CS140: Algorithms

Fall 2000

Solutions to Assignment 7

1. Problem 16-2 on page 325 of the text but modify the cost criterion to read: "We wish to minimize the sum, over all lines, of the cubes of the numbers of extra space characters at the ends of lines." (This makes the problem slightly easier.) We'll denote this minimum sum for the first i words as $C(i)$.

   Before you describe and analyze your dynamic programming algorithm, give an inductive definition of the cost function $C(i)$. Hint: How many words can possibly be printed on the last line?

   Ans. Suppose j words are printed on the last line in an optimal solution. Then $C(n) = C(n-j-1) + s(n-j, n)$, where $s(i,k)$, $i \leq k$, is the cube of the number of spaces left on a line containing words i through k. (If the words don't fit on a single line we define $s(i,k) = \infty$.)

   Of course we don't know j, but at most $\min(M/2, n)$ words can be written on single line so $j \in [0, \min(M/2-1, n-1)]$. Thus we can define $C(i)$ recursively as follows. $C(0) = 0$ and for $i > 0$

   $$C(i) = \min_{j=0,\ldots,\min(M/2-1,i-1)} C(i-j-1) + s(i-j, i).$$

   The dynamic programming algorithm simply computes $C(i)$ for increasing values of i from 0 to n. The solution is $C(n)$. It takes $O(1)$ time to compute $C(i)$ if we know $C(0), \ldots, C(i-1)$, so the running time of the algorithm is $O(n)$.

2. Problem 16-3 on page 325 of the text but restrict the edit operations to copy, insert, delete, and substitute. Further assume that cost(copy)= 0, cost(insert)=cost(delete)=2 and cost(substitute)= 3. Before you describe and analyze your dynamic programming algorithm, give an inductive definition of the edit distance $D(x_1, \ldots, x_m; y_1, \ldots, y_n)$. Hint: You'll find this problem is similar (but not identical) to Longest Common Subsequence.

   If we knew the last operation in a optimal transformation, we could define $D(x_1, \ldots, x_m; y_1, \ldots, y_n)$ as follows:

Last Operation	$D(x_1, \ldots, x_m; y_1, \ldots, y_n) =$
Copy	$D(x_1, \ldots, x_{m-1}; y_1, \ldots, y_{n-1})$
Substitute	$3 + D(x_1, \ldots, x_{m-1}; y_1, \ldots, y_{n-1})$
Insert	$2 + D(x_1, \ldots, x_m; y_1, \ldots, y_{n-1})$
Delete	$2 + D(x_1, \ldots, x_{m-1}; y_1, \ldots, y_n)$

Even though we don't know the last operation we can define $D(\cdot;\cdot)$ recursively as follows. $D(\epsilon;\epsilon) = 0$, $D(x_1,\ldots,x_k;\epsilon) = 2*k$, and $D(\epsilon;y_1,\ldots,y_k) = 2*k$, where ϵ is the empty string. Then for $i > 0$ and $j > 0$

$$D(i,j) = \min(3 * 1_{x_i \neq y_j} + D(i-1,j-1), 2 + D(i-1,j), 2 + D(i,j-1)).$$

(Recall 1_P is 1 if the proposition P is true and 0 if P is false.)

We can compute $D(i,j)$ for increasing values of $|i-j|$. Each computation takes constant time. The desired solution is $D(n,m)$. The running time of the algorithm is $O(nm)$.