CS 170 Homework 9

Due **Sunday 11/3/2024, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, explicitly write "none".

2 Applications of Max-Flow Min-Cut

Review the statement of max-flow min-cut theorem and prove the following two statements.

(b) Let $G = (L \cup R, E)$ be a unweighted bipartite graph[1]. Then G has a L-perfect matching (a matching[2] with size $|L|$) if and only if, for every set $X \subseteq L$, X is connected to at least $|X|$ vertices in R. You must prove both directions.

Hint: Use the max-flow min-cut theorem on the cut that forms X and $L \setminus X$.

[1] A bipartite graph $G = (L \cup R, E)$ is defined as a graph that can be partitioned into two disjoint sets of vertices (i.e. L and R) such that no two vertices within the same set are adjacent.

[2] A matching is defined as a set of edges that share no common vertices.

3 A Cohort of Secret Agents

A cohort of k secret agents residing in a certain country needs escape routes in case of an emergency. They will be travelling using the railway system which we can think of as a directed graph $G = (V, E)$ with V being the cities and E being the railways. Each secret agent i has a starting point $s_i \in V$, and all s_i's are distinct. Every secret agent needs to reach the consulate of a friendly nation; these consulates are in a known set of cities $T \subseteq V$. In order to move undetected, the secret agents agree that at most c of them should ever pass through any one city. Our goal is to find a set of paths, one for each of the secret agents (or detect that the requirements cannot be met).

Model this problem as a flow network. Specify the vertices, edges, and capacities; show that a maximum flow in your network can be transformed into an optimal solution for the original problem. You do not need to explain how to solve the max-flow instance itself.

4 Office Hour Scheduling

You're the new CS 170 head TA and need to assign TAs to host office hours. Unfortunately for you, the department has bizarre rules about when TAs can hold office hours:

- There are N office hours slots in total. You can assume that all N slots are consecutive.
- There are T TAs in total. The i^{th} TA is only available to hold office hours during the contiguous block from slot a_i to slot b_i. The numbers $\{a_i, b_i\}$ for $i \in \{1 \dots T\}$ are given as input.
- Each office hours slot must have exactly k TAs (no more, and no less).
- Not every TA needs to hold office hours.
- If a TA holds office hours, department rules require them to hold office hours for their entire availability, e.g. the i^{th} TA either holds office hours for every slot from a_i to b_i, or does not hold office hours at all.

Your goal is to determine which TAs to assign office hours so that all the above constraints are met.

Devise an efficient algorithm that solves the problem by constructing a directed graph G with a source s and sink t, and computing the maximum flow from s to t.

(a) Describe the nodes of the graph G. How many nodes does G have?

(b) Describe the edges of the graph G (draw a picture if needed). How many edges does G have?

(c) Given a maximum flow in the graph G, how can you determine which TAs to assign to hold office hours?

5 Weighted Rock-Paper-Scissors

You and your friend used to play rock-paper-scissors, and have the loser pay the winner 1 dollar. However, you then learned in CS170 that the best strategy is to pick each move uniformly at random, which took all the fun out of the game.

Your friend, trying to make the game interesting again, suggests playing the following variant: If you win by beating rock with paper, you get 2 dollars from your opponent. If you win by beating scissors with rock, you get 1 dollars. If you win by beating paper with scissors, you get 4 dollar.

For this problem, you are allowed to use code or online tools to automatically solve your LPs. The coding problem (Q6) walks you through how to express your LPs in `Python` and solve them using a `Python` library. However, if you prefer, feel free to use a different LP solver such as: `https://online-optimizer.appspot.com/`.

(a) Draw the payoff matrix for this game. Assume that you are the maximizer, and your friend is the minimizer.

(b) Write an LP to find the optimal strategy in your perspective. You do not need to solve the LP.

The following subparts are independent of the previous subparts.

Your friend now wants to make the game even more interesting and suggests that you assign points based on the following payoff matrix:

		Your friend: rock	paper	scissors
	rock	-10	3	3
You:	paper	4	-1	-3
	scissors	6	-9	2

(c) Write an LP to find the optimal strategy for yourself. What is the optimal strategy and expected payoff?

Feel free to use your code from Q6 or an online LP solver.

(d) Now do the same for your friend. What is the optimal strategy and expected payoff? How does the expected payoff compare to the answer you get in part (c)?

6 [Coding] LP / Max Flow / Min Cut

For this week's homework, we'll walk you through solving LPs in Python, and then you'll implement an algorithm to compute the maximum flow in a network and then compute the minimum cut from the flow. There are two ways that you can access the notebook and complete the problems:

1. **On Datahub**: click here and navigate to the `hw09` folder.
2. **On Local Machine**: `git clone` (or if you already cloned it, `git pull`) from the coding homework repo,

   `https://github.com/Berkeley-CS170/cs170-fa24-coding`

   and navigate to the `hw09` folder. Refer to the `README.md` for local setup instructions.

Notes:

- *Submission Instructions:* Please download your completed submission `.zip` file and submit it to the Gradescope assignment titled "Homework 9 Coding Portion".
- *Getting Help:* Conceptual questions are always welcome on Edstem and office hours; *note that support for debugging help during OH will be limited.* If you need debugging help first try asking on the public Edstem threads. To ensure others can help you, make sure to:

  1. Describe the steps you've taken to debug the issue prior to posting on Ed.
  2. Describe the specific error you're running into.
  3. Include a few small but nontrivial test cases, alongside both the output you expected to receive and your function's actual output.

  If staff tells you to make a private Ed post, make sure to include *all of the above items* plus your full function implementation. If you don't provide them, we will ask you to provide them.
- *Academic Honesty Guideline:* We realize that code for some of the algorithms we ask you to implement may be readily available online, but we strongly encourage you to not directly copy code from these sources. Instead, try to refer to the resources mentioned in the notebook and come up with code yourself. That being said, we **do acknowledge** that there may not be many different ways to code up particular algorithms and that your solution may be similar to other solutions available online.

