CS375 Assignment 15

Due November 17th 10PM ET

A general note for CS375: When writing up your homework, please write neatly and explain your answers clearly, giving all details needed to make your answers easy to understand. Graders may not award full credit to incomplete or illegible solutions.

You are encouraged to discuss the problems with your classmates, **but your submission should be your own.** When writing your solutions up, you should not be looking at any other student's submission.

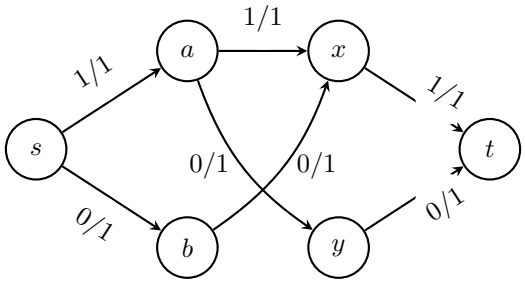
Clear communication **is the point**, on every assignment. In general in CS375, unless explicitly specified otherwise, answers should be accompanied by explanations. Answers without explanations may not receive full credit.

Recall the Maximum Flow problem and the Ford–Fulkerson algorithm from class:

> Maximum Flow
> **Input:** A directed graph $G = (V, E)$, a source vertex s, a sink vertex t, and a capacity $c(e)$ for each edge $e \in E$.
> **Output:** A flow f that satisfies the capacity constraints and flow conservation, and maximizes the total flow from s to t.

Problem 1. Recall the Maximum Flow problem and the Ford–Fulkerson algorithm from class.

Consider the following flow network. Each edge is labeled with its current flow and capacity (in that order).



(a) Find an augmenting path in the residual graph. Explain why this path is an augmenting path.

(b) Augment the flow along your path. Draw the resulting flow network, labeling each edge with its new flow and capacity.

(c) What is the value of the resulting flow? Explain why it is a maximum flow.

(d) Notice that if we only considered paths consisting of forward edges in the original graph, we would not find an augmenting path. Explain why it is important that Ford–Fulkerson considers backward edges in the residual graph.

Problem 2. Recall the following problem discussed in class:

Bipartite Matching

Input: A collection of buyers $B = \{b_1, \ldots, b_n\}$ and items $I = \{i_1, \ldots, i_m\}$. Each buyer b_k is interested in some subset of the items. Each buyer b_k has an (integer) budget q_k, indicating the maximum number of items they can purchase. Each item i_j has a stock s_j, indicating the maximum number of copies available. Assume that each item just costs \$1.

Output: An allocation of items to buyers that maximizes the total number of items sold, subject to the buyers' budgets, the available stock, and the buyers' interests.

In class, we saw how to use a maximum flow algorithm to solve the special case where every buyer has a budget of 1 and every item has a stock of 1.

Show how we can use a maximum flow algorithm to solve the more general problem described above.

Specifically:

(a) Describe how to construct a flow network from an instance of this problem. Specify all vertices, edges, and capacities.

(b) Explain why any feasible allocation of k items gives rise to a flow of value k in your network.

(c) Explain why any integer-valued flow of value k in your network gives rise to a feasible allocation of k items.

(d) Explain why a maximum flow in your network gives an allocation that sells the maximum possible number of items.