CSE
565

LECTURE 10

Divide and Conquer

- Closest Pair of Points
- Integer Multiplication
- Matrix Multiplication
- Median and Order Statistics

Sofya Raskhodnikova

S. Raskhodnikova; based on slides by E. Demaine, C. Leiserson, A. Smith, K. Wayne

Review questions

- Find the solution to the recurrence using MT: $T(n)=8T(n/2)+cn$.

- Draw the recursion tree for this recurrence.

  a. What is its height?

  b. What is the number of leaves in the tree?

Review questions

- Find the solution to the recurrence using MT: $T(n)=8T(n/2)+cn$.

(Answer: $\Theta(n^3)$.)

- Draw the recursion tree for this recurrence.

a. What is its height?

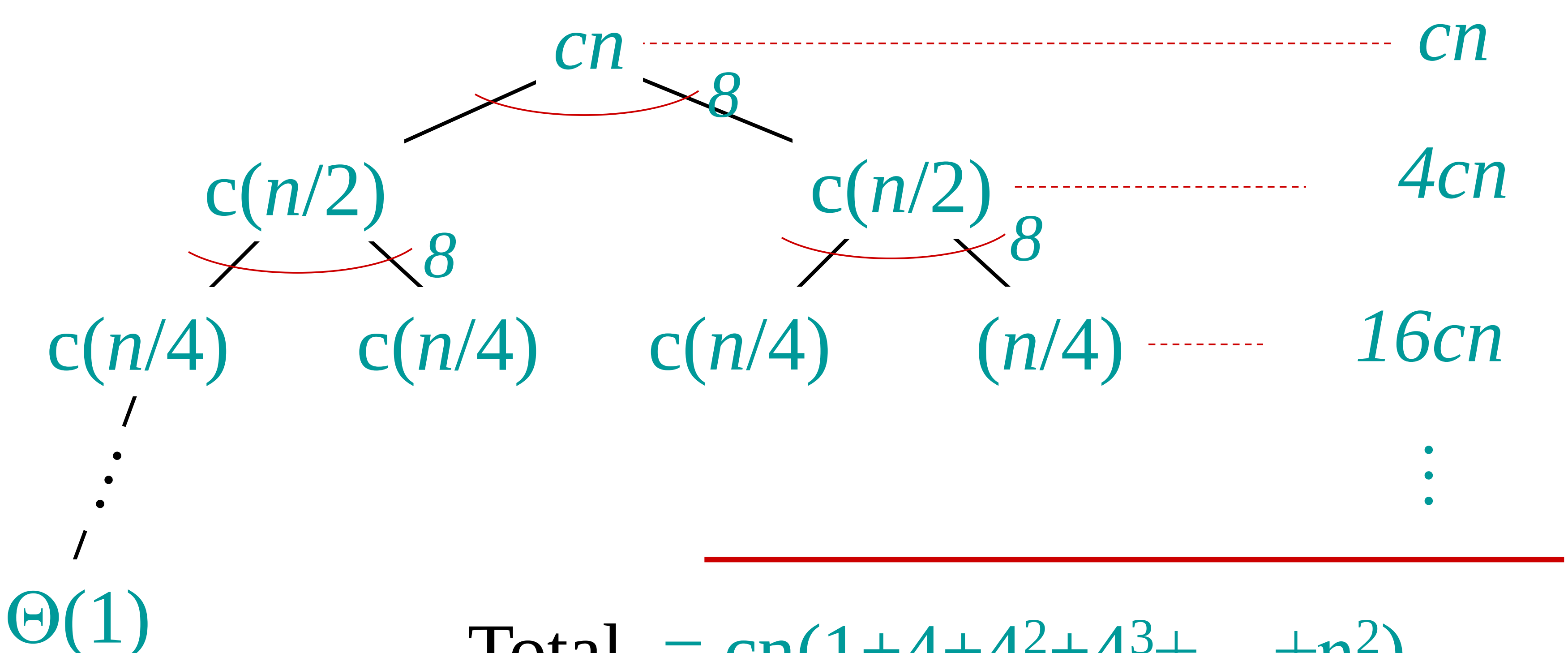
(Answer: $h=\log n$.)

b. What is the number of leaves in the tree?

(Answer: $8^h = 8^{\log n} = n^{\log 8} = n^3$.)

Review questions: recursion tree

Solve $T(n)=8T(n/2)+cn$:



$$\text{Total} = cn(1+4+4^2+4^3+\ldots+n^2)$$
$$= \Theta(n^3)$$

geometric series

Reminder: geometric series

$$1 + x + x^2 + \cdots + x^n = \frac{1 - x^{n+1}}{1 - x} \quad \text{for } x \neq 1$$

$$1 + x + x^2 + \cdots = \frac{1}{1 - x} \quad \text{for } |x| < 1$$

Divide and Conquer

– Break up problem into several parts.
– Solve each part recursively.
– Combine solutions to sub-problems into overall solution.

Divide et impera.
Veni, vidi, vici.
- Julius Caesar

Closest Pair of Points

Closest Pair of Points

Given n points in the plane, find a pair with smallest Euclidean distance between them.

- Fundamental geometric primitive.
  - Graphics, computer vision, geographic information systems, molecular modeling, air traffic control.
  - Special case of nearest neighbor, Euclidean MST, Voronoi.

fast closest pair inspired fast algorithms for these problems

- Brute force:

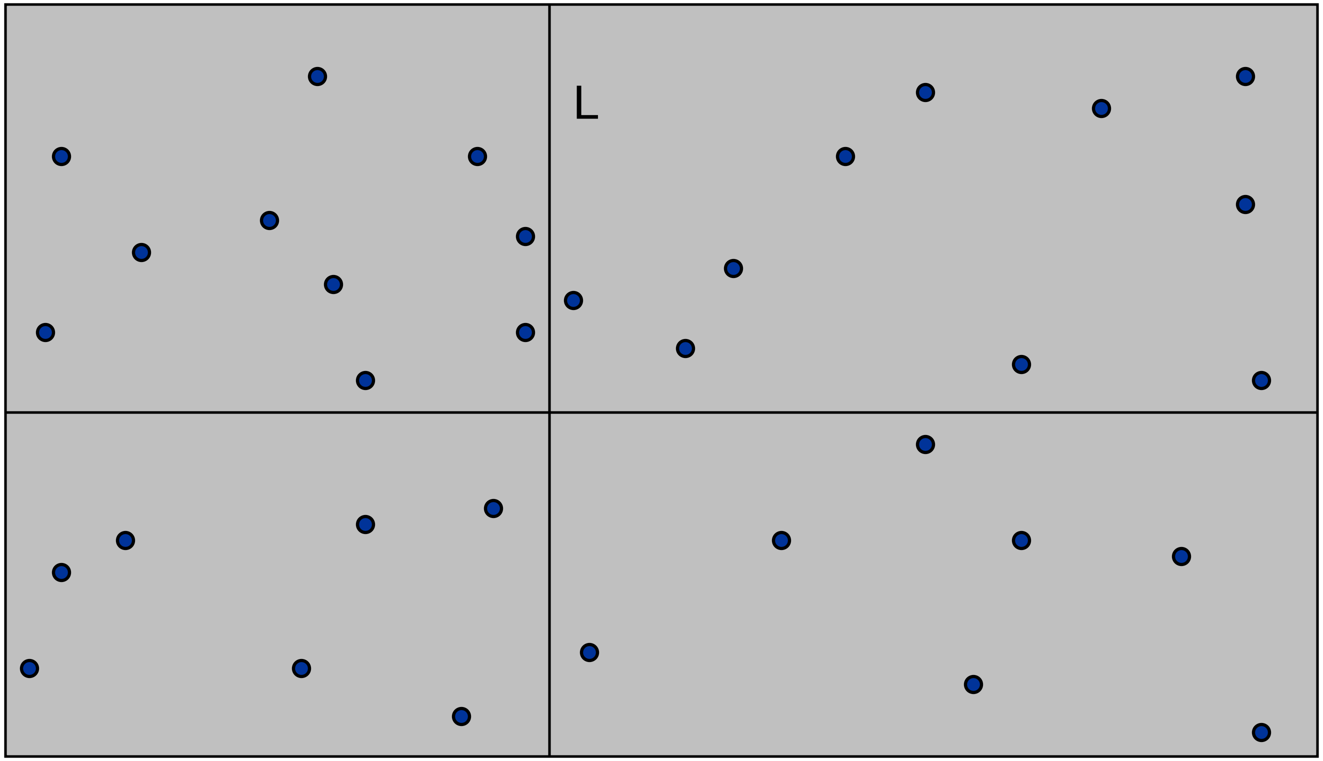
  Check all pairs of points p and q with $\Theta(n^2)$ comparisons.

- 1-D version: O(n log n) is easy if points are on a line.
- Assumption: No two points have same x coordinate.

to make presentation cleaner

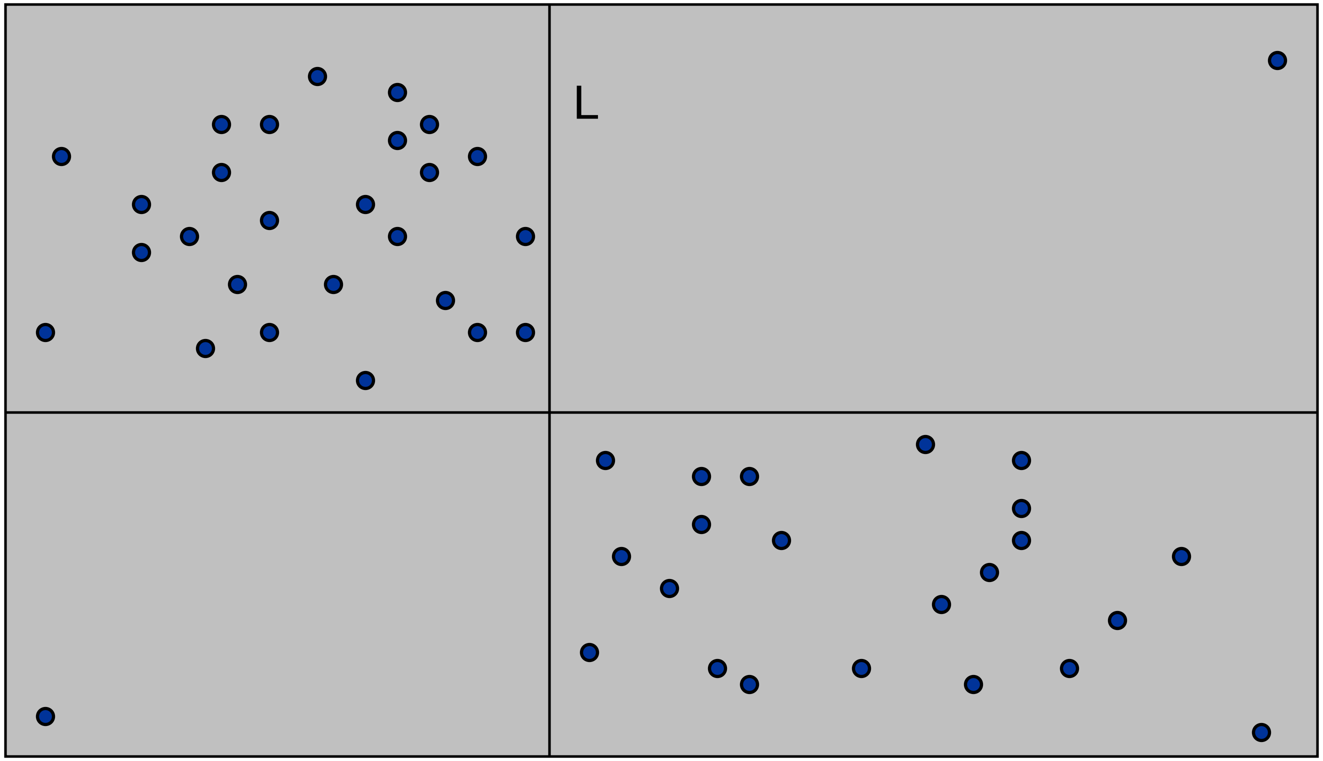
Closest Pair of Points: First Attempt

- Divide. Sub-divide region into 4 quadrants.



