CS 170 HW 8

Due **2020-10-26, at 10:00 pm**

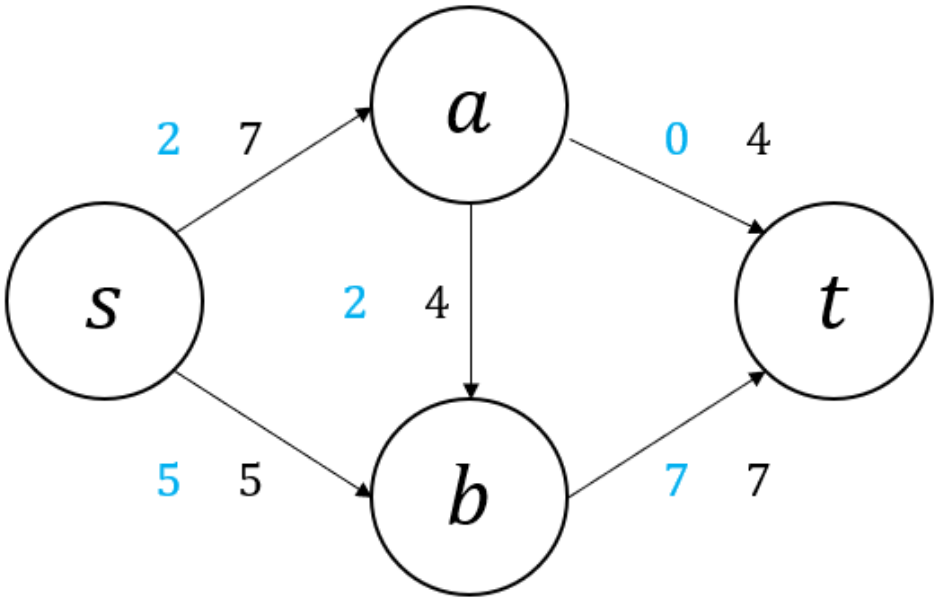
1 Study Group

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

In addition, we would like to share correct student solutions that are well-written with the class after each homework. Are you okay with your correct solutions being used for this purpose? Answer "Yes", "Yes but anonymously", or "No"

2 Practice With Residual Graphs

(a) Consider the following network and flow on this network. An edge is labelled with its flow value (in blue) and capacity (in black). e.g. for the edge (s, a), we are currently pushing 2 units of flow on it, and it has capacity 7.

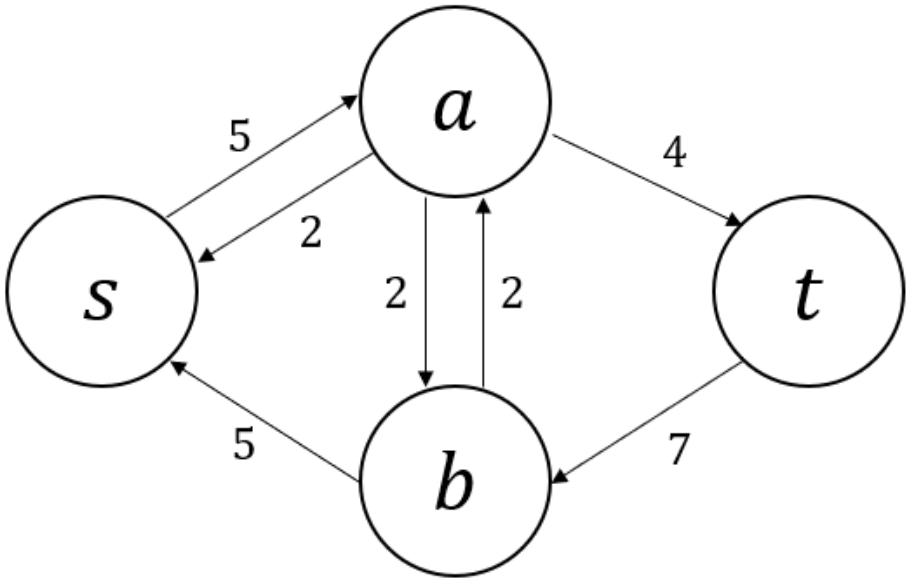


Draw the residual graph for this flow.

