CS6033 Homework Assignment 11[*]

Due Dec 3rd at 5:00 pm

1. Show the longest common subsequence table, L, for the following two strings: $X =$ "notice", $Y =$ "encircle". What is a longest common subsequence between these strings?
2. Give a memoized version of LCS-LENGTH that runs in $O(mn)$ time.

   Show how your algorithm works on the strings "notice" and "encircle"
3. When computing the optimal binary search tree, we could have also created another table called $root[1..n][1..n]$ which holds which key which is the root of the optimal binary search tree for keys i through j. Fill in the tables w, e and $root$ for the optimal binary search tree algorithm presented in class[1] where $n = 7$ and the word frequencies are:

begin	6%
end	6%
while	22%
if	15%
else	10%
return	30%
for	24%

4. Prof. Thi N. Kalot suggests the following greedy strategy for the computing optimal binary search tree. Find the node with the highest probability and make that node the root. The recursively solve for the left and right subtrees using the same approach. Either prove this Prof. Thi N. Kalot's algorithm works or prove it doesn't work by providing a counter example.
5. (15 points) Question 15-11 on page 411 of CLRS with the following modification. Instead of providing an algorithm:

[*]Many of these questions came from outside sources.
[1]I have included the pseudo code on the posted slides.

- provide the recurrence formula
- create an example to show how your algorithm works

6. Scientists performing DNA sequence alignment often desire a more parametrized way of aligning two strings than simply computing a longest common subsequence between these two strings. One way to achieve such flexibility is to use the Smith-Waterman algorithm, which is generalization of the dynamic programming algorithm for computing a longest common subsequence so that it can handle weighted scoring functions. In particular, for two strings, X and Y, suppose we are given functions, defined on characters, a, from X, and b from Y , as follows:

   $M(a, b)$ = the positive benefit for a match, if $a = b$

   $M(a, b)$ =the negative cost for a mismatch, if $a \neq b$

   $I(a)$ = the negative cost for inserting a at a position in X

   $D(b)$ = the negative cost of deleting b from some position in Y.

   As a first step in solving this problem, create the recurrence relation that when given a string, X, of length n and a string, Y, of length m, finds the maximum-weight way of transforming X into Y, according to the above weight functions, using the operations of matching, substitution, insertion in X, and deletion from Y.

   How many subproblems does your recurrence formula compute? If you implemented your recurrence formula using dynamic programming how long would it take? Write your answer using big-Oh notation.

   Question modified from Goodrich, Michael T.; Tamassia, Roberto. Algorithm Design and Applications

7. You are taking another road trip!

   Since you are taking *Design and Analysis of Algorithms*, you are going to optimize your route.

   You will be traveling with your best friend. Since your friend owns the car, he has decide which route you will take. You have, however, convinced your friend you are the one to decide what hotels to stop at on your way (they all are mysteriously on the side of the road). You are going to optimize your route so it costs the least (you are a graduate student after all and need to save money).

   On your route are a series of hotels $h_1, h_2, \ldots, h_n$. The hotel h_i costs c_i. If you travel more than 300 miles per day, you realize you will spend more on *coffee*, specifically you will spend $(n/2)$ for n miles over 500 (your cousin crashes hard, and hiring someone to help move him into the hotel room is reasonably expensive)

   You would like to minimize your expenses.

   For this problem:

- provide the recurrence formula
- prove that the problem has optimal substructure
- use your recurrence formula to find the minimum cost to get to California which is $2,700$ miles away for the case where the hotels are at mile markers: $400, 850, 926, 1420, 1540, 1950, 2100, 2400$ and the hotels cost \$56, \$75, \$46, \$76,\$49, \$60, \$55, \$87.

In your answer define what your variables mean.

8. After a horrifying road trip with your friend to California. You now need to return home. You vow to study hard in class (and thus get a job) so next time you visit California you will be driving your own car (you were unaware your friend refused to drink coffee). After the horrifying experience on the trip down (you nearly crashed 42 times), you have revised your return plan. On your trip home, you refuse to travel more than 300 miles per day. You have the list of hotel from last time at mile markers $m_1, m_2, \ldots, m_n$. You wish to get home in as few days as possible.

   Design an algorithm and provide run time of your algorithm using big-Oh notation.

9. You and a friend are taking a class from professor E.Z. Ya. He has assigned a long group homework assignment. It is composed of a list of n problems (where n is an even number). Each problem q_i, has an "easiness" factor associated with it, e_i.

   To make sure you don't miss a problem you decide to start solving the problems from either end (i.e. you start by solving q_1 or q_n. If you chose to first solve q_1 then the next time you could choose q_2 or q_n. If you first chose q_n then the next time you could choose q_1 or q_{n-1}. Etc.)

   To finish quicker, you and your friend will simultaneously solve a problem. You get to choose the first problem (you) will solve. Your friend then choses which problem to solve.

   How do you ensure that when you are done, you have done the minimum amount of work? Design an efficient algorithm to solve this problem where you assume your friend will also be choosing to minimize his/her work load.

10. (3 bonus points) Think of a good exam question for the topic in the last lecture.

Our amazing GA Fan Bu has found the following problems you can use as practice problems. Do **not** turn in any of the problems below.

1. Only one transaction is allowed (buy and sell once) https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/

2. Unlimited transactions are allowed. But you have to sell before you buy again. https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/description/

3. At most 2 transactions are allowed. But you have to sell before you buy again. https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iii/description/

4. At most k transactions are allowed. But you have to sell before you buy again. https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iv/description/

5. Unlimited transactions are allowed with a transaction fee. But you have to sell before you buy again. https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-transaction-fee/description/

6. Unlimited transactions are allowed with a cooldown. But you have to sell before you buy again. https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-cooldown/

The first 2 questions are not really about dynamic programming, but they are easy to solve and related.