Math 3322: Graph Theory

Chapters 5–7

Mikhail Lavrov (mlavrov@kennesaw.edu)

Spring 2021

Two notions of connectivity

We are about to start our discussion of connectivity of graphs. This involves measuring how resilient graphs are to being disconnected.

There are two natural ways to quantify the resilience of a connected graph:

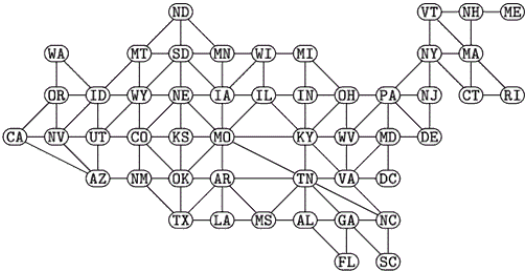
1. Edge connectivity: how many edges must be deleted to disconnect a graph?

   (The answer is just 1, if the graph has a bridge.)

2. Vertex connectivity: how many **vertices** must be deleted to disconnect a graph?

Connectivity of the US

The graph of the continental US has a single bridge: the edge between ME and NH.



Accordingly, NH is a **cut vertex**: deleting it disconnects the graph.

NY is also a cut vertex, though no bridges are involved. If you're a wanted criminal in NY, and can't go there without being arrested, you can't get from GA to VT!

Definition of a cut vertex

In general, we say that a vertex v of a graph G is a **cut vertex** if $G - v$ has more components than v.

(Usually, G will be connected, in which case this means $G - v$ is disconnected.)

We will also prove the following characterization of cut vertices:

Theorem. If G is a connected graph, a vertex v is a cut vertex of G iff there are vertices $u, w \neq v$ such that v lies on every $u - w$ path.

In other words, v is a cut vertex if it is the only way to get from u to w for some u and w.

(When G is not connected, the theorem is still true if any $u - w$ paths actually do exist.)

Cut vertices and bridges

As we saw with NH and ME, a bridge in a graph tends to result in cut vertices. There is an exception to this: if vw is a bridge, but v has degree 1, then v is not a cut vertex! (Just like ME.)

Theorem. If vw is a bridge of G, and $\deg(v) \geq 2$, then v is a cut vertex.

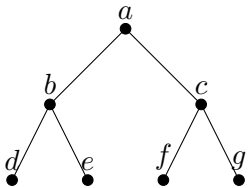
Proof. Let u be a neighbor of v other than w. There is at least one $u - w$ path: (u, v, w).

Suppose there is a $u - w$ path $(u = v_0, v_1, \ldots, v_{k-1}, v_k = w)$ that **does not** contain v. Then $(u, v_1, \ldots, v_{k-1}, w, v, u)$ is a cycle containing vw, and vw would not be a bridge.

Since vw is a bridge, there cannot be a $u - w$ path that does not contain v, so v is a cut vertex. □

Cut vertices in trees

What are the cut vertices in a tree?



In the diagram above, a, b, c are cut vertices; d, e, f, g are not.

In general, the cut vertices of a tree are exactly those that are not leaves! That's because every edge is a bridge, so both endpoints of each edge are cut vertices, **unless** they're leaves of the tree.

(If v is a leaf of T, then $T - v$ is a smaller tree, so v isn't a cut vertex.)

Cut vertices and paths

Let's prove the theorem:

Theorem. If G is a connected graph, a vertex v is a cut vertex of G iff there are vertices $u, w \neq v$ such that v lies on every $u - w$ path.

Proof. First, let's prove the forward direction: assume G is a connected graph, and v is a cut vertex of G.

That tells us that $G - v$ is not connected. So let's choose u, w in two different components of $G - v$. Then:

- Because G is connected, there are one or more $u - w$ paths in G.
- None of them survive to $G - v$, because there are no $u - w$ paths in $G - v$.

Therefore all of the $u - w$ paths in G must contain v.

Cut vertices and paths II

Let's prove the reverse direction. Suppose G is a connected graph and u, v, w are three vertices such that v lies on every $u - w$ path.

Then $G - v$ is not connected: there cannot be any $u - w$ paths in $G - v$. Therefore v is a cut vertex. □

We have already seen how to use this theorem. The vertices u and w are a sort of "witness" to v being a cut vertex.

- If we assume v is a cut vertex in a proof, we can "summon" u and w out of nowhere.
- If we prove that no such u and w can exist, then we conclude that v is not a cut vertex.

Cut vertices and distance

Theorem. Let u be any vertex in G, and let v be a vertex as far from u as possible: maximizing $d(u, v)$. Then v is not a cut vertex.

Proof. Your textbook gives a different proof, but we have a bit more machinery than the textbook, so here's a quicker argument.

Let T be the shortest-path tree of G starting from u. A vertex at distance k from u has one neighbor at distance $k-1$, and potentially some neighbors at distance $k+1$.

In particular, v only has one neighbor (there are no vertices at higher distance from u). So v is a leaf of T. Therefore $T - v$ is still connected.

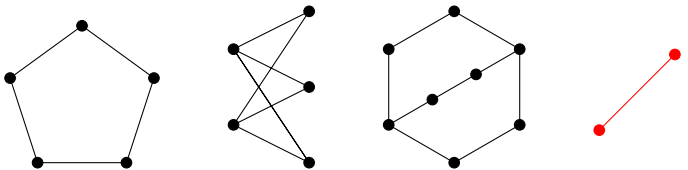
Therefore $G - v$ is also connected (it has $T - v$ as a subgraph) and v is not a cut vertex. □

Nonseparable graphs

We say that a graph is 2-**connected** or **nonseparable** if it is connected and has no cut vertices.

That is, G is connected, and for any vertex v, $G - v$ is still connected.

Some examples (note that K_2 is an iffy case):



For $n \geq 3$, a 2-connected graph G with n vertices needs $\delta(G) \geq 2$, but that by itself is not enough. (It's not even enough to ensure that G is connected.)

2-connected graphs and cycles

As usual, we want a characterization of 2-connected graphs to give us more to work with. ("No cut vertices" is a negative condition; often that's not what we want in proofs.)

Theorem. A graph G with $n \geq 3$ vertices is 2-connected if and only every two vertices u, v are part of a cycle together.

A cycle containing u and v means that there are two $u - v$ paths sharing no vertices: if we delete any one vertex, one of those $u - v$ paths survives.

If this is true for all u and v, then there cannot be cut vertices.

This is a proof that all graphs with this condition are 2-connected. The other direction is harder; brace yourself.

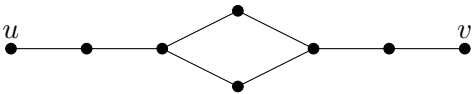
Trying to prove the other direction

Here is what my first thought for a proof was: it did not work.

Suppose that the graph is 2-connected, and u, v are two vertices.

- By connectivity, we must have a $u - v$ path,
- If we delete any vertex on that path, there must still be a $u - v$ path without that vertex;
- Maybe those two paths form a cycle?

They don't always. We might also see:



(Also, if u, v are adjacent, we need a different argument!)

Induction on distance

To help control what the $u - v$ paths do, let's induct on the distance $d(u, v)$. (We'll see how this helps in a moment).

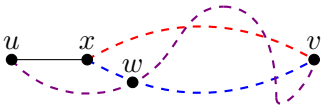
If $d(u, v) = 1$, there is an edge uv. Since $n \geq 3$, edge uv cannot be a bridge (or else u or v would be a cut vertex). So edge uv is contained in a cycle, which was what we wanted.

If $d(u, v) = k$ for some $k > 1$, then u has some neighbor x "on the way to v" such that $d(x, v) = k - 1$. If we're inducting on the distances, we get to assume that there's a cycle containing v and x: in other words, two $x - v$ paths that don't share any vertices.

That's the motivation for induction. We get two $u - v$ paths that **almost** don't share any vertices: only x is shared!

Finishing the proof

Here's a picture of what we have so far (the red and blue dashed lines are paths with no common vertices):



Additionally, there must still be a $u - v$ path if we delete x (in purple). If this doesn't share any vertices with the red or blue paths, we'd be done; but it might.

Let w be the first time that the purple path meets the red or blue path. Say it first meets the blue path. Then we find a cycle by following the purple path to w; then the blue path to v; then the red path to x; then straight to u. □

Connected components: a review

Let's quickly summarize how we define the connected components of a graph G. There are two ways:

1. Define a relation on $V(G)$: $u \sim v$ if there is a $u - v$ walk. We prove that this is an equivalence relation, and then we get a partition of $V(G)$ into equivalence classes $V_1 \cup V_2 \cup \cdots \cup V_k$.

   The connected components $G_1, G_2, \ldots, G_k$ are the subgraphs induced by these equivalence classes.

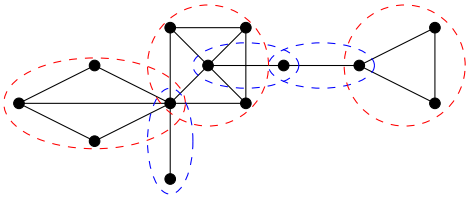
2. A connected component of G is a maximal connected subgraph H: it's not contained in any larger connected subgraph.

