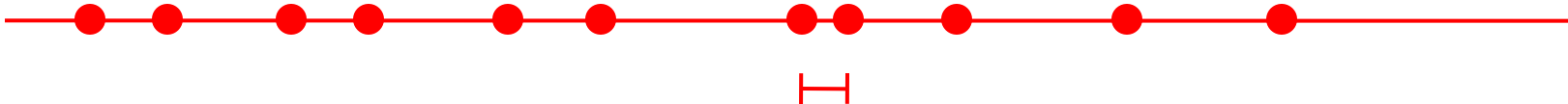

A Divide & Conquer Example: Closest Pair of Points

closest pair of points: 1 dimensional version

Given n points on the real line, find the closest pair



Closest pair is *adjacent* in ordered list

Time $O(n \log n)$ to sort, if needed

Plus $O(n)$ to scan adjacent pairs

closest pair of points: 2 dimensional version

Closest pair. 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 $\Theta(n^2)$ time.

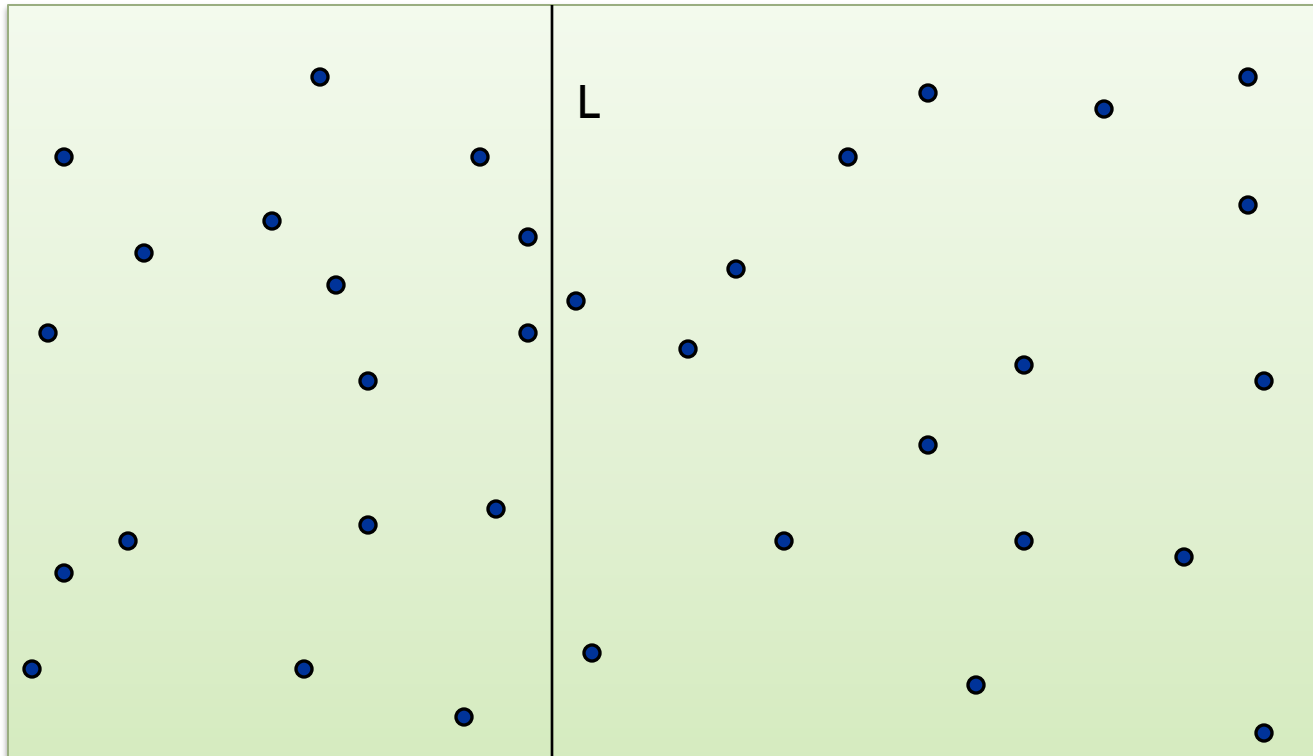
1-D version. $O(n \log n)$ easy if points are on a line.

Assumption. No two points have same x coordinate.

↑
Just to simplify presentation

Algorithm.

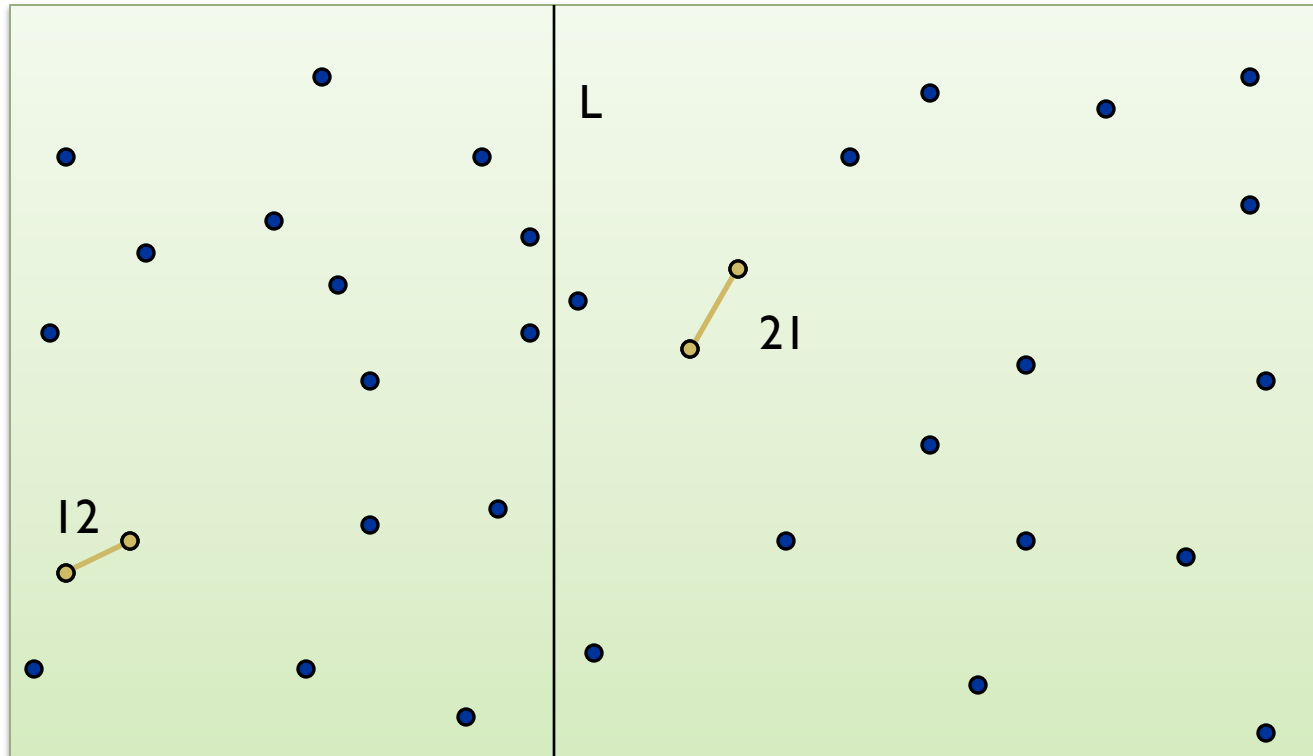
Divide: draw vertical line L with $\approx \mathbf{n/2}$ **points on** each side.



Algorithm.

Divide: draw vertical line L with $\approx n/2$ **points on** each side.

Conquer: find closest pair on each side, recursively.



