You are expected to work on this assignment on your own

Use pseudocode, Python-like or English to describe your algorithms. Absolutely no C++/C/Java

When designing an algorithm, you are allowed to use any algorithm or data structure we explained in class, without giving its details, unless the question specifically requires that you give such details

Always remember to analyze the time complexity of your algorithms

Homework has to be submitted electronically on Gradescope by the deadline. No late assignments will be accepted

Unless otherwise noted, for all questions about graphs, $n = |V|$ is the number of vertices/nodes, and $m = |E|$ is the number of edges/links
**Problem 1.** (25 points) [Dynamic Programming]

Given an array \( A = \{a_1, a_2, \ldots, a_n\} \) of integers, we say that a subsequence \( \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\} \) is (monotonically) increasing if for every \( i_s < i_t \), we have \( a_{i_s} < a_{i_t} \). Given an array \( A \) of size \( n \), we want to compute the length of the longest increasing subsequence (LIS) in \( A \). For instance, if \( A = \{9, 5, 2, 8, 7, 3, 1, 6, 4\} \) the length of the LIS is 3, because \( (2, 3, 4) \) (or \( (2, 3, 6) \)) are LIS of \( A \). Give a \( O(n^2) \) dynamic programming algorithm for this problem\(^1\). Analyze the time- and space-complexity of your solution.

**Answer:**

\(^1O(n \log n)\) is possible, but the easier \( O(n^2) \) solution suffices
**Problem 2.** (25 points) [Graph Traversals on Graphs]

You are given a weighted 2-banded graph, which is a undirected graph $G = (V, E)$ with $2n$ vertices. Each vertex is labeled as $(i, j)$ where $i = 1$ or $i = 2$, and $j = 1, \ldots, n$. There is an edge between $(i, j)$ to $(i, j + 1)$ for every $i = 1, 2$ and $j = 1, \ldots, n - 1$. There is also an edge between $(1, j)$ and $(2, j)$ for all possible $j$’s. Below is such a graph for $n = 4$. Furthermore, each vertex has a weight, which is a positive integer.

An *independent set* $W$ is a subset of $V$, such that for any two vertices $u, v \in W$, there is no edge between $u$ and $v$ in the graph $G$. The total weight of a set $W$ is the sum of the weights of all vertices in $W$; the total weight of the empty set is 0. There are many possible independent sets in $G$; the independent set with the largest total weight is the *maximum independent set*.

Give a $O(n)$ dynamic programming algorithm that, given a weighted two-banded graph, outputs the weight of its maximum independent set (it is not necessary to find the maximum independent set, just compute the weight.)

**Answer:**
Problem 3. (25 points) [Dynamic Programming on Graphs]

Given a directed graph with non-negative integer edge weights, a pair of vertices $s$ and $t$, and integers $K$ and $W$, describe a dynamic-programming algorithm for deciding whether there exists a path from $s$ to $t$ that has total weight $W$ and uses exactly $K$ edges. Your algorithm should run in time $O((n + m)WK)$. Analyze the time- and space-complexity of your solution. Hint: You will have to define a three-dimensional table for the recurrence relation.

Answer:
Problem 4. (25 points) [Divide-and-conquer on Graphs]

Let $G = (V, E)$ be an undirected graph. A *triangle* in $G$ is a cycle consisting of exactly three vertices (or, equivalently, three edges). Suppose that $G$ is represented as an adjacency matrix. Give an algorithm to determine whether $G$ contains any triangle in $O(n \log_2 n)$ time.

Answer: