Homework 10

Robbie McKinstry, Jack McQuown, Cyrus Ramavarapu

26 September 2016

Problem 19:

0.1 Part A

First, we will give our recursive solution, and then we will explain how it works.

```
Function: Maximum Matching Sequence (MMS)
Input: $T = t_1, \dots, t_n, P = p_1, \dots, p_n, c_i \forall t_i \in T$ Output: Integer
if $P$ is Empty then
    Return: 0
if $S$ is Empty then
    Return: 0
Let $subproblems = ZeroedArray[n]$
foreach $t_i$ in $T$ do
    if $t_i = P_n$ then
        $subproblem[i] = MMS(t[: i], p[: n-1]) + c_i$
Return: $Max(subproblems)$
```

We consider the final letter of P. That letter must match some letter in T, since it must be the last letter of the sequence. Thus, we search T for all of the sites that could end the sequence. Each of those sites is a subproblem. The subproblem consists of all of the letters up to and excluding the matched letter in T (as it was the final letter of the sequence thus far) and all of the letters in P save the final one, which was just matched. The return value is the maximum cost subproblem, as defined in the problem specification.

To convert this into a recursive solution, create a 2D array $A[n][n]$. Populate A by iterating through the rows, and then the columns. The cell $A[i, j]$ represents the subproblem with $T_{i,j} = T[: i]$ and $P_{i,j} = P[: j]$. Thus, each cell is populated by looking at $c_i + A[i-1, j-1]$ if $T_i = Pj$. The output is the final cell, and the sequence can be found by tracing back through the array and taking a letter from T when value in the cell changes.

0.2 Part B

Notice first that the solution is obviously subcollection of T. Thus, one can identify a solution by search the space of all subcollections of T until finding a match on P.

To do this, for each t_i in T, starting with the empty root node, populate the ith level of the tree by taking all nodes from the $i-1$ level and giving them two children, one for the sequence that concatenates the empty string, and one for the sequence that concatenates t_i (the nodes of the tree store the subsequence). Next, prune any nodes on the same level that do not contain a prefix of P. If they do not contain a prefix of P, then they will never match P. Finally, prune the minimum-valued node if there are two nodes on the same level of the same length. (Since they are the same length and are prefixes, by the 1st rule, they are the same string.)

Problem 22: