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:

- <u>Greedy choice property</u> — of all the optimal solutions, at least one
- <u>Optimal substructure</u> — 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 ⇐ ∅
for each ACT in LIST do
  if ACT compatible with Sched
  then Sched ⇐
```

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_1`.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_1`}` 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

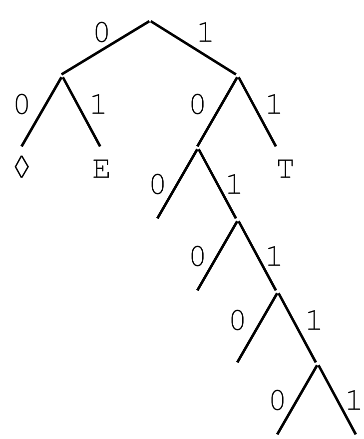
◊	A	E	H	N	R	S	T

Contribution

Prefix Code

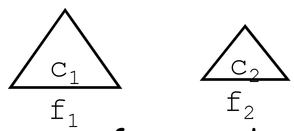
No code word is

11101001001011010011



Greedy Algorithm for Code Generation

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



Frequency of merged code is

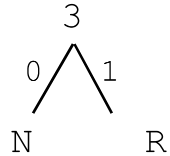
Until

Huffman Code Example

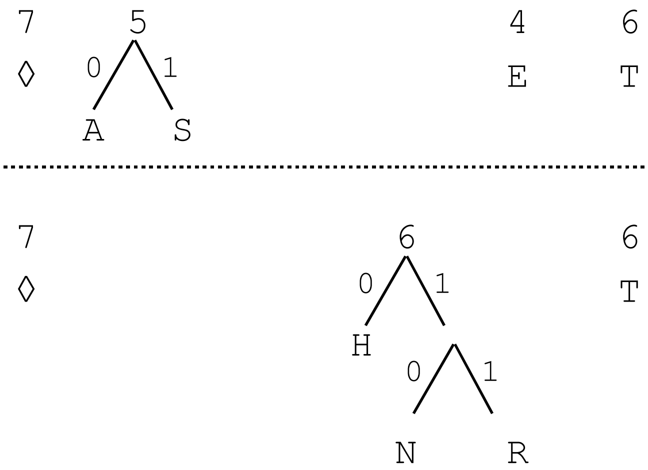
7	3	4	3	1	2	2	6
◊	A	E	H	N	R	S	T

7	3	4	3	2	6
◊	A	E	H	S	T

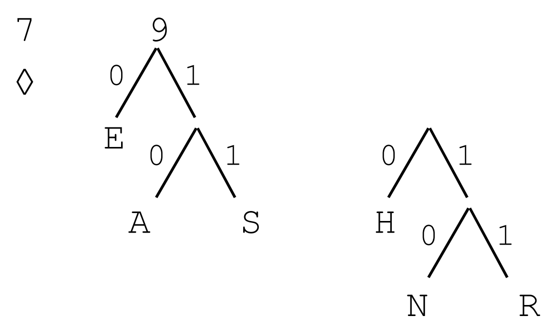
7	4	3	3	6
◊	E	H		T



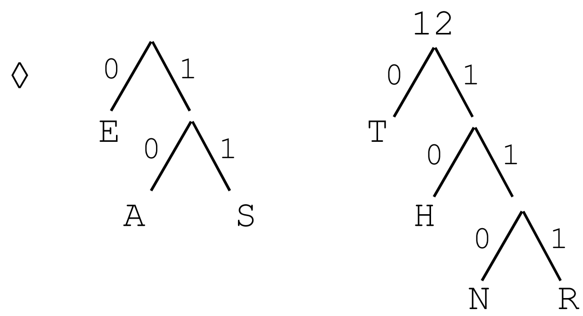
Huffman Code Example 2



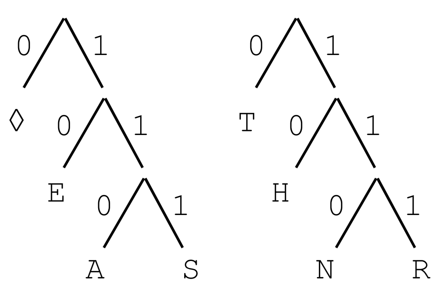
Huffman Code Example 3



Huffman Code Example 4



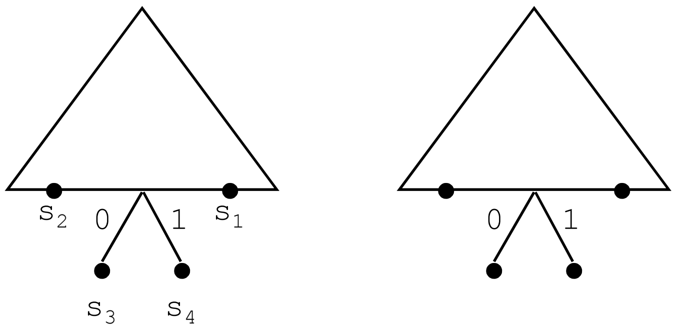
Huffman Code Example 5



Why does it work?

Greedy Choice

- Greedy choice



d_1
d_2
d_{max}

Greedy Choice 2

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

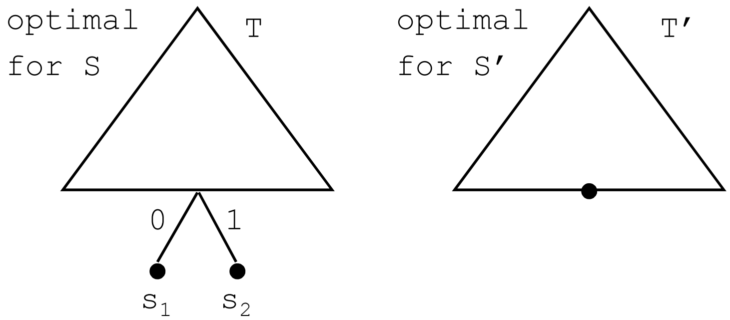
$$\cdot d_{max} + \quad \cdot d_{max} + \quad \cdot d_1 + \quad \cdot d_2$$

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



cost(T) = cost(T')