Lindsey Bieda and Joe Frambach
Dynamic Programming Problems
10.05.2011

20. The input to this problem is two sequences $T = t_1, \ldots, t_n$ and $P = p_1, \ldots, p_k$ such that $k = n$, and a positive integer cost c_i associated with each t_i. The problem is to find a subsequence of T that matches P with maximum aggregate cost. That is, find the sequence $i_1 < \ldots < i_k$ such that for all j, $1 = j = k$, we have $t_{i_j} = p_j$ and $\sum_{j=1}^{k} c_{i_j}$ is maximized.
So for example, if $n = 5$, $T = XY\ XXY$, $k = 2$, $P = XY$, $c_1 = c_2 = 2$, $c_3 = 7$, $c_4 = 1$ and $c_5 = 1$, then the optimal solution is to pick the second X in T and the second Y in T for a cost of $7 + 1 = 8$.

(a) Give a recursive algorithm to solve this problem. Then explain how to turn this recursive algorithm into a dynamic program.

A function, Weighted Sub Sequence (WSS), is defined:
The recursive algorithm is called initally as wss(n,k) with T, P, and C being globally accessable.
The algorithm works by examining substrings of both of the given sequences and then determining where the values at a given position are equal and maximizes the values at these positions.
wss(i,j):
    if i = 0 or j = 0: *outside the bounds of either string*
        return 0 *no value here*
    if i ¡ j: *if the length of P is less than the length of T there is no solution*
        return $-\infty$
    else if $T_i = P_j$: *The last characters are equal. Either use it or ignore and continue.*
        return max(v_i + wss(i-1,j-1), wss(i-1,j)) *check if there is a better location elsewhere in the string*
    else:
        return wss(i-1,j) *check the rest of the string*

Given the above recursive definition we can draw a call tree and then determine what pruning rules to apply. From there we map the tree to an array based on these pruning rules.
The pruning rules are based on i and j passed into the WSS call, so the complexity is polynomial in terms of n and k.

(b) Give a dynamic programming algorithm based on enumerating subsequences of T and using the pruning method.

Tree description:
Each node of the tree is represented by a start and end position in T, and is assigned two values: the total number of letters in this substring that matches P and the cost given by the matching letters inside the string determined by these start and end locations. The tree is rooted at [0,0] representing the empty string with values: 0,0 as previously described.

Ruling Prunes, Pruning Rules, Pruling Runes:

(a) At every level

(c) Give a dynamic programming algorithm based on enumerating subsequences of P and using the pruning method.