CCS Discrete III — Professor: Padraic Bartlett

Lecture 2: The Max-Flow Min-Cut Theorem

Weeks 3-4 — *UCSB 2015*

1 Flows

The concept of currents on a graph is one that we've used heavily over the past few weeks. Specifically, we took a concept from electrical engineering — the idea of viewing a graph as a circuit, with voltage and current functions defined on all of our vertices and edges — and used this concept to solve problems on random walks that (at first glance) looked completely unrelated!

As always, in mathematics, when something looks useful, what should our first instinct be? To **generalize** it! In specific, let's consider the concept of **current** from our circuits.

On a given graph G, the **current** was a function $i : V(G) \times V(G) \to \mathbb{R}$ such that

- $i_{xy} = -i_{yx}$; that is, the current flowing from x to y is just -1 times the current flowing from y to x. This made intuitive sense; if we have 3 units of something flowing from x to y, then if we were to look at things "backwards" we would have -3 units flowing from y to x.
- $i_{xy} = 0$ if $\{x, y\} \notin E(G)$. (Another way to phrase this restriction is to claim that i is a function with **domain** $E^+(G)$, the set of all edges in our graph along with all possible orientations on those edges, and to say that i is not defined on any pair $\{x, y\}$ with $\{x, y\} \notin E(G)$. Both ideas get to the same concept, which is that we don't want to work with current on edges that do not exist.)
- For any vertex x other than the source and ground vertices, we have Kirchoff's law:

$$\sum_{y \in N(x)} i_{xy} = 0.$$

  In other words, the "total current" flowing through x is 0. This makes sense; if we think of current as something that is "preserved," then we're just saying that the total amount of current flowing into x is the same amount that flows out — none gets left behind, and none magically is created!

These ideas — that flow measured in one direction is just the negative of flow measured in the other direction, and that at any other vertices other than two "special" ones labeled source and sink, the total flow through that vertex should be zero — are ones that capture and describe phenomena far beyond that of currents! Consider the following real-life instances of other sorts of "flows" that obey these rules:

1. Suppose that we have a house that we're installing the plumbing for. If we view our pipes as edges, and our joins between various pipes as vertices, we have a graph! Moreover, if we turn the water on, we encounter behaviors identical to those above:

- There is a special vertex, the source (the pipe connecting to the city water hookup,) where water flows from; there is also a (literal!) sink vertex connecting to the city sewage lines.
- On any pipe $\{x, y\}$, define w_{xy} to be the total net amount of flow of water out of x into y over a given unit of time. Then, if we measure the water w_{xy} flowing from its start point x to its end point y, we can immediately see that this is just $-w_{yx}$, the negative of the water flowing from y to x! That is, if we have $3cm^3/min$ flowing out of x to y in one direction in our pipe, then clearly the "net flow" from y to x is precisely $-3cm^3/min$, as we have $3cm^3/min$ flowing **into** y, i.e. $-3cm^3/min$ flowing **out** of y!
- As well, if we measure any join(i.e. vertex) in our plumbing network, we expect the total flow of water into that join to be the total flow of water out of that join (otherwise, that pipe is leaking / somehow magically drawing water out of the surrounding atmosphere / problematic, in either case!)

2. Conversely, suppose that we are modeling a highway system; we can think of our roads as edges, and our interchanges/etc where roads meet up as vertices. Again, this looks like a "flow:"

   - There are special vertices: the on-ramps and off-ramps to our highways, where cars can leave or exit.
   - On any stretch of road from interchange x to interchange y, we can let r_{xy} denote the total net amount of traffic per unit of time leaving from x to y. Again, it is reasonably clear that $r_{xy} = -r_{yx}$; if we have 30 net cars per minute leaving from south to north on a stretch of road, then if we're measuring from north to south we have 30 cars per minute **entering** our south interchange. In other words, our south interchange is **gaining** 30 cars per minute along the $\{x, y\}$ road; so its net flow **from** y **to** x is -30!
   - As well, if we measure any interchange (i.e. vertex) in our road network, we expect the total flow of cars into that interchange to be the total flow of cars out of that join (otherwise, we have a massive pileup of cars there or a rogue Honda manufacturing plant operating there, neither of which are things we want as highway planners.)

1.1 Flows: Definitions and Examples

There are dozens of other examples of this phenomena; try to think of some! Accordingly, this motivates us to make the following generalized definition for a **flow**:

Definition. A **real-valued flow** on a graph G with a pair of distinguished vertices s, t is a map $f : V(G) \times V(G) \to \mathbb{R}$ that satisfies the following three properties:

1. For any $\{v, w\} \notin E(G)$, we have $f(v, w) = 0$.
2. For any $\{v, w\} \in E(G)$ we have $f(v, w) = -f(w, v)$.

3. For any vertex $v \neq s, t$, we satisfy Kirchoff's law: namely, that

$$\sum_{w \in N(v)} f(v, w) = 0.$$

We call such graphs G with distinguished vertices s, t **networks**. We will often denote $f(v, w)$ as $f_{v,w}$ for shorthand.

Given this idea of **flow**, there are a number of natural extensions we can make. For example, we often want to study graphs where the edges are representing pipes with some fixed **capacity**: in other words, if we think of our water example you might want to find out what flows are **feasible** on a graph where all of the edges can carry at most one gallon per minute (or in the traffic situation, no more than 100 cars per hour; or in the electricity situation, where your wires are only rated to carry up to 1 ampere. This motivates the following definitions:

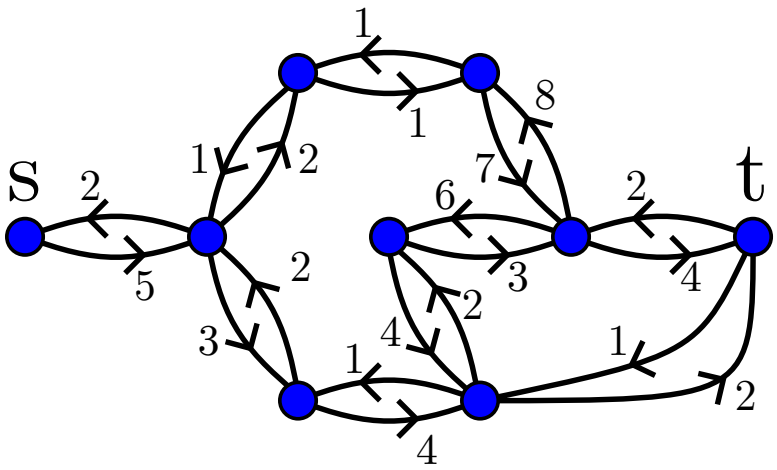
Definition. A **capacity function** on a graph G is a function $c : V(G) \times V(G) \to \mathbb{R}^+ \cup \{0, \infty\}$, that assigns nonnegative capacity values to every edge.

In this definition, we ask that if $\{x, y\} \notin V(G)$ that $c_{xy} = 0 = c_{yx}$, as that edge does not exist (and thus should have zero capacity.) Unlike with flows, we do **not** ask that $c_{xy} = c_{yx}$ or $-c_{yx}$, as there is no inherent reason to assume we have such symmetry; for instance, we could have a road with three lanes headed north and only one headed south (so it would have asymmetric traffic capacities) or a pipe with a one-way gasket fitted to it (so water could only travel in one direction; i.e. its capacity the other direction would be zero!)

Given a graph G and capacity function c, we say that a flow f on G is **feasible** if $f_{xy} \leq c(x, y)$, for every $x, y \in V(G)$. Again, as with flows, we will often denote $c(x, y)$ as c_{xy} for shorthand.

This motivates the following question: given a graph G with some given capacity function c, how "big" of a feasible flow can we define on G? In other words, if we're thinking of G as a collection of pipes with certain capacities, what's the most water we can pump out of the source vertex? If we think of G as a highway system, what is the maximum number of cars we can handle entering our system at a time? If G is a circuit, what's the biggest battery we can install without burning out our wires?

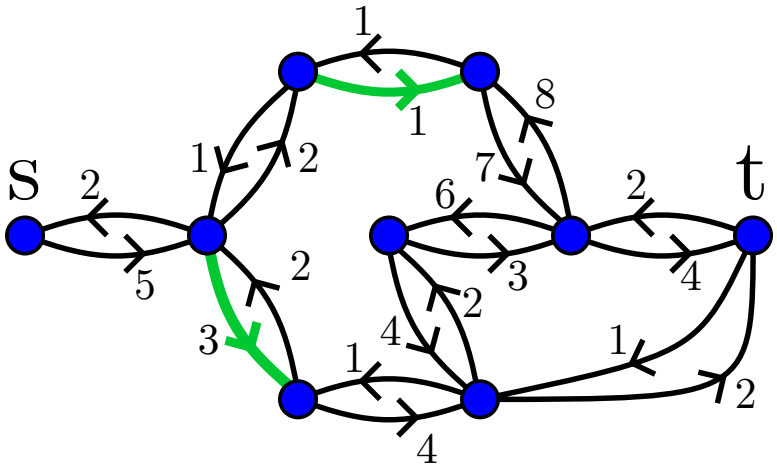
At first, answering this question seems difficult. Consider the graph below, where we've drawn in capacities on all of our edges:



A graph with capacities labeled on its edges. Because our capacity function is "oriented," we have doubled all of the edges in our graph and labeled each with the "oriented" capacity for ease of visualization.

One quick observation that we can make is that no flow on this graph can pump any more than ≤ 5 units of flow out of the source node; this is because the capacity of the only edge leaving the source is 5.

We can actually improve on this observation: in fact, we can observe that the largest any flow on this graph can be is 4! To see this, look at the two "green" edges below:



These two edges form a "bottleneck" for our flow, as all paths from the source s to the sink t must go through one of these two edges. However, the maximum capacity of these two edges is 4; therefore, it would seem impossible for any flow to pump more than 4 units out of the source!

We prove this claim, and in general set up the machinery to find upper bounds on the maximum sizes of flows, in our next section:

1.2 The Concept of Cuts

Definition. For a connected graph G with a flow f, define the **value** of f, $|f|$, as the total amount of current leaving the source node s : i.e. $\sum_{y \in N(s)} f_{sy}$.

More generally, given a subset U of $V(G)$, we say that the **total flow** through that subset U is just the sum

$$\sum_{\substack{x \in U, y \notin U, \\ \{x,y\} \in E(G)}} f_{xy}.$$

We denote this quantity as $f(U)$. In this notation, the value of f is just $f^(s)$; as well, we can reformulate Kirchoff's law as demanding that $f(v) = 0$ for every non-source/sink vertex v.

Notice that if U is any subset of $V(G)$ that does not contain either the source or the sink, then the total flow through U is zero! We quickly prove this here:

Theorem. If G is a graph with distinguished source vertex s and sink vertex t, f is a flow on G, and U is a subset of G's vertices, then the total flow through U is zero.

Proof. Take any such graph G, and let $U \subset V(G)$ be any subset of the vertices of G that does not contain either the source vertex s or the sink vertex t.

Take any vertex $x \in U$. Break the set $N(x)$ of x's neighbors into two disjoint sets:

- A_x, the neighbors of x contained in U.
- B_x, the neighbors of x not contained in U.

Consider the sum

$$\sum_{x \in U} \sum_{y \in N(x)} f_{xy}.$$

On one hand, by Kirchoff's law, we know that the internal sum is 0 for every $x \in U$; so this entire sum is just $\sum_{x \in U} 0 = 0$.

On the other hand, look at what happens if we break the internal sum above into two parts: Consider the sum

$$\sum_{x \in U} \sum_{y \in N(x)} f_{xy} = \sum_{x \in U} \left(\sum_{y \in A_u} f_{xy} + \sum_{y \in B_u} f_{xy} \right) = \left(\sum_{x \in U} \sum_{y \in A_u} f_{xy} \right) + \left(\sum_{x \in U} \sum_{y \in B_u} f_{xy} \right).$$

The second double-sum at right, by definition, is just the sum of f_{xy} over all edges where $x \in U, y \notin U$: in other words, this is just $f(U)$.

The first double-sum at right, by definition, is the sum of f_{xy} over all possible ways to pick two vertices $x, y \in U$ such that $\{x, y\}$ is an edge. Notice that as a result, each edge $\{x, y\}$ is counted **twice**: once by f_{xy} and another time by f_{yx}.

However, we know that $f_{xy} = -f_{yx}$; so every edge connecting two elements in U contributes 0 to this sum! In other words, the entire sum is just 0, as all of its terms cancel out; so we actually have that

$$\sum_{x \in U} \sum_{y \in N(x)} f_{xy} = \left(\sum_{x \in U} \sum_{y \in B_u} f_{xy} \right) = f(U).$$

But we proved earlier that the left-hand side was zero; therefore we have proven that $f(U) = 0$ as claimed. □

There is a simple corollary to this result that is very valuable:

Definition. For a connected graph G, a **cut** is a collection of vertices S such that $s \in S$ and $t \notin S$.

Corollary. If G is a graph with distinguished source vertex s / sink vertex t, S is a cut on G, and f is a flow on G, then $f(S) = f(s)$. That is, the flow through any cut is the flow through the source vertex.

Proof. Take S, and write $S = \{s\} \cup S'$, where S' is the set $S \setminus \{s\}$. By our result above, we know that $f(S') = 0$.

We claim that $f(S) = f(s) + f(S')$; combining this result with $f(S') = 0$ will prove our corollary. To see this, simply examine the sum

$$\sum_{x \in S} \sum_{y \in N(x)} f_{xy}.$$

As shown in our earlier proof, this object is just $f(S)$, as all of the f_{xy}'s corresponding to "internal" edges in S (i.e. edges with both endpoints in S) get canceled out by f_{yx} terms that also show up in this sum.

However, by considering the two cases where either $x = s$ or $x \neq s$, we can break this sum up into the two sums

$$\left(\sum_{y \in N(s)} f_{sy} \right) + \left(\sum_{x \in S'} \sum_{y \in N(x)} f_{xy} \right).$$

The one at left is just $f(s)$ by definition, and the one at right is just $f(S')$ by (again) our earlier proof. So we have proven that $f(S) = f(s) + f(S') = f(s) + 0 = f(s)$, as claimed. □

The reason we care about flow values and cuts is that they let us make our "bottleneck" idea from before rigorous. Consider the following definition and corollary of our earlier results:

Definition. The **capacity** of any cut S, $c(S)$, is the sum

$$\sum_{x \in S, y \notin S} c_{xy}.$$

Corollary 1. *Suppose that G is a graph with source vertex x and sink vertex t, a capacity function c, and a feasible flow f (that is, a flow bounded above by this capacity function.)*

Then $f(s) \leq c(S)$ for any cut S.

Proof. We proved earlier that $f(s) = f(S)$ for any cut S. As well, by definition we know that

$$f(S) = \sum_{x \in S, y \notin S} f_{xy} \leq \sum_{x \in S, y \notin S} c_{xy} = c(S).$$

So we have proven our claim! □

Surprisingly, the above simple bound — that the value of any flow is bounded above by the capacity of any cut — turns out to be the best possible! We prove this in the next section, via the (famous!) Ford-Fulkerson max-flow min-cut theorem!

1.3 The Max-Flow Min-Cut Theorem

Theorem 2. *(Ford-Fulkerson:) Suppose that G is a graph with source and sink nodes s, t, and a rational- and finite-valued capacity function c. Then the maximum value of any feasible flow f on G is equal to the minimum value of any cut on G.*

Proof. We prove our claim here via the Ford-Fulkerson algorithm, defined as follows:

1. Input: any currently feasible flow f. Let R be the set currently given by the single source vertex $\{s\}$, and S be the subset of R currently defined as $\emptyset$.

2. Iteration: Pick any vertex $x \in R$ that's not in S. For every vertex $y \notin R$ connected to x with $f_{xy} < c_{xy}$, add w to R. (We think of these as the edges in our graph that "could have more flow on them.")

   Once you've done this search process over all of v's neighbors, add x to the set S (the "searched" set.) Repeat this step until $R = S$.