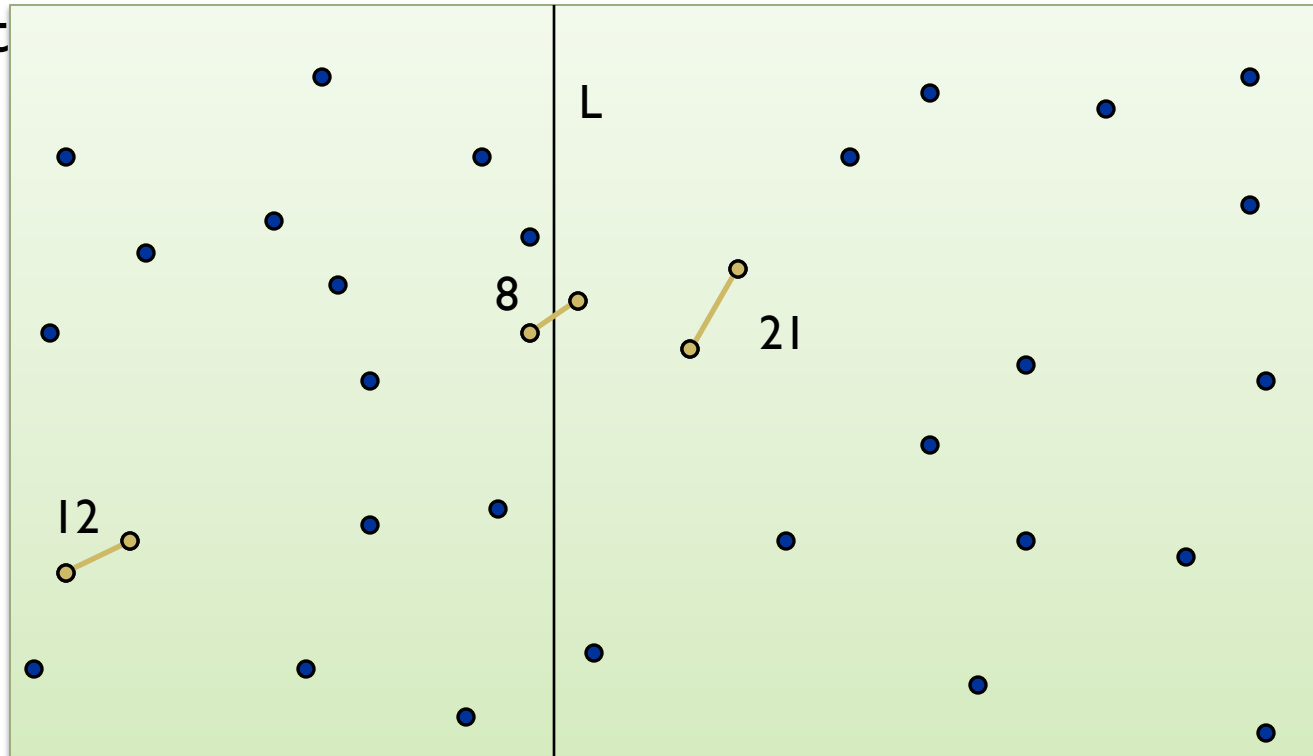
Algorithm.

Divide: draw vertical line L with $\approx n/2$ **points on** each side.

Conquer: find closest pair on each side, recursively.