(b) We are given a network $G = (V, E)$ whose edges have integer capacities c_e, and a maximum flow f from node s to node t. Explicitly, f is given to us in the representation of integer flows along every edge e, (f_e).
However, we find out that one of the capacity values of G was wrong: for edge (u, v), we used c_{uv} whereas it really should have been $c_{uv} - 1$. This is unfortunate because the flow f uses that particular edge at full capacity: $f_{uv} = c_{uv}$. We could run Ford Fulkerson from scratch, but there's a faster way.

Describe an algorithm to fix the max-flow for this network in $O(|V| + |E|)$ time. Give a three-part solution.

Solution:



(a)

(b) **Main Idea:** Since we know the flows along every edge, we can construct the residual graph in $O(|V| + |E|)$ time as there are $2|E|$ edges in the residual graph.

In this residual graph with the original capacity for (u, v), use DFS to find a path from t to v and from u to s. Join these paths by adding the edge (v, u) in between them to get a path p. Update f by pushing 1 unit of flow from t to s on p (i.e. for each $(i, j) \in p$, reduce f_{ji} by 1, and increase f_{ij} by 1 for all edges except (u, v)). Finally, use DFS to see if the residual graph of the resulting graph, has an $s - t$ path. If so, push 1 unit of flow on this path.

Proof of Correctness: Consider the residual graph of G. If (u, v) is at capacity, then there is a flow of at least 1 unit going from v to t, and since f_e are integral there is a path from v to t with at least one unit of flow moving through it. So the residual graph will have a path from t to v. Similarly, there will be a path from u to s in the residual graph, so the algorithm can always find p correctly.

Note that the size of the maximum flow in the new network will be either the same or 1 less than the previous maximum flow (because changing one edge can change the capacity of the min-cut by at most 1). In the former case, after pushing flow from t to s on p, it is possible to push more flow from s to t, so there must be an $s - t$ path in the new residual graph, so the Ford Fulkerson algorithm will find and push 1 unit of flow because all capacities and flow values are integral. Thus the algorithm finds the new max flow correctly. In the latter case, we will already have the max flow after pushing flow backwards on p, so our algorithm is still correct.

Runtime: The runtime is $O(|V| + |E|)$ because the algorithm just calls DFS on the residual thrice, and it takes $O(|V| + |E|)$ to construct the residual.

3 Global Min-Cut via Ford-Fulkerson

Given a connected undirected unweighted graph G, recall that a cut is any partition of V into two non-empty sets A, B, and the size of the cut is the number of edges with one endpoint in A and one in B. The global min-cut is the cut with the smallest size.

Give an algorithm based on the Ford-Fulkerson algorithm that given G and c, either outputs the global min-cut if it has size at most c, or correctly outputs that the global min-cut has size greater than c. The algorithm should run in time $O(|V||E|c)$. Give a three-part solution.

(Recall that the Ford-Fulkerson algorithm repeatedly finds an s-t path in the residual graph and pushes as much flow as possible on this path)

Solution: Main idea: Fix some vertex s. We will run Ford-Fulkerson to find the max-flow from s to every other vertex, but we will only use $c + 1$ iterations of Ford-Fulkerson in each run.

If any run of Ford-Fulkerson is unable to increase its flow value at any point in the first $c + 1$ iterations, we know there is no augmenting path from s to t in the residual graph, i.e. the vertices reachable from s form one side of the minimum $s - t$ cut. In this case, we will store this cut and its size.

At the end, if any of the cuts we stored has size at most c, we output the smallest cut we stored. Otherwise we output that the global min-cut has size greater than c.

Proof of correctness: By the max-flow min-cut theorem, if the global min-cut has size at most c, the max-flow between two vertices on opposite sides of this cuts is at most c. In particular, s and some vertex t are on opposite sides of this cut. When we run Ford-Fulkerson between s and t, each iteration increases the size of the flow by at least 1, so we will find a max $s - t$ flow in at most c iterations, and thus find a min $s - t$ cut in this many iterations. So we always find the global min-cut if it has size at most c.