3. If the sink vertex t is not in R at this time, terminate.

4. Otherwise, the sink vertex is in R. Moreover, there is a path

   $$P = (x_0, \{x_0, x_1\}, x_1, \{x_1, x_2\}, x_2 \ldots, \{x_{n-1}, x_n\}, t)$$

   from $x_0 = s$ to $x_n = t$, along edges where we have $f_{x_i, x_{i+1}} < c_{x_i, x_{i+1}}$ for every i.

   Call such a path an **augmenting path** for f, and let $\epsilon = \min_{0 \leq i \leq n} (c_{x_i, x_{i+1}} - f_{x_i, x_{i+1}})$.

   Now, (as the name suggests,) **augment** f along this path! That is, define a new flow f' as follows:

   - $f'_{x_i, x_{i+1}} = f_{x_i, x_{i+1}} + \epsilon$, for any edge $\{x_i, x_{i+1}\}$ involved in our path P that we traveled from x_i to x_{i+1}. (For consistency's sake, also update the "backwards" flow f_{x_{i+1}, x_i} to $-(f_{x_i, x_{i+1}} + \epsilon)$, to preserve the antisymmetry property that makes this a flow.)
   - $f_{yz} = f_{yz}$, for any edge $\{y, z\}$ not in our path.
   - $f_{yz} = 0$ for any $y, z \in V(G)$ not connected by an edge.

Notice that by definition, for any vertex x_i in our path other than s, t, the net flow through x_i is still 0; we added ϵ to the flow entering x_i along the edge $\{x_{i-1}, x_i\}$, and canceled this out by adding ϵ to the flow leaving x_i along the edge $\{x_i, x_{i+1}\}$. As well, for any vertex y not in our path, we have changed nothing about the edges entering or leaving that vertex; so the flow through y is unchanged and thus still 0. So f' is indeed still a flow, as we still have our antisymmetry/Kirchoff's law properties.

Also notice that by definition, $f_{x_i,x_{i+1}} + \epsilon \leq f_{x_i,x_{i+1}} + (c_{x_i,x_{i+1}} - f_{x_i,x_{i+1}}) = c_{x_i,x_{i+1}}$; therefore our flow f' is still feasible!

Finally, note that f' has value ϵ-greater than our old flow, as we increased the flow out of s by ϵ.

Go to (1) with our new flow f', and repeat this process.

We know that in each run of this process, ϵ is at least $1/a$, where a is the least-common denominator of the capacities of the edges. So each run of this process increases the size of our flow by a fixed constant. As well, we know that our capacity function only has finite values, and therefore that the capacity of the cut $\{s\}$ is finite.

Therefore, because the value of any flow is bounded above by the size of any cut, and we are increasing by a fixed constant at each step, our process above must terminate after finitely many steps!

We can use this to prove the max-flow min-cut theorem as follows. Start by inputting the feasible flow f_0 that's identically 0 on all edges. Our process will then return two things for us: a flow f, and a set S generated on the last run of our process that caused the process to terminate!

In other words, S is a set with the following two properties:

1. S contains s and not t, and thus is a cut.
2. S is specifically the collection of vertices v in $V(G)$ for which there is a path P from s to v, on which $f(e) < c(e)$ for every edge in that path.

This means that for any pair $x \in S, y \notin S$, we must have $f(x, y) = c(x, y)$, as otherwise, we would have added y to R and thus eventually to S!

But this means that

$$f(S) = \sum_{x \in S, y \notin S} f_{xy} = \sum_{x \in S, y \notin S} c_{xy} = c(S)!$$

In other words, we have proven that there is a cut S such that the value of our flow f is equal to the capacity of our cut.

We know that the capacity of any cut is at least the value of any flow. Therefore, in particular, this means that there is no smaller cut S' (as if there was, we would have $c(S') < c(S) = f(S) \Rightarrow c(S') < f(S)$, a contradiction.) As well, we know that there can be no greater flow than f, by the same logic! So, in fact, we know that S is the cut with the smallest possible capacity on our graph, and that f is the flow with the largest possible value on our graph.

In other words, we've proven our claim: that the capacity of the smallest cut is the value of the largest flow. □

Two important things to recognize in the above theorem are the following:

1. If our capacity function c is finite and integer-valued, then the maximum flow we create is integer-valued! This is not hard to see: if our capacity function is integer-valued, then ϵ will be integer-valued, and thus every f' starting from the all-zeroes flow on up will be integer-valued.

2. This theorem very, very strongly needed our capacities to be rational-valued, as otherwise our algorithm would not terminate! In fact, you can create irrational capacity functions for which the above algorithm will neither terminate nor even converge on a flow that's maximal (see the homework for an example!)

   However, with that said, the **theorem** is still true. In other words, the following result is how people usually refer to the max-flow min-cut theorem:

   Theorem 3. *(Ford-Fulkerson:) Suppose that G is a graph with source and sink nodes s, t and a capacity function c. Then the maximum value of any feasible flow f on G is equal to the minimum value of any cut on G.*

   Proving this stronger version of the theorem is a task for the homework!

2 Using Max-Flow Min-Cut to Prove Theorems

Given the nature of the max-flow min-cut theorem, it is perhaps not surprising that it has a lot of practical applications and uses; electricity, water and traffic (amongst other phenomena) are pretty practical objects to want to optimize! On the homework, and later in these notes, we will look at some of these applications.

In this section, however, I want to focus on something fairly surprising; the max-flow min-cut theorem is remarkably useful in proving other theorems in combinatorics! In this section, we prove several famous theorems in mathematics with little to no effort using the max-flow min-cut theorem.

2.1 Hall's Marriage Theorem

We start by proving Hall's Marriage Theorem, a result you may remember from winter quarter's homework sets:

Theorem 4. *Suppose that G is a bipartite graph (V_1, V_2, E), with $|V_1| = |V_2| = n$ for some integer n. Then G has a perfect matching[1] if and only if the following condition holds:*

$$\forall S \subseteq V_1, |S| \leq |N(S)|.$$

Proof. We first notice that the condition above is necessary for a perfect matching to exist; indeed, if we had a subset $S \subseteq V_1$ with $|S| > |N(S)|$, then there is no way to match up all

[1] In case you forgot: a **perfect matching** in a bipartite graph G is a collection of disjoint edges M that contains all of the vertices in G. In a sense, it is a way to "match" all of the vertices in V_1 to the vertices in V_2.

of the $|S|$ elements of S along edges with elements in V_2 without using some elements more than once.

We now prove that our condition is sufficient, via the max-flow min-cut theorem. Take our graph G, and add in a source vertex s and sink vertex t. Connect s to every vertex of V_1 by adding in n edges, and similarly connect t to every vertex of V_2 by adding in n more edges.

Let c be a capacity function that's defined as follows:

- $c_{s,v_1} = 1$ for any $v_1 \in V_1$.
- $c_{v_2,t} = 1$ for any $v_2 \in V_2$.
- $c_{v_1,v_2} = \infty$ for any $v_1 \in V_1, v_2 \in V_2$ such that $\{v_1, v_2\} \in E(G)$.
- $c_{v_2,v_1} = 0$ for any $v_1 \in V_1, v_2 \in V_2$ such that $\{v_1, v_2\} \in E(G)$. (In other words, we disallow any flow from going from V_2 back to V_1.)
- $c_{x,y} = 0$ for any $\{x, y\} \notin E(G)$.

By Ford-Fulkerson, there is a maximal flow f and minimal cut S on this graph, with $c(S) = f(s)$. We know that f is integer-valued, because c was integer valued.

Moreover, we know that f only adopts the values 0 or 1 on any edge! This is not hard to see: take any $v_1 \in V_1$, and consider the net flow entering v_1. It is at most 1: this is because the capacity of the edge $\{s, v_1\}$ is at most 1, there are no edges from other V_1-vertices to v_1 because our graph is bipartite, and the capacity of any edge from V_2 to our v_1 is zero by construction. Therefore, because our flow is integer-valued, the net flow entering v_1 is either 0 or 1, and thus the net flow leaving v_1 is either 0 or 1. Consequently, the flow along any edge leaving v_1 is either 0 or 1, as the sum of the flow along all leaving edges is either 0 or 1!

A similar argument proves (check it!) that the flow along any edges through v_2 must also be 0 or 1, and completes our proof of this claim.

