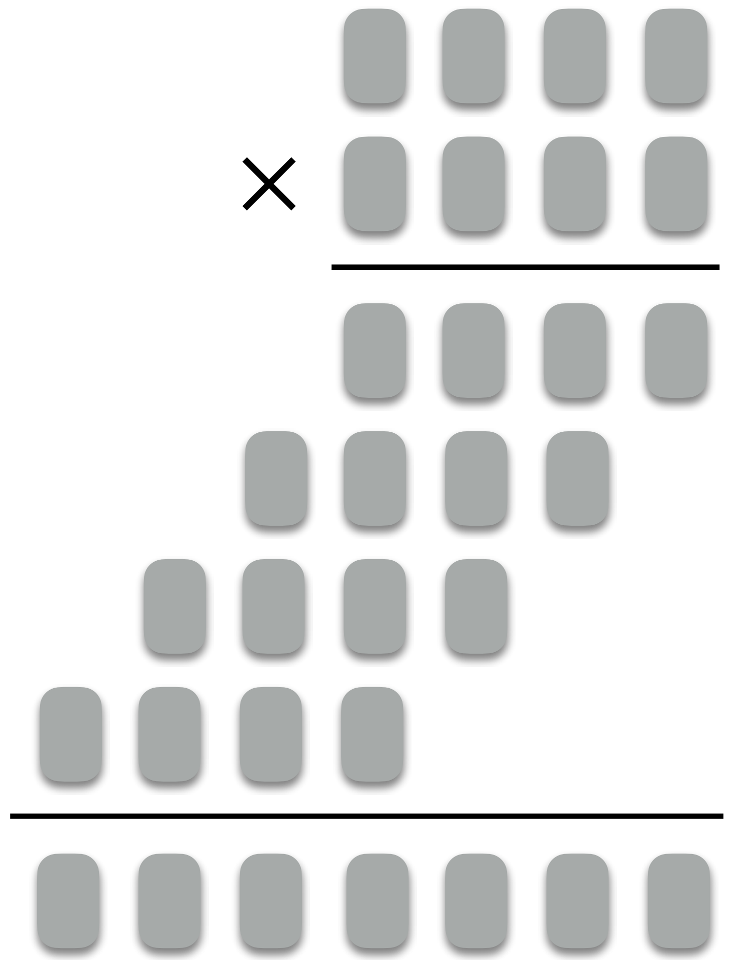
CSE202
Design and Analysis of Algorithms

Week 2 — Divide & Conquer 1:
How fast can we multiply?

Naive Multiplication



Input: two n-digit integers

n multiplications $+\ O(n)$ carries

$\vdots$

n multiplications $+\ O(n)$ carries

$O(n^2)$ additions $+\ O(n)$ carries

Output: $\leq 2n$ digits

For integers $\leq N$ this is $O(\log^2 N)$

Total: $O(n^2)$ digit (or bit) operations

Quadratic algorithm: #operations $O(n^2)$ for an input size n

One Second of Computation

With a good polynomial or integer library,
1 sec. is sufficient to

multiply two integers with 30,000,000 digits;

