Wyatt Kormick

Assignment 4 CSCI4041

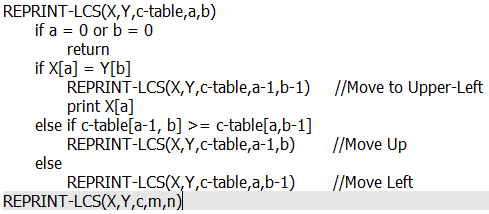
DUE: Sunday, November 6, 10:00pm (+1hr55min grace period) Fall 2016

***IF YOUR ASSIGNMENT IS NOT ACCEPTED BY TURNITIN BECAUSE IT DOES NOT CONTAIN 120+ WORDS OF DIGITAL TEXT, 3 POINTS WILL BE DEDUCTED FROM YOUR HOMEWORK.***

**1. Constructing Solutions After Calculating Optimal Cost (15.4-2)**

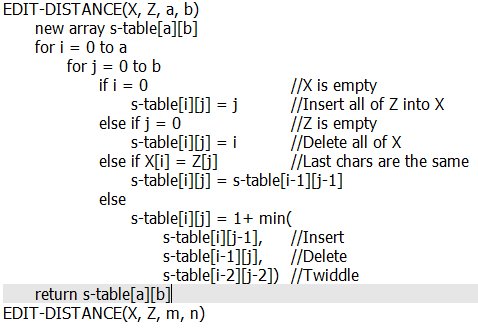
Give pseudocode to reconstruct an LCS from the completed c table and original sequences X = <x1, x2, …, xm> and Y = < y1, y2, …, yn> in O(m+n) time, without using the b table.

O(m+n) because at worst it moves all the way up, m, then all the way left, n (no diagonal movements through the c-table)



**2. Editing Distance (version of 15-5)**

Write a polynomial time algorithm to calculate optimal cost (minimal editing distance) between 2 strings, as described in problem 15-5. However, use only the operators Delete, Insert, and Twiddle. This can be iterative or recursive. State the runtime with a brief justification.

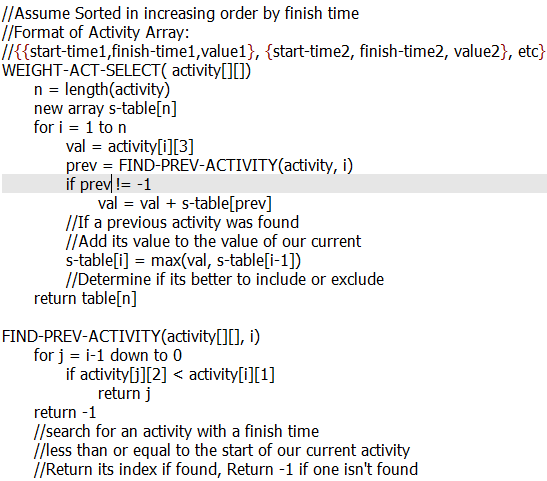


O(mn)

It iteratively fills in a table with dimensions equal to the lengths (m,n) of the two strings. It is an O(1) operation to get the value once the table has been filled in.

**3. Modified Activity Selection**

Consider a modification to the activity-selection problem in which each activity ai has, in addition to a start and finish time, a value vi. The objective is no longer to maximize the number of activities scheduled, but instead to maximize the total value of the activities scheduled. Give a polynomial-time algorithm for this problem. Justify the runtime.



This algorithm has O(n2) running time. It iterates through a new table, filling in the values of all of the subproblems. Inside the loop another loop occurs, iterating through the list of activities as a search for the previous activity. It is then O(1) to get the value of our problem out of the generated s-table. This could be reduced to O(n log n) by changing the search algorithm into a binary search.

**4. Optimal Substructure**

Consider the problem of given a set { x1, x2, …, xn } of points on the real line, determine the smallest set of unit-length closed intervals that contains all of the given points. Prove that this problem exhibits optimal substructure.

Y = sort(X) = {y1, y2, … , yn}

The first (left-most) interval is [y1, y1 + 1]

The optimal solution S is the union of the left-most interval [y1, y1 + 1] and the optimal

solution of all of the intervals to the right of y1 + 1

Proof:

Claim: There is an optimal solution S that contains [y1, y1 + 1]

Suppose: There is an optimal solution S’ that contains [x’, x’ + 1] which covers y1, and x’ < 1

Since y1 is the left-most point of our set, we can replace [x’, x’+1] with [y1, y1+1] to get

another optimal solution.

Therefore: This problem exhibits optimal substructure.

**5. Greedy Choice Property**

Suppose you are given two sets A and B, each containing n positive integers. You can choose to reorder each set however you like. After reordering, let ai the ith element of set A, and let bi be the ith element of set B. You then receive a payoff of 𝚷 { i over 1 to n } ai^bi. Prove that the greedy choice property holds for this algorithm.

Both a and b should be sorted in monotonically increasing order.

Some global solution contains our greedy choice ag paired to bg

Given another global solution without our greedy choice, swap in our greedy choice

Our solution with the greedy choice swapped in is as good as our greedy solution