Runtime analysis: Each iteration of Ford-Fulkerson takes $O(|E|)$ time, and we run $O(|V|)$ copies for $c + 1$ iterations, so the runtime is $O(|V||E|c)$.

4 Meal Replacement

We are trying to eat cheaply but still meet our minimum dietary needs. We want to consume at least 500 calories of protein per day, 100 calories of carbs per day, and 400 calories of fat per day. We have three options for food we're considering buying: meat, bread, and protein shakes.

- We can consume meat, which costs 5 dollars per pound, and gives 500 calories of protein and 500 calories of fat per pound.
- We can consume bread, which costs 2 dollars per pound, and gives 50 calories of protein, 300 calories of carbs, and 25 calories of fat per pound.

- We can consume protein shakes, which cost 4 dollars per pound, and gives 300 calories of protein, 100 calories of carbs, and 200 calories of fat per pound.

Our goal is to find a combination of these options that meets our daily dietary needs while being as cheap as possible.

(a) Formulate this problem as a linear program.

(b) Take the dual of your LP from part (a).

(c) Suppose now there is a pharmacist trying to assign a price to three pills, with the hopes of getting us to buy these pills instead of food. Each pill provides exactly one of protein, carbs, and fiber.

Interpret the dual LP variables, objective, and constraints as an optimization problem from the pharmacist's perspective.

Solution:

(a) Let m be the pounds of meat we consume, b be the pounds of bread, and s be the pounds of shakes. The objective is to minimize cost, and we have a constraint for each of protein/carbs/fat.

$$\begin{aligned} \min\ & 5m + 2b + 4s \\ 500m + 50b + 300s &\geq 500 \\ 300b + 100s &\geq 100 \\ 500m + 25b + 200s &\geq 400 \\ m, b, s &\geq 0 \end{aligned}$$

(b) Let p, c, f be variables corresponding to the protein, carb, and fat constraints. The dual is:

$$\begin{aligned} \max\ & 500p + 100c + 400f \\ 500p + 500f &\leq 5 \\ 50p + 300c + 25f &\leq 2 \\ 300p + 100c + 200f &\leq 4 \\ p, c, f &\geq 0 \end{aligned}$$

(c) The variables can be interpreted as the price per calorie for the protein, carb, and fat pills.

The objective says that the pharmacist wants to maximize the total revenue he gets from selling enough of these pills to us to meet our dietary needs.

The constraints say that no combination of pills should cost more than a pound of food providing the same dietary needs (otherwise, we would just buy that food instead of these pills).

5 Vertex Cover Dual

This is a solo question.

In this problem, we consider the unweighted vertex cover problem. In this problem, we are given a graph and want to find the smallest set of vertices S such that every edge has at least one endpoint in S.

Recall the LP for this problem:

$$\min \sum_v x_v \text{s. t. } \forall (u,v) \in E : x_u + x_v \geq 1, \forall v \in V : x_v \geq 0$$

In an integral solution, $x_v = 1$ if we include v in our vertex cover, and $x_v = 0$ if we don't include v.

(i) Write the dual of the vertex cover LP. Your dual LP should have a variable y_e for every edge.

(ii) Consider the integer version of the dual you wrote, i.e. we enforce $y_e \in \{0, 1\}$. Similarly to vertex cover, we can interpret $y_e = 1$ as indicating that we include e in our solution and $y_e = 0$ if we don't include e.

Using this interpretation, what does the objective say? What do the constraints say? What problem is this? (You don't have to be formal.)

(iii) True or False: If we have an integer primal solution with cost C and a fractional dual solution with cost at least $C/2$, the size of the vertex cover corresponding to the primal solution is at most twice the size of the smallest vertex cover. Briefly justify your answer.

Solution:

(i) Let E_v all the edges incident on vertex v. The dual LP is given by

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} y_e \\ \text{subject to} \quad & \sum_{e \in E_v} y_e \leq 1 \quad \forall v \in V \\ & y_e \geq 0 \qquad \forall e \in E \end{aligned}$$

(ii) The objective is to maximize the number of edges in our solution.

The constraint says no two edges in our solution share an endpoint.

This is exactly the maximum matching problem. (You don't need to name the maximum matching problem for full credit).

