COMP251: Divide-and-Conquer (2)

Jérôme Waldispühl
School of Computer Science
McGill University

Based on (Kleinberg & Tardos, 2005) & (Cormen et al., 2009)
How to determine the running time of a divide-and-conquer algorithm?

The Master Theorem
Recursive definition

$T(n)$: execution time on an input of size $n$.

Merge Sort: $T(n) = 2 \cdot T \left( \frac{n}{2} \right) + n$

Binary Search: $T(n) = T \left( \frac{n}{2} \right) + 1$

Karatsuba: $T(n) = 3 \cdot T \left( \frac{n}{2} \right) + n$

Number of recursive calls

Time to merge

Size of sub-problems
Master method

Goal. Recipe for solving common divide-and-conquer recurrences:

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

Terms.
- \(a \geq 1\) is the number of subproblems.
- \(b > 0\) is the factor by which the subproblem size decreases.
- \(f(n)\) = work to divide/merge subproblems.

Recursion tree.
- \(k = \log_b n\) levels.
- \(a^i\) = number of subproblems at level \(i\).
- \(n / b^i\) = size of subproblem at level \(i\).
Case 1: total cost dominated by cost of leaves

**Ex 1.** If $T(n)$ satisfies $T(n) = 3 \, T(n/2) + n$, with $T(1) = 1$, then $T(n) = \Theta(n^{\log_3 3})$.
Case 2: total cost evenly distributed among levels

**Ex 2.** If \( T(n) \) satisfies \( T(n) = 2 \, T(n / 2) + n \), with \( T(1) = 1 \), then \( T(n) = \Theta(n \log n) \).

\[
\begin{align*}
T(n) & = 2 \, T(n / 2) + n \\
& = 2 \, (2 \, T(n / 4) + n / 2) + n \\
& = 2^2 \, (2 \, T(n / 8) + n / 4) + n / 2 \\
& \vdots \\
& = 2^k \, (T(1) + n / 2^k) + n / 2^k \\
& = 2^k \, T(1) + n \\
& = 2^k \, \log_2 n \\
& = n \, (\log_2 n + 1)
\end{align*}
\]
Case 3: total cost dominated by cost of root

Ex 3. If $T(n)$ satisfies $T(n) = 3T(n/4) + n^5$, with $T(1) = 1$, then $T(n) = \Theta(n^5)$.

\[
\begin{align*}
T(n) &= 3T(n/4) + n^5 \\
T(n/4) &= 3T(n/4) + n^5 \\
T(n/16) &= 3T(n/16) + n^5 \\
&\vdots \\
T(1) &= 3T(1) + n^5 \\
\end{align*}
\]

$$r = 3/4^5 < 1 \quad n^5 \leq T(n) \leq (1 + r + r^2 + r^3 + \ldots) n^5 \leq \frac{1}{1-r} n^5$$
Master theorem

**Master theorem.** Suppose that $T(n)$ is a function on the nonnegative integers that satisfies the recurrence

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

where $n/b$ means either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Let $k = \log_b a$. Then,

**Case 1.** If $f(n) = O(n^{k-\varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^k)$.

**Ex.** $T(n) = 3 \cdot T(n/2) + n$.

- $a = 3, \ b = 2, \ f(n) = n, \ k = \log_2 3$.
- $T(n) = \Theta(n^{\log_2 3})$.

*The formula works with $\varepsilon = \log_2 3 - 1 > 0$.*

$$f(n) = n = O\left(n^{\log_2 3 - (\log_2 3 - 1)}\right)$$
Master theorem

**Master theorem.** Suppose that $T(n)$ is a function on the nonnegative integers that satisfies the recurrence

$$T(n) = a \ T\left(\frac{n}{b}\right) + f(n)$$

where $n/b$ means either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Let $k = \log_b a$. Then,

