Introduction to Algorithms: 6.006
Massachusetts Institute of Technology
Erik Demaine, Ronald L. Rivest, and Nir Shavit

November 22, 2013
Handout # 6

Problem Set 6, Problem 2

Kevin Peng
Collaborators: Genghis Chau, Jodie Chen, Anubhav Jain
Recitation: Vladislav Kontsevoi WF3

This problem set is due **Thursday, December 5** at **11:59PM**.

This solution template should be turned in through our submission site. [1]

For written questions, full credit will be given only to correct solutions that are described clearly *and concisely*.

Please fill in the TA and recitation section you attend. Otherwise you may not be able to get your problem sets back in section!

---

[1] Register an account, if you haven't done so. Then go to Homework, Problem Set 6, and upload your files.

Problem 6-2. [35 points] **Adaptive Algorithms: Fast If Lucky**

Your goal in this problem is to develop an algorithm to compute the ***edit distance*** $e(x, y)$ between two n-character strings x and y, that is, the minimum number of character insertions, deletions, and/or substitutions to transform x into y. Unlike the $O(n^2)$ algorithm presented in class, however, your algorithm will run faster whenever $e(x, y)$ happens to be $o(n)$.

(a) [10 points] Suppose you knew an upper bound E such that $e(x, y) \leq E$. Describe how to compute $e(x, y)$ in $O(n \cdot E)$ time. Be sure to analyze your algorithm..

First, we note that edit distance is defined as the number of operations instead of the sum of the cost of the operations in this case. We can think of this as the in-class definition, where the cost per operation is 1. We know that each subproblem takes $O(1)$ to solve, so to reduce this to an $O(n\dot{E})$ algorithm, we would like to reduce the number of subproblems we will have to consider to $O(nE)$. We can insert a check to make sure that $|i - j| \leq E$. Once we hit a cell where $|i - j| > E$, we can skip all the cells to the left and any cells in the above row where $|i - j| > E$. We end up at a cell that satisfies $|i - j| \leq E$ and then we could continue. (We would have to assign the cell to the right (if there is one) with an infinite cost so that the recursive cost function still works). Using this algorithm, each row will have at most $E + 2$ subproblems that we actually consider, and there are n rows, so the total running time is $O(nE)$.

```
for i = n downto 0:
  for j = n downto 0:
    if |i - j| > E:
      e(i, j) = inf
      jump to next subproblem where |i - j| <= E and
      label cell to the right (if there is one) with inf cost
    else:
      e(i, j) = min(1 + e(i + 1, j) if i < |x|,
                    1 + e(i, j + 1) if j < |y|,
                    1 + e(i + 1, j + 1) if i < |x|, j < |y|)
```

(b) [10 points] Given a value E, how would you determine whether $e(x, y) \leq E$ in $O(n \cdot E)$ time?

Hint: Use your algorithm from part (a). How can you tell whether it succeeded?

We can use our algorithm from part (a) and then check the value of $e(0, 0)$, which holds the value of the problem we're trying to solve. If $e(0, 0)$ is not infinity, we know that there is an edit distance $\leq E$. Otherwise, there isn't. This takes the same amount of time as part (a) plus a constant check, so this runs in $O(nE)$ time.

(c) [15 points] Now give an algorithm to compute $e(x, y)$ in $O(n \cdot e(x, y))$ time without any assumptions. (In particular, the to-be-computed value of $e(x, y)$ is not known to your algorithm ahead of time.)

Hint: Use your algorithm from part (b), for some sequence of choices for E.

We use our algorithm from part (b), except we pick a specific sequence of choices: 1, 2, 4, $\ldots 2^k$ for some $k \in \mathbb{Z}$ such that $2^k \geq e(x, y)$ and k as small as possible. Using an amortized analysis, this will take a total of $n + 2n + 4n + \ldots 2^k n = (2^{k+1} - 1)n$ time. $2^{k-1} < e(x, y)$ because of the way we defined k, so $2^{k+1} < 4e(x, y)$, so this algorithm will run in $O(4ne(x, y)) = O(ne(x, y))$ time.