COT 6405
ANALYSIS OF ALGORITHMS

Divide-and-Conquer

Computer & Electrical Engineering and Computer Science Department
Florida Atlantic University

Outline

- Divide-and-conquer method
- Analyzing RT
- Problems solved using divide-and-conquer
  - Binary Search
  - Finding the closest pair of points
  - Integer multiplication
  - Strassen’s matrix product

Divide-and-conquer method

- Recursive approach
- Three steps at each level of the recursion:
  - **Divide** the problem into a number of subproblems of smaller input size
  - **Conquer** the subproblems by solving them recursively.

    Base case: if the subproblem sizes are small enough, just solve them in a straightforward manner
  - **Combine** the solutions of the subproblems into a solution for the original problem

Analyzing Divide-and-Conquer

- Express the RT using a recurrence

$$\boxed{T(n) = a\,T(n/b) + f(n)}$$

$a \geq 1,\ b > 1$

  - **conquer step**: solve *a* subproblems, each of which is *1/b* the size of the original problem
  - **divide and combine steps** together take *f(n)* time

- Solve the recurrence using the Master Theorem

Binary Search

- *Given a sorted array A of n numbers, determine whether a given number x belongs to the array.*

General problem: search x into A[p..r]

- Divide the array into two halves q = $\lfloor (p+r)/2 \rfloor$
- Compare x with the middle element A[q]
  - If they have the same value, then return x's location
  - If x < A[q], then search x into A[p..q-1]
  - If x > A[q], then search x into A[q+1..r]

conquer

Example: Binary Search for x = 35

Input array A[1..14]

5	12	21	27	35	55	68	71	79	96	133	160	200	249

Compare x with 68

Since x < 68, we select the left half

5	12	21	27	35	55

Compare x with 21

Since x > 21, we select the right half

27	35	55

Compare x with 35

Since x = 35, the search is successful. The algorithm returns the index of x, which is 5.

```
BinarySearch(A,p,r,x)
if p > r then return not found
if p == r
  if x == A[p] then return p
  else return x not found
q = ⌊(p+r)/2⌋
if x == A[q] then return q
elseif x < A[q]
  BinarySearch(A,p,q-1,x)
else
  BinarySearch(A,q+1,r,x)
```

Base case

Divide

Conquer

Initial call: BinarySearch(A,1,n,x)

RT analysis

$T(n) = T(n/2) + \Theta(1)$

Case 2 of the Master Theorem:

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

Finding the closest pair of points

Reference: *Algorithm Design, by Jon Klainberg and Eva Tardos, Chapter 5.4*

Problem: given n points in the plane, find the pair that is closest together.

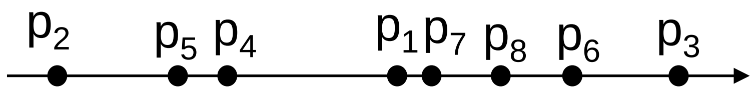
- considered by M. Shamos and D. Hoey in 1970s
- $O(n^2)$ solution – compute the distance between each pair of points and take the minimum
- $O(n\log n)$ solution – using divide-and conquer

Notations

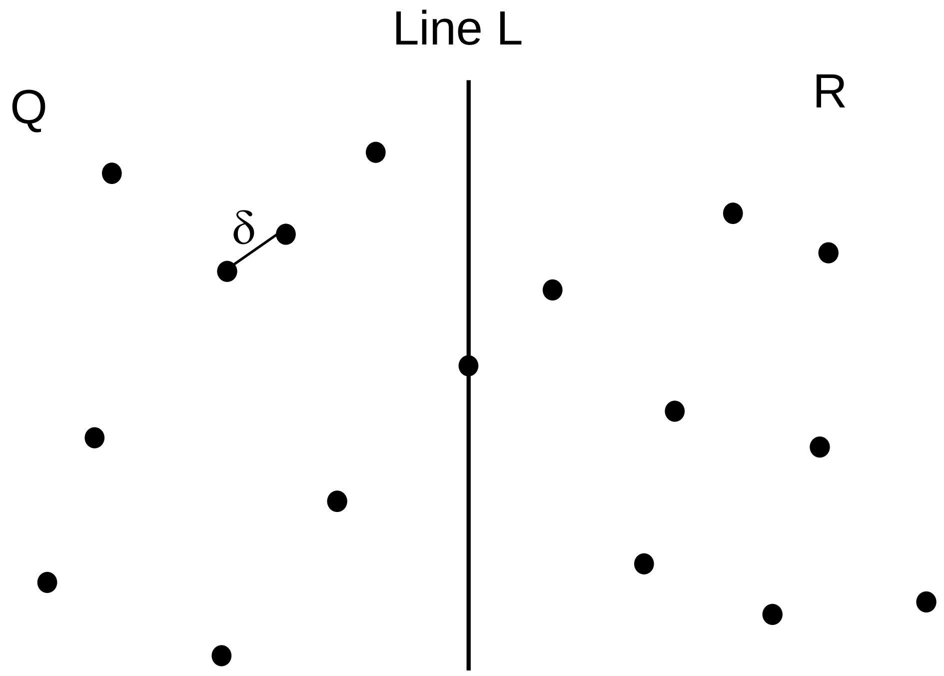
- set of points $P = \{p_1, p_2, \ldots, p_n\}$
- p_i has coordinates (x_i, y_i)
- $d(p_i, p_j)$ – Euclidean distance between p_i and p_j
- assume that no two points have the same x-coordinate or the same y-coordinate

One-dimensional version:

- closest pair of points on a line
- O(n logn) solution:
  - sort them in O(n logn) time
  - walk through the sorted list computing the distance between consecutive points

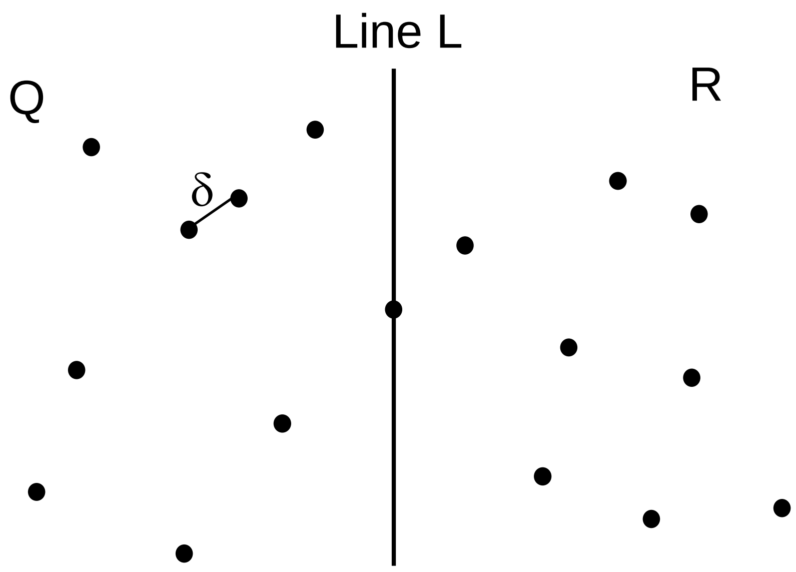


Divide-and-conquer approach



- **Divide**: the point set P is divided evenly into Q and R by the line L
- **Conquer**: recursively find the closest pair among the points in Q and among the points in R
- **Combine**: find the overall solution from subproblems. This step should take linear time O(n).