**Case 2.** If $f(n) = \Theta(n^k \log^p n)$, then $T(n) = \Theta(n^k \log^{p+1} n)$.

**Ex.** $T(n) = 2 \ T(n/2) + \Theta(n \log n)$.

- $a = 2$, $b = 2$, $f(n) = n \log n \ k = \log_2 2 = 1$, $p = 1$.
- $T(n) = \Theta(n \log^2 n)$.

$$f(n) = \Theta(n \log n) = \Theta(n^{\log_2 2} \log n)$$
Master theorem

**Master theorem.** Suppose that $T(n)$ is a function on the nonnegative integers that satisfies the recurrence

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

where $n/b$ means either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Let $k = \log_b a$. Then,

**Case 3.** If $f(n) = \Omega(n^k + \varepsilon)$ for some constant $\varepsilon > 0$ and if $a f(n/b) \leq c f(n)$ for some constant $c < 1$ and all sufficiently large $n$, then $T(n) = \Theta(f(n))$.

**Ex.** $T(n) = 3 \cdot T(n/4) + n^5$.
- $a = 3$, $b = 4$, $f(n) = n^5$, $k = \log_4 3$.
- $T(n) = \Theta(n^5)$.

1st property satisfied with $\varepsilon = 1 - \log_4 3$

$$f(n) = n^5 = \Omega(n^{\log_4 3 + (1 - \log_4 3)})$$

2nd property satisfied with $c = \frac{3}{4}$

$$3 \cdot \left(\frac{n}{4}\right)^5 \leq c \cdot n^5$$
Master theorem

**Master theorem.** Suppose that $T(n)$ is a function on the nonnegative integers that satisfies the recurrence

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

where $n/b$ means either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Let $k = \log_b a$. Then,

**Case 1.** If $f(n) = O(n^{k-\varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^k)$.

**Case 2.** If $f(n) = \Theta(n^k \log^p n)$, then $T(n) = \Theta(n^k \log^{p+1} n)$.

**Case 3.** If $f(n) = \Omega(n^{k+\varepsilon})$ for some constant $\varepsilon > 0$ and if $a f(n/b) \leq c f(n)$ for some constant $c < 1$ and all sufficiently large $n$, then $T(n) = \Theta(f(n))$.

**Pf sketch.**

- Use recursion tree to sum up terms (assuming $n$ is an exact power of $b$).
- Three cases for geometric series.
- Deal with floors and ceilings.
Applications

\[ T(n) = 3 \times T(n/2) + n^2 \]
\[ \Rightarrow T(n) = \Theta(n^2) \quad \text{(case 3)} \]

\[ T(n) = T(n/2) + 2^n \]
\[ \Rightarrow T(n) = \Theta(2^n) \quad \text{(case 3)} \]

\[ T(n) = 16 \times T(n/4) + n \]
\[ \Rightarrow T(n) = \Theta(n^2) \quad \text{(case 1)} \]

\[ T(n) = 2 \times T(n/2) + n \log n \]
\[ \Rightarrow T(n) = n \log^2 n \quad \text{(case 2)} \]

\[ T(n) = 2^n \times T(n/2) + n^n \]
\[ \Rightarrow \text{Does not apply!!} \]

\[ k = \log_2 1 = 0; f(n) = 2^n \]
\[ 2^n = \Omega(n^{0+\log 2}) \]
\[ 1 \cdot 2^{\frac{n}{2}} \leq \frac{1}{2} \cdot 2^n \]

\[ k = \log_2 3; f(n) = n^2 \]
\[ n^2 = \Omega(n^\log_2^3 + (2-\log_2 3)) \]
\[ 3 \cdot \left(\frac{n}{2}\right)^2 \leq \frac{3}{4} \cdot n^2 \]

\[ k = \log_4 16 = 2; f(n) = n \]
\[ n = O(n^{2-1}) \]

\[ k = \log_2 2 = 1; f(n) = n \log n \]
\[ n \log n = \Theta(n^{1 \log n}) \]
Akra-Bazzi theorem

Desiderata. Generalizes master theorem to divide-and-conquer algorithms where subproblems have substantially different sizes.