   Here, the hard part is making sure that two such components don't share any vertices, so that we get what we expect.

Blocks

A **block** is like a connected component, but for 2-connectivity. We'll define it in the second way first: a subgraph H is a block of G if it is 2-connected, and not contained in any larger 2-connected subgraph.

What do these look like? Consider an example graph:



This graph has 6 blocks. Three of them come from the bridges, and three are more complicated 2-connected subgraphs. Each cut vertex is contained in multiple blocks.

Block properties

Let me summarize the features of blocks we see in this picture, and which hold for all graphs:

- The blocks are edge-disjoint (they share no edges), and every edge is in some block.
- Two blocks overlap in either no vertices, or else at most one vertex.
- If a vertex is part of multiple blocks, then it is a cut vertex of the graph.

Moreover, the blocks of any graph form a kind of tree-like structure. (They cannot be connected in a loop, or else they'd form one big block.) But this is tricky to even state precisely.

Blocks and equivalence relations

We can also define blocks using an equivalence relation. But the blocks no longer partition the vertices...

...Instead, we define blocks by an equivalence relation on edges. This equivalence relation says $e \backsim f$ if $e = f$ or the edges e, f lie on a common cycle.

Claims that are implied:

- This is actually an equivalence relation. In particular, if $e_1 \backsim e_2$ and $e_2 \backsim e_3$, then $e_1 \backsim e_3$.
- The equivalence classes match the blocks defined earlier.
- In particular: if G is a 2-connected graph, then any two **edges** lie on a common cycle. (We'll prove this later!)

Definitions

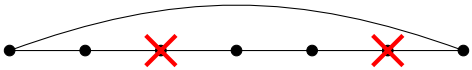
Generalizing what we've done so far:

- A **vertex cut** in a graph G is a subset $U \subseteq V(G)$ such that $G - U$ (G with all vertices in U removed) is disconnected.
- We say that a graph G **is k-connected** if all vertex cuts in G have at least k vertices; "1-connected" is the same as "connected".
- The **vertex connectivity** (or just connectivity) of G, denoted $\kappa(G)$, is the minimum number of vertices in any vertex cut.

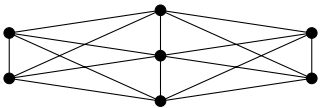
The relationship between these definitions: "G is k-connected" means "$\kappa(G) \geq k$". This has a special name because $\kappa(G)$ often needs to be **at least** k to make us happy.

Examples

The connectivity $\kappa(C_n)$ is 2. Deleting any one vertex from a cycle leaves a path; deleting 2 vertices disconnects the cycle.



What is the connectivity of the graph G below?

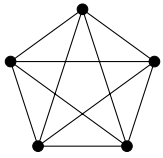


$\kappa(G) \leq 3$: delete the 3 vertices in the middle column.

$\kappa(G) \geq 3$: the middle vertex must go, and then there's a cycle through the other 6 vertices, needing 2 more to be deleted.

Complete graphs: an awkward case

What should be the value $\kappa(K_n)$: the minimum number of vertices needed to disconnect K_n?



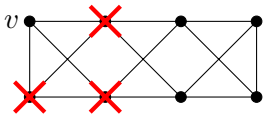
We have a problem: we can't disconnect this graph by deleting any number of vertices! We'll define $\kappa(K_n) = n - 1$ so that our future theorems about connectivity also apply to complete graphs.

Note: this means $\kappa(K_2) = 1$, though I said last time K_2 was 2-connected. Awkward! It needs to be 2-connected when we're looking for blocks, but is a special case otherwise.

An upper bound on connectivity

Claim. For all graphs G, $\kappa(G) \leq \delta(G)$.

Proof. If G is not a complete graph, pick a vertex v with $\deg(v) = \delta(G)$.



Delete every vertex adjacent to v: now v is an isolated vertex, and can't get to any remaining vertices.

This doesn't work when $G = K_n$: if we delete every vertex adjacent to v, we are left with v and nothing else, so the result is still connected. But $\delta(K_n) = n - 1$ and $\kappa(K_n) = n - 1$, so $\kappa(K_n) \leq \delta(K_n)$. □

Edge connectivity

We can make similar definitions for edges:

- An **edge cut** in a graph G is a subset $X \subseteq E(G)$ such that $G - X$ (G with all edges in X, and no vertices, removed) is disconnected.

  For connected graphs, an edge cut of size 1 is just a bridge.

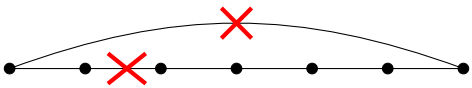
- We say that G **is k-edge-connected** if all edge cuts in G have at least k edges; "1-edge-connected" is still the same as "connected".

- The **edge connectivity** of a graph G, denoted $\kappa'(G)$, is the minimum number of edges in any edge cut.

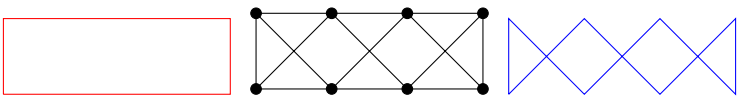
Note on notation: I use κ' (your textbook uses λ) as part of a general rule that adding a prime to a Greek letter gives us the "edge version" of that property. Fewer Greek letters to learn!

Examples

The edge connectivity $\kappa'(C_n)$ is still 2. Deleting any edge leaves a path; deleting 2 edges disconnects the cycle.



What is the edge connectivity of the graph below?



$\kappa'(G) \leq 3$: we can just delete all three edges out of a corner vertex.

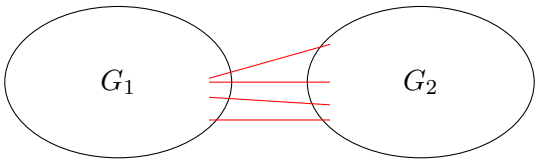
$\kappa'(G) > 2$: the red and blue cycles must each lose at least 2 edges. But they only share 2 edges, and deleting those doesn't work.

An improved bound

Theorem. For all graphs G, $\kappa(G) \leq \kappa'(G) \leq \delta(G)$.

Proof. We already saw by example why $\kappa'(G) \leq \delta(G)$: delete all edges out of a minimum-degree vertex.

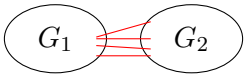
A proof that $\kappa(G) \leq \kappa'(G)$ should look like this: take an edge cut with k edges, and use it to find a vertex cut with at most k vertices.



Intuition: deleting a vertex deletes its endpoints, so by deleting k (or fewer) vertices, we can delete these k edges as well!

Proof of the bound

We have an edge cut $\{e_1, \ldots, e_k\}$ and need to find k vertices to delete.



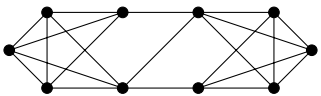
For each i, delete any endpoint of e_i. This also destroys e_i, so we delete $\leq k$ vertices and disconnect G_1 and $G_2 \ldots$ unless we accidentally delete all vertices in G_1 (or in G_2).

To avoid this, pick $u \in V(G_1)$ and $v \in V(G_2)$ to preserve. Now do the same thing, but if e_i is incident to u or v, delete the other endpoint. This disconnects u from $v \ldots$ unless edge uv is in our edge cut.

To avoid this, pick u, v such that uv is not an edge. Or, if all edges between G_1 and G_2 are present, then $\kappa(G) \leq n - 1 \leq k$ by counting. *(This last bit skips some steps.)* □

All three can be different!

There exist graphs G for which $\kappa(G) < \kappa'(G) < \delta(G)$. An example:



Here, $\kappa(G) = 2$, $\kappa'(G) = 3$, and $\delta(G) = 4$.

In some special cases, we know more. For example, when G is 3-regular, we must have $\kappa(G) = \kappa'(G)$.

(But we could have $\kappa(G) = \kappa'(G) = 0$, or $\kappa(G) = \kappa'(G) = 1$, or $\kappa(G) = \kappa'(G) = 2$, or $\kappa(G) = \kappa'(G) = 3$.)

Separating s from t

Let s, t be two vertices in a graph G. An $s-t$ **separator** or $s-t$ **cut** is a vertex cut U that separates s from t: $G-U$ still contains s and t, but they're in different components. We write

$$\kappa(s,t) = \min\{|U| : U \text{ is an } s-t \text{ cut}\}.$$

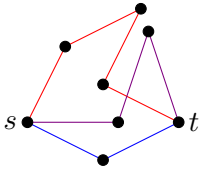
Motivation:

- We will not discuss **how**, but we can use network flows (the Edmonds–Karp algorithm and friends) to compute $\kappa(s,t)$.
- Sometimes, if s and t are a "starting point" and "destination" in our problem, we care about $\kappa(s,t)$ more than $\kappa(G)$.
- Other times, we can compute $\kappa(G)$ by computing several $\kappa(s,t)$'s and taking the minimum.

Vertex-disjoint paths

If you want to demonstrate that $\kappa(s,t) \leq k$, it's enough to find an $s-t$ cut U with $|U| = k$. What if you want to demonstrate that $\kappa(s,t) \geq k$?

