Algorithms and Data Structures
Strassen’s Algorithm
Tutorials

- Start in week (week 3)
- Tutorial allocations are linked from the course webpage
  http://www.inf.ed.ac.uk/teaching/courses/ads/
The Master Theorem for solving recurrences

**Theorem**

Let \( n_0 \in \mathbb{N} \), \( k \in \mathbb{N}_0 \) and \( a, b \in \mathbb{R} \) with \( a > 0 \) and \( b > 1 \), and let \( T : \mathbb{N} \rightarrow \mathbb{R} \) satisfy the following recurrence:

\[
T(n) = \begin{cases} 
\Theta(1) & \text{if } n < n_0, \\
a \cdot T(n/b) + \Theta(n^k) & \text{if } n \geq n_0.
\end{cases}
\]

Let \( c = \log_b(a) \); we call \( c \) the **critical exponent**. Then

\[
T(n) = \begin{cases} 
\Theta(n^c) & \text{if } k < c \quad (I), \\
\Theta(n^c \cdot \log(n)) & \text{if } k = c \quad (II), \\
\Theta(n^k) & \text{if } k > c \quad (III).
\end{cases}
\]

Theorem also holds if we replace \( a \cdot T(n/b) \) above by \( a_1 \cdot T(\lfloor n/b \rfloor) + a_2 \cdot T(\lceil n/b \rceil) \) for any \( a_1, a_2 \geq 0 \) with \( a_1 + a_2 = a \).
The Master Theorem (cont’d)

- We don’t have time to prove the Master Theorem in class. You can find the proof in Section 4.6 of [CLRS]. Section 4.4 of [CLRS], 2nd ed.
  Their version of the M.T. is a bit more general than ours.
- Consider the following examples:

  \[ T(n) = 4T(n/2) + n, \]
  \[ T(n) = 4T(\lfloor n/2 \rfloor) + n^2, \]
  \[ T(n) = 4T(n/2) + n^3. \]

  Could alternatively unfold-and-sum to prove the first and third of these (and to get an estimate for the second).

CLASS EXERCISE
Matrix Multiplication

Recall

The product of two \((n \times n)\)-matrices

\[ A = (a_{ij})_{1 \leq i, j \leq n} \quad \text{and} \quad B = (b_{ij})_{1 \leq i, j \leq n} \]

is the \((n \times n)\)-matrix \(C = AB\) where \(C = (c_{ij})_{1 \leq i, j \leq n}\) with entries

\[ c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}. \]

The Matrix Multiplication Problem

\textit{Input:} \((n \times n)\)-matrices \(A\) and \(B\)

\textit{Output:} the \((n \times n)\)-matrix \(AB\)
Matrix Multiplication

\[ \text{row } i \quad \begin{array}{c} \text{column } j \end{array} \quad = \quad \begin{array}{c} \text{a}_{i1} \quad \text{a}_{i2} \quad \cdots \quad \text{a}_{in} \end{array} \quad \begin{array}{c} \text{b}_{1j} \\ \text{b}_{2j} \\ \vdots \\ \text{b}_{nj} \end{array} \]

- There are \( n^2 \) different \( c_{ij} \) entries.
- There are \( n \) multiplications and \( n \) additions for each \( c_{ij} \).
Matrix Multiplication

- $n$ multiplications and $n$ additions for each $c_{ij}$.
- there are $n^2$ different $c_{ij}$ entries.
A straightforward algorithm

Algorithm **MatMult**\((A, B)\)

1. \(n \leftarrow \) number of rows of \(A\)
2. for \(i \leftarrow 1\) to \(n\) do
3. \hspace{1em} for \(j \leftarrow 1\) to \(n\) do
4. \hspace{2em} \(c_{ij} \leftarrow 0\)
5. \hspace{1em} for \(k \leftarrow 1\) to \(n\) do
6. \hspace{2em} \(c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}\)
7. return \(C = (c_{ij})_{1 \leq i, j \leq n}\)

Requires

\[\Theta(n^3)\]

arithmetic operations (additions and multiplications).
A naïve divide-and-conquer algorithm