Theorem. [Akra-Bazzi] Given constants $a_i > 0$ and $0 < b_i \leq 1$, functions $h_i(n) = O(n / \log^2 n)$ and $g(n) = O(n^c)$, if the function $T(n)$ satisfies the recurrence:

$$T(n) = \sum_{i=1}^{k} a_i T(b_i n + h_i(n)) + g(n)$$

Then $T(n) = \Theta \left( n^p \left( 1 + \int_{1}^{n} \frac{g(u)}{u^{p+1}} du \right) \right)$ where $p$ satisfies $\sum_{i=1}^{k} a_i b_i^p = 1$.

Ex. $T(n) = 7/4 \cdot \left( T(\lfloor n / 2 \rfloor) + T(\lfloor 3/4 n \rfloor) \right) + n^2$.

- $a_1 = 7/4, \quad b_1 = 1/2, \quad a_2 = 1, \quad b_2 = 3/4 \quad \Rightarrow \quad p = 2$.
- $h_1(n) = \lfloor 1/2 n \rfloor - 1/2 n, \quad h_2(n) = \lfloor 3/4 n \rfloor - 3/4 n$.
- $g(n) = n^2 \quad \Rightarrow \quad T(n) = \Theta(n^2 \log n)$. 
Another Divide-and-Conquer Algorithms
Dot product

**Dot product.** Given two length $n$ vectors $a$ and $b$, compute $c = a \cdot b$.

**Grade-school.** $\Theta(n)$ arithmetic operations.

\[
a \cdot b = \sum_{i=1}^{n} a_i b_i
\]

\[
a = [0.70 \ 0.20 \ 0.10] \\
b = [0.30 \ 0.40 \ 0.30]
\]

\[
a \cdot b = (0.70 \times 0.30) + (0.20 \times 0.40) + (0.10 \times 0.30) = 0.32
\]

**Remark.** Grade-school dot product algorithm is asymptotically optimal.
Matrix multiplication

Matrix multiplication. Given two $n$-by-$n$ matrices $A$ and $B$, compute $C = AB$.

Grade-school. $\Theta(n^3)$ arithmetic operations.

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

\[
\begin{bmatrix}
c_{11} & c_{12} & \cdots & c_{1n} \\
c_{21} & c_{22} & \cdots & c_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
c_{n1} & c_{n2} & \cdots & c_{nn}
\end{bmatrix}
= 
\begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}
\times
\begin{bmatrix}
b_{11} & b_{12} & \cdots & b_{1n} \\
b_{21} & b_{22} & \cdots & b_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
b_{n1} & b_{n2} & \cdots & b_{nn}
\end{bmatrix}
\]

\[
\begin{bmatrix}
.59 & .32 & .41 \\
.31 & .36 & .25 \\
.45 & .31 & .42
\end{bmatrix}
= 
\begin{bmatrix}
.70 & .20 & .10 \\
.30 & .60 & .10 \\
.50 & .10 & .40
\end{bmatrix}
\times
\begin{bmatrix}
.80 & .30 & .50 \\
.10 & .40 & .10 \\
.10 & .30 & .40
\end{bmatrix}
\]

Q. Is grade-school matrix multiplication algorithm asymptotically optimal?
Block matrix multiplication

\[
C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21} = \begin{bmatrix} 0 & 1 \\ 4 & 5 \end{bmatrix} \times \begin{bmatrix} 16 & 17 \\ 20 & 21 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix} \times \begin{bmatrix} 24 & 25 \\ 28 & 29 \end{bmatrix} = \begin{bmatrix} 152 & 158 \\ 504 & 526 \end{bmatrix}
\]
Matrix multiplication: warmup

To multiply two $n$-by-$n$ matrices $A$ and $B$:

- Divide: partition $A$ and $B$ into $\frac{1}{2}n$-by-$\frac{1}{2}n$ blocks.
- Conquer: multiply 8 pairs of $\frac{1}{2}n$-by-$\frac{1}{2}n$ matrices, recursively.
- Combine: add appropriate products using 4 matrix additions.