One way to do it is to find k **internally disjoint** $s-t$ **paths**:



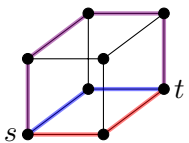
To disconnect s from t, at least one vertex from each path must be deleted. No vertex deletion can destroy two paths, so we need to delete at least k vertices.

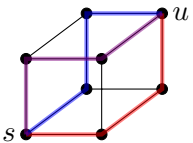
Later this week, we'll see that such a demonstration always exists.

Example: connectivity of the cube graph

Claim. $\kappa(Q_3) = 3$.

Proof. We consider vertices s, t that are two steps apart, and vertices s, u that are three steps apart. (Adjacent vertices can never be separated by a vertex cut.)





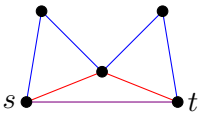
Therefore $\kappa(s,t), \kappa(s,u) \geq 3$. This means deleting two vertices cannot separate a vertex from any other vertex; so $\kappa(Q_3) \geq 3$.

But $\kappa(Q_3) \leq \delta(Q_3) = 3$, so $\kappa(Q_3) = 3$ exactly. □

Edge separators

Everything we've done for vertices can be done for edges as well. We define $\kappa'(s,t)$ to be the least size of an $s-t$ **edge cut**: a set $S \subseteq E(G)$ such that s,t are in different components of $G-S$.

Lower bounds on $\kappa'(s,t)$ can come from edge-disjoint $s-t$ paths:



One important difference: when there is an edge st, $\kappa(s,t) = \infty$. We can never disconnect s from t by deleting vertices! But $\kappa'(s,t)$ is not affected as much: an edge st just gives one more $s-t$ path.

Finding $\kappa(G)$ from $\kappa(s,t)$

Suppose you have a magical genie that will tell you $\kappa(s,t)$ for any two vertices $s,t \in V(G)$. How can we compute $\kappa(G)$?

Brute force: just let $\kappa(G)$ be the minimum of $\kappa(s,t)$ over all s,t.

Better approach: pick a vertex v, and let $\kappa(G)$ be the minimum of

- $\kappa(v,w)$ over all w not adjacent to v. *(This takes care of cases where v is not in the cut.)*
- $\kappa(x,y)$ over all x,y adjacent to v. *(This takes care of cases where v is in the cut.)*

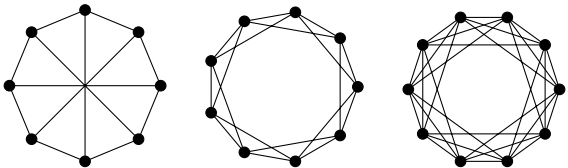
Edge connectivity is simpler: just pick any v, and let $\kappa'(G)$ be the minimum of $\kappa'(v,w)$ over all $w \neq v$.

Harary graphs and connectivity

What is the least number of edges we need to make a k-connected graph on n vertices?

We know that $\kappa(G) \leq \delta(G)$, so we want $\delta(G) \geq k$. If all vertices have degree at least k, then we need at least $\frac{1}{2}kn$ edges.

When $k = 1$, this says "we need at least $\frac{1}{2}n$ edges" though actually we need $n - 1$ edges to connect a graph. But for $k \geq 2$, this is the best bound there is. The Harary graphs, which we've seen before, achieve it:



The theorem on Harary graphs

Theorem. When r is even, the Harary graph $H_{n,r}$ has $\frac{1}{2}nr$ edges and $\kappa(H_{n,r}) = r$.

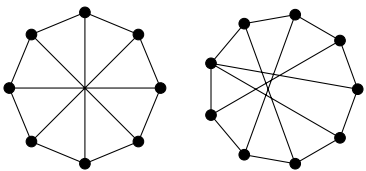
(We look at the even case because it's easier to describe: each vertex has an edge to the $\frac{r}{2}$ closest vertices in either direction.)

Proof. Strategy: suppose $|U| \leq r - 1$. We show that $H_{n,r} - U$ is connected by picking two vertices $v, w \in V(H_{n,r} - U)$, and checking that there's a $v - w$ path.

We try to go either "clockwise" or "counterclockwise" from v to w. One of those directions has fewer than $\frac{r}{2}$ vertices of U. In that direction, there must be a $v - w$ path: it would take $\frac{r}{2}$ consecutive deleted vertices to stop our progress. □

Harary graphs for odd r

For odd r, there are two cases. When n is even, we still have an r-regular graph $H_{n,r}$ with $\frac{1}{2}nr$ edges, and it still has $\kappa(H_{n,r}) = r$.



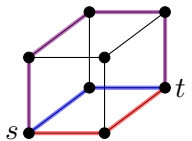
When n and r are both odd, we can construct a graph H with $\frac{1}{2}(nr+1)$ edges that has $\kappa(H) = r$. Add $\frac{r+1}{2}$ edges to $H_{n,r-1}$ that connect nearly opposite vertices.

As you might imagine, these two cases, and especially the last, are more annoying in proofs, so we'll skip them.

$s-t$ cuts and $s-t$ paths

Recall: if s,t are vertices in G, a vertex set $U \subseteq V(G) - \{s,t\}$ is an $s-t$ **cut** if s,t are in different components of $G-U$. We write $\kappa(s,t)$ for the minimum number of vertices in an $s-t$ cut.

We saw earlier that if G has a set of k internally disjoint $s-t$ paths (that is, $s-t$ paths that share no vertices except for their endpoints s and t) then $\kappa(s,t) \geq k$: this is a short way to prove lower bounds.



Menger's theorem. A short proof of this form always exists: if $\kappa(s,t) = k$, then G has k internally disjoint $s-t$ paths.

Menger's theorem for $k = 2$

We have already proven that in a 2-connected graph G, any two vertices lie on a common cycle.

A cycle C containing two vertices s and t is the same as two vertex-disjoint $s - t$ paths: informally, one path goes "clockwise" and one "counterclockwise" around C. So this is essentially Menger's theorem for the $\kappa(G) = 2$ case.

There's a subtle difference:

- We proved that if $\kappa(G) \geq 2$, then any two vertices are on a common cycle.
- Menger's theorem says that if only some pairs s, t have $\kappa(s, t) \geq 2$, but the graph as a whole is not 2-connected, there is still a common cycle containing any such pair.

Strong duality

If s, t are two vertices in G, we have two optimization problems:

1. What is k, the **minimum** number of vertices in an $s-t$ cut?
2. What is ℓ, the **maximum** number of internally disjoint $s-t$ paths?

These are opposing problems: we already saw that $k \geq \ell$. In fact, the number of vertices in any $s-t$ cut must be larger than the number of paths in any set of internally disjoint $s-t$ paths.

(If you have a set of internally disjoint $s-t$ paths, any cut must delete a separate vertex from each path.)

This relationship is known as **weak duality**. Menger's theorem provides an example of **strong duality**: $k = \ell$. The optimal solution to one problem meets the optimal solution to the other.

Proof of the theorem

Theorem. Let s, t be two vertices of a graph G with $st \notin E(G)$ and let $\kappa(s,t) = k$. Then G has k internally disjoint $s-t$ paths.

There's two equivalent ways to set up the proof I will present:

1. Induct on the number of edges in G. Find ways to simplify G to a smaller graph to which the induction hypothesis can be applied.
2. Suppose the theorem is false, and take the counterexample with the fewest edges.

Your textbook does the first thing; it follows Dirac's 1966 proof of Menger's theorem, which is written in the second way. Either way, we get to assume that the theorem holds for every graph with fewer edges than G.

One consequence of minimality

The first thing we can do is look at an edge $xy \in E(G)$ and ask what happens in $G - xy$ (for which the theorem holds).

If $\kappa(G - xy) = k = \kappa(G)$, then we're in good shape! Since $G - xy$ is a smaller graph, it has k internally disjoint $s - t$ paths, and then G has those paths as well.

If $\kappa(G - xy) = k - 1$, we can't immediately conclude anything. But it will later be useful for us to know that if $\{u_1, u_2, \ldots, u_{k-1}\}$ is an $s - t$ cut in $G - xy$, then:

- $\{u_1, u_2, \ldots, u_{k-1}, x\}$ is an $s - t$ cut in G (if $x \neq s, t$).
- $\{u_1, u_2, \ldots, u_{k-1}, y\}$ is an $s - t$ cut in G (if $y \neq s, t$).

Paths of length 2

Whenever there is a vertex v adjacent to both s and t:

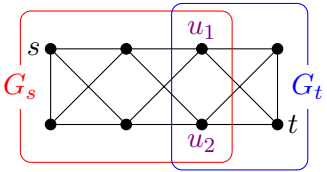
- we get an $s-t$ path (s,v,t) that's very easy to use in our collection of $s-t$ paths: it is internally disjoint from every path not containing v.
- In $G-v$, $\kappa(s,t)=k-1$, because v **must** be part of every $s-t$ cut. That's the only way to destroy the path (s,t,v).

The graph $G-v$ is smaller, so there are $k-1$ internally disjoint $s-t$ paths in $G-v$. Those $k-1$ paths, together with the path (s,v,t), give us k internally disjoint $s-t$ paths in G.

So from now on, we can assume that G has no vertices adjacent to both s and t.

