

## 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

## 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 = { }

## Schedule Algorithm

Sched  $\leftarrow \emptyset$

```
for each ACT in LIST do  
  if ACT compatible with Sched  
  then Sched  $\leftarrow$ 
```

Complexity:

Correct?

Let LIST =

## 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

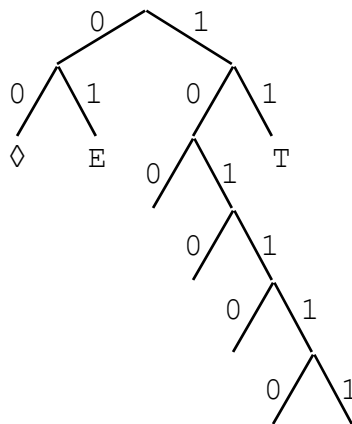
7

Algorithm Design & Analysis

## Prefix Code

No code word is

111010010010111010011



Lecture 2

David Maier

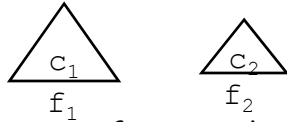
8

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

Algorithm Design & Analysis

## Greedy Algorithm for Code Generation

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



Frequency of merged code is

Until

Lecture 2

David Maier

9

Algorithm Design & Analysis

## 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

---

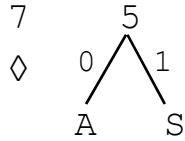
7		4	3	3		6
◇		E	H	0	1	T
				N	R	

Lecture 2

David Maier

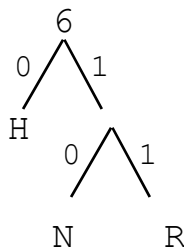
10

Huffman Code Example 2



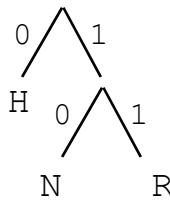
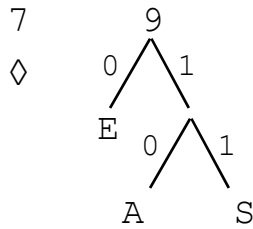
4    6  
E    T

7  
◇

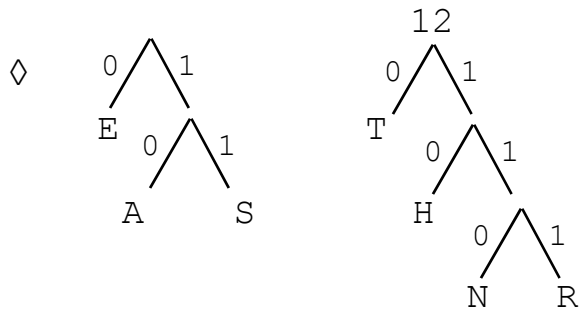


6  
T

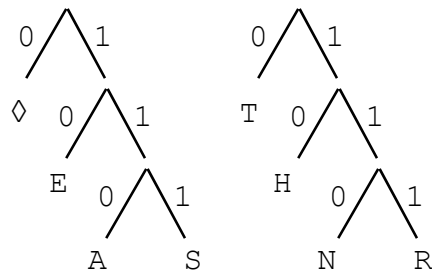
Huffman Code Example 3



Huffman Code Example 4



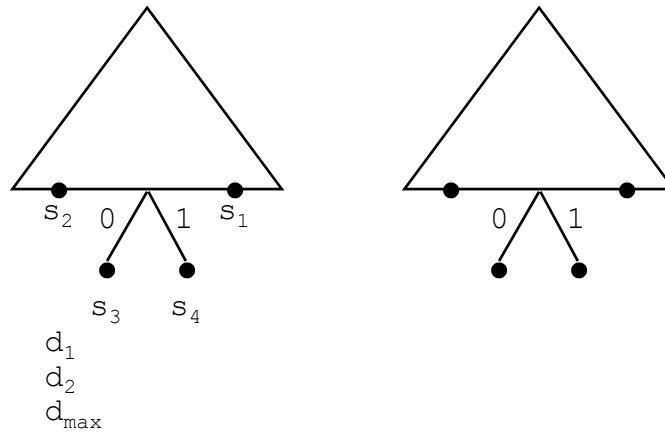
Huffman Code Example 5



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

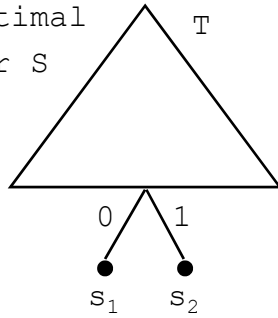
new  $(f_1 \cdot d_{\max} + f_3 \cdot d_1)$

subtract

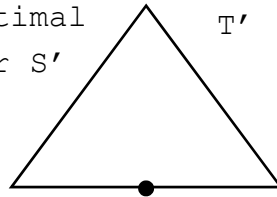
## Optimal Substructure

Let  $S$  be the set of symbols

optimal  
for  $S$



optimal  
for  $S'$



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