

Memoizing

Dynamic Programming in a recursive (top-down) framework

- Whenever you compute the value of a subproblem

f becomes mf plus MEMO table

```
mf(x)
  if MEMO[x] defined then
  else
  return
```

Replace all calls to f by calls to mf

- includes
- but not

Problem 16-2

Pretty Paragraphs

words w_1, w_2, \dots, w_n
lengths

separated by spaces on lines of length

If w_i, \dots, w_j on a line, then extra spaces

$pad(i, j) = M - \Sigma$

Minimize $(pad(i, j))$ over all lines but the last

$pad(i, n) =$

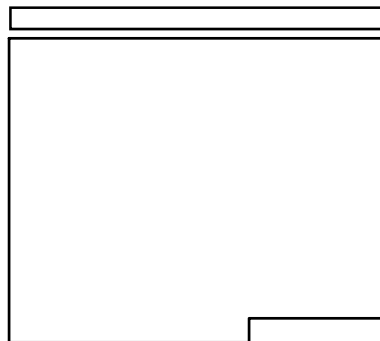
Example

```
this is an pad(1,3)
example group pad(4,5)
of words
```

M =

Recursive Solution

Decompose



first line

rest of words

Calculating Cost

Let $BEST[i, n]$ be the cost of the best layout of words

$$BEST[i, n] = \min_k (\text{pad}(i, k) + BEST[k+1, n])$$

How big is largest p we must consider?

Strategy

Compute

$BEST[n, n]$
 $BEST[n-1, n]$
 $BEST[n-2, n]$
...
 $BEST[1, n]$

helps if $BEST[n+1, n] =$

How long does each one take?

- k ranges over
- $BEST[k+1, n]$
- $\text{pad}(i, k)$ might seem like

Computing $\text{pad}(i,j)$

But notice

$$\begin{aligned}\text{pad}(i,i) &= \\ \text{pad}(i,i+1) &= \text{pad}(i,i) \\ \text{pad}(i,i+2) &= \text{pad}(i,i+1) \\ &\dots \\ \text{pad}(i,p) &= \text{pad}(i,p-1)\end{aligned}$$

so time complexity is $O(n \cdot M)$

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 Choice

- Suppose $Sched$ is a maximal legal schedule that
- Let A_i be the activity in $Sched$ with
- Then $A_i.finish$
- 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

