Design and Analysis of Algorithm

M. Sakthi Balan

Department of Computer Science & Engineering
LNMIIT, Jaipur

Contents

Multiplication of integers

Problem

Suppose you are given two numbers of n-bit. What is the complexity of the multiplication operation?

Multiplication of integers

Complexity is $\mathcal{O}(n^2)$

Can we beat that complexity?

Yes! Use divide and conquer!

Multiplication of integers

Complexity is $\mathcal{O}(n^2)$

Can we beat that complexity?

Yes! Use divide and conquer!

Multiplication of integers

Complexity is $\mathcal{O}(n^2)$

Can we beat that complexity?

Yes! Use divide and conquer!

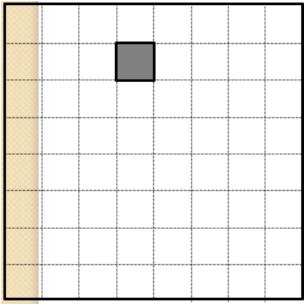
Multiplication of integers

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

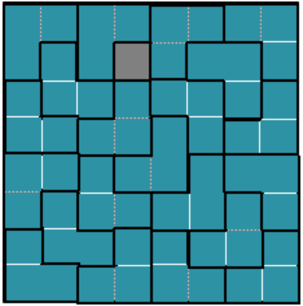
Tromino Tiling

A tromino:

A $2^n \times 2^n$ board with a hole:

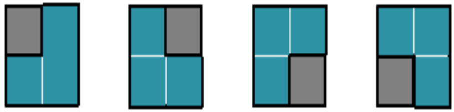


After tiling:



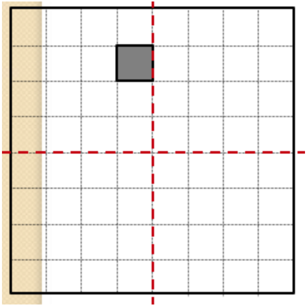
Tromino Tiling

Tiling a 2×2 board is trivial:



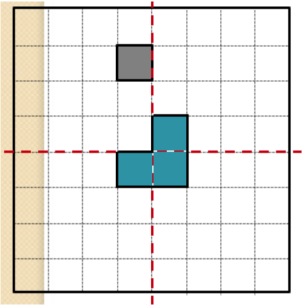
Can we reduce a general $2^n \times 2^n$ into 2×2 board that we know how to solve?

Tromino Tiling



- Can we reduce the problem size?
- Yes how about reducing to $2^{n-1} \times 2^{n-1}$?
- But only one square (out of 4) will have a hole!

Tromino Tiling



- Insert one tromino at the center such that it covers one square of all the 3 squares that do not have a hole!
- Now we can consider all 4 squares having a hole in it!
- More importantly we can do this recursively until you get 2×2 that can be tiled easily!

Tromino Tiling

Algorithm 2.1: $\textsc{Tile}(n, L)$

if $n = 1$

then $\begin{cases} \textit{Trivial case} \\ \textit{Tile with one tromino} \\ \textit{return} \end{cases}$

Divide the board into four equal-sized boards

Place one tromino at the center to cut out 3 additional holes

Let $L1$, $L2$, $L3$, $L4$ *denote the positions of the 4 holes*

$\textsc{Tile}(n - 1, L1)$

$\textsc{Tile}(n - 1, L2)$

$\textsc{Tile}(n - 1, L3)$

$\textsc{Tile}(n - 1, L4)$

Matrix Multiplication

Input: $A = [a_{ij}], B = [b_{ij}].$

Output: $C = [c_{ij}] = A \cdot B.$

$i, j = 1, 2, \ldots, n.$

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

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

Matrix Multiplication

Let $A = (a_{ij})$ and $B = (b_{ij})$ be two matrices to be multiplied. Multiplication algorithm is given as follows:

```
SQUARE-MATRIX-MULTIPLY(A, B)
n = A.rows
let C be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        c_ij = 0
        for k = 1 to n
            c_ij = c_ij + a_ik · b_kj
return C
```

From CLRS

Matrix Multiplication

What is the complexity of Matrix multiplications?

Divide and Conquer will it help to reduce this further?

Matrix Multiplication

What is the complexity of Matrix multiplications?

Divide and Conquer will it help to reduce this further?

Matrix Multiplication

Using Divide and Conquer:

Suppose that we partition each of A, B, and C into four $n/2 \times n/2$ matrices

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}, \tag{4.9}$$

so that we rewrite the equation $C = A \cdot B$ as

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}. \tag{4.10}$$

Equation (4.10) corresponds to the four equations

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}, \tag{4.11}$$

$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}, \tag{4.12}$$

$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}, \tag{4.13}$$

$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}. \tag{4.14}$$

From CLRS

Matrix Multiplication

What is the complexity of the divide and conquer technique?

