Memoizing

Dynamic Programming in a recursive (top-down) framework

- Whenever you compute the value of a subproblem

\[ f \text{ becomes } m_f \text{ plus MEMO table} \]

\[ m_f(x) \]

- if MEMO[x] defined then
- else
- return

Replace all calls to \( f \) by calls to \( m_f \)

- includes
- but not

Greedy Algorithm

A series of locally optimum choices produces a globally optimum solution

What is needed:

- **Greedy choice property** — of all the optimal solutions, at least one

- **Optimal substructure** — as with
Greedy Example

Activity Selection
Have a set of possible activities to do
What is
Have start and finish times
Order by
LIST A B C D E F G
2-5 1-6 3-6 6-8 5-9 10-11 10-12

Sched = { }
Greedy Choice

- Suppose \( Sched \) is a maximal legal schedule that
- Let \( A_i \) be the activity in \( Sched \) with
- Then \( A_i.\text{finish} \)

So \( sched' = \) is also a maximum legal schedule.

Optimal Substructure

- Assume \( Sched \) is maximal legal schedule for
- Let \( A_i \) be the first activity with
- Then \( Sched - \{A_i\} \) is a maximal legal schedule for
Huffman Codes

◊ A E H N R S T
000 001 010 011 100 101 110 111

THE ◊ RAT ◊ SAT ◊ AT ◊ THE ◊ HEN ◊ REST ◊

Frequencies

◊ A E H N R S T

Contribution

Prefix Code

No code word is
11101001001011010011

0 1

0 1

◊ E 0 1 0 1

0 1

0 1

0 1
Greedy Algorithm for Code Generation

Start with \( n \) separate 1-character codes
Merge two codes with lowest frequencies

\[
\begin{array}{c}
\text{Frequency of merged code is} \\
\end{array}
\]

Until

Huffman Code Example

<table>
<thead>
<tr>
<th>7</th>
<th>3</th>
<th>4</th>
<th>3</th>
<th>1</th>
<th>2</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>◊</td>
<td>A</td>
<td>E</td>
<td>H</td>
<td>N</td>
<td>R</td>
<td>S</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7</th>
<th>3</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>◊</td>
<td>A</td>
<td>E</td>
<td>H</td>
<td>S</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7</th>
<th>4</th>
<th>3</th>
<th>3</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>◊</td>
<td>E</td>
<td>H</td>
<td>0 1</td>
<td>T</td>
</tr>
</tbody>
</table>

N   R
Huffman Code Example 2

7  5  4  6
◊ 0  1  E  T
A  S

Huffman Code Example 3

7  9
◊ 0  1
E  A  S  H  N  R

Algorithm Design & Analysis

David Maier
Algorithm Design & Analysis

Huffman Code Example 4

Why does it work?
Greedy Choice

- Greedy choice

\[
\begin{align*}
    f_3 \cdot d_{\text{max}} + f_4 \cdot d_{\text{max}} + f_1 \cdot d_1 + f_2 \cdot d_2 \\
    \cdot d_{\text{max}} + \cdot d_{\text{max}} + \cdot d_1 + \cdot d_2
\end{align*}
\]

\[s_1 \rightarrow s_3\] switch:
old \quad (f_3 \cdot d_{\text{max}} + f_1 \cdot d_1)
new \quad (f_1 \cdot d_{\text{max}} + f_3 \cdot d_1)
subtract
Optimal Substructure

Let $S$ be the set of symbols optimal for $S$ and optimal for $S'$.

$\begin{align*}
    \text{cost}(T) &= \text{cost}(T') \\
    \text{for } S &\quad \text{for } S' \\
    0 &\quad 1 \\
    s_1 &\quad s_2
\end{align*}$