Code Jam 2012 - Round 1C

Analysis: Box Factory

This problem is a variation of the Longest Common Subsequence problem, which is to find the longest string of characters which occurs as a subsequence of each of two strings (a string S is a subsequence of a string T if S occurs in T, possibly with more characters inserted between the elements of S.) In this case, the first "string" is the sequence of types of each box produced by the factory, and the second "string" is the sequence of types of each toy produced by the factory.

A dynamic programming algorithm for this problem is to find the maximum number of toys that could be placed in boxes using the first x boxes and the first y toys. Let this value be f[x][y]. Then f[x][y] is equal to the maximum of:

- f[x-1][y]
- f[x][y-1]
- f[x-1][y-1]+1

with the last case only applying if the x^{th} letter of the first string is equal to the y^{th} letter of the second string. These three cases correspond to the last action taken being to drop a box, to drop a toy, and to put a toy in a matching box, respectively.

Unfortunately, even though the number of runs of boxes and toys is small, the total number of boxes and toys can be very large, so this algorithm is not practical. But we can modify it so that f[x][y] will be the maximum number of toys that could be placed in boxes using the first x **runs** of boxes and the first y **runs** of toys. Now f[x][y] is the maximum of:

- f[x-1][y]
- f[x][y-1]
- f[a][b]+g(a,b,x,y), for all a<x, b<y

Similarly to before, the last case only applies if the type of box run x matches the type of toy run y. It corresponds to only using boxes of that type between runs a+1 and x, and toys of that type between runs b+1 and y. g(a,b,x,y) is the minimum of the number of those toys and the number of those boxes, which is the number of toys that can be placed in boxes in that range.

This algorithm is $O(n^4)$. Another improvement can reduce this again to $O(n^3)$, which we leave as an exercise to the reader!