COMP 3711 – Design and Analysis of Algorithms
2021 Fall Semester – Written Assignment # 5
Distributed: November 19, 2021
Due: November 30, 2021, 11:59 PM

Your solution should contain
(i) your name, (ii) your student ID #, and (iii) your email address
at the top of its first page.

Some Notes:

- Please write clearly and briefly. Your solutions should follow the guidelines given at
  https://canvas.ust.hk/courses/38226/pages/assignment-submission-guidelines
  In particular, your solutions should be written or printed on *clean* white paper with no watermarks, i.e., student society paper is not allowed.

- Please also follow the guidelines on doing your own work and avoiding plagiarism as described on the class home page.
  You must acknowledge individuals who assisted you, or sources where you found solutions. Failure to do so will be considered plagiarism.

- The term *Documented Pseudocode* means that your pseudocode must contain documentation, i.e., comments, inside the pseudocode, briefly explaining what each part does.

- Many questions ask you to explain things, e.g., what an algorithm is doing, why it is correct, etc. To receive full points, the explanation must also be *understandable* as well as correct.

- Please make a *copy* of your assignment before submitting it. If we can't find your submission, we will ask you to resubmit the copy.

- Submit a SOFTCOPY of your assignment to CASS (not Canvas) by the deadline. The softcopy should be one PDF file (no word or jpegs permitted, nor multiple files).
  If your submission is a scan of a handwritten solution, make sure that it is of high enough resolution to be easily read. At least 300dpi and possibly denser.

Q1 Converse of the Cut Lemma [10 pts]

Assume that the edges in a connected graph $G = (V, E)$ have distinct weights. Let $T = (V, E')$ denote the unique MST of G.

Mathematically prove the following Lemma.

Lemma: If $e \in E'$, i.e., e is in the MST T, then there exists a subset of vertices $S_e \subset V$ such that e is the minimum cost edge among all edges in E with exactly one endpoint in S_e.

Requirements and Comments:

- Your proof must be "from scratch". That is, you may not assume the correctness of any fact taught in class.
- Your proof must be mathematically formal. Split it into small paragraphs with space between each paragraph. Make it very clear what each paragraph is assuming and proving.
- The lemma statement above uses the terminology of the Cut Lemma taught in class. In symbols, the statement of the lemma above is that

  "If $e \in E'$ then there exists some subset of vertices $S_e \subset V$ such that $w(e) = \min\{w(e') : e' = (u', v') \in E,\, u' \in S_e,\, v' \notin S_e\}$".

  This is the *Converse* of the statement of the Cut Lemma.

Q2 Generalization of Bipartite Matching [25 pts]

This problem is taken, slightly modified, from the book Algorithms by Jeff Erikson (http://algorithms.wtf)

The Department of Commuter Silence at Sham-Poobanana University has a flexible curriculum with a complex set of graduation requirements.

The department offers n different *courses*, and there are m different graduation *requirements*.

Each requirement specifies a subset of the n courses and the minimum number of courses that must be taken from that subset.

The subsets for different requirements may overlap, **but each course can be used to satisfy at most one requirement** (In HKUST terms, this means that a course can not be double-counted to satisfy two requirements).

As an example, suppose there are $n = 5$ courses $\{C_1, C_2, C_3, C_4, C_5\}$ and $m = 2$ graduation requirements:

- You must take at least 2 courses from the subset $\{C_1, C_2, C_3\}$.
- You must take at least 2 courses from the subset $\{C_3, C_4, C_5\}$.

Then a student who has only taken courses C_2, C_3, C_4 cannot graduate, but a student who has taken either C_1, C_2, C_3, C_4 or C_2, C_3, C_4, C_5 can graduate.

Describe and analyze an $O(n^2 m)$ time algorithm to determine whether a given student can or cannot graduate.

The input to your algorithm is
the list of m requirements given by (i) $m \times n$ matrix $X_{i,j}$ and (ii) size m array N_i and
(iii) the set of courses the student has taken, given by size n array L_j.
In the three definitions below, $1 \le i \le m$ and $1 \le j \le n$.

$$X_{i,j} = \begin{cases} 1 & \text{If course } C_j \text{ is in requirement list } i, \\ 0 & \text{Otherwise} \end{cases}$$

$$N_i = \text{Number of courses that must be taken to satisfy requirement } i.$$

$$L_j = \begin{cases} 1 & \text{If the student took course } j. \\ 0 & \text{If the student did not take course } j. \end{cases}$$

You may assume that you have also been given a procedure that runs the Ford-Fulkerson Max-Flow algorithm and can use that as a subroutine.

(A) Clearly describe your algorithm for solving the problem.

(B) Explain why your algorithm runs in $O(n^2 m)$ time.

(C) Explain why your algorithm is correct.

Q3 Max-Flow and Graph Reliability [35 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.

