

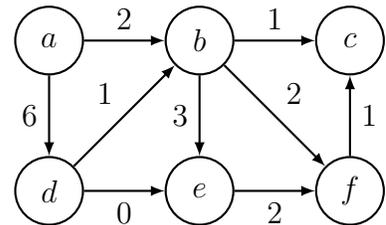
## Problem Set 6

Please write your solutions in the  $\text{\LaTeX}$  and Python templates provided. Aim for concise solutions; convoluted and obtuse descriptions might receive low marks, even when they are correct.

### Problem 6-1. [15 points] Dijkstra Practice

Consider weighted graph  $G = (V, E, w)$  below, which is acyclic and has nonnegative edge weights.

- (a) Run both DAG Relaxation and Dijkstra on  $G$  from vertex  $a$ . Each algorithm tries to relax every edge  $(u, v) \in E$ , but does so in a different order. For each algorithm, write down a list of all edges in **relaxation order**: the order the algorithm tries to relax them. If there is ambiguity on which vertex or outgoing adjacency to process next, process them in **alphabetical order**.



- (b) List  $\delta(a, v)$  for each  $v \in V$  (via either algorithm).

### Problem 6-2. [10 points] Short Circuits

Given a weighted directed graph  $G = (V, E, w)$  and vertex  $s \in V$ , with the property that, for every vertex  $v \in V$ , some minimum-weight path from  $s$  to  $v$  traverses at most  $k$  edges, describe an algorithm to find the shortest-path weight from  $s$  to each  $v \in V$  in  $O(|V| + k|E|)$  time.

### Problem 6-3. [20 points] Dynamite Detonation

Superspy Bames Jond is fleeing the alpine mountain lair of Silvertoe, the evil tycoon. Bames has stolen a pair of skis and a trail map listing the mountain's clearings and slopes ( $n$  in total), and she wants to ski from the clearing  $L$  by the lair to a clearing  $S$  where a snowmobile awaits.

- Each **clearing**  $c_i \in C$  has an integer **elevation**  $e_i$  above sea level.
- Each **slope**  $(c_i, c_j, \ell_{ij})$  connects a pair of clearings  $c_i$  and  $c_j$  with a **monotonic** trail (strictly decreasing or increasing in elevation) with positive integer length  $\ell_{ij}$ . Bames doesn't have time to ski uphill, so she will only traverse slopes so as to decrease her elevation.
- On her way up the mountain, Bames laced the mountain with **dynamite** at known locations  $C_D \subset C$ . Detonating the dynamite will immediately change the elevation of clearings  $c_i \in C_D$  by a known amount, from  $e_i$  to a lower elevation  $e'_i < e_i$ . The **detonator** exists at clearing  $D \in C$  (where there is no dynamite). If she reaches clearing  $D$ , she can choose to detonate the dynamite before continuing on.

Given Bames' map and dynamite data, describe an  $O(n)$ -time algorithm to find the minimum distance she must ski to reach the snowmobile (possibly detonating the dynamite along the way).

**Problem 6-4.** [10 points] **Conservative Cycles**

In Wentonian physics, a force field acts on the motion of a particle by requiring a certain positive or negative amount of **work** to move along any path. A force field is **conservative** if the total sum of work is zero **along every closed loop**. Given a discrete force field represented by  $n$  possible particle locations  $\ell_i$  and  $O(n)$  possible particle transitions<sup>1</sup>, where transition  $(\ell_i, \ell_j, w_{ij})$  moves a particle along a **directed** path from location  $\ell_i$  to location  $\ell_j$  requiring (positive or negative) integer work  $w_{ij}$ , describe an  $O(n^2)$ -time algorithm to determine whether the force field is conservative.

**Problem 6-5.** [20 points] **SparkPlug Derby**

LightQueen McNing is a race car that wants to drive to California to compete in a road race: the annual SparkPlug Derby. She has a map depicting:

- the  $n$  intersections in the country, where each intersection  $x_i$  is marked with its positive integer **elevation**  $e_i$  and whether it contains a **gas station**; and
- the  $r$  roads connecting pairs of them, where each road  $r_j$  is marked with the positive integer  $t_j$  denoting the **driving time** it will take LightQueen to drive along it in either direction.

Some intersections connect to many roads, but the average number of roads at any intersection is less than 5. LightQueen needs to get from Carburetor Falls at intersection  $s$  to the SparkPlug Derby race track at intersection  $t$ , subject to the following conditions:

- LightQueen’s gas tank has a positive integer **capacity**  $g < n$ : it can hold up to  $g$  units of gas at a time (starting full). Along the way, she can refill her tank (by any integer amount) at any intersection marked with a gas station. It takes exactly  $t_G$  time for her to fill up 1 unit of gas.
- LightQueen uses gas only when **driving uphill**. Specifically, if she drives on a road from intersection  $x_i$  to  $x_j$  at elevations  $e_i$  and  $e_j$  respectively, LightQueen will use exactly  $e_j - e_i$  units of gas to travel along it if  $e_j > e_i$ , and will use zero units of gas otherwise.

Given LightQueen’s map, describe an  $O(n^2 \log n)$ -time algorithm to return a fastest route to the race that keeps a **strictly positive** amount of gas in her tank at all times (if such a route exists).

**Problem 6-6.** [25 points] **Johnson’s Algorithm**

In this problem, you will implement Johnson’s Algorithm to compute all-pairs shortest-path weights in a general weighted directed graph  $G$ , as described in Lecture 14. The input to your algorithm will be: a positive integer  $n$  representing the number of vertices of your graph identified by consecutive integers 0 to  $n - 1$ , and a tuple  $S$  of triples, where each triple  $(u, v, w)$  corresponds to a directed edge from vertex  $u$  to  $v$  of weight  $w$ . Your output should be a length- $n$  tuple  $D$  of length- $n$  tuples where  $D[u][v] = \delta(u, v)$  for all  $u, v \in \{0, \dots, n - 1\}$ , except when the input graph contains a **negative-weight cycle** when you should return `None`.

Please implement the `johnson(n, S)` function in the template code provided. Your code template contains working implementations of Bellman–Ford and Dijkstra (using a binary heap as a priority queue) modified from the recitation notes. Note that you will have to construct your own adjacency list and weight function if you would like to use this code. You can download the code template and some test cases from the website.

<sup>1</sup>Note that a transition from  $\ell_i$  to  $\ell_j$  does not imply the existence of a transition from  $\ell_j$  to  $\ell_i$ .

MIT OpenCourseWare  
<https://ocw.mit.edu>

6.006 Introduction to Algorithms  
Spring 2020

For information about citing these materials or our Terms of Use, visit: <https://ocw.mit.edu/terms>