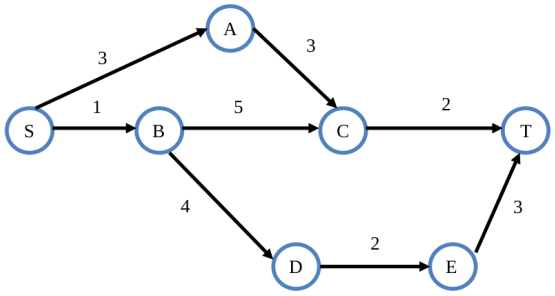
CSCI 570 - Spring 2021 - HW 4

Due April 1, 2021

Note. You are to solve problems 2, 3 and 4 by using the following steps:
1. Describe how to construct a flow network.
2. Make a claim. Something like "this problem has a feasible solution if and only if the max flow is ..."
3. Prove the above claim in both directions

1 Running the Algorithm (15 points)

You are given the following graph G. Each edge is labeled with the capacity of that edge.



a. Find a max-flow in G using the Ford-Fulkerson algorithm. Draw the residual graph G_f corresponding to the max flow. You do not need to show all intermediate steps. (10 pts)

b. Find the max-flow value and a min-cut. (5 pts)

2 Trading Currencies (20 points)

A group of traders are leaving Switzerland, and need to convert their Francs into various international currencies. There are n traders $t_1, t_2, ..., t_n$ and m currencies $c_1, c_2, ..., c_m$. Trader t_k has F_k Francs to convert. For each currency c_j, the bank can convert at most B_j Franks to c_j. Trader t_k is willing to trade as much as S_{kj} of his Francs for currency c_j.(For example, a trader with 1000 Francs might be willing to convert up to 200 of his Francs for USD, up to 500 of his Francs for Japanese's Yen, and up to 200 of his Francs for Euros). Assuming that all traders give their requests to the bank at the same time, design an algorithm that the bank can use to satisfy the requests (if it is possible).

3 Getting Vaccinated (20 points)

After getting vaccinated, students will be able to return to in-person classes next semester. There will be n students, $s_1, s_2, ..., s_n$, return to in-person classes, and there will be k in-person classes. Each in-person class consists of several students and a student can be enrolled in more than one in-person class. We need to select one student from each in-person class and the maximum times a student is selected should be less than m. Design an algorithm that decides if such a selection exists, or not.

4 Visiting Campus (25 points)

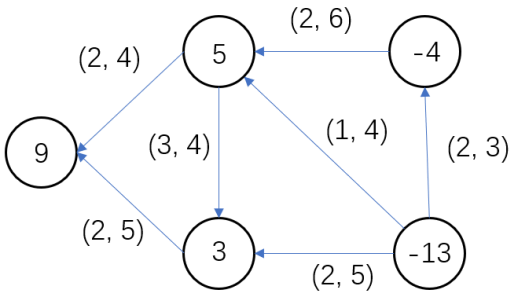
USC Admissions Center needs your help in planning paths for Campus tours given to prospective students or interested groups. Let USC campus be modeled as a weighted, directed graph G containing locations V connected by one-way roads E. On a busy day, let k be the number of campus tours that have to be done at the same time. It is required that the paths of campus tours do not use the same roads. Let the tour have k starting locations $A = \{ a_1, a_2, ..., a_k \} \subset V$. From the starting locations the groups are taken by a guide on a path through G to some ending location in $B = \{ b_1, b_2, ..., b_k \} \subset V$. Your goal is to find a path for each group i from the starting location, a_i , to any ending location b_j such that no two paths share any edges, and no two groups end in the same location b_j.

a. Design an algorithm to find k paths $a_i \rightarrow b_j$ that start and end at different vertices and such that they do not share any edges. (15 pts)

b. Modify your algorithm to find k paths $a_i \rightarrow b_j$, that start and end in different locations and such that they share neither vertices nor edges. (10 pts)

5 Circulation (20 points)

In the network below, the demand values are shown on vertices (supply value if negative). Lower bounds on flow and edge capacities are shown as (lower bound, capacity) for each edge. Determine if there is a feasible circulation in this graph. You need to show all your steps.



a. Turn the circulation with lower bounds problem into a circulation problem without lower bounds. (8 pts)

b. Turn the circulation with demands problem into the maximum flow problem. (8 pts)

c. Does a feasible circulation exist? Explain your answer. (4 pts)

6 Installing Software (Not Graded)

Suppose that you have just bought a new computer and you want to install software on that. Specifically, two companies, which you can think of like Microsoft and Apple, are trying to sell their own copy of n different products, like Operation System. Spread Sheet, Web Browser. For each product i, $i \in \{1, 2, \ldots, n\}$, we have

- the price $p_i \geq 0$ that Microsoft charges and the price $p'_i \geq 0$ that Apple charges.
- the quality $q_i \geq 0$ of Microsoft version and the quality $q'_i \geq 0$ of Apple version.

For example, Apple may provide a better Web Browser Safari, but Microsoft a better Word Processor. You want to assemble your favorite computer by installing exactly one copy of each of the n products, e.g. you want to buy one operating system, one Web Browser, one Word Processor, etc. However, you don't want to spend too much money on that. Therefore, your goal is to maximize the quality minus total price.

However, as you may know, the products of different companies may not be compatible. More concretely, for each product pair (i, j), we will suffer a penalty $\tau_{ij} \geq 0$ if we install product i of Microsoft and product j of Apple. Note that τ_{ij} may not be equal to τ_{ji} just because Apple's Safari does not work well on Microsoft Windows doesn't mean that Microsoft's Edge does not work well in Mac-OS. We assume that products are always compatible internally, which means that there is no penalty for installing two products from the same company. All pairwise penalties will be subtracted from the total quality of the system.

Your task is then to give a polynomial-time algorithm for computing which product i to purchase from which of the two companies (Apple and Microsoft) for all $i \in \{1, 2, \ldots, n\}$, to maximize the total system quality (including the penalties) minus the total price.