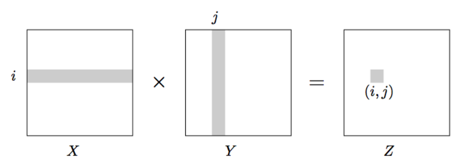
Strassen's Algorithm for Matrix Multiplication

Chapter 4

Matrix Multiplication



Matrix Multiplication

Input: Two $n \times n$ (square) matrices, $A = (a_{ij})$ and $B = (b_{ij})$.
Output: $n \times n$ matrix $C = (c_{ij})$, where $C = A \cdot B$, i.e.,

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

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

Need to compute n^2 entries of C. Each entry is the sum of n values.

Obvious Algorithm

SQUARE-MAT-MULT(A, B, n)

let C be a new $n \times n$ matrix

for $i = 1$ **to** n

  for $j = 1$ **to** n

    $c_{ij} = 0$

    for $k = 1$ **to** n

      $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$

return C

Running Time?

o-Notation (Little-Oh)

- Not to be confused with O-notation (Big-Oh)
- Big-Oh
  - Asymptotic upper bound may or may not be asymptotically tight
  - $2n^2 = O(n^2)$ $2n = O(n^2)$
- Little-Oh
  - Upper bound that is not asymptotically tight
  - $2n^2 \neq o(n^2)$ $2n = o(n^2)$

Matrix Multiplication

- Running time: $\Theta(n^3)$
- Question: is it $\Omega(n^3)$?
  - Must compute n^2 entries
  - Each entry is the sum of n terms
- Answer: no! It is $o(n^3)$

Strassen's Algorithm

- Strassen reduced the asymptotic complexity to $\Theta(n^{\lg 7})$
- 2.80 > lg7 > 2.81
- This is asymptotically better than the simple algorithm

Divide-and-Conquer

- To keep things simple, assume that n is a power of 2

Partition each of A, B, C into four $n/2 \times n/2$ matrices:

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

Divide and Conquer

Rewrite $C = A \cdot B$ as

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

giving the four equations

$$\begin{aligned} C_{11} &= A_{11} \cdot B_{11} + A_{12} \cdot B_{21}, \\ C_{12} &= A_{11} \cdot B_{12} + A_{12} \cdot B_{22}, \\ C_{21} &= A_{21} \cdot B_{11} + A_{22} \cdot B_{21}, \\ C_{22} &= A_{21} \cdot B_{12} + A_{22} \cdot B_{22}. \end{aligned}$$

Divide and Conquer

REC-MAT-MULT(A, B, n)
let C be a new $n \times n$ matrix
if $n == 1$
    $c_{11} = a_{11} \cdot b_{11}$
else partition A, B, and C into $n/2 \times n/2$ submatrices
    $C_{11} = \text{REC-MAT-MULT}(A_{11}, B_{11}, n/2) + \text{REC-MAT-MULT}(A_{12}, B_{21}, n/2)$
    $C_{12} = \text{REC-MAT-MULT}(A_{11}, B_{12}, n/2) + \text{REC-MAT-MULT}(A_{12}, B_{22}, n/2)$
    $C_{21} = \text{REC-MAT-MULT}(A_{21}, B_{11}, n/2) + \text{REC-MAT-MULT}(A_{22}, B_{21}, n/2)$
    $C_{22} = \text{REC-MAT-MULT}(A_{21}, B_{12}, n/2) + \text{REC-MAT-MULT}(A_{22}, B_{22}, n/2)$
return C

How to Partition a Matrix?

- How long would it take to create 12 new n/2 x n/2 matrices?
- $\Theta(n^2)$
- Trick: use index calculations!

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

Analysis

- Dividing takes $\Theta(1)$ time, using index calculations. *[Otherwise, $\Theta(n^2)$ time.]*
- Conquering makes 8 recursive calls, each multiplying $n/2 \times n/2$ matrices $\Rightarrow 8T(n/2)$.
- Combining takes $\Theta(n^2)$ time to add $n/2 \times n/2$ matrices four times. *[Doesn't even matter asymptotically whether we use index calculations or copy: would be $\Theta(n^2)$ either way.]*

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ 8T(n/2) + \Theta(n^2) & \text{if } n > 1 . \end{cases}$$

Analysis

- $T(n) = \Theta(n^3)$
- No better than simple matrix multiply ☹

Strassen's Method

- Idea: Make the recursion tree less bushy
  - Perform only 7 recursive multiplications of n/2 x n/2 matrices, rather than 8.
  - Will cost several additions of n/2 x n/2 matrices, but just a constant number more
    - can still absorb the constant factor for matrix additions into the $\Theta(n^2)$ term.

The Algorithm

1. As in the recursive method, partition each of the matrices into four $n/2 \times n/2$ submatrices. Time: $\Theta(1)$.
2. Create 10 matrices $S_1, S_2, \ldots, S_{10}$. Each is $n/2 \times n/2$ and is the sum or difference of two matrices created in previous step. Time: $\Theta(n^2)$ to create all 10 matrices.
3. Recursively compute 7 matrix products $P_1, P_2, \ldots, P_7$, each $n/2 \times n/2$.
4. Compute $n/2 \times n/2$ submatrices of C by adding and subtracting various combinations of the P_i. Time: $\Theta(n^2)$.

Analysis

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ 7T(n/2) + \Theta(n^2) & \text{if } n > 1 . \end{cases}$$

$$T(n) = \Theta(n^{\lg 7})$$

Details

Step 2: Create the 10 matrices

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

Add or subtract $n/2 \times n/2$ matrices 10 times $\Rightarrow$ time is $\Theta(n/2)$.

Details

Step 3: Create the 7 matrices

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

The only multiplications needed are in the middle column; right-hand column just shows the products in terms of the original submatrices of A and B.

Details

Step 4: Add and subtract the P_i to construct submatrices of C:

$$C_{11} = P_5 + P_4 - P_2 + P_6 ,$$
$$C_{12} = P_1 + P_2 ,$$
$$C_{21} = P_3 + P_4 ,$$
$$C_{22} = P_5 + P_1 - P_3 - P_7 .$$

Notes

- Strassen's algorithm was the first to beat Θ (n^3) time, but it's not the asymptotically fastest known.
- A method by Coppersmith and Winograd runs in $O(n^{2.376})$ time.

Example

- Use Strassen's algorithm to compute the matrix product

```
| 1   2 |     | 5   6 |
| 3   4 |  *  | 7   8 |
```