It is $\Theta(n^3)$. Not helping us!

Strassen's Matrix Multiplication Method that runs in $\Theta(n^{2.81})$

Matrix Multiplication

What is the complexity of the divide and conquer technique?

It is $\Theta(n^3)$. Not helping us!

Strassen's Matrix Multiplication Method that runs in $\Theta(n^{2.81})$

Matrix Multiplication

What is the complexity of the divide and conquer technique?

It is $\Theta(n^3)$. Not helping us!

Strassen's Matrix Multiplication Method that runs in $\Theta(n^{2.81})$

Matrix Multiplication

Using Divide and Conquer:

```
SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)
n = A.rows
let C be a new n × n matrix
if n == 1
    c11 = a11 · b11
else partition A, B, and C as in equations (4.9)
    C11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B21)
    C12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B22)
    C21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B21)
    C22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B22)
return C
```

From CLRS

Matrix Multiplication

Using Divide and Conquer:

$$
\begin{aligned}
S_1 &= B_{12} - B_{22} \,, \\
S_2 &= A_{11} + A_{12} \,, \\
S_3 &= A_{21} + A_{22} \,, \\
S_4 &= B_{21} - B_{11} \,, \\
S_5 &= A_{11} + A_{22} \,, \\
S_6 &= B_{11} + B_{22} \,, \\
S_7 &= A_{12} - A_{22} \,, \\
S_8 &= B_{21} + B_{22} \,, \\
S_9 &= A_{11} - A_{21} \,, \\
S_{10} &= B_{11} + B_{12} \,.
\end{aligned}
$$

From CLRS

Matrix Multiplication

Using Divide and Conquer:

$$
\begin{aligned}
P_1 &= A_{11} \cdot S_1 = A_{11} \cdot B_{12} - A_{11} \cdot B_{22}, \\
P_2 &= S_2 \cdot B_{22} = A_{11} \cdot B_{22} + A_{12} \cdot B_{22}, \\
P_3 &= S_3 \cdot B_{11} = A_{21} \cdot B_{11} + A_{22} \cdot B_{11}, \\
P_4 &= A_{22} \cdot S_4 = A_{22} \cdot B_{21} - A_{22} \cdot B_{11}, \\
P_5 &= S_5 \cdot S_6 = A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22}, \\
P_6 &= S_7 \cdot S_8 = A_{12} \cdot B_{21} + A_{12} \cdot B_{22} - A_{22} \cdot B_{21} - A_{22} \cdot B_{22}, \\
P_7 &= S_9 \cdot S_{10} = A_{11} \cdot B_{11} + A_{11} \cdot B_{12} - A_{21} \cdot B_{11} - A_{21} \cdot B_{12}.
\end{aligned}
$$

From CLRS

Matrix Multiplication

Using Divide and Conquer:

$$
\begin{aligned}
C_{11} &= P_5 + P_4 - P_2 + P_6 \\
C_{12} &= P_1 + P_2 \\
C_{21} &= P_3 + P_4 \\
C_{22} &= P_5 + P_1 - P_3 - P_7
\end{aligned}
$$

Counting Inversions

- Movie streaming site ranks your preference of movies
- It recommends people with similar tastes
- How it is done!?

Counting Inversions

Similarity Metric

- My rank for movies: $1, 2, \ldots, n$
- Rank of someone: $a_1, a_2, \ldots, n$
- Movies i and j are said to be inverted if $i < j$ and $a_i > a_j$

	A	B	C	D	E
My rank	1	2	3	4	5
Other	1	3	4	2	5

There are two inversions: $(3, 2)$ and $(4, 2)$

What is the complexity of brute force algorithm?
Can we beat it?

Counting Inversions

Similarity Metric

- My rank for movies: $1, 2, \ldots, n$
- Rank of someone: $a_1, a_2, \ldots, n$
- Movies i and j are said to be inverted if $i < j$ and $a_i > a_j$

	A	B	C	D	E
My rank	1	2	3	4	5
Other	1	3	4	2	5

There are two inversions: $(3, 2)$ and $(4, 2)$

What is the complexity of brute force algorithm?
Can we beat it?

Counting Inversions

Similarity Metric

- My rank for movies: $1, 2, \ldots, n$
- Rank of someone: $a_1, a_2, \ldots, n$
- Movies i and j are said to be inverted if $i < j$ and $a_i > a_j$

	A	B	C	D	E
My rank	1	2	3	4	5
Other	1	3	4	2	5

There are two inversions: $(3, 2)$ and $(4, 2)$

What is the complexity of brute force algorithm?
Can we beat it?

