Strassen's Matrix Multiplication
Andreas Klappenecker

[partially based on slides by Prof. Welch]
Matrix Multiplication

Consider two $n \times n$ matrices $A$ and $B$

Recall that the matrix product $C = AB$ of two $n \times n$ matrices is defined as the $n \times n$ matrix that has the coefficient

$$c_{kl} = \sum_m a_{km} b_{ml}$$

in row $k$ and column $l$, where the sum ranges over the integers from 1 to $n$; the scalar product of the $k^{th}$ row of $a$ with the $l^{th}$ column of $B$.

The straightforward algorithm uses $O(n^3)$ scalar operations.

Can we do better?
Idea: Use Divide and Conquer

The divide and conquer paradigm is important general technique for designing algorithms. In general, it follows the steps:

- divide the problem into subproblems
- recursively solve the subproblems
- combine solutions to subproblems to get solution to original problem
Divide-and-Conquer

Let write the product $A \times B = C$ as follows:

$$\begin{array}{c|c}
A_0 & A_1 \\
\hline
A_2 & A_3 \\
\end{array} \times \begin{array}{c|c}
B_0 & B_1 \\
\hline
B_2 & B_3 \\
\end{array} = \begin{array}{c|c}
A_0 \times B_0 + A_1 \times B_2 & A_0 \times B_1 + A_1 \times B_3 \\
\hline
A_2 \times B_0 + A_3 \times B_2 & A_2 \times B_1 + A_3 \times B_3 \\
\end{array}$$

- Divide matrices $A$ and $B$ into four submatrices each
- We have 8 smaller matrix multiplications and 4 additions. Is it faster?
Divide-and-Conquer

Let us investigate this recursive version of the matrix multiplication.

Since we divide $A$, $B$ and $C$ into 4 submatrices each, we can compute the resulting matrix $C$ by

- 8 matrix multiplications on the submatrices of $A$ and $B$,
- plus $\Theta(n^2)$ scalar operations
Divide-and-Conquer

• Running time of recursive version of straightforward algorithm is
  
  \[ T(n) = 8T(n/2) + \Theta(n^2) \]
  
  \[ T(2) = \Theta(1) \]

  where \( T(n) \) is running time on an \( n \times n \) matrix

• Master theorem gives us:
  
  \[ T(n) = \Theta(n^3) \]

• Can we do fewer recursive calls (fewer multiplications of the \( n/2 \times n/2 \) submatrices)?
Strassen’s Matrix Multiplication

\[ \begin{array}{cc} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array} \times \begin{array}{cc} B_{11} & B_{12} \\ B_{21} & B_{22} \end{array} = \begin{array}{cc} C_{11} & C_{12} \\ C_{21} & C_{22} \end{array} \]

\[ \begin{align*}
P_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
P_2 &= (A_{21} + A_{22}) * B_{11} \\
P_3 &= A_{11} * (B_{12} - B_{22}) \\
P_4 &= A_{22} * (B_{21} - B_{11}) \\
P_5 &= (A_{11} + A_{12}) * B_{22} \\
P_6 &= (A_{21} - A_{11}) * (B_{11} + B_{12}) \\
P_7 &= (A_{12} - A_{22}) * (B_{21} + B_{22}) \end{align*} \]
Strassen's Matrix Multiplication

- Strassen found a way to get all the required information with only 7 matrix multiplications, instead of 8.

- Recurrence for new algorithm is
  
  \[ T(n) = 7T(n/2) + \Theta(n^2) \]
Solving the Recurrence Relation

Applying the Master Theorem to

\[ T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n) \]

with \( a = 7 \), \( b = 2 \), and \( f(n) = \Theta(n^2) \).

Since \( f(n) = O\left(n^{\log_b(a)-\epsilon}\right) = O\left(n^{\log_2(7)-\epsilon}\right) \),

case a) applies and we get

\[ T(n) = \Theta\left(n^{\log_b(a)}\right) = \Theta\left(n^{\log_2(7)}\right) = O(n^{2.81}). \]
Discussion of Strassen's Algorithm

- Not always practical
  - constant factor is larger than for naïve method
  - specially designed methods are better on sparse matrices
  - issues of numerical (in)stability
  - recursion uses lots of space

- Not the fastest known method
  - Fastest known is $O(n^{2.3727})$ [Winograd–Coppersmith algorithm improved by V. Williams]
  - Best known lower bound is $\Omega(n^2)$