

CMPS 102 Homework 7, Spring '05

5 problems, 30 points, due Thursday June 2nd

Reading: Sections 25.1, 25.2, 16.1, 16.2, 16.3 in the text and the Backtracking/Branch and Bound handout. For more information on Backtracking/Branch and Bound, you can consult *Fundamentals of Algorithmics* (ch. 9) by Brassard and Bratley on reserve in the Science library.

1. (5 pts) Show how the branch and bound algorithm from the handout solves the knapsack instance with capacity 15 and the following items:

Item #	1	2	3	4	5
value	90	48	56	30	4
weight	10	6	8	6	2
util.	9	8	7	5	2

Draw the tree of partial solutions with the current value and remaining capacity for each partial solution inside the node, and the node's bound next to the node.

2. (5 pts) The subset sum problem is: given a list of n positive integer values, v_1, \dots, v_n , and a goal g , find a subset S of the integers summing up to g (or report that no such subset exists).

In other words, a feasible solution is a subset that sums up to exactly the goal value. A partial solution is a choice for the first items of whether or not they are in the subset. If the subset for a partial solution sums up to more than g then no extension of that partial solution can be feasible. Similarly, if the subset for a partial solution plus all the as-yet-unconsidered items sums up to less than g then no extension of the partial solution is feasible.

Show how backtracking with these rules to finds a solution to the subset sum instance with $g = 20$ and values 15, 10, 7, 6, 2, 2. Consider the items in the order listed and, at each node in the tree, first consider adding the next item to S and then not including it.

3. (5 pts) Coin changing. Recall that the coin changing problem is:

Given a set of n denominations $d_1 = 1 < d_2 < d_3 < \dots < d_n$ and a value C to give in change, find a way to give value C in change using as few coins as possible. Assume that there are as many coins as needed of each denomination.

One greedy algorithm takes as many high-valued coins as possible, and then moves to the next lower denomination. This doesn't always give an optimal solution, as the counterexample $C = 16$, $d_1 = 1$, $d_2 = 8$, $d_3 = 10$ shows. Find another counterexample showing that this greedy algorithm is not optimal even when each $d_i \leq d_{i+1}/2$.

4. (8 pts) Home Video CD. Assume you have n home video clips V_1, V_2, \dots, V_n where each clip V_i is m_i minutes long. Assume that the clips are numbered shortest to longest, so

$m_1 < m_2 < m_3 < \dots < m_n$. You would like to store as many complete clips as possible on a single CD that can hold C minutes of video ($C \ll \sum_i m_i$, so they don't all fit).

One greedy algorithm takes clips "shortest first" with the following pseudo-code (recall that the clips are sorted "shortest first").

1. `timeLeft = C;`
2. `for i = 1 to n do`
3. `if $m_i \leq \text{timeLeft}$ then`
4. `take clip V_i`
5. `timeLeft = timeLeft - m_i`

Either prove that the greedy algorithm that picks clips shortest first always maximizes the number of clips stored on the CD or find a counterexample.

5. (7 pts) Consider a modification to the above problem where the goal is to fill up as many minutes of the CD as possible (copying each clip at most once). Either prove that the following greedy algorithm taking the longest clip that will still fit on the CD always takes as many minutes of video as possible or find a counterexample.

1. `timeLeft = C;`
2. `for i = n downto 1 do`
3. `if $m_i \leq \text{timeLeft}$ then`
4. `take clip V_i`
5. `timeLeft = timeLeft - m_i`