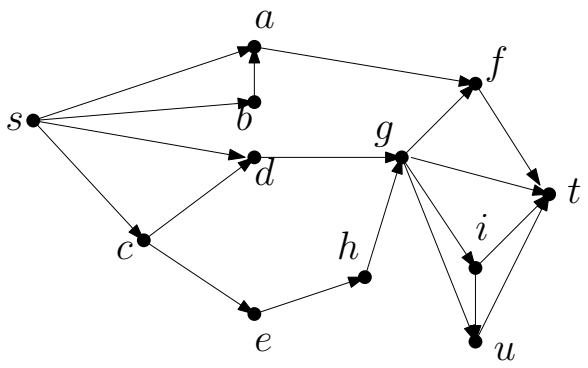
A subset of edges $E' \subseteq E$ is a *separating edge set* if, after removing E' from G, no s-t path exists.

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. Note that 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.

Both (a) and (b) should contain 3 parts. Part (i) should describe your algorithm clearly (code is not necessary). Part (ii) should prove its correctness.

Part (iii) should derive its 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, Comments and Recommendations:

- 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?

  Hints. Let S, T be ANY s-t cut. Then the set of edges crossing between S, T is a separating edge set. (Prove this)

  Consider the Min-Cut you get when solving the edge disjoint s-t path problem. The statement above implies that the max number of edge disjoint paths is an upper bound on the size of the smallest separating edge set (why?). Can you show that these two values must be equal?

- For (b), modify G to create a new graph $G' = (V', E')$.

  (i) For every vertex $v \in V$, create two vertices v_ℓ, v_r in V'.

  (ii) For every vertex $v \in V - \{s, t\}$, Create edge (v_ℓ, v_r) in E'.

  (iii) For every edge $(u, w) \in E$, create edge (u_r, w_ℓ) in E'.

  (iv) Remove vertices s_ℓ and t_r from V'.

  Note that if P is an $s - t$ path in G that passes through vertex v, the corresponding path P' in G' is an $s_r - t_\ell$ path that passes through edge (v_ℓ, v_r).

- The modified graph G' contains two very different type of edges:

$$\begin{aligned} E'_1 &= \{(v_\ell, v_r) : v \in V - \{s, t\}\}, \\ E'_2 &= \{(u_r, v_\ell) : (u, v) \in E\}. \end{aligned}$$

  A separating vertex set in G corresponds to a separating edge set in G' in which all edges are in E'_1 and vice-versa.

  So solving (b) corresponds to finding a smallest separating edge set in which all edges are in E'_1.

  The idea is to appropriately modify the algorithm for part (a) to find such a set.

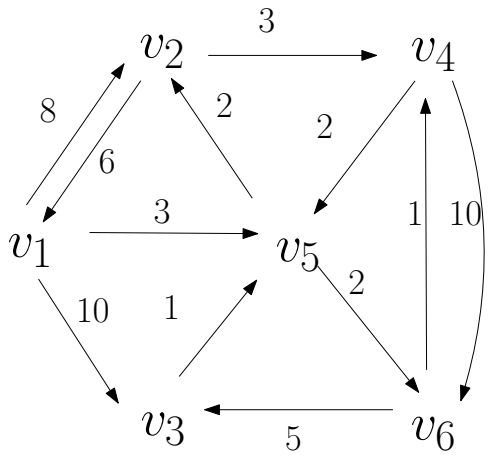
- Suppose you assign capacity 1 to all edges in E'_1 and capacity 2 to all edges in E'_2. Prove that the edges crossing a Min-Cut must all be of type E'_1.

  (Proving this requires understanding the structure of the min-cut in the proof of the correctness of the Ford-Fulkerson algorithm.)

- Use the observations above and (a modified version of) the result of part (a) to solve (b).

Q4 All-Pairs Shortest-Paths [10 pts]

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



$$W = \left(\begin{array}{c|c|c|c|c|c} 0 & 8 & 10 & \infty & 3 & \infty \\ \hline 6 & 0 & \infty & 3 & \infty & \infty \\ \hline \infty & \infty & 0 & \infty & 1 & \infty \\ \hline \infty & \infty & \infty & 0 & 2 & 10 \\ \hline \infty & 2 & \infty & \infty & 0 & 2 \\ \hline \infty & \infty & 5 & 1 & \infty & 0 \end{array}\right)$$

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

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

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

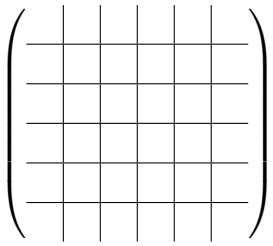
where $d_{ij}^{(1)} = w_{ij}$ so $D^{(1)} = W$ and, in general, $D^{(m)}$ is the matrix $\left[d_{ij}^{(m)} \right]$. For this problem you need to fill in the matrices $D^{(2)}$, $D^{(4)}$ and the final solution $D^{(8)} (= D^{(5)})$.