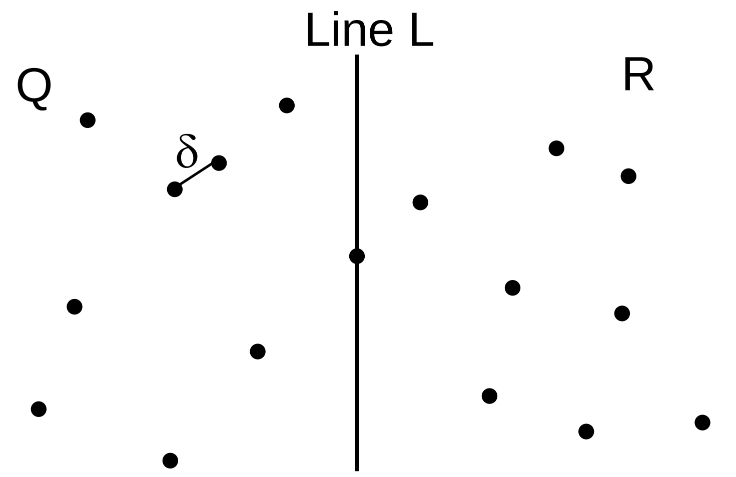
Algorithm details



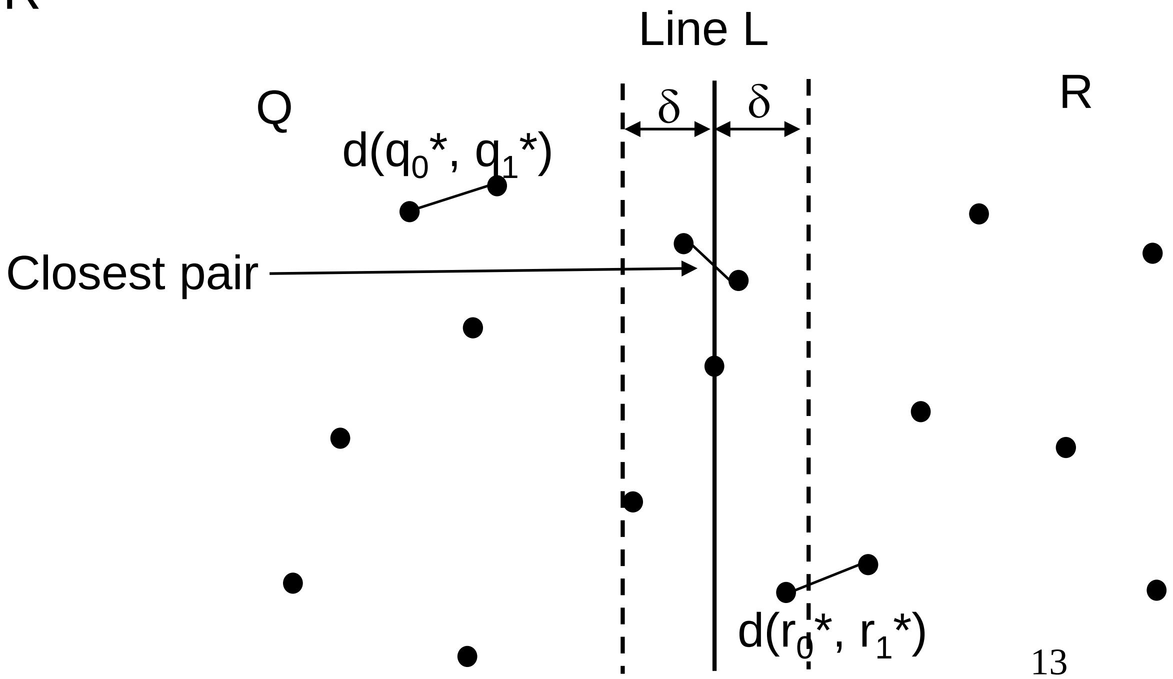
- For any set of points P
  - let P_x denotes the points sorted by increasing x – coordinate
  - let P_y denotes the points sorted by increasing y – coordinate
- First level of recursion:
  - Q is the "*left half*" of P – the first $\lceil n/2 \rceil$ points in P_x
  - R is the "*right half*" of P – the last $\lfloor n/2 \rfloor$ points in P_x
  - one pass through each of P_x and P_y in O(n) can create Q_x, Q_y, R_x, and R_y
    - Q_x,R_x – points in Q and R sorted in increasing x – coordinate
    - Q_y,R_y– points in Q and R sorted in increasing y – coordinate
  - recursively find the closest pair of points in Q and R
    - Let q_0^* and q_1^* be the closest pair of points in Q
    - Let r_0^* and r_1^* be a closest pair of points in R