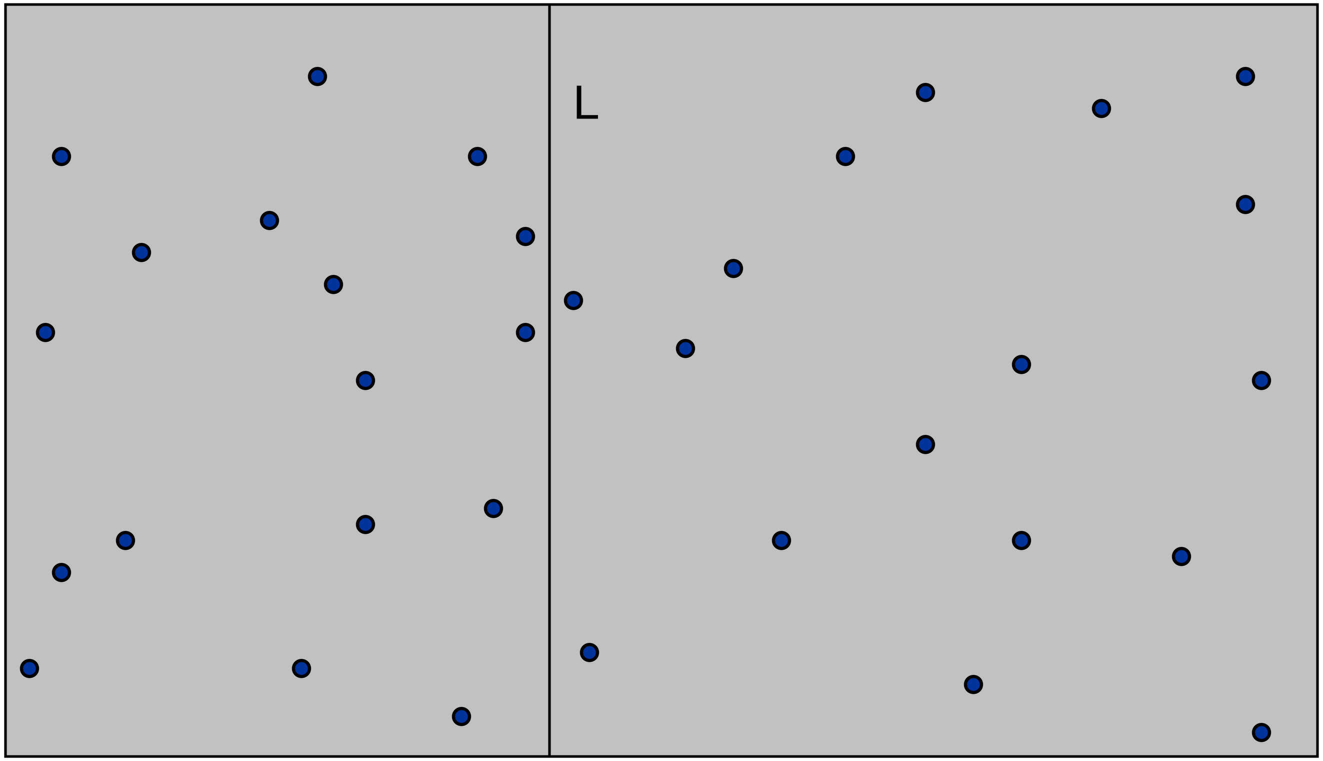
Closest Pair of Points: First Attempt

- Divide. Sub-divide region into 4 quadrants.
- Obstacle. Impossible to ensure $n/4$ points in each piece.



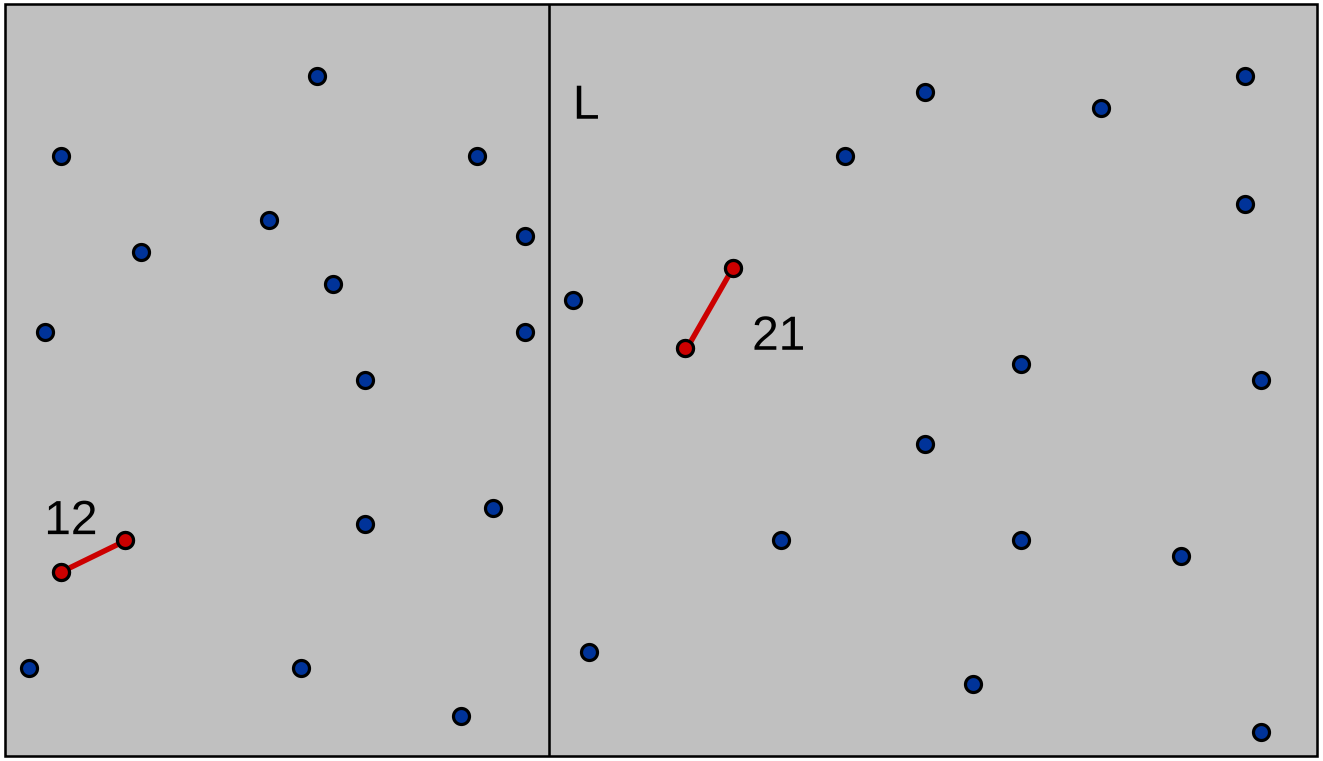
Closest Pair of Points

- Algorithm.
  - Divide: draw vertical line L, so that roughly $n/2$ points on each side.



Closest Pair of Points

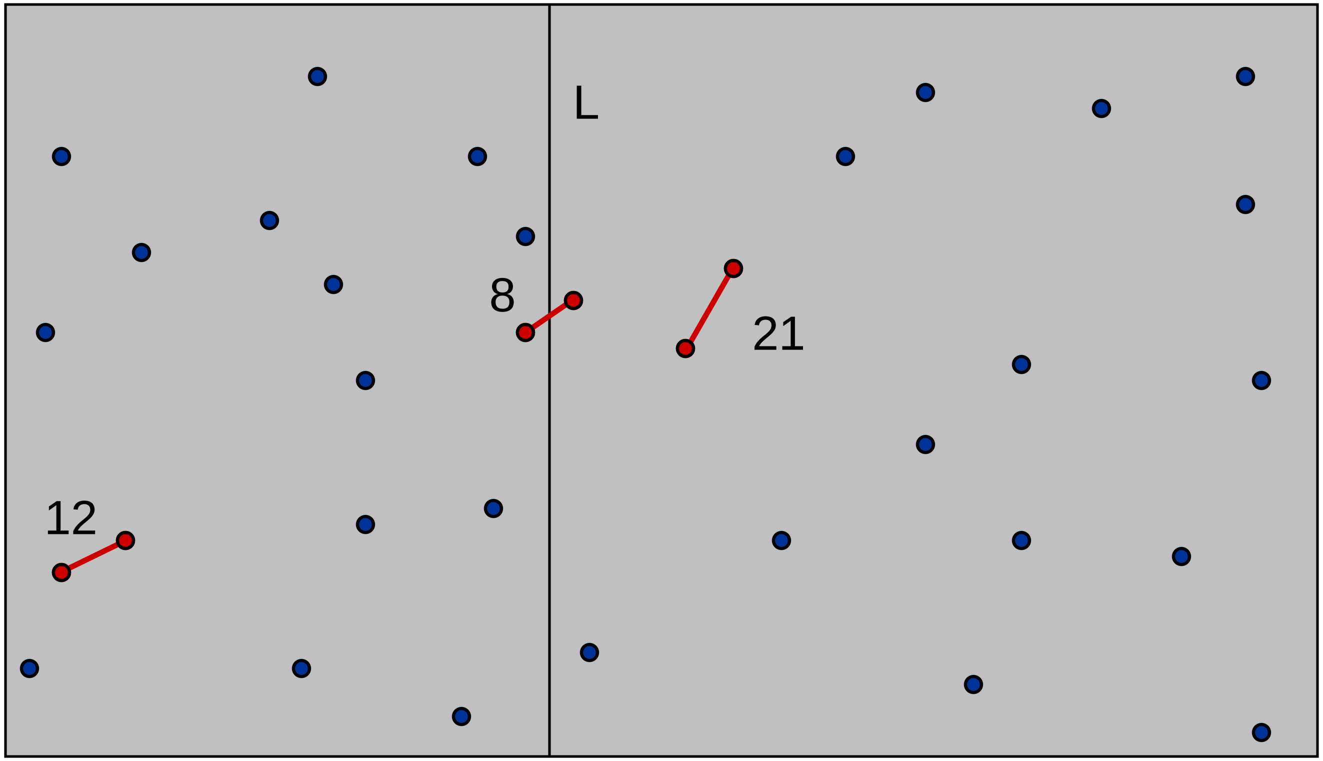
- Algorithm.
  - Divide: draw vertical line L, so that roughly $n/2$ points on each side.
  - Conquer: find closest pair in each side recursively.



Closest Pair of Points

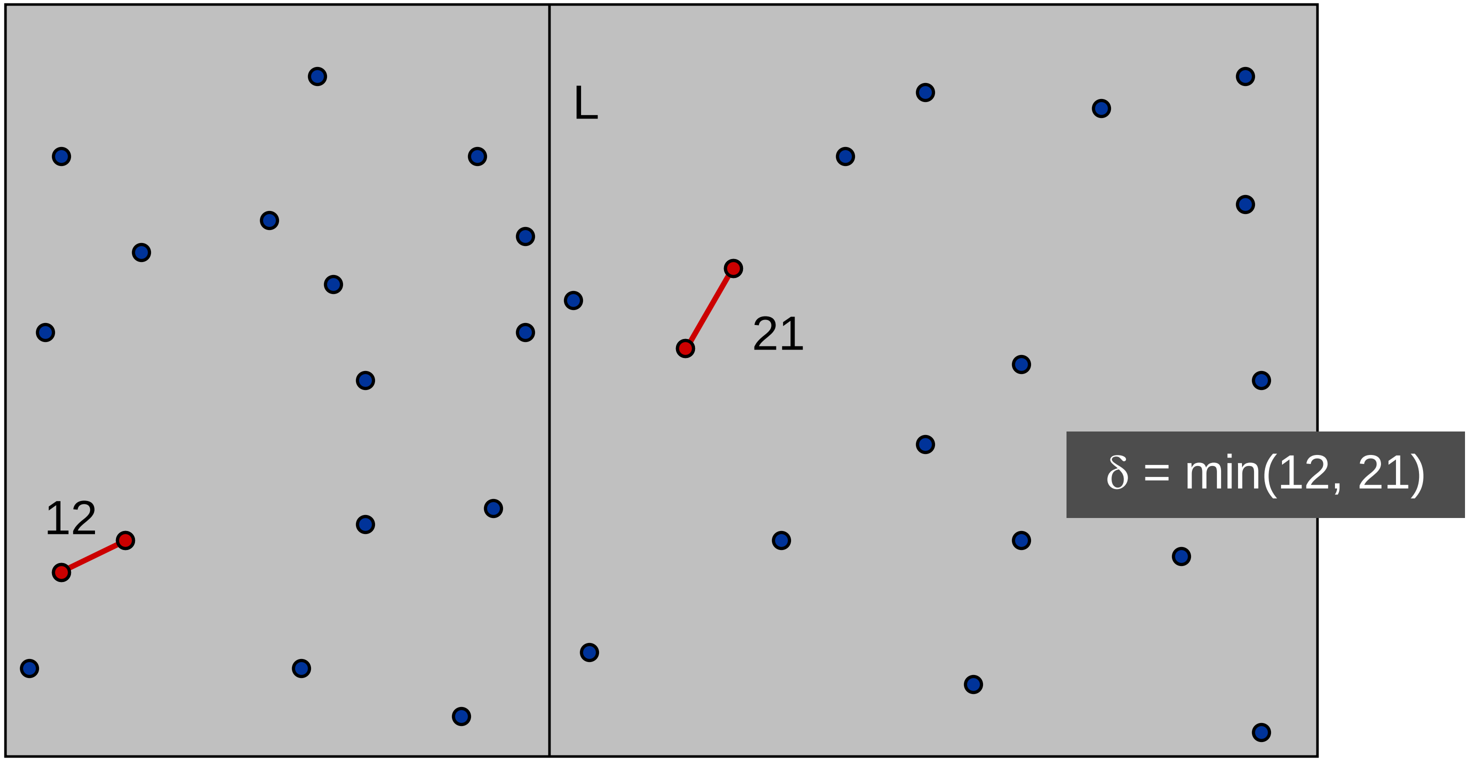
- Algorithm.
  - Divide: draw vertical line L, so that roughly $n/2$ points on each side.
  - Conquer: find closest pair in each side recursively.
  - Combine: find closest pair with one point in each side; return best of 3 solutions.

