Divide-and-conquer algorithm

**Idea:**

$n \times n$ matrix = $2 \times 2$ matrix of $(n/2) \times (n/2)$ submatrices:

\[
\begin{bmatrix}
  r & s \\
  t & u
\end{bmatrix} =
\begin{bmatrix}
  a & b \\
  c & d
\end{bmatrix} \cdot
\begin{bmatrix}
  e & f \\
  g & h
\end{bmatrix}
\]

\[
C = A \cdot B
\]

\[
\begin{align*}
  r &= ae + bg \\
  s &= af + bh \\
  t &= ce + dh \\
  u &= cf + dg
\end{align*}
\]

\[
\begin{align*}
  &\text{recursive} & 8 \text{ mults of } (n/2) \times (n/2) \text{ submatrices} \\
  &\text{8 mults of } (n/2) \times (n/2) \text{ submatrices} & 4 \text{ adds of } (n/2) \times (n/2) \text{ submatrices}
\end{align*}
\]

© 2001–4 by Charles E. Leiserson
Analysis of D&C algorithm

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

- \# submatrices
- submatrix size
- work adding submatrices
Analysis of D&C algorithm

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

# submatrices \quad work adding submatrices

submatrix size

\[ n^{\log_b a} = n^{\log_8 8} = n^3 \Rightarrow \text{CASE 1} \Rightarrow T(n) = \Theta(n^3). \]
Analysis of D&C algorithm

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

- \# submatrices
- submatrix size
- work adding submatrices

\[ n^{\log_b a} = n^{\log_2 8} = n^3 \Rightarrow \text{CASE 1} \Rightarrow T(n) = \Theta(n^3). \]

No better than the ordinary algorithm.
Strassen’s idea

• Multiply $2 \times 2$ matrices with only 7 recursive mults.
Strassen’s idea

- Multiply $2 \times 2$ matrices with only 7 recursive mults.

\[
\begin{align*}
P_1 &= a \cdot (f - h) \\
P_2 &= (a + b) \cdot h \\
P_3 &= (c + d) \cdot e \\
P_4 &= d \cdot (g - e) \\
P_5 &= (a + d) \cdot (e + h) \\
P_6 &= (b - d) \cdot (g + h) \\
P_7 &= (a - c) \cdot (e + f)
\end{align*}
\]
Strassen’s idea

- Multiply $2 \times 2$ matrices with only 7 recursive mults.

$P_1 = a \cdot (f - h)$
$P_2 = (a + b) \cdot h$
$P_3 = (c + d) \cdot e$
$P_4 = d \cdot (g - e)$
$P_5 = (a + d) \cdot (e + h)$
$P_6 = (b - d) \cdot (g + h)$
$P_7 = (a - c) \cdot (e + f)$

$r = P_5 + P_4 - P_2 + P_6$
$s = P_1 + P_2$
$t = P_3 + P_4$
$u = P_5 + P_1 - P_3 - P_7$
Strassen’s idea

- Multiply $2 \times 2$ matrices with only 7 recursive mults.

$$P_1 = a \cdot (f - h)$$
$$P_2 = (a + b) \cdot h$$
$$P_3 = (c + d) \cdot e$$
$$P_4 = d \cdot (g - e)$$
$$P_5 = (a + d) \cdot (e + h)$$
$$P_6 = (b - d) \cdot (g + h)$$
$$P_7 = (a - c) \cdot (e + f)$$

$$r = P_5 + P_4 - P_2 + P_6$$
$$s = P_1 + P_2$$
$$t = P_3 + P_4$$
$$u = P_5 + P_1 - P_3 - P_7$$

7 mults, 18 adds/subs.

Note: No reliance on commutativity of mult!
Strassen’s idea

• Multiply $2 \times 2$ matrices with only 7 recursive mults.

\[
P_1 = a \cdot (f - h)
\]
\[
P_2 = (a + b) \cdot h
\]
\[
P_3 = (c + d) \cdot e
\]
\[
P_4 = d \cdot (g - e)
\]
\[
P_5 = (a + d) \cdot (e + h)
\]
\[
P_6 = (b - d) \cdot (g + h)
\]
\[
P_7 = (a - c) \cdot (e + f)
\]

\[
r = P_5 + P_4 - P_2 + P_6
\]
\[
\quad = (a + d)(e + h) + d(g - e) - (a + b)h + (b - d)(g + h)
\]
\[
\quad = ae + ah + de + dh + dg - de - ah - bh + bg + bh - dg - dh
\]
\[
\quad = ae + bg
\]
Strassen’s algorithm

1. **Divide:** Partition $A$ and $B$ into \((n/2) \times (n/2)\) submatrices. Form terms to be multiplied using $+$ and $-$.