Observe

If

\[ A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \]  and  \[ B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \]

for \((n/2 \times n/2)\)-submatrices \(A_{ij}\) and \(B_{ij}\) then

\[
AB = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}
\]

note: We are assuming \(n\) is a power of 2.
A naïve divide-and-conquer algorithm

Suppose \( i \leq n/2 \) and \( j > n/2 \). Then

\[
    c_{ij} = \sum_{k=1}^{n/2} a_{ik} b_{kj} = \sum_{k=n/2+1}^{n} a_{ik} b_{kj} + n \sum_{k=n/2+1}^{n} a_{ik} b_{kj}
\]

\( \in A_{11} \cup A_{12} \)

\( \in B_{11} \cup B_{12} \)

\( \in A_{21} \cup A_{22} \)

\( \in B_{21} \cup B_{22} \)
A naïve divide-and-conquer algorithm

Suppose \( i \leq \frac{n}{2} \) and \( j > \frac{n}{2} \). Then

\[
c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} = \sum_{k=1}^{n/2} a_{ik} b_{kj} + \sum_{k=n/2+1}^{n} a_{ik} b_{kj}
\]

\[\in A_{11}B_{12}\]  \(\in A_{12}B_{22}\)
A naïve divide-and-conquer algorithm (cont’d)

Assume \( n \) is a power of 2.

**Algorithm** \( \text{D&C-MATMULT}(A, B) \)

1. \( n \leftarrow \) number of rows of \( A \)
2. if \( n = 1 \) then return \((a_{11}b_{11})\)
3. else
4. Let \( A_{ij}, B_{ij} \) (for \( i, j = 1, 2 \)) be \((n/2 \times n/2)\)-submatrices s.th.
   \[
   A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}
   \]
5. Recursively compute \( A_{11}B_{11}, A_{12}B_{21}, A_{11}B_{12}, A_{12}B_{22}, A_{21}B_{11}, A_{22}B_{21}, A_{21}B_{12}, A_{22}B_{22} \)
6. Compute \( C_{11} = A_{11}B_{11} + A_{12}B_{21}, C_{12} = A_{11}B_{12} + A_{12}B_{22}, C_{21} = A_{21}B_{11} + A_{22}B_{21}, C_{22} = A_{21}B_{12} + A_{22}B_{22} \)
7. return \( \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \)
Analysis of D&C-MatMult

\( T(n) \) is the number of operations done by D&C-MatMult.

- Lines 1, 2, 3, 4, 7 require \( \Theta(1) \) arithmetic operations
- Line 5 requires \( 8T(n/2) \) arithmetic operations
- Line 6 requires \( 4(n/2)^2 = \Theta(n^2) \) arithmetic operations.

Remember! Size of matrices is \( \Theta(n^2) \), NOT \( \Theta(n) \)

We get the recurrence

\[
T(n) = 8T(n/2) + \Theta(n^2).
\]

Since \( \log_2(8) = 3 \), the Master Theorem yields

\[
T(n) = \Theta(n^3).
\]
Analysis of D&C-MatMult

$T(n)$ is the number of operations done by D&C-MatMult.

- Lines 1, 2, 3, 4, 7 require $\Theta(1)$ arithmetic operations
- Line 5 requires $8T(n/2)$ arithmetic operations
- Line 6 requires $4(n/2)^2 = \Theta(n^2)$ arithmetic operations.

**Remember!** Size of matrices is $\Theta(n^2)$, NOT $\Theta(n)$

We get the recurrence

$$T(n) = 8T(n/2) + \Theta(n^2).$$

Since $\log_2(8) = 3$, the Master Theorem yields

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

(No improvement over MatMult ... why? CLASS? ...)

ADS (2017/18) – Lecture 4 – slide 11
Strassen’s algorithm (1969)

Assume $n$ is a power of 2.
Let

\[ A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}. \]

We want to compute

\[ AB = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}. \]

Strassen’s algorithm uses a trick in applying Divide-and-Conquer.
Strassen’s algorithm (cont’d)

Let

\[ P_1 = (A_{11} + A_{22})(B_{11} + B_{22}) \]
\[ P_2 = (A_{21} + A_{22})B_{11} \]
\[ P_3 = A_{11}(B_{12} - B_{22}) \]
\[ P_4 = A_{22}(-B_{11} + B_{21}) \]
\[ P_5 = (A_{11} + A_{12})B_{22} \]
\[ P_6 = (-A_{11} + A_{21})(B_{11} + B_{12}) \]
\[ P_7 = (A_{12} - A_{22})(B_{21} + B_{22}) \]