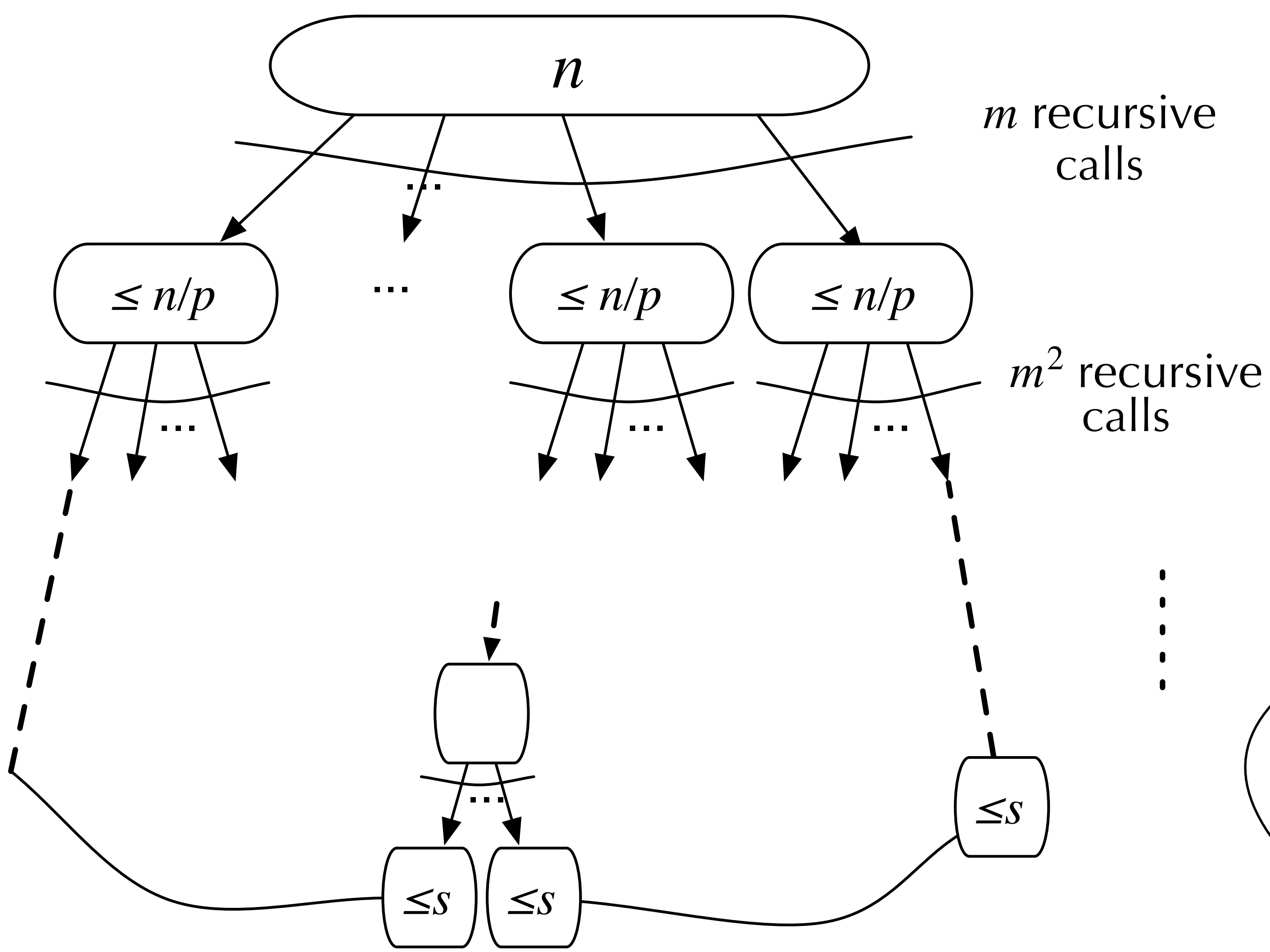
multiply two polynomials of degree 650,000;

multiply two matrices of size 850x850;

(but factor an integer with 42 digits only).

1 sec. is in the asymptotic regime of the algorithms

Divide and Conquer



The total cost may be

- mostly at the top (quickselect)
- mostly at the leaves (Karatsuba, Strassen)
- balanced along the levels (binary powering, mergesort)

this week

I. Polynomials

Polynomials and Integers

Polynomials		Integers
$(7x^2+3x+5)\times(2x+3)$		735
$= (7x^2+3x+5)\times 3$		$\times\ 23$
$+(7x^2+3x+5)\times 2x$	$\longrightarrow$	
$= 21x^2 + 9x + 15$	$x \mapsto 10$	2205
$+14x^3 + 6x^2 + 10x$		$+\ 1470$
$= 14x^3 + 27x^2 + 19x + 15$		16905
degree n_1 $\times$ degree n_2		n_1 digits $\times$ n_2 digits

$\longrightarrow$ $(n_1+1)(n_2+1)$ multiplications
$+O(n_1 n_2)$ additions

of $\left\{\begin{array}{l}\text{coefficients}\\ \text{digits}\end{array}\right.$

Polynomials behave like integers, without carries

Divide & Conquer by Itself Doesn't Help

$$F \text{ and } G \text{ of degree } < n \mapsto H := FG$$

1. If $n = 1$ return FG
2. Let $k := \lceil n/2 \rceil$
3. Split $F = F_0 + x^k F_1, G = G_0 + x^k G_1$
   F_0, F_1, G_0, G_1 of degree $< k$
4. Compute recursively
   $H_0 := F_0 G_0,\ H_1 := F_0 G_1,\ H_2 := F_1 G_0,\ H_3 := F_1 G_1$
5. Return $H_0 + x^k (H_1 + H_2) + x^{2k} H_3$

