Strassen's algorithm

Matrices (repetition)

Matrix = square of numbers:

$$\begin{bmatrix} 1 & 6 & 4 \\ 2 & 5 & 7 \\ 9 & 1 & 1 \end{bmatrix}$$

The above is 3×3

Today: all matrices are n×n square matrices. (i.e.n indicates the side length of the matrices.)

Matrices

Plus for matrices:

$$\begin{bmatrix}1 & 6 & 4\\ 2 & 5 & 7\\ 9 & 1 & 1\end{bmatrix} + \begin{bmatrix}3 & 2 & 1\\ 4 & 3 & 2\\ 5 & 4 & 3\end{bmatrix} = \begin{bmatrix}? & ? & ?\\ ? & ? & ?\\ ? & ? & ?\end{bmatrix}$$

n=3

n= 3
$$\begin{bmatrix}1 & 6 & 4\\ 2 & 5 & 7\\ 9 & 1 & 1\end{bmatrix} + \begin{bmatrix}3 & 2 & 1\\ 4 & 3 & 2\\ 5 & 4 & 3\end{bmatrix} = \begin{bmatrix}1+3 & 6+2 & 4+1\\ 2+4 & 5+3 & 7+2\\ 9+5 & 1+4 & 1+3\end{bmatrix} = \begin{bmatrix}4 & 8 & 5\\ 6 & 8 & 9\\ 14 & 5 & 4\end{bmatrix}$$

entries:3*3=3^2

Time? $\Theta(n^2)$. you need to interate through n^2 elements, for each element you add another element, which is O(1) (constant). The

Optimal, since the output is of size n^2 .

Matrices

Multiplication for matrices:

$$\begin{bmatrix} 1 & 6 & 4 \\ 2 & 5 & 7 \\ 9 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 & 1 \\ 4 & 3 & 2 \\ 5 & 4 & 3 \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

$$\begin{bmatrix} 1 & 6 & 4 \\ \cancel{2} & \cancel{5} & \cancel{7} \\ 9 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 & \cancel{1} \\ 4 & 3 & \cancel{2} \\ 5 & 4 & \cancel{3} \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & 33 \\ ? & ? & ? \end{bmatrix}$$

$$33 = 2 \cdot 1 + 5 \cdot 2 + 7 \cdot 3$$

$$\begin{bmatrix} 1 & 6 & 4 \\ 2 & 5 & 7 \\ \cancel{9} & \cancel{1} & \cancel{1} \end{bmatrix} \cdot \begin{bmatrix} 3 & \cancel{2} & 1 \\ 4 & \cancel{3} & 2 \\ 5 & \cancel{4} & 3 \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & 33 \\ ? & 25 & ? \end{bmatrix}$$

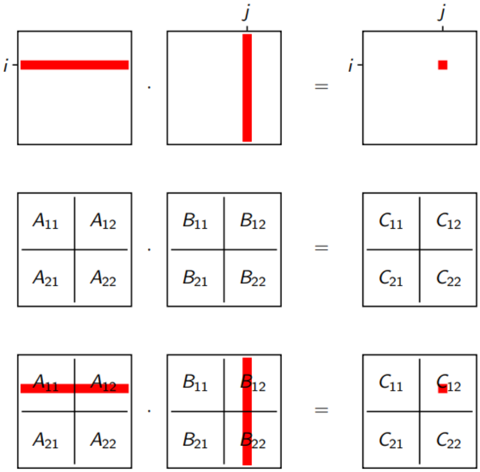
$$25 = 9 \cdot 2 + 1 \cdot 3 + 1 \cdot 4$$

3x3 matrix, with 3 multiplications = 3^3

n^2 elements in a matrix, and have n multiplications, so n^2*n= n^3

Time? $\Theta(n^3)$. Optimal?? Other algorithms??

Recursive algorithm for multiplication?



where they meet

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

Skriv tekst her

Recursive algorithm for multiplication?

$$\begin{array}{|c|c|} \hline A_{11} & A_{12} \\ \hline A_{21} & A_{22} \\ \hline \end{array} \cdot \begin{array}{|c|c|} \hline B_{11} & B_{12} \\ \hline B_{21} & B_{22} \\ \hline \end{array} = \begin{array}{|c|c|} \hline C_{11} & C_{12} \\ \hline C_{21} & C_{22} \\ \hline \end{array}$$

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

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

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

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

Matrix addition: $O(n^2)$

Matrix multiplication: Recursive call to matrix multiplication on $n/2 \times n/2$ matrices. (Base case: $n=1 \Rightarrow$ multiplication of numbers.)

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

8 Multiplications

Recursive algorithm for multiplication

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

Master theorem:

- $\alpha = \log_b(a) = \log_2(8) = 3$
- $f(n) = n^2$

$n^2 = O(n^{\alpha - 0.1}) \Rightarrow$ Case 1

$$T(n) = \Theta(n^\alpha) = \Theta(n^3)$$

Same as the regular algorithm.

Streets [1969]

Calculate:

$S_1 = B_{12} - B_{22}$	$S_6 = B_{11} + B_{22}$
$S_2 = A_{11} + A_{12}$	$S_7 = A_{12} - A_{22}$
$S_3 = A_{21} + A_{22}$	$S_8 = B_{21} + B_{22}$
$S_4 = B_{21} - B_{11}$	$S_9 = A_{11} - A_{21}$
$S_5 = A_{11} + A_{22}$	$S_{10} = B_{11} + B_{12}$

Time: $O(n^2)$, since both addition and subtraction take this time.

Streets [1969]

Calculate: Both A and B have a side length of n/2, the same does S.

$P_1 = A_{11} \cdot S_1$

$P_2 = S_2 \cdot B_{22}$

$P_3 = S_3 \cdot B_{11}$

$P_4 = A_{22} \cdot S_4$

$P_5 = S_5 \cdot S_6$

$P_6 = S_7 \cdot S_8$

$P_7 = S_9 \cdot S_{10}$

red dots is recursive multiplication

7 recursive calls to matrix multiplication on $n/2 \times n/2$ matrices.

7 multiplications instead of 8

only work for a matrices of 2^x x 2^x
because you are diving by 2.

unless you fill out with 0!

Streets [1969]

Now check that the following applies:

$$P_5+P_4-P_2+P_6 = A_{11}\cdot B_{11}+A_{12}\cdot B_{21}$$

$$P_1+P_2 = A_{11}\cdot B_{12}+A_{12}\cdot B_{22}$$

$$P_3+P_4 = A_{21}\cdot B_{11}+A_{22}\cdot B_{21}$$

$$P_5+P_1-P_3-P_7 = A_{21}\cdot B_{12}+A_{22}\cdot B_{22}$$

That is, output can be calculated in $O(n^2)$ time from $P_1, \ldots, P_7$, since

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

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

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

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

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

Streets [1969]

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

Master theorem:

- $\alpha = \log_b(a) = \log_2(7) = 2.80735...$
- $f(n) = n^2$

$n^2 = O(n^{\alpha - 0.1}) \Rightarrow$ Case 1

$$T(n) = \Theta(n^\alpha) = O(n^{2.81})$$

Better than the regular algorithm!

In the classic approach for multiplying two 2×2 Matrices, you need 8 multiplications and 4 additions. Strassen’s method, by cleverly reorganizing the computation, reduces the number of multiplications to 7. Although it performs more additions (18, in fact) for 2×2 matrices, the net effect of these extra additions is still beneficial because when the algorithm is applied recursively, the overall number of operations (both multiplications and additions) across all levels of recursion is reduced.