CS 170 HW 10

Due **2021-04-13, at 10:00 pm**
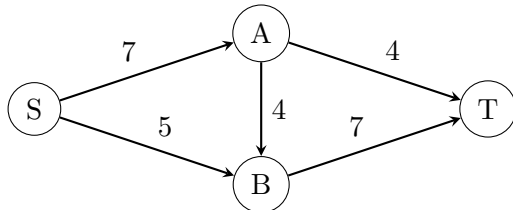
1 Study Group

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

2 Max Flow, Min Cut, and Duality

In this exercise, we will demonstrate that LP duality can be used to show the max-flow min-cut theorem.

Consider this instance of max flow:

Let f_1 be the flow pushed on the path $\{S, A, T\}$, f_2 be the flow pushed on the path $\{S, A, B, T\}$, and f_3 be the flow pushed on the path $\{S, B, T\}$. The following is an LP for max flow in terms of the variables f_1, f_2, f_3:

$$
\begin{aligned}
\max \quad & f_1 + f_2 + f_3 \\
& f_1 + f_2 \leq 7 && \text{(Constraint for } (S, A)) \\
& f_3 \leq 5 && \text{(Constraint for } (S, B)) \\
& f_1 \leq 4 && \text{(Constraint for } (A, T)) \\
& f_2 \leq 4 && \text{(Constraint for } (A, B)) \\
& f_2 + f_3 \leq 7 && \text{(Constraint for } (B, T)) \\
& f_1, f_2, f_3 \geq 0
\end{aligned}
$$

The objective is to maximize the flow being pushed, with the constraint that for every edge, we can't push more flow through that edge than its capacity allows.

(a) Find the dual of this linear program, where the variables in the dual are x_e for every edge e in the graph.

(b) Consider any cut in the graph. Show that setting $x_e = 1$ for every edge crossing this cut and $x_e = 0$ for every edge not crossing this cut gives a feasible solution to the dual program.

(c) Based on your answer to the previous part, what problem is being modelled by the dual program? By LP duality, what can you argue about this problem and the max flow problem?

3 How to Gamble With Little Regret

Suppose that you are gambling at a casino. Every day you play at a slot machine, and your goal is to minimize your losses. We model this as the experts problem. Every day you must take the advice of one of n experts (i.e. a slot machine). At the end of each day t, if you take advice from expert i, the advice costs you some c_i^t in $[0, 1]$. You want to minimize the regret R, defined as:

$$R = \frac{1}{T}\left(\sum_{t=1}^{T} c_{i(t)}^t - \min_{1 \le i \le n}\sum_{t=1}^{T} c_i^t\right)$$

where $i(t)$ is the expert you choose on day t. Your strategy will be probabilities where p_i^t denotes the probability with which you choose expert i on day t. Assume an all powerful adversary (i.e. the casino) can look at your strategy ahead of time and decide the costs associated with each expert on each day. Let C denote the set of costs for all experts and all days. Compute $\max_C(\mathbb{E}[R])$, or the maximum possible (expected) regret that the adversary can guarantee, for each of the following strategies.

(a) Choose expert 1 at every step, i.e. $p_1^t = 1$ for all t.

(b) Any deterministic strategy, i.e. for each t, there exists some i such that $p_i^t = 1$.

(c) Always choose an expert according to some fixed probability distribution at every time step. That is, fix some $p_1, \ldots, p_n$, and for all t, set $p_i^t = p_i$.

   What distribution minimizes the regret of this strategy? In other words, what is $\text{argmin}_{p_1,\ldots,p_n} \max_C(\mathbb{E}[R])$?

This analysis should conclude that a good strategy for the problem must necessarily be randomized and adaptive.

4 Global Mincut to Min s-t Cut

In class, we showed how to use maximum s-t flow to solve the minimum s-t cut. Now we ask you to consider the *global mincut* problem. Given a weighted, undirected graph G, the problem asks you to find a minimum weight set of edges whose removal disconnects the graph. (Note that there is no notion of s and t).

Show how to use maxflow or min s-t cut algorithms to solve this problem efficiently. Prove that your reduction would lead to the optimal solution. How many times do you need to invoke the maxflow or min s-t cut subroutine?

5 Dijsktra's Sort

Show how to use Dijsktra's algorithm to sort n real numbers (not necessarily non-negative or integral) in ascending order in $O(n \log n)$ time. You may use the intermediate outputs of Dijsktra's to do the sorting (as opposed to using it as a blackbox).

Argue that this means that improving upon the $O(m + n \log n)$ run-time of Dijsktra's algorithm (with Fibonacci heap) would lead to a faster sorting algorithm than merge and quick sort.

Hint: Given the numbers, construct a graph such that the order of vertices being extracted from priority queue by Dijsktra's algorithm corresponds to the right sorting order.