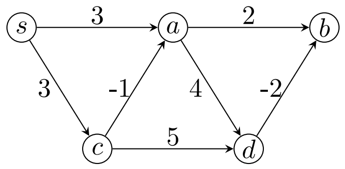
Problem 1 (30 points).

1. Run the dynamic programming algorithm for the single-source shortest path problem on the following graph with s being the source vertex: write and fill the dynamic programming table *dist*.
2. Run the Bellman-Ford algorithm on the following graph with s being the source vertex: give the value of array *dist* after each iteration.
3. Run the Floyd-Warshall algorithm on the following graph: write the matrix corresponding to $dist(\cdot,\cdot,0)$ and $dist(\cdot,\cdot,1)$.



Problem 2 (20 points). In the Bellman-Ford algorithm (and the dynamic programming algorithm), we can maintain an array *prev* to store the previous vertex in the shortest path for each vertex (as we did in the Dijkstra's algorithm), so that we can reconstruct the shortest path (in addition to the distance) for each vertex. Specifically, when we relax edge (u,v) we set $prev[v]$ to u if we have $dist[u] + l(u,v) < dist[v]$.

1. Prove that edges $\{(v, prev[v]) \mid v \in V\}$ form a tree.
2. Compute $prev[v]$ for every vertex $v \in V$ in the graph used in Problem 1.

Problem 3 (10 points). Given directed graph $G = (V,E)$, edge length $l(e)$ for $e \in E$, vertex $s \in V$, the dynamic programming algorithm (and the Bellman-Ford algorithm) can be used to determine whether there exists negative cycles in G that can be reached from s. Design a new algorithm for the following problem: given directed graph $G = (V,E)$ and edge length $l(e)$ for $e \in E$ to decide whether G contains negative cycles.

Problem 4 (20 points). Given a directed acyclic graph $G = (V,E)$, vertex $s \in V$, design a dynamic programming algorithm to compute the number of distinct paths from s to v for any $v \in V$. You should define subproblems, write recursion, give the pseudo-code, and analyze the running time.

Problem 5 (20 points). Shortest path algorithms can be applied in currency trading. Let $c_1, c_2, \cdots, c_n$ be various currencies. For any two currencies c_i and c_j, there is an exchange rate $r_{i,j}$; this means that you can purchase $r_{i,j}$ units of currency c_j in exchange for one unit of c_i. These exchange rates satisfy the condition that $r_{i,j} \cdot r_{j,i} < 1$, so that if you start with a unit of currency c_i, change it into currency c_j and then convert back to currency c_i, you end up with less than one unit of currency c_i (the difference is the cost of the transaction).

1. Give an efficient algorithm for the following problem: given a set of exchange rates $r_{i,j}$, and two currencies s and t, find the most advantageous sequence of currency exchanges for converting currency s into currency t. Toward this goal, you should represent the currencies and rates by a graph whose edge lengths are real numbers.

The exchange rates are updated frequently. Occasionally the exchange rates satisfy the following property: there is a sequence of currencies $c_1, c_2, \cdots, c_k$ such that $r_{1,2} \cdot r_{2,3} \cdot r_{3,4} \cdots r_{k-1,k} \cdot r_{k,1} > 1$. This means that by starting with a unit of currency c_1 and then successively converting it to currencies c_2, c_3, $\cdots$, c_k, and finally back to c_1, you would end up with more than one unit of currency c_1.

2. Give an efficient algorithm for detecting the presence of such an anomaly. Use the graph representation you found above.