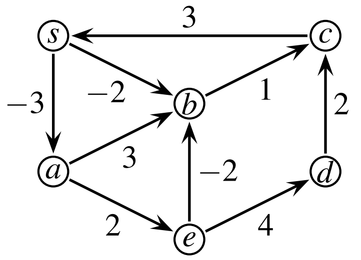
Lecturer: Thomas Rothvoss Due date: Friday, April 21, 2017, in class

Problem Set 3

409 - Discrete Optimization

Spring 2017

Exercise 1
Consider the following directed graph $G = (V,E)$ (edges are labelled with edge cost $c(e)$).



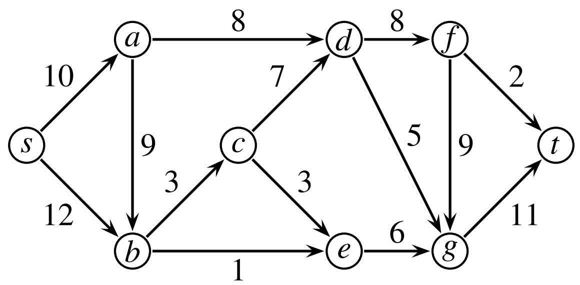
a) Use the Moore-Bellman-Ford algorithm to compute the distances from s to each other node v (call those values $\ell(v)$; you can use your favourite ordering for the edges).

b) Use the labels $\ell(v)$ from $a)$ as **potentials** $\pi(v)$. In the above graph, label the nodes with their potential and label the edges with their **reduced costs**. Are the potentials feasible? Is c conservative?

c) Let $c_\pi(e)$ be the reduced cost of edge e that you computed in $b)$. Now run Dijkstra's algorithm with cost function c_π and source node a. Use the symbol $\ell'(v)$ to denote the computed a-v distances.

d) How do you translate the values $\ell'(v)$ from $c)$ into the actual a-v distances w.r.t. to the original cost function c?

Exercise 2
Consider the following network (G,u,s,t) (edges e are labelled with capacities $u(e)$):



a) Run the Ford-Fulkerson algorithm to compute a maximum s-t flow. After each iteration draw the current flow f and the corresponding residual graph G_f. What is the optimum flow value?

b) For the optimum flow f that you computed, define $S := \{v \in V \mid v \text{ is reachable from } s \text{ in } G_f\}$. Which are the nodes in S and what is the value $u(\delta^+(S))$ of the cut?

Exercise 3

In this problem, we consider the single-source shortest paths problem on an important class of directed graphs. Throughout this problem, $G = (V,E)$ will denote a weighted directed graph containing **no directed cycles**. We say that an ordering $v_1, \ldots, v_n$ of the vertex set V is a *topological sorted order* for G if x precedes y in the order whenever (x,y) is a directed edge of G.

a) Give an $O(|V|+|E|)$-time algorithm for computing a topological sorted order of G. Prove that your algorithm is correct and runs in $O(|V|+|E|)$ time. In order to attain the desired running time, you may assume that, for a node $u \in V$ of out-degree d, you can access *all* the outgoing edges and corresponding edge weights in total time $O(d)$. (For example, this assumption holds when the input graph G is represented by adjacency lists.)

b) Fix a source vertex s. Give an $O(|V|+|E|)$-time algorithm that computes $\ell(v)$ for all vertices $v \in V$, where $\ell(v)$ is the minimum possible total weight along a directed path from s to v. Again, prove that your algorithm is correct and has the desired running time. In order to attain the desired running time, you may assume that, for a node $u \in V$ of out-degree d, you can access *all* the outgoing edges and corresponding edge weights in total time $O(d)$. **Hint:** Your algorithm should make "updates" similar to the updates in Moore-Bellman-Ford and Dijkstra. In order to prove that $\ell(v) = d(s,v)$, you might consider proving the two inequalities $\ell(v) \geq d(s,v)$ and $\ell(v) \leq d(s,v)$ separately, as we did in the proof of the Moore-Bellman-Ford algorithm.