(iii) True. The size of the vertex cover corresponding to the integer primal solution is C. Let P_{OPT} be the optimal value of the primal LP, D_{OPT} be the optimal value of the dual LP, and V_{OPT} be the size of the smallest vertex cover. $C/2 \leq D_{OPT}$ since the dual optimal solution's objective is at least that of any feasible dual solution, $D_{OPT} = P_{OPT}$ by duality, and $P_{OPT} \leq V_{OPT}$ because the smallest vertex cover gives a feasible solution to the vertex cover LP with cost V_{OPT}. Putting it all together, we have $C \leq 2V_{OPT}$.

6 Domination

This is a solo question.

In this problem, we explore a concept called *dominated strategies*. Consider a zero-sum game with the following payoff matrix for the row player:

		Column: A	B	C
Row:	D	1	2	-3
	E	3	2	-2
	F	-1	-2	2

(a) If the row player plays optimally, can you find the probability that they pick D without directly solving for the optimal strategy? Justify your answer.

(Hint: How do the payoffs for the row player picking D compare to their payoffs for picking E?)

(b) Given the answer to part a, if the both players play optimally, what is the probability that the column player picks A?

(c) Given the answers to part a and b, what are both players' optimal strategies?

Note: All parts of this problem can be solved without using an LP solver or solving a system of linear equations.

Solution:

(a) 0. Regardless of what option the column player chooses, the row player always gets a higher payoff picking E than D, so any strategy that involves a non-zero probability of picking D can be improved by instead picking E.

(b) 0. We know that the row player is never going to pick D, i.e. will always pick either E or F. But in this case, picking B is always better for the column player than picking A (A is only better if the row player picks D). That is, conditioned on the row player playing optimally, B dominates A.

(c) Based on the previous two parts, we only have to consider the probabilities the row player picks E or F and the column player picks B or C. Looking at the 2-by-2 submatrix corresponding to these options, it follows that the optimal strategy for the row player is to pick E and F with probability 1/2, and similarly the column player should pick B, C with probability 1/2.

7 Zero-Sum Battle

This is a solo question.

Two Pokemon trainers are about to engage in battle! Each trainer has 3 Pokemon, each of a single, unique type. They each must choose which Pokemon to send out first. Of course

each trainer's advantage in battle depends not only on their own Pokemon, but on which Pokemon their opponent sends out.

The table below indicates the competitive advantage (payoff) Trainer A would gain (and Trainer B would lose). For example, if Trainer B chooses the fire Pokemon and Trainer A chooses the rock Pokemon, Trainer A would have payoff 2.

		Trainer B: ice	water	fire
Trainer A:	dragon	-10	3	3
	steel	4	-1	-3
	rock	6	-9	2

Feel free to use an online LP solver to solve your LPs in this problem.
Here is an example of an online solver that you can use: `https://online-optimizer.appspot.com/`.

1. Write an LP to find the optimal strategy for Trainer A. What is the optimal strategy and expected payoff?

2. Now do the same for Trainer B. What is the optimal strategy and expected payoff? How does the expected payoff compare to the answer you get in part (a)?

Solution:

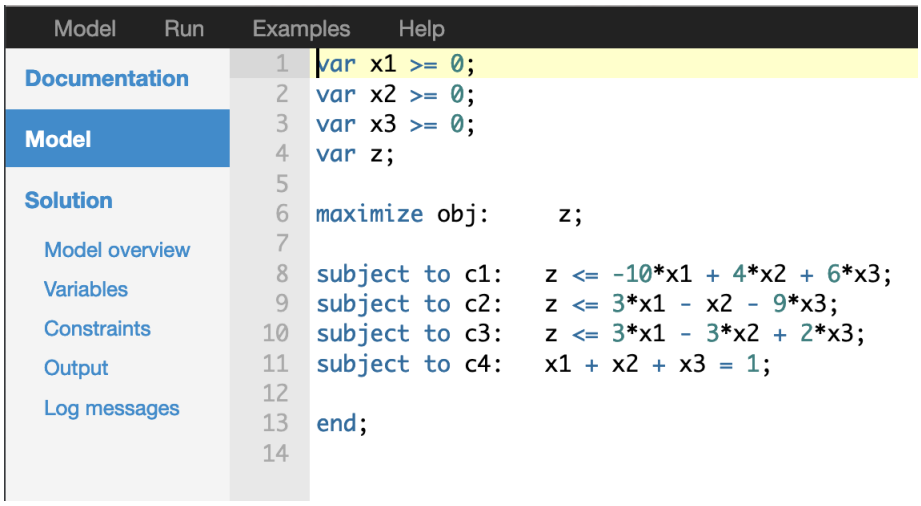
1. d = probability that A picks the dragon type
   s = probability that A picks the steel type
   r = probability that A picks the rock type

