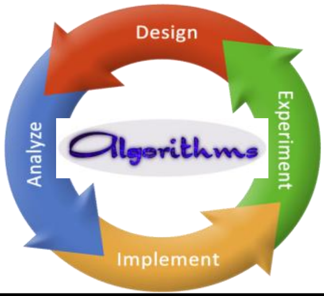
Algorithm Design and Analysis

Victor Adamchik — CS 15-451 Spring 2014

Lecture 2 — Jan 15, 2014 — Carnegie Mellon University

# Strassen's Algorithm



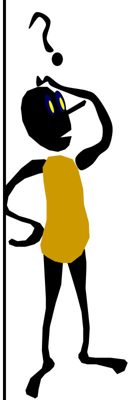

## Matrix Multiplication

A B C = AxB

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

Let A and B be nxn matrices, then their product C=A*B is

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



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

What is the complexity of this algorithm (in terms of multiplications)?

$O(n^3)$

## Divide and Conquer

The idea is to divide the size of the problem in half. This corresponds to dividing each of the matrices into quarters, each n/2 x n/2 size, and multiply those quarters.

## Algorithm

Let $n = 2^k$ and M(A,B) denote the matrix product

1. if A is 1x1 matrix, return $a_{11} * b_{11}$.
2. write $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ $B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$

   where $A_{ij}$ and $B_{ij}$ are n/2 x n/2 matrices.
3. Compute $C_{ij} = M(A_{i1},B_{1j}) + M(A_{i2},B_{2j})$
4. Return $\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$

## Correctness

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

This basically says that if the entries of A and B are themselves matrices, the usual matrix multiplication works by substituting the blocks into the formula.

Let us prove it for the left upper entry.

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

## Correctness

Since $1 \le i, j \le n/2$.

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

$$c_{ij} = \sum_{k=1}^{n/2} a_{ik}b_{kj} + \sum_{k=n/2+1}^{n} a_{ik}b_{kj} = \sum_{k=1}^{n/2} A_{ik}B_{kj} + \sum_{k=n/2+1}^{n} A_{ik}B_{kj}$$

$$= (A_{11} \cdot B_{11})(i,j) + (A_{12} \cdot B_{21})(i,j)$$

$$= (A_{11} \cdot B_{11} + A_{12} \cdot B_{21})(i,j)$$

Similar proof for the other blocks.

## Worst-case complexity

$C_{ij} = M(A_{i1},B_{1j}) + M(A_{i2},B_{2j})$

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

On each step we compute 4 matrices $C_{ij}$, each requires two recursive calls.

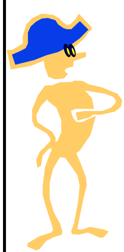
Let T(n) denote the number of multiplications, then

$$T(n) = 8T(n/2) + O(n^2)$$
$$T(1) = 1$$

Matrix addition

The Master Theorem gives $\Theta(n^3)$.

## Strassen's Algorithm (1968)



Do we need all 8 multiplications or can we find a clever way of doing it with fewer?

Strassen a German mathematician born in 1936

## Strassen's Algorithm

For 2 x 2 matrices

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} s_1 + s_2 - s_4 + s_6 & s_4 - s_5 \\ s_6 + s_7 & s_2 - s_3 + s_5 - s_7 \end{pmatrix}$$

$s_1 = (a_{12}-a_{22})(b_{21}+b_{22})$
$s_2 = (a_{11}+a_{22})(b_{11}+b_{22})$
$s_3 = (a_{11}-a_{21})(b_{11}+b_{12})$
$s_4 = (a_{11}+a_{12})\, b_{22}$
$s_5 = a_{11}(b_{12}-b_{22})$
$s_6 = a_{22}(b_{21}-b_{11})$
$s_7 = (a_{21}+a_{22})\, b_{11}$

It takes 7 multiplications

## Correctness

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} s_1 + s_2 - s_4 + s_6 & s_4 - s_5 \\ s_6 + s_7 & s_2 - s_3 + s_5 - s_7 \end{pmatrix}$$

Proof for a lower left entry:

$s_6 = a_{22}(b_{21}-b_{11})$
$s_7 = (a_{21}+a_{22})\, b_{11}$

$s_6+s_7 = a_{21} b_{11} + a_{22} b_{21}$

$s_6+s_7 = a_{22}(b_{21}-b_{11}) + (a_{21}+a_{22})\, b_{11} =$
$a_{22} b_{21} - a_{22} b_{11} + a_{21} b_{11} + a_{22} b_{11} = a_{21} b_{11} + a_{22} b_{21}$

## Strassen's Algorithm

This holds for a block matrix multiplication

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} S_1 + S_2 - S_4 + S_6 & S_4 - S_5 \\ S_6 + S_7 & S_2 - S_3 + S_5 - S_7 \end{pmatrix}$$

where $A_{ij}$ and $B_{ij}$ are n/2 x n/2 matrices and matrices $S_1$, …., $S_7$ are defined on the previous slide.

## Worst-case complexity

Let T(n) denote the number of multiplications, then

$$T(n) = 7\,T(n/2) + O(n^2)$$
$$T(1) = 1$$

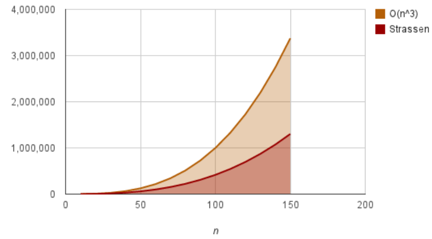
Matrix addition

The Master Theorem gives $\Theta(n^{\log 7}) = \Theta(n^{2.807})$.