Let M be the collection of all edges $\{v_1, v_2\}$ on which $f_{v_1,v_2} = 1$. We can observe that this is a **matching**: to see why, notice that because the net flow out of any vertex $v_1 \in V_1$ is 1, for any $v_1 \in V_1$, there is at most one $\{v_1, v_2\} \in M$ with $f_{v_1,v_2} = 1$. Similarly, because the net flow into any $v_2 \in V_2$ is 1, for any $v_2 \in V_2$, there is at most one $\{v_1, v_2\} \in M$ with $f_{v_1,v_2} = 1$. In other words, M is a matching!

We claim that M is a perfect matching; that is, that $|M| = n$, the size of $|V_1|$. To do this, note that $|M| = f(s)$, as we have one edge in M for each edge with flow 1 in our flow! We also know that $f(s) = c(S)$, where S was our minimal cut.

Therefore, if we can prove that the capacity of any cut in our graph is at least n, then we are done, as this will prove that $|M| = f(s) = c(S) \geq n$!

Let $X = S \cap V_1$, and $Y = S \cap V_2$. We know that the capacity of any edge originally in G is infinite; consequently, if $x \in X$ has a neighbor $y \in V_2$ such that $y \notin Y$, the capacity of this cut would be infinite!

But we know that there is a finite cut: in specific, the cut $\{s\}$ has capacity n, as s is connected to n other vertices by edges of capacity 1. Therefore, if S is a minimal cut, then if $x \in X$ has $y \in N(x)$, we must have $y \in Y$. In other words, $N(X) \subseteq Y$, and thus no edge in $E(G)$ has $v_1 \in S, v_2 \notin S$.

But if we look at the only possibilities that are left, this means that if $\{u, v\}$ is an edge in $E(G)$ with $u \in S, v \notin S$, our edge must either be

- $\{s, v_1\}$, for some $v_1 \in V_1 \setminus X$, or
- $\{y, t\}$, for some $y \in Y$!

Therefore, if we look at $c(S)$, we can see that

$$\begin{aligned} c(S) &= \sum_{\substack{v \in S, w \notin S, \\ \{v,w\} \in E(G)}} c_{v,w} \\ &= \sum_{v \in V_1 \setminus X} c(s, v) + \sum_{y \in Y)} c(y, t) \\ &= \sum_{v \in V_1 \setminus X} 1 + \sum_{y \in Y)} 1 \\ &= |V_1 \setminus X| + |Y| \\ &\leq (n - |X|) + |N(X)| \\ &\leq n - |X| + |X| \qquad\qquad = n. \end{aligned}$$

So any cut has capacity at least n, and we've proven our claim! □

2.2 Menger's Theorem

We now continue with a classical theorem of Menger, for which we need the following definition:

Definition. Let G be a graph. We say that a set K of edges **separates** two vertices s, t if and only if there are no paths from s to t that do not use elements in K.

Theorem 5. *Take any graph G on finitely many edges/vertices. For any two vertices $s, t \in V(G)$, the minimal size of any set K that separates s, t is equal to the maximal number of edge-disjoint paths that start at s and end at t.*

Proof. Take G. We first notice that the size of any separating set is at least as large as the number of edge-disjoint paths starting at s and ending at t; this is because if K separates s from t, it must contain at least one edge out of every edge-disjoint path (otherwise, we could simply use that path to travel from s to t!)

As with the max-flow min-cut theorem[2], to show that the maximum number of edge-disjoint paths is equal to the minimal size of any separating set, it suffices to find a collection P of paths and separating set K such that $|P| = |K|$ to prove our claim, as any such P must be maximal / K must be minimal.

[2]To repeat our logic from that proof; this is because if we **could** find any such sets P, K, then (because we proved above that for any collection P' of paths / any separating set K, that $|P'| \leq |K|$) for any P', K' we would have $|P'| \leq |K| = |P| \leq |K'|$, and therefore that $|P'| \leq |P|, |K| \leq |K'|$ for any P', K'. That is, P is maximal and K is minimal, as desired.

Let s, t be the source and sink nodes of G, respectively, and let c be a capacity function on G that has $c_{x,y} = 1$ for every edge in $E(G)$. By using the max-flow min-cut theorem, find a integer-valued maximal flow f on G, and a corresponding minimal cut S.

I claim that f corresponds to $f(s)$-many edge-disjoint paths in our graph from s to t. To see this, consider the following algorithm for finding a path from s to t given that $f(s) \geq 1$:

1. Because f is integer-valued, $f(s) \geq 1$, and the capacity of every edge in G is 1, there is some edge leaving s to some vertex v_1. Make this the first edge in our path.

2. Assume that we have gotten to the vertex v_i in our process. If $v_i = t$, then we are done: we have created a path from s to t! Otherwise, $v_i \neq t$, and therefore is a vertex at which Kirchoff's law holds. We got "to" v_i via an edge $\{v_{i-1}, v_i\}$ carrying a unit of flow into v_i; therefore, because our flow is integral and the capacity of every edge in our graph is 1, there must be some edge **leaving** v_i with a unit of flow! Let $\{v_i, v_{i+1}\}$ be this edge, and let us travel to v_{i+1}.

3. Repeat step 2 until we halt (this is guaranteed to occur because G is finite;) this generates the promised path from s to t along flow-edges!

If $f(s) \geq 1$, this process certainly finds us one path from s to t. Given such a path, consider the new flow f' formed by decreasing our flow to 0 along each edge in our path! This remains a flow, because we still satisfy Kirchoff's law at every non-source/sink vertex (as we took 1 away from the flow entering and balanced this by taking 1 away from the flow leaving as well.)

Therefore, we can run our algorithm **again** on this new flow to find another path! Notice that this path cannot use any of the edges in our first path, as the flow along any edge used in an earlier path is 0; so in fact these two paths are edge-disjoint. In fact, repeating this process $f(s)$ times will generate our desired $f(s)$ edge-disjoint paths, as desired.

Now, I claim that our cut S corresponds to a separating set K of edges of size $c(S)$. Proving this claim will demonstrate that the number of edge-disjoint paths created above, $f(s)$, is equal to the size of our separating set $c(S)$, and thus prove our claim.

To do this, define the set K as follows:

$$K = \big\{\{x, y\} \mid x \in S, y \notin S, \{x, y\} \in E(G)\big\}.$$

On one hand, this is clearly a set of size $c(S)$, as by definition $c(S) = \sum_{\substack{x \in S, y \notin S, \\ \{x,y\} \in E(G)}} c_{xy}$ for our graph, and the capacity of each edge is 1.

On the other hand, S is a set that contains s and not t. Therefore, **any** path from s to t will at some point in time need to have at least one edge of the form $\{x, y\}$ with $x \in S, y \notin S$, which is by definition in our set above. Therefore, K is a separating set, as no path can go from s to t without passing through an element of K! This finishes our proof, as we have found a separating set with size equal to that of a collection of edge-disjoint paths. $\square$

2.3 Dilworth's Theorem

We now turn to Dilworth's theorem as a third example of how flows can solve combinatorial theorems!

To state this theorem, we need a few definitions first:

Definition. Given a set P, a **relation** on P is any function from $P \times P \to \{T, F\}$. You have seen many different kinds of relations before in this class, most notably **equivalence relations**.

A **partially ordered set** $P = (X, <)$ is a set X with a relation $<$ on P that satisfies the following two properties:

- **Antisymmetry**: For all $x, y \in X$, if $x < y$, we do not have $y < x$.
- **Transitivity**: For all $x, y \in X$, if $x < y$ and $y < z$, then we have $x < z$.

Notice that we do **not** inherently know that any two elements are related: if we had the third property that for any $x \neq y$ in X¡ we have $x < y$ or $y < x$, we would have something called a **total** ordering.

For an example of a poset, consider the set P of all breakfast foods, with the relation $>$ defined by "is tastier than." For instance, we definitely have

$$\text{(delicious perfect pancakes)} > \text{(horribly burnt pancakes)},$$

so these two objects are comparable. However, some other objects are **not comparable**: i.e.

$$\text{(delicious perfect pancakes)} \text{ , } \text{(delicious perfect french toast)}$$

are two different objects such that neither are really obviously "tastier" than the other. Most things in life that we put orderings on are usually posets, as there are almost always "edge cases" or other things that we can't compare/think are as good as each other /etc.