Two separate problems

Let $U = \{u_1, u_2, \ldots, u_k\}$ be an $s - t$ cut of size $k = \kappa(s, t)$.



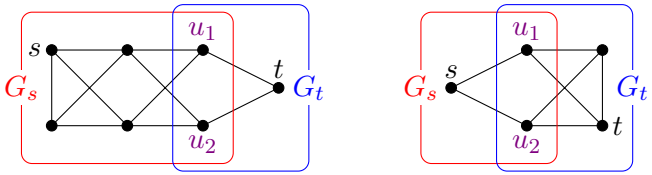
G has a subgraph G_s induced by U and everything on s's side of $G - U$, and a subgraph G_t induced by U and everything on t's side.

To find the k internally disjoint $s - t$ paths we want, we need:

- Internally disjoint $s - u_1, s - u_2, \ldots, s - u_k$ paths in G_s.
- Internally disjoint $u_1 - t, u_2 - t, \ldots, u_k - t$ paths in G_t.

Two smaller problems!

We can often simplify our problem to two smaller ones: one with the same G_s and a smaller G_t, and one with the same G_t and a smaller G_s.



If these are actually both smaller, then we can use the first graph to find paths in G_s, and the second graph to find paths in G_t.

It remains to deal with the case where one of these is **not** smaller.

What this case looks like: for every $s-t$ cut U with $|U|=k$, either s is adjacent to every vertex of U, or else t is.

The remaining case

We are now left with a graph where:

- There are no $s-t$ paths of length 2; $d(s,t) \geq 3$.
- Every $s-t$ cut with k vertices either consists only of neighbors of s, or else only of neighbors of t (and no neighbors of s).

Let $(s, x, y, \ldots, t)$ be an $s-t$ geodesic. We know that $G-xy$ has a $(k-1)$-vertex $s-t$ cut U, and that both $U \cup \{x\}$ and $U \cup \{y\}$ are k-vertex $s-t$ cuts in G.

This is a contradiction! Because $U \cup \{x\}$ contains x, a neighbor of s, all of U must be adjacent to s. Because $U \cup \{y\}$ contains y, a non-neighbor of s, none of U can be adjacent to s.

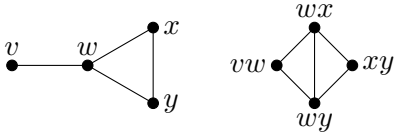
This rules out the last case and finishes the proof. □

Menger's theorem for edge cuts

Theorem. If s, t are two vertices of a graph G, then there are $\kappa'(s,t)$ edge-disjoint $s-t$ paths.

(These are allowed to share vertices, but not edges.)

Idea of proof. We apply Menger's theorem to a graph $L(G)$ called the "line graph" of G. Edges of G become vertices in $L(G)$, and edges that share an endpoint become adjacent vertices. For example:



With a few modifications, edge cuts in G become vertex cuts in $L(G)$, and edge-disjoint paths become vertex-disjoint ones.

Paths and global connectivity

These two versions of Menger's theorem relate $s-t$ paths to $\kappa(s,t)$ and $\kappa'(s,t)$. What about $\kappa(G)$ and $\kappa'(G)$?

- If $\kappa(G)=k$, then $\kappa(s,t)\geq k$ for all $s,t\in V(G)$, so we can find k internally disjoint paths between any two non-adjacent vertices s,t.
- If s and t are adjacent, we can still find k paths: since $G-st$ is $(k-1)$-connected, we can find $k-1$ paths there, and then the last is just the path (s,t).
- Conversely, if all these paths exist, then G must be k-connected, because $\kappa(G)=\min_{s,t}\kappa(s,t)$.
- All the same things happen for $\kappa'(G)$ end edge-disjoint paths; for these, we don't even have to worry about non-adjacent vertices.

Expansion lemma

Lemma. If G is a k-connected graph, and G' is obtained from G by adding a new vertex x with at least k neighbors in G, then G' is also k-connected.

Proof. Almost all $(k-1)$-vertex cuts in G' would also be $(k-1)$-vertex cuts in G, or possibly even $(k-2)$-vertex cuts, if x was one of the deleted vertices.

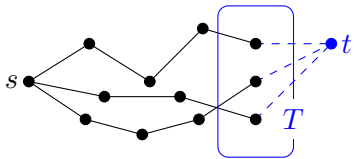
There is one exception.

If there is a vertex cut U such that the components of $G' - U$ are the isolated vertex x and the rest of the graph, then U is not a vertex cut in G.

But in this case, U must contain all the neighbors of x: at least k vertices. □

Dirac's fan lemma

If s is a vertex and T is a set of vertices not containing s, an $s-T$ **fan** is a set of internally disjoint paths starting at s and ending at distinct vertices in T. The **size** of a fan is the number of paths.



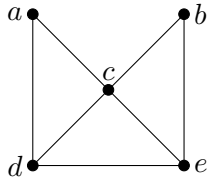
Lemma. Let G be a k-connected graph, $T \subseteq V(G)$ with $|T| \geq k$, and $s \in V(G) - T$. Then G has an $s-T$ fan of size k.

Proof. Add a new vertex t adjacent to every vertex in T, then use Menger's theorem to find k internally disjoint $s-t$ paths. □

Eulerian walks and tours

We will be talking about two traversability problems on graphs. The first of these problems is finding Eulerian walks and tours.

A **Eulerian walk** in a graph G is a walk that includes each edge exactly once: for example, (d, c, b, e, c, a, d, e) in the graph below.

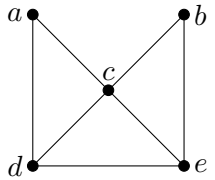


An **Eulerian tour** in a graph G is a closed Eulerian walk: one that starts and ends at the same vertex. We call a graph G Eulerian if it has an Eulerian tour.

Hamiltonian paths and cycles

We will be talking about two traversability problems on graphs. The second of these problems is finding Hamiltonian paths and cycles.

A **Hamiltonian path** in a graph G is a path that includes each vertex exactly once: for example, (d, a, c, b, e) in the graph below.



A **Hamiltonian cycle** in a graph G is a cycle that includes each vertex exactly once: for example, (d, a, c, b, e, d). We call G **Hamiltonian** if it has a Hamiltonian cycle (and **traceable** if it has a Hamiltonian path).

Edge and vertex problems

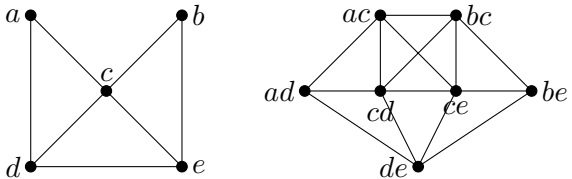
This is not the first or last time we see a "vertex version" and "edge version" of the same problem. Sometimes (as for Menger's theorem) the two problems are very similar... but for traversability, they're very different:

- We can efficiently find Eulerian walks and tours, and there's a simple test to see if a graph is Eulerian.
- Finding Hamiltonian paths or cycles, or testing if a graph has either one, is a famously hard computational problem.

In the case of Menger's theorem, we turned a vertex problem into an edge problem by looking at the line graph. Why doesn't this work for traversability problems?

Problem #1: line graphs only work one way

Consider the same graph we've been looking at, and its line graph:



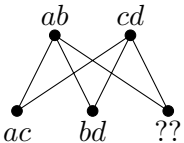
The Eulerian walk (d, c, b, e, c, a, d, e) in G gives a Hamiltonian path $(cd, bc, be, ce, ac, ad, de)$ in $L(G)\ldots$

$\ldots$ but the Hamiltonian path $(ad, ac, bc, cd, ce, be, de)$ in $L(G)$ gives nothing useful in G.

Problem #2: not all graphs are line graphs

If every graph were the line graph of something, we could at least sometimes win: we could try to find a Hamiltonian path/cycle in $L(G)$ by finding an Eulerian walk/tour in G.

But some graphs are not the line graphs of any other graph! For example, take $K_{2,3}$:



The top two vertices are not adjacent; if this were a line graph, they'd correspond to edges ab and cd with no common endpoint.

Their only possible common neighbors are ac, ad, bc, bd. And if we choose any three of these, two of them would be adjacent!

Necessary conditions

Here are some conditions we can prove **must** hold for a graph G to have an Eulerian walk or an Eulerian tour.

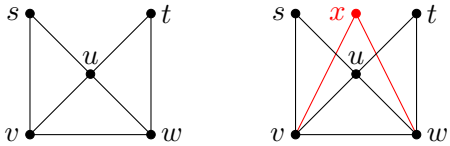
- G must be connected (or else we can't visit all the edges in a single walk), with one exception: isolated vertices are fine.
- If G has a $v - w$ Eulerian walk, every vertex $u \neq v, w$ must have even degree: if u is visited k times, then the walk has $2k$ edges incident to u, so $\deg(u) = 2k$.
- If G has an Eulerian tour, it doesn't matter where we start, so all vertices must have even degree. If G has an Eulerian walk that's not a tour, the start and end must have odd degree.

Plans for the future

Future theorem. Let G be a connected graph in which every vertex has even degree. Then G is Eulerian.

Corollary. Let G be a connected graph in which every vertex has even degree, except for two: v and w. Then G has a $v - w$ Eulerian walk.

