Homework 5

Robbie McKinstry, Jack McQuown, Cyrus Ramavarapu

12 September 2016

Greedy Problems

Problem 7:

Problem 17:

Dynamic Programming

Problem 2:

The longest common subsequence between the three can be initially defined recursively by considering the last letter in common between the three strings and then seeing if this letter is in the longest common subsequence of one letter shorter.

$$S_A = A_1, A_2, A_3, \ldots, A_{i-1}, A_i$$

$$S_B = B_1, B_2, B_3, \ldots, B_{j-1}, B_j$$

$$S_C = C_1, C_2, C_3, \ldots, C_{k-1}, C_k$$

This analysis leads to the following recursive algorithm:

Function: LCS

Input: *int i, int j, int k*

if $i \equiv j \equiv k \equiv 0$ **then**
  | *return 0*
end

if $A_i \equiv B_j \equiv C_k$ **then**
  | $LCS(i-1, j-1, k-1) + A_i$
else if $(A_i \equiv B_j) \neq C_k$ **then**
  | $\max(LCS(i-1,\ j-1,\ k),\ LCS(i,\ j,\ k-1))$
else if $(A_i \equiv C_k) \neq B_j$ **then**
  | $\max(LCS(i-1,\ j,\ k-1),\ LCS(i, j-1, k))$
else if $A_i \neq (B_j \equiv C_k)$ **then**
  | $\max(LCS(i,\ j-1,\ k-1),\ LCS(i-1,\ j,\ k))$
else
  | $\max(LCS(i,\ j-1,\ k-1), LCS(i-1,\ j,\ k-1), LCS(i-1,\ j-1,\ k))$
end

Although this algorithm produces the longest common subsequence for three given strings, it runs in exponential time due to the numerous recursive calls that operate on a problem of only 1 letter smaller.

By moving to an array based solution and changing the recursive calls to array look-ups, a polynomial runtime algorithm can be developed.