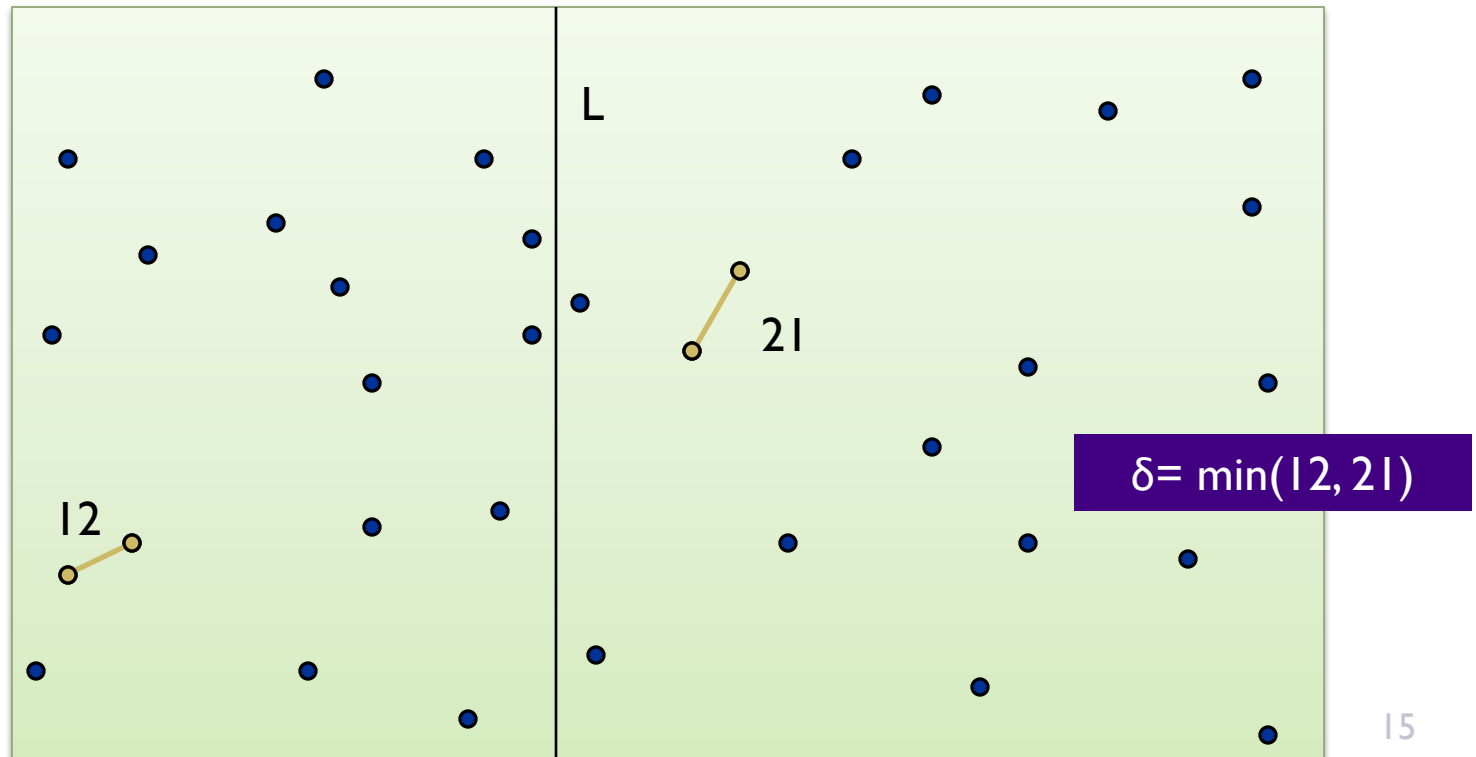
Combine to find closest pair overall

Ret



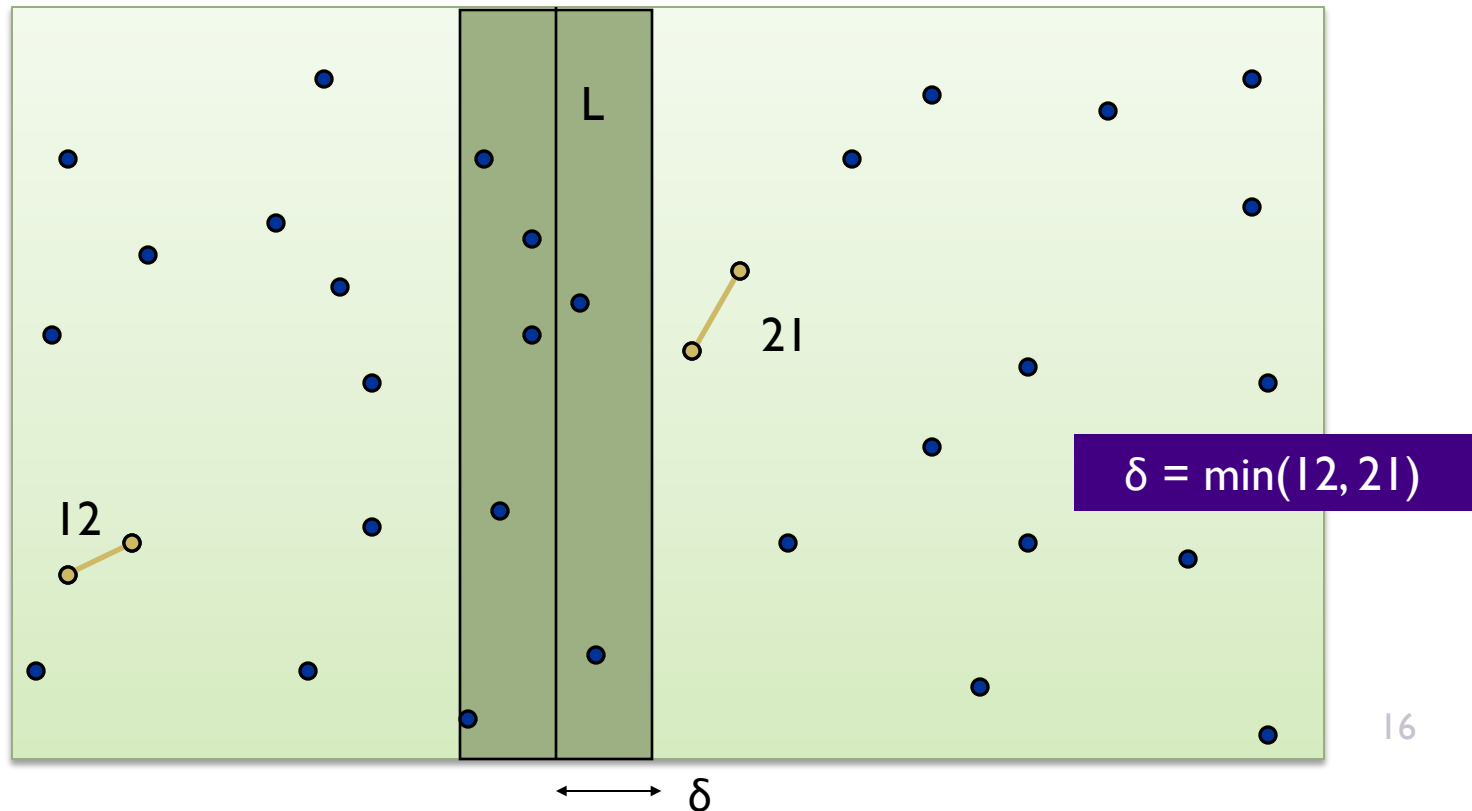
seems
like
 $\Theta(n^2)$?

Find closest pair with one point in each side,
assuming distance $< \delta$.



Find closest pair with one point in each side,
assuming distance $< \delta$.

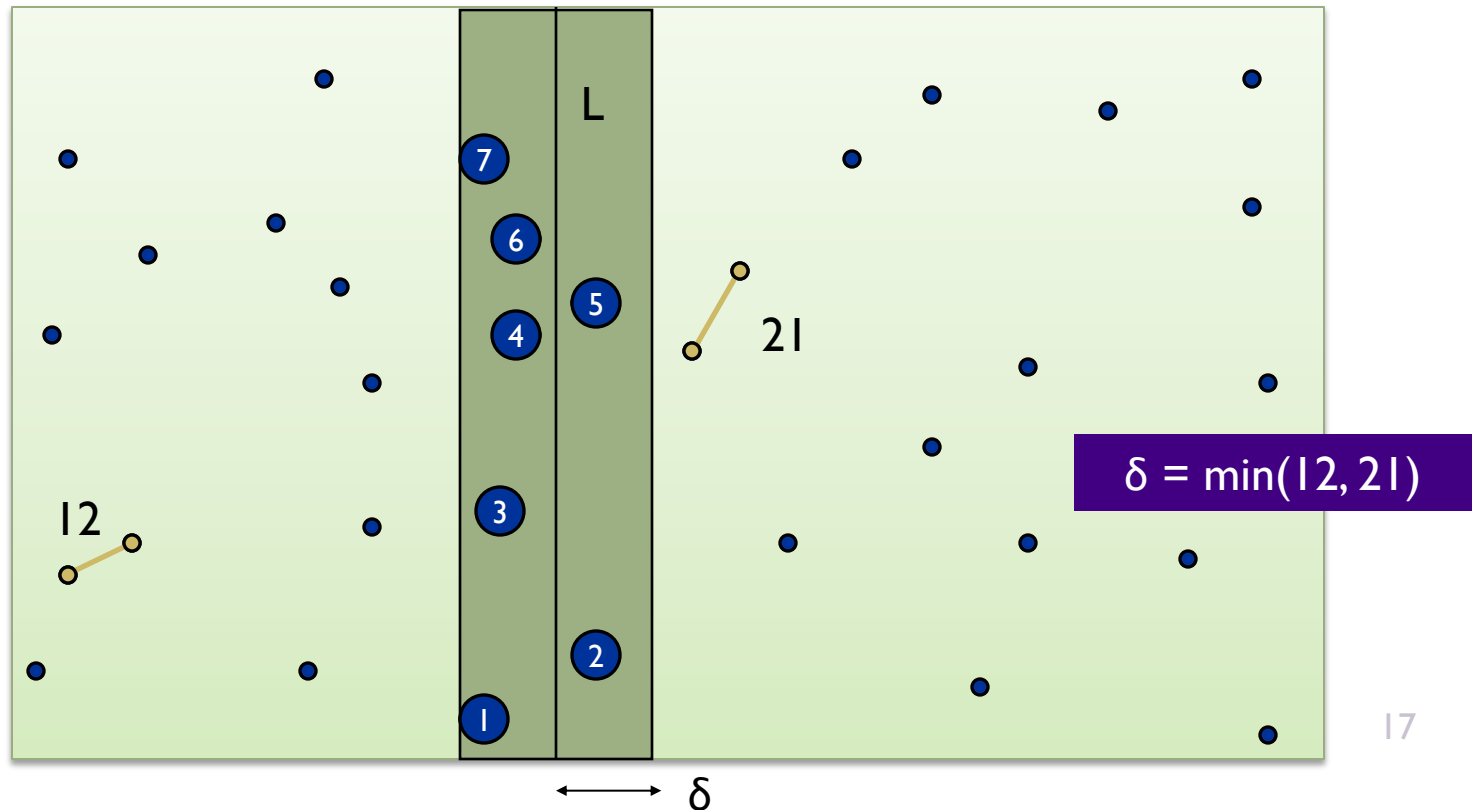
Observation: suffices to consider points within δ of line L .



Find closest pair with one point in each side, *assuming* distance $< \delta$.

Observation: suffices to consider points within δ of line L .