Combine step

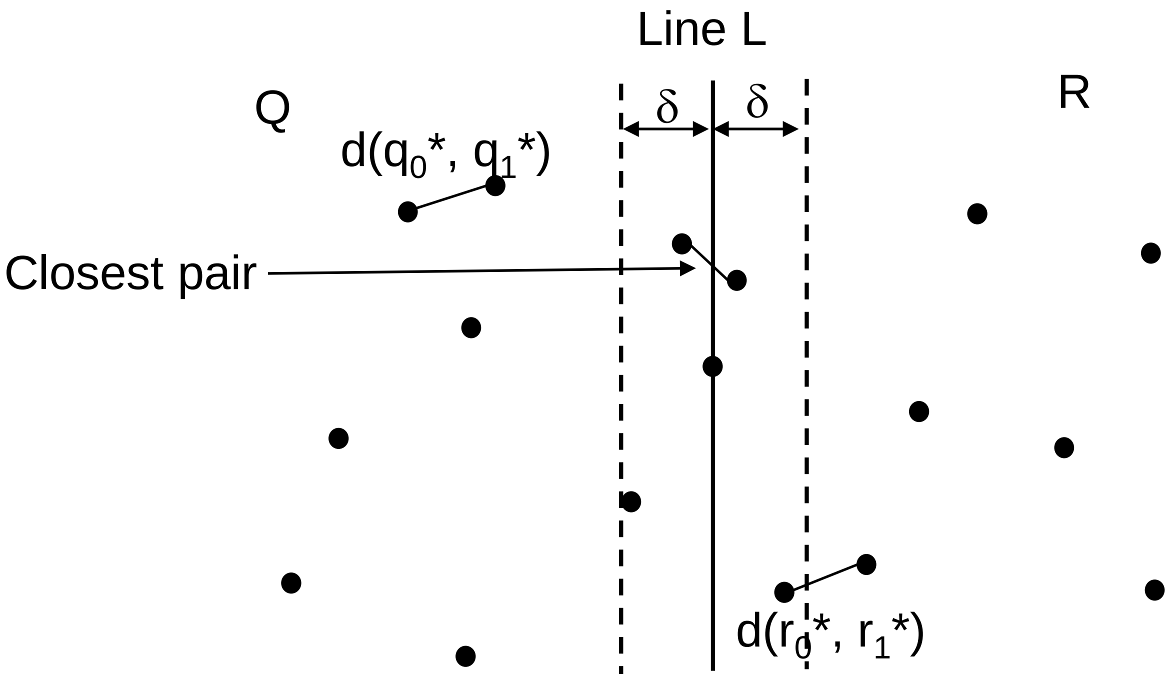
- Objective: linear time O(n)
- Let $\delta = \min\{d(q_0^*, q_1^*), d(r_0^*, r_1^*)\}$
- Are there $q \in Q$ and $r \in R$ such that $d(q,r) < \delta$?
- Notations:
  - x^* be the x-coordinate of the rightmost point in Q
  - L is the vertical line $x = x^*$ separating Q and R



Property: If there exist $q \in Q$ and $r \in R$ such that $d(q,r) < \delta$ then each of q and r lies within a distance δ of L.



Combine step



- let S be the points in P within distance δ of L
- observation:
    - S might be the whole P
    - checking all the pairs is $O(n^2) \Rightarrow$ too large !!!

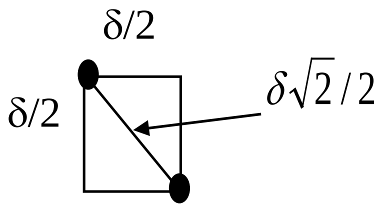
- Let S_y be the points in S sorted by increased y-coordinate
    - constructed by a single pass through $P_y \Rightarrow O(n)$

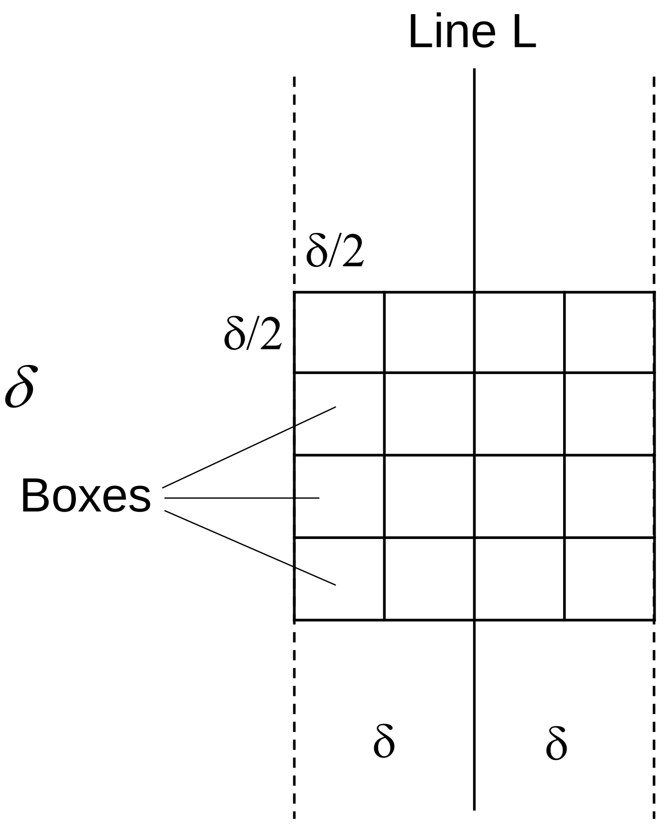
Property: If s, s'$\in$ S have the property that d(s,s') < δ, then s and s' are within 15 positions of each other in the sorted list S_y.