(\(\ast\))
Strassen’s algorithm (cont’d)

Let

\[ P_1 = (A_{11} + A_{22})(B_{11} + B_{22}) \]
\[ P_2 = (A_{21} + A_{22})B_{11} \]
\[ P_3 = A_{11}(B_{12} - B_{22}) \]
\[ P_4 = A_{22}(-B_{11} + B_{21}) \]
\[ P_5 = (A_{11} + A_{12})B_{22} \]
\[ P_6 = (-A_{11} + A_{21})(B_{11} + B_{12}) \]
\[ P_7 = (A_{12} - A_{22})(B_{21} + B_{22}) \]

Then

\[ C_{11} = P_1 + P_4 - P_5 + P_7 \]
\[ C_{12} = P_3 + P_5 \]
\[ C_{21} = P_2 + P_4 \]
\[ C_{22} = P_1 + P_3 - P_2 + P_6 \]
We will check the equation for $C_{11}$ is correct. Strassen’s algorithm computes $C_{11} = P_1 + P_4 - P_5 + P_7$. We have

\[
P_1 = (A_{11} + A_{22})(B_{11} + B_{22}) = A_{11}B_{11} + A_{11}B_{22} + A_{22}B_{11} + A_{22}B_{22}.
\]
\[
P_4 = A_{22}(-B_{11} + B_{21}) = A_{22}B_{21} - A_{22}B_{11}.
\]
\[
P_5 = (A_{11} + A_{12})B_{22} = A_{11}B_{22} + A_{12}B_{22}.
\]
\[
P_7 = (A_{12} - A_{22})(B_{21} + B_{22}) = A_{12}B_{21} + A_{12}B_{22} - A_{22}B_{21} - A_{22}B_{22}.
\]
Checking Strassen’s algorithm - C11

We will check the equation for \( C_{11} \) is correct. Strassen’s algorithm computes \( C_{11} = P1 + P4 - P5 + P7 \). We have

\[
P1 = (A11 + A22)(B11 + B22) \\
= A11B11 + A11B22 + A22B11 + A22B22.
\]

\[
P4 = A22(-B11 + B21) = A22B21 - A22B11.
\]

\[
\]

\[
P7 = (A12 - A22)(B21 + B22) \\
= A12B21 + A12B22 - A22B21 - A22B22.
\]

Then \( P1 + P4 = A11B11 + A11B22 + A22B22 + A22B21 \).
Checking Strassen’s algorithm - $C_{11}$

We will check the equation for $C_{11}$ is correct.
Strassen’s algorithm computes $C_{11} = P1 + P4 - P5 + P7$. We have

\[
\]

\[
P4 = A22(-B11 + B21) = A22B21 - A22B11.
\]

\[
\]

\[
\]

Then $P1 + P4 = A11B11 + A11B22 + A22B22 + A22B21$.
Then $P1 + P4 - P5 = A11B11 + A22B22 + A22B21 - A12B22$. 

ADS (2017/18) – Lecture 4 – slide 14
Checking Strassen’s algorithm - \( C_{11} \)

We will check the equation for \( C_{11} \) is correct.

Strassen’s algorithm computes \( C_{11} = P_1 + P_4 - P_5 + P_7 \). We have

\[
P_1 = (A_{11} + A_{22})(B_{11} + B_{22}) = A_{11}B_{11} + A_{11}B_{22} + A_{22}B_{11} + A_{22}B_{22}.
\]

\[
P_4 = A_{22}(-B_{11} + B_{21}) = A_{22}B_{21} - A_{22}B_{11}.
\]

\[
P_5 = (A_{11} + A_{12})B_{22} = A_{11}B_{22} + A_{12}B_{22}.
\]

\[
P_7 = (A_{12} - A_{22})(B_{21} + B_{22}) = A_{12}B_{21} + A_{12}B_{22} - A_{22}B_{21} - A_{22}B_{22}.
\]

Then \( P_1 + P_4 = A_{11}B_{11} + A_{11}B_{22} + A_{22}B_{22} + A_{22}B_{21} \).

Then \( P_1 + P_4 - P_5 = A_{11}B_{11} + A_{22}B_{22} + A_{22}B_{21} - A_{12}B_{22} \).

Then \( P_1 + P_4 - P_5 + P_7 = A_{11}B_{11} + A_{12}B_{21} \), which is \( C_{11} \).
Checking Strassen’s algorithm - $C_{11}$

We will check the equation for $C_{11}$ is correct. Strassen’s algorithm computes $C_{11} = P1 + P4 - P5 + P7$. We have

\[
\]

\[
P4 = A22(-B11 + B21) = A22B21 - A22B11.
\]

\[
\]

\[
\]

Then $P1 + P4 = A11B11 + A11B22 + A22B22 + A22B21$.

Then $P1 + P4 - P5 = A11B11 + A22B22 + A22B21 - A12B22$.

Then $P1 + P4 - P5 + P7 = A11B11 + A12B21$, which is $C_{11}$.

**homework:** check other 3 equations.
Strassen’s algorithm (cont’d)

Crucial Observation

Only 7 multiplications of \( (n/2 \times n/2) \)-matrices are needed to compute \( AB \).

Algorithm \( \text{STRASSEN}(A, B) \)

1. \( n \leftarrow \text{number of rows of } A \)
2. if \( n = 1 \) then return \( (a_{11} b_{11}) \)
3. else
4. Determine \( A_{ij} \) and \( B_{ij} \) for \( i, j = 1, 2 \) (as before)
5. Compute \( P_1, \ldots, P_7 \) as in (*)
6. Compute \( C_{11}, C_{12}, C_{21}, C_{22} \) as in (**)
7. \[ \text{return } \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \]
Analysis of Strassen’s algorithm

Let $T(n)$ be the number of arithmetic operations performed by \texttt{Strassen}.

- Lines 1–4 and 7 require $\Theta(1)$ arithmetic operations
- Line 5 requires $7T(n/2) + \Theta(n^2)$ arithmetic operations
- Line 6 requires $\Theta(n^2)$ arithmetic operations. remember.

We get the recurrence

$$T(n) = 7T(n/2) + \Theta(n^2).$$

Since $\log_2(7) \approx 2.807 > 2$, the Master Theorem yields

$$T(n) = \Theta(n^{\log_2(7)}).$$
Breakthroughs on matrix multiplication

- Coppersmith & Winograd (1987) came up with an improved algorithm with running time of

\[ \Theta(n^{2.376}) \].

- ... many years of silence ...

- Then in his 2010 PhD thesis, Andrew Stothers from the School of Maths, at the University of Edinburgh got an algorithm with \( \Theta(n^c) \) for \( c < 2.3737 \) ...
  - \( \Rightarrow \) Coppersmith/Winograd not optimal.
  - But Stothers didn’t publish.

- In December 2011, Virginia Vassilevska Williams of Stanford, came up with a \( \Theta(n^c) \) algorithm, for \( c < 2.3727 \)
  (partly, but not only, making use of some of Stothers’ ideas)
Remarks on Matrix Multiplication

- In practice, the “school” MatMult algorithm tends to outperform Strassen’s algorithm, unless the matrices are huge.
- The best known lower bound for matrix multiplication is \( \Omega(n^2) \).

This is a trivial lower bound (need to look at all entries of each matrix). Amazingly, \( \Omega(n^2) \) is believed to be “the truth”!

Open problem: Can we find a \( O(n^{2+o(1)}) \)-algorithm for Matrix Multiplication of \( n \times n \) matrices?
Reading Assignment

[CLRS] (3rd ed) Section 4.5 “The Master method for solving recurrences” (Section 4.3 “Using the Master method” of [CLRS], 2nd ed)
[CLRS] (3rd ed) Section 4.2 (Section 28.2 of [CLRS], 2nd ed)

Problems

1. Exercise 4.5-2 of [CLRS] (3rd ed) Exercise 4.3-2 of [CLRS], 2nd ed.
2. Exercise 4.2-1 of [CLRS], 3rd ed. Exercise 28.2-1 [CLRS], 2nd ed.
3. Week 3 tutorial sheet :-)}