Week 9: Lab

Module 4: Techniques

COLLABORATION LEVEL 0 (NO RESTRICTIONS). OPEN NOTES.

1. **Knapsack:** Consider the knapsack problem discussed in class: We have a backpack of capacity W and a set of n items, each item with weight w_i and value v_i. We denote by $K(i, w)$ the maximal value for packing a backpack of capacity w with a subset of items 1 through i. Consider the top-down recursive DP algorithm with memoization which fills in the table $table[0..W][0..n]$:

```
OPTKNAPSACKDP(i, w)
// global variable table[1..n][1..W] initialized to −1. Also global v[1..n] and w[1..n].
// returns the max value to pack a knapsack of capacity w using items 1 through i.
if (w == 0): return 0
if (i ≤ 0): return 0
IF (table[i][w] ≠ 0): RETURN table[i][w]
IF w[i] ≤ w: with = v[i]+ OPTKNAPSACKDP(i − 1, w − w[i])
ELSE: with = 0
without = OPTKNAPSACKDP(i − 1, w)
table[i][w] = max { with, without }
RETURN table[i][w]
```

Assume we have a backpack of capacity $W = 3$, and three items ($n = 3$): a hat of weight 1 and value 1, a ball of weight 2 and value 4, and a bottle of water of weight 3 and value 6.

(a) Draw the tree of recursive calls triggered by $K(3, 3)$ (with $K(3, 3)$ at the root and an edge from a to b if a generates a recursive call to b). Then show this recursion on the table below.

	x=0	x=1	x=2	x=3
i=0	0	0	0	0
i=1	0			
i=2	0			
i=3	0			

	x=0	x=1	x=2	x=3
i=0	0	0	0	0
i=1	0			
i=2	0			
i=3	0			

(b) Follow the recursion and calculate the values that will be stored in the table. Only show the values that are actually filled in. Which entries in the table will stay at their initial value (i.e. not filled in) ?

(c) Number the entries in the table that are filled in by the recursion in the order in which they are filled in.

(d) Assemble the full solution for $K(3, 3)$: list the entries that you will you visit.

2. **Pharmacist problem:** A pharmacist has W pills and n empty bottles. Bottle i can hold p_i pills and has an associated cost c_i. Given W, $\{p_1, p_2, ..., p_n\}$ and $\{c_1, c_2, ..., c_n\}$, you want to store all pills using a set of bottles in such a way that the total cost of the bottles is minimized. So the problem is to find the minimum cost for storing the W pills and what bottles to use. Note: If you use a bottle you have to pay for its cost no matter if you fill it to capacity or not.

   (a) Explain how the problem has optimal substructure.

   Answer: Consider an optimal solution O, and consider one of the bottles in it. Let's say this is bottle k, and it holds p_k pills. Then we know that the remaining bottles in O must be the optimal way to store ....

   (b) Define a subproblem and give a recursive formulation.

   (c) Give pseudocode for a top-down recursive dynamic programming algorithm with memoization and analyze its running time.

3. **Greedy pharmacist?** Someone proposes the following greedy strategy to solve the pharmacist problem (above): Pick the bottle with the smallest cost-per-pill, and recurse on the remaining pills with the remaining bottles. Show that this greedy strategy is not correct by giving a counterexample.

4. **The skis and skiers problem[1]:** You've decided to become a ski bum, and hooked up with Sugarloaf Ski Resort. They'll let you ski for free all winter, in exchange for helping their ski rental shop with an algorithm to assign skis to skiers.

   Ideally, each skier should obtain a pair of skis whose height matches his or her own height exactly. Unfortunately, this is generally not possible. We define the disparity between a skier and his/her skis as the absolute value of the difference between the height of the skier and the height of the skis. The objective is to find an assignment of skis to skiers that minimizes the sum of the disparities.

   (a) First, let's assume that there are n skiers and n skis. Consider the following algorithm: consider all possible assignments of skis to skiers; for each one, compute the sum of the disparities; select the one that minimizes the total disparity. How much time would this algorithm take on a 1GHz computer, if there were 50 skiers and 50 skis?

   (b) Consider the general case where there are m skiers and n pairs of skis (and $n \geq m$).

   Here is some notation that may help you. Sort the skiers and skis by increasing height. Let h_i denote the height of the ith skier in sorted order, and s_j denote the height of the jth pair of skis in sorted order. Let OPT(i, j) be the optimal cost (disparity) for matching the first i skiers with skis from the set $\{1, 2, ..., j\}$. The solution we seek is simply OPT(m, n).

   Define OPT(i, j) recursively.

   (c) Now describe a dynamic programming algorithm for the problem. What is the running time of your algorithm?

[1] Adapted from Harvey Mudd College.