Combine step

Property: If $s, s' \in S$ have the property that $d(s,s') < \delta$, then s and s' are within 15 positions of each other in the sorted list S_y.

- let Z – plane containing all points within distance δ of L
- partition Z into a grid $\delta/2 \times \delta/2$
- **each box contains at most one point of S**
    - each pair in Q or in R has distance $\geq \delta$
    - points in one box belong either to Q or R
    - distance between any two points in a box is $\leq \delta\sqrt{2}/2 < \delta$





- two points $s, s' \in S$ which are at least 16 position apart in S_y have $d(s, s') > \delta$
    - separated by at least 3 rows $\Rightarrow$ distance $> 3\delta/2$

Combine step

- Note that the value of 15 can be reduced, but for our purpose it is important to be a constant

Combine step:

- make one pass through S_y
  - for each $s \in S_y$ compute its distance to the next 15 points in S_y
  - record the smallest distance

  - if the smallest distance is $< \delta$, then this is the closest pair in P
  - otherwise the pair (in Q or R) with dist = δ is the closest pair in P

- Combine step takes O(n)

Closest-Pair(P)

construct P_x and P_y // O(nlogn)

(p_0^*, p_1^*) = Closest-Pair-Rec(P_x, P_y)

Closest-Pair-Rec(P_x, P_y)

if $|P| \leq 3$ // Base Case

find the closest pair by measuring all pairwise distances

construct Q_x, Q_y, R_x, R_y // O(n) // Divide

(q_0^*, q_1^*) = Closest-Pair-Rec(Q_x, Q_y) // Conquer

(r_0^*, r_1^*) = Closest-Pair-Rec(R_x, R_y)

$\delta = \min(d(q_0^*, q_1^*), d(r_0^*, r_1^*))$ // Combine

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 $s \in S_y$ // O(n)

compute the distance from s to each of the next 15 points in S_y

let s, s' be the pair with the minimum distance

if $d(s, s') < \delta$

return (s, s')

else if $d(q_0^*, q_1^*) < d(r_0^*, r_1^*)$

return (q_0^*, q_1^*)

else

return (r_0^*, r_1^*)

RT Analysis

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

Case 2 of the Master Theorem $\Rightarrow T(n) = \Theta(n\lg n)$

Integer Multiplication

Reference: *Algorithm Design, by Jon Klainberg and Eva Tardos, Chapter 5.5*

Problem: multiplication of two n-digit numbers x and y.

$$
\begin{array}{r}
12 \\
\times\ 13 \\
\hline
36 \\
12 \\
\hline
156
\end{array}
\qquad\qquad
\begin{array}{r}
1100 \\
\times\ 1101 \\
\hline
1100 \\
0000 \\
1100 \\
1100 \\
\hline
10011100
\end{array}
$$

Standard solution in $O(n^2)$:

- compute partial products by multiplying each digit of y by x
- add up all the partial products
- n partial products; takes $O(n)$ to compute each partial product $\Rightarrow O(n^2)$

Divide-and-Conquer algorithm

- Assume numbers are in base-2 (it doesn't matter)

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

x_1 is the high-order n/2 bits; x_0 is the low-order n/2 bits
similar for y_1, y_0

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

- Four subproblems: x_1y_1, x_1y_0, x_0y_1, x_0y_0
- Combining the solutions of the subproblems takes O(n)

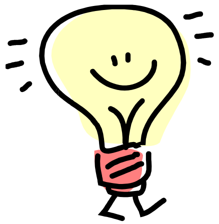
$$\mathbf{T(n) = 4T(n/2) + cn}$$

Divide-and-Conquer algorithm

$T(n) = 4T(n/2) + cn$

Case 1 of Master Thm $\Rightarrow T(n) = O(n^2)$

- No improvement in the RT !