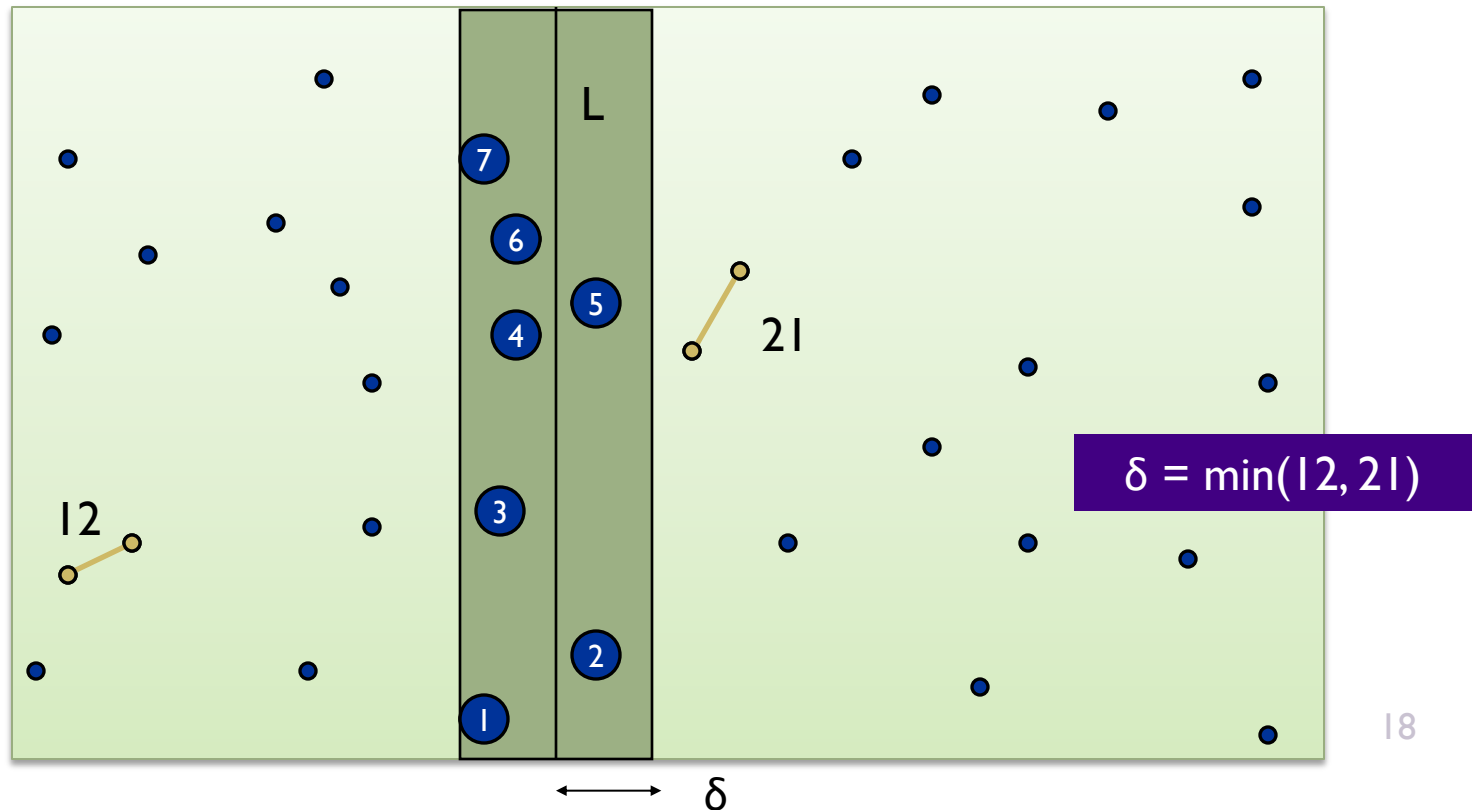
Almost the one-D problem again: Sort points in 2δ -strip by their y coordinate.



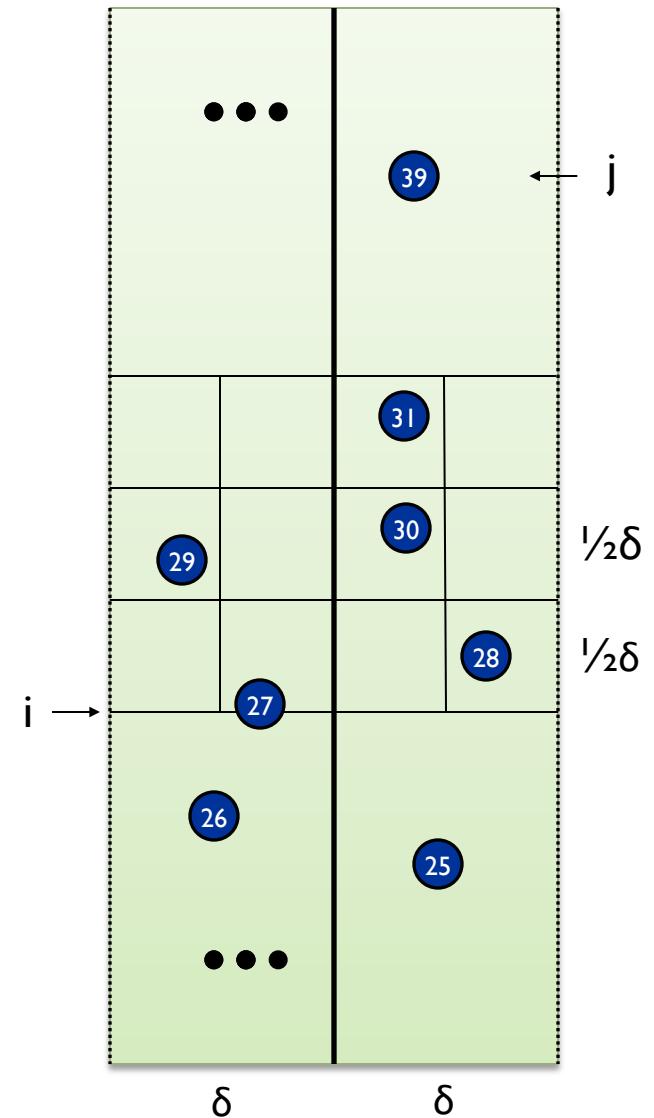
Find closest pair with one point in each side, *assuming* distance $< \delta$.

Observation: suffices to consider points within δ of line L .

Almost the one-D problem again: Sort points in 2δ -strip by their y coordinate. Only check pts within δ in sorted list!



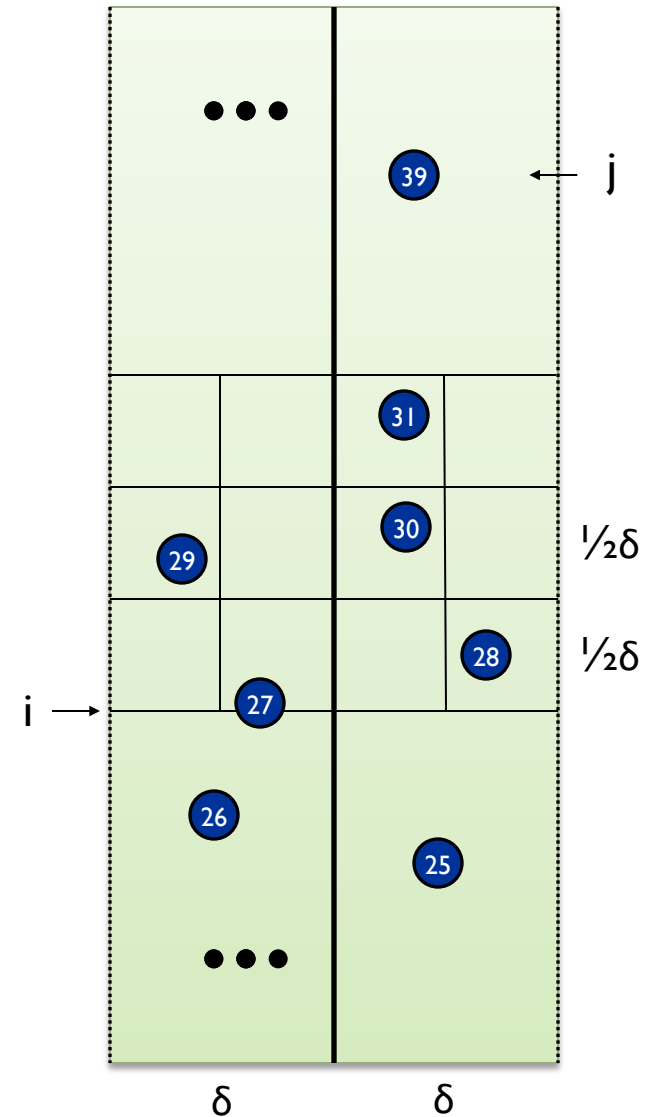
Claim: No two points lie in the same $\frac{1}{2}\delta$ -by- $\frac{1}{2}\delta$ box.



Claim: No two points lie in the same $\frac{1}{2}\delta$ -by- $\frac{1}{2}\delta$ box.