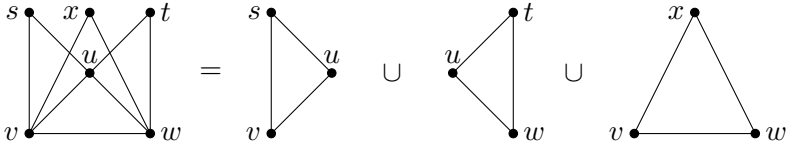
Proof of corollary. We can reduce the problem of finding a $v - w$ Eulerian walk in G to finding an Eulerian tour in a bigger graph H.Let H have one more vertex, x, with two more edges, vx and wx:



Find an Eulerian tour in H, then delete the (v, x, w) segment.

Cycle decompositions

A **cycle decomposition** of a graph G is a collection of **edge-disjoint** cycles $H_1, H_2, \ldots, H_k$ that, together, include each edge of G once.



These have a similar necessary condition for existing: in order for G to have a cycle decomposition, every vertex must have even degree. (Every vertex has degree 0 or 2 in every H_i, and these add up to even totals.)

A graph does not need to be connected to have a cycle decomposition.

Proof strategies

How might we prove our theorem characterizing Eulerian graphs?

Your textbook uses the extremal principle: find the longest closed walk that does not repeat edges, and show that it covers everything.

Our proof will be similar, but we'll take a more satisfying approach: we'll actually describe how to find an Eulerian tour, giving a proof that could be turned into an algorithm.

There are two well-known algorithms for Eulerian tours.

1. **Fleury's algorithm.** "Start a walk, and keep going arbitrarily, but avoid one specific type of mistake."
2. **Hierholzer's algorithm (ish).** "Find a cycle decomposition, then put it together into an Eulerian tour."

Cycle decompositions

Theorem. If every vertex of G has even degree, then G has a cycle decomposition.

Proof. We use strong induction on the number of edges: we assume the theorem for **all** smaller graphs in order to prove it for G.

Base case: if G has no edges, then it has a cycle decomposition consisting of no cycles.

If G has $m > 0$ edges and every vertex of G has even degree, take a connected component G_i with more than one vertex. Then $\delta(G_i) \geq 2$, so G_i contains a cycle C.

$G - C$ has $m - k$ edges, where k is the length of C, and still all even degrees. By induction, $G - C$ has a cycle decomposition. Add C to it, and we get a cycle decomposition of G. □

Proof of the Eulerian graph theorem

Now suppose that G is a **connected** graph with a cycle decomposition $H_1, H_2, \ldots, H_k$.

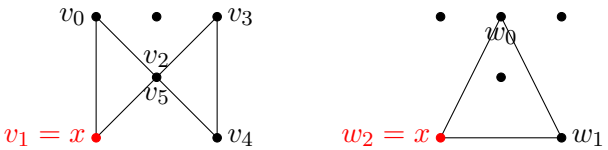
Proof strategy: Start with k closed walks which are just the cycles going around $H_1, \ldots, H_k$. Then combine two closed walks into one over and over, until we are left with just one closed walk.

In order to make this proof strategy work, we need to know the following: if we find two closed walks that share a vertex, we can combine them into a single closed walk using exactly the same edges.

Key point: where does connectivity enter in? Well, at any point before we're done, there must be two closed walks that share a vertex. Otherwise, their vertex sets would give us multiple connected components!

Gluing together two closed walks

Suppose we have two closed walks $(v_0, v_1, \ldots, v_k, v_0)$ and $(w_0, w_1, \ldots, w_\ell, w_0)$ with $v_i = w_j = x$.



Step 1: Rotate the closed walks to start and end at x:

$(x, v_{i+1}, \ldots, v_k, v_0, \ldots, v_{i-1}, x)$ and $(x, w_{j+1}, \ldots, w_\ell, w_0, \ldots, w_{j-1}, x)$.

Step 2: Just put them together (removing the duplicate x):

$$(xv_{i+1}, \ldots, v_k, 0, \ldots, v_{i-1}, x, w_{j+1}, \ldots, w_\ell, w_0, \ldots, w_{j-1}, x).$$

Hierholzer's algorithm

We know how to find a cycle: just pick any starting point and walk until you return to a vertex. So this proof gives us an algorithm for finding an Eulerian tour.

Hierholzer's algorithm is a variant on this idea. Given a closed walk and the subgraph G' of the edges not used in that walk:

1. Find a vertex x of the closed walk with positive degree in G'.
2. Start a walk from x in G'; keep walking **until you return to** x.
3. Glue together the two closed walks, as we just did.

Repeating steps 1–3 (starting from a walk (x) of length 0) grows the closed walk until it is an Eulerian tour.

Fleury's algorithm

Fleury's algorithm is another method to find an Eulerian tour, which I mention for completeness.

It goes like this. From any vertex of the graph, start a walk. Keep going to adjacent vertices, satisfying two conditions:

1. Never use an edge you've used before. Otherwise, we wouldn't get an Eulerian tour at the end!
2. Never take an edge that's a bridge in the graph G' of unused edges, unless it's the last edge out of the current vertex.

   (We don't want to leave two big components, but leaving behind isolated vertices is fine.)

We will not prove that this works.

How do circuits count?

You're designing an electronic device, and you want it to count from 0 to 7. But everything in the device is stored as on/off switches with two states: 0 and 1. How do you count?

In computer software, this is solved by counting in binary:

$$0 = 000 \quad 1 = 001 \quad 2 = 010 \quad 3 = 011 \quad 4 = 100 \quad \dots$$

At the circuit level, and in high-latency applications, this sometimes causes problems. To switch from "011" to "100", you have to flip all three switches simultaneously!

If the switches don't all flip at the same time, you may end up counting

$$0 \quad 1 \quad 02 \quad 3 \quad 764 \quad \dots$$

Gray codes

The **Gray code** is another way to represent the numbers $0, 1, \ldots, 2^n - 1$ by n switches. For $n = 3$, it gives us the representation

$$0 \rightsquigarrow 000 \qquad 1 \rightsquigarrow 001 \qquad 2 \rightsquigarrow 011 \qquad 3 \rightsquigarrow 010$$
$$4 \rightsquigarrow 110 \qquad 5 \rightsquigarrow 111 \qquad 6 \rightsquigarrow 101 \qquad 7 \rightsquigarrow 100$$

Any two adjacent numbers (as well as the first and last) differ by only one switch, which means that when counting up from 0 to 7, we can always go directly from k to $k + 1$ without any intermediate states as several switches flip.

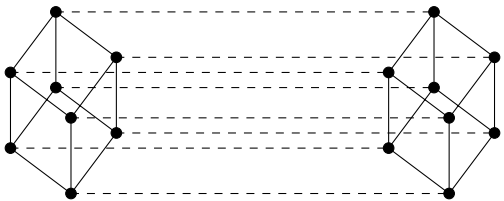
In graph-theoretic terms: the correct graph to look at is the hypercube graph Q_n. Its vertices are just all possible states of n switches, with two vertices adjacent if they only differ in one place.

An n-switch encoding like the one above is a Hamiltonian cycle in Q_n.

The hypercube theorem

Theorem. For all $n \geq 2$, the hypercube graph Q_n is Hamiltonian.

The key property of Q_n at work is that it consists of two copies of Q_{n-1}, with corresponding vertices joined by edges:



This makes us want to do an induction proof, where we take a Hamiltonian cycle in each copy of Q_{n-1}, and then do something to combine them into a single Hamiltonian cycle in Q_n.

Proving the theorem

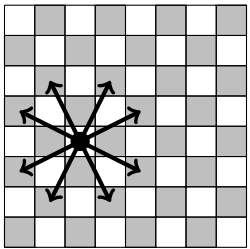
Our base case is $n = 2$. $Q_2 \cong C_4$, so it has a Hamiltonian cycle (which we'll think of as a spanning subgraph isomorphic to a cycle graph).

Assume Q_{n-1} has a Hamiltonian cycle. Draw it in both of the copies of Q_{n-1} sitting inside Q_n:

For a single edge uv of the cycle in Q_{n-1}, and its copy $u'v'$ in the other Q_{n-1}, replace uv and $u'v'$ by uu' and vv'. □

The knight's tour problem

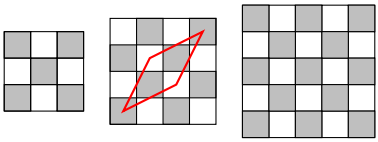
You have an 8×8 chessboard, and a knight: a chess piece that moves by hopping to a square 2 steps away in one direction and one step away in a different direction (as shown below)



The problem is to make 64 moves and return to the start, jumping to each square exactly once: a "knight's tour" or a Hamiltonian cycle in the knight graph. We will not solve this problem.

Smaller chessboards

Instead, here are some smaller chessboards where we can prove that there is no solution!



- For the 3×3: you can never enter or leave the center square.
- For the 4×4: the moves that enter and leave a corner square are forced, but they result in a 4-cycle, not a Hamiltonian cycle.
- For the 5×5: moves alternate between dark and light squares: the knight graph is bipartite. There are 13 dark and 12 light squares, so the first and last square are both dark, and can't be adjacent.

Deleting vertices from cycles

You may think that all of these arguments are very specific to their individual graphs. But in fact, all three arguments we gave are special cases of one general argument.

Lemma. If you delete $k \geq 1$ vertices from a cycle graph C_n, you are left with at most k connected components.

Proof. The first vertex we delete leaves P_{n-1}, which is still connected. After that:

Each vertex v_i has two neighbors: v_{i-1} and v_{i+1}. So if we delete v_i, its connected component is split into at most two pieces: one containing v_{i-1} and one containing v_{i+1}.

So with each vertex we delete after the first, the number of components goes up by at most 1. □

Tough graphs

We call a graph G **tough** if deleting $k \geq 1$ vertices from G leaves at most k connected components. (We just proved that C_n is tough.)

Theorem. If a graph is Hamiltonian, it must be tough.

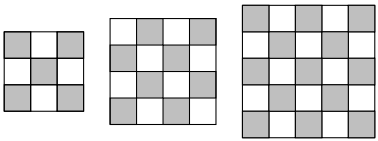
Proof. An n-vertex Hamiltonian graph contains an n-vertex cycle. So we can think of it as C_n with some more edges added.

Adding more edges to a tough graph can't stop it from being tough: it only makes it easier for things to be connected! □

Warning: The converse is not true. Many tough graphs are not Hamiltonian. (For example, the Petersen graph.)

Smaller chessboards, revisited

All of our small chessboards correspond to graphs that aren't tough.



- For the 3×3: delete a corner vertex, and you have 2 components.
- For the 4×4: delete the four center vertices, and you have 6 components.
- For the 5×5: delete the 12 light squares, and you have 13 components.

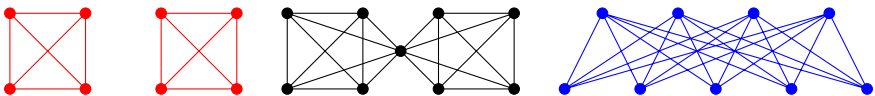
Dense graphs that aren't Hamiltonian

If you want lots of edges in a graph that isn't Hamiltonian, what can you do?

You can have two very large connected components.

You can even have them share a vertex; the result still isn't tough, so it's not Hamiltonian.

You can also have a bipartite graph with unbalanced parts.



In all of these, the typical vertex is adjacent to just under **half** of the other vertices.

Dirac's theorem

Theorem (Dirac). If G is an n-vertex graph with $\delta(G) \geq \frac{n}{2}$, then G is Hamiltonian.

Proof strategy. First, observe that G must be connected: any two vertices must either be adjacent, or share a neighbor.

We'll start with a length-0 path, and build it up into a Hamiltonian cycle, in the following steps:

1. If we can extend our path from either end, do so.
2. If neither endpoint has any neighbors on the path, turn the path into a cycle. (How? See next slide.)
3. If we have a cycle that does not include all n vertices, turn it into a longer path.

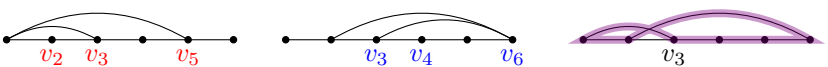
Turning paths into cycles

Lemma. Suppose G has a path $(v_1, v_2, \ldots, v_k)$, and v_1 and v_k each have at least $\frac{k}{2}$ neighbors on the path. Then there is a cycle containing the vertices $\{v_1, v_2, \ldots, v_k\}$.

Proof. Let $S \subseteq \{v_2, \ldots, v_k\}$ (red) be the set of neighbors of v_1.

Let $T \subseteq \{v_2, \ldots, v_k\}$ (blue) be the set of vertices **after** neighbors of v_k.

We have $|S| \geq \frac{k}{2}$ and $|T| \geq \frac{k}{2}$, but they're chosen from a total of $k-1$ vertices, so there is some $v_i \in S \cap T$.



Then $(v_1, v_2, \ldots, v_{i-1}, v_k, v_{k-1}, \ldots, v_i, v_1)$ is a cycle. □

Finding a Hamiltonian cycle

Now we are ready to prove Dirac's theorem.

1. Start with any path and extend it from both ends until you can't anymore.
2. Then we have a path $(v_1, v_2, \ldots, v_k)$. Because $\delta(G) \geq \frac{n}{2} \geq \frac{k}{2}$, v_1 and v_k each have at least $\frac{k}{2}$ neighbors on the path.

   So we can apply the lemma to turn this path into a cycle.
3. If we have a cycle $(v_1, v_2, \ldots, v_k, v_1)$ with $k < n$, then because the graph is connected, some v_i has a neighbor w not on the cycle.

   This gives us a longer path: $(v_{i+1}, v_{i+2}, \ldots, v_k, v_1, v_2, \ldots, v_i, w)$.

These steps let us keep going until we have a Hamiltonian cycle. □

Bondy–Chvátal lemma

I like to call this the "The-magic-was-inside-you-all-along lemma."

Lemma (Bondy–Chvátal). Let G be an n-vertex graph and let u, v be two vertices of G with no edge uv, and with $\deg(u) + \deg(v) \geq n$.

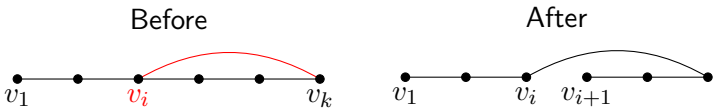
If $G + uv$ is Hamiltonian, then so is G.

Our plan for today:

1. Prove this lemma. (The proof is similar to the key step in Dirac's theorem.)
2. Deduce Dirac's theorem as a corollary.
3. See what else we can prove by using this theorem.

Pósa rotations

A "Pósa rotation" is an operation you can do on paths in a graph.



Formally, suppose $P = (v_1, v_2, v_3, \ldots, v_k)$ is a path and v_i is a vertex in the middle of the path adjacent to v_k.

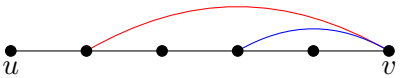
Then the path $P' = (v_1, v_2, v_3, \ldots, v_i, v_k, v_{k-1}, \ldots, v_{i+1})$ is the result of the Pósa rotation. "We rotated P around $v_k v_i$."

This always gives a path of the same length. If v_k has d neighbors on P, we can get d different paths this way. (One of v_k's neighbors is v_{k-1}, and rotating P around $v_k v_{k-1}$ gives back P.)

Proof of Bondy–Chvátal, step I

We assume $u, v \in V(G)$, $uv \notin E(G)$, $\deg(u) + \deg(v) \geq n$, and $G + uv$ has a Hamiltonian cycle.

If that cycle doesn't contain edge uv, it survives in G. But even if the cycle does contain uv, G still has a Hamiltonian $u - v$ path.



Since v has $\deg(v)$ neighbors on the path, we can do Pósa rotations to get $\deg(v)$ different Hamiltonian paths starting at u.

Proof of Bondy–Chvátal, step II

We have $\deg(v)$ different Hamiltonian paths starting at u.

Since $\deg(u) + \deg(v) \geq n$, we have at least $n - \deg(u)$ such paths.

We get a Hamiltonian cycle if any one of these paths has an edge leading back to u:

One of these edges **must** exist! Among the $n - 1$ vertices other than u, $\deg(u)$ are adjacent to u, and $n - \deg(u)$ are the endpoints of one of these paths. There has to be overlap. □

Bondy–Chvátal closure

We now know that the operation "find two vertices u, v with $uv \notin E(G)$ and $\deg(u) + \deg(v) \geq n$" doesn't change the "Hamiltonicity" of a graph.

- If $G + uv$ is Hamiltonian, so is G, by the lemma.
- If $G + uv$ is not Hamiltonian, neither is G.

The **Bondy–Chvátal closure** of a G is the graph H we get by repeating this operation as many times as possible, until all pairs u, v with $\deg(u) + \deg(v) \geq n$ in H are already adjacent.

If H is Hamiltonian, so is G. In particular, if H is the complete graph K_n, then G is Hamiltonian.

Conditions for Hamiltonian graphs

Starting from an n-vertex graph G, repeatedly add an edge whenever you see two non-adjacent u, v with $\deg(u) + \deg(v) \geq n$. The result is the **closure** of G.

Theorem (Bondy–Chvátal). If G's closure is Hamiltonian, so is G.

How do we use this? Here's an example.

Corollary (Dirac's theorem). If G has $n \geq 3$ vertices and $\delta(G) \geq \frac{n}{2}$, then G is Hamiltonian.

Proof. If u, v are **any** two non-adjacent vertices in G, then $\deg(u) + \deg(v) \geq \frac{n}{2} + \frac{n}{2} = n$, so uv is an edge in the closure of G. Therefore the closure of G is K_n.

K_n is Hamiltonian, so G is Hamiltonian. □

Other theorems

Theorem (Ore). If G has $n \geq 3$ vertices and $\deg(u) + \deg(v) \geq n$ whenever u, v are not adjacent in G, then G is Hamiltonian.

Proof. In this case, too, the closure of G is K_n. □

Suppose we write down the degree sequence of G in increasing order: $d_1 \leq d_2 \leq \cdots \leq d_n$.