Idea: use only 3 subproblems !

$$(x_1+x_0)(y_1+y_0) = x_1y_1+\underline{x_1y_0+x_0y_1}+x_0y_0$$

$$xy = x_1y_12^n + \underline{(x_1y_0+x_0y_1)}2^{n/2}+x_0y_0$$

Divide-and-Conquer algorithm

$xy = x_1y_12^n + \underline{(x_1y_0+x_0y_1)}2^{n/2}+x_0y_0$

$(x_1+x_0)(y_1+y_0) = x_1y_1+\underline{x_1y_0+x_0y_1}+x_0y_0$

Recursive-Multiply(x, y)

write $x = x_12^{n/2} + x_0$ // Divide

$y = y_12^{n/2} + y_0$

compute $x_1 + x_0$ and $y_1 + y_0$

p = Recursive-Multiply $(x_1 + x_0, y_1 + y_0)$ // Conquer

x_1y_1 = Recursive-Multiply (x_1, y_1)

x_0y_0 = Recursive-Multiply (x_0, y_0)

return $x_1y_12^n + (p - x_1y_1 - x_0y_0)2^{n/2} + x_0y_0$ // Combine

RT analysis:

$T(n) = 3\,T(n/2) + cn$

Case 1 of the Master Thm $\Rightarrow$ $T(n) = \Theta(n^{\log_2^3}) = \Theta(n^{1.59})$

Integer Multiplication, example

$x = 1100$

$y = 1101 \qquad n = 4$

Divide

$x_1 = 11, \quad x_0 = 00$

$y_1 = 11, \quad y_0 = 01$

$x_1 + x_0 = 11$

$y_1 + y_0 = 100$

Conquer

$p = (x_1 + x_0)(y_1 + y_0) = 11 \cdot 100 = 1100$

$x_0 y_0 = 00 \cdot 01 = 0$

$x_1 y_1 = 11 \cdot 11 = 1001$

Combine

$xy = x_1 y_1 2^n + (p - x_0 y_0 - x_1 y_1) 2^{n/2} + x_0 y_0 = 10010000 + (1100 - 0 - 1001)\, 2^2 + 0 = 10011100$

Strassen's Matrix Product

Reference: *Algorithms, by* Richard Johnsonbaugh and Marcus Schaefer, *Chapter 5.4*

Problem: multiplication of two matrices A and B.

- A_{ij} – element row i, column j
- matrix product C = AB requires A and B to be compatible
  - if A is $m \times p$ and B is $p \times n$
  - then C is $m \times n$

Matrix Product

Input parameters: matrices A, B of size $n \times n$
Output parameter: matrix C of size $n \times n$

C = AB

$$C_{ij} = \sum_{k=1}^{p} A_{ik} B_{kj}$$

<u>MatrixProduct(A, B, C)</u>
n = A.last
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]

RT = $\Theta(n^3)$

Can we do better?

Divide-and-Conquer algorithm

- Assume A and B have size n × n, where n is a power of 2
- If n > 1, divide A and B into four n/2 × n/2 matrices

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

and compute the matrix product as:

$$C = AB = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

- Base case: when n = 1

Divide-and-Conquer algorithm

- What is the RT for the combine step?

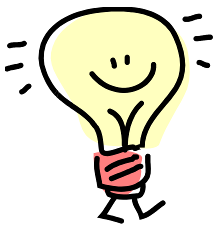
$$RT = \Theta(n^2)$$

- How to express RT for this Divide-and-Conquer using a recurrence?

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

- What is the RT ?

case 1 of the Master Thm $\Rightarrow T(n) = \Theta(n^3)$
no improvement …



- Idea: use 7 subproblems (instead of 8)

Divide-and-Conquer algorithm

Strassen's algorithm computes 7 subproblems:

$$q_1 = (a_{11} + a_{22}) * (b_{11} + b_{22})$$
$$q_2 = (a_{21} + a_{22}) * b_{11}$$
$$q_3 = a_{11} * (b_{12} - b_{22})$$
$$q_4 = a_{22} * (b_{21} - b_{11})$$
$$q_5 = (a_{11} + a_{12}) * b_{22}$$
$$q_6 = (a_{21} - a_{11}) * (b_{11} + b_{12})$$
$$q_7 = (a_{12} - a_{22}) * (b_{21} + b_{22})$$

Matrix product is computed as:

$$AB = \begin{pmatrix} q_1 + q_4 - q_5 + q_7 & q_3 + q_5 \\ q_2 + q_4 & q_1 + q_3 - q_2 + q_6 \end{pmatrix}$$

Strassen's algorithm, RT Analysis

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

Case 1 of the Master Thm $\Rightarrow$

$$T(n) = \Theta(\, n^{\log_2^7}\,) = \Theta(n^{2.807})$$