Spring 2019 CS372 Assignment #8 solutions.

1. Recall the longest common subsequence (LCS) problem discussed in class. Recall that c[i,j] is the length of LCS of X_i and Y_j, where X_i is the prefix of X of length i and Y_j is the prefix of Y of length j. LCS recursive solution is given by the following

$$c[i\,,j] = \begin{cases} c[i-1, j-1] + 1, & if\ x[i] = y[j] \\ \max(c[i, j-1], c[i-1, j]), & if\ x[i] \neq y[j] \\ 0, & if\ i = 0\ or\ j = 0 \end{cases}$$

Find the length of LCS of “AABCBABA” AND “BCBCAAB”, as well as an actual longest common subsequence. Show your work - draw a table and fill it in.

Answer:
X="AABCBABA", Y="BCBCAAB".

Y		0	1	2	3	4	5	6	7
X			B	C	B	C	A	A	B
0		0	0	0	0	0	0	0	0
1	A	0	0	0	0	0	1	1	1
2	A	0	0	0	0	0	1	2	2
3	B	0	1	1	1	1	1	2	3
4	C	0	1	2	2	2	2	2	3
5	B	0	1	2	3	3	3	3	3
6	A	0	1	2	3	3	4	4	4
7	B	0	1	2	3	3	4	4	5
8	A	0	1	2	3	3	4	5	5

The length of LCS is 5. "BCBAA" is a longest common subsequence.

2. Use dynamic programming algorithm to find edit distance between strings "STUDIES" and "SUCCESS". Show your work - draw a table and fill it in. (The algorithm is described in Section 6.3)

Answer:
X="STUDIES", Y="SUCCESS".

Y		0	1	2	3	4	5	6	7
X			S	U	C	C	E	S	S
0		0	1	2	3	4	5	6	7
1	S	1	0	1	2	3	4	5	6
2	T	2	1	1	2	3	4	5	6
3	U	3	2	1	2	3	4	5	6
4	D	4	3	2	2	3	4	5	6
5	I	5	4	3	3	3	4	5	6
6	E	6	5	4	4	4	3	4	5
7	S	7	6	5	5	5	4	3	4

Edit distance is 4.

3. Exercise 6.1. Hint: Subproblems are D(i) ($0 \le i \le n$) where D(i) is the largest sum of a (possibly empty) contiguous subsequence ending exactly at position i. You need to write a recursion which can be used to compute D(i), explain your algorithm and show that its running time is linear.

Answer:
Subproblems: Define an array of subproblems D(i) for $0 \le i \le n$. D(i) will be the largest sum of a (possibly empty) contiguous subsequence ending exactly at position i.
Algorithm and Recursion: The algorithm will initialize D(0) = 0 and update the D(i)'s in ascending order according to the rule:

$$D(i) = \max\{0, D(i-1) + a_i\}$$

The largest sum is then given by the maximum element $D(i*)$ in the array D. The contiguous subsequence of maximum sum will terminate at $i*$. Its beginning will be at the first index $j \le i*$ such that $D(j-1) = 0$, as this implies that extending the sequence before j will only decrease its sum.
Correctness: The contiguous subsequence of largest sum ending at i will either be empty or contain a_i. In the first case, the value of the sum will be 0. In the second case, it will be the sum of ai and the best sum we can get ending at $i-1$, i.e. $D(i-1) + a_i$. Because we are looking for the largest sum, D(i) will be the maximum of these two possibilities.
Running Time: The running time for this algorithm is O(n), as we have n subproblems and the solution of each can be computed in constant time. Moreover, the identification of the optimal subsequence only requires a single O(n) time pass through the array D.

4. Exercise 6.7. Hint: Subproblems are L(i,j) ($1 \le i \le j \le n$) where L(i,j) is the length of the longest palindromic subsequence of string x[i,…,j]. You need to write a recursion which can be used to solve the subproblems, explain your algorithm and show that its running time is $O(n^2)$.

Answer:
Subproblems: Define variables L(i, j) for all $1 \le i \le j \le n$ so that, in the course of the algorithm, each L(i, j) is assigned the length of the longest palindromic subsequence of string x[i,…,j].
Algorithm and Recursion: The recursion will then be:

$L(i, j) = L(i+1, j-1) + 2$ if the first and last characters (x_i and x_j) are the same ($j \ge i+2$),

$L(i, j) = \max\{L(i+1, j), L(i, j-1)\}$ if the first and last characters are not the same ($j \ge i+1$).

The initialization is the following:
for all i, $1 \le i \le n$, $L(i, i) = 1$
for all i, $1 \le i \le n-1$, $L(i, i+1) = 2$ if both characters (x_i and x_{i+1}) are the same.
Correctness and Running Time: Consider the longest palindromic subsequence s of x[i,…,j] and focus on the elements x_i and x_j . There are then three possible cases:
– If both x_i and x_j are in s then they must be equal and $L(i, j) = L(i+1, j-1) + 2$
– If x_i is not a part of s, then $L(i, j) = L(i+1, j)$.
– If x_j is not a part of s, then $L(i, j) = L(i, j-1)$.
Hence, the recursion handles all possible cases correctly. The running time of this algorithm is $O(n^2)$, as there are $O(n^2)$ subproblems and each takes O(1) time to evaluate according to our recursion.

5. Let X, Y, Z be 3 strings. Design a dynamic programming algorithm that would find the length of the longest common subsequence of X, Y, and Z.
(a) Define a suitable subproblem.
(b) Give a recursive solution to the subproblem.
(c) Give a pseudocode for a dynamic programming algorithm that solves the problem.
(d) Analyze the running time of your algorithm.

Answer:
The solution to the problem of the longest common subsequence for three strings is essentially identical to the solution with 2 strings.
(a) Subproblems:
define T(i, j, k) to be the length of the longest common subsequence of X_i, Y_j, and Z_k.
(b) Recursive solution:

$$T(i,j,k) = \begin{cases} 0 \quad if\ i = 0\ or\ j = 0\ or\ k = 0 \\ T(i-1, j-1, k-1) + 1 \qquad if\ x[i] = y[j] = z[k] \\ \max(T(i,j,k-1), T(i,j-1,k), T(i-1,j,k) \text{ otherwise} \end{cases}$$

(c) Let a be the length of X, b be the length of Y, and c be the length of Z.

```
for j = 0 to b
        for k = 0 to c
                T(0,j,k)=0
for k = 0 to c
        for i = 0 to a
                T(i,0,k)=0
for i = 0 to a
        for j = 0 to b
                T(i,j,0)=0
for i = 1 to a
        for j = 1 to b
                for k = 1 to c
                        if X(i) = Y(j) = Z(k) then
                                T(i,j,k)=T(i-1,j-1,k-1) + 1
                        else
                                T(i,j,k)= MAX(T(i,j,k-1),T(i, j-1, k), T(i-1, j, k))
```

(d) Running time is O(abc) or O(n^3) where n is MAX(a,b,c).