seems like $\Theta(n^2)$



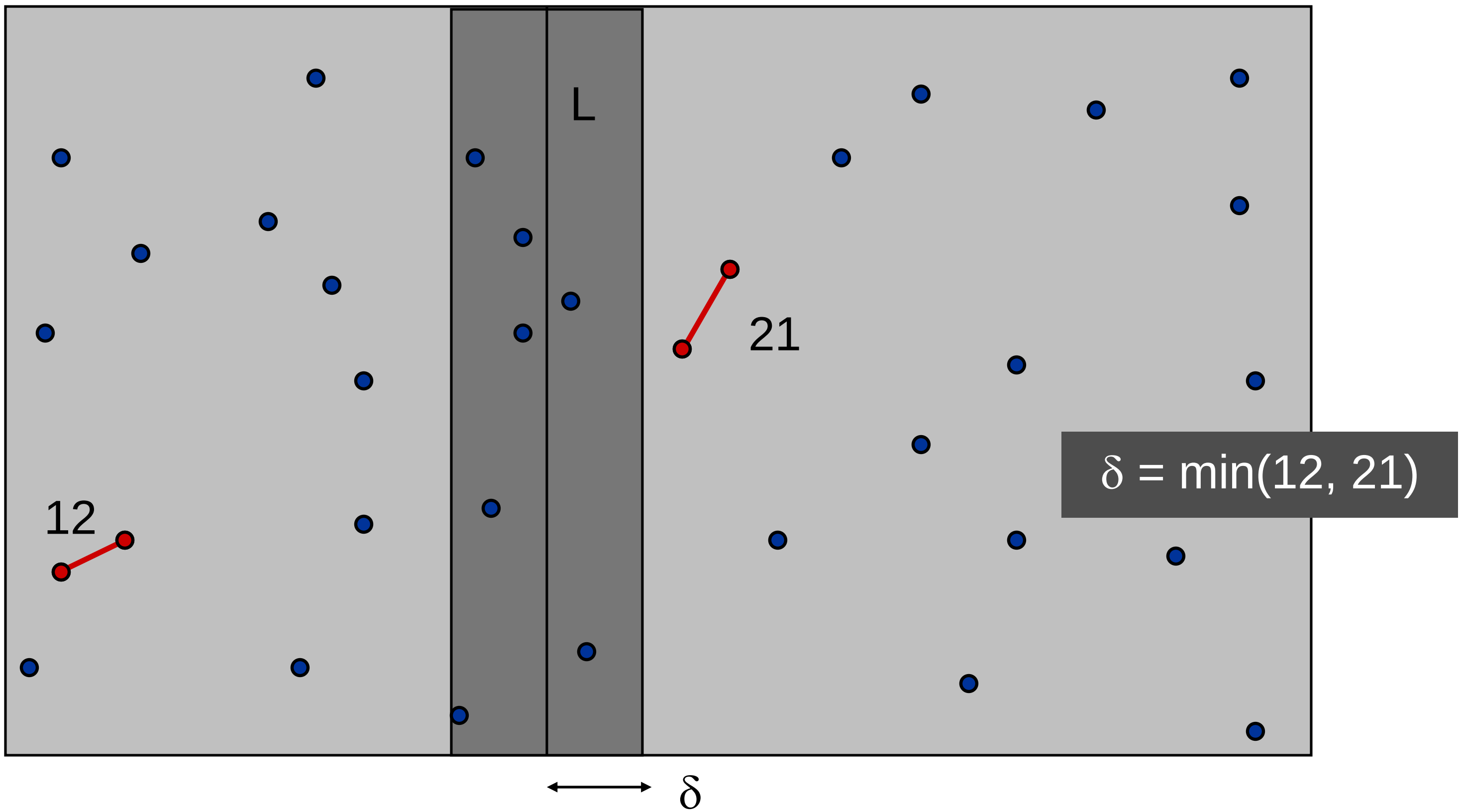
Closest Pair of Points

- Find closest pair with one point in each side, assuming that distance < δ.



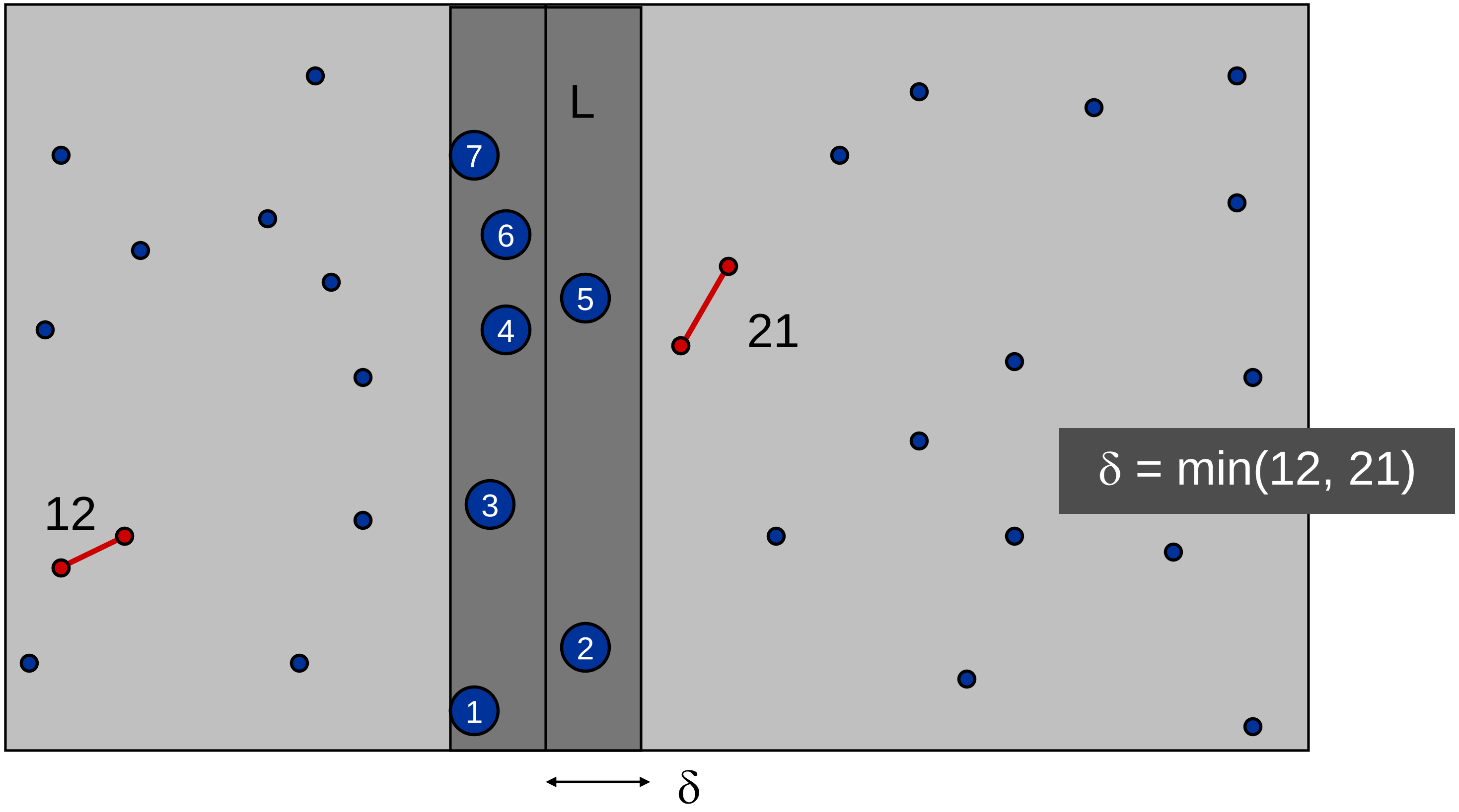
Closest Pair of Points

- Find closest pair with one point in each side, assuming that distance < δ.
  - Observation: only need to consider points within δ of line L.



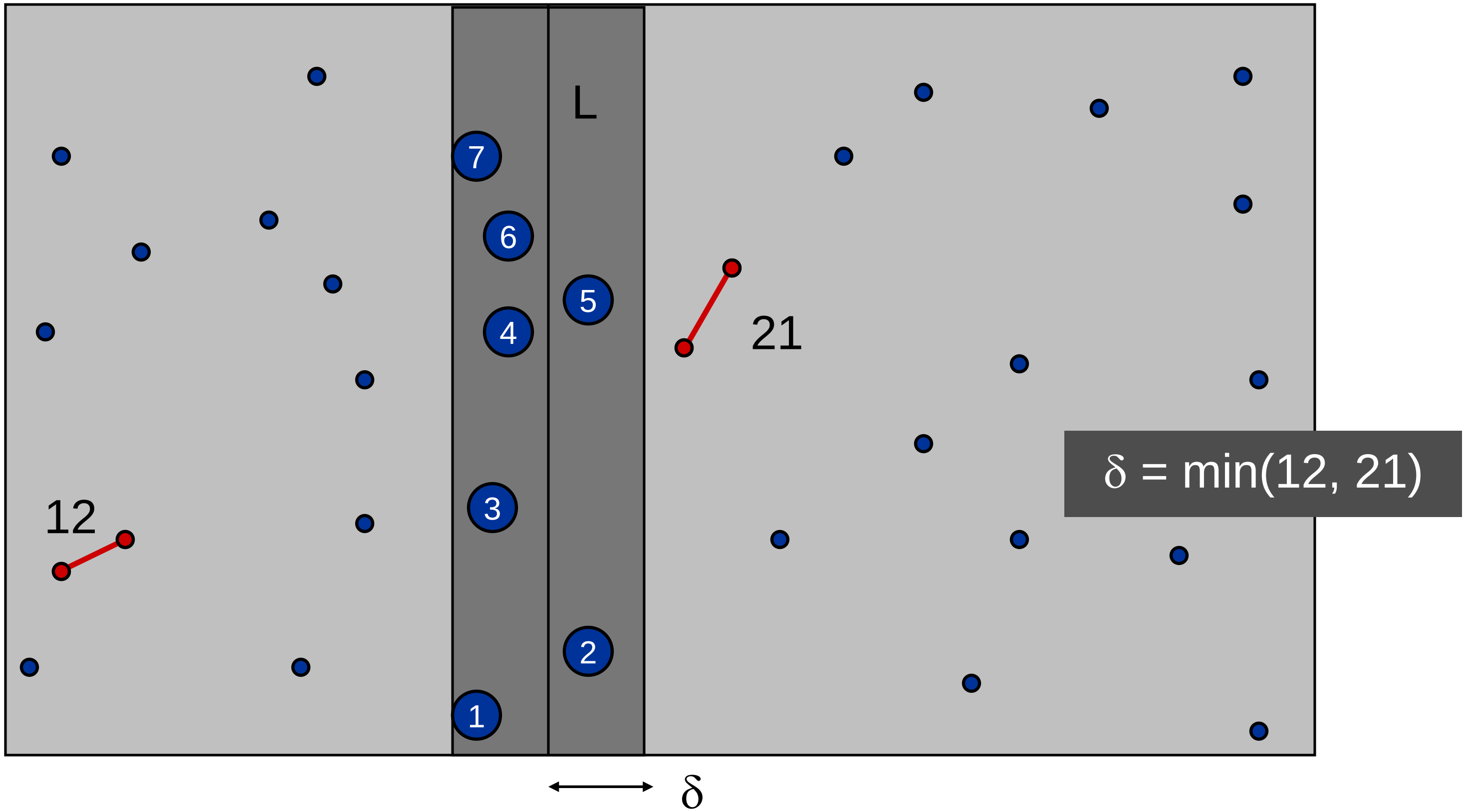
Closest Pair of Points

- Find closest pair with one point in each side, assuming that distance < δ.
  - Observation: only need to consider points within δ of line L.
  - Sort points in 2δ-strip by their y coordinate.



