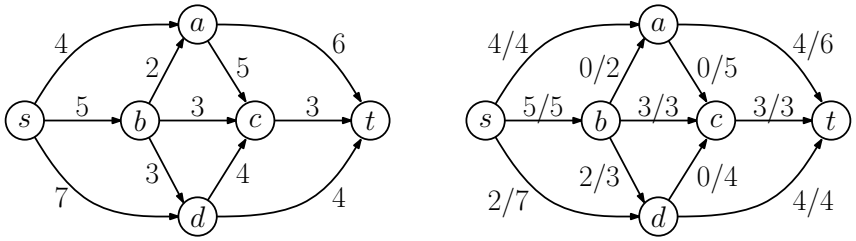
Problem Set 8: Network Flows

Handed out Monday, October 21. Due at the start of class Monday, October 28.

Note: When asked to present an "efficient algorithm," it suffices to provide a reduction to network flow. (Unless the problem asks specifically for this information, you do not need to explain which network-flow algorithm will be used to solve the problem.)

Problem 1. (20 points) In this step, we will trace the partial execution of the Ford-Fulkerson algorithm on a sample network.

(a) Consider the s-t network G shown in Fig. 1(a), and consider the initial flow f in Fig. 1(b). Show the residual network G_f for this flow.



(a) Initial network G (b): Initial flow f

Figure 1: Problem 1: Ford-Fulkerson.

(b) Find any s-t path in G_f. How much flow can you push along this path? Show the updated flow (in the same manner as Fig. 1(b)).

(c) Show the residual network that results for your flow from (b).

(d) Is this the final maximum flow in this network? (If not, keep running Ford-Fulkerson until you get the maximum flow, and show the final flow.) What is the value of the maximum flow?

(e) Show the residual network for your maximum flow from (d). (If the flow from (c) was already maximum, then state this.)

(f) Show the cut that results by partitioning the network into two subsets of vertices, the vertices X that are reachable from s and the remaining vertices $Y = V \setminus X$. What is the capacity of this cut? (It should match your flow value, if you did everything correctly.)

Problem 2. (10 points) An edge of a flow network is called *critical* if decreasing the capacity of this edge results in a decrease in the maximum flow value. Present an efficient algorithm that, given an s-t network G finds any critical edge in a network (assuming one exists).

Problem 3.(15 points) The computer science department at a major university has a tutoring program. There are m tutors, $\{t_1, \ldots, t_m\}$ and n students who have requested the tutoring service $\{s_1, \ldots, s_n\}$. Each tutor t_i has a set T_i of topics that he/she/they knows, and each student s_j has a set of topics S_j that he/she/they wants help with. We say that tutor t_i is *suitable* to work with student s_j if $S_j \subseteq T_i$. (That is, the tutor t_i knows all the topics of interest to student s_j.) Finally, each tutor t_i has a range $[a_i, b_i]$, indicating that this tutor would like to work with at least a_i students and at most b_i students.

Given a list of students, a list of tutors, the ranges $[a_i, b_i]$ for the tutors, and a list of suitable tutors for each student, present an efficient algorithm that determines whether it is possible to generate a pairing of tutors to students such that:

- Each student is paired with *exactly* one tutor.
- Each tutor t_i is paired with *at least* a_i and *at most* b_i students.
- Each student is paired only with a *suitable* tutor.