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.

$$S_A = A_1, A_2, A_3, \ldots, A_{i-1}, A_i$$
$$S_B = B_1, B_2, B_3, \ldots, B_{j-1}, B_j$$
$$S_C = C_1, C_2, C_3, \ldots, C_{k-1}, C_k$$
$$LCS(A_{i-1}, B_{j-1}, C_{k-1}) + A_i \overset{?}{\geq} LCS(A_{i-1}, B_{j-1}, C_{k-1})$$

This analysis leads to the following recursive algorithm:

```
function LCS(int i,j,k)
    if i ≡ 0 ∨ j ≡ 0 ∨ k ≡ 0 then
        return 0
    end if
    if A_i ≡ B_j ≡ C_k then
        LCS(i − 1, j − 1, k − 1) + A_i
    else
        if (A_i ≡ B_j) ≠ C_k then
        else
            if (A_i ≡ C_k) ≠ B_j then
            else
                if A_i ≠ (B_j ≡ C_k) then
                else
                end if
            end if
        end if
    end if
end function
```