Edit Distance

input: A pair $I = (A[1..m], B[1..n])$ of strings.

output: The *edit distance* between A and B. This is the minimum number of *edit operations* needed to transform A into B. An edit operation either deletes one character, inserts one character, or replaces one character with another. The edit distance is the minimum size of any sequence of edit operations that transforms A into B.

Here is the algorithm. Fix any input instance $(A[1..m], B[1..n])$.

subproblems. For each pair (i, j) with $i \in \{0, 1, \ldots, m\}$ and $j \in \{0, 1, \ldots, n\}$, define $D[i, j]$ to be the edit distance between $A[1..i]$ and $B[1..j]$.

recurrence. $D[i, j] = \max(i, j)$ if $0 \in \{i, j\}$, and, otherwise

$$D[i,j] = \begin{cases} D[i-1, j-1] & \text{if } A[i] = B[j], \\ \min(1 + D[i-1, j], 1 + D[i, j-1], 1 + D[i-1, j-1]) & \text{if } A[i] \neq B[j]. \end{cases}$$

The intuition is as follows. In the case that $A[i] = B[j]$, there is an optimal sequence x^* of edit operations that leaves $A[i]$ untouched, so $A[i]$ ends up as $B[j]$. So x^* also transforms $A[1..i-1]$ into $B[1..j-1]$, and is a shortest-possible sequence that does so. So in this case, $D[i, j] = D[i-1, j-1]$. In the remaining case, when $A[i] \neq B[j]$, each sequence x that transforms $A[1..i]$ into $B[1..j]$ can be obtained in one of the following three ways:

(a) take any sequence x' that transforms $A[1..i-1]$ into $B[1..j]$, then add "delete $A[i]$".

(b) take any sequence x' that transforms $A[1..i]$ into $B[1..j-1]$, then add "insert $B[j]$".

(c) take any x' that transforms $A[1..i-1]$ into $B[1..j-1]$; add "replace $A[i]$ by $B[j]$".

(Ways (a) and (b) are not mutually exclusive.) In each case, the resulting sequence x has one more operation than x' does. That is, $|x| = 1 + |x'|$. By this reasoning,

$$\begin{aligned} D[i,j] &= \min \begin{cases} \min\{|x| : x \text{ can be obtained via (a)}\}, \\ \min\{|x| : x \text{ can be obtained via (b)}\}, \\ \min\{|x| : x \text{ can be obtained via (c)}\} \end{cases} \\ &= \min \begin{cases} \min\{1 + |x'| : x' \text{ transforms } A[1..i-1] \text{ into } B[1..j], \\ \min\{1 + |x'| : x' \text{ transforms } A[1..i] \text{ into } B[1..j-1], \\ \min\{1 + |x'| : x' \text{ transforms } A[1..i-1] \text{ into } B[1..j-1] \end{cases} \\ &= \min \begin{cases} 1 + D[i-1, j], \\ 1 + D[i, j-1], \\ 1 + D[i-1, j-1]. \end{cases} \end{aligned}$$

time. There are $(n+1) \cdot (m+1) = O(nm)$ subproblems, and for each, the right-hand side of the recurrence takes constant time to evaluate (assuming the smaller subproblems are already solved). So the total time is $O(nm)$.

correctness. To show correctness, we would use the ideas sketched above to carefully prove that the recurrence relation is correct. Correctness of the algorithm would follow by induction (on the size of the subproblem $D[i, j]$).

This algorithm is equivalent to running the Shortest Path algorithm on an appropriate graph.

External resources on Edit Distance

- CLRS Chapter 15. (problem 15-5). Dasgupta et al. Chapter 6.
- Jeff Edmond's Lecture 5

  `http://jeffe.cs.illinois.edu/teaching/algorithms/notes/05-dynprog.pdf`
- previous CS 5800 lecture video

  `https://neu.tegrity.com/#/recording/cdcdb48b-7db1-4540-8d93-e003145d6a39?startTime=3558976`
- MIT lecture video

  `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/lecture-21-dp-iii-parenthesization-edit-distance-knapsack`