COMP 6651: Assignment 7

Fall 2020

Submission through Moodle is due by November 8th at 23:55

1. You are given a DAG $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$ and a dedicated source vertex s. Your goal is to compute for each vertex $v \in V$ the weight of the *longest weighted path* from s to v. Use dynamic programming.

   (a) State the semantic array.

   (b) State the computational array. What is (are) base case(s)? In what order should the entries of the computational array be filled in?

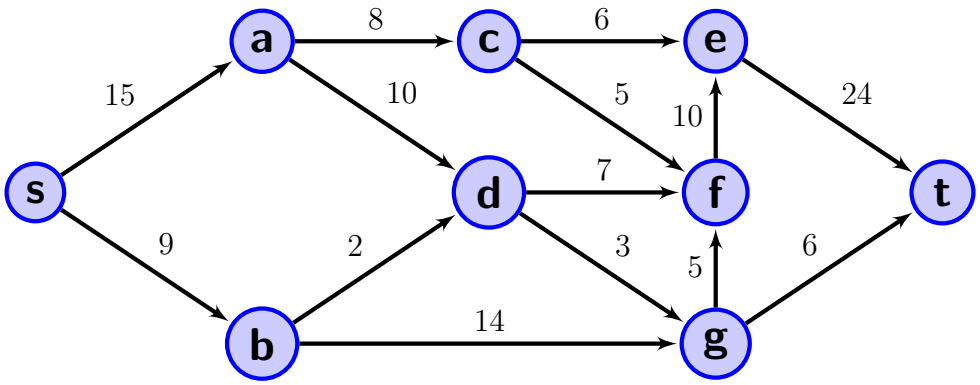
   (c) Justify why the two arrays are equivalent.

   (d) State the running time of the algorithm.

   (e) Does your algorithm work with negative edge weights? Why or why not?

   (f) Does your algorithm work if G is not a DAG? Why or why not?

2. Consider the following network graph.



   (a) Compute maximum flow f and minimum cut (S, T).

   (b) Draw the residual graph G_f - don't forget to state the capacities. Indicate the minimum cut (S, T) in the residual graph by circling S and T.

   (c) An edge is called *constricting* if increasing its capacity leads to an increase in the value of maximum flow. List all constricting edges in the above network.

   (d) Find a small (at most 4 nodes) example of a network graph that has no constricting edges.

   (e) Describe in plain English an efficient algorithm to find all constricting edges. Argue correctness by using results from lectures/textbook. State the running time of your algorithm.

3. Consider a flow network $G = (V, E), s, t, c$ with integral capacities together with an additional edge-price function $p : E \to \mathbb{N}$. The value $p(e)$ denotes the price to increase the capacity of edge e by one unit. Suppose we have already computed maximum flow f in the network. Now, we would like to increase this maximum flow f by one unit by increasing capacities of some edges. The goal is to do this with the least possible cost. Design an efficient algorithm to compute which edge capacities to increase.

   (a) Briefly describe your algorithm in plain English.

   (b) Describe your algorithm in pseudocode.

   (c) Provide a concise argument of correctness of your algorithm. You may use results proven in class/textbook, but make sure to cite them accurately.

   (d) State and justify the runtime of your algorithm.