# CSE 2320 Notes 17: Greedy Algorithms

(Last updated 11/24/06 6:21 PM)

CLRS, 16.1-16.3

CONCEPTS

Commitments are based on *local* decisions:

NO backtracking (as occurred in stack rat-in-a-maze)

NO exhaustive search (as occurred with dynamic programming)

MAIN ISSUE: NOT efficiency . . . Quality of Solution instead

Special situations - exact solution

Prim's MST Dijkstra's shortest path MCP for network flow

More frequently - heuristic (approximation)

Basketball tryout with min-heap

EXAMPLE – activity scheduling (unweighted interval scheduling)

*n* actitivites

Start time (activity starts *exactly* at time)

Finish time (activity finishes *before* this time)

One room

Goal: Maximize *number* of activities. (Unlike Weighted Interval Scheduling in Notes 16)

Greedy Solution:

1. Sort activities by ascending order of finish time.

2. Consider each activity according to sorted order:

   Include activity in schedule only if it does not overlap with other activities in schedule

*Optimal or heuristic?*

Optimality Proof:

1. Suppose there is an alternate schedule with a different first activity:

   $s_? \ldots f_?$ < rest of schedule >

   But $s_1 \ldots f_1$ can replace $s_? \ldots f_?$ since $f_1 \leq f_?$

2. Same argument applies to replacing other activities in the schedule

*Problems that can be solved optimally by a greedy method have a simpler structure than problems that require dynamic programming.*

## KNAPSACK PROBLEM

Can carry *k* pounds (to sell) in your knapsack.

Wish to maximize the amount of revenue.

Greedy approach: Choose according to descending order of $$$/lb.

Fractional (divisible) version:

$$$/lb for each divisible item.

Example:

*k* = 10 lbs

| | |
|---|---|
| Perfume: | $1000/lb, 3 lbs available |
| Chocolate: | $30/lb, 5 lbs available |
| Beans: | $2/lb, 5 lbs available |
| Rice: | $1/lb, 5 lbs available |

*Optimal or heuristic?*

0/1 (indivisible) version:

Example:

$k$ = 10 lbs

Bottle of wine: 5 lbs, \$500 (\$100/lb)

Rare book: 7 lbs, \$900 (\$129/lb)

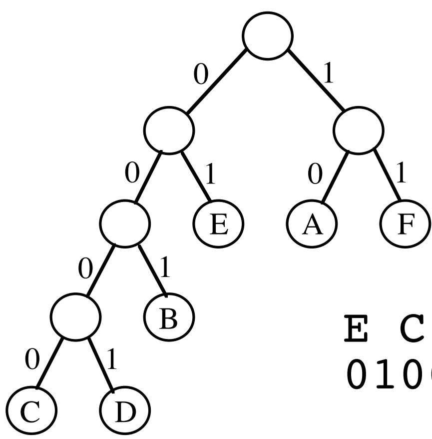
Sword: 4 lbs, \$500 (\$125/lb)

Greedy says to choose _______________, but optimal is ___________________.

(Aside: Dynamic programming solves in $O(kn)$ time when $k$ and all $2n$ input values are integers. If all objects have the same \$\$\$/lb ratio, the resulting *subset sum* problem can still take exponential time.)

HUFFMAN CODES - elementary data compression for a *static* distribution of symbols in an alphabet.