$$
\begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix} =
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix} \times
\begin{bmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{bmatrix}
$$

$$
C_{11} = (A_{11} \times B_{11}) + (A_{12} \times B_{21})
C_{12} = (A_{11} \times B_{12}) + (A_{12} \times B_{22})
C_{21} = (A_{21} \times B_{11}) + (A_{22} \times B_{21})
C_{22} = (A_{21} \times B_{12}) + (A_{22} \times B_{22})
$$

Running time. Apply case 1 of Master Theorem.

$$
T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2) \quad \Rightarrow \quad T(n) = \Theta(n^3)
$$
Strassen's trick

**Key idea.** multiply 2-by-2 blocks with only 7 multiplications.
(plus 11 additions and 7 subtractions)

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

\[
\begin{align*}
C_{11} &= P_5 + P_4 - P_2 + P_6 \\
C_{12} &= P_1 + P_2 \\
C_{21} &= P_3 + P_4 \\
C_{22} &= P_1 + P_5 - P_3 - P_7
\end{align*}
\]

**Pf.** $C_{12} = P_1 + P_2$

\[
\begin{align*}
&= A_{11} \times (B_{12} - B_{22}) + (A_{11} + A_{12}) \times B_{22} \\
&= A_{11} \times B_{12} + A_{12} \times B_{22}. \quad \checkmark
\end{align*}
\]
**Strassen's algorithm**

\[ \text{STRASSEN}(n, A, B) \]

**IF** \((n = 1)\) **RETURN** \(A \times B.\)

Partition \(A\) and \(B\) into 2-by-2 block matrices.

\[
P_1 \leftarrow \text{STRASSEN}(n / 2, A_{11}, (B_{12} - B_{22})).
\]

\[
P_2 \leftarrow \text{STRASSEN}(n / 2, (A_{11} + A_{12}), B_{22}).
\]

\[
P_3 \leftarrow \text{STRASSEN}(n / 2, (A_{21} + A_{22}), B_{11}).
\]

\[
P_4 \leftarrow \text{STRASSEN}(n / 2, A_{22}, (B_{21} - B_{11})).
\]

\[
P_5 \leftarrow \text{STRASSEN}(n / 2, (A_{11} + A_{22}) \times (B_{11} + B_{22})).
\]

\[
P_6 \leftarrow \text{STRASSEN}(n / 2, (A_{12} - A_{22}) \times (B_{21} + B_{22})).
\]

\[
P_7 \leftarrow \text{STRASSEN}(n / 2, (A_{11} - A_{21}) \times (B_{11} + B_{12})).
\]

\[
C_{11} = P_5 + P_4 - P_2 + P_6.
\]

\[
C_{12} = P_1 + P_2.
\]

\[
C_{21} = P_3 + P_4.
\]

\[
C_{22} = P_1 + P_5 - P_3 - P_7.
\]

**RETURN** \(C.\)
Analysis of Strassen's algorithm

**Theorem.** Strassen's algorithm requires $O(n^{2.81})$ arithmetic operations to multiply two $n$-by-$n$ matrices.

**Pf.** Apply case 1 of the master theorem to the recurrence:

$$T(n) = 7T(n/2) + \Theta(n^2) \Rightarrow T(n) = \Theta(n^{\log_2 7}) = O(n^{2.81})$$

**Q.** What if $n$ is not a power of 2?

**A.** Could pad matrices with zeros.

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 4 & 5 & 6 & 0 \\ 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 10 & 11 & 12 & 0 \\ 13 & 14 & 15 & 0 \\ 16 & 17 & 18 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 84 & 90 & 96 & 0 \\ 201 & 216 & 231 & 0 \\ 318 & 342 & 366 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
Strassen's algorithm: practice

Implementation issues.

• Sparsity.
• Caching effects.
• Numerical stability.
• Odd matrix dimensions.
• Crossover to classical algorithm when $n$ is "small".

