Daily Coding Problem #31

Problem

This problem was asked by Google.

The edit distance between two strings refers to the minimum number of character insertions, deletions, and substitutions required to change one string to the other. For example, the edit distance between “kitten” and “sitting” is three: substitute the “k” for “s”, substitute the “e” for “i”, and append a “g”.

Given two strings, compute the edit distance between them.

Solution

First, notice that we can probably define this problem recursively. How can we notice this? If we look at the example (kitten -> sitting) and its solution path (kitten -> sitten -> sittin -> sitting), we can see that it's the minimum distance between

sitten and sitting plus one.

The recurrence, then, looks like this:

- If either `s1` or `s2` are empty, then return the size of the larger of the two strings (since we can trivially turn an empty string into a string by inserting all its characters)
- Otherwise, return the minimum between:
    - The edit distance between each string and the last n - 1 characters of the other plus one
    - If the first character in each string is the same, then the edit distance between s1[1:] and s2[1:], otherwise the same edit distance + 1

So, the naive recursive solution would look like this:

```
def distance(s1, s2):
    if len(s1) == 0 or len(s2) == 0:
        return max(len(s1), len(s2))

    return min(distance(s1[1:], s2) + 1,
               distance(s1, s2[1:]) + 1,
               distance(s1[1:], s2[1:]) if s1[0] == s2[0]
               else distance(s1[1:], s2[1:]) + 1)
```

However, this runs very slowly due to repeated subcomputations. We can speed it up by using dynamic programming and storing the subcomputations in a 2D matrix. The index at i, j will contain the edit distance between `s1[:i]` and `s2[:j]`. Then, once we fill it up, we can return the value of the matrix at A[-1][-1].

```
def distance(s1, s2):
    x = len(s1) + 1 # the length of the x-coordinate
```

```
y = len(s2) + 1 # the length of the y-coordinate

A = [[-1 for i in range(x)] for j in range(y)]
for i in range(x):
    A[0][i] = i

for j in range(y):
    A[j][0] = j

for i in range(1, y):
    for j in range(1, x):
        if s1[j- 1] == s2[i - 1]:
            A[i][j] = A[i - 1][j - 1]
        else:
            A[i][j] = min(
                    A[i - 1][j] + 1,
                    A[i][j - 1] + 1,
                    A[i - 1][j - 1] + 1
                    )
return A[y - 1][x - 1] # return the edit distance between the two strings
```

This now takes O(N * M) time and space, where N and M are the lengths of the strings.