Theorem (Pósa). If $d_j > j$ for every $j \leq n/2$, then G is Hamiltonian. *(We'll prove this on the next slide.)*

Theorem (Chvátal). If, for every $j \leq n/2$, either $d_j > j$ or $d_{n-j} \geq n - j$, then G is Hamiltonian.

(Chvátal's theorem is the end of the line: the strongest degree condition that guarantees that a graph is Hamiltonian.)

Pósa's theorem

Suppose the degrees of G satisfy $d_1 > 1, d_2 > 2, \ldots, d_{n/2} > n/2$.

Let G's closure H have the degree sequence $d'_1 \leq d'_2 \leq \cdots \leq d'_n$. It also satisfies $d'_1 > 1, d'_2 > 2, \ldots, d'_{n/2} > n/2$.

We want to prove $H = K_n$. Suppose not: suppose $v_i v_j \notin E(H)$. Actually, take the pair i, j where $i < j$ and $i + j$ is as large as possible.

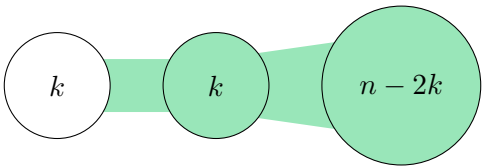
We have $d'_i + d'_j \leq n - 1$, otherwise $v_i v_j$ would be in the closure. Therefore $d'_i \leq \frac{n-1}{2}$. Therefore $i \leq \frac{n-1}{2}$. Therefore $d'_i > i$. Finally,

$$d'_j < n - 1 - i.$$

However, because $i + j$ is as large as possible for a non-edge, v_j must be adjacent to all of $v_{i+1}, v_{i+2}, \ldots, v_n$. That's $n - i$ vertices; even if one of them is v_j itself, we get $d'_j \geq n - 1 - i$. Contradiction! □

An extremal example

Let G be the graph below:



That is, $V(G) = X \cup Y \cup Z$ with $|X| = |Y| = k$ and $|Z| = n - 2k$. It has all edges between X and Y, between Y and Z, in Y, and in Z.

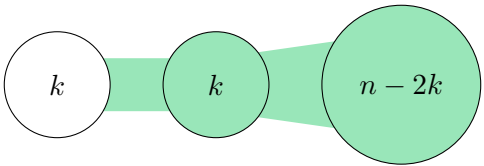
This just barely fails Pósa's theorem: it has degree sequence

$$\underbrace{k, k, \ldots, k}_{k \text{ times}}, \underbrace{n-k-1, \ldots, n-k-1}_{n-2k \text{ times}}, \underbrace{n-1, n-1, \ldots, n-1}_{k \text{ times}}.$$

Is it Hamiltonian?

An extremal example

Let G be the graph below:



That is, $V(G) = X \cup Y \cup Z$ with $|X| = |Y| = k$ and $|Z| = n - 2k$. It has all edges between X and Y, between Y and Z, in Y, and in Z.

Is this graph Hamiltonian?

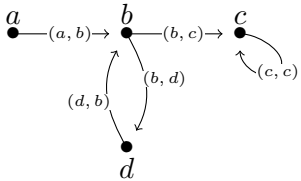
No, because it is not tough. Delete the k vertices in Y. This leaves $k + 1$ components: each vertex of X becomes an isolated vertex, and all of Z is a single connected component.

Directed graphs

Formally, a directed graph (or digraph) has a set of vertices V and a set of **directed edges** or **arcs**: a subset $E \subseteq V \times V$.

We think of an arc $(v, w) \in E$ as an edge pointing from v to w.

In a diagram, we represent these by arrows:



The arc (c, c) represents a loop from c back to c. The arcs (b, d) and (d, b) can both be present at the same time.

What stays the same

Lots of things we've done for graphs are almost the same for digraphs.

- Walks, paths and cycles are very similar. In a directed graph, we want a walk $(v_1, v_2, \ldots, v_k)$ to have arcs (v_i, v_{i+1}).
- As before, we can define the distance $d(u, v)$ as the length of a shortest $u - v$ path. (Your textbook writes $\vec{d}(u, v)$.)
- We can define subgraphs of directed graphs, and ask whether two graphs are isomorphic, in the same way as for graphs.
- Menger's theorem holds, with $\kappa(s, t)$ counting vertices that must be deleted to destroy all directed $s - t$ paths.
- Directed graphs can be Hamiltonian or Eulerian, and we will look at the conditions for these things happening.

What changes

Keep an eye out for the following changes.

Vertex degrees behave differently; we'll see how in a moment.

Connectedness is much more complicated, since an arc that lets us get from u to v may not let us go from v to u. As a result:

- We **could** define trees in directed graphs, but they become much less significant, and we won't cover them in this class.
- Vertex and edge connectivity are also problematic to define. (Menger's theorem is an exception, because in the directed case it specifically asks about ways to get from s to t.)

Often, when a concept doesn't make sense for a digraph, we look at the **underlying graph**, which "forgets" arc directions.

Indegrees and outdegrees

In a directed graph, a vertex v doesn't just have a degree. It has:

- An **indegree**: the number of arcs of the form (u, v). This is written $\deg^-(v)$ by most sources; your textbook writes $\mathrm{id}(v)$.
- An **outdegree**: the number of arcs of the form (v, w). This is written $\deg^+(v)$ by most sources; your textbook writes $\mathrm{od}(v)$.

A version of the degree sum formula holds. For any digraph G,

$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v).$$

We can prove this by induction on arcs: with no arcs, we get $0 = 0$, and each added arc increases both sides of the equation by 1.

Weak and strong connectivity

To resolve the problem of defining connected components in directed graphs, there are three solutions.

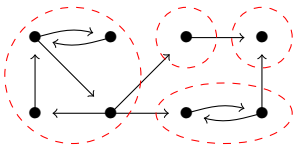
1. A directed graph is **weakly connected** if the underlying graph is connected. This is sometimes useful, but often it's just giving up.
2. A directed graph is **rooted** if we can find some vertex s such that there is an $s - t$ (directed) path for every t.

   (I mention this definition for completeness, but there's not much to say about it.)
3. A directed graph is **strongly connected** or just **strong** if there is a (directed) path from any vertex to any other vertex.

Strong(-ly connected) components

Fact. The relation "$v \sim w$ if there is both a path from v to w and a path from w to v" is an equivalence relation on the vertices of a directed graph.

This fact lets us separate a digraph into **strong components**, such that any vertex can get to any other vertex in the same strong component.

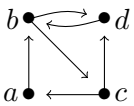


Here are the strong components of the digraph above. Note that there can be arcs between strong components!

Spanning closed walks

An undirected graph is connected iff it has a spanning tree. The spanning tree is the least you can do to get connectivity.

The analogous object for a strong digraph is a **spanning closed walk**. For example, take the strong digraph below:



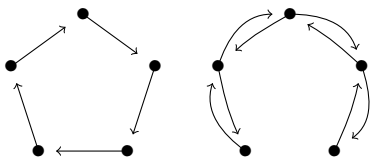
The walk (a, b, d, b, c, a) is a spanning closed walk. ("Spanning" is an abuse of terminology; it's a closed walk that visits each vertex.)

To get from any v vertex to any other vertex w, follow the closed walk, starting at v, until you reach w.

Minimal strong components

In the case of undirected graphs, spanning trees—the minimal connected graphs—always have $n-1$ edges (when they have n vertices). What about directed graphs?

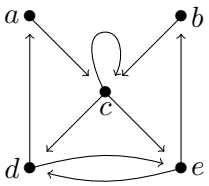
The easiest way to have a spanning closed walk is to have a cycle. This gives us n vertices and n arcs.



At the other extreme, we could have $2(n-1)$ arcs: 2 arcs for every edge of an undirected path. Each arc is needed to get a strong graph!

Eulerian digraphs

A digraph is Eulerian if it has an Eulerian tour: a closed walk that uses each arc exactly once.



In the digraph above, $(a, c, c, d, e, b, c, e, d, a)$ is an Eulerian tour.

Just like undirected graphs, we can prove that a digraph is Eulerian if and only if satisfies the right degree condition and the right connectivity condition.

Necessary conditions

By analogy with the undirected case, its seems that we want the following.

1. All Eulerian digraphs must be strong. (Except possibly for vertices with indegree and outdegree 0.)

   Why? Because an Eulerian tour is a spanning closed walk.

2. Every vertex v in an Eulerian digraph must satisfy $\deg^-(v) = \deg^+(v)$.

   Why? Because if we use each arc exactly once, we enter v $\deg^-(v)$ times, and leave it $\deg^+(v)$ times.

Next: are these conditions also sufficient?

Cycle-finding lemma

Lemma. If $\deg^+(v) \geq 1$ for every vertex v in a digraph, then the digraph contains a cycle.

Proof. Once again, our strategy is to imitate what we've done for undirected graphs.

Pick any starting vertex, and begin a walk from that vertex in the digraph, stopping when we see a vertex for the second time.

The condition that $\deg^+(v) \geq 1$ for all v ensures that if we enter a vertex for the first time, we can leave it and continue the walk.