Closest Pair of Points

- Find closest pair with one point in each side, assuming that distance < δ.
  - Observation: only need to consider points within δ of line L.
  - Sort points in 2δ-strip by their y coordinate.
- **Theorem**: Only need to check distances of those within 11 positions in sorted list!



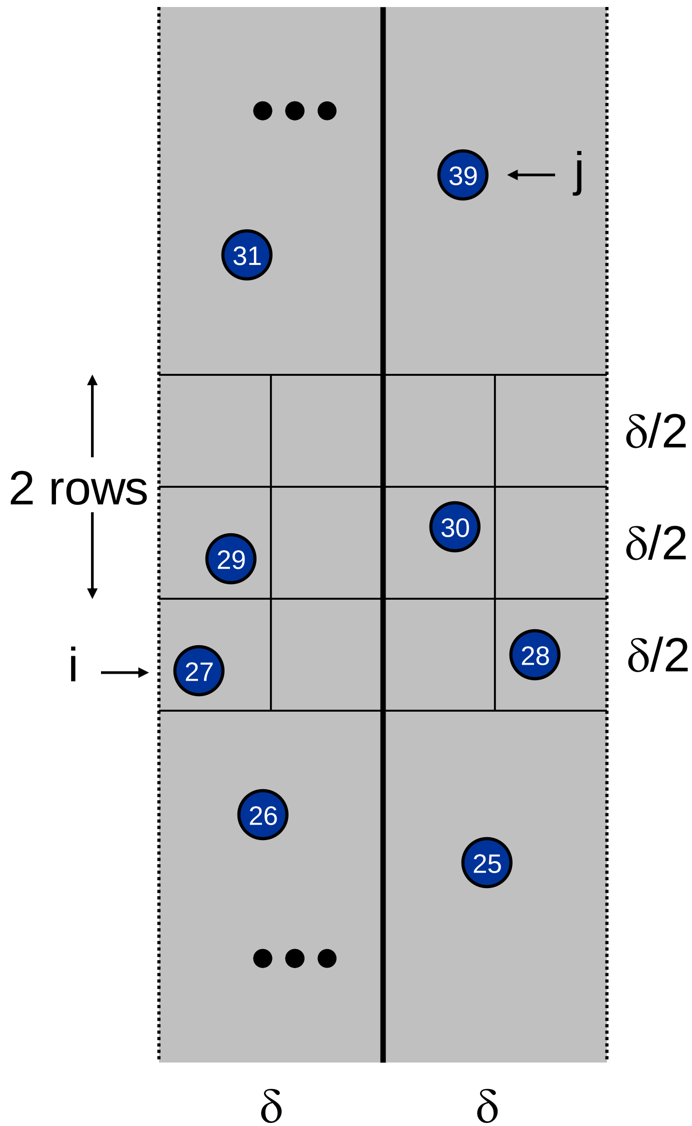
Closest Pair of Points

Definition. Let s_i be the point in the 2δ-strip, with the i^{th} smallest y-coordinate.

Claim. If $|i - j| \geq 12$, then the distance between s_i and s_j is at least δ.

Proof:

- No two points lie in same δ/2-by-δ/2 box because otherwise min distance would by < δ.
- Two points at least 2 rows apart have distance ≥ 2(δ/2). ▪



Fact. Still true if we replace 12 with 7.

Closest Pair Algorithm

```
Closest-Pair(p₁, …, pₙ) {
   Compute separation line L such that half the points
   are on one side and half on the other side.

   δ₁ = Closest-Pair(left half)
   δ₂ = Closest-Pair(right half)
   δ  = min(δ₁, δ₂)

   Delete all points further than δ from separation line L

   Sort remaining points by y-coordinate.

   Scan points in y-order and compare distance between
   each point and next 11 neighbors. If any of these
   distances is less than δ, update δ.

   return δ.
}
```

O(n log n)

2T(n / 2)

O(n)

O(n log n)

O(n)

Closest Pair of Points: Analysis

- Running time.

$$T(n) \le 2T(n/2) + O(n \log n) \Rightarrow T(n) = O(n \log^2 n)$$

by case 2 of Mater Theorem

- Q. Can we achieve O(n log n)?
- A. Yes. Don't sort points in strip from scratch each time.
  - Sort entire point set by x-coordinate only once
  - Each recursive call takes as input a set of points sorted by x coordinates and returns the same points sorted by y coordinate (together with the closest pair)
  - Create new y-sorted list by merging two outputs from recursive calls

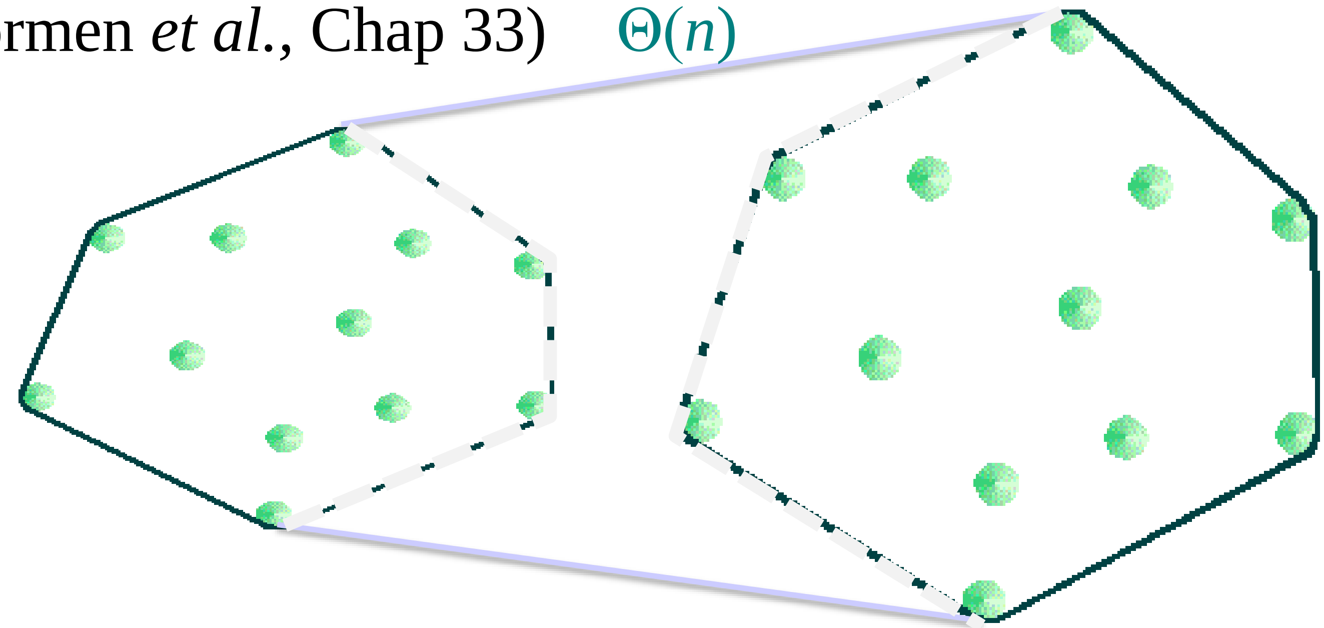
$$TotalTime(n) = O(n \log(n)) + T(n)$$

$$T(n) \le 2T(n/2) + O(n) \Rightarrow T(n) = O(n \log n)$$

Divide and Conquer in Low-Dimensional Geometry

- Powerful technique for low-dimensional geometric problems
  - Intuition: points in different parts of the plane don't interfere too much
  - Example: convex hull in O(n log (n)) time a la MergeSort

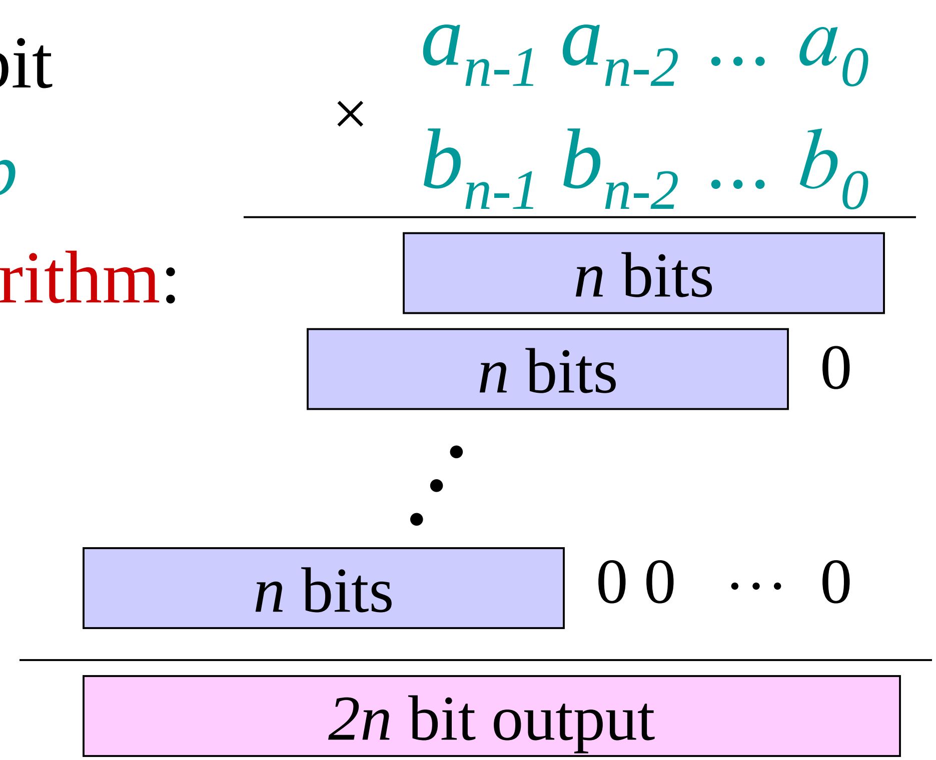
1. Convex-Hull(left-half) $T(n/2)$
2. Convex-Hull(right-half) $T(n/2)$
3. Merge (see Cormen *et al.*, Chap 33) $\Theta(n)$



Integer multiplication

Arithmetic on Large Integers

- **Addition**: Given n-bit integers a, b (in binary), compute $c=a+b$
  - O(n) bit operations.
- **Multiplication**: Given n-bit integers a, b, compute $c=ab$
- **Naïve (grade-school) algorithm**:
  - Write a,b in binary
  - Compute n intermediate products
  - Do n additions
  - Total work: $\Theta(n^2)$



Multiplying large integers

- **Divide and Conquer** (warmup):
  - Write $a = A_1\, 2^{n/2} + A_0$
    $b = B_1\, 2^{n/2} + B_0$
  - We want $ab = A_1 B_1\, 2^n + (A_1 B_0 + B_1 A_0)\, 2^{n/2} + A_0 B_0$
  - Multiply $n/2$ –bit integers recursively
  - $T(n) = 4T(n/2) + \Theta(n)$
  - Alas! this is still $\Theta(n^2)$ (Master Theorem, Case 1)

Multiplying large integers

- **Divide and Conquer** (Karatsuba's algorithm):
  - Write $a = A_1 \, 2^{n/2} + A_0$
    $b = B_1 \, 2^{n/2} + B_0$
  - We want $ab = A_1B_1 \, 2^n + (A_1B_0 + B_1A_0) \, 2^{n/2} + A_0B_0$
  - Multiply $n/2$ –bit integers recursively
  - Karatsuba's idea:
    $(A_0+A_1)\,(B_0 + B_1) = A_0B_0 + A_1B_1 + (A_0B_1 + B_1A_0)$
  - We can get away with 3 multiplications! (in yellow)
    $\mathbf{x} = A_1B_1 \qquad \mathbf{y} = A_0B_0 \qquad \mathbf{z} = (A_0+A_1)(B_0+B_1)$
  - Now we use $ab = A_1B_1 \, 2^n + (A_1B_0 + B_1A_0) \, 2^{n/2} + A_0B_0$
    $= \mathbf{x} \; 2^n + (\mathbf{z}-\mathbf{x}-\mathbf{y}) \, 2^{n/2} + \mathbf{y}$

