

## CS 170 Spring 2008 - Solutions to Homework #9

### Problem 6.16 (12 points)

For a subset of garage sales  $S \subseteq \{g_1, \dots, g_n\}$  and  $g_j \in S$ , let  $C(S, j)$  be the profit of the most profitable path starting at home, visiting only garage sales in  $S$  and ending at  $g_j$ . To express  $C(S, j)$  in terms of smaller subproblems, consider the last garage sale  $g_i \in S - \{g_j\}$  visited before  $g_j$ . The path profit up to  $g_i$  is  $C(S - \{g_j\}, i)$ , while the marginal profit of visiting  $g_j$  is  $p_j - d_{ij}$ . We must pick  $i$  as to maximize the profit, so we have:

$$C(S, j) = \max_{g_i \in S: i \neq j} \{C(S - \{g_j\}, i) + p_j - d_{ij}\}$$

Our base case initializes  $C(\{g_i\}, i)$  to be  $p_i + d_{0i}$  for all  $1 \leq i \leq n$ . We then solve the subproblems in increasing order of the size of  $S$ . The final solution is the minimum of  $C(S, j) + d_{j0}$  over all subsets  $S$  and  $g_j \in S$ , where  $d_{j0}$  is the cost of returning home.

Moreover, we can keep a record for every  $C(S, j)$  of which garage  $g_i$  yielded a maximum in the recursion; this  $g_i$  is the last garage sale visited before getting to  $g_j$  in a best path for  $C(S, j)$ . We can then use this information to backtrack from our final solution to reconstruct the entire optimal path.

This algorithm solves  $O(n2^n)$  subproblems, each in  $O(n)$  time, so it has a total running time of  $O(n^22^n)$ .

### Problem 6.29 (12 points)

For  $i \in \{1, \dots, n\}$ , let  $W(i)$  be the the weight of the best subset of consistent partial matches for  $x[1, \dots, i]$ . To compute  $W(i)$ , we consider the two following cases:

- the best subset of partial matches contains a match  $j$  with  $r_j = i$ ,
- the best subset of partial matches does not contain such  $j$

In the first case,  $W(i)$  will be the sum of  $w_j$  and the weight of the best match on the remaining of the string, i.e.  $W(l_j - 1)$ . In the second case, we will just have  $W(i) = W(i - 1)$ . This shows that the following recursion is correct:

$$W(i) = \max\{W(i - 1), \max_{j:r_j=i} \{W(l_j - 1) + w_j\}\}$$

The algorithm will then proceed computing  $W(i)$  in ascending order of  $i$  and will output  $W(n)$  as best total weight achievable. To reconstruct the actual sequence of partial matches, it suffices to keep track, for all  $W(i)$ , of which  $j$  maximizes the expression in the recursion, when the second maximum is the larger. We can then follow these pointers from  $W(n)$  backwards to identify the optimal alignment. The running time is  $O(n + m)$ , where  $m$  is the number of partial matches, as we have  $n$  subproblems and each partial match is considered once over all the maximizations.

### Problem 7.1 (6 points)

The feasible region has four vertices:  $(0,0)$ ,  $(2,5)$ ,  $(5,2)$ , and  $(5,0)$ . The optimal solution is achieved at the vertex  $(5,2)$  and has value  $5x + 3y = 31$ .

### Problem 7.4 (10 points)

Let  $R$  the quantity of RegularDuff beer and  $S$  of DuffStrong beer that Moe is ordering per week. Our linear program is:

$$\begin{aligned} & \max R + 1.5S \\ & R \geq 2S \\ & R + S \leq 3000 \\ & R, S \geq 0 \end{aligned}$$

The feasible region has three vertices:  $(R, S) = (0, 0)$ ,  $(3000, 0)$ , and  $(2000, 1000)$ . The optimum is obtained at the vertex  $(R, S) = (2000, 1000)$  with an optimum profit of \$3,500.