If we count additions, then

$$T(n) = 7\,T(n/2) + 18(n/2)^2$$
$$T(1) = 1$$

## Time complexity




## Space Complexity

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}\begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} S_1 + S_2 - S_4 + S_6 & S_4 - S_5 \\ S_6 + S_7 & S_2 - S_3 + S_5 - S_7 \end{pmatrix}$$

$S_1 = (A_{12}-A_{22})(B_{21}+B_{22})$
$S_2 = (A_{11}+A_{22})(B_{11}+B_{22})$
$S_3 = (A_{11}-A_{21})(B_{11}+B_{12})$
$S_4 = = (A_{11}+A_{12})\,B_{22}$
$S_5 = A_{11}\,(B_{12}-B_{22})$
$S_6 = A_{22}\,(B_{21}-B_{11})$
$S_7 = (A_{21}+A_{22})\,B_{11}$

We need to compute and then store matrices $S_1, \ldots, S_7$

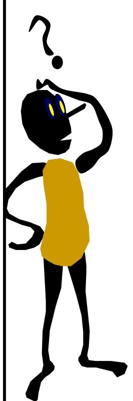
To compute them we need two scratch arrays, so 9 total.

## Space Complexity

Let W(n) be the space complexity

$$W(n) = W(n/2) + 9(n/2)^2$$
$$W(1) = 1$$

Solving this gives $W(n) = 3\,n^2$.



How would you extend Strassen's algorithm to matrix dimensions differ from $2^k$?

Pad the matrices with zeros.

## Polynomial Multiplication

How would you multiply two polynomials?

## Polynomial Multiplication

$$A(x) = \sum_{k=0}^{n} a_k x^k \qquad B(x) = \sum_{k=0}^{n} b_k x^k$$

$$C(x) = A(x)B(x) = \sum_{j=0}^{n} \sum_{k=0}^{n} a_j b_k x^{j+k}$$

This has $O(n^2)$ complexity. We can do much better!

## Karatsuba Revisited

$$A(x) = \sum_{k=0}^{n} a_k x^k \qquad B(x) = \sum_{k=0}^{n} b_k x^k$$

$$A(x) = A_1\, x^{n/2} + A_0$$ Same for B(x)

For example, $1 + 3x + x^2 + 7x^3 = (1 + 3x) + x^2\,(1+7x)$

$$A(x)B(x) = C_2\, x^n + C_1 x^{n/2} + C_0$$

where

$$C_2 = A_1 B_1$$
$$C_1 = (A_0 + A_1)(B_0 + B_1) - A_0 B_0 - A_1 B_1$$
$$C_0 = A_0 B_0$$

## Polynomial Multiplication

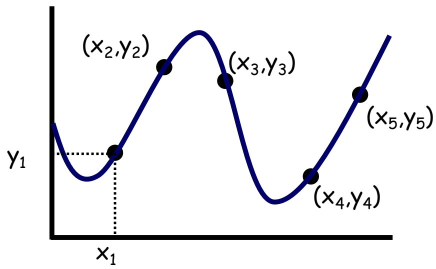
Karatsuba's polynomial multiplication can be done with at most $O(n^{\log 3})$ operations.

Observe, the algorithm in fact multiplies only linear polynomials (2 terms) with three scalar multiplications.

In the next slides we outline a slightly different approach that is based on interpolation.

## Interpolation

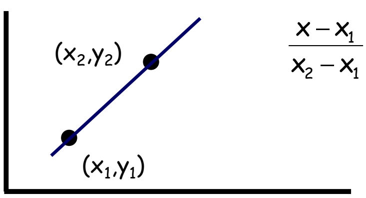
Say you're given a bunch of "data points"




Can you find a (low-degree) polynomial which "fits the data"?

## Interpolation

There is a unique linear polynomial going through 2 points




$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

Correspondence between a set of 2 points and a line (a polynomial of the first order)

## Uniqueness

Correspondence between a set of points and a polynomial

Theorem:

There is exactly one polynomial P(x) of degree at most n such that $P(a_k) = b_k$ for all $k = 0, \ldots, n$.

## Multiplication by Interpolation

Let us multiply polynomials of degree one

$A(x) = a_0 + a_1 x, \quad B(x) = b_0 + b_1 x$

Suggested points for evaluation: 0, 1, $\infty$

$A(0) = a_0$, $A(1) = a_0 + a_1$, $A(\infty) = a_1$

$B(0) = b_0$, $B(1) = b_0 + b_1$, $B(\infty) = b_1$

$\infty$ means, take the leading coefficient

Compute:

$c_0 = a_0 b_0$, $c_1 = (a_0 + a_1)(b_0 + b_1)$, $c_2 = a_1 b_1$

Find a polynomial passing through these points!

## Karatsuba again…

Find a polynomial passing through these points

$(0, a_0 b_0)$, $(1, (a_0 + a_1)(b_0 + b_1))$, $(\infty, a_1 b_1)$

Clearly it must be a quadratic polynomial

$c_0 + c_1 x + c_2 x^2$

Setting $x = 0$, gives that $c_0 = a_0 b_0$

Setting $x = \infty$, gives that $c_2 = a_1 b_1$

Setting $x = 1$, gives that $c_0 + c_1 + c_2 = (a_0 + a_1)(b_0 + b_1)$

It follows, $c_1 = (a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1$

Wow, exactly like in Karatsuba's algorithm

## Toward to the Fast Fourier Transform

To compute the polynomial product $A(x)B(x)$,

1) evaluate $A(x)$ and $B(x)$ at some points $x_k$,
2) multiply $A(x_k)B(x_k)$,
3) then find the polynomial which passes through these points.