Once we see a vertex twice, we have a cycle starting and ending at that vertex. □

Sufficient conditions

Theorem. Suppose that G is a digraph such that $\deg^-(v) = \deg^+(v)$ for every vertex v. Then:

1. G has a cycle decomposition.
2. If, additionally, G is weakly connected (maybe plus some isolated vertices), then G is Eulerian.

Proof. Once again, we do the same thing as before! Things to check:

First, that removing a cycle from a digraph preserves the $\deg^- = \deg^+$ condition.

Second, that we can glue together closed walks in a digraph exactly when we can do it in an undirected graph. □

Weakly or strongly connected?

We proved a necessary condition that all Eulerian graphs must be strongly connected.

But we also proved a theorem saying that being weakly connected is enough. How does that make sense?

The answer is: if $\deg^-(v) = \deg^+(v)$ for every vertex v, then all weakly connected graphs are automatically strongly connected!

We could give this a separate proof. But actually, what we've just done is a proof of this statement, via the fact that all such graphs are Eulerian.

De Bruijn sequences

The sequence

AAABAACABBABCACBACCBBBCBCCCAA

has a special property. It has 29 letters, so you can take 27 blocks of 3 consecutive letters (AAA, AAB, ABA, and so on through CCC).

That's true of any 29-letter sequence, but in this one, the 27 blocks are all different, and include each of the 27 possible blocks exactly once.

Such a sequence is called a **de Bruijn sequence**. Specifically, it is the de Bruijn sequence for alphabet $\{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$ and a block size of 3.

Does a de Bruijn sequence exist for any alphabet and block size? And how can we find it?

From sequences to graphs

We could phrase this problem as a graph problem in two ways.

1. Take a directed graph whose vertices are all the possible blocks (AAA, AAB, AAC, and so on).

   Add an arc from each block to all the possible blocks that could follow it: for example, ABC will have arcs to BCA, BCB, and BCC.

   Find a Hamiltonian path in this graph.

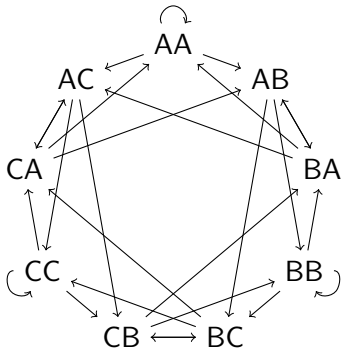
2. Define the same graph, but with blocks that are one letter shorter: AA, AB, AC, BA, BB, BC, CA, CB, CC.

   The 3-letter blocks we want to see are the **arcs** of this graph: for example, ABC corresponds to the arc $(\mathrm{AB}, \mathrm{BC})$.

   So it's enough to find an Eulerian walk in this graph, instead.

An example of a de Bruijn graph

Here's what this digraph looks like for our special case:



The sequence "AAABAACABBABCACBACCBBBCBCCCAA" should correspond to an Eulerian tour.

Existence

Theorem. For any k-letter alphabet and any $n \geq 1$, a de Bruijn sequence for blocks of size n exists.

Proof. We check the conditions for the digraph we found to be Eulerian.

It is strongly connected: we can change any $(n-1)$-letter block $x_1x_2\ldots x_{n-1}$ to any other in $n-1$ steps.

Each vertex has outdegree k: from $x_1x_2\ldots x_{n-1}$, we can go to $x_2x_3\ldots x_{n-1}y$ for any letter y in the alphabet.

Each vertex has indegree k: we can get to $x_1x_2\ldots x_{n-1}$ from a vertex $yx_1x_2\ldots x_{n-2}$ for any letter y in the alphabet.

So $\deg^+(v) = \deg^-(v)$ for all vertices v. □

The least strong digraphs

We've already seen strongly connected digraphs: ones in which there is a path from any vertex v to any vertex w. Other directed graphs decompose into strongly connected components.

A **directed acyclic graph** or **dag** is the polar opposite of this. It is a directed graph with no cycles. This means that for any two vertices v, w, there might be a path from v to w, or a path from w to v, or maybe neither—but never both.

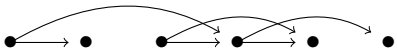
What are these good for?

- In applications, a list of prerequisites will (hopefully!) be a dag.
- They will help us describe the structure of digraphs in general.

An example of a dag

Topological sorting

Theorem. We can place the vertices of any dag G in a line such that all arcs point from left to right.



Proof. Induct on the number of vertices of G.

We know that if $\deg^+(v) \geq 1$ for every vertex v, then G contains a directed cycle. Since we know G **does not** have a directed cycle, there must be a vertex v with $\deg^+(v) = 0$.

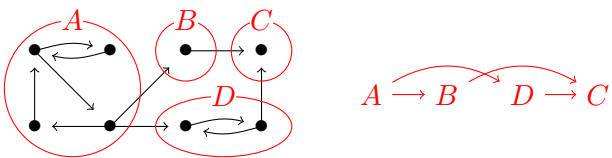
By induction, order the vertices of $G - v$ to satisfy the theorem. Then, put v at the right end: all arcs involving v will point from left to right, because all of them point into v. □

General structure of a digraph

Strong graphs and dags combine to describe the connectivity of any directed graph.

Fact. If we replace each strongly connected component of a digraph by a single vertex, the connections between these vertices form a directed acyclic graph.

Here is a graph whose strongly connected components we've found:



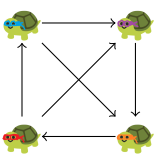
On the right is the digraph we get by replacing its strongly connected components by vertices.

Tournaments

A **tournament** is an orientation of a complete graph. That is, for every pair of vertices $\{v, w\}$, it has exactly one of the arcs (v, w) or (w, v).

Take an actual, non-graph tournament. Suppose it is a "round-robin" tournament (everyone plays against everyone else) and there are no draws.

	L	D	R	M
Leonardo	—	Win	Loss	Win
Donatello		—	Loss	Win
Raphael			—	Loss
Michaelangelo				—



A directed graph whose vertices are the contestants, with an arc (v, w) whenever v beats w, is a tournament in the graph theory sense.

Examples

Up to isomorphism, there are only two 3-vertex tournaments:

The second example is one that exists for any number of vertices: the **transitive tournament**. This has vertices $\{v_1, v_2, \ldots, v_n\}$ with an arc (v_i, v_j) whenever $i < j$, and is the only acyclic tournament.

The number of possible tournaments grows very quickly with the number of vertices: it begins

$$1, 1, 1, 2, 4, 12, 56, 456, 6880, 191536 \ldots$$

Score sequences

Every graph has a degree sequence. For directed graphs, things become more complicated, since each vertex v has an indegree $\deg^-(v)$ and an outdegree $\deg^+(v)$.

For tournaments, things become simple again: in an n-vertex tournament, $\deg^-(v) + \deg^+(v) = n - 1$ for all vertices v. So we can describe the degrees of the vertices by a **score sequence**: the sequence of all outdegrees.

Which score sequences are possible?

- One requirement is that all n numbers must add up to $\binom{n}{2} = \frac{n(n-1)}{2}$.
- But there's more. $1, 1, 1, 1, 5, 6, 6$ is not a possible score sequence, even though $1 + 1 + 1 + 1 + 5 + 6 + 6 = \binom{7}{2}$. (Why not?)

Characterizing score sequences

If a tournament's score sequence is sorted as $p_1 \leq p_2 \leq \cdots \leq p_n$, we have two necessary conditions:

$$p_1 + p_2 + \cdots + p_k \geq \binom{k}{2} \qquad k = 1, \ldots, n-1$$

$$p_1 + p_2 + \cdots + p_n = \binom{n}{2}$$

Theorem (Landau). These conditions are also sufficient.

Proof sketch. We can induct on n, and check that the conditions still hold for the first $n-1$ vertices if we assume that the n^{th} vertex beats the p_n vertices with the **highest** score. □

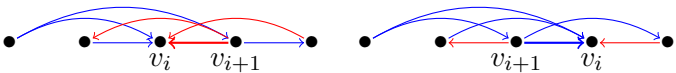
Rédei's theorem

One of the nice features of tournaments is that they give us one of the simplest Hamiltonian-path theorems to state.

Theorem (Rédei). Every tournament has a Hamiltonian path.

Proof. Choose an ordering of the vertices $v_1, v_2, \ldots, v_n$ that maximizes the number of arcs (v_i, v_j) with $i < j$.

Then the arc between each v_i and v_{i+1} must be oriented (v_i, v_{i+1}), or else we can swap v_i with v_{i+1} and increase the number of forward arcs. (No other arc changes direction when we do that.)



Therefore $(v_1, v_2, \ldots, v_n)$ is a Hamiltonian path. □

Camion's theorem

Many tournaments are not Hamiltonian, because they do not have Hamiltonian **cycles**. The transitive tournament, for example, has no cycles at all.

Theorem (Camion). Every strongly connected tournament is Hamiltonian.

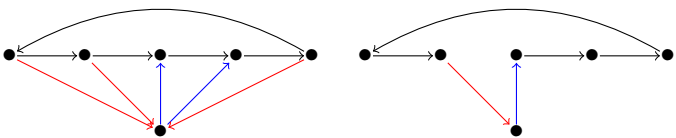
Proof outline. Because the tournament is strongly connected, it has a closed walk. Following that walk until it revisits a vertex creates a cycