COMP 3711 – Design and Analysis of Algorithms
2019 Spring Semester – Written Assignment # 4
Distributed: April 26, 2019 – Updated: May 4, 2019
Due: May 10, 2019

Your solutions should contain (i) your name, (ii) your student ID #, and (iii) your email address
Notes:

- Follow the guidelines on doing your own work and avoiding plagiarism given on the class home page.
  In particular ***don't forget to acknowledge individuals who assisted you, or sources where you found solutions.*** Failure to do so will be considered plagiarism.

- Write clearly and follow the submission guidelines on the class web page. Use white, unwatermarked paper, e.g., no student society stationary.

  - If handwritten, solutions should be single sided, start a new page for every problem, be single column and leave space between consecutive lines and more space between paragraphs.
  - If typed, try to use an equation editor, e.g., latex , the equation editor in MS-WORD or whatever the equivalent is in whatever typesetting system you are using

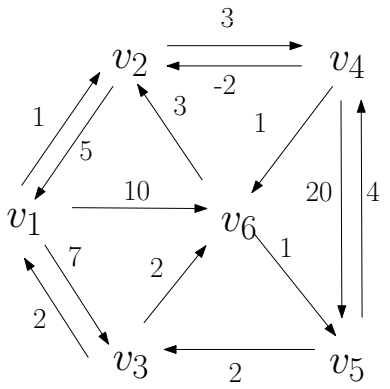
- This assignment is due by 23:59 on May 10, 2019 in BOTH hard AND soft copy formats. A hard copy should be deposited in one of the two COMP3711 assignment collection boxes outside of room 4210. A soft copy for our records in PDF format should also be submitted via the online CASS system. See the Assignment 1 page in Canvas for information on how to submit online.

- The default base for logarithms will be 2, i.e., $\log n$ will mean $\log_2 n$. If another base is intended, it will be explicitly stated, e.g., $\log_3 n$.

- Correction. May 4, 2019. In the original distribution, the example on page 7 stated that $\{g, f\}$ is a *unique* smallest separating set. This was false. $\{g, a\}$ is also a smallest separating set. That error has now been fixed.

Problem 1: All-Pairs Shortest Path [25 pts]

Let $G = (V, E)$ be the directed weighted graph shown below with its associated weighted adjacency matrix.



$$W = \left(\begin{array}{c|c|c|c|c|c} 0 & 1 & 7 & \infty & \infty & 10 \\ \hline 5 & 0 & \infty & 3 & \infty & \infty \\ \hline 2 & \infty & 0 & \infty & \infty & 2 \\ \hline \infty & -2 & \infty & 0 & 20 & 1 \\ \hline \infty & \infty & 2 & 4 & 0 & \infty \\ \hline \infty & 3 & \infty & \infty & 1 & 0 \end{array}\right)$$

This problem asks you to solve the all-pairs shortest-path problem on this graph in two different ways.

1. Run the 2nd dynamic-programming solution from the slides on this graph. Recall that in the 2nd dynamic-programming solution

$$d_{ij}^{(2s)} = \min_{1 \le k \le n} \left\{ d_{ik}^{(s)} + d_{kj}^{(s)} \right\}$$

where $d_{ij}^{(1)} = w_{ij}$ so $D^{(1)} = W$ and, in general, $D^{(m)}$ is the matrix $\left[d_{ij}^{(m)} \right]$. For this problem you need to fill in the matrices $D^{(2)}$, $D^{(4)}$ and the final solution $D^{(8)} (= D^{(5)})$.

$$D^{(2)} = \left(\begin{array}{|c|c|c|c|c|c|} \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline \end{array}\right) \quad D^{(4)} = \left(\begin{array}{|c|c|c|c|c|c|} \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline \end{array}\right) \quad D^{(8)} = \left(\begin{array}{|c|c|c|c|c|c|} \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline \end{array}\right)$$

2. Now run the Floyd-Warshall algorithm on the graph. Recall that in the Floyd-Warshall algorithm

$$d_{ij}^{(k)} = \min\left\{d_{ij}^{(k-1)},\, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}$$

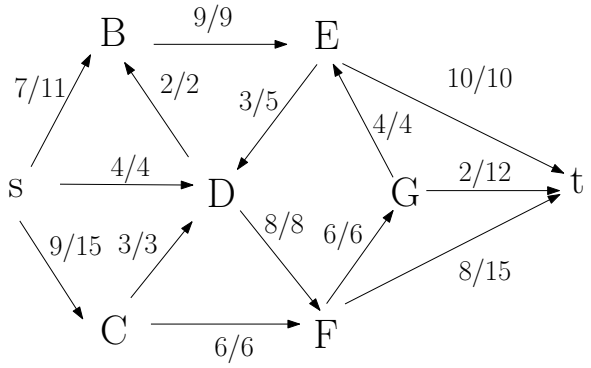
where $d_{ij}^{0} = w_{ij}$ so $D^{(0)} = W$, and $D^{(m)}$ is the matrix $\left[d_{ij}^{(m)}\right]$. For this problem you need to fill in the matrices $D^{(i)}$, where $i = 1, 2, 3, 4, 5, 6$ and $D^{(6)}$ is the final solution.

$D^{(1)} =$ $D^{(2)} =$ $D^{(3)} =$

$D^{(4)} =$ $D^{(5)} =$ $D^{(6)} =$

Problem 2: The Ford-Fulkerson Algorithm [25 pts]

Consider the given flow in directed graph below. The labels on the edges are in the form f/c where c is the capacity of the edge and f is the flow value across the edge:



(a) Draw the residual network of the given graph and flow.

(b) Find an augmenting path p in the residual network and draw it. Explictly state the maximum flow $c_f(p)$ that can be pushed through p.

(c) Draw the new flow that results by adding the flow pushed through the augmenting path to the old flow. To answer this question you must redraw the graph. Relabel each edge in the format f'/c where f' is the new flow value.

(d) Is this new flow a maximum flow? PROVE your answer. (A proof that it is maximum would be a cut with capacity equal to the flow value. A proof that it isn't maximum would be a larger flow).

Problem 3: MST Redux [25 pts]

Let $G = (V, E)$ be a weighted undirected graph inputted as an adjacency list. Assume that all edges have distinct weights so the *Minimum Spanning Tree* T is unique and that you have already run a MST algorithm that has found and stored T.

Now arbitrarily *increase* the weight of any one edge $e = (u, v)$ in E. You may assume that, even after the increase, all edges still have distinct weights.

Let T' be the MST of the graph with the new weights. i.e, after replacing $w(u, v)$ with its new weight.

(a) Prove that after increasing one weight either $T' = T$ or that T and T' differ by at most one edge.

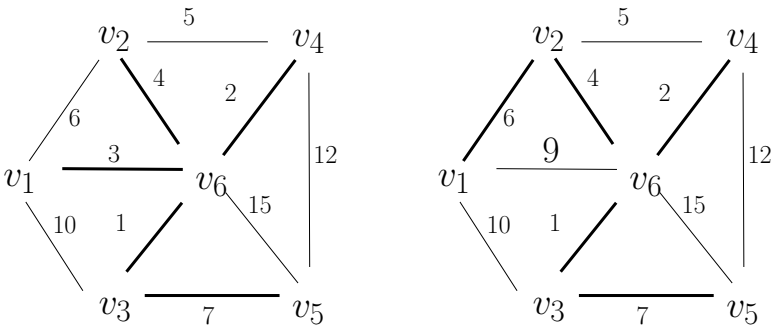
The only fact that you may use from class is the *Cut Lemma*. Any other fact must be proven from scratch.

(b) Give an $O(|E|)$ time algorithm for finding T'. It is not necessary to give it in Pseudocode.

You may use any algorithm we taught in class or the tutorials as a subroutine. If you do use such an algorithm from the class/tutorials as a subroutine, you must explicitly state the name of that algorithm, its run time, why the input you are feeding into is of the correct type and the output of that algorithm.

It suffices to briefly describe what the algorithm does. Your description must explicitly justify why your algorithm is correct and why it is $O(|E|)$.

Example: below, in the graph on the left, the MST is the set of 5 heavy edges. If the weight of (v_1, v_6) *was increased from* 3 *to* 5.5*, the MST would stay the same. But, if the weight of* (v_1, v_6) *was increased from* 3 *to* 9 *the new MST would be* T'*, the 5 heavy edges on the right, i.e.,* T' *is* T *with the edge* (v_1, v_2) *being swapped for* (v_1, v_6)*.*



Hint: Consider the "cut lemma" and how it was used to prove the uniqueness of T *by showing that every edge in* T *must be in ALL MSTs. How can you modify*

that proof to show (a) above? For (b) consider WHY the cut lemma says that e is in T. If another edge needs to replace e, what property does that other edge need to satisfy?

Problem 4: Max-Flow and Graph Reliability [25 pts]

Let $G = (V, E)$ be a directed graph with two specified vertices $s, t \in V$. Assume that G contains at least one s-t path.

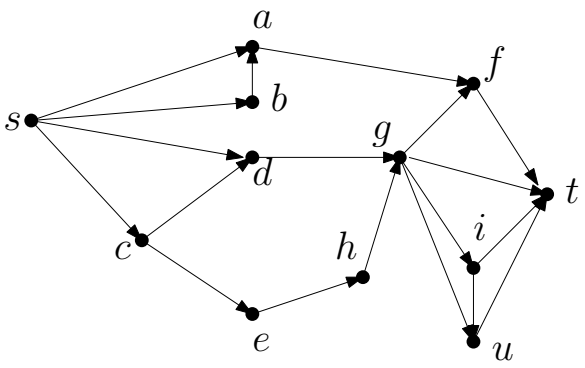
A subset of edges $E' \subseteq E$ is a *separating edge set* if, after removing E' from G, no s-t path exists.

A subset $V' \subseteq V - \{s, t\}$ of vertices is a *separating vertex set* if, after removing V' and all edges connected to V' from G, no s-t path exists. If $(s, t) \notin E$, a separating vertex set always exists.

E' is a *smallest separating edge set* if it contains the smallest number of edges among all separating edge sets. V' is a *smallest separating vertex set* if it contains the smallest number of vertices among all separating vertex sets.

Note that a smallest separating edge set E' might not be unique. A smallest separating vertex set V' also might not be unique.

In the graph below $\{(a, f), (d, g), (e, h)\}$ is a smallest separating edge set, but there are others as well. $\{g, f\}$ and $\{g, a\}$ are the only smallest separating vertex sets.



Smallest separating sets are indicators of failure points in a network and are therefore used in studies of network reliability.

In what follows you may use any facts or algorithms taught in class as long as you explicitly reference them (which means including their statement and where in the class or tutorial notes you found them).

(a) Give an $O(|E|^2)$ time algorithm for finding a smallest separating edge set for G.

(b) Assume $(s, t) \notin E$. Give an $O(|V||E|)$ time algorithm for finding a smallest separating vertex set for G.

For both (a) and (b) describe your algorithm clearly (code is not necessary) and prove its correctness and running time. Your proof of correctness must

contain a section that *EXPLICITLY* justifies why your output is a smallest separating edge or vertex set.

Hints: For (a), consider the algorithm for finding the maximum number of edge disjoint s-t paths taught in class. How can you modify that to solve this problem?

For (b) Modify your graph as follows. For every vertex $v \in V$,
(i) remove v *and create two new vertices* v_ℓ, v_r,
(ii) create new edge (v_ℓ, v_r).
(iii) Replace every edge $(u, w) \in E$ *with edge* (u_r, w_ℓ).

Note that if P *is an* $s - t$ *path in* G *that passes through vertex* v*, the corresponding path in the modified graph must pass through edge* (v_ℓ, v_r). *Now appropriately modify the algorithm for part (a).*