ALGORITHMS, FALL 2018, HOMEWORK 8

Due Thursday, November 1 at 11:59pm.
Worth 1% of the final grade.
Recall that looking up algorithms online is not allowed.

1. Design any polynomial-time algorithm to get a LCS (not just the length of the LCS) of two strings of length n, using $O(n)$ space.

Note: The following problem will be part of homework 9, due on November 8. I recommend working on it soon because homework 9 will also include a problem on amortization, which will require some getting used to.

- Suppose that you have two strings, S_1, S_2, each consisting of characters from some small alphabet. The lengths of the strings are k and n respectively. You can assume that $n \geq k$ but not that k is necessarily some small value.

  If we alternate reading characters from one string and then the other, we create a "blend" of the two strings. For example if
  $S_1 = DYNOGRASOAMA$
  and
  $S_2 = AMICPRMMINGISZING$
  we can create this blend:
  DYNAMICPROGRAMMINGISSOAMAZING

  Note that the length of a blend is $n+k$, and there can be many blends.

  Now for the problem:
  Given three strings S_1, S_2 and C, determine if C is a blend of S_1 and S_2. If S_1 and S_2 have size k and n respectively, the time complexity should be $O(kn)$.

  You should address the following in your answer:
  Formulate the problem recursively, including an informal justification for its correctness. (How can you categorize subproblems that will be helpful to solve first?) How many distinct subproblems are there? Mention what you consider to be a base case. What is the size of the dynamic programming table that you will use, how do you fill it in, and where are the base cases and final answer located?