Implementation of Multiplication

MULTIPLY (n, x, y)

\\ x and y are n-bit integers

\\ Assume n is a power of 2 for simplicity

1. **If** $n < 2$ **then** use grade-school algorithm **else**
2. $x_1 \leftarrow x$ div $2^{n/2}$; $y_1 \leftarrow y$ div $2^{n/2}$;
3. $x_0 \leftarrow x$ mod $2^{n/2}$; $y_0 \leftarrow y$ mod $2^{n/2}$.
4. **A** $\leftarrow$ MULTIPLY($n/2$, x_1, y_1)
5. **C** $\leftarrow$ MULTIPLY($n/2$, x_0, y_0)
6. **B** $\leftarrow$ MULTIPLY($n/2$, x_1+x_0 , y_1+y_0)
7. **Output** **A** 2^n + (**B-A-C**)$2^{n/2}$ + **C**

Integer Multiplication: Run Time

- The resulting recurrence
  $$T(n) = 3T(n/2) + \Theta(n)$$
- Master Theorem, **Case 1**:
  $$T(n) = \Theta\left(n^{\log_2 3}\right) = \Theta(n^{1.59\ldots})$$
- Algorithm based on Fast Fourier Transform: $\Theta(n \log n \log\log n)$ (more on it later in the course).
- Fürer's Algorithm (2007): $n \cdot \log n \cdot 2^{\Theta(\log^* n)}$

Matrix multiplication

Matrix multiplication

Input: $A = [a_{ij}]$, $B = [b_{ij}]$.
Output: $C = [c_{ij}] = A \times B$.

$$\left.\begin{array}{l} \\ \\ \end{array}\right\} \quad i, j = 1, 2, \ldots, n.$$

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

Standard algorithm

for $i \leftarrow 1$ **to** n
    do for $j \leftarrow 1$ **to** n
        do $c_{ij} \leftarrow 0$
            for $k \leftarrow 1$ **to** n
                do $c_{ij} \leftarrow c_{ij} + a_{ik} \times b_{kj}$

Running time $= \Theta(n^3)$

Divide-and-conquer algorithm

IDEA:
$n \times n$ matrix = 2×2 matrix of $(n/2) \times (n/2)$ submatrices:

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$C \quad = \quad A \quad \times \quad B$$

$$\left.\begin{aligned} C_{11} &= (A_{11} \times B_{11}) + (A_{12} \times B_{21}) \\ C_{12} &= (A_{11} \times B_{12}) + (A_{12} \times B_{22}) \\ C_{21} &= (A_{21} \times B_{11}) + (A_{22} \times B_{21}) \\ C_{22} &= (A_{21} \times B_{12}) + (A_{22} \times B_{22}) \end{aligned}\right\}$$

recursive

8 mults of $(n/2) \times (n/2)$ submatrices
4 adds of $(n/2) \times (n/2)$ submatrices

Analysis of D&C algorithm

$$T(n) = 8\,T(n/2) + \Theta(n^2)$$

 # submatrices

submatrix size

work adding submatrices

$$n^{\log_b a} = n^{\log_2 8} = n^3 \;\Rightarrow\; \text{CASE 1} \;\Rightarrow\; T(n) = \Theta(n^3).$$

No better than the ordinary algorithm.

Strassen's idea

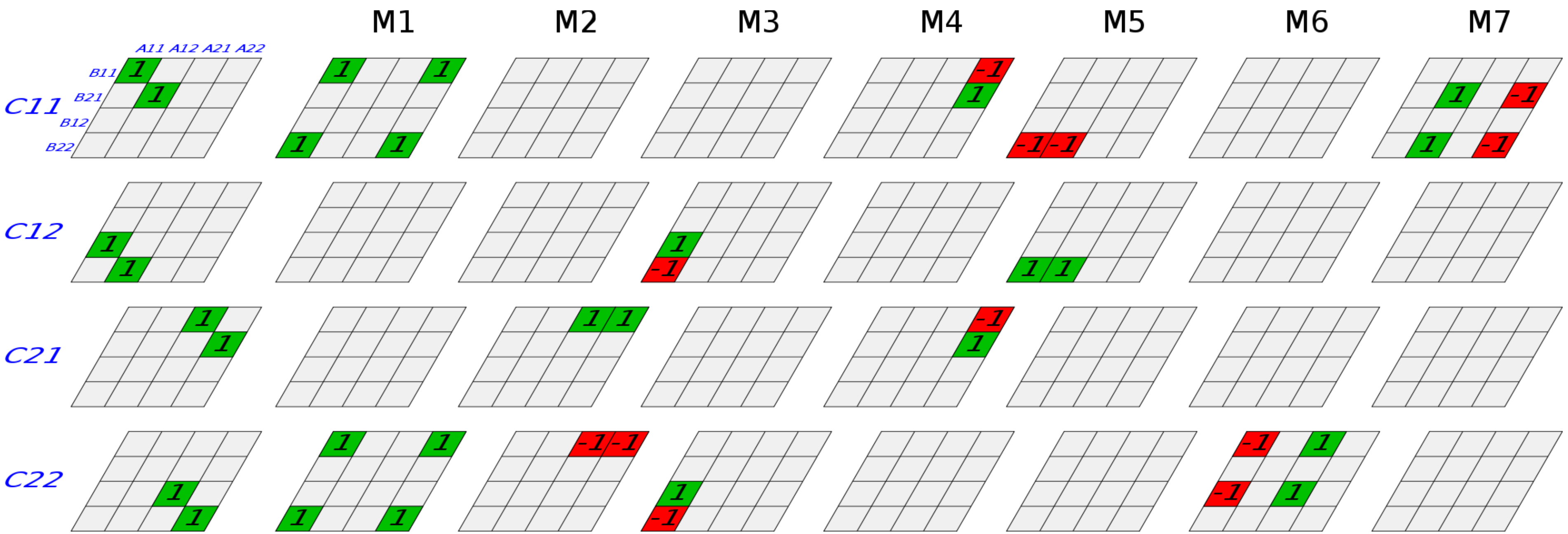
- Multiply 2×2 matrices with only 7 recursive mults.

$$
\begin{aligned}
M_1 &= A_{11} \times (B_{12} - B_{22}) \\
M_2 &= (A_{11} + A_{12}) \times B_{22} \\
M_3 &= (A_{21} + A_{22}) \times B_{11} \\
M_4 &= A_{22} \times (B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{22}) \times (B_{11} + B_{22}) \\
M_6 &= (A_{12} - A_{22}) \times (B_{21} + B_{22}) \\
M_7 &= (A_{11} - A_{21}) \times (B_{11} + B_{12})
\end{aligned}
$$

$$
\begin{aligned}
C_{11} &= M_5 + M_4 - M_2 + M_6 \\
C_{12} &= M_1 + M_2 \\
C_{21} &= M_3 + M_4 \\
C_{22} &= M_5 + M_1 - M_3 - M_7
\end{aligned}
$$

7 mults, 18 adds/subs.
Note: No reliance on commutativity of multiplication!

Pictorial Explanation



- The left column represents $2{\times}2$ multiplication. Naïve matrix multiplication requires one multiplication for each "1" of the left column.
- Each of the other columns represents a single one of the 7 multiplications in the algorithm, and the sum of the columns gives the full matrix multiplication on the left.

.
Source: wikipedia.

Strassen's algorithm

1. ***Divide:*** Partition A and B into $(n/2)$ x $(n/2)$ submatrices. Form terms to be multiplied using $+$ and $-$.
2. ***Conquer:*** Perform 7 multiplications of $(n/2)$ x $(n/2)$ submatrices recursively.
3. ***Combine:*** Form product matrix C using $+$ and $-$ on $(n/2)$ x $(n/2)$ submatrices.

$$T(n) = 7\,T(n/2) + \Theta(n^2)$$

Analysis of Strassen

$$T(n) = 7\,T(n/2) + \Theta(n^2)$$

$$n^{\log_b a} = n^{\log_2 7} \approx n^{2.81} \;\Rightarrow\; \text{CASE 1} \;\Rightarrow\; T(n) = \Theta(n^{\lg 7}).$$

- Number 2.81 may not seem much smaller than 3.
- But the difference is in the exponent.
- The impact on running time is significant.
- Strassen's algorithm beats the ordinary algorithm on today's machines for $n \geq 32$ or so.

Coppersmith-Winograd, 1987 : $O(n^{2.376\cdots})$.

Currently best, 2014: $O(n^{2.3728\cdots})$.

Median and Order Statistics

Order statistics

Select the ith smallest of n elements (the element with ***rank*** i).

- $i = 1$: ***minimum***;
- $i = n$: ***maximum***;
- $i = \lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$: ***median***.

Naive algorithm: Sort and index ith element.

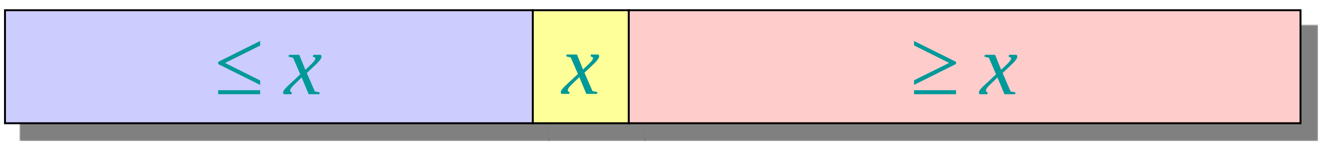
Worst-case running time $= \Theta(n \lg n) + \Theta(1)$
$= \Theta(n \lg n)$,

using merge sort or heapsort (*not* quicksort).

Divide and conquer

Order Statistics in an n-element array:

1. ***Divide:*** Partition the array into two subarrays around a ***pivot*** x such that elements in lower subarray $\leq x \leq$ elements in upper subarray.



2. ***Conquer:*** Recurse on one subarray.
3. ***Combine:*** Trivial.

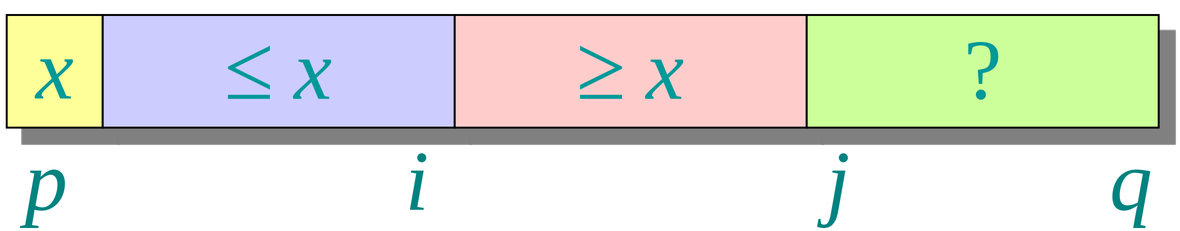
Key: *Linear-time partitioning subroutine.*

Partitioning subroutine

PARTITION(A, p, q) ⊳ $A[p\,.\,.\,q]$
    $x \leftarrow A[p]$ ⊳ pivot $= A[p]$
    $i \leftarrow p$
    for $j \leftarrow p + 1$ **to** q
        do if $A[j] \le x$
            then $i \leftarrow i + 1$
                exchange $A[i] \leftrightarrow A[j]$
    exchange $A[p] \leftrightarrow A[i]$
    return i

Running time $= O(n)$ for n elements.

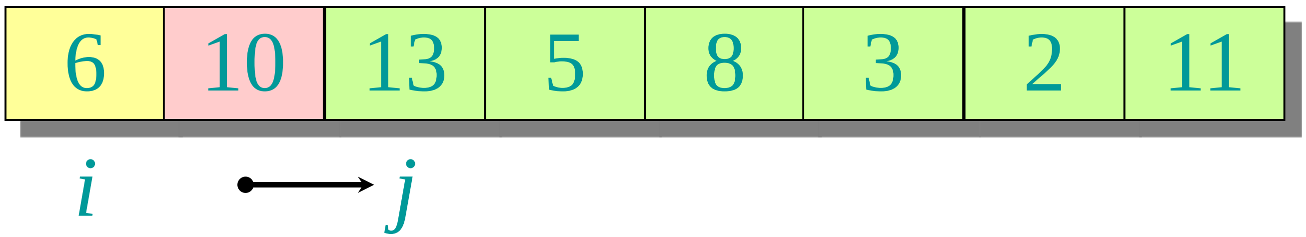
Invariant:



Example of partitioning

6	10	13	5	8	3	2	11
i	j						

Example of partitioning



Example of partitioning

6	10	13	5	8	3	2	11

i $\longrightarrow$ j

Example of partitioning

6	10	13	5	8	3	2	11
6	5	13	10	8	3	2	11
	i		*j*				

Example of partitioning

6	10	13	5	8	3	2	11

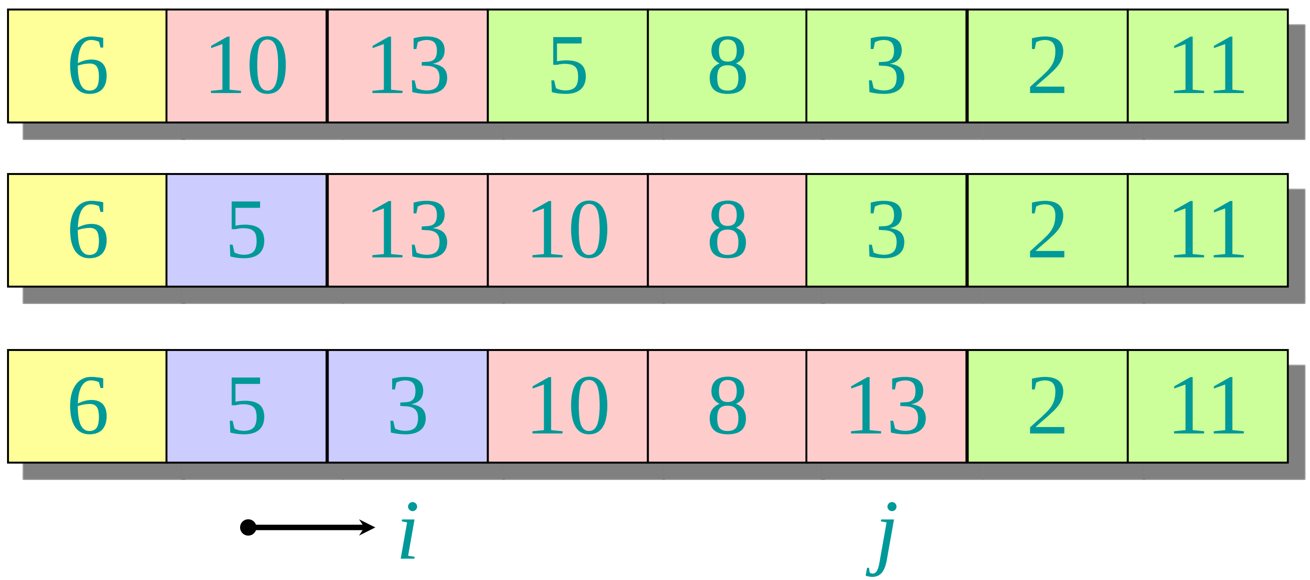
6	5	13	10	8	3	2	11
	i			j			

Example of partitioning

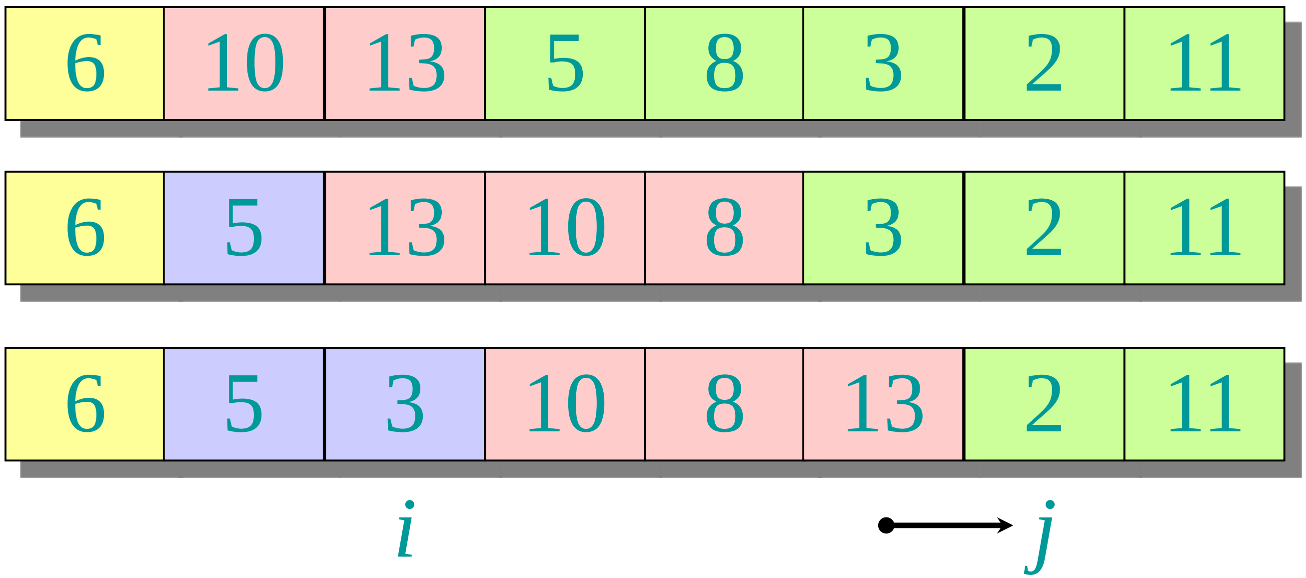
6	10	13	5	8	3	2	11

6	5	13	10	8	3	2	11
	i				j		

Example of partitioning



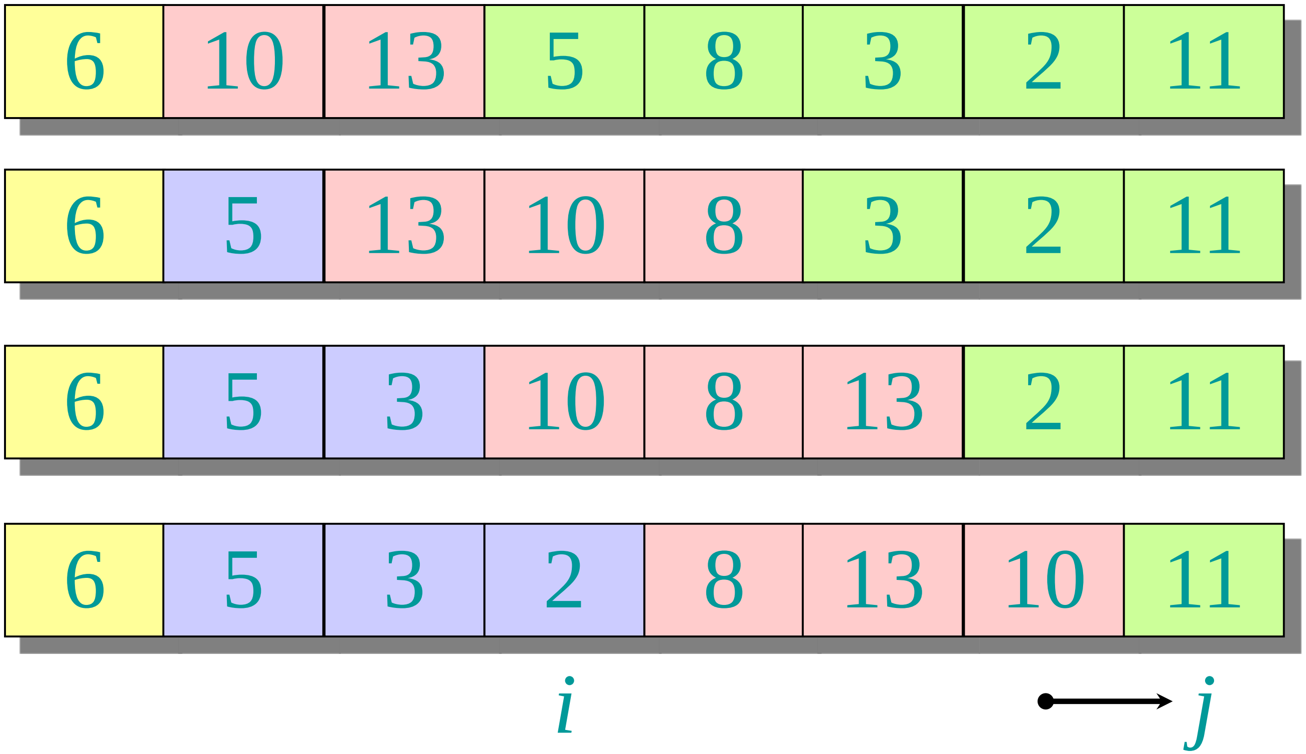
Example of partitioning



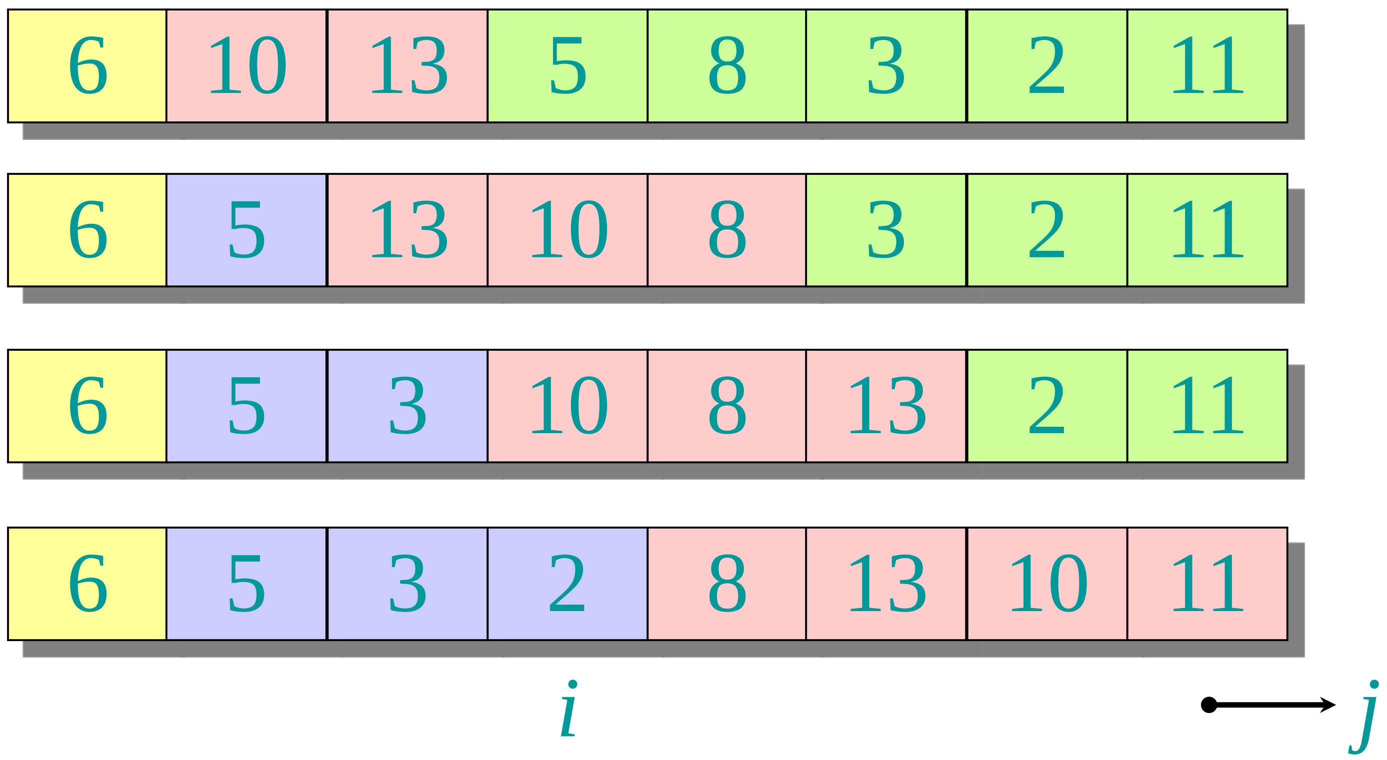
Example of partitioning

6	10	13	5	8	3	2	11
6	5	13	10	8	3	2	11
6	5	3	10	8	13	2	11
6	5	3	2	8	13	10	11
			i			j	

Example of partitioning



Example of partitioning



Example of partitioning

6	10	13	5	8	3	2	11

6	5	13	10	8	3	2	11

6	5	3	10	8	13	2	11

6	5	3	2	8	13	10	11

2	5	3	6	8	13	10	11

i

Divide-and-conquer algorithm

SELECT(A, p, q, i) ⊳ ith smallest of $A[p \,..\, q]$

  if $p = q$ **then return** $A[p]$

  $r \leftarrow$ pivot ⊳ **Later: how to choose the pivot**

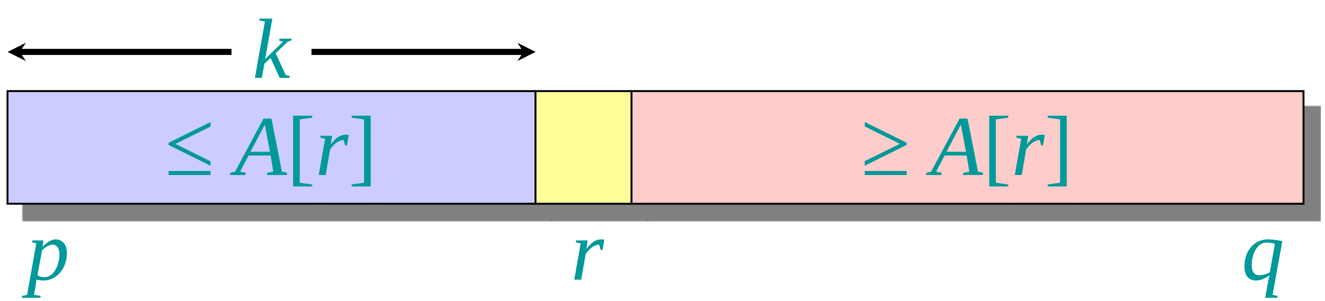
  $k \leftarrow r - p + 1$ ⊳ $k = \text{rank}(A[r])$

  if $i = k$ **then return** $A[r]$

  if $i < k$

    then return SELECT$(A, p, r - 1, i)$

    else return SELECT$(A, r + 1, q, i - k)$



Example

Select the $i = 7$th smallest:

6	10	13	5	8	3	2	11	$i = 7$

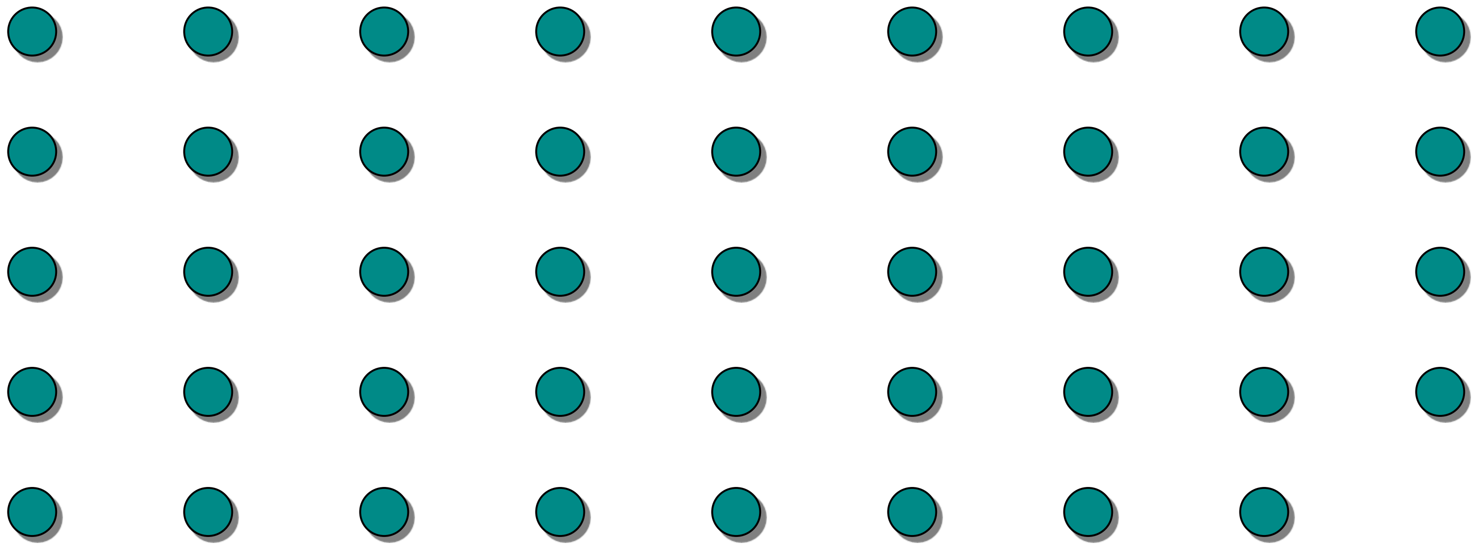
pivot

Partition:

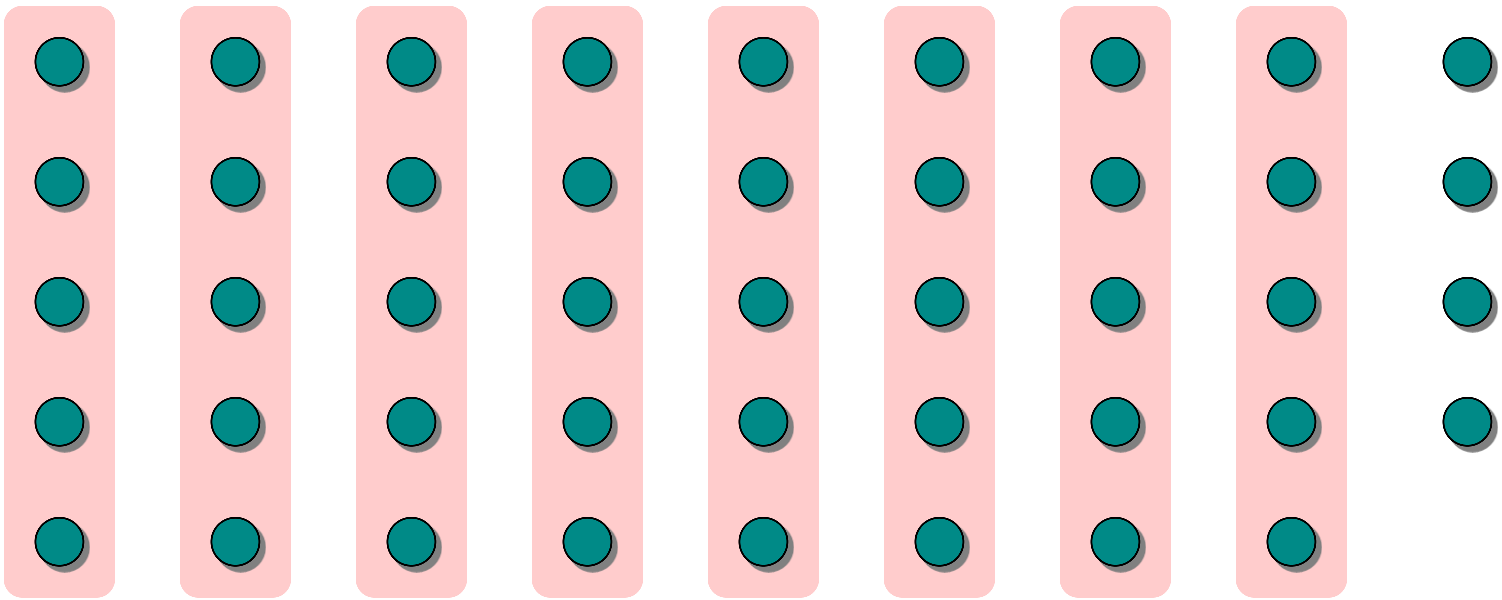
2	5	3	6	8	13	10	11	$k = 4$

Select the $7 - 4 = 3$rd smallest recursively.

Choosing the pivot

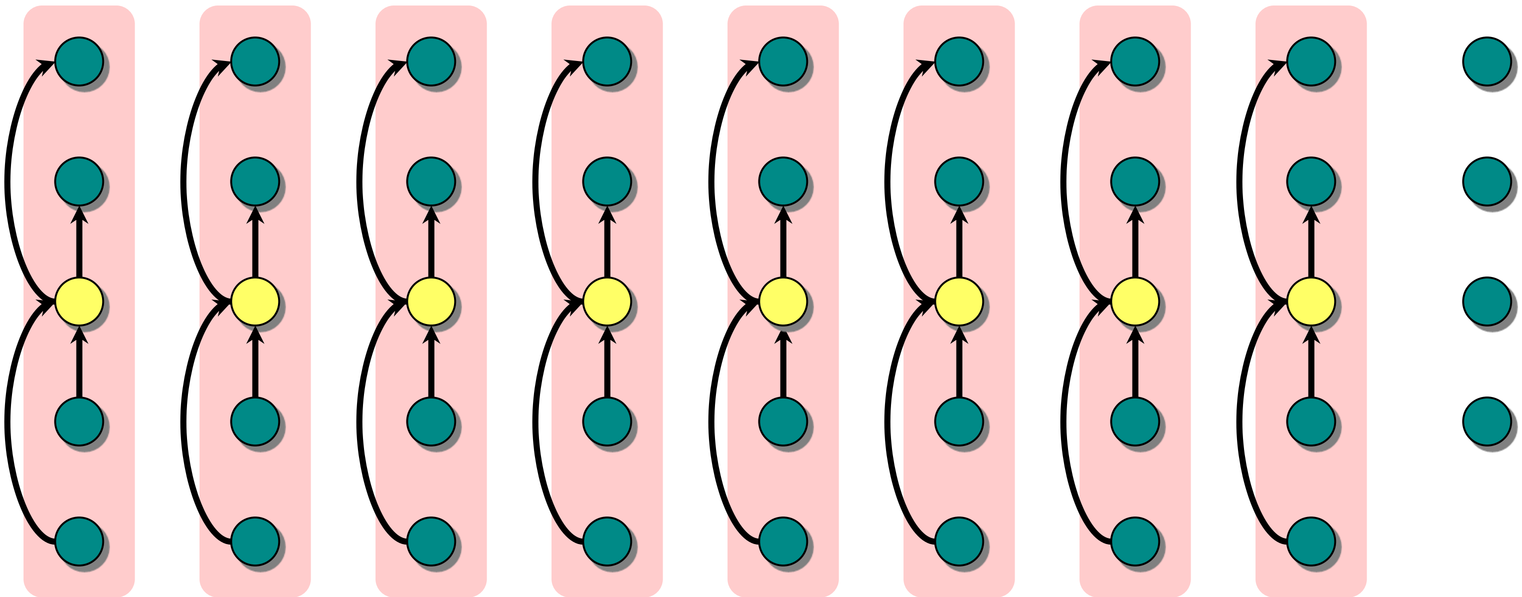


Choosing the pivot

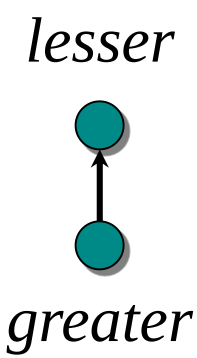


1. Divide the n elements into groups of 5.

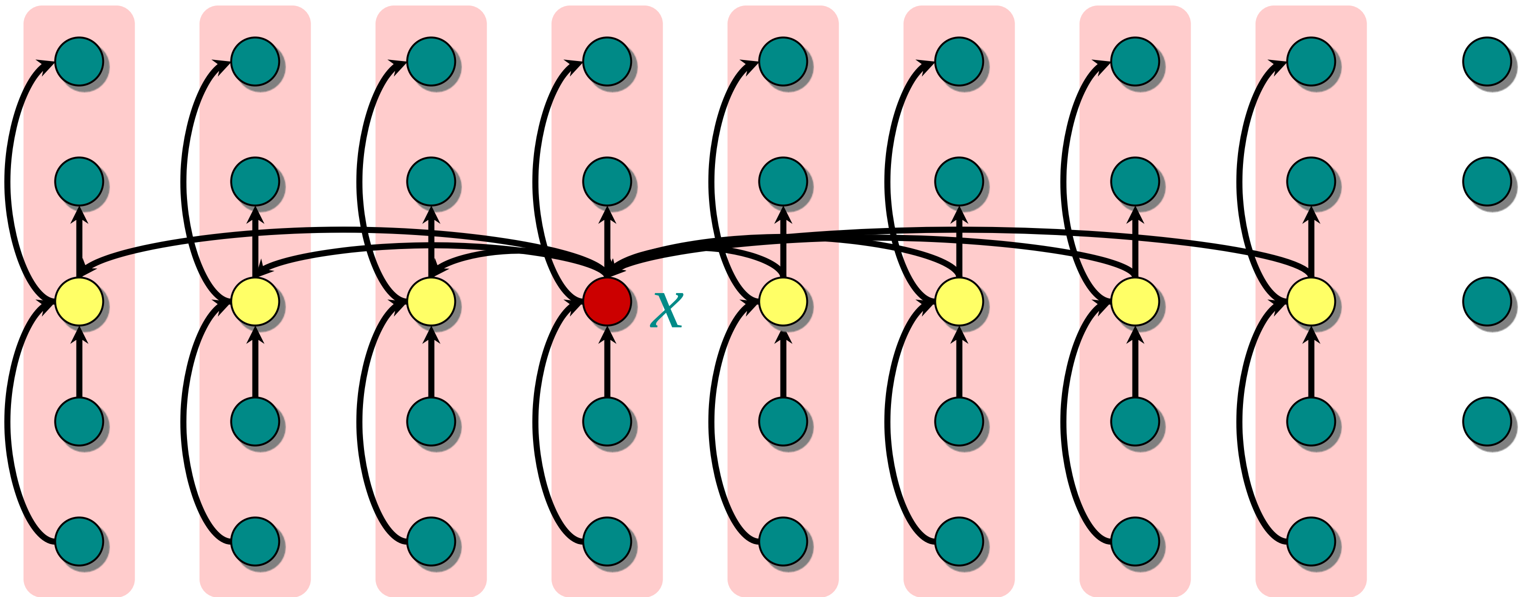
Choosing the pivot



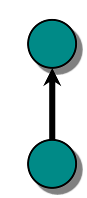
1. Divide the n elements into groups of 5. Find the median of each 5-element group.



