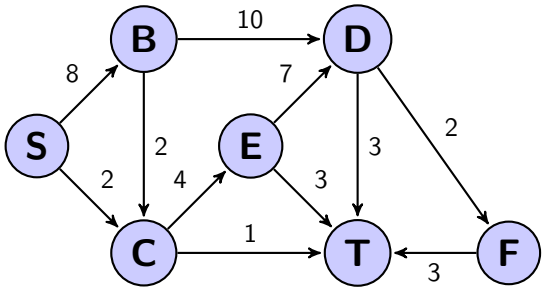
This homework is due by the start of class on Friday, Mar 11th. You must submit the homework via the course page on T-Square, and you may hand in a printed version with any graph drawings, if you choose not to include them in the online copy.

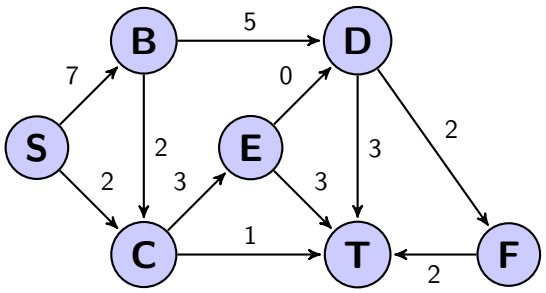
Homework Policies:

- Your work will be graded on correctness and clarity. Write complete and precise answers.
- You may collaborate with up to *three* other classmates on this problem set. However, you must *write your own solutions* in your own words and *list your collaborators*.
- You may portions of our two textbooks that we have covered. The purpose of these homeworks is NOT only to teach you the algorithms that we cover, but also to teach you how to problem-solve. Therefore using *ANY* outside resources like the internet, previous students etc. is *not* allowed!
- **Optional** questions are for you to practice and learn the basics before you advance. They may ask you to follow the steps of an algorithm covered in class, or review material from an earlier class. Solutions to these problems will not be graded, but you must be are able to do them, and you are responsible for material that is addressed by these optional questions.
- **Basic** questions *must be solved and written up alone*; you may not collaborate with others. These will help you build the framework needed to solve the harder questions. Please do these first, before meeting in groups, for both you and your group's benefit.
- **Group** questions may be solved in collaboration with up to ***three*** other classmates. However, you must still *write up your own solutions* alone and in your own words. These will form the bulk of your homework questions.
- **Bonus** questions are like group questions, but they will be more challenging and will be graded more rigorously. Bonus points are tallied separately from your normal points, and will be applied to your grade *after* the curve. Bonus points are worth more than normal points, and can have a significant positive affect on your grade. Most importantly, bonus problems are fun!

1. (optional) Find a max-flow from s to t in this flow network, and prove that it is optimal by finding a corresponding min-cut with the same value.



Solution: The max flow has value 9. The flow is



The cut (S, B, D) has a total outflow of 9, so we know that 9 is optimal.

2. (10 points) (basic) Say you are given a flow graph G with multiple sources and sinks (disjoint sources and sinks). Give a max-flow based algorithm that can find the maximum total flow in this graph from all sources to all sinks. Hint: Run the standard max-flow algorithm on a modified graph.

Solution: Create a new source node s and attach it to all the source nodes with edges of infinite capacity. Similarly create a source node t, and attach edges from all the sink nodes to t with infinite capacity.

There is a one-to-one correspondence between flows from s to t in our new graph and multiple source/sink flows in the original graph. To see this, we note that every flow from multiple sources to sinks in the original graph can be represented as an $s - t$ flow in the new graph, where the new flow from s to any source is the total outflow from the source in the original flow, and the new flow from any sink to t is the total inflow to the sink in the original flow. Similarly, there is a backwards correspondence as well, any new $s - t$ flow corresponds to a multiple source-sink flow.

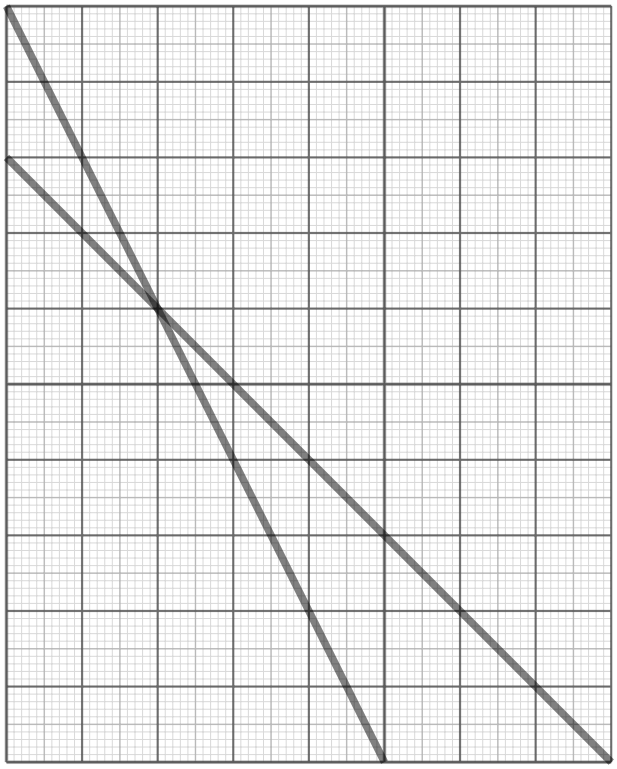
Therefore, $s - t$ flows in the new graph are in bijection with max sources/sinks flows in the original. Running the max flow algorithm in the new graph will therefore get us a max sources/sinks from in the original.