Prefix Code Tree



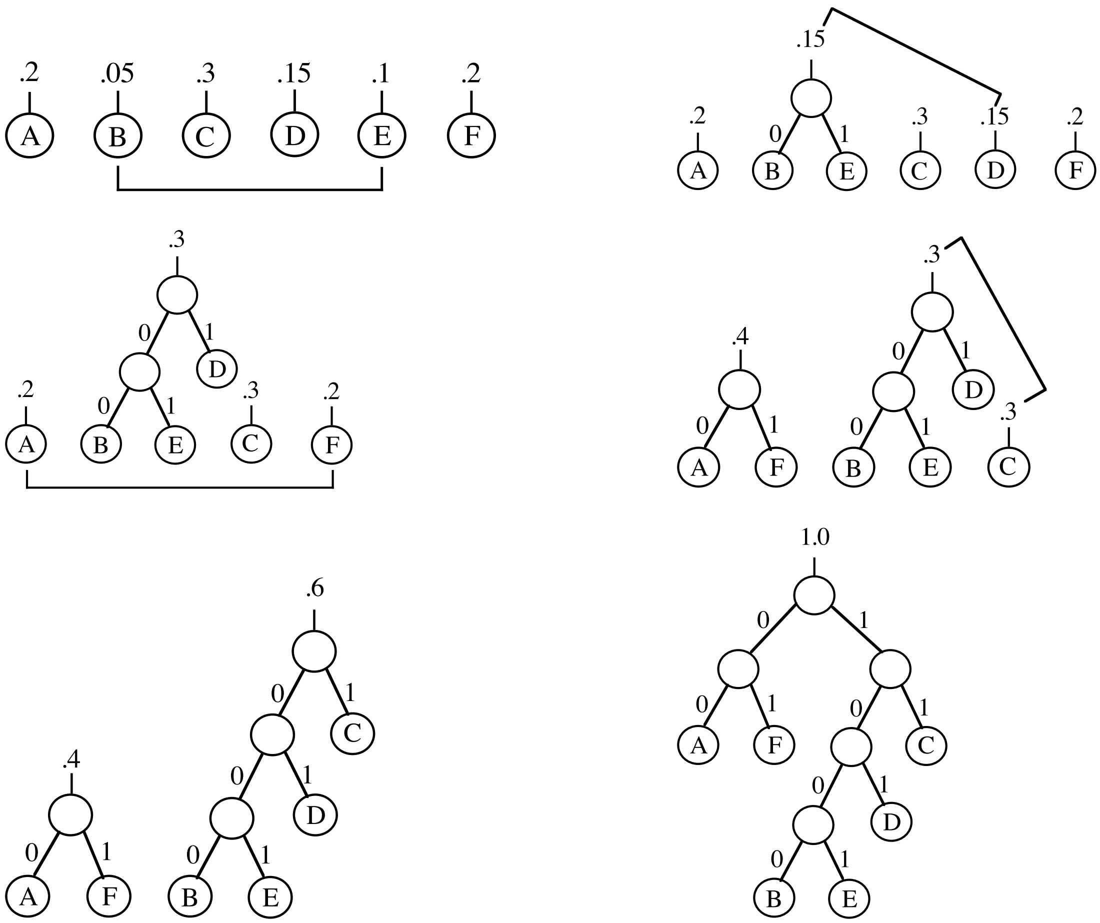

```
E C     D     F A E B
0100000001111001001
```

Concept: Letters that appear more often (higher probability) should be assigned shorter codes.

Evaluating a particular code tree (even if not optimal)

| Symbol | Probability | Bits | Probability*Bits |
|---|---|---|---|
| A | .2 | 2 | .4 |
| B | .05 | 3 | .15 |
| C | .3 | 4 | 1.2 |
| D | .15 | 4 | .6 |
| E | .1 | 2 | .2 |
| F | .2 | 2 | .4 |
| | === | | === |
| | Σ=1.0 | | Σ=2.95= Expected bits per symbol |

Algorithm: Build up subtrees by pairing trees with lowest probabilities (use min-heap).




| Symbol | Probability | Bits | Probability*Bits |
|---|---|---|---|
| A | .2 | 2 | .4 |
| B | .05 | 4 | .2 |
| C | .3 | 2 | .6 |
| D | .15 | 3 | .45 |
| E | .1 | 4 | .4 |
| F | .2 | 2 | .4 |
| | === | | === |
| | Σ=1.0 | | Σ=2.45= Expected bits per symbol |

Optimality: If the two minimum-weight trees are *not* the ones combined, then the expected bits per symbol will be larger than would be computed by the algorithm.

Time: If there are $n$ symbols, then there are $n$ - 1 subtree combining steps to perform. Each step calls HEAP-EXTRACT-MIN twice and MIN-HEAP-INSERT once. $O(n \log n)$ overall.