CSCI3160: Regular Exercise Set 7

Prepared by Yufei Tao

Problem 1. Let s and t be strings with lengths m and n respectively, satisfying the condition that $s[m] = t[n]$. In the lecture, we proved:

$$edit(s,t) = \min \begin{cases} edit(s[1..m-1], t[1..n-1]) \\ 1 + edit(s, t[1..n-1]) \\ 1 + edit(s[1..m-1], t). \end{cases}$$

Prove: the above result can be simplified into: $edit(s,t) = edit(s[1..m-1], t[1..n-1])$.
(Hint: you can leverage the above result in your proof.)

Solution. One way to convert $s[1..m-1]$ to $t[1..n-1]$ is to first insert $s[m]$ and then perform $edit(s, t[1..n-1])$ operations to obtain t. This shows $edit(s[1..m-1], t[1..n-1]) \le 1 + edit(s, t[1..n-1])$. A similar argument shows $edit(s[1..m-1], t[1..n-1]) \le 1 + edit(s[1..m-1], t)$.

Problem 2*. In the class we proved the "grand lemma" only for the case where $s[m] = t[n]$. In this problem, we will cover the other case where $s[m] \neq t[n]$. Let s and t be strings with lengths m and n respectively, satisfying the condition that $s[m] \neq t[n]$. Prove:

$$edit(s,t) = \min \begin{cases} 1 + edit(s[1..m-1], t[1..n-1]) \\ 1 + edit(s, t[1..n-1]) \\ 1 + edit(s[1..m-1], t). \end{cases}$$

Solution. Let Σ^* be an optimal sequence of operations that turns s into t. We claim that at least one of the following situations will occur:

- Situation 1: there exists an operation sequence of length $|\Sigma^*| - 1$ that turns $s[1..m-1]$ into $t[1..n-1]$.
- Situation 2: there exists an operation sequence of length $|\Sigma^*| - 1$ that turns s into $t[1..n-1]$.
- Situation 3: there exists an operation sequence of length $|\Sigma^*| - 1$ that turns $s[1..m-1]$ into t.

This claim will imply the equation we are trying to prove.

To prove the claim we distinguish three possibilities:

1. *The last character of s survives till the end of Σ^* and matches $t[n]$.* In this case, Σ^* must contain a single operation that concerns the last character of s; furthermore, that operation must be a substitution that replaces the character with $t[n]$. Removing the operation gives a sequence for Situation 1.

2. *The last character of s survives till the end, and but does not match $t[n]$.* In this case, Σ^* must contain an insertion that inserts the character — say c — eventually used to match $t[n]$. Furthermore, that insertion is the only operation that concerns c. Removing the operation gives a sequence for Situation 2.

3. *The last character of s is deleted.* In this case, Σ^* must contain a deletion that deletes the last character of s. Furthermore, that deletion is the only operation concerning that character. Removing the operation gives a sequence for Situation 3.

Problem 3. Let s be a sequence of n letters. Design an $O(n)$-time algorithm to decide whether it is possible to delete $n-6$ letters from s so that the remaining sequence of 6 letters reads "secret". For example, the answer is yes for "as**s**df**ec**fas**r**df**e**s**t**", but no for "assdfecfaserdfst".

Solution. Define string $t =$ "secret". For each $i \in [1, n]$ and $j \in [1, 6]$, define $\mathit{deledit}(i, j)$ to be the length of the shortest sequence of deletions that turns $s[1..i]$ into $t[1..j]$; if no such sequences exist, define $\mathit{deledit}(i, j) = \infty$. Specially, define $\mathit{deledit}(0, 0) = 0$, $\mathit{deledit}(0, j) = \infty$ for any $j \geq 1$, and $\mathit{deledit}(i, 0) = i$ for any $i \geq 1$.

Consider $i \geq 1, j \geq 1$. In general, if $s[i] = t[j]$, we have:

$$\mathit{deledit}(i, j) \quad = \quad \min \begin{cases} \mathit{deledit}(s[1..i-1], t[1..j-1]) \\ 1 + \mathit{deledit}(s[1..i-1], j). \end{cases}$$

whereas if $s[i] \neq t[j]$, we have:

$$\mathit{deledit}(i, j) \quad = \quad 1 + \mathit{deledit}(s[1..i-1], j).$$

Note that there are $O(n)$ choices for i and $O(1)$ choices for j. Dynamic programming therefore can be used to evaluate $\mathit{deledit}(n, 6)$ in $O(n)$ time.

Problem 4 (Longest Common Subsequence; Section 15.4 of the Textbook). Let σ and s be two strings such that $|\sigma| \leq |s|$. We call σ a *subsequence* of s if it is possible to turn s into σ by repeatedly deleting letters. For example, "hell" is a subsequence of "asdfhljeljlasfdf" but "hello" is not and neither is "hlle".

You are given two strings s, t with lengths m and n, respectively. Give an $O(mn)$-time algorithm to find a common subsequence of s and t that has the greatest length. For example, if $s =$ "algorithm" and $t =$ "logarithmic", a possible output can be "grithm".

Solution. For each $i \in [1, n]$ and $j \in [1, m]$, define $\mathit{lcs}(i, j)$ to be the greatest length of common subsequence of $s[1..i]$ and $t[1..j]$. Specially, define $\mathit{deledit}(0, 0) = 0$, $\mathit{deledit}(0, j) = 0$ for any $j \geq 1$, and $\mathit{deledit}(i, 0) = 0$ for any $i \geq 1$.

Consider $i \geq 1, j \geq 1$. In general, if $s[i] = t[j]$, we have:

$$\mathit{lcs}(i, j) \quad = \quad \max \begin{cases} 1 + \mathit{lcs}(i-1, j-1) \\ \mathit{lcs}(i-1, j) \\ \mathit{lcs}(i, j-1). \end{cases}$$

whereas if $s[i] \neq t[j]$, we have:

$$\mathit{lcs}(i, j) \quad = \quad \max \begin{cases} \mathit{lcs}(i-1, j-1) \\ \mathit{lcs}(i-1, j) \\ \mathit{lcs}(i, j-1). \end{cases}$$

There are $O(m)$ choices for i and $O(n)$ choices for j. Dynamic programming therefore can be used to evaluate $\mathit{lcs}(m, n)$ in $O(mn)$ time.

Remark: You can actually simplify the above recursive functions — you may refer to the textbook for details. But the simplification will not affect the running time.