DSC 40B - Homework 08

Due: Wednesday, March 3

Write your solutions to the following problems by either typing them up or handwriting them on another piece of paper. Unless otherwise noted by the problem's instructions, show your work or provide some justification for your answer. Homeworks are due via Gradescope on Wednesday at 11:59 p.m.

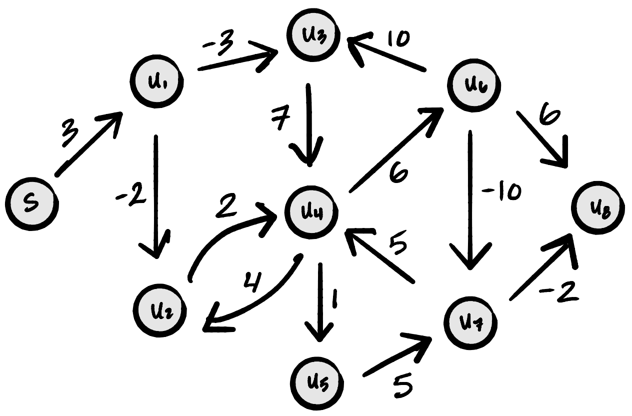
Important!: We cannot accept slip days for this homework! This is because the midterm is on Thursday, and we'd like to release the solutions on Wednesday at midnight so that they are available for you to look at before starting the exam.

This homework is somewhat shorter than a typical homework due to the exam.

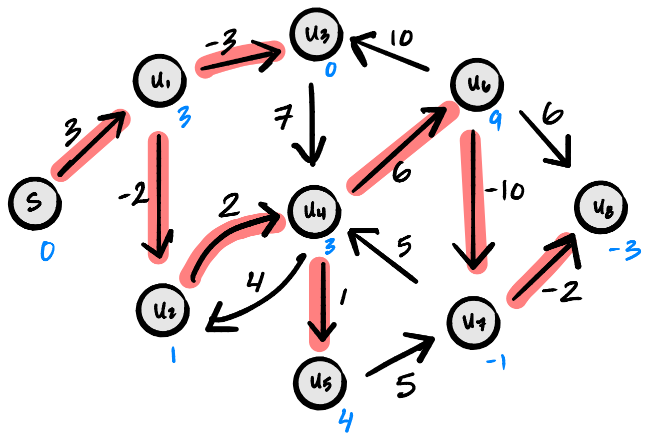
Problem 1. *(Eligible for Redemption)*

Run Bellman-Ford on the following graph using node s as the source. Below each node u, write the shortest path length from s to u. Mark the predecessor of u by highlighting it or making a bold arrow. You can assume that `graph.edges` produces the graph's edges in the following order:

$$(u_2, u_4), (u_7, u_8), (u_6, u_3), (s, u_1), (u_1, u_2), (u_4, u_5), (u_1, u_3),$$
$$(u_3, u_4), (u_6, u_7), (u_4, u_6), (u_5, u_7), (u_7, u_4), (u_4, u_2), (u_6, u_8)$$



Solution:



Problem 2. *(Eligible for Redemption)*

Suppose $G = (V, E, \omega)$ is a directed, weighted graph, and let s, u, and v be three nodes in the graph. Suppose we know that the shortest path distance from s to u is 100. If the weight of the edge (u, v) is -5, is it possible that the shortest path distance from s to v is 96? Why or why not?

Solution: It is not possible. Consider the path from s to v which is constructed by taking a shortest path from s to u and appending the edge (u, v). The total weight of this path is $100 + (-5) = 95$. Therefore, the shortest path distance from s to v is at most 95; it cannot be 96.

