Computer Science CSC373H
St. George Campus

February 24, 2021
University of Toronto

Homework Assignment #4

Due: March 10, 2021, by 11:00 am

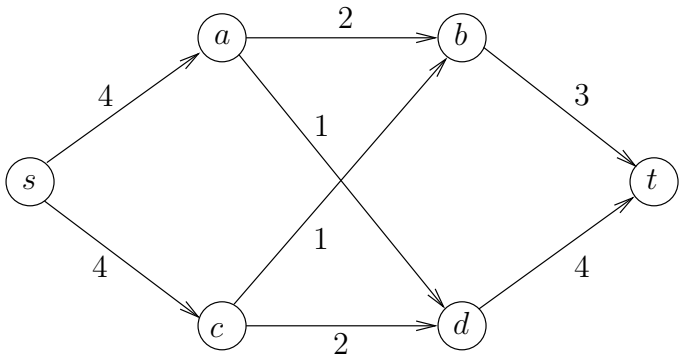
- **You must submit your assignment through the Crowdmark system.** You will receive by email an invitation through which you can submit your work. If you haven't used Crowdmark before, give yourself plenty of time to figure it out!
- You must submit a **separate** PDF document with for **each** question of the assignment.
- To work with one or two partners, you and your partner(s) must form a **group** on Crowdmark (one submission only per group). We allow groups of up to three students, submissions by groups of more than three students will not be graded.
- The PDF file that you submit for each question must be typeset (**not** handwritten) and clearly legible. To this end, we encourage you to learn and use the LaTeX typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You can use other typesetting systems if you prefer, but handwritten documents are not accepted.
- If this assignment is submitted by a group of two or three students, for each assignment question the PDF file that you submit should contain:
  1. The name(s) of the student(s) who *wrote* the solution to this question, and
  2. The name(s) of the student(s) who *read* this solution to verify its clarity and correctness.
- By virtue of submitting this assignment you (and your partners, if you have any) acknowledge that you are aware of the homework collaboration policy for this course, as stated in: `http://www.cs.toronto.edu/~sam/teaching/373/#HomeworkCollaboration`.
- For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class (in lectures or tutorials) by referring to it.
- Unless we explicitly state otherwise, you should justify your answers. Your paper will be graded based on the correctness and efficiency of your answers, and the clarity, precision, and conciseness of your presentation.
- The total length of your pdf submission should be no more than 4.5 pages long in a 11pt font.

Question 1. (10 marks) Let $\mathcal{F} = (G, s, t, c)$ be a flow network with integral capacities. In this question, by "cut" we mean "s-t cut". For any cut (S, T) of $\mathcal{F}$, a *crossing* from S to T is an edge (u, v) with $u \in S$ and $v \in T$. Explain how to find (in polynomial-time) a minimum cut of $\mathcal{F}$ that has the *smallest number of crossings* among all minimum cuts of $\mathcal{F}$.

HINT: Modify the capacities of $\mathcal{F}$ to create a new flow network $\mathcal{F}'$ in which *any* minimum cut in $\mathcal{F}'$ is a minimum cut *with the smallest number of crossings* in $\mathcal{F}$. *Prove that your modification works.*

Question 2. (20 marks) Let (G, s, t, c) be a flow network. We say that an edge e of G is *a bottleneck edge* if by increasing the capacity of e and only of e, we increase the maximum flow of the flow network. More precisely, e is a bottleneck edge of (G, s, t, c), if there is a capacity function c' for the edges of G such that $c'(e) > c(e)$ and $c'(e') = c(e')$ for every edge e' of G other than e, and the value of a maximum flow of (G, s, t, c') is (strictly) greater than the value of a maximum flow of (G, s, t, c).

Consider the following flow network.



a. Show a maximum flow and a minimum cut of this flow network. Do not show how you obtained your answer, but explain why your answer must be correct using a theorem we proved in class.

b. Show the residual graph of this flow network with respect to the maximum flow you found in part (a).

c. Give the set of bottleneck edges of this flow network.

d. Show the simplest flow network you can think of that has no bottleneck edges.

e. Give an efficient algorithm that takes as input a flow network (G, s, t, c), where all the edge capacities are integers, and produces as output the set of bottleneck edges of (G, s, t, c). Explain why your algorithm is correct, and analyse its running time. (Hint: Find a maximum flow f of (G, s, t, c), and use the corresponding residual graph G_f.)

Question 3. (10 marks) We are given a directed graph $G = (V, E)$ and two distinguished nodes $s, t \in V$. A set of s-to-t paths in G are *node-disjoint* if no two paths in the set share a node other than s and t. We wish to find a maximum-size set of node-disjoint s-to-t paths in G. Give an efficient algorithm that does this. Explain why your algorithm is correct, and analyse its time complexity in terms of the number of nodes n and the number of edges m in G. For the purposes of this problem, you can assume (without loss of generality) that there are no edges into s or out of t in the graph G.

HINT: Reduce the problem of finding *node*-disjoint paths in G, to the problem of finding *edge*-disjoint paths (no two paths share an edge) in a related graph.