Counting Inversions

Divide and Conquer:

Divide:	separate list into two halves A and B
Conquer:	recursively count inversions in each list
Combine:	count inversions (a, b) with $a \in A$ and $b \in B$
Output:	Return sum of three counts

Counting Inversions

input

1	5	4	8	10	2	6	9	3	7

count inversions in left half A

1	5	4	8	10

5-4

count inversions in right half B

2	6	9	3	7

6-3 9-3 9-7

count inversions (a, b) with a ∈ A and b ∈ B

1	5	4	8	10

2	6	9	3	7

4-2 4-3 5-2 5-3 8-2 8-3 8-6 8-7 10-2 10-3 10-6 10-7 10-9

output 1 + 3 + 13 = 17

From Kleinberg and Eva Tardos

Counting Inversions

How to count the inversions (a, b) when $a \in A$ and $b \in B$?

Get A and B in sorted form!

- Sort A and B
- For each element $b \in B$, find how elements in A are greater than b

LNMIIT

Counting Inversions

How to count the inversions (a, b) when $a \in A$ and $b \in B$?

Get A and B in sorted form!

- Sort A and B
- For each element $b \in B$, find how elements in A are greater than b

Counting Inversions

How to count the inversions (a, b) when $a \in A$ and $b \in B$?

Get A and B in sorted form!

- Sort A and B
- For each element $b \in B$, find how elements in A are greater than b

LNMIIT

Counting Inversions

list A

7	10	18	3	14

list B

17	23	2	11	16

sort A

3	7	10	14	18

sort B

2	11	16	17	23

binary search to count inversions (a, b) with a ∈ A and b ∈ B

3	7	10	14	18

2	11	16	17	23
5	2	1	1	0

From Kleinberg and Eva Tardos

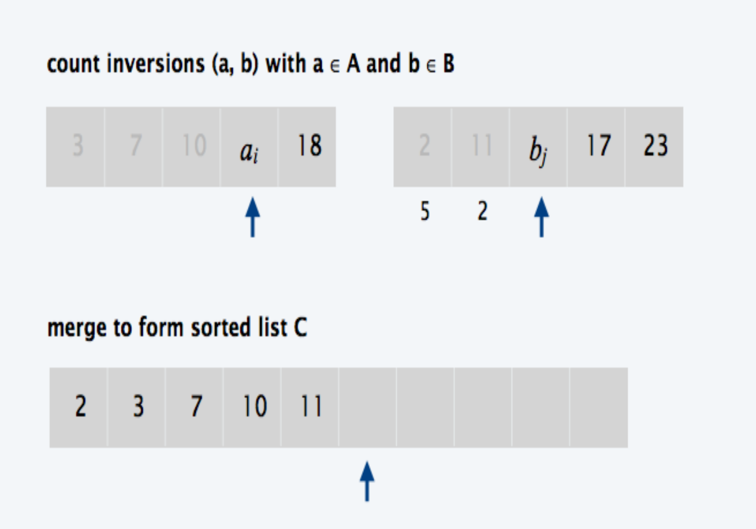
Counting Inversions

Count inversions (a, b) with $a \in A$ and $b \in B$, assuming A and B are sorted.

- Scan A and B from left to right.
- Compare a_i and b_j.
- If $a_i < b_j$, then a_i is not inverted with any element left in B.
- If $a_i > b_j$, then b_j is inverted with every element left in A.
- Append smaller element to sorted list C.

From Kleinberg and Eva Tardos

Counting Inversions



From Kleinberg and Eva Tardos

Counting Inversions

SORT-AND-COUNT (L)

IF list L has one element

RETURN $(0, L)$.

DIVIDE the list into two halves A and B.

$(r_A, A) \leftarrow$ SORT-AND-COUNT(A).

$(r_B, B) \leftarrow$ SORT-AND-COUNT(B).

$(r_{AB}, L') \leftarrow$ MERGE-AND-COUNT(A, B).

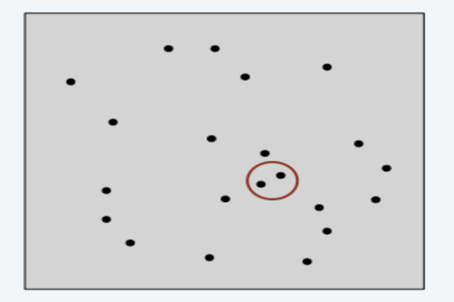
RETURN $(r_A + r_B + r_{AB},\ L')$.

From Kleinberg and Eva Tardos

Closest Pair of Points

Problem

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



From Kleinberg and Eva Tardos

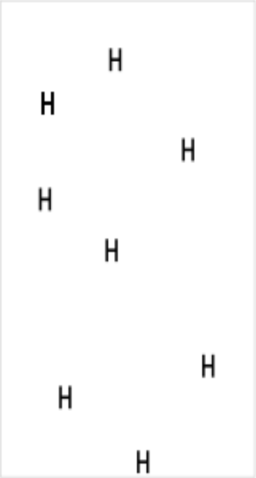
Closest Pair of Points

Applications

Graphics, computer vision, geographic information systems, molecular modeling, air traffic control, special case of nearest neighbor, Euclidean MST, Voronoi.

Applications: Voronoi diagram

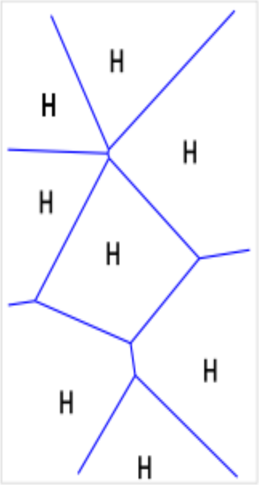
Given ambulance posts in a country, in case of an emergency somewhere, where should the ambulance come from?



LNMIIT

Applications: Voronoi diagram

Given ambulance posts in a country, in case of an emergency somewhere, where should the ambulance come from?



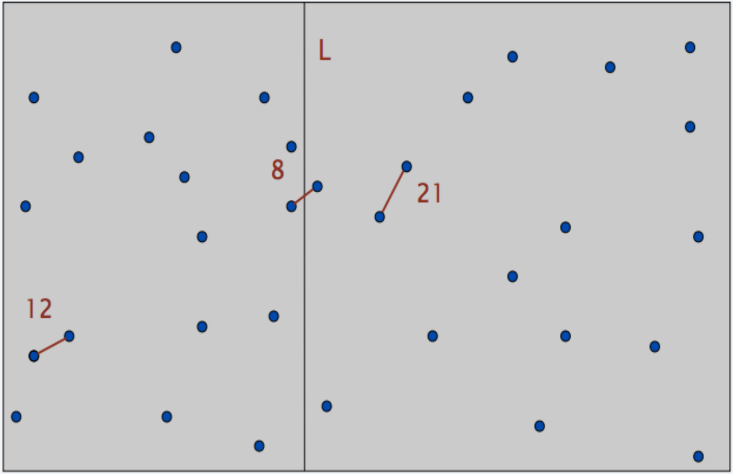
Closest Pair of Points

- How about 1-dimension? Can we solve it? What will be the complexity?
- Why 2-D is different?
- What is the brute force complexity for 2-D?
- Can we get a better algorithm to solve this problem?
- Yes! Divide and Conquer it is...
- Algorithm given by Shamos in 1975

Closest Pair of Points

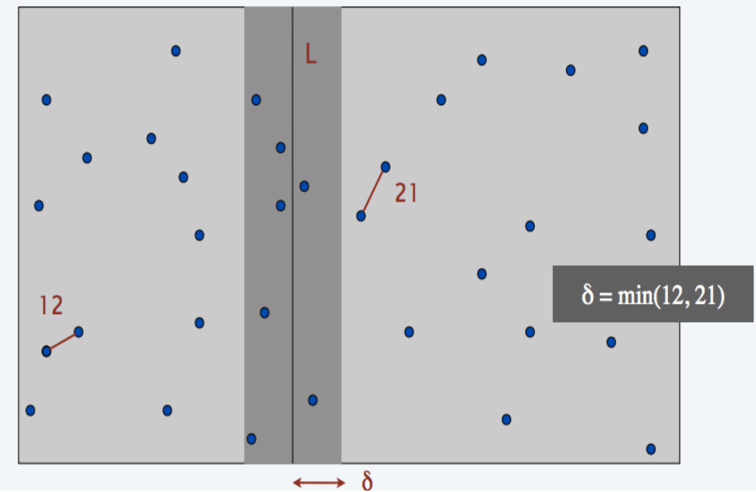
Divide	How are we going to divide the problem?
Conquer	Once we divide conquering is easy
Combine	Combine is not straight forward!

Closest Pair of Points



From Kleinberg and Eva Tardos

Closest Pair of Points



From Kleinberg and Eva Tardos

Closest Pair of Points

- How to find the closest pair when the points occur in different regions?
- That's where Mathematical reasoning helps us!
- δ-strip around the divider line is sufficient to check for those points!

