After Lecture 15 & 16 – Answer any questions on HW4 (due today)
Practice Problems (all taken from previous exams)

1. In dynamic programming, the technique of storing the previously calculated values is called ____________

   a) Saving value property
   b) Storing value property
   c) Memoization
   d) Mapping

2. What is the time complexity of the brute force algorithm used to find the longest common subsequence for sequence length m and sequence length n $(m < n)$?

   a) $O(mn)$
   b) $O((mn)^2)$
   c) $O(n2^m)$
   d) $O(2^m 2^n)$

3. When dynamic programming is used, it takes less time compared to algorithmic methods that don't utilize overlapping subproblems.

   a) True.
   b) False.

4. Using the dynamic programming solution, determine an LCS of $\{1, 0, 0, 1, 0, 1, 0, 1\}$ and $\{0, 1, 0, 1, 1, 0, 1, 1, 0\}$. Show all your work.

5. Given a sequence of n numbers a_1, a_2, a_3, $\ldots$, a_n (some of them might be negative) stored in an array, we want to find two indicies $i \leq j$ such that the sum of the numbers from a_i to a_j is maximum, among all possible i j pairs $1 \leq i \leq j \leq n$.

   5a) Write pseudocode to sum each contiguous subsequence (from a_i to a_j) and keep track of the maximum one. What is the runtime of your algorithm? The runtime is $O(n^2)$

Algorithm 8.1 Maximum Subsequence

```
function MaxSubsequence
    bestval ← −∞
    for i ← 1 . . . n do
        sumCurrent ← a[i]
        if sumCurrent > bestval then
            bestval ← sumCurrent
            besti ← i
            bestj ← i
        end if
        for j ← i + 1 . . . n do
            sumCurrent ← sumCurrent + a[j]
            if sumCurrent > bestval then
                bestval ← sumCurrent
                besti ← i
                bestj ← j
            end if
        end for
    end for
    return bestval, besti, bestj
end function
```

5b) Now find an $O(n)$ algorithm. Give pseudocode.

Algorithm 8.2 Improved Maximum Subsequence

```
function ImprovedMaximumSubsequence
    M[j] ← max sum over all contiguous sequences ending at a[j]
    a[j] ← either extends the previous contiguous sequence, or a[j] starts a new contiguous sequence
    M[j] ← max{M[j − 1] + a[j], a[j]}
end function
```

6. Prove that a binary tree that is not full (every node has 0 or 2 children) cannot correspond to an optimal prefix code. An optimal prefix code is a predix code that gives the shortest possible encoded file length. If we have a prefix code that corresponds to a binary tree that is not full, let n be a node that only has 1 child. Then we could form another binary tree by removing n and moving up n's child. The codewords of all the characters that were descendants of n have now all be decreased by 1, and so the original binary tree could not correspond to an optimal prefix code.