(All this explanations isn't necessary for your homework, I just wanted to be extra rigorous.)

3. (10 points) (basic) You have 8 acres of land on which to plant wheat or barley. You earn \$5000 per acre of wheat, and \$3000 per acre of barley. Each acre of wheat needs 2 tons of manure, and each acre of barley needs 1 ton of manure, and you only have 10 tons. Formulate an LP to find the maximum profit that you can make, and then find it by drawing the feasible region of the LP and testing all the vertices. You don't need to submit a drawing, but specify what the coordinates of the vertices are, and what the profit at each vertex is.

Solution:

$$\begin{aligned} \max \ \ & 5000W + 3000B \\ W, B &\geq 0 \\ W + B &\leq 8 \\ 2W + B &\leq 10 \end{aligned}$$

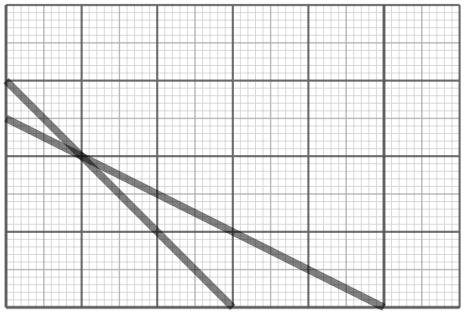


The (W, B) vertices of intersection are $(0, 8)$, $(2, 6)$, and $(5, 0)$ (and $(0, 0)$). By testing all of these, we see that the maximum profit can be had at $(2, 6)$ with \$28000 profit.

4. (15 points) (group) State the dual LP of the above problem, and find an assignment of the dual variables that matches the optimal solution of the original.

Solution: The dual problem is

$$\begin{aligned} \min \ \ & 8X + 10Y \\ X, Y &\geq 0 \\ X + 2Y &\geq 5000 \\ X + Y &\geq 3000 \end{aligned}$$



Of the points of intersection, the dual minimum occurs at $(1000, 2000)$, for a value of \$28000, which matches the maximum of the primal problem.

5. (15 points) (group) Everyday, you walk by a specialty frozen yogurt place that sells only one type of gourmet yogurt per day. You can buy as much as you want, and the schedule for the next n days is posted in advance. On day i, the yogurt for that day will give you h_i happiness per ounce that you eat. However, your doctor told you not to eat more than $M = 100$ ounces of yogurt in **any** k day period, where $k << n$.

   Formulate an LP to find the maximum sum of happiness you can get over the course of the next n days. Determine the total number of variables and contraints as a function of M, k, and n.

Solution: Create a value y_i to represent the amount of yogurt you eat at day i. The happiness that you are trying to maximize is then

$$\max \sum_i y_i \cdot h_i.$$

Your constraints, translated, are that for every j from 1 to $n - k + 1$, $\sum_{i=0}^{k-1} y_{j+i} \leq M$.

You have a total of $n - k + 1$ constraints, and n variables.

6. (20 points) (group) Say you are given a flow graph G with capacities on both edges c_e and vertices c_v. We now also have the restriction that the total amount of flow passing through a vertex must be at most c_v. Design a max-flow based algorithm that can find the max flow in this new type of flow graph, with proof of correctness and analysis of runtime.

Solution: Create a new graph G. For every vertex v, create two vertices v_{in} and v_{out}. For every edge (u, v), add the edge from (u_{out}, v_{in}) with the same capacity. Add an edge from (v_{in}, v_{out}) with capacity c_v. The total flow passing through v must then go through this edge, since all incoming edges go into v_{in} and all outgoing edges come out from v_{out}. Thus we have restricted the total flow passing through v by c_v, while maintaining all the edge capacities between vertices. We can now run a max-flow algorithm from s_{out} to t_{in}.

7. (30 points) (group) Say we are playing a tournament with many games, and we are partway through the season. The problem here is to decide if one team has no possible chance of finishing

in (or tieing for) first place. This can be a subtle problem, and sports writers regularly get it wrong. Here's a hypothetical situation with 5 teams.

Team	Win-Loss	Remaining	A	B	C	D	E
A	22-5	17	-	5	4	3	5
B	21-6	17	5	-	2	5	4
C	15-11	17	4	2	-	7	4
D	13-14	17	3	5	7	-	2
E	9-18	15	5	4	4	2	-

If team E wins all 15 remaining games, they would have 24 wins total, which makes it seem like they might have a shot at winning. However, Teams A and B have 5 remaining games with each other, and the only way for E to finish in first is if team A wins two of it's games against B, and B wins the remaing 3, and both A and B lose all games against all the other teams. But then teams C and D will each get 21 wins, and the 7 games between C and D will leave one of them with at least 25 wins. Thus team E can never win.

In an real world situation with many teams, finding a logical argument like the above to show that a team can never win can get arbitrarily complex.

Design an algorithm that can decide if a particular team can win using max flow. You are given the total wins for each team so far, and the number of remaining games that each pair of teams are playing. Give a proof of correctness and analysis of running time. Your final result should not take much longer if all given numbers were multiplied by 1000.

Hint: Team E can finish with at most 24 wins. Thus, team E is still in the running iff there is a way for team A to get at most 2 additional wins, and B to get at most 3, and C at most 9, etc. Create a flow graph with a node for each remaining game, a node for each team, and interpret a win as a flow. Then, see if you can find a way to use fewer nodes.

Solution: For notation, let w_i be the total number of wins for each team i, g_i is the number of remaining games that team i is playing, $g_{i,j}$ if the number of remaining games that team i is playing with j. Without loss of generality, say we are interested in figuring out if team Z can win. When I say every team or every pair of teams below, I am excluding team Z.

Create: A source node s, $\binom{n}{2}$ nodes, one for each pair of teams, n nodes - one for each team i, and a sink node.

For each pair of teams, create a flow with capacity $g_{i,j}$ from s to node (i, j). This flow represents each of remaining games to be played between i and j. Add edges from (i, j) to nodes i and j with infinite capacity (or with $g_{i,j}$ capacity, it doesn't matter). A unit of flow from (i, j) to i represents a win for team i, and a unit of flow to j represents a win for team j.

As in the hint, we calulate the most number of wins $w_Z + g_Z$ that team Z can get if it wins all remaining games. For each other team i, $w_Z + g_Z - w_i$ represents the most number of remaining games that team i can win in order for Z to finish first in the season. Add edges with this capacity from i to t for each i.

Now any flow from s to t passes through some (i, j) (a game for i, j), then either i or j (depending on who won), and then to t. If the max flow in this graph is equal to the sum of the remaining games if all edges to all (i, j) are saturated, (all remaining games are played),

then that means that there was a way to assign winners to each of those games such that the total amount of wins that each team i got was under $w_Z + g_Z - w_i$ -¿ and therefore team Z will have a shot at finishing in first. If the min cut is smaller, then that means that there is no way to play every game subject to this restriction, and if we were to play every game, then some team would surpass $w_Z + g_Z$ total wins.

8. (bonus) You are the royal political mastermind for the nation of Computopia, and are trying to maximize your nation's wealth through diplomacy. Every other country $c \in \{1, \ldots, n\}$ that you sign an treaty with will either give you tribute or demand tribute, which will cause a net impact of $w_c \in \mathbb{Z}$ on your wealth. Due to their own political dealings, each country c also has a list $l_c \subseteq \{1, \ldots, n\}$ of other countries that they demand you also sign treaties with, if want to sign an agreement with c. These lists are not necessarily symmetric, that is if 1 demands that you sign a treaty with 3, it may not be true that 3 will demand that you sign a treaty with 1.

   Find an algorithm that can determine the optimal set of countries to sign treaties with to maximize your country's wealth. Give a proof of correctness.

   Hint: Think about this as a directed min-cut problem. Create a node for each country, and a source node and sink node. Add infinite capacity arcs to represent the prerequisites for each country. The countries on one side of the min cut will be the ones you sign with. Remember that the directed edges that count in an s-t min cut are only the ones that go from the side with s to the side with t.

   Solution: Create a source node s and sink node t. We will set up the flow/cut graph so that the final set of countries to sign treaties with are those on the same side as s.

   If a country a will demand k dollars from you, add an edge from a to t with capacity k. This way, choosing a in your set of countries (putting it on s's side) will cost you k in the min-cut, and not choosing a (putting it on t's side) will not cost you anything.

   Similarly, if a country a will give you k dollars, add an edge from a to s of capacity k. This way, not choosing a will add a penalty of k to the min cut, and choosing a will save you that amount. The net-gain for choosing a is still correct.

   If country a requires you to sign a treaty with country b, add a directed edge of infinite capacity from a to b. This way, choosing a without b would cause an infinite capacity edge to cross the cut, which will not happen in any min cut. Thus any finite min-cut will satisfy all the prerequisite treaty constraints. Since the cuts out of s and t are finite, the min-cut is also finite.

   Then, any finite cut in this graph will represent a consistent set of countries with which to sign treaties, and the min cut will be the one that nets you the most profit.