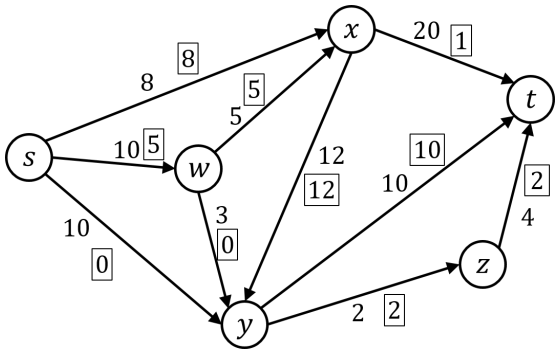
COSC 311: ALGORITHMS
HW4: NETWORK FLOW

Due Friday, November 17, 12pm

Please type up your solutions (you may hand-draw the graphs). If you discuss any of the problems with your classmates, please note at the top of your submission with whom you consulted.

1 Warmup

1) Finding max flows and min cuts.
Here is a graph (the numbers in boxes represent the amount of flow along an edge, and the "unadorned" numbers represent the edge capacities):



(a) What is the value of the current flow in the graph?

(b) Draw the residual graph for this flow. Use the residual graph to explain why the current flow is or is not a max flow.

(c) What is the capacity of the min cut in this graph? Find a min cut with this capacity.

2 Network Flow Theory

2) Rational edge capacities.
So far we have assumed that all of the edge capacities in our graph are positive integers. In this problem we will investigate ways in which we can relax this constraint.

(a) In class, we proved that the Ford-Fulkerson algorithm does in fact terminate. Where in that argument did we use the fact that the edge capacities are integers? Why does the argument break if our edge capacities are positive real numbers?

(b) Suppose we allow our edge capacities to be positive *rational* numbers. Given a graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges and positive rational edge capacities $c(e)$, give an $O(m)$ algorithm that transforms the rational-capacity max-flow problem into the standard integer-capacity max-flow problem. Explain why finding a max flow for the new problem (with integer capacities), means that you have also found a max flow for the original problem.

3) Network flow with vertex capacities.
In the standard Network Flow setting we assume that edges have capacities that limit the amount of flow along an edge, but we are allowed to send as much flow as we want through each vertex. Consider the following alternative version of the Maximum Flow problem:

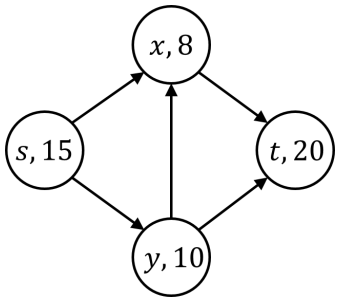
Input:

- A directed, *unweighted* graph $G = (V, E)$, where $|V| = n$ and $|E| = m$
- For each node $v \in V$, a positive integer capacity $c(v)$
- A source node $s \in V$
- A target node $t \in V$

Output: An s-t flow of maximum value that satisfies the usual conservation constraints, as well as new node-capacity constraints $\sum_{(u,v)} f(u, v) \leq c(v)$ (that is, the amount of flow passing through a node can be at most the capacity of the node).

(a) Give an algorithm that solves this problem (i.e., that finds a maximum s-t flow in this node-capacitated graph). Your algorithm should transform the node-capacitated problem into a "standard" max flow problem.

(b) Show how your transformation works on the example graph below (i.e., draw a picture of the new graph produced by your transformation).



(c) What is the runtime required for your transformation (this should be polynomial in n and m)?

(d) Explain why solving the "standard" max flow problem on your transformed graph yields a correct solution to the node-capacitated problem.

3 Applications

4) Blood transfusions.
Suppose you work for a hospital, and your job is to determine whether the current supply of blood will be enough to meet the projected demand over the next week for blood transfusions.

Unfortunately, it's not as simple as checking whether the number of liters currently available is greater than the number of liters required: people have *blood types* that constrain the types of blood they can receive. A person's blood supply has certain antigens present, and a person cannot receive blood that contains a particular antigen if their own blood does not have this antigen. Concretely, blood is divided into four types: A, B, AB, and O. Blood of type A has antigen A, blood of type B has antigen B, blood of type AB has both antigens, and blood of type O has neither. So a patient with type-A blood can receive only blood types A or O in a transfusion, a patient with type-B blood can receive only blood types B or O, a patient with type-AB blood can receive any of the four types, and a patient with type-O blood can only receive blood type O.

(a) Let s_A, s_B, s_{AB}, and s_O denote the supply of each of the different blood types (assume these are all positive integers). Let d_A, d_B, d_{AB}, and d_O denote the demand of each of the different blood types in the next week. Give an algorithm to determine if the current supply of blood is enough to meet the demand. Your algorithm should model the blood-supply problem as a network flow problem (i.e., represent the problem using a graph, and then run a max-flow algorithm on the resulting graph). In your algorithm description, it should be clear (i) what the vertices and edges represent in the graph you create, and (ii) how, after running a max-flow algorithm on the graph you created, you can determine if the current blood supply is sufficient.

(b) Consider the following example:

blood type	**supply**	**demand**
O	50	45
A	36	42
B	11	8
AB	8	3

Show how your algorithm works on this example. You should draw a picture of the graph you create and describe one possible run of Ford-Fulkerson on the graph.

(c) In the example from part (b), is it possible to meet the full demand? Explain why or why not, using an argument based on a min cut.

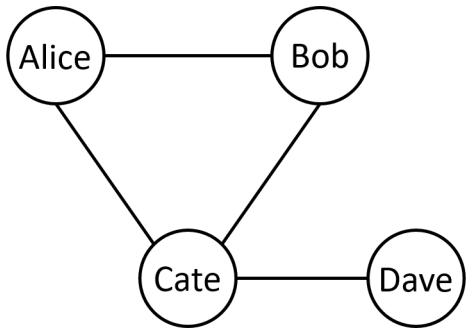
(d) In general, if it is not possible to meet the full demand, the output of your algorithm still is meaningful. What does it tell you?

5) Friend cohesiveness.
Graphs often are used to model friendship connections. For example, one could imagine a graph representing connections on Facebook, where there is a node for each user and an undirected edge between two people if they are friends on Facebook. These types of graphs are used to study things like the spread of information, the spread of disease, and even how to identify pairs of people who are in a romantic relationship.

In this problem, we're concerned with *cohesion* in friend groups—that is, we want to understand how "close-knit" a group is. Let's assume that as described above, we are given a graph $G = (V, E)$, where $|V| = n$ and $|E| = m$, in which the nodes represent people and there is an undirected edge between two nodes if those people are friends. For a subset S of nodes, let $e(S)$ denote the number of edges in S—that is, the number of edges $e = (u, v)$ such that both u and v are in S. We define the *cohesiveness* of S to be $e(S)/|S|$, where $|S|$ is the number of nodes in S.

(a) Give an algorithm that takes a rational number α and determines whether there exists a set S with cohesiveness at least α. Your algorithm should involve solving a max-flow problem on some graph. In your algorithm description, it should be clear (i) what the vertices and edges represent in the graph on which you are running a max-flow algorithm, and (ii) how, after running a max-flow algorithm on your graph, you can determine if there is a group with cohesiveness at least α.

(b) Consider the following example of a friend graph:



Draw a picture of the "transformed" graph that your algorithm produces, on which you then run a max-flow algorithm (you do not need to show the actual run of Ford-Fulkerson).

(c) What is the runtime required for your transformation (this should by polynomial in n and m)?

4 Submission

This assignment is due on Friday, November 17, at 12pm. Please type up your solutions and bring a hard copy of your typed responses to submit in class.