Complexity: $C(n) \leq 4C(\lceil n/2 \rceil) + \lambda n$ coefficient operations

Complexity of the Naive DAC

Notation:

$$\lceil x/2 \rceil_1 = \lceil x/2 \rceil$$

$$\lceil x/2 \rceil_{k+1} = \left\lceil \lceil x/2 \rceil_k / 2 \right\rceil$$

N : power of 2 s.t. $n \le N < 2n$

$$C(n) \le 4C(\lceil n/2 \rceil) + \lambda n$$

iterate once

$$\le \lambda n + 4\lambda \lceil n/2 \rceil + 16C(\lceil n/2 \rceil_2)$$

iterate k-1 times

$$\le \lambda \left(n + 2(2\lceil n/2 \rceil_1) + \cdots + 2^{k-1}(2^{k-1}\lceil n/2 \rceil_{k-1}) \right) + 2^{k+1}C(\lceil n/2 \rceil_k)$$

use N

$$\le \lambda N \left(1 + 2 + \cdots + 2^{k-1} \right) + 4^k C(\lceil n/2 \rceil_k)$$

bound geometric series

$$\le 4^k \left(\lambda \frac{N}{2^k} + C(\lceil n/2 \rceil_k) \right)$$

use $k = \lceil \log_2 n \rceil$

$$= O(n^2)$$

An extra idea is needed to beat the naive algorithm

Polynomials of Degree 1

$$F = f_0 + f_1 T, \quad G = g_0 + g_1 T \quad \mapsto \quad H := FG = h_0 + h_1 T + h_2 T^2$$

Naive algorithm:

$$H = (\underbrace{f_0 g_0}) + (\underbrace{f_0 g_1} + \underbrace{f_1 g_0})T + \underbrace{f_1 g_1}\, T^2$$

4 multiplications
& 1 addition

Interpolation from 3 values:

$$h_0 = F(0)G(0) = f_0 g_0$$ 1 mult.

$$h_2 = \text{"}F(\infty)G(\infty)\text{"} = f_1 g_1$$ 1 mult.

$$\tilde{h}_1 = h_0 + h_1 + h_2 = F(1)G(1) = (f_0 + f_1)(g_0 + g_1)$$ 1 mult.

$$FG = h_0 + (\tilde{h}_1 - h_0 - h_2)T + h_2 T^2$$

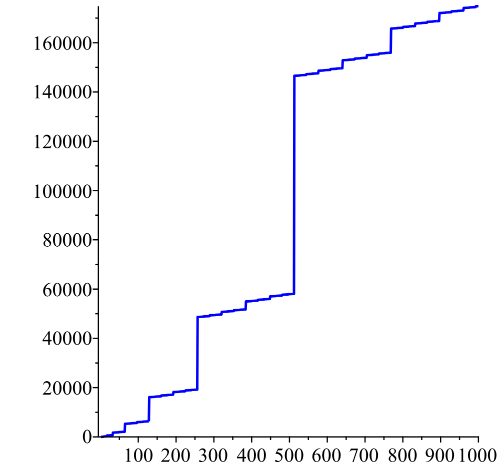
3 multiplications, 2 additions, 2 subtractions

Karatsuba's Algorithm

F and G of degree $< n \mapsto H := FG$

Idea: Evaluate $FG = h_0 + (\tilde{h}_1 - h_0 - h_2)T + h_2T^2$ at $T = x^k$.

1. If n is small, use naive multiplication
2. Let $k := \lceil n/2 \rceil$
3. Split $F = F_0 + x^kF_1, G = G_0 + x^kG_1$
   F_0, F_1, G_0, G_1 of degree $< k$
4. Compute recursively
   $H_0 := F_0G_0,\ H_2 := F_1G_1,\ \tilde{H}_1 := (F_0 + F_1)(G_0 + G_1)$
5. Return $H_0 + x^k(\tilde{H}_1 - H_0 - H_2) + x^{2k}H_2$



$u_n = n + 3u_{\lceil n/2 \rceil}, u_1 = 1$

Complexity: $C(n) \leq 3C(\lceil n/2 \rceil) + \lambda n$ coefficient operations

Complexity of Karatsuba's Algorithm

Notation:

