CS483 - Practice Problems

Jana Košecká

Chapter 6

1. (0) Read solved exercise 1 in the book
2. (10) Problem 1

   (a) Consider the sequence of weights 3, 6, 5. The greedy algorithm will pick the middle node (as its weight is maximum) and since the other two nodes are neighbor they gets removed and the algorithm completes but the maximum weight independent weight consist of the first and third.
   (b) Consider the sequence of weights 3, 1, 2, 3. The given algorithm will pick the maximum total weigth of set S1 (3, 2) or S2 (1, 3) which comes out to be S1 with weight 5, while the maximum weight independent set consists of the first and fourth with weight equals 6.
   (c) Let S_i denote an independent set on $\{v_1, ..., v_i\}$, and let X_i denote its weight. Define X_0 and note that $X_1 = w_1$. Now, for i ¿ 1, either v_i belongs to S_i or it doesn't. In the first case, we know that v_{i-1} cannot belong to S_i, and so $X_i = w_i + X_{i-2}$. In the second case, $X_i = X_{i-1}$. Thus we have the recurrence: $X_i = \max(X_{i-1}, w_i + X_{i-2})$. We thus can compute the values of X_i, in increasing order from $i = 1$ to n. We have to find X_n, and we can compute S_n by tracking back through the computations of the *max* operator. Since we spend constant time per iteration, over n iterations, the total running time is $O(n)$.
3. (10) Problem 4

   (a) Suppose that $M = 10$, $\{N_1, N_2, N_3\} = \{1, 4, 1\}$, and $\{S_1, S_2, S_3\} = \{20, 3, 20\}$. Then the optimal plan would be $[NY, NY, NY]$, while the proposed greedy algorithm would return $[NY, SF, NY]$.
   (b) Suppose that $M = 10$, $\{N_1, N_2, N_3, N_4\} = \{1, 40, 1, 40\}$, and $\{S_1, S_2, S_3, S_4\} = \{40, 1, 40, 1\}$. Then the plan $[NY, SF, NY, SF]$ has cost 34, and it moves three times. Any other plan pays at least 40, and so is not optimal.
   (c) The basic observation is, the optimal plan either ends in NY, or in SF. If it ends in NY, it will pay N_n plus one of the following two quantities:

   - The cost of the optimal plan on $n - 1$ months, ending in NY, or
   - The cost of the optimal plan on $n - 1$ months, ending in SF, plus a moving cost of M.

   An analogous observation holds if the optimal plan ends in SF. Thus, if $OPT_N(j)$ denotes the minimum cost of a plan on months 1, ..., j ending in NY, and $OPT_S(j)$ denotes the minimum cost of a plan on months 1, ..., j ending in SF, then:
   $OPT_N(n) = N_n + \min(OPT_N(n-1), \text{M}+OPT_S(n-1))$
   $OPT_S(n) = S_n + \min(OPT_S(n-1), \text{M}+OPT_N(n-1))$
   This can be translated directly into an algorithm:

   ```
   $OPT_N(0)$ = $OPT_S(0)$ = 0
   for $i = 1, ..., n$
       $OPT_N(i)$ = $N_i$ + min($OPT_N(i-1)$, M+$OPT_S(i-1)$)
       $OPT_S(i)$ = $S_i$ + min($OPT_S(i-1)$, M+$OPT_N(i-1)$)
   end
   return the smaller of $OPT_N(n)$ and $OPT_S(n)$
   ```

4. (5) Compute the optimal solution to the following instance of the Knapsack problem and show the nW table where you computed the solution to the smaller sub-problems. Item 1($50, 5 lbs), item 2 ($60, 10 lbs) and item 3 ($140, 20 lbs), with the total knapsack weight W = 30 lbs.
5. (8) See Table 1, Optimal solution = {item 2, item 3}
   Compute the optimal solution to the following instance of the Knapsack problem and show the $n \times W$ table where you computed the solution to the smaller sub-problems. Item 1($50, 5 lbs), item 2 ($60, 10 lbs) and item 3 ($140, 20 lbs), with the total knapsack weight $W = 30$ lbs. x

Table 1: Knapsack Problem

	0	5	10	15	20	25	30
{}	0	0	0	0	0	0	0
{1}	0	50	50	50	50	50	50
{1, 2}	0	50	60	110	110	110	110
{1, 2, 3}	0	50	60	110	140	190	200

6. Fractional Knapsack Problem Optimal solution = {item 1, item 3, half of item 2}. Each time choose the remaining one with highest $\frac{v_i}{w_i}$ value. Greedy algorithm on picking the largest amount of the most valuable item while there is space in knapsack.

7. (0) Problem 2
Hint : (a) Check for instance $\{l_1, l_2, l_3\} = \{2, 2, 2\}$, and $\{h_1, h_2, h_3\} = \{1, 5, 10\}$.
(b) Let $OPT(i)$ denote the maximum revenue achievable in the input instance restricted to weeks 1 through i. The optimal solution for the input instance restricted to weeks 1 through i will select *some* job in week i, since it's not worth skipping all jobs – there are no future high-stress jobs to prepare for. If it selects a low-stress job, it can behave optimally up to week $i - 1$, followed by this job, while if it selects a high-stress job, it can bahave optimally up to week $i - 2$, followed by this job. Thus we have justified the following recurrence: $OPT(i) = \max(l_i + OPT(i-1), h_i + OPT(i-2))$. We can compute all OPT values by invoking this recurrence for $i = 1, ..., n$.

8. (0) Problem 3
Hint : (a) Check for graph on nodes v_1, ..., v_5 with edges (v_1, v_2), (v_1, v_3), (v_2, v_5), (v_3, v_4) and (v_4, v_5).
(b) Can be solved by dynamic programming. The simplest version to think of using the subproblem $OPT[i]$ for the length of the longest path from v_1 to v_i. One point to be careful of is that not all nodes v_i necessarily have a path from v_1 to v_i. We will use the value -∞ for the $OPT[i]$value in this case. We use $OPT(1) = 0$ as the longest path from v_1 to v_1 (as it has 0 edges).

9. (0) Problem 6
Hint : We observe that last line ends with word w_n and has to start with some word w_j; breaking off words w_j, ..., w_n we are left with a recursive subproblem on w_1, ..., w_{j-1}.