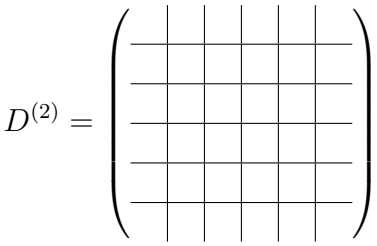
$$D^{(2)} = \left(\begin{array}{c|c|c|c|c|c} & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \end{array}\right) \quad D^{(4)} = \left(\begin{array}{c|c|c|c|c|c} & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline & & & & & \\ \hline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

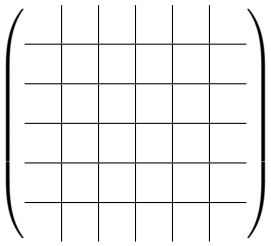
(b) Now run the Floyd-Warshall algorithm on the graph. Recall that in the Floyd-Warshall algorithm

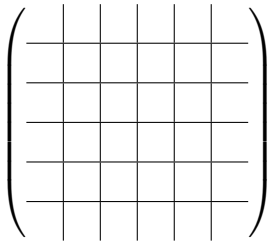
$$d_{ij}^{(k)} = \min\left\{d_{ij}^{(k-1)},\, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}$$

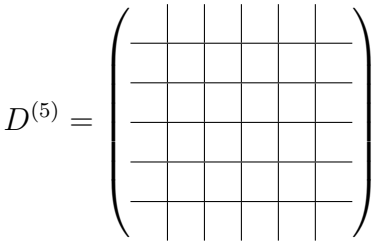
where $d_{ij}^{0} = w_{ij}$ so $D^{(0)} = W$, and $D^{(m)}$ is the matrix $\left[d_{ij}^{(m)}\right]$. For this problem you need to fill in the matrices $D^{(i)}$, where $i = 1, 2, 3, 4, 5, 6$ and $D^{(6)}$ is the final solution.

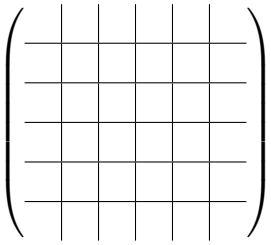
$D^{(1)} =$ 

$D^{(2)} =$ 

$D^{(3)} =$ 

$D^{(4)} =$ 

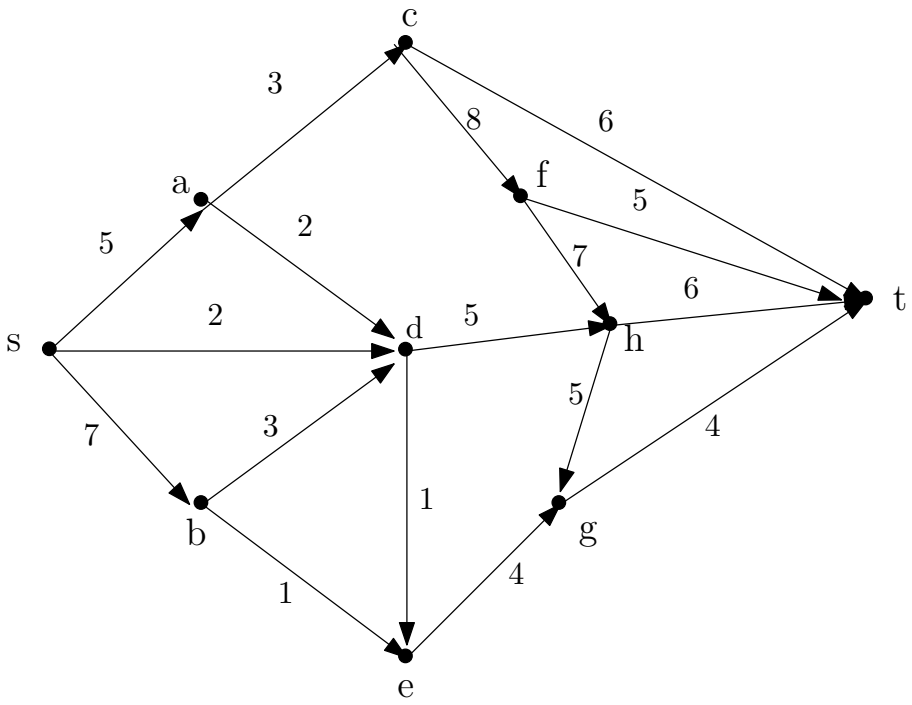
$D^{(5)} =$ 

$D^{(6)} =$ 

Q5 The Ford-Fulkerson Algorithm [20 pts]

Consider the graph below with the given capacities.

Start with a flow that has $f(e) = 0$ for every edge and run the Ford-Fulkerson Max Flow Algorithm on the graph below to find a max-flow from s to t



(a) Show every step of the algorithm. That is, for every step show the current flow f, its associated residual graph G_f and the augmenting path P you choose, along with the value $c_f(P)$.

All of the information for each step should be on the same page. (If it's not too crowded, try putting all of the information for two steps on each page.)

(b) Show your final flow and provide a cut with capacity equal to that of the flow.

Note: If you don't have access to another drawing tool you may use the PPT slide we provide as the basis for your drawing, editing/copying it to show all of the intermediate residual and flow graphs.