$\lceil x/2 \rceil_1 = \lceil x/2 \rceil$

$\lceil x/2 \rceil_{k+1} = \left\lceil \lceil x/2 \rceil_k / 2 \right\rceil$

N : power of 2 s.t. $n \le N < 2n$

$$C(n) \le 3C(\lceil n/2 \rceil) + \lambda n$$

iterate once

$$\le \lambda n + 3\lambda \lceil n/2 \rceil + 9C(\lceil n/2 \rceil_2)$$

iterate k-1 times, use N

$$\le \lambda N \left(1 + \frac{3}{2} + \cdots + \left(\frac{3}{2}\right)^{k-1} \right) + 3^k C(\lceil n/2 \rceil_k)$$

reorder sum

$$\le \lambda N \left(\frac{3}{2}\right)^{k-1} \left(1 + 2/3 + \cdots + (2/3)^{k-1}\right) + 3^k C(\lceil n/2 \rceil_k)$$

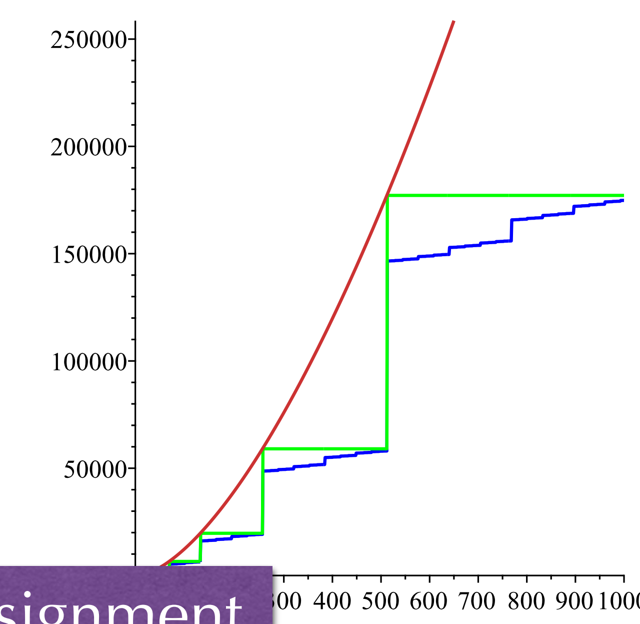
bound geometric series

$$\le 3^k \left(2\lambda \frac{N}{2^k} + C(\lceil n/2 \rceil_k) \right)$$

use $k = \lceil \log_2 n \rceil$

$$\le (2\lambda + 1) 3^{\lceil \log_2 n \rceil} = O(n^{\log_2 3})$$

≈ 1.58



Not the final word. See assignment.

II. Integers

From Polynomials to Integers

Recall: Polynomials behave like integers, without carries

No theorem of complexity equivalence exists, but the algorithms over polynomials can often be adapted to integers, with the same complexity.

Karatsuba's Algorithm for Integers

F and G integers $< 2^n \mapsto H := FG$

1. If n is small, use naive multiplication
2. Let $k := \lceil n/2 \rceil$
3. Split $F = F_0 + 2^k F_1, G = G_0 + 2^k G_1$

   $F_0, F_1, G_0, G_1 < 2^k$
4. Compute recursively

   $H_0 := F_0 G_0, \quad H_2 := F_1 G_1, \quad \tilde{H}_1 := (F_0 + F_1)(G_0 + G_1)$
5. Return $H_0 + 2^k(\tilde{H}_1 - H_0 - H_2) + 2^{2k} H_2$

Obtained by changing x into 2 in the polynomial version.

Same algorithm as for polynomials,
similar (not exactly the same) complexity analysis.

$\to O(n^{\log_2 3})$ bit operations ≈ 1.58

Which of these Algorithms is Best?

None of them!

