CS 6363: Homework 4, Part 1

Version 1.0

Instructor: Benjamin Raichel

Due Thursday November 29, at 1:00 pm

Homework is to be submitted online through eLearning by the above deadline. Typeset solutions (in LaTeX) are strongly preferred, but not required. If you chose to hand wright and scan your solutions, if what is written is illegible you will be marked down accordingly. Please make sure your name is on each page of the submission. You do not need to restate the problem, just the problem number.

Explanations are to be given for each problem, unless the problem specifically states it is not necessary. Explanations should be clear and concise. Rambling and/or imprecise explanations will be marked down accordingly.

Problem 1. Running the algorithms (12 points)

In this problem you will simulate Dijkstra and Bellman-Ford on a graph.
For clarity, these algorithms are as follows:

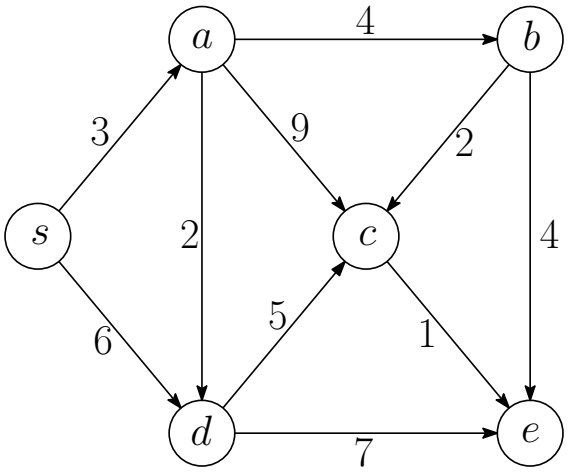
```
1: procedure RELAX(u → v)
2:     dist(v) = dist(u) + w(u → v)
3:     pred(v) = u
```

```
1: procedure INITSSSP(s)
2:     dist(s) = 0
3:     pred(s) = NULL
4:     for all vertices v ≠ s do
5:         dist(v) = ∞
6:         pred(v) = NULL
```

```
1: procedure DIJKSTRA(s)
2:     InitSSSP(s)
3:     Init min dist() priority queue, Q
4:     Put s in Q
5:     while Q not empty do
6:         u = Q.Pop()
7:         for all edges u → v do
8:             if u → v tense then
9:                 Relax(u → v)
10:                Put v in Q
```

```
1: procedure BELLMAN-FORD(s)
2:     InitSSSP(s)
3:     loop |V| − 1 times
4:         for every edge u → v do
5:             if u → v tense then
6:                 Relax(u → v)
7:     for every edge u → v do
8:         if u → v tense then
9:             return "Negative cycle!"
```

The input graph is the following:



(a) (6 points) First simulate Dijkstra's algorithm by filling out the following table. Each column represents a single round of Dijkstra's algorithm. In the top row write the "active" vertex of that round, i.e. the one popped from the queue which is the one being set as the predecessor. Then for the other column entries, write the current distance values, where you only need to fill in the entry if it changed since the previous round. The first round, where s is popped off the queue, has already been filled in. No explanation required.

Active	Null	s					
s	0						
a	∞	3					
b	∞						
c	∞						
d	∞	6					
e	∞						

(b) (6 points) Bellman-Ford does not specify which order it loops over the edges, but if the right order is chosen Bellman-Ford only does a single loop. Give an ordering of the edges in G such that after a single round of the Bellman-Ford algorithm the $pred(v)$ and $dist(v)$ values correctly describe a shortest path tree. No explanation required.

Problem 2. SSSP (12 points)

Let $G = (V, E)$ be a directed graph with positive edge weights, and let s be an arbitrary vertex of G. Your roommate claims to have written a program which computes the single source shortest path tree from s. Specifically, for every vertex $v \in V$, your roommate tells you the $pred(v)$ and $dist(v)$ values output by the program. Unfortunately, your roommate is not that reliable and so you want to check their solution. Give an $O(|V| + |E|)$ time algorithm to check whether the given values correctly describe some shortest path tree of G.

For simplicity you can assume that all vertices in the graph are reachable from s, and hence for every vertex in your roommates solution $dist(v) \neq \infty$, and moreover s is the only vertex such that $pred(s) = NULL$.

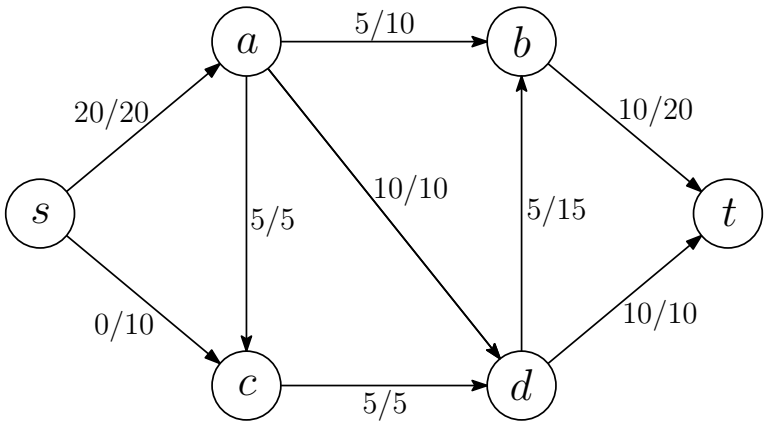
Problem 3. Best Edge (12 points)

Let $G = (V, E)$ be a directed graph with non-negative edge weights, let s and t be vertices of G, and let $H = (V, F)$ be a subgraph of G obtained by deleting some edges, that is $F \subset E$. Suppose we want to reinsert exactly one edge from $E \setminus F$ back into H, so that the shortest path from s to t in the resulting graph is as short as possible. Describe and analyze an algorithm that chooses the best edge to reinsert, in $O(|E| \log |V|)$ time.

Problem 4. Computing Flows and Cuts (9 points)

No explanation required for any part of this problem.

(a) Draw the residual graph of the following flow network.



(b) Give the ordered list of vertices of an augmenting path in your residual graph.

(c) Give a minimum s-t cut (i.e. give the vertex bi-partition).