$$\begin{aligned} \max \quad & z & \\ -10d + 4s + 6r &\geq z & \text{(B chooses ice)} \\ 3d - s - 9r &\geq z & \text{(B chooses water)} \\ 3d - 3s + 2r &\geq z & \text{(B chooses fire)} \\ d + s + r &= 1 & \\ d, s, r &\geq 0 & \end{aligned}$$

   The optimal strategy is $d = 0.3346,\ s = 0.5630,\ r = 0.1024$ for an optimal payoff of -0.48.

   If you are using the website suggested in this problem, here is what you should put in the model tab:

2. i = probability that B picks the ice type
   w = probability that B picks the water type
   f = probability that B picks the fire type

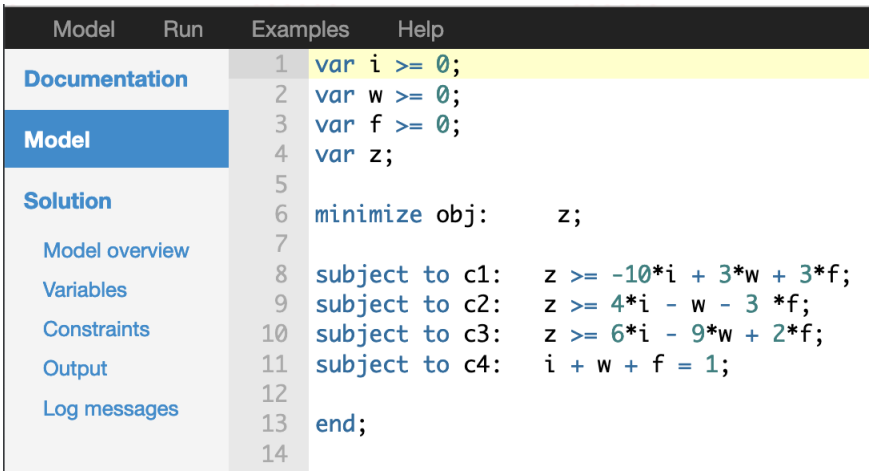


$$\begin{aligned} \min \quad & z & \\ -10i + 3w + 3f &\leq z & \text{(A chooses dragon)} \\ 4i - w - 3f &\leq z & \text{(A chooses steel)} \\ 6i - 9w + 2f &\leq z & \text{(A chooses rock)} \\ i + w + f &= 1 & \\ i, w, f &\geq 0 & \end{aligned}$$

B's optimal strategy is $i = 0.2677,\ w = 0.3228,\ f = 0.4094$. The value for this is -0.48, which is the payoff for A. The payoff for B is the negative of A's payoff, i.e. 0.48, since the game is zero-sum.

If you are using the website suggested in this problem, here is what you should put in the model tab:

(Note for grading: Equivalent LPs are of course fine. It is fine for part (b) to maximize B's payoff instead of minimizing A's. For the strategies, fractions or decimals close to the solutions are fine, as long as the LP is correct.)


Model
Run
Examples
Help
Documentation
Model
Solution
Model overview
Variables
Constraints
Output
Log messages
1 var i >= 0;
2 var w >= 0;
3 var f >= 0;
4 var z;
5
6 minimize obj: z;
7
8 subject to c1: z >= -10*i + 3*w + 3*f;
9 subject to c2: z >= 4*i - w - 3 *f;
10 subject to c3: z >= 6*i - 9*w + 2*f;
11 subject to c4: i + w + f = 1;
12
13 end;
14