Definition. In a poset $P = \langle X, < \rangle$, a **chain** is a set $C \subset X$ such that any two elements of C are comparable; that is, if $x \neq y$ are both in C, then either $x < y$ or $y < x$.

Similarly, an **antichain** is a set $S \subset X$ such that no two elements in S are comparable; that is, if x, y are both in C, then neither $x < y$ or $y < x$ can hold.

We state Dilworth's theorem here:

Theorem 6. *(Dilworth.) Take any partially ordered finite set $P = \langle X, < \rangle$. A **decomposition** of P into m chains is any way to find m chains $C_1, \ldots C_m$ such that the union of those m chains is P itself.*

Let m be the size of the smallest number of chains needed to decompose P. Then m is the size of the largest antichain in P.

As before, we can easily observe that the size of any antichain is at most the size of any decomposition; because no elements in an antichain are comparable, any two elements in an antichain must belong to different chains. So, as always, it suffices to create an antichain with size equal to a given decomposition of our poset into chains.

Unlike our earlier theorems, this one does not seem to be quite as amenable to a flow-based attack. In particular, our language of "maximal flows" seems to be good at giving us **edge-disjoint** paths: however, in Dilworth's theorem, we actually want **vertex**-disjoint paths, because we're trying to decompose P into chains (which are made out of vertices, not the inequalities linking them.)

How can we do this? Well: what we **really** want to do is have a flow that, instead of having some sort of maximum upper bound, has a **minimal** lower bound: i.e. that has to send at least one unit of flow into every vertex! Our desire to do this motivates the following two extensions of the max-flow min-cut theorem:

Theorem. Suppose that G is a directed graph with source and sink nodes s, t. Suppose further that G comes with a capacity function $c : V(G) \times V(G) \to \mathbb{R} \cup \{\infty\}$.

Notice that c can adopt negative values here, which is different from before[3]. As before, we say that a flow f is **feasible** given these two capacity functions if $f_{xy} \leq c_{xy}$ for any pair x, y of vertices in $V(G)$.

Suppose that there is **any** feasible flow f_0 on our graph[4] Then, from this feasible flow f_0, we can create another feasible flow f and cut S such that the following holds:

- For any x, y with one of x, y in S and the other not in S, we have $f_{xy} = c_{xy}$.

As before, this flow 's value is equal to the capacity of our cut, and thus proves that there is a maximal flow with value equal to that of the minimal cut.

Proof. HW! □

To do "minimization" problems with this result, we can simply create a problem where our capacity function takes on negative values: the "maximal" flow, then, is the flow that attains the smallest negative values possible (and thus if we scale things by -1, is the flow that attains the smallest positive values possible!) Then, if we set up a network where our flow is forced to be negative, "maximizing" this flow will minimize the number of paths we create, which in theory will give us the optimal number of paths!

We use these ideas here:

Proof. Let $P = \{X, <\}$ be a partially ordered finite set. We turn P into a graph G, as follows:

1. For each element $x \in X$, create a pair of vertices x_1, x_2 in our graph.

[3]One natural interpretation of a "negative" capacity is that this is a way to enforce a mandatory **minimum** flow! That is: suppose that we have any edge $\{x, y\}$ such that $c_{yx} = -3$. What does this mean? Well: it means that the total flow from y to x must be at most -3; in other words, that the total flow from x to y is at **least** 3! In particular, if we have any edge $\{x, y\}$ that we want to enforce a minimum flow of l and a maximum flow of u from x to y, we can set $c_{xy} = u, c_{yx} = l$.

Again, this is a natural thing to want to consider; in plumbing, for example, you often want to insure a minimal amount of flow through your pipes to stop them from bursting when it drops below freezing in the winter. (This motivation works less well at Santa Barbara than it does where I learned it in Chicago.)

[4]Unlike before, it is no longer obvious that every graph has a feasible flow: indeed, it is easy to create a graph that has no feasible flow! Have a mandatory minimum flow from our source to some vertex v be something huge, and then have the maximum flow out of v be tiny; this will immediately violate Kirchoff's laws!

2. Create an edge $\{x_1, x_2\}$ between each such pair of vertices.

3. Whenever $x, y \in X$ are two elements of P such that $x < y$, add the edge $\{x_2, y_1\}$ to our graph.

4. Let $A = \{a \in X : \nexists\ y \text{ s.t. } a < y\}$, and $B = \{b \in X : \nexists\ y \text{ s.t. } y < b\}$. Create two new vertices s, t, and add the edges $\{s, a_2\}$ and $\{b_1, t\}$ for all of the elements in A, B respectively.

5. Define a capacity function c on G's edges as follows:

   - Set $c_{x_2,x_1} = -1, c_{x_1,x_2} = \infty$ for each element x of X. This in a sense mandates that **at least one** unit of flow must pass from x_1 to x_2 for each such pair, and that there is no upper limit on this flow in this direction.
   - For any two elements $x, y \in X$ with $x < y$, set $c_{y_1,x_2} = 0, c_{x_2,y_1} = \infty$. This mandates that all of our flow must travel **from** x to y, and none flows "backwards" from y to x. Notice that this flow is going "backwards" from how we might expect; we're forcing our flow to go "upwards" to the source, instead of the normal direction of "down" to the sink!
   - As well, for any $x \in B$, set $c_{t,x_1} = \infty, c_{x_1,t} = 0$; this says, strangely enough, that we only let things flow "out" of the sink.
   - Finally, for any $x \in A$, set $c_{x_2,s} = \infty, c_{s,x_2} = 0$; this says (again, this is strange) that we only let things flow "into" the sink.

We generate a starting feasible flow f_0 on our network G as follows: initially, start with an all-zero flow. Now, take any element $x \in X$, and construct a path from s to t through the edge $\{x_1, x_2\}$ as follows:

1. Let C be the largest chain in P that contains x. Let l be the smallest element in C; we can see that $l \in A$, as otherwise there would be some element that is both comparable to and smaller than l (in which case we could have added it to our chain, contradicting our maximality.) As well, if u is the largest element in our chain, then $u \in B$ by similar reasoning.

2. Travel from t to s along this chain in order: i.e. if we write the elements of our chain as $(c^1, c^2 \dots c^n)$ in order, take the path

$$t \to c_1^1 \to c_2^1 \to c_1^2 \to c_2^2 \to c_1^2 \to c_2^3 \to \dots \to c_1^n \to c_2^n \to s$$

Add -1 to our flow's values along every edge in this path in the $s \to t$ direction, or 1 to every edge in this path in the $t \to s$ direction. Do this for every $x \in X$; the result of this process is a feasible flow, as we have at least one negative unit of flow from $x_1 \to x_2$ for any $x \in X$ — that is, at least one unit of flow from x_2 to x_1 — and all of our flow is going "backwards" from t to s, as our capacity functions desire.

Therefore, we can apply the modified max-flow min-cut theorem here to get a maximal integer-valued flow f and minimal cut S with respect to our capacity function.

I claim that our maximal flow corresponds to $f(s)$-many chains in our poset (not necessarily disjoint.) To see this, simply reverse-engineer the algorithm used to turn chains to paths above. That is, given a maximal flow f with $f(s) \leq -1$, do the following:

- Start from t. There must be at least one unit of flow from t to some x_1 with $x \in A$; travel to this x_1. By Kirchoff, this unit of flow must travel to x_2; go there.
- From x_2, our one unit of flow must leave somewhere else; in specific, it must go either to s or to some y_1 with $x < y$, as these are the only edges we have created that have positive capacity. Travel to that y_1, and then to y_2.
- Repeat this step until we get to s.

This generates a path from t to s along edges where the flow is at least 1. This path corresponds to a chain in our poset, by starting with the first element in A we began at and ending at the last element in B before we went to t.

If we increase our flow by 1 on each of these edges, we get a new flow with value $f(s)+1$, and have created a single chain. Do this $|f(s)|$ times, to get $|f(s)|$ chains, and a flow whose value is 0; i.e. a flow whose value on every edge is 0, as no cycles can exist on our graph because of our capacity function constraints!

This means that our $f(s)$ paths must have gone through each $\{x_1, x_2\}$ edge in our graph; in other words, that our $f(s)$ chains contain any $x \in X$, and therefore that we can write P as the union of $f(s)$-many chains.

We now turn our attention to our minimal cut. We first note that a finite cut exists: consider the set $S = \{s\} \cup \{x_2 \mid x \in A\}$. The only edges that have one endpoint in S and another not in S are those of the form $\{x_1, x_2\}$ for $x \in A$; therefore, this has capacity $-|A|$, which is finite as P itself is finite.

Therefore, if S is any minimal cut, we know that the following must hold:

- For any $x \in X$, if $x_1 \in S$, then x_2 is also in S, as the capacity $c_{x_1,x_2} = \infty$.
- Similarly, if x, y are a pair of elements of X with $x < y$, and $x_2 \in S$, then y_1 is also in S, as the capacity c_{x_2,y_1} is infinite.
- Suppose that $x_1 \in S$ and $x \in X$ is the element corresponding to x_1. Suppose that there is no $y_2 \in S$ with corresponding $y \in X$ such that $y < x$. Then deleting x_1 from our cut decreases the capacity of our cut, as we have replaced the sum $\sum_{y_2 \in G} c_{x_1,y_2} = 0$ with $c_{x_2,x_1} = -1$. So all minimal cuts have this property!
- Using similar logic, suppose that $x_2 \in A$ and $x_2 \notin S$. Then adding x_2 to our cut strictly decreases its capacity, as (because $x_1 \notin S$ in such a situation by our first bullet point) we'd replace $c_{s,x_2} = 0$ with $c_{x_2,x_1} = -1$. So any minimal cut must contain each $x_2 \in A$.

In particular, this implies the following observation:

Observation. Suppose that S is a minimal cut on our graph. Then the only "crossing" edges for S — that is, the only edges $\{u, v\}$ such that $u \in S, v \notin S$ holds — are those of the form $\{x_1, x_2\}$, for some $x \in X$. Moreover, if $\{x_1, x_2\}$ and $\{y_1, y_2\}$ are a pair of crossing edges corresponding to a pair of elements $x, y \in X$, then we cannot have $x < y$ or $y < x$.

In other words, the collection of cut-edges of our minimal cut corresponds to an antichain in P!

Moreover, **any** antichain corresponds to a cut: take all of the x_2's corresponding to elements in that antichain, and everything above them in the poset. This is a cut, with value equal to -1 times the number of elements in our antichain. Therefore, a minimal cut corresponds to the largest possible antichain, as the values of all of these cuts are negative!

This finishes our proof: we have shown that for any poset P, the size of the largest antichain is the size of the smallest decomposition of P into chains. Success!

□

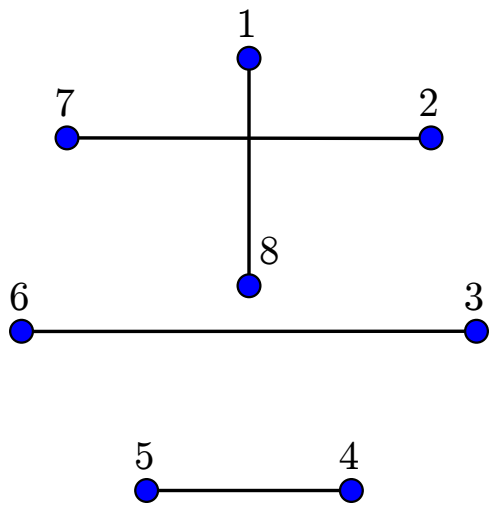
2.4 Baranyai's Theorem on Tournaments

We close here by studying a famous result of Baranyai on tournaments! To understand this result, start with the following question:

Question 7. *Suppose that we've decided to create a Rock-Scissors*[5] *tournament for eight people. Specifically, in each round, we want to divide people up into pairs and have each pair play each other. How many rounds do we need in order to have each player face each other player exactly once?*

Solution. Trivially, we will need at least seven rounds; this is because each team is playing exactly one other team in each round and there are eight total teams.

As it turns out, this bound is achievable! To see why, consider the following picture:



This is a schedule for the first round; to acquire schedules for later rounds, simply take this picture and rotate it by appropriate multiples of $2\pi/7$.

This observation can in fact be generalized: if we want to create a tournament for $2n$ teams that requires $2n-1$ rounds for everyone to play each other, we can do this with the following:

- Let G be an abelian group of order $2n-1$, say $\mathbb{Z}/(2n-1)\mathbb{Z}$.
- For each $g \in G$, let $M_g = \{\{g, \infty\} \cup \{a, b\} : a + b = 2g, a \neq b\}$.

[5] Rock-Scissors is like Rock-Paper-Scissors, but without paper.

- Identify our teams with the set $G \cup \{\infty\}$, and each of our $2n-1$ rounds with the $2n-1$ distinct M_g's.

The picture above can be thought of as the above process ran on $G = \mathbb{Z}/7\mathbb{Z}$, where the ∞ vertex is the 8 vertex. (If you are unpersuaded, try tilting your head by 90°.)

I claim this yields the desired tournament (and leave the proof for the homework!)

This is all well and good for 2-player games. But what if we want to consider multi-player games, like (say) Super Smash Bros. Melee? Or n-player chess[6]?

In other words: suppose we have a game that requires k teams to play, and we want to make a tournament with n teams. How many rounds do we need to insure that every possible combination of players encounter each other?

For the moment, let's assume that k divides n for convenience's sake. In the best case, then, we'll need to have $\binom{n}{k} \cdot \frac{k}{n} = \binom{n-1}{k-1}$ many rounds; we have $\binom{n}{k}$ many possible k-subsets of our players that we want to have encounter each other, and can fit n/k many such subsets into each round.

Can we achieve this "optimal" efficiency for every such k, n where $k|n$? Or is it possible that some tournaments will need to be less efficient than this? We asked this question above for $k = 2$, and (if you do the homework problem mentioned above) saw that the answer is yes!

For $k = 3, 4$, however, this problem is **far** harder; moreover, for any value of $k \geq 5$ there are no constructive[7] proofs known like the above. Consequently, the following result of Baranyai was **really** surprising when it first came out — with far less effort than any case other than $k = 1, 2$, we can settle all of these problems at once!

Theorem. If k divides n, then the set of all k-element subsets of the set $\{1, \dots n\}$ can be divided into $\binom{n-1}{k-1}$ many disjoint k-parallel classes.[8]

Proof. Let n, k be given, let $m = n/k$, and $M = \binom{n-1}{k-1}$. We will prove a stronger version of our claim: for any integer $1 \leq l \leq n$, there is a collection $A_1, \dots A_M$ of m-partitions[9] of $\{1, \dots l\}$, with the property that each subset $S \subseteq \{1, \dots l\}$ occurs $\binom{n-l}{k-|S|}$ many times in these partitions A_i.

At first glance, this ... does not look like a stronger version of our claim. It may not look like our claim at all, in fact! To see why this actually **is** our claim, think about what happens when we let $n = l$. If our claim holds for this value of l, we will have made a

[6] n-player chess is played like normal chess, except with n players arranged in a cycle, alternating play on the white and black sides. It is recommended to play with an odd number of players, for maximal complexity.

[7] A proof is called constructive when the proof's methods actually construct an instance of the object they claim exists; for instance, in our argument that 2-player games can have efficient tournaments, we proved this by creating efficient tournaments. Not all proofs are constructive: think of our probabilistic proofs from the winter, where we showed that certain things must exist without actually explaining how you would make them!

[8] A **k-parallel class** A is a collection of subsets $\{S_1, \dots S_{n/k}\}$ of subsets of $\{1, \dots n\}$, all size k, that partition the set $\{1, \dots n\}$: in other words, every element of $\{1, \dots n\}$ is in exactly one subset S_i. Two such parallel classes are called **disjoint** if they do not contain any of the same k-element subsets. We will usually just say "parallel class" instead of k-parallel class if the value of k is understood.