Closest Pair of Points

- How to find the closest pair when the points occur in different regions?
- That's where Mathematical reasoning helps us!
- δ-strip around the divider line is sufficient to check for those points!

Closest Pair of Points

Theorem

Suppose there are two points p and q where p lies in the left region L and q lies in the right region R such that $d(p, q) < \delta$ then each p and q lies within a distance of δ of the divider line.

Proof

Let $p = (p_x, p_y)$ and $q = (q_x, q_y)$ be those points. Let L be the divider line represented as $x = x^*$. By assumption $p_x \leq x^* \leq q_x$.

$$x^* - p_x \leq q_x - p_x \leq \delta(p, q) < \delta$$

and

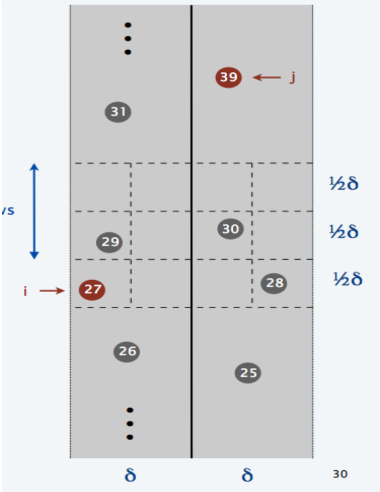
$$q_x - x^* \leq q_x - p_x \leq \delta(p, q) < \delta$$

Closest Pair of Points

- How to choose those points in the 2δ-strip?
- Also what if all the points lie in that strip? We may not gain anything! (Will be $\mathcal{O}(n^2)$!)
- Again logical reasoning helps us....

Closest Pair of Points

- Divide the regions into boxes of size $\delta/2 \times \delta/2$
- Each square cannot hold more than one point
- Now for a point in L there will be only finite number of points!
- So even if all the points of L are there in the δ-strip we need to check for only finite number of points in R for each of those points in L! Hence it is $\mathcal{O}(n)$! In fact $\Theta(n)$!



Closest Pair of Points

CLOSEST-PAIR $(p_1, p_2, \ldots, p_n)$

Compute separation line L such that half the points are on each side of the line.	← $O(n \log n)$
$\delta_1 \leftarrow$ CLOSEST-PAIR (points in left half).	
$\delta_2 \leftarrow$ CLOSEST-PAIR (points in right half).	← $2T(n/2)$
$\delta \leftarrow \min\{\delta_1, \delta_2\}$.	
Delete all points further than δ from line L.	← $O(n)$
Sort remaining points by y-coordinate.	← $O(n \log n)$
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.	← $O(n)$
RETURN δ.	

LNMIIT

Closest Pair of Points

$$T(n) = 2.T(n/2) + \mathcal{O}(n \log n)$$

What is the complexity then?

$$T(n) = n \log^2 n$$

Can we reduce? Yes!
Sort the points with respect to y co-ordinate also that will give the recurrence equation as,

$$T(n) = 2.T(n/2) + \mathcal{O}(n)$$

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

LNMIIT

Closest Pair of Points

$$T(n) = 2.T(n/2) + \mathcal{O}(n \log n)$$

What is the complexity then?

$$T(n) = n \log^2 n$$

Can we reduce? Yes!
Sort the points with respect to y co-ordinate also that will give the recurrence equation as,

$$T(n) = 2.T(n/2) + \mathcal{O}(n)$$

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

Closest Pair of Points

```
Closest-Pair(P)
  Construct P_x and P_y  (O(n log n) time)
  (p*_0, p*_1) = Closest-Pair-Rec(P_x,P_y)

Closest-Pair-Rec(P_x, P_y)
  If |P| ≤ 3 then
    find closest pair by measuring all pairwise distances
  Endif

  Construct Q_x, Q_y, R_x, R_y (O(n) time)
  (q*_0,q*_1) = Closest-Pair-Rec(Q_x, Q_y)
  (r*_0,r*_1) = Closest-Pair-Rec(R_x, R_y)

  δ = min(d(q*_0,q*_1), d(r*_0,r*_1))
  x* = maximum x-coordinate of a point in set Q
  L = {(x,y) : x = x*}
  S = points in P within distance δ of L.

  Construct S_y (O(n) time)
  For each point s ∈ S_y, compute distance from s
     to each of next 15 points in S_y
     Let s, s' be pair achieving minimum of these distances
     (O(n) time)

  If d(s,s') < δ then
     Return (s,s')
  Else if d(q*_0,q*_1) < d(r*_0,r*_1) then
     Return (q*_0,q*_1)
Else
   Return (r*_0,r*_1)
Endif
```