Tutorial 6, Design and Analysis of Algorithms, 2019
Use *Dynamic Programming* to design the algorithms

1. Solve the following instance of the *Matrix Chain Multiplication Problem:* $< A_{1\times 2}, B_{2\times 3}, C_{3\times 4}, D_{4\times 5}, E_{5\times 6}, F_{6\times 7} >$.

2. Describe the subproblem graph for matrix-chain multiplication with an input chain of length n. How many vertices does it have? How many edges does it have, and which edges are they? Draw the subproblem graph for input $< A, B, C, D >$.

3. Prove that the number of different binary trees with n nodes is
$$\frac{1}{n+1}\binom{2n}{n}$$
.

4. Give an $O(n^2)$-time algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers. Show the working of your algorithm for the input $< 9, 4, 1, 3, 4, 2, 9, 7, 3, 4 >$.

5. A palindrome is a nonempty string over some alphabet that reads the same forward and backward. Examples of palindromes are all strings of length 1, `civic`, `racecar`, and `aibohphobia` (fear of palindromes). Give an efficient algorithm to find the longest palindrome that is a subsequence of a given input string. For example, given the input `character`, your algorithm should return `carac`. What is the running time of your algorithm? Show the working of your algorithm on input `character`.

6. Determine the cost and structure of an optimal binary search tree for a set of $n = 4$ keys $K = \{\text{do}, \text{if}, \text{int}, \text{while}\}$ with the following probabilities:
$(p_i)_{i=1}^{4} = \left(\frac{1}{16}, \frac{1}{16}, \frac{3}{16}, \frac{3}{16}\right)$
$(q_i)_{i=0}^{4} = \left(\frac{1}{16}, \frac{1}{16}, \frac{2}{16}, \frac{3}{16}, \frac{1}{16}\right)$

7. Let $G = (V, E)$ be a directed graph with nodes $v_1, ..., v_n$. We say that G is an ordered graph if it has the following properties:
(1) Each edge goes from a node with a lower index to a node with a higher index. That is, every directed edge has the form (v_i, v_j) with $i < j$.
(2) Each node except v_n has at least one edge leaving it. That is, for every node v_i, $i = 1, 2, ..., n-1$, there is at least one edge of the form (v_i, v_j).
Design an efficient algorithm that takes an ordered graph G and returns the length of the longest path that begins at v_1 and ends at v_n (the length of a path is the number of edges in the path). Find the time complexity of your algorithm. Show the working of your algorithm on the example graph in Figure 1.

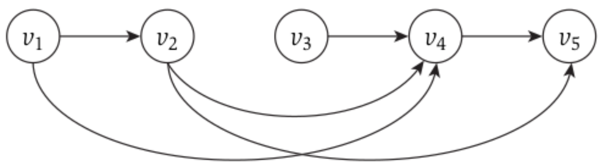

Figure 1: Example graph for problem 7.