GMP (the Gnu Multiprecision Library) uses:

# 64-bit words	approx # digits	Algorithm
0	0	Naive
26	500	Karatsuba
73	1,400	Toom - 3
208	4,000	Toom - 4
4736	90,000	FFT

Assignment this week

III. Matrices

Matrix Multiplication

Input: two $n \times n$ matrices A, X with $n = 2^k$

Output: AX

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

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

Naive algorithm: $O(n^3)$

Matrix Multiplication: Strassen's Algorithm

Input: two $n \times n$ matrices A, X with $n = 2^k$

Output: AX

1. If $n = 1$, return AX
2. Split $A = \left(\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right), X = \left(\begin{array}{c|c} x & y \\ \hline z & t \end{array}\right)$, with $(n/2) \times (n/2)$ blocks
3. Compute recursively the 7 products

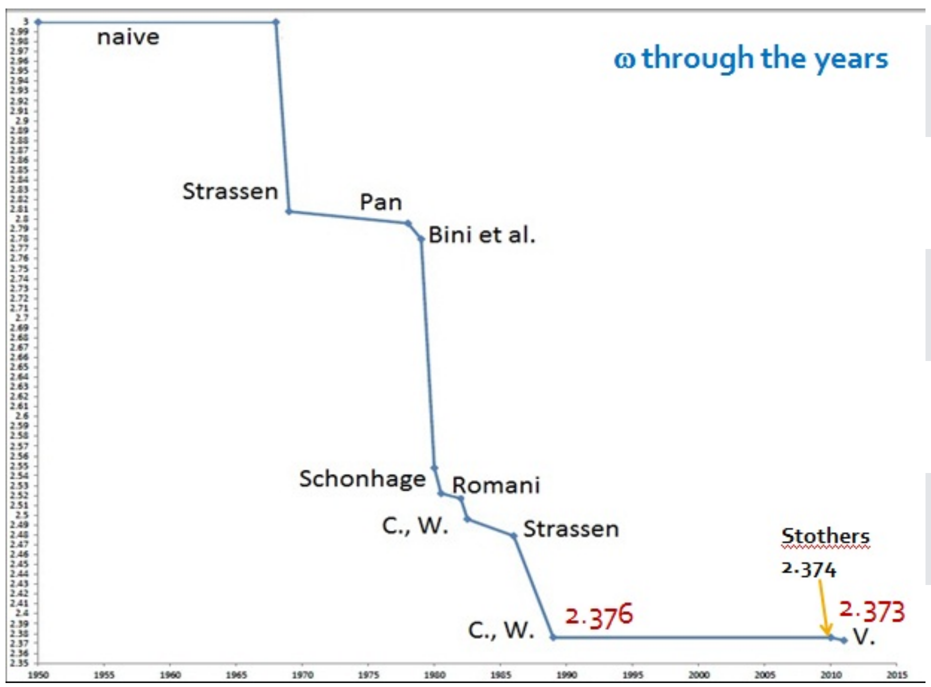
$$q_1 = a(x+z),\ q_2 = d(y+t),\ q_3 = (d-a)(z-y),$$
$$q_4 = (b-d)(z+t),\ q_5 = (b-a)z,$$
$$q_6 = (c-a)(x+y),\ q_7 = (c-d)y$$

4. Return $\begin{pmatrix} q_1 + q_5 & q_2 + q_3 + q_4 - q_5 \\ q_1 + q_3 + q_6 - q_7 & q_2 + q_7 \end{pmatrix}$

Exercise:
prove the complexity
in $O(n^{\log_2 7})$ operations.

World Records

Best algorithm for matrix multiplication



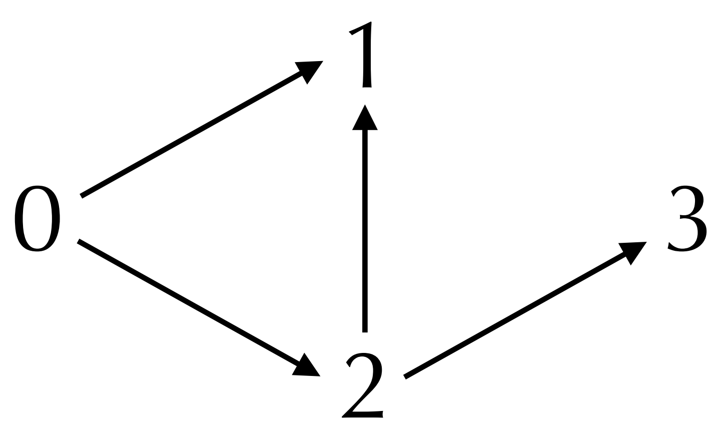
Year	Algorithm	Exponent
	Classical	3
1969	Strassen	2,808
1978	Pan	2,796
1981	Schönhage	2,522
1982	Romani	2,517
1981	Coppersmith-Winograd	2,496
1986	Strassen	2,479
1989	Coppersmith-Winograd	2,376
2012	Vassilevska Williams	2,3729
2014	Le Gall	2,3728

Big Open Question:
is 2 the limit?

