Conquer

- Monday's Algorithms
  - $O(n^{2.80})$  Matrix Multiplication (Strassen)
  - $O(n)$  Median Algorithm
    - Quicksort style algorithm
    - Complicated mechanism to make it deterministic
- Today's Algorithms
  - Counting Inversions
  - Integer Multiplication
  - Closest Pair (in 2D)

# Inversion Problem

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

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

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

*- O(n log n)*

# 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

# Counting Inversions

$$T(n) = 2T\left(\frac{n}{2}\right) + \frac{n^2}{4}$$

11	12	4	1	7	2	3	15	9	5	16	8	6	13	10	14
----	----	---	---	---	---	---	----	---	---	----	---	---	----	----	----

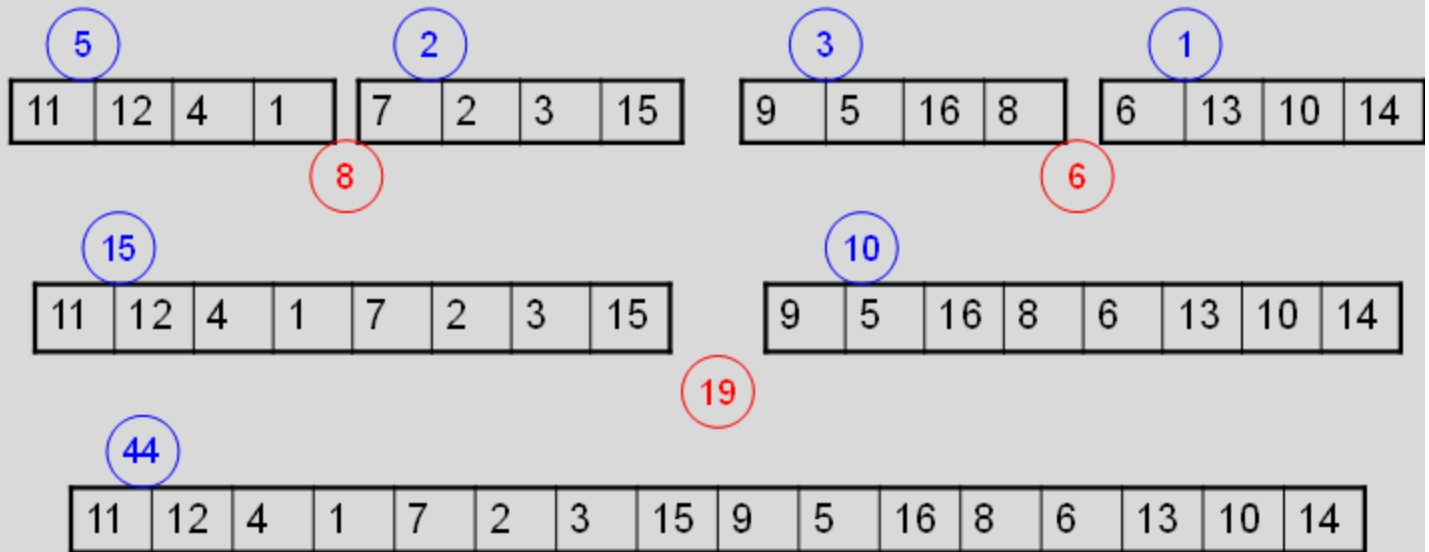
Count inversions on lower half

Count inversions on upper half

Count the inversions between the halves

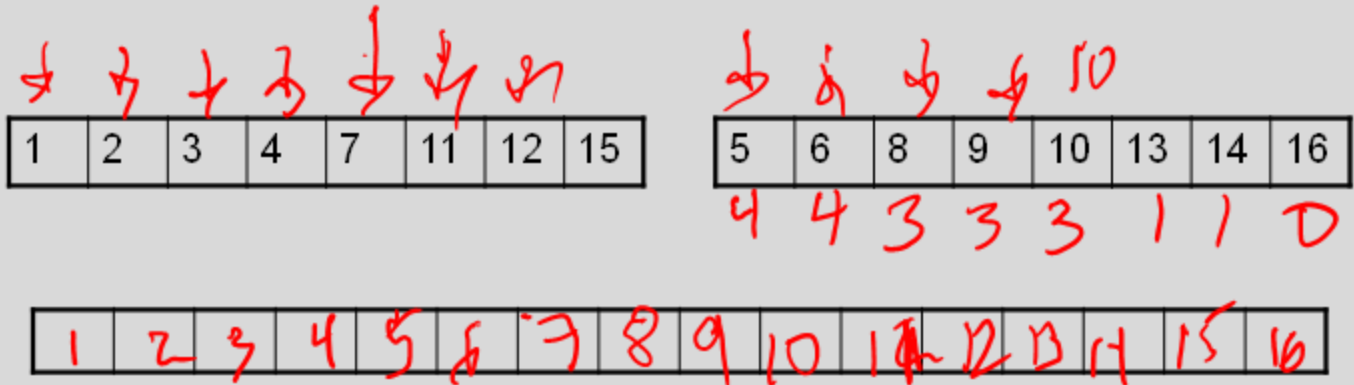
$O(n)$

# Count the Inversions



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



