Note: Your TA probably will not cover all the problems. This is totally fine. The discussion worksheets are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 LP Basics

Linear Program. A *linear program* is an optimization problem that seeks the optimal assignment for a linear objective over linear constraints. Let $x \in \mathbb{R}^n$ be the set of variables and $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$. The canonical form of a linear program is

$$\begin{aligned} &\text{maximize } c^\top x \\ &\text{subject to } Ax \leq b \\ &\qquad\qquad x \geq 0 \end{aligned}$$

Any linear program can be written in canonical form.

Let's check this is the case:

(i) What if the objective is minimization?

(ii) What if you have a constraint $Ax \geq b$?

(iii) What about $Ax = b$?

(iv) What if the constraint is $x \leq 0$?

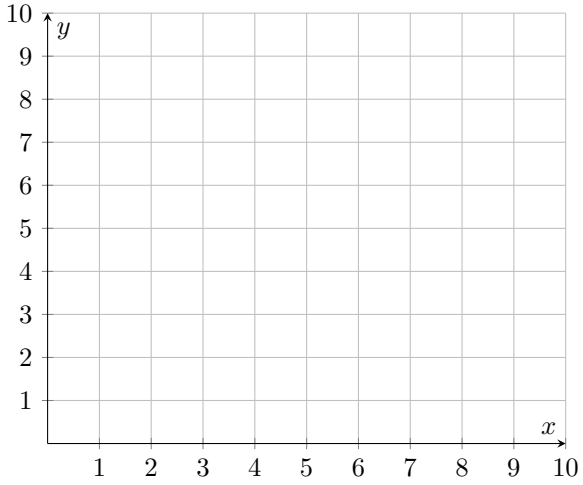
(v) What if the constraint is $x \geq -2$?

2 Simply Simplex

Consider the following linear program.

$$\max\ 2x + 3y$$
$$\text{subject to } \begin{cases} x + 3y \geq 6 \\ 2x - y \leq 12 \\ x + y \leq 9 \\ x \geq 0, y \geq 0 \end{cases}$$

(a) Sketch the feasible region below.



(b) Run the Simplex algorithm on this LP starting at $(6, 0)$. What are the vertices visited? **Please show your work.**

3 LP Meets Linear Regression

One of the most important problems in the field of *statistics* is the *linear regression problem.* Roughly speaking, this problem involves fitting a straight line to statistical data represented by points (x_1, y_1), $(x_2, y_2), \ldots, (x_n, y_n)$ on a graph. Denoting the line by $y = a + bx$, the objective is to choose the constants a and b to provide the "best" fit according to some criterion. The criterion usually used is the *method of least squares*, but there are other interesting criteria where linear programming can be used to solve for the optimal values of a and b.

Suppose instead we wish to minimize the sum of the absolute deviations of the data from the line:

$$\min \sum_{i=1}^{n} |y_i - (a + bx_i)|$$

Write a linear program with variables a, b to solve this problem.

Hint: Create new variables z_i and new constraints to help represent $|y_i - (a + bx_i)|$ in a linear program.

4 Job Assignment

There are I people available to work J jobs. The value of person i working 1 day at job j is a_{ij} for $i = 1, \ldots, I$ and $j = 1, \ldots, J$. Each job is completed after the sum of the time of all workers spend on it add up to be 1 day, though partial completion still has value (i.e. person i working c portion of a day on job j is worth $a_{ij}c$). The problem is to find an optimal assignment of jobs for each person for one day such that the total value created by everyone working is optimized. No additional value comes from working on a job after it has been completed.

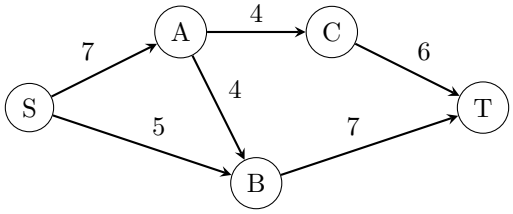
(a) What variables should we optimize over? In other words, in the canonical linear programming definition, what is x?

(b) What are the constraints we need to consider? Hint: there are three major types.

(c) What is the maximization function we are seeking?

5 Residual in graphs

Consider the following graph with edge capacities as shown:



(a) Consider pushing 4 units of flow through $S \to A \to C \to T$. Draw the residual graph after this push.

(b) Compute a maximum flow of the above graph. Find a minimum cut. Draw the residual graph of the maximum flow.

6 Max-Flow Min Cut Basics

For each of the following, state whether the statement is True or False. If true, provide a short proof. If false, give a counterexample.

(a) If all edge capacities are distinct, the max flow is unique.

(b) If all edge capacities are distinct, the min cut is unique.

(c) If all edge capacities are increased by an additive constant, the min cut remains unchanged.

(d) If all edge capacities are multiplied by a positive integer, the min cut remains unchanged.

(e) In any max flow, there is no directed cycle on which every edge carries positive flow.

(f) There exists a max flow such that there is no directed cycle on which every edge carries positive flow.

Simplex:

1. Start at an arbitrary vertex.
2. Denote the current vertex as x.
3. Look at all neighboring vertices y to x.
4. Find the neighbor y^* with the best objective value (if the LP is minimizing, y^* should have the smallest objective value, etc.).
5. If y^* has a better value than x, then we move to (i.e. visit) y^* and repeat steps 2 to 4. Otherwise, we have found the optimizer of the LP, and simply return x.

$\hookrightarrow$ Runtime: worst case exponential time, average case polynomial time.

LP Solver Runtime: Any LP can be solved in (worst case) polynomial time using the Ellipsoid or Interior Point Method (IPM). *Note: you do not need to know how the Ellipsoid method or IPM work, but you should know that they can be used to solve an LP in polynomial time.*

Flow. The *capacity* indicates how much flow can be allowed on an edge. Given a directed graph $G = (V, E)$ with edge capacites $c(u, v)$ (for all $(u, v) \in E$) and vertices $s, t \in V$, a flow is a mapping $f : E \to \mathbb{R}^+$ that satisfies

- **Capacity constraint:** $f(u, v) \leq c(u, v)$, the flow on an edge cannot exceed its capacity.
- **Conservation of flows:** $f^{\text{in}}(v) = f^{\text{out}}(v)$, flow in equals flow out for any $v \notin \{s, t\}$

Here, we define $f^{\text{in}}(v) = \sum_{u:(u,v)\in E} f(u, v)$ and $f^{\text{out}}(v) = \sum_{u:(v,u)\in E} f(u, v)$. We also define $f(v, u) = -f(u, v)$, and this is called *skew-symmetry*. Note that the total flow in the graph is $\sum_{v:(s,v)\in E} f(s, v) = \sum_{u:(u,t)\in E} f(u, t)$, where s is the source node and t is the target node.

Residual Graph. Given a flow network (G, s, t, c) and a flow f, the *residual capacity* (w.r.t. flow f) is denoted by $c_f(u, v) = c_{uv} - f_{uv}$. And the *residual network* $G_f = (V, E_f)$ where $E_f = \{(u, v) : c_f(u, v) > 0\}$.

Max Flow. Given a directed graph $G = (V, E)$ with edge capacities $c(u, v)$ for all $(u, v) \in E$ and vertices $s, t \in V$, the goal is to compute the maximum flow that can go from s to t. This can be solved with either Ford-Fulkerson (or its optimized variant Edmonds-Karp).

Ford-Fulkerson. Keep pushing along $s-t$ paths in the residual graph and update the residual graph accordingly. The runtime of this algorithm is $O(mF)$, where $m = |E|$ and F is the value of the max flow.

Edmonds-Karp. We can implement Ford-Fulkerson using BFS to find the $s - t$ paths. This yields a faster runtime of $O(nm^2)$, where $n = |V|, m = |E|$.

Min-Cut. Given a directed graph $G = (V, E)$ with edge weights $w(u, v)$ for all $(u, v) \in E$ and vertices $s, t \in V$, the goal is to compute the lightest $s - t$ cut (i.e. the cut separating t from s with the smallest sum of edge weights crossing it). Note that the Min Cut problem is the dual of the Max Flow problem.

