Problem 1 (10 Points) - Persistent Vertices

Let $G = (V, E)$ be a finite directed graph (without self-loops). Call a vertex v *persistent* if there exists an infinite directed walk starting at v. Vertices and edges may be repeated in a walk.

(a) Characterize persistent vertices in terms of directed cycles.

(b) Design an $O(n + m)$-time algorithm that finds all persistent vertices.

Example instance. Let

$$V = \{a, b, c, d, e\},$$

and

$$E = \{(a, b), (b, c), (c, b), (d, e)\}.$$

Expected answer for the example. The persistent vertices are

$$\{a, b, c\}.$$

For example, an infinite walk from a is

$$a, b, c, b, c, b, c, \ldots.$$

Vertex d has positive out-degree but is not persistent: every walk from d reaches e and stops.

Problem 2 (10 Points) - Greedy

There are n jobs, each requiring exactly one unit of processing time on a single machine. Job i has an integer deadline d_i, where $1 \leq d_i \leq n$, and profit $p_i > 0$.

A job earns its profit only if it finishes by its deadline. You may select a subset of the jobs and schedule them in the unit-length slots

$$[0, 1], [1, 2], \ldots, [n-1, n].$$

Consider the following algorithm:

- Process jobs in nonincreasing order of profit.
- For each job, place it in the latest currently empty slot that finishes by its deadline.
- If no such slot exists, discard the job.

(a) Prove that this algorithm maximizes total profit.

(b) Give an $O(n^2)$-time implementation.

Example instance. The jobs are

Job	d_i	p_i
A	2	100
B	1	90
C	2	20

Expected answer for the example. An optimal schedule places B in $[0, 1]$ and A in $[1, 2]$, earning total profit

$$90 + 100 = 190.$$

Job C is discarded.

Problem 3 (10 Points) - Counting Shortest Paths

Let $G = (V, E)$ be a directed graph with strictly positive edge weights and source s. You are also given an integer $M \geq 2$.

Design an algorithm that computes, for every vertex v:

- The shortest-path distance from s to v.
- The number of distinct shortest paths from s to v, modulo M.

Your algorithm should run in $O((n + m) \log n)$ time, assuming arithmetic modulo M takes constant time.

Example instance. Let $M = 3$, $V = \{s, a, b, t\}$, and let the weighted edges be

$$(s, a, 1), \quad (s, b, 1), \quad (a, t, 2), \quad (b, t, 2), \quad (s, t, 3).$$

Expected answer for the example. The required output is

v	$d(v)$	Number of shortest paths modulo 3
s	0	1
a	1	1
b	1	1
t	3	0

The three shortest paths to t are

$$s \to t, \qquad s \to a \to t, \qquad s \to b \to t.$$

The empty path is counted as the unique shortest path from s to itself.

Problem 4 (10 Points) - Minimum Spanning Tree with required edges

Let $G = (V, E)$ be a weighted undirected graph and let $R \subseteq E$ be a subset of edges that forms a forest and that must belong to the spanning tree.

Design an $O(m \log n)$-time algorithm that finds a minimum-weight spanning tree T satisfying $R \subseteq T$ or reports that no such spanning tree exists.

Example instance. Let $V = \{a, b, c, d\}$, with weighted edges

$$(\{a, b\}, 1), \quad (\{b, c\}, 2), \quad (\{a, c\}, 4), \quad (\{c, d\}, 1), \quad (\{b, d\}, 5).$$

The required set is $R = \{\{a, b\}, \{a, c\}\}$.

Expected answer for the example. The optimal feasible tree is

$$T = \{\{a, c\}, \{a, b\}, \{c, d\}\},$$

with total weight

$$4 + 1 + 1 = 6.$$

Note that if $\{a, c\}$ were not required the MST would have weight 4.