Pf: Such points would be within

$$\delta \sqrt{\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2} = \delta \sqrt{\frac{1}{2}} = \delta \frac{\sqrt{2}}{2} \approx 0.7\delta < \delta$$



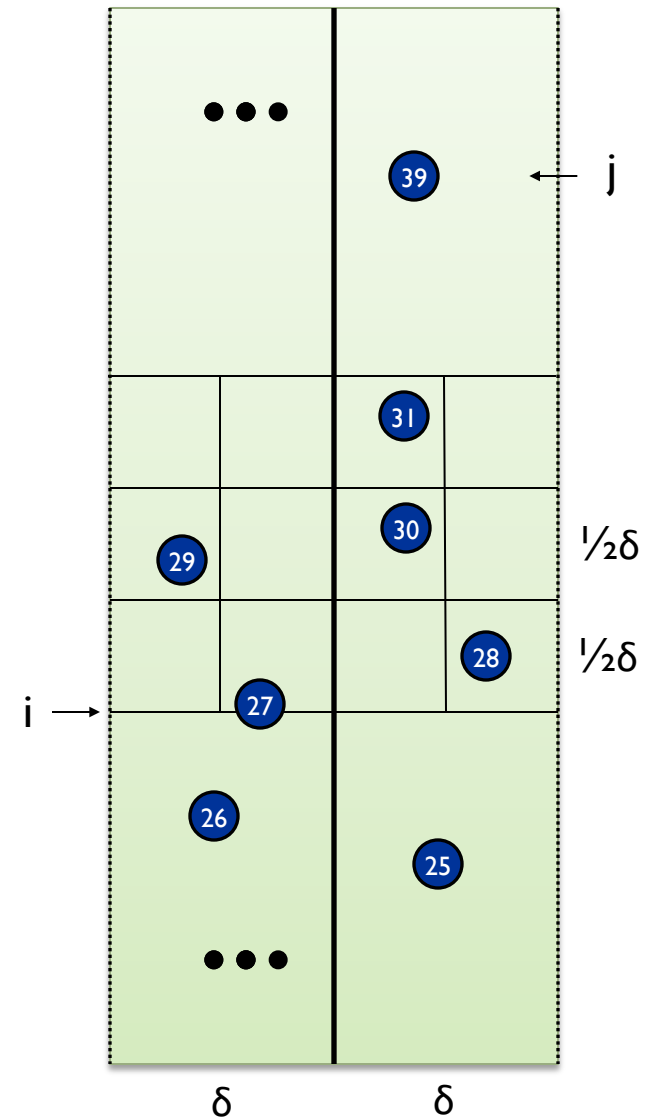
Claim: No two points lie in the same $\frac{1}{2}\delta$ -by- $\frac{1}{2}\delta$ box.

Pf: Such points would be within

$$\delta \sqrt{\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2} = \delta \sqrt{\frac{1}{2}} = \delta \frac{\sqrt{2}}{2} \approx 0.7\delta < \delta$$

Def. Let s_i have the i^{th} smallest y -coordinate among points in the 2δ -width-strip.

Claim: If $|i - j| > l$, then the distance between s_i and s_j is $> \delta$.



Claim: No two points lie in the same $\frac{1}{2}\delta$ -by- $\frac{1}{2}\delta$ box.

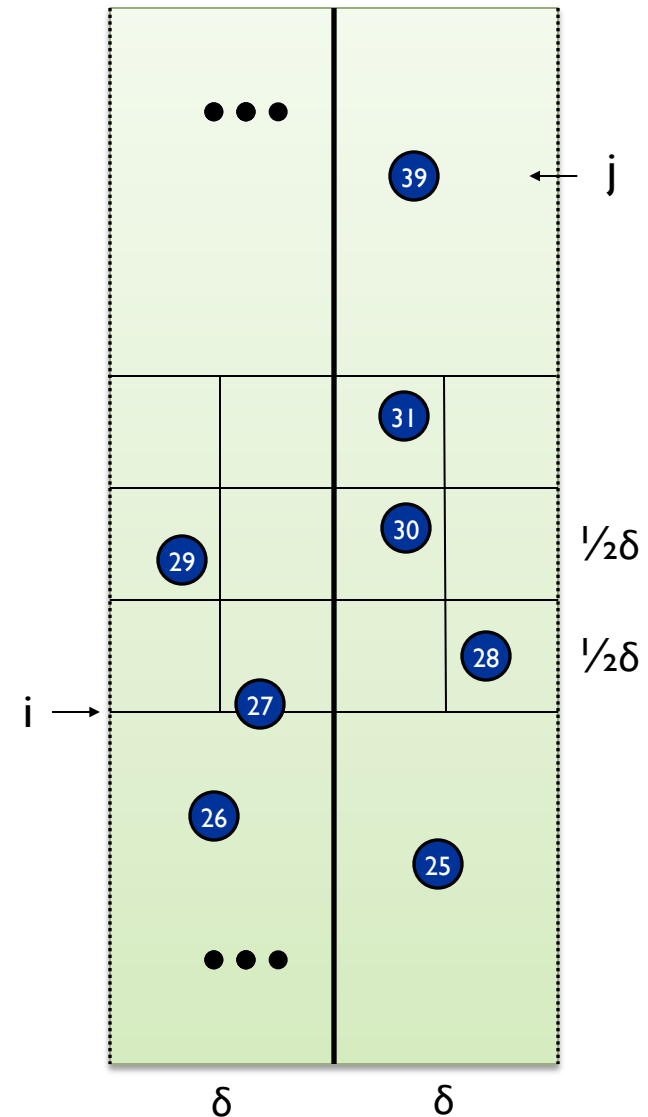
Pf: Such points would be within

$$\delta \sqrt{\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2} = \delta \sqrt{\frac{1}{2}} = \delta \frac{\sqrt{2}}{2} \approx 0.7\delta < \delta$$

Def. Let s_i have the i^{th} smallest y-coordinate among points in the 2δ -width-strip.

Claim: If $|i - j| > 1$, then the distance between s_i and s_j is $> \delta$.

Pf: only 1 boxes within $+\delta$ of $y(s_i)$.



```
Closest-Pair( $p_1, \dots, p_n$ ) {  
    if( $n \leq ??$ ) return ??  
  
    Compute separation line  $L$  such that half the points  
    are on one side and half on the other side.  
  
     $\delta_1$  = Closest-Pair(left half)  
     $\delta_2$  = Closest-Pair(right half)  
     $\delta$  = min( $\delta_1, \delta_2$ )  
  
    Delete all points further than  $\delta$  from separation line  
     $L$   
  
    Sort remaining points  $p[1] \dots p[m]$  by y-coordinate.  
  
    for  $i = 1 \dots m$   
        for  $k = 1 \dots 11$   
            if  $i+k \leq m$   
                 $\delta = \min(\delta, \text{distance}(p[i], p[i+k]));$   
  
    return  $\delta$ .  
}
```

Analysis, I: Let $D(n)$ be the number of pairwise distance calculations in the Closest-Pair Algorithm when run on $n > 1$ points

$$D(n) \leq \begin{cases} 0 & n = 1 \\ 2D(n/2) + 11n & n > 1 \end{cases} \Rightarrow D(n) = O(n \log n)$$

BUT – that's only the number of *distance calculations*

What if we counted running time?

Analysis, II: Let $T(n)$ be the running time in the Closest-Pair Algorithm when run on $n > 1$ points

$$T(n) \leq \begin{cases} 0 & \text{if } n = 1 \\ 2T(n/2) + O(n \log n) & \text{if } n > 1 \end{cases} \Rightarrow T(n) = O(n \log^2 n).$$

Q. Can we achieve $O(n \log n)$?

A. Yes. Don't sort points from scratch each time.

Sort by x at top level only.

Each recursive call returns δ and list of all points sorted by y

Sort by **merging** two pre-sorted lists.

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

Recurrences

Applications:

multiplying numbers

multiplying matrices

computing medians

Idea:

“Two halves are better than a whole”

if the base algorithm has super-linear complexity.

“If a little's good, then more's better”

repeat above, recursively

Applications: Many.

Binary Search, Merge Sort, (Quicksort), Closest points, Integer multiply,...

Recurrences

Above: Where they come
from, how to find them

Next: how to solve them

$T(n) = aT(n/b) + cn^d$ then

$a > b^d \Rightarrow T(n) = \Theta(n^{\log_b a})$ [many subprobs $\rightarrow$ leaves dominate]

$a < b^d \Rightarrow T(n) = \Theta(n^d)$ [few subprobs $\rightarrow$ top level dominates]

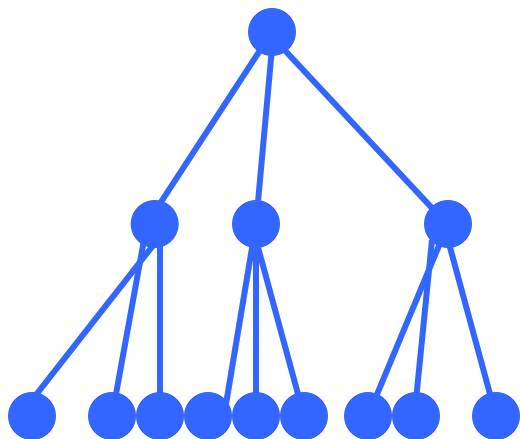
$a = b^d \Rightarrow T(n) = \Theta(n^d \log n)$ [balanced $\rightarrow$ all $\log n$ levels contribute]

Fine print:

$a \geq 1; b > 1; c, d \geq 0; T(1) = c;$

a, b, k, t integers.

Solve: $T(n) = a T(n/b) + cn^d$



Level	Num	Size	Work
0	$1 = a^0$	n	cn^d
1	a^1	n/b	$ac(n/b)^d$
2	a^2	n/b^2	$a^2c(n/b^2)^d$
...	...	...	...
i	a^i	n/b^i	$a^i c (n/b^i)^d$
...	...	...	...
k-1	a^{k-1}	n/b^{k-1}	$a^{k-1} c(n/b^{k-1})^d$
k	a^k	$n/b^k = 1$	$a^k T(1)$

$$n = b^k ; k = \log_b n$$

$$\text{Total Work:} = \sum_{i=0}^{\log_b n} a^i c(n/b^i)^d$$

(add last col) 

Theorem:

$$1 + x + x^2 + x^3 + \dots + x^k = (x^{k+1} - 1)/(x - 1)$$

proof:

$$S = 1 + x + x^2 + x^3 + \dots + x^k$$

$$xS = x + x^2 + x^3 + \dots + x^k + x^{k+1}$$

$$xS - S = x^{k+1} - 1$$

$$S(x - 1) = x^{k+1} - 1$$

$$S = (x^{k+1} - 1)/(x - 1)$$

$$T(1) = d$$

$$T(n) = a T(n/b) + cn^d, \quad a > b^d$$

$$T(n) = \sum_{i=0}^{\log_b n} a^i c(n/b^i)^d$$

$$= cn^d \sum_{i=0}^{\log_b n} (a/b^d)^i$$

$$= cn^d \frac{\left(\frac{a}{b^d}\right)^{\log_b n + 1} - 1}{\left(\frac{a}{b^d}\right) - 1}$$

$$\sum_{i=0}^k x^i =$$

$$\frac{x^{k+1} - 1}{x - 1}$$

$$(x \neq 1)$$

Solve: $T(1) = d$

$$T(n) = a T(n/b) + cn^d, \quad a > b^d$$

$$cn^d \frac{\left(\frac{a}{b^d}\right)^{\log_b n+1} - 1}{\left(\frac{a}{b^d}\right) - 1} < cn^d \frac{\left(\frac{a}{b^d}\right)^{\log_b n+1}}{\left(\frac{a}{b^d}\right) - 1}$$

$$= c \left(\frac{n^d}{b^{d \log_b n}} \right) \left(\frac{a}{b^d} \right) \frac{a^{\log_b n}}{\left(\frac{a}{b^d}\right) - 1}$$

$$= c \left(\frac{a}{b^d} \right) \frac{a^{\log_b n}}{\left(\frac{a}{b^d}\right) - 1}$$

$$= O(n^{\log_b a})$$

$$\begin{aligned} n^d &= \left(b^{\log_b n} \right)^d \\ &= b^{d \log_b n} \end{aligned}$$

$$\begin{aligned} a^{\log_b n} &= \left(b^{\log_b a} \right)^{\log_b n} \\ &= \left(b^{\log_b n} \right)^{\log_b a} \\ &= n^{\log_b a} \end{aligned}$$

Solve: $T(1) = d$

$$T(n) = a T(n/b) + cn^d, \quad a < b^d$$

$$T(n) = \sum_{i=0}^{\log_b n} a^i c(n/b^i)^d$$

$$= cn^d \sum_{i=0}^{\log_b n} a^i / b^{id}$$

$$= cn^d \frac{1 - \left(\frac{a}{b^d}\right)^{\log_b n + 1}}{1 - \left(\frac{a}{b^d}\right)}$$

$$< cn^d \frac{1}{1 - \left(\frac{a}{b^d}\right)}$$

$$= O(n^d)$$

$$\sum_{i=0}^k x^i =$$

$$\frac{x^{k+1} - 1}{x - 1}$$

$$(x \neq 1)$$

Solve: $T(1) = d$
 $T(n) = a T(n/b) + cn^d, a = b^d$

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log_b n} a^i c(n / b^i)^d \\ &= cn^d \sum_{i=0}^{\log_b n} a^i / b^{id} \\ &= O(n^d \log_b n) \end{aligned}$$

$T(n) = aT(n/b) + cn^d$ for $n > b$ then

$a > b^d \Rightarrow T(n) = \Theta(n^{\log_b a})$ [many subprobs $\rightarrow$ leaves dominate]

$a < b^d \Rightarrow T(n) = \Theta(n^d)$ [few subprobs $\rightarrow$ top level dominates]

$a = b^d \Rightarrow T(n) = \Theta(n^d \log n)$ [balanced $\rightarrow$ all $\log n$ levels contribute]

Fine print:

$a \geq 1; b > 1; c, d \geq 0; T(1) = c;$

a, b, k, t integers.

Integer Multiplication

Add. Given two n -bit integers a and b , compute $a + b$.

Add

	1	1	1	1	1	1	0	1	
		1	1	0	1	0	1	0	1
+	0	1	1	1	1	1	1	0	1
	1	0	1	0	1	0	0	1	0

$O(n)$ bit operations.

Add. Given two n -bit integers a and b , compute $a + b$.

Add

$O(n)$ bit operations.

Multiply. Given two n-bit integers a and b, compute $a \times b$.

The “grade school” method:

1	1	1	1	1	1	0	1	
	1	1	0	1	0	1	0	1
+	0	1	1	1	1	1	0	1
1	0	1	0	1	0	0	1	0

Multiply

[illegible]

Add. Given two n -bit integers a and b , compute $a + b$.

Add

$O(n)$ bit operations.

Multiply. Given two n-bit integers a and b, compute $a \times b$.

The “grade school” method:

$\Theta(n^2)$ bit operations.

1	1	1	1	1	1	0	1	
	1	1	0	1	0	1	0	1
+	0	1	1	1	1	1	0	1
1	0	1	0	1	0	0	1	0

Multiply

[illegible]

To multiply two 2-digit integers:

Multiply four 1-digit integers.

Add, shift some 2-digit integers to obtain result.

$$\begin{aligned}
 x &= 10 \cdot x_1 + x_0 \\
 y &= 10 \cdot y_1 + y_0 \\
 xy &= (10 \cdot x_1 + x_0)(10 \cdot y_1 + y_0) \\
 &= 100 \cdot x_1 y_1 + 10 \cdot (x_1 y_0 + x_0 y_1) + x_0 y_0
 \end{aligned}$$

Same idea works for *long* integers –
can split them into 4 half-sized ints

$$\begin{array}{r}
 \begin{array}{|c|c|} \hline 4 & 5 \\ \hline 3 & 2 \\ \hline \end{array} & \begin{array}{l} y_1 \ y_0 \\ x_1 \ x_0 \end{array} \\
 \hline
 \begin{array}{|c|c|} \hline 1 & 0 \\ \hline \end{array} & x_0 \times y_0 \\
 \begin{array}{|c|c|} \hline 0 & 8 \\ \hline \end{array} & x_0 \times y_1 \\
 \begin{array}{|c|c|} \hline 1 & 5 \\ \hline \end{array} & x_1 \times y_0 \\
 \begin{array}{|c|c|} \hline 1 & 2 \\ \hline \end{array} & x_1 \times y_1 \\
 \hline
 \begin{array}{|c|c|c|c|} \hline 1 & 4 & 4 & 0 \\ \hline \end{array}
 \end{array}$$

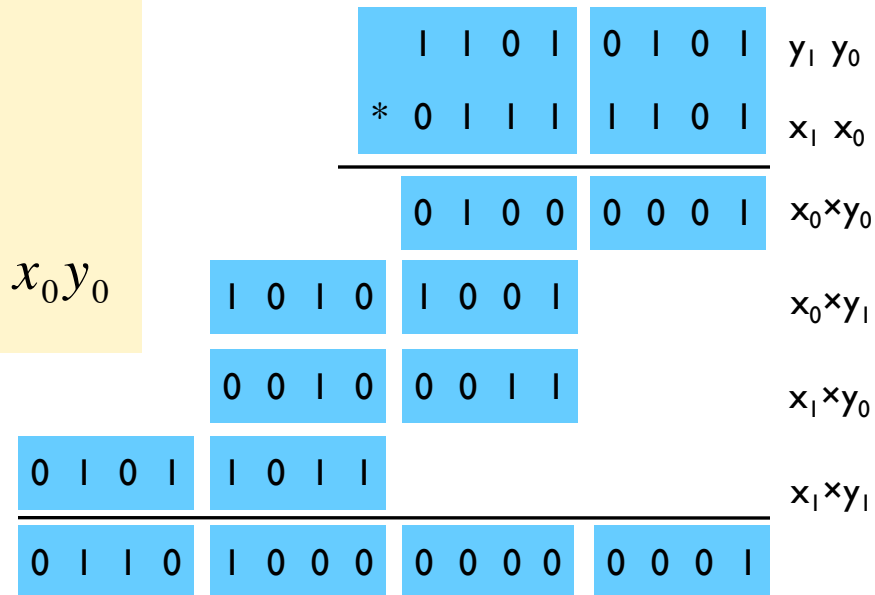
To multiply two n-bit integers:

Multiply four $\frac{1}{2}n$ -bit integers.

Add two $\frac{1}{2}n$ -bit integers, and shift to obtain result.

$$\begin{aligned} x &= 2^{n/2} \cdot x_1 + x_0 \\ y &= 2^{n/2} \cdot y_1 + y_0 \\ xy &= \left(2^{n/2} \cdot x_1 + x_0\right) \left(2^{n/2} \cdot y_1 + y_0\right) \\ &= 2^n \cdot x_1 y_1 + 2^{n/2} \cdot (x_1 y_0 + x_0 y_1) + x_0 y_0 \end{aligned}$$

$$T(n) = \underbrace{4T(n/2)}_{\text{recursive calls}} + \underbrace{\Theta(n)}_{\text{add, shift}}$$



To multiply two n-bit integers:

Multiply four $\frac{1}{2}n$ -bit integers.

Add two $\frac{1}{2}n$ -bit integers, and shift to obtain result.

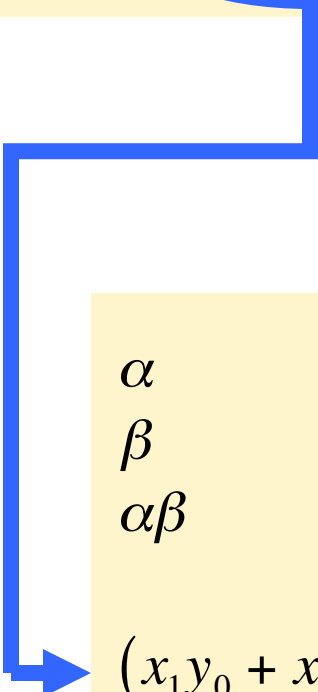
$$\begin{aligned} x &= 2^{n/2} \cdot x_1 + x_0 \\ y &= 2^{n/2} \cdot y_1 + y_0 \\ xy &= \left(2^{n/2} \cdot x_1 + x_0\right) \left(2^{n/2} \cdot y_1 + y_0\right) \\ &= 2^n \cdot x_1 y_1 + 2^{n/2} \cdot (x_1 y_0 + x_0 y_1) + x_0 y_0 \end{aligned}$$

$$T(n) = \underbrace{4T(n/2)}_{\text{recursive calls}} + \underbrace{\Theta(n)}_{\text{add, shift}} \Rightarrow T(n) = \Theta(n^2)$$

[illegible]

$$\begin{aligned}x &= 2^{n/2} \cdot x_1 + x_0 \\y &= 2^{n/2} \cdot y_1 + y_0 \\xy &= (2^{n/2} \cdot x_1 + x_0)(2^{n/2} \cdot y_1 + y_0) \\&= 2^n \cdot x_1 y_1 + 2^{n/2} \cdot (x_1 y_0 + x_0 y_1) + x_0 y_0\end{aligned}$$

Well, ok, 4 for 3 is more accurate...


$$\begin{aligned}\alpha &= x_1 + x_0 \\ \beta &= y_1 + y_0 \\ \alpha\beta &= (x_1 + x_0)(y_1 + y_0) \\ &= x_1 y_1 + (x_1 y_0 + x_0 y_1) + x_0 y_0 \\ (x_1 y_0 + x_0 y_1) &= \alpha\beta - x_1 y_1 - x_0 y_0\end{aligned}$$

To multiply two n-bit integers:

Add two $\frac{1}{2}n$ bit integers.

Multiply **three** $\frac{1}{2}n$ -bit integers.

Add, subtract, and shift $\frac{1}{2}n$ -bit integers to obtain result.

$$\begin{aligned}x &= 2^{n/2} \cdot x_1 + x_0 \\y &= 2^{n/2} \cdot y_1 + y_0 \\xy &= 2^n \cdot x_1 y_1 + 2^{n/2} \cdot (x_1 y_0 + x_0 y_1) + x_0 y_0 \\&= \underbrace{2^n \cdot x_1 y_1}_A + 2^{n/2} \cdot \underbrace{(x_1 + x_0)(y_1 + y_0)}_B - \underbrace{x_1 y_1}_A - \underbrace{x_0 y_0}_C + \underbrace{x_0 y_0}_C\end{aligned}$$

Theorem. **[Karatsuba-Ofman, 1962]** Can multiply two n-digit integers in $O(n^{1.585})$ bit operations.

$$\begin{aligned}T(n) &\leq 3T(n/2) + O(n) \\ \Rightarrow T(n) &= O(n^{\log_2 3}) = O(n^{1.585})\end{aligned}$$

Naïve: $\Theta(n^2)$

Karatsuba: $\Theta(n^{1.59\dots})$

Amusing exercise: generalize Karatsuba to do 5 size $n/3$ subproblems $\rightarrow \Theta(n^{1.46\dots})$

Best known: $\Theta(n \log n \log \log n)$

"Fast Fourier Transform"

Another Example:

Matrix Multiplication –

Strassen's Method

Multiplying Matrices

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} + a_{14}b_{42} & a_{11}b_{14} + a_{12}b_{24} + a_{13}b_{34} + a_{14}b_{44} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} + a_{24}b_{41} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} + a_{24}b_{42} & a_{21}b_{14} + a_{22}b_{24} + a_{23}b_{34} + a_{24}b_{44} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} + a_{34}b_{41} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} + a_{34}b_{42} & a_{31}b_{14} + a_{32}b_{24} + a_{33}b_{34} + a_{34}b_{44} \\ a_{41}b_{11} + a_{42}b_{21} + a_{43}b_{31} + a_{44}b_{41} & a_{41}b_{12} + a_{42}b_{22} + a_{43}b_{32} + a_{44}b_{42} & a_{41}b_{14} + a_{42}b_{24} + a_{43}b_{34} + a_{44}b_{44} \end{bmatrix}$$

n^3 multiplications, $n^3 - n^2$ additions

for i = 1 to n

 for j = 1 to n

 C[i,j] = 0

 for k = 1 to n

 C[i,j] = C[i,j] + A[i,k] * B[k,j]

n^3 multiplications, $n^3 - n^2$ additions

Multiplying Matrices

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} + a_{14}b_{42} & a_{11}b_{14} + a_{12}b_{24} + a_{13}b_{34} + a_{14}b_{44} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} + a_{24}b_{41} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} + a_{24}b_{42} & a_{21}b_{14} + a_{22}b_{24} + a_{23}b_{34} + a_{24}b_{44} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} + a_{34}b_{41} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} + a_{34}b_{42} & a_{31}b_{14} + a_{32}b_{24} + a_{33}b_{34} + a_{34}b_{44} \\ a_{41}b_{11} + a_{42}b_{21} + a_{43}b_{31} + a_{44}b_{41} & a_{41}b_{12} + a_{42}b_{22} + a_{43}b_{32} + a_{44}b_{42} & a_{41}b_{14} + a_{42}b_{24} + a_{43}b_{34} + a_{44}b_{44} \end{bmatrix}$$

Multiplying Matrices

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} + a_{14}b_{42} & a_{11}b_{14} + a_{12}b_{24} + a_{13}b_{34} + a_{14}b_{44} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} + a_{24}b_{41} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} + a_{24}b_{42} & a_{21}b_{14} + a_{22}b_{24} + a_{23}b_{34} + a_{24}b_{44} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} + a_{34}b_{41} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} + a_{34}b_{42} & a_{31}b_{14} + a_{32}b_{24} + a_{33}b_{34} + a_{34}b_{44} \\ a_{41}b_{11} + a_{42}b_{21} + a_{43}b_{31} + a_{44}b_{41} & a_{41}b_{12} + a_{42}b_{22} + a_{43}b_{32} + a_{44}b_{42} & a_{41}b_{14} + a_{42}b_{24} + a_{43}b_{34} + a_{44}b_{44} \end{bmatrix}$$

Multiplying Matrices

$$\begin{bmatrix} \overset{\text{A}_{11}}{a_{11}} & \overset{\text{A}_{12}}{a_{12}} & \overset{\text{A}_{13}}{a_{13}} & \overset{\text{A}_{14}}{a_{14}} \\ \overset{\text{A}_{21}}{a_{21}} & \overset{\text{A}_{22}}{a_{22}} & \overset{\text{A}_{23}}{a_{23}} & \overset{\text{A}_{24}}{a_{24}} \\ \overset{\text{A}_{31}}{a_{31}} & \overset{\text{A}_{32}}{a_{32}} & \overset{\text{A}_{33}}{a_{33}} & \overset{\text{A}_{34}}{a_{34}} \\ \overset{\text{A}_{41}}{a_{41}} & \overset{\text{A}_{42}}{a_{42}} & \overset{\text{A}_{43}}{a_{43}} & \overset{\text{A}_{44}}{a_{44}} \end{bmatrix} \cdot \begin{bmatrix} \overset{\text{B}_{11}}{b_{11}} & \overset{\text{B}_{12}}{b_{12}} & \overset{\text{B}_{13}}{b_{13}} & \overset{\text{B}_{14}}{b_{14}} \\ \overset{\text{B}_{21}}{b_{21}} & \overset{\text{B}_{22}}{b_{22}} & \overset{\text{B}_{23}}{b_{23}} & \overset{\text{B}_{24}}{b_{24}} \\ \overset{\text{B}_{31}}{b_{31}} & \overset{\text{B}_{32}}{b_{32}} & \overset{\text{B}_{33}}{b_{33}} & \overset{\text{B}_{34}}{b_{34}} \\ \overset{\text{B}_{41}}{b_{41}} & \overset{\text{B}_{42}}{b_{42}} & \overset{\text{B}_{43}}{b_{43}} & \overset{\text{B}_{44}}{b_{44}} \end{bmatrix}$$

$$= \begin{bmatrix} \overset{\text{A}_{11}\text{B}_{11}+\text{A}_{12}\text{B}_{21}}{a_{11}b_{11}+a_{12}b_{21}+a_{13}b_{31}+a_{14}b_{41}} & \overset{\text{A}_{11}\text{B}_{12}+\text{A}_{12}\text{B}_{22}}{a_{11}b_{12}+a_{12}b_{22}+a_{13}b_{32}+a_{14}b_{42}} & \circ & \overset{\text{A}_{11}\text{B}_{14}+\text{A}_{12}\text{B}_{24}}{a_{11}b_{14}+a_{12}b_{24}+a_{13}b_{34}+a_{14}b_{44}} \\ \overset{\text{A}_{21}\text{B}_{11}+\text{A}_{22}\text{B}_{21}}{a_{21}b_{11}+a_{22}b_{21}+a_{23}b_{31}+a_{24}b_{41}} & \overset{\text{A}_{21}\text{B}_{12}+\text{A}_{22}\text{B}_{22}}{a_{21}b_{12}+a_{22}b_{22}+a_{23}b_{32}+a_{24}b_{42}} & \circ & \overset{\text{A}_{21}\text{B}_{14}+\text{A}_{22}\text{B}_{24}}{a_{21}b_{14}+a_{22}b_{24}+a_{23}b_{34}+a_{24}b_{44}} \\ \overset{\text{A}_{31}\text{B}_{11}+\text{A}_{32}\text{B}_{21}}{a_{31}b_{11}+a_{32}b_{21}+a_{33}b_{31}+a_{34}b_{41}} & \overset{\text{A}_{31}\text{B}_{12}+\text{A}_{32}\text{B}_{22}}{a_{31}b_{12}+a_{32}b_{22}+a_{33}b_{32}+a_{34}b_{42}} & \circ & \overset{\text{A}_{31}\text{B}_{14}+\text{A}_{32}\text{B}_{24}}{a_{31}b_{14}+a_{32}b_{24}+a_{33}b_{34}+a_{34}b_{44}} \\ \overset{\text{A}_{41}\text{B}_{11}+\text{A}_{42}\text{B}_{21}}{a_{41}b_{11}+a_{42}b_{21}+a_{43}b_{31}+a_{44}b_{41}} & \overset{\text{A}_{41}\text{B}_{12}+\text{A}_{42}\text{B}_{22}}{a_{41}b_{12}+a_{42}b_{22}+a_{43}b_{32}+a_{44}b_{42}} & \circ & \overset{\text{A}_{41}\text{B}_{14}+\text{A}_{42}\text{B}_{24}}{a_{41}b_{14}+a_{42}b_{24}+a_{43}b_{34}+a_{44}b_{44}} \end{bmatrix}$$

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11}+A_{12}B_{21} & A_{11}B_{12}+A_{12}B_{22} \\ A_{21}B_{11}+A_{22}B_{21} & A_{21}B_{12}+A_{22}B_{22} \end{pmatrix}$$

Counting arithmetic operations:

$$T(n) = 8T(n/2) + 4(n/2)^2 = 8T(n/2) + n^2$$

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 8T(n/2) + n^2 & \text{if } n > 1 \end{cases}$$

By Master Recurrence, if

$T(n) = aT(n/b) + cn^d$ & $a > b^d$ then

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 8}) = \Theta(n^3)$$

$$P_1 = A_{12}(B_{11} + B_{21})$$

$$P_3 = (A_{11} - A_{12})B_{11}$$

$$P_5 = (A_{22} - A_{12})(B_{21} - B_{22})$$

$$P_6 = (A_{11} - A_{21})(B_{12} - B_{11})$$

$$P_7 = (A_{21} - A_{12})(B_{11} + B_{22})$$

$$C_{11} = P_1 + P_3$$

$$C_{21} = P_1 + P_4 + P_5 + P_7$$

$$P_2 = A_{21}(B_{12} + B_{22})$$

$$P_4 = (A_{22} - A_{21})B_{22}$$

$$C_{12} = P_2 + P_3 + P_6 - P_7$$

$$C_{22} = P_2 + P_4$$

Strassen's algorithm

Multiply 2×2 matrices using **7** instead of **8** multiplications
(and lots more than 4 additions)

$$T(n) = 7 T(n/2) + cn^2$$

$7 > 2^2$ so $T(n)$ is $\Theta(n^{\log_2 7})$ which is $O(n^{2.81})$

Fastest algorithms use $O(n^{2.376})$ time