[9] A **m-partition** of a set X is a collection A of m pairwise disjoint subsets of X, some of which can be empty, so that their union is X.

collection of M different m-partitions of $\{1, \dots n\}$ with the following property: each subset S of $\{1 \dots n\}$ shows up in $\binom{0}{k-|S|}$-many collections. In other words, the only subsets that show up are those with $|S| = k$ (because that's the only value for which $\binom{0}{k-|S|} \neq 0$,) and those all show up precisely once. In other words, we will have created a collection of $M = \binom{n-1}{k-1}$ many partitions of $\{1, \dots n\}$, each of which is made out of different k-element subsets. In other words, we have $\binom{n-1}{k-1}$ disjoint parallel classes, as desired!

We prove this claim by induction on l. Notice that for $l = 1$, our claim is very easy to prove: we are asking for a collection $A_1, \dots A_M$ of m-partitions of $\{1\}$ with the property that each subset S of $\{1\}$ shows up in $\binom{n-1}{k-|S|}$ of these partitions.

In this case, we can notice that we actually know what all of these objects must be:

1. If A is any m-partition of $\{1\}$, then A consists of one copy of the set $\{1\}$ and $m-1$ copies of the set $\emptyset$; this is the only way to break $\{1\}$ into m disjoint subsets!

2. If we do this, then the empty set shows up in all $M = \binom{n-1}{k-1}$ of these partitions, occurring $m-1 = \frac{n}{k} - 1 = \frac{n-k}{k}$ times in each partition. This gives us $\frac{n-k}{k} \cdot \frac{(n-1)!}{(k-1)!(n-k)!} = \frac{(n-1)!}{k!(n-k-1)!} = \binom{n-1}{k}$ many occurrences of the empty set, which is $\binom{n-1}{k-|\emptyset|}$ as claimed.

3. Similarly, the number of times the set $\{1\}$ shows up is once in each m-partition. There are $M = \binom{n-1}{k-1} = \binom{n-1}{k-|\{1\}|}$ such partitions, which tells us that this set also shows up the claimed number of times.

4. There are no other subsets of $\{1\}$, so we've verified our claim for all possible subsets and thus checked our base case!

We now proceed to the inductive step. Assume that we've demonstrated our claim for some value of $l < n$, and that we have the desired $\{A_1, \dots A_M\}$ set of m-partitions of $\{1, \dots l\}$. We will prove our claim for $l+1$ using the max-flow min-cut theorem, by considering the following network:

- Let s, t be source and sink vertices. For each of our m-partitions A_i create a vertex A_i, and for each subset $S \subset \{1, \dots l\}$ create a vertex S.

- For every i, create an edge $\{s, A_i\}$ from the source to the A_i vertex. Similarly, for every subset S, create an edge $\{S, t\}$ from the S-vertex to the sink. As well, add in the edge $\{A_i, S\}$ whenever $S \in A_i$.

- Define a capacity function on this graph by setting $c(s, A_i) = 1, c(A_i, s) = 0$, $c(A_i, S) = \infty, c(S, A_i) = 0$, $c(S, t) = \binom{n-l-1}{k-|S|-1}$, and $c(t, S) = 0$.

Consider the following flow f:

- $f(s, A_i) = 1$, for all A_i.

- $f(A_i, S) = \frac{k-|S|}{n-l}$, and

- $f(S, t) = \binom{n-l-1}{k-|S|-1}$.

I claim this is actually a flow. To see this, note that the sum of the flows leaving A_i is

$$\begin{aligned}\sum_{S \in A_i} \frac{k-|S|}{n-l} &= \frac{1}{n-l} \cdot \sum_{S \in A_i} (k-|S|) \\ &= \frac{1}{n-l} \cdot \left(mk - \sum_{S \in A_i} |S| \right) \\ &= \frac{1}{n-l} \cdot (mk-l) \\ &= \frac{1}{n-l} \cdot (n-l) \\ &= 1,\end{aligned}$$

which is precisely the flow entering A_i.

As well, the sum of values entering a vertex S is

$$\begin{aligned}\sum_{i:S \in A_i} \frac{k-|S|}{n-l} &= \frac{k-|S|}{n-l} \cdot \binom{n-l}{k-|S|} \\ &= \binom{n-l-1}{k-|S|-1},\end{aligned}$$

which is precisely the flow leaving S.

So this is a flow! Furthermore, it's a maximal flow with value M, because it saturates all of the edges leaving s.

We... don't actually care about this flow. The only thing we care about is that we have proven that there **is** a flow with value M. So, in particular, we know that any maximal flow has value M, and in particular is maxed out on all of the $\{s, A_i\}$ and $\{S, t\}$ edges. Therefore, if we use the max-flow min-cut theorem to get a maximal flow, because all of the capacity values are integral, we'll get a flow where all of the values are integers that stays maxed out on these $\{s, A_i\}$, $\{S, t\}$ edges. This is the flow we want!

Let f' be such an integral flow. Because f is 1 on all of the $\{s, A_i\}$ edges, it is one on exactly one $\{A_i, S\}$ edge and 0 on all of the others. We can think of this as matching each A_i to exactly one S subset. For each S, furthermore, we know that this S is picked out by $\binom{n-l-1}{k-|S|-1}$-many different A_i's, because this is the flow leaving each S.

Finally, turn this collection of $\{A_1, \dots A_M\}$'s into a collection $\{A'_1, \dots A'_M\}$, by taking A_i and replacing its assigned subset S_i with $S_i \cup \{l+1\}$.

How many times does each subset $S \subseteq \{1, 2, \dots l+1\}$ show up in our new collection $\{A'_1, \dots A'_n\}$? First, consider any subset of the form $S = S' \cup \{l+1\}$, where $S' \subset \{1, \dots l\}$. By construction, we know that this set shows up in each A'_i-partition that had S as its assigned chosen subset, and in no other places (as there was no other way to get the $l+1$ term.) There were precisely $\binom{n-l-1}{k-|S|-1} = \binom{n-(l+1)}{k-(|S|+1)}$-many such subsets; so our claim holds for all of these kinds of sets!

Now, examine the subsets that don't contain $l+1$: i.e. $S \subset \{1, \dots l\}$. There were originally $\binom{n-l}{k-|S|}$-many of these subsets, and we took away $\binom{n-l-1}{k-|S|-1}$ of these to form new $l+1$ subsets; this leaves

$$\begin{aligned}\binom{n-l}{k-|S|} - \binom{n-l-1}{k-|S|-1} &= \frac{n-l}{k-|S|} \cdot \binom{n-l-1}{k-|S|-1} - \binom{n-l-1}{k-|S|-1} \\ &= \frac{n-l-k+|S|}{k-|S|} \cdot \binom{n-l-1}{k-|S|-1} \\ &= \binom{n-l-1}{k-|S|},\end{aligned}$$

which is again what we claimed. So we have proven our result by induction: setting $l = n$ then gives us our desired collection of disjoint parallel classes.

□

Tournaments! Surprisingly hard.

3 Algebraic Flows

We finish up our notes by inverting the way we've been using flows so far; instead of using flows to study other problems in combinatorics, why not study flows themselves via combinatorial methods? We do this via the following slight "generalization" of flows to abelian groups:

4 Definitions and Fundamental Results

Definition. Let A be an abelian group. An **A-circulation** on a graph G is any function $f : V(G) \times V(G) \to A$ with the following properties:

1. For any $x, y \in V(G)$ with $\{x, y\} \notin E(G)$, we have $f_{xy} = 0$.
2. For any $x, y \in V(G)$ we have $f_{xy} = -f_{yx}$.
3. For any vertex $x \in V(G)$, we satisfy Kirchoff's law at x: that is,

$$\sum_{y \in N(x)} f_{x,y} = 0.$$

This is kind-of like a flow from before, except that we don't have a capacity function, nor do we have special source/sink vertices; we instead require our graph to satisfy Kirchoff's law everywhere!

Definition. Let A be an abelian group. An **A-flow** on a graph G is an A-circulation where $f_{xy} \neq 0$ for any edge $\{x, y\}$ in $E(G)$.

An A-flow is simply a circulation where we make sure that our flow along every edge is nonzero; in other words, the flow doesn't "pretend" some of our edges are not there.