CS 410/584, Algorithm Design & Analysis: Lecture 2

Algorithm Design & Analysis

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

Lecture 2

David Maier

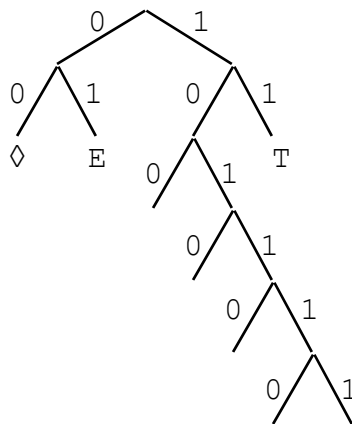
13

Algorithm Design & Analysis

Prefix Code

No code word is

111010010010111010011



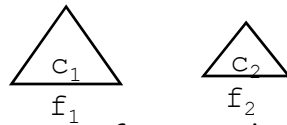
Lecture 2

David Maier

14

Greedy Algorithm for Code Generation

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



Frequency of merged code is

Until

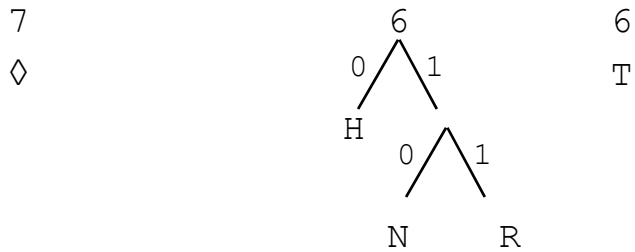
Huffman Code Example

7	3	4	3	1	2	2	6
◇	A	E	H	N	R	S	T

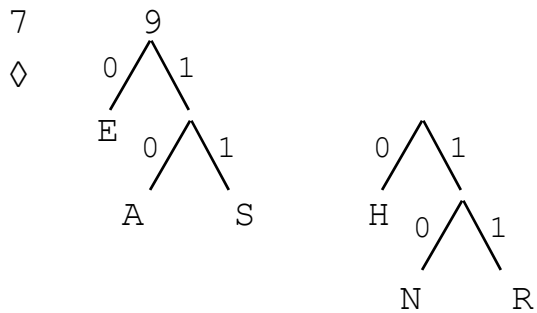
7	3	4	3		2	6
◇	A	E	H		S	T

7		4	3	3		6
◇		E	H	$\begin{matrix} & & 3 & & \\ & 0 & / & \backslash & 1 \\ & N & & & R \end{matrix}$		T

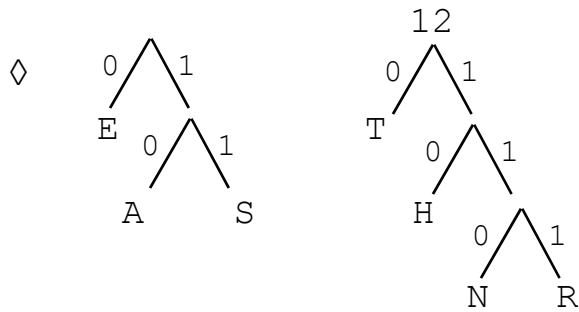
Huffman Code Example 2



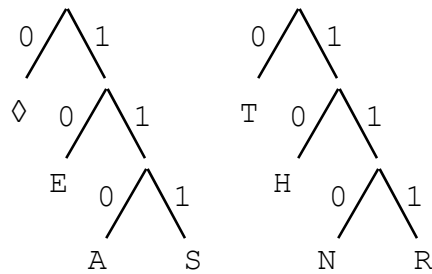
Huffman Code Example 3



Huffman Code Example 4



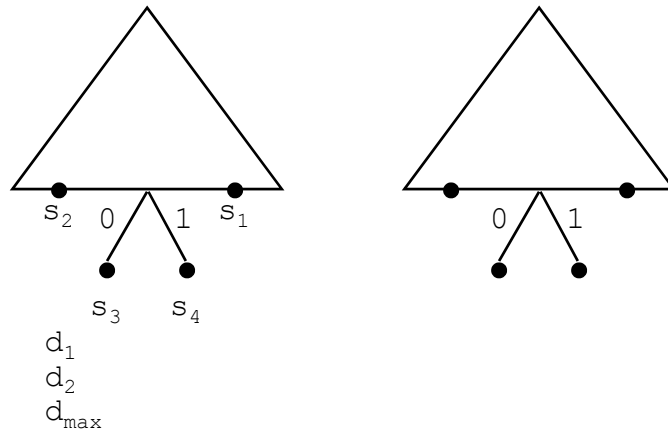
Huffman Code Example 5



Why does it work?

Greedy Choice

- Greedy choice



Greedy Choice 2

$$f_3 \cdot d_{\max} + f_4 \cdot d_{\max} + f_1 \cdot d_1 + f_2 \cdot d_2$$

$$\cdot d_{\max} + \cdot d_{\max} + \cdot d_1 + \cdot d_2$$

s_1-s_3 switch:

old $(f_3 \cdot d_{\max} + f_1 \cdot d_1)$

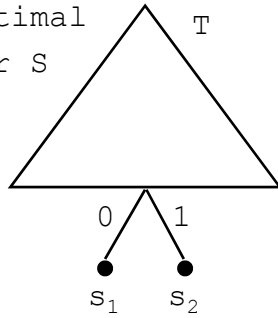
new $(f_1 \cdot d_{\max} + f_3 \cdot d_1)$

subtract

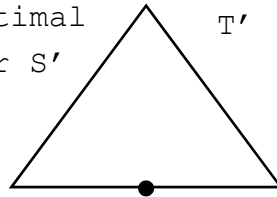
Optimal Substructure

Let S be the set of symbols

optimal
for S



optimal
for S'



$$\text{cost}(T) = \text{cost}(T')$$