Collaboration is allowed, but you must write your own solutions. Proving correctness of algorithms is a must.

**Problem 1 (Dynamic Programming)**
Given a tree with (possibly negative) weights assigned to its vertices, give a polynomial time algorithm to find a subtree with maximum weight. Note that a *subtree* is a connected subgraph of a tree.

**Problem 2**
Let $G = (V, E)$ be a directed acyclic graph (i.e. it does not contain any directed cycle).

1. Prove that the graph must have a vertex $t$ that has no outgoing edge.

2. Suppose $|V| = n$. A *topological ordering* of the acyclic graph is a labeling of its vertices by integers from 1 to $n$ such that
   - Any two distinct vertices receive distinct labels.
   - Every (directed) edge goes from a vertex with a lower label to a vertex with a higher label.

   Give a polynomial time algorithm to find a topological ordering of the graph.

3. Fix a node $t$ that has no outgoing edge. For every node $v \in V$, let $P(v)$ be the number of distinct paths from $v$ to $t$. Define $P(v) = 0$ if no such path exists and define $P(t) = 1$ for convenience. Give a polynomial time algorithm to compute $P(v)$ for every node $v$.

**Problem 3 (Dynamic Programming)**
Solve: [Kleinberg Tardos]: Chapter 6, Problem 21, on page 330.

**Problem 4 (Dynamic Programming)**
Solve: [Kleinberg Tardos]: Chapter 6, Problem 22, on page 330.

**Problem 5 (Dynamic Programming)**
Solve: [Kleinberg Tardos]: Chapter 6, Problem 28, on page 334.
Problem 6 (Dynamic Programming)

An independent set $I$ in a graph is called maximal if the graph does not contain an independent set $I'$ such that $I \subseteq I'$ and $|I| < |I'|$.

Given a tree on $n$ vertices, and an integer $0 \leq k \leq n$, give a polynomial time algorithm to determine whether the tree has a maximal independent set of size $k$. (Hint: Design an algorithm that solves the problem for all possible values of $k$.)

Problem 7


Note: In (a), SEARCH need not run in logarithmic time. In (b), define an appropriate potential function so that insertion runs in amortized $O(1)$ time. In (c), one does not expect the implementation to be very efficient.

Problem 8


Note: Assume that $\alpha$ is strictly larger than $\frac{1}{2}$ (and strictly less than 1). Ignore deletions (as everything would be similar to insertions).