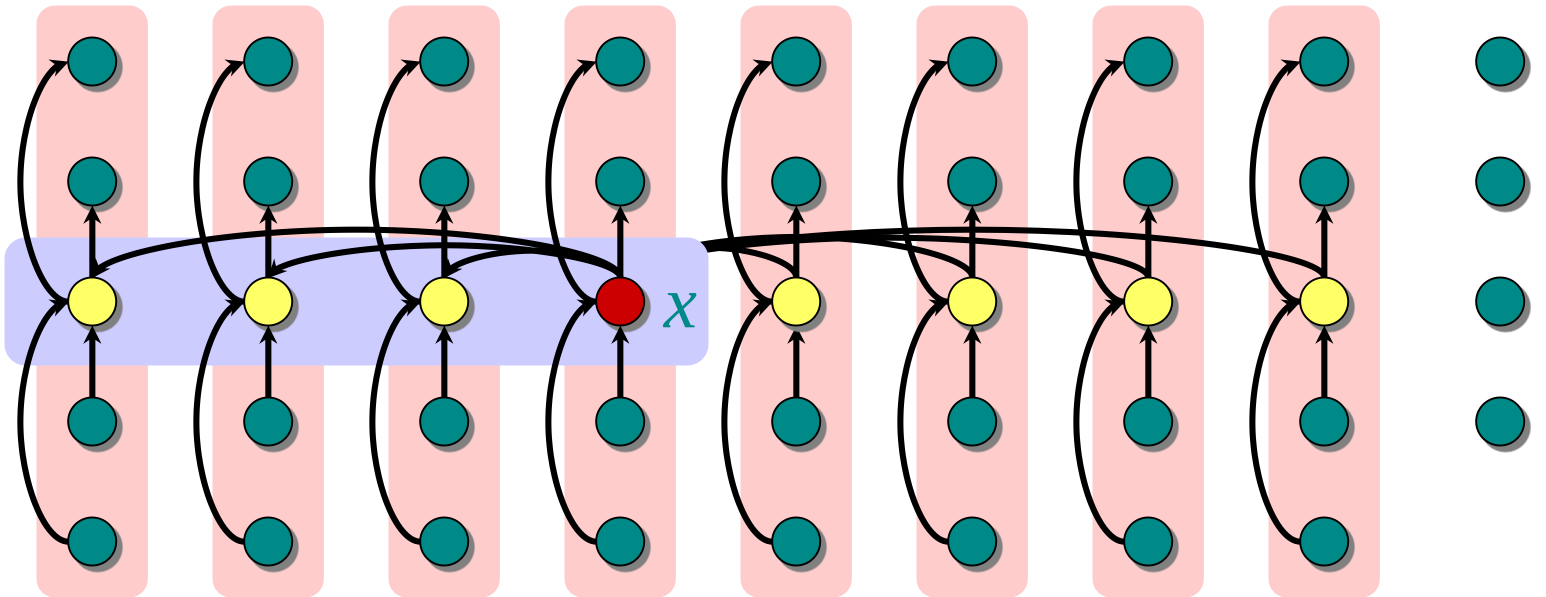
Choosing the pivot



1. Divide the n elements into groups of 5. Find the median of each 5-element group.
2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.

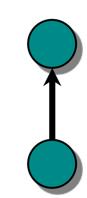


Analysis



At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ group medians.

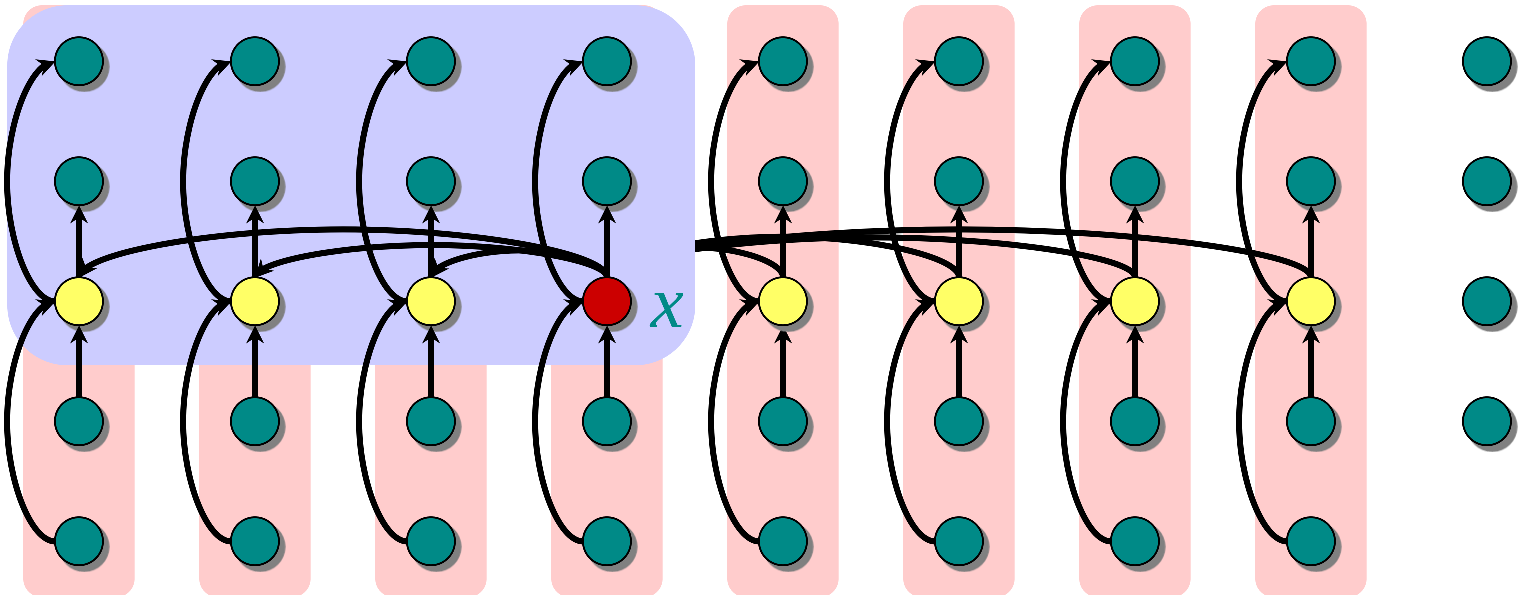
lesser



greater

Analysis

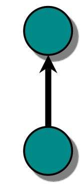
(Assume all elements are distinct.)



At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ group medians.

- Therefore, at least $3\lfloor n/10 \rfloor$ elements are $\leq x$.

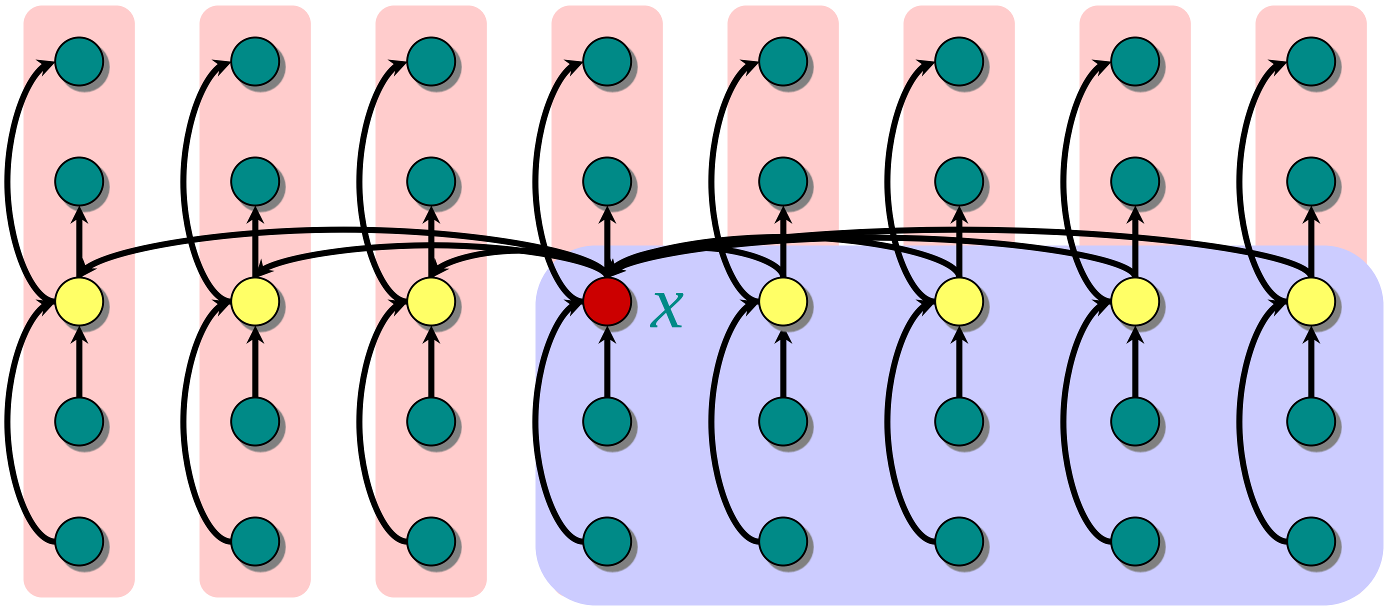
lesser



greater

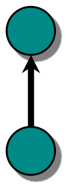
Analysis

(Assume all elements are distinct.)



At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ group medians.

- Therefore, at least $3\lfloor n/10 \rfloor$ elements are $\leq x$.
- Similarly, at least $3\lfloor n/10 \rfloor$ elements are $\geq x$.



Developing the recurrence

$T(n)$	SELECT(i, n)
$\Theta(n)$	1. Divide the n elements into groups of 5. Find the median of each 5-element group.
$T(n/5)$	2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.
$\Theta(n)$	3. Partition around the pivot x. Let $k = \text{rank}(x)$.
$T(7n/10)$	4. **if** $i = k$ **then return** x **elseif** $i < k$ **then** recursively SELECT the ith smallest element in the lower part **else** recursively SELECT the $(i–k)$th smallest element in the upper part

Solving the recurrence

$$T(n) = T\left(\frac{1}{5}n\right) + T\left(\frac{7}{10}n\right) + cn$$

$T(n) \geq cn$

Recursion Tree: $T(n) \leq cn\left(1 + \frac{9}{10} + \left(\frac{9}{10}\right)^2 + \ldots\right)$

$$= cn\frac{1}{1 - \frac{9}{10}} = O(n)$$

T(n)=$\Theta(n)$

Conclusion

- In practice, this algorithm runs slowly, because the constant in front of n is large.
- There is a randomized algorithm that runs in expected linear time.
- The randomized algorithm is far more practical.

Exercise: *Why not divide into groups of 3?*