2. **Conquer:** Perform 7 multiplications of \((n/2) \times (n/2)\) submatrices recursively.

3. **Combine:** Form $C$ using $+$ and $-$ on \((n/2) \times (n/2)\) submatrices.
Strassen’s algorithm

1. **Divide:** Partition $A$ and $B$ into $(n/2) \times (n/2)$ submatrices. Form terms to be multiplied using $+$ and $-$.

2. **Conquer:** Perform 7 multiplications of $(n/2) \times (n/2)$ submatrices recursively.

3. **Combine:** Form $C$ using $+$ and $-$ on $(n/2) \times (n/2)$ submatrices.

\[ T(n) = 7 T(n/2) + \Theta(n^2) \]
Analysis of Strassen

\[ T(n) = 7 \cdot T(n/2) + \Theta(n^2) \]
Analysis of Strassen

\[ T(n) = 7 T(n/2) + \Theta(n^2) \]

\[ n^{\log_b a} = n^{\log_2 7} \approx n^{2.81} \Rightarrow \text{CASE 1} \Rightarrow T(n) = \Theta(n^{\log_7 7}). \]
Analysis of Strassen

\[ T(n) = 7 T(n/2) + \Theta(n^2) \]

\[ n^{\log_b a} = n^{\log_2 7} \approx n^{2.81} \Rightarrow \text{CASE 1} \Rightarrow T(n) = \Theta(n^{\lg 7}). \]

The number 2.81 may not seem much smaller than 3, but because the difference is in the exponent, the impact on running time is significant. In fact, Strassen’s algorithm beats the ordinary algorithm on today’s machines for \( n \geq 32 \) or so.
Analysis of Strassen

\[ T(n) = 7 T(n/2) + \Theta(n^2) \]

\[ n^{\log_b a} = n^{\log_2 7} \approx n^{2.81} \Rightarrow \text{CASE 1} \Rightarrow T(n) = \Theta(n^{\log_2 7}). \]

The number 2.81 may not seem much smaller than 3, but because the difference is in the exponent, the impact on running time is significant. In fact, Strassen’s algorithm beats the ordinary algorithm on today’s machines for \( n \geq 32 \) or so.

**Best to date** (of theoretical interest only): \( \Theta(n^{2.376\ldots}) \).
VLSI layout

**Problem:** Embed a complete binary tree with $n$ leaves in a grid using minimal area.
**Problem:** Embed a complete binary tree with \( n \) leaves in a grid using minimal area.

\[
W(n) \\
H(n)
\]
VLSI layout

**Problem:** Embed a complete binary tree with $n$ leaves in a grid using minimal area.

\[ H(n) = H(n/2) + \Theta(1) = \Theta(\lg n) \]
VLSI layout

Problem: Embed a complete binary tree with \( n \) leaves in a grid using minimal area.

\[
H(n) = H(n/2) + \Theta(1) \\
= \Theta(\lg n)
\]

\[
W(n) = 2W(n/2) + \Theta(1) \\
= \Theta(n)
\]
Problem: Embed a complete binary tree with $n$ leaves in a grid using minimal area.

\[
H(n) = H(n/2) + \Theta(1) = \Theta(\lg n)
\]
\[
W(n) = 2W(n/2) + \Theta(1) = \Theta(n)
\]

\[
\text{Area} = \Theta(n \lg n)
\]
H-tree embedding

$L(n)$

$L(n)$
H-tree embedding

\[ L(n) \]

\[ L(n/4) \quad \Theta(1) \quad L(n/4) \]
H-tree embedding

\[ L(n) = 2L(n/4) + \Theta(1) = \Theta(\sqrt{n}) \]

Area = \Theta(n)
Conclusion

• Divide and conquer is just one of several powerful techniques for algorithm design.

• Divide-and-conquer algorithms can be analyzed using recurrences and the master method (so practice this math).

• The divide-and-conquer strategy often leads to efficient algorithms.