Problem 3. *(Eligible for Redemption)*

Dijkstra's Algorithm may fail if a graph has negative edge weights, but your friend has proposed a way of using Dijkstra's Algorithm to find shortest paths in such graphs. Suppose that $G = (V, E, \omega)$ is a weighted graph in which some edges are negative. Your friend's idea is to:

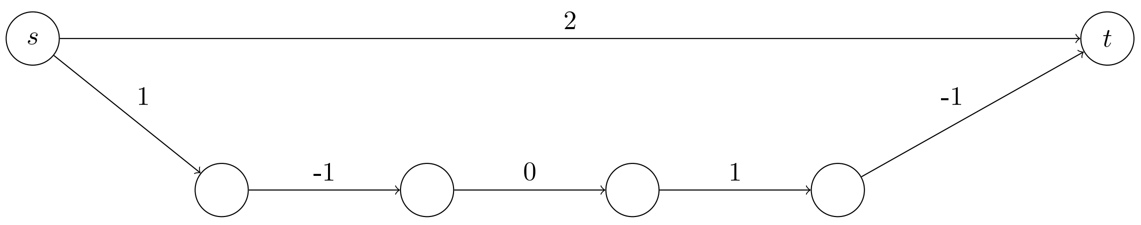
1. Pick a large constant c and add it to every edge weight so that the weight of every edge becomes positive. This results in a new weighted graph, $H = (V, E, \omega')$, where $\omega'(u, v) = \omega(u, v) + c$.
2. Run Dijkstra's algorithm on the new graph H and return the shortest paths as a dictionary of predecessors.

Your friend claims that the shortest paths of H will be the same as the shortest paths of the original graph, G, but you are skeptical.[1] Prove your friend wrong with a counterexample. That is, find a weighted, directed graph G and a constant c such that the shortest paths of G are different than the shortest paths of the graph obtained by adding the constant c to each edge of G. Your example G should not have a negative cycle.

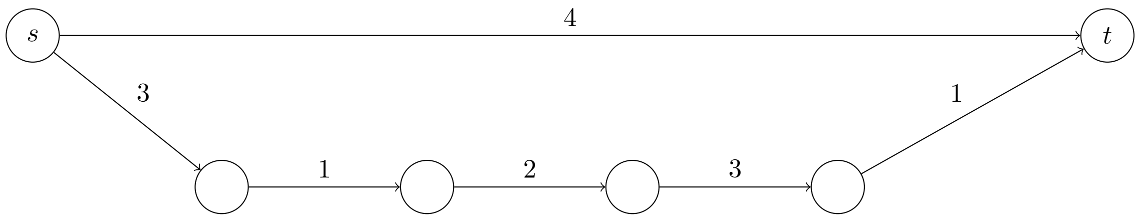
Solution:

Consider the graph below. The shortest path in this graph from s to t takes the bottom route with total weight $1 - 1 + 0 + 1 - 1 = 0$.

[1]Note that your friend agrees that their algorithm will return the wrong shortest path *distances*; that is, `est` will be wrong. However, they argue that the dictionary of predecessors will be right.



Now suppose we add a sufficiently-large constant $c = 2$ to each edge weight so that all edge weights are positive:



The shortest path from s to t in this new graph takes the upper route, and so the shortest paths of the new graph are not the same as the shortest paths of the old graph.

Problem 4.

Recall that the Bellman-Ford algorithm (with early stopping) will terminate early if, after updating every edge, no predecessors have changed. Suppose it is known that the greatest shortest path distance in a graph $G = (V, E)$ has $\Theta(\sqrt{V})$ edges. What is the worst case time complexity of Bellman-Ford when run on this graph? State your answer using Θ notation.

Solution: The loop invariant for Bellman-Ford says that after updating all edges α times, all nodes whose shortest path distance is $\leq \alpha$ has the correct shortest path distance. Since the longest shortest path is $\Theta(\sqrt{V})$, we'll find all shortest paths after $\Theta(\sqrt{V})$ (outer) iterations. Every iteration involves $\Theta(E)$ work because it updates every edge, so the total work of the loop is $\Theta(E\sqrt{V})$.

We also spend $\Theta(V)$ time in the set up to initialize the estimated distance for each node in the graph. Therefore the total time is $\Theta(V + \sqrt{V}E)$.