ESO207: Data Structures and Algorithms

Programming Assignment *4* *Due: 11 November 23:59 hrs*

Problem 1. Edit Distance: Dynamic Programming

Given two strings $x[1 \ldots m]$ and $y[1 \ldots n]$ over some alphabet, we have to find the minimum cost transformation that would transform x to y. Suppose we think of having a sequence z, which is initially empty, and we transform x and the result of the transformation is z. The following transformations are allowed, each transformation applies to a single character of the input. Let i be the current index into x (initially 1) and j be the next empty index of z (initially 1).

1. *Copy* the character $x[i]$ into $z[j]$. Increment both i and j by 1.
2. *Replace* the character $x[i]$ by another character c and set $z[j]$ to c. Increment i and j each by 1.
3. *Insert* a character c at position $z[j]$ and increment j by 1 (i is unchanged).
4. *Delete* the current character at $x[i]$ by incrementing i by 1 but leaving j unchanged.

At the end of a sequence of transformations, x has been transformed into z. The goal of these transformations is to transform x into the other given string y.

Now each of these operations are given some costs. For example, the cost of copying could be 0, the cost of replace could be 1, and the cost of insert and delete could be 2 each. For example, under this cost function, the strings $x =$ `goodly` and $y =$ `godly` have an edit distance of 2, that is x can be transformed to y by deleting one extra 'o' character, incurring a cost of 2, and copying the rest of the characters, with a cost of 0. Given strings $x =$ `attcgat` and $y =$`atucgb`, we can transform x into y as follows: Copy a, copy t, replace t by u, copy c, copy g, replace a by b, delete t. The cost of this transformation is: $1 \cdot 2$ (replace) $+2 \cdot 1$ (delete) $= 4$. This is the lowest cost transformation among all possible transformations (under the given cost array) that converts x into y.

The edit distance between two strings is the minimum cost transformation that transforms x to y. (If insert cost equals delete cost, then this function is symmetric). The problem is to design a dynamic programming algorithm that takes as input two strings x and y (in separate input lines) and the cost array (in another line) and computes the edit distance between x and y and prints a sequence of operations that transforms x into y. An example input could be as follows.

```
attcgat
atucgb
0 1 2 2
```

Here, we assume that the entries in the cost array are given as a sequence of four integer or real valued numbers, the first number is the cost of copying, the second is the cost of replace, the third is the cost of insertion and the fourth is the cost of deletion. The output for this instance is

```
4 C C Ru C C Rb D
```

The first number in the output is the calculated edit distance that is the minimum cost transformation for transforming x to y. The next is a string over the alphabet $\{C, R, I, D\}$, which specifies the transformation that transforms x to y and attains the edit distance. Here, C stands for copy, Rc for replace the current character of x by the character c , Ic for insert the character c and D for delete the current character.

The standard algorithm for computing the edit distance is a dynamic programming algorithm. For $0 \leq i \leq m, 0 \leq j \leq n$, let $E[i, j]$ denote the edit distance of the prefix $x[1 \ldots i]$ and $y[1 \ldots j]$. We can now set up a recurrence equation for $E[i, j]$. The boundary case is when $i = 0$ and $j \geq 0$–in this case, $E[0, j] = j \cdot \text{cost(insert)}$. Similarly, $E[i, 0] = i \cdot \text{cost(delete)}$. In the general case, let $i, j \geq 1$.

In the general case, consider $x[i]$ and $y[j]$. There are three possibilities:

1. Replace $x[i]$ by $y[j]$ (or copy $x[i]$ if $x[i] = y[j]$) and transform $x[1 \ldots i-1]$ and to $y[1 \ldots j-1]$. The cost is cost(Replace) (or, cost of copy) $+E[i-1, j-1]$.
2. Delete $x[i]$ and transform $x[1 \ldots i-1]$ to $y[1 \ldots j]$. The cost is cost(delete) $+E[i-1, j]$.
3. Insert $y[i]$ and transform $x[1 \ldots i]$ to $y[1 \ldots j-1]$. The cost is cost (insert) $+E[i, j-1]$.

We choose the minimum of these possibilies.

We can write this as a recurrence equation as follows:

$$E[i,j] = \min \begin{cases} \text{Cost(copy)} + E[i-1, j-1] & \text{if } x[i] = y[j] \\ \text{Cost(replace)} + E[i-1, j-1] & \text{if } x[i] \neq y[j] \\ \text{Cost(insert)} + E[i, j-1] & \\ \text{Cost(delete)} + E[i-1, j] & \end{cases}$$

together with the cost for boundary cases $i = 0$ or $j = 0$. This can be translated into appropriate code.