# Use the merge algorithm to count inversions

1	4	11	12
---	---	----	----

2	3	7	15
---	---	---	----

--	--	--	--	--	--	--	--

5	8	9	16
---	---	---	----

6	10	13	14
---	----	----	----

--	--	--	--	--	--	--	--

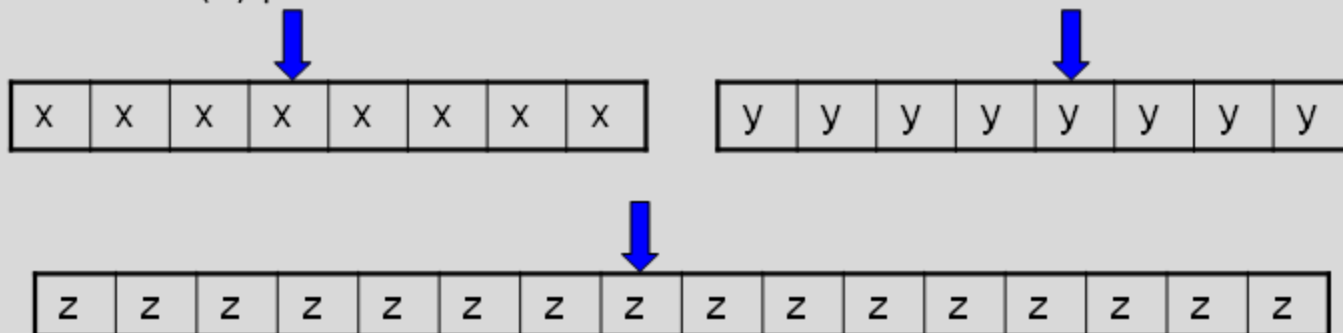
Indicate the number of inversions for each element detected when merging

CSE 417

9

# Inversions

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



- Algorithm summary
  - Satisfies the “Standard recurrence”
  - $T(n) = 2 T(n/2) + cn \approx O(n \log n)$

# Integer Arithmetic

```

9715480283945084383094856701043643845790217965702956767
+ 1242431098234099057329075097179898430928779579277597977

```

$O(n)$

↑

42378  
49921

Runtime for standard algorithm to add two  $n$  digit numbers:

```

2095067093034680994318596846868779409766717133476767930
X 5920175091777634709677679342929097012308956679993010921

```

$O(n^2)$

Runtime for standard algorithm to multiply two  $n$  digit numbers:

11

# Recursive Multiplication Algorithm (First attempt)

$$A = B 2^{n/2} + C$$

$$x = x_1 2^{n/2} + x_0$$

$$y = y_1 2^{n/2} + y_0$$

$$xy = (x_1 2^{n/2} + x_0) (y_1 2^{n/2} + y_0)$$

$$= x_1 y_1 2^n + (x_1 y_0 + x_0 y_1) 2^{n/2} + x_0 y_0$$

Recurrence:  $T(n) \leq n + 4T(\frac{n}{2})$

Run time:  $4^{\log n} = 2^{2 \log n} = n^2$

$$\boxed{B} \boxed{C} \quad \boxed{E} \boxed{F}$$

# Simple algebra

$$x = x_1 2^{n/2} + x_0$$

$$y = y_1 2^{n/2} + y_0$$

$$xy = x_1 y_1 2^n + (x_1 y_0 + x_0 y_1) 2^{n/2} + x_0 y_0$$

$$A = x_1 y_1 \quad B = x_0 y_0 \quad C = p - A - B$$

$$p = (x_1 + x_0)(y_1 + y_0) = x_1 y_1 + x_1 y_0 + x_0 y_1 + x_0 y_0$$

$$xy = A \cdot 2^n + C \cdot 2^{n/2} + B$$

Complete with 3 multipliers  $xy$

# Karatsuba's Algorithm

Multiply n-digit integers x and y

Let  $x = x_1 2^{n/2} + x_0$  and  $y = y_1 2^{n/2} + y_0$

Recursively compute

$$a = x_1 y_1$$

$$b = x_0 y_0$$

$$p = (x_1 + x_0)(y_1 + y_0)$$

Return  $a2^n + (p - a - b)2^{n/2} + b$

Recurrence:  $T(n) = 3T(n/2) + cn$

$$3 \log_2 a = n \log_2 3$$

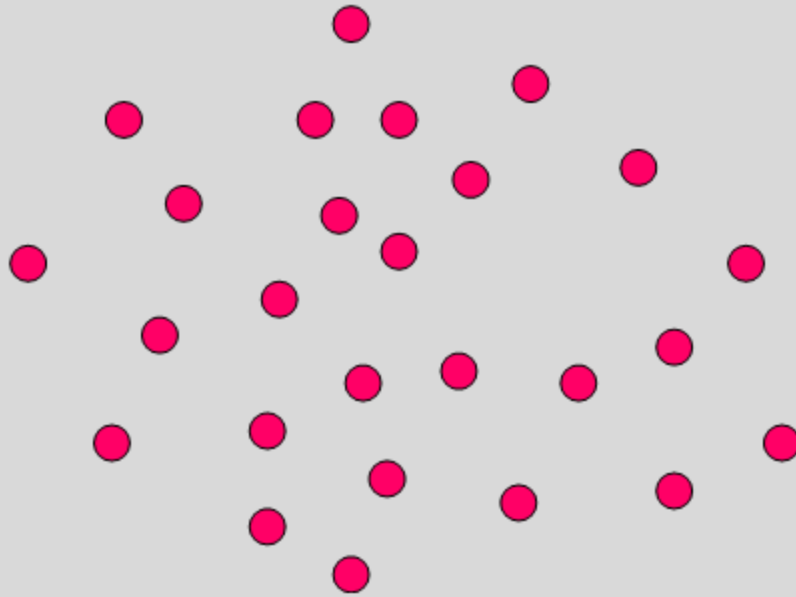
$\log_2 3 = 1.58496250073...$

CSE417

14

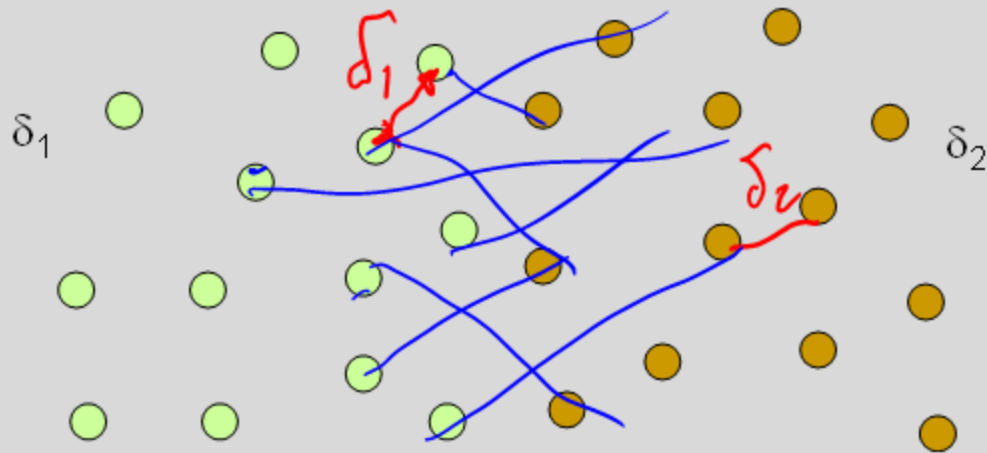
# Closest Pair Problem (2D)

- Given a set of points find the pair of points  $p, q$  that minimizes  $\text{dist}(p, q)$



# Divide and conquer

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





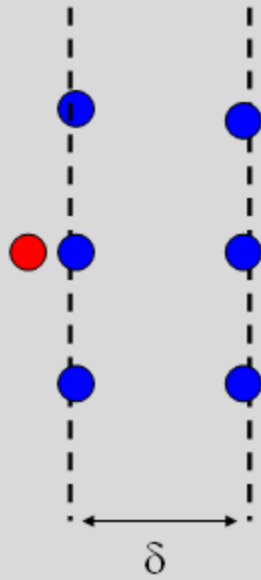
# Packing Lemma

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

# Combining Solutions

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

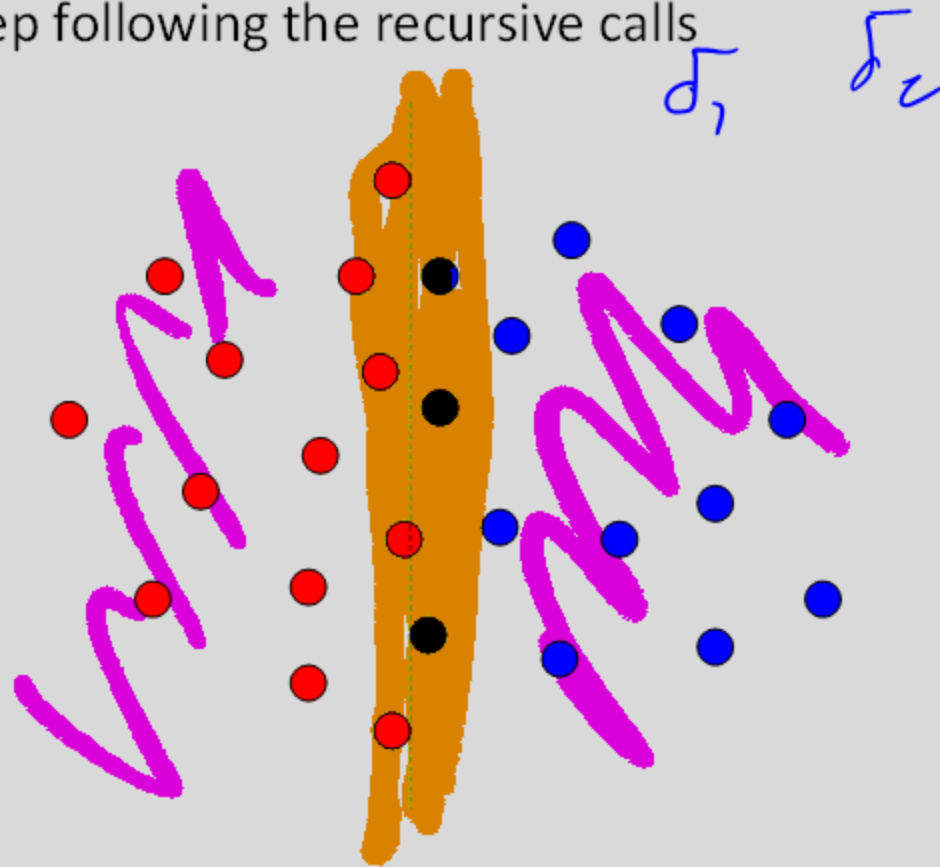
# A packing lemma bounds the number of distances to check



# 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  $\delta$  above
    - Find lowest point on the other side that is at most  $\delta$  below
    - Compare with the points in this interval (there are at most 6)

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



# Algorithm run time

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



Closest  
cross border  
point  $O(n)$

$$\tau(n) = 2\tau\left(\frac{n}{2}\right) + c$$

$$O(n \log n)$$