CSCI 532 – Algorithm Design
Assignment 7

Name: Badarika Namineni **CWID:** 50233279

Question 1. Show that equation (15.4) follows from equation (15.3) and the initial condition T (0) = 1

Solution:

The equation in (15.4) is T (n) = 2^n and the equation in (15.3) is T (n) = $1+\sum_{j=0}^{n-1} T(j)$
Let's say, n =0, T (o) = $2^0 = 1$
We apply inductive hypothesis on (15.3) to get

T (n) = $1+\sum_{j=0}^{n-1} T(j)$

= $1+\sum_{j=0}^{n-1} 2\hat{}j$

= $1+2^n-1/2-1 = 1+ 2^n – 1 =$ **2^n**

Question 2. The Fibonacci numbers are defined by recurrence (3.22). Give an O(n)-time dynamic-programming algorithm to compute the nth Fibonacci number. Draw the subproblem graph. How many vertices and edges are in the graph?

Solution:

Fibonacci numbers are formed by adding 2 previous numbers in the sequence.

The subproblem graph consists of n + 1 vertices, has two leaving edges: to vertex v_{i-1} and to vertex v_{i-2} . No edges leave vertices v0 and v1. Thus, the subproblem graph has 2n -2 edges.

Question 3. Give a recursive algorithm MATRIX-CHAIN-MULTIPLY (A, s, i, j) that actually performs the optimal matrix-chain multiplication, given the sequence of matrices (A1, A2…An), the s table computed by MA TRIX-CHAIN-ORDER, and the indices i and j. (The initial call would be MATRIX-CHAIN-MULTIPLY (A, s, 1, n)

Solution:

```
MATRIX_CHAIN_MULTIPLY(A,s,i,j)
    If (i == j)
        return A[i]
    if (j == i+1)
        return A[i]*A[j];
    else
        B1 = MATRIX_CHAIN_MULTIPLY (A,s,i,S [i,j])
        B2 = MATRIX_CHAIN_MULTIPLY (A,s,S [i,j]+1,j)
        return B1*B2
```

Question 4. Draw the recursion tree for the MERGE-SORT procedure from Section 2.3.1 on an array of 16 elements. Explain why memoization fails to speed up a good divide-and-conquer algorithm such as MERGE-SORT.

Solution:
The recursion tree will have [1..n] as its root, and at any node [i..j] will have [i..(j-i)/2] and [(j-i)/2+1..j] as its left and right children respectively.

The merge-sort procedure performs at most a single call to any pair of indices of the array that is being sorted. In other words, the subproblems do not overlap and therefore memoization will not improve the running time.

Question 5. Determine an LCS of (1, 0, 0, 1, 0, 1, 0, 1) and (0, 1, 0, 1, 1, 0, 1, 1, 0).

Solution:

Given two sequences X and Y, we say that a sequence Z is a common subsequence of X and Y if Z is a subsequence of both X and Y.

An LCS is {1 ,0, 1, 0, 1, 0}. A concise way of seeing this is by noticing that the first list contains a “00” while the second contains none, Also, the second list contains two copies of “11” while the first contains none. In order to reconcile this, any LCS will have to skip at least three elements. Since we managed to do this, we know that our common subsequence was maximal.

Question 6. Suppose that we are given a directed acyclic graph G = (V, E) with real-valued edge weights and two distinguished vertices s and t. Describe a dynamic programming approach for finding a longest weighted simple path from s to t. What does the subproblem graph look like? What is the efficiency of your algorithm?

Solution:

Given a directed acyclic graph G = (V, E), we need to first topologically sort this graph G, then compute the longest path to the vertices one by one in the sorted order. The start node s comes the first, whose distance is initialized to be zero, and the last node is t.

At every iteration, a vertex is picked to compute the longest path by comparing u.distance+w(u,v) for every vertex u which is connected to v through an edge (u,v). That is, every iteration willbe responsible for a subproblem, which is to find the longest weighted simple path from s upto the current node v.

Once all the vertices have been calculated, the longest weighted simple path from s to t is the t.distance, or the maximum distance among all the vertices. This algorithm topologically sorts the graph (which takes O(V+E)), and visits every vertex while checking all the edges during the computation of the longest paths for every vertex, O(V+E). Overall, this algorithm takes O(V+E)