Application: Graph Transitive Closure

Def. A **graph** is a pair (V,E) where

1. V is a finite set of nodes/vertices
2. $E \subseteq V \times V$ is a finite set of edges



({0,1,2,3},
{(0,1), (0,2), (2,1), (2,3)})

Adjacency matrix

$$
\begin{array}{c c}
\text{Nodes} & \begin{matrix} 0 & 1 & 2 & 3 \end{matrix} \\
\begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{array}
$$

Boolean matrix

Boolean Matrices: Multiplication

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

$$c_{ij} = \bigvee_{k=1}^{n} a_{ik} \wedge b_{kj}$$

Boolean ops (and, or) instead of algebraic ops (sum, product)

Algebraic multiplication + flatten all non-zero entries to 1

Graph Transitive Closure

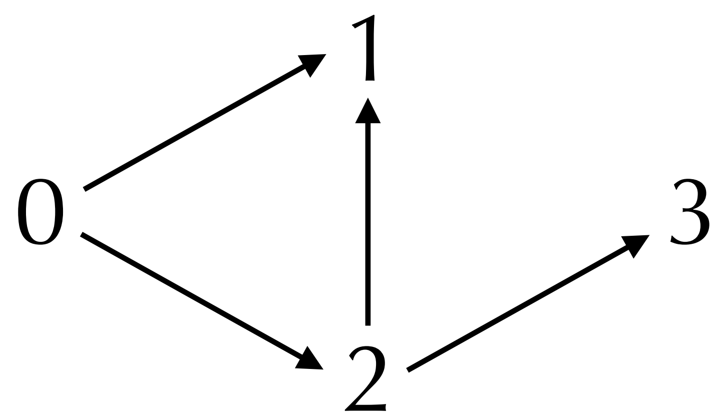
Let $G = (V,E)$ be a graph

A path from i to j is a sequence of edges $e_1,\ldots,e_n$ such that:

- the source of e_1 is i and the target of e_n is j
- for every $1 \leq k < n$, the target of e_k is the source of e_{k+1}

The transitive closure $G^* = (V, E^*)$ is the graph where $(u,v) \in E^*$ iff there is a path from u to v

Application: Graph Transitive Closure



({0,1,2,3},
{(0,1), (0,2), (2,1), (2,3)})

Adjacency matrix

$$\begin{array}{c c} \text{Nodes} & \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} & \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

If A is the adjacency matrix of a graph, then

- $(A^k)_{ij} = 1$ iff there is a path of length k from i to j
- if I is the identity matrix, then $(A \vee I)^k_{ij} = 1$ iff there exists a path of length at most k from i to j

Application: Graph Transitive Closure

Let G = (V,E) be a graph with n vertices

If A is the adjacency matrix of G, then $(A \vee I)^{n-1}$ is the adjacency matrix of G*

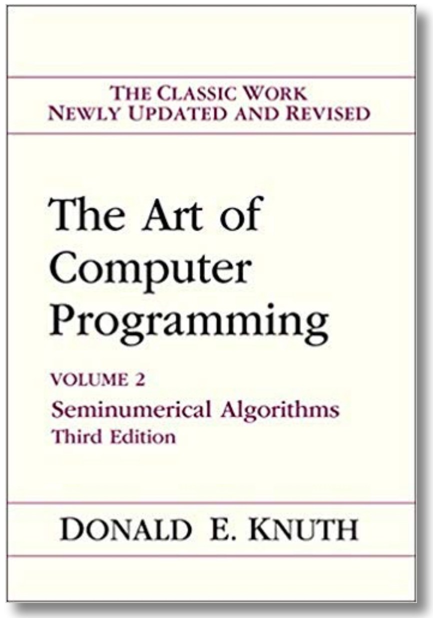
The matrix $(A \vee I)^{n-1}$ can be computed by log n squaring operations/multiplications

$O(n^{\log_2 7} \cdot log_2 n)$

References for this lecture

The slides are designed to be self-contained.

They were prepared using the following
books that I recommend if you want to learn more:



Next

Assignment this week: generalisation of Karatsuba's algorithm

Next tutorial: multiplication, transitive closure

Next week: algorithms related to statistics

Feedback

Moodle

Questions: constantin.enea@polytechnique.edu