Very surprisingly, the

We state a few surprising results here, the first of which was on your HW yesterday:

Theorem 8. *For any graph G, there is a polynomial P such that the number of distinct A-flows on G is given by $P(n-1)$, where $|A| = n$.*

Proof. (Sketch:) Work by induction on $|E|$. Take any edge e_0 in your graph, and consider the two graphs $G \setminus \{e_0\}$, where you just delete the edge, and $G/\{e_0\}$, where you shrink the edge to a point. To each of these graphs, apply our inductive hypothesis, to get a pair of polynomials $P_2(x)$ and $P_1(x)$; finally, use a counting argument on the A-flows of G to show that $P_2(n) - P_1(n)$ is the number of distinct A-flows on G. □

The main reason we mention the above result is that it, in some sense, says that our choice of group for a A-flow doesn't really matter! – the only important thing is how many elements it has in it. To formally state this as a corollary:

Corollary 9. *If A, B are a pair of groups with $|A| = |B|$, and G is a graph, then there is an A-flow on G if and only if there is a B-flow on G.*

In a sense, we have reduced our study of A-flows to the study of $\mathbb{Z}/k\mathbb{Z}$-flows, which should make our life remarkably easier. One thing we might wonder, now, is whether we can turn these $\mathbb{Z}/k\mathbb{Z}$-flows into actual $\mathbb{Z}$-flows: i.e. whether by being clever, we can transform the weaker requirement of

$$\sum_{w \in N^+(v)} g(v,w) - \sum_{w \in N^-(v)} g(w,v) \equiv 0 \mod k$$

into the stronger requirement

$$\sum_{w \in N^+(v)} g(v,w) - \sum_{w \in N^-(v)} g(w,v) = 0.$$

As it turns out, we can!

Definition. A k-flow f on a graph G is a mapping $f : E(G) \to \mathbb{Z}$, such that $0 < |f(e)| < k$ and f satisfies Kirchoff's laws at every vertex.

Theorem 10. *A graph G has a k-flow iff it has a $\mathbb{Z}/k\mathbb{Z}$-flow.*

Proof. (Sketch:) Any k-flow on G satisfies the relation

$$\sum_{w \in N^+(v)} g(v,w) - \sum_{w \in N^-(v)} g(w,v) = 0.$$

at every vertex v. Therefore, this flow trivially induces a $\mathbb{Z}/k\mathbb{Z}$-flow by interpreting all of its values as elements of $\mathbb{Z}/k\mathbb{Z}$ (because if a sum of things is equal to 0, it's certainly equal to $0 \mod k$.)

To go the other way: take any flow $f : E(G) \to \mathbb{Z}/k\mathbb{Z}$. Interpret f as a function $E(G) \to \{1, \ldots k-1\}$. Then all we have to do is for every edge e, decide whether we map it to $f(e)$ or $f(e) - k$, in a sufficiently consistent way that we insure Kirchoff's laws are still obeyed.

It turns out that if you pick an interpretation that minimizes the Kirchoff-sums $\left|\sum_{w \in N^+(v)} g(v,w) - \sum_{w \in N^-(v)} g(w,v)\right|$ at every vertex, it actually does this! (You can show this by contradiction; it's not terribly surprising, and is mostly sum-chasing. □

5 Some Calculations

To recap: we've proven that for an abelian group A with $|A| = k$, a graph G has an A flow iff it has a $\mathbb{Z}/k\mathbb{Z}$-flow iff it has a k-flow. Therefore, in a sense, the only interesting questions to be asked here is the following: given a graph G, for what values of k does G have a k-flow?

We classify some cases here:

Proposition 11. *A graph G has a 1-flow iff it has no edges.*

Proof. This is trivial, as a 1-flow consists of a mapping of G's edges to 0, such that no edge is mapped to, um, 0. □

Proposition 12. *A graph G has a 2-flow iff the degree of all of its vertices are even.*

Proof. A 2-flow consists of a mapping of G's edges to $\{\pm 1\}$, or equivalently a mapping of G's edges to 1 in $\mathbb{Z}/2\mathbb{Z}$, in a way that satisfies Kirchoff's law. However, we trivially satisfy Kirchoff's laws in the $\mathbb{Z}/2\mathbb{Z}$ case iff the degree of every vertex is even; so we know that this condition is equivalent to having a 2-flow. □

Definition. For later reference, call any graph where the degrees of all of its vertices are even a **even** graph; relatedly, call any graph where the degree of any of its vertices is 3 a **cubic** graph.

Proposition 13. *A cubic graph G has a 3-flow iff it is bipartite.*

Proof. A 3-flow consists of a mapping of G's edges to $\{1, 2\}$ in $\mathbb{Z}/3\mathbb{Z}$, in a way that satisfies Kirchoff's law. Suppose that we have some such flow on our graph: call it f.

Take any cycle $(v_1, v_2, , v_n)$ in this graph, and consider any two consecutive edges $(v_1, v_2), (v_2, v_3)$. Suppose f assigned these the same value, and let w be v_2's third distinct neighbor: then, by Kirchoff's law, we know that

$$-f(v_1, v_2) + f(v_2, v_3) + f(v_2, w) = 0 \Rightarrow f(v_2, w) = 0.$$

But f is a $\mathbb{Z}/3\mathbb{Z}$-flow: so this cannot happen! Therefore, the values 1 and 2 have to occur alternately on this cycle, and therefore it must have even length. Having all of your cycles be of even length is an equivalent condition to being bipartite, so we know that our graph must be bipartite.

To see the other direction: suppose that G is bipartite, with bipartition V_1, V_2. Define $f(x, y) = 1$ and $f(y, x) = -1 = 2$, for all $x \in V_1, y \in V_2$; this evaluation means that for any $x \in V_1$, we have

$$f(x, y_1) + f(x, y_2) + f(x, y_3) = 1 + 1 + 1 \equiv 0 \mod 3$$

and for any $y \in V_2$, we have

$$f(y, x_1) + f(y, x_2) + f(y, x_3) = 2 + 2 + 2 \equiv 0 \mod 3$$

by summing over their three neighbors in the other part of the partition. So this is a 3-flow, and we've proven the other direction of our claim. □

For simplicity's sake, we introduce the function $\varphi(G)$ to denote the smallest value of k for which G admits a k-flow: if no such value exists, we say that $\varphi(G) = \infty$.

We now list here a series of claims whose proofs we leave for the HW:

Proposition 14. $\varphi(K_2) = \infty$.

Proposition 15. *More generally, a connected graph G has $\varphi(G) = \infty$ whenever G has a **bridge**: i.e. there is an edge $e \in E(G)$ such that removing e from G disconnects G.*

Proposition 16. $\varphi(K_4) = 4$.

Proposition 17. *If n is even and not equal to 2 or 4, then $\varphi(K_n) = 4$.*

Proposition 18. *A graph G has a 4-flow if and only if we can write it as the **union** of two even graphs: i.e. there are a pair of graphs G_1, G_2, with possibly overlapping edge and vertex sets, such that $V(G) = V(G_1) \cup V(G_2), E(G) = E(G_1) \cup E(G_2)$.*

Proposition 19. *A cubic graph G has a 4-flow if and only if it is three-edge colorable.*

Corollary 20. *The Petersen graph has no 4-flow.*

On the HW, you (hopefully) showed that the Petersen graph does have a 5-flow earlier in the week; therefore, we know that $\varphi(\text{Pete}) = 5$.

6 Open Conjectures

Surprisingly, when you start computing more of these numbers, it seems pretty much impossible to find anything that's got a value of φ bigger than 5 and not yet infinity. This motivated Tutte to make the following conjectures on k-flows, which are still open to this day!

Conjecture 21. *Every bridgeless graph has a 5-flow.*

More surprisingly, it seems like the Petersen graph, in a sense, is unique amongst graphs that have 5-flows:

Conjecture 22. *Every bridgeless graph that doesn't contain the Petersen graph as a minor has a 4-flow.*

Currently, the best known result is a theorem of Seymour, from 1981, that proves that every bridgeless multigraph has a 6-flow: due to the lack of time, we omit this proof here. (Interested students should come and talk to me if they want to see it!) Still, the gap between 6 and 5 is wide-open, as is the improvement to 4 in the case of graphs that don't contain the Petersen graph as a minor.