CMPT 307 (2013)
Assignment 5
November 15, 2013

The following problems are related to dynamic programming. The sections on RNA secondary structures (6.5), sequence alignment (6.6, 6.7) are not covered.

Problems from the text (page 312):

6.1 Hints (c): $Opt(i)$ = Total weight of the optimal maximum weighted independent set of the path $P_i =< v_1, v_2, ..., v_i >$. Suppose $S(i) = 0$ if v_i is in the optimal solution of P_i, otherwise $S(i) = 1$. If v_i is in the optimal solution, $Opt(i) = w_i + Opt(i-2)$, since if v_i is in the solution, v_{i-1} can't be in the solution. What is the other case? What is the memoized version? What is the bottom-up version? What is the running time? How do you keep track of the optimal independent set?

6.2 Hints (b): $Opt(i)$ = maximum value of the optimal plan for weeks 1 through i. If the low stress job is selected for week i, $Opt(i) = l_i + Opt(i-1)$. What happens when the high stress job is selected for week i? What is the memoized version? What is the bottom-up version? What is the running time? How do you keep track of the optimal plan?

6.3 Hints(b) $Opt(i)$ is the longest path cost from v_1 to v_i. If v_i doesn't have any incoming edge, $Opt(i) = 0$. If it does have at least one incoming edge $e = (v_k, v_i)$ (can have more that one incoming edge), $Opt(i)$ is at least $1 + Opt(k)$ $k < j$. What is the memoized version? What is the bottom-up version? What is the running time? How do you keep track of the longest path?

6.5 Hints: Given a string $y = y_1 y_2 \dots y_n$, a segmentation of y is a partition of y into meaningful words. For $Opt(j)$, the last word ending with y_j can start from some y_i where i ranges from 1 to k.

6.6 Hints: Let $S(i, j)$ denote the slack of a line containing the words $w_i, w_{i+1}, \dots, w_j$. $S(i, j)$ is ∞ if the line is small to fit $w_i, w_{i+1}, \dots, w_j$. Argue that

$$Opt(j) = min_{1 \le k \le j}\{Opt(k-1) + S^2(k, j)$$

6.7

6.10 Consider $Opt(i, j)$ to be the optimal value for the first i days given that the las reboot was performed on day $i - j$ (i.e. j days before day i).

6.13 The cost of a trading cycle C in G is $\prod_{(i,j)\in C} r_{ij}$. A trading cycle C is an opportunity cycle if and only if

$$\prod_{(i,j)\in C} r_{ij} > 1$$

i.e.

$$\sum_{(i,j)\in C} \log r_{ij} > 0$$

or

$$\sum_{(i,j)\in C} -\log r_{ij} < 0$$

Thus, a trading cycle C is an opportunity cycle if and only if it has a negative cycle. How can you now solve the problem?

6.17 Consider $Opt(j)$ to be the longest increasing sequence in $P[1..j]$ that either includes $P[j]$ (provided $P[j] > P[1]$) or doesn't include $P[j]$.

Other Problems

1. **Knapsack with repetition** Modify the algorithm when an item can be picked more than once. You need to modify the solution to the Knapsack problem. Can you solve it using an array of size W?

2. Consider the following 2-partition problem. Given integers $a_1, a_2, \ldots, a_n$, we want to determine whether it is possible to partition $\{1, 2, ..., n\}$ into two disjoint subsets I, J such that

$$\sum_{i\in I} a_i = \sum_{j\in J} a_j = \frac{1}{2}\sum_{i=1}^{n} a_i$$

   For example, for the input $\{1, 2, 3, 4, 4, 4\}$ the answer is *yes* because there is a 2-partition.

   Devise and analyze a dynamic programming that runs in time polynomial in n and in $\sum_i a_i$.

3. **Subset Sum** SUBSET-SUM is a very simple variation of the knapsack problem. The problem is defined as follows: Given an array A, is it possible to find a subset that sums exactly to a bound B? The answer is either *yes* or *no*. Let $Opt(i, j)$ indicate whether it is possible to have a subset of the elements $A[1..i]$ that sum to exactly j. If the answer is *yes* then $Opt(i, j) = 1$, otherwise it is 0. Now write a recursive definition for $Opt(i, j)$. This is very similar to the knapsack problem.