Common misperception. “Strassen is only a theoretical curiosity.”

• Apple reports 8x speedup on G4 Velocity Engine when $n \approx 2,048$.
• Range of instances where it's useful is a subject of controversy.
Linear algebra reductions

**Matrix multiplication.** Given two $n$-by-$n$ matrices, compute their product.

<table>
<thead>
<tr>
<th>problem</th>
<th>linear algebra</th>
<th>order of growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix multiplication</td>
<td>$A \times B$</td>
<td>$\Theta(MM(n))$</td>
</tr>
<tr>
<td>matrix inversion</td>
<td>$A^{-1}$</td>
<td>$\Theta(MM(n))$</td>
</tr>
<tr>
<td>determinant</td>
<td>$</td>
<td>A</td>
</tr>
<tr>
<td>system of linear equations</td>
<td>$Ax = b$</td>
<td>$\Theta(MM(n))$</td>
</tr>
<tr>
<td>LU decomposition</td>
<td>$A = LU$</td>
<td>$\Theta(MM(n))$</td>
</tr>
<tr>
<td>least squares</td>
<td>$\min</td>
<td></td>
</tr>
</tbody>
</table>

numerical linear algebra problems with the same complexity as matrix multiplication
Fast matrix multiplication: theory

Q. Multiply two 2-by-2 matrices with 7 scalar multiplications?
A. Yes! [Strassen 1969] \( \Theta(n^\log_2 7) = O(n^{2.807}) \)

Q. Multiply two 2-by-2 matrices with 6 scalar multiplications?
A. Impossible. [Hopcroft and Kerr 1971] \( \Theta(n^{\log_2 6}) = O(n^{2.59}) \)

Q. Multiply two 3-by-3 matrices with 21 scalar multiplications?
A. Unknown. \( \Theta(n^{\log_3 21}) = O(n^{2.77}) \)

Begun, the decimal wars have. [Pan, Bini et al, Schönhage, ...]
- Two 20-by-20 matrices with 4,460 scalar multiplications. \( O(n^{2.805}) \)
- Two 48-by-48 matrices with 47,217 scalar multiplications. \( O(n^{2.7801}) \)
- A year later. \( O(n^{2.7799}) \)
- December 1979. \( O(n^{2.521813}) \)
- January 1980. \( O(n^{2.521801}) \)
### History of asymptotic complexity of matrix multiplication

<table>
<thead>
<tr>
<th>year</th>
<th>algorithm</th>
<th>order of growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>brute force</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>1969</td>
<td>Strassen</td>
<td>$O(n^{2.808})$</td>
</tr>
<tr>
<td>1978</td>
<td>Pan</td>
<td>$O(n^{2.796})$</td>
</tr>
<tr>
<td>1979</td>
<td>Bini</td>
<td>$O(n^{2.780})$</td>
</tr>
<tr>
<td>1981</td>
<td>Schönhage</td>
<td>$O(n^{2.522})$</td>
</tr>
<tr>
<td>1982</td>
<td>Romani</td>
<td>$O(n^{2.517})$</td>
</tr>
<tr>
<td>1982</td>
<td>Coppersmith-Winograd</td>
<td>$O(n^{2.496})$</td>
</tr>
<tr>
<td>1986</td>
<td>Strassen</td>
<td>$O(n^{2.479})$</td>
</tr>
<tr>
<td>1989</td>
<td>Coppersmith-Winograd</td>
<td>$O(n^{2.376})$</td>
</tr>
<tr>
<td>2010</td>
<td>Strother</td>
<td>$O(n^{2.3737})$</td>
</tr>
<tr>
<td>2011</td>
<td>Williams</td>
<td>$O(n^{2.3727})$</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>$O(n^{2+\varepsilon})$</td>
</tr>
</tbody>
</table>

The table above summarizes the history of asymptotic complexity of matrix multiplication, showing the year of discovery, the algorithm used, and the order of growth of the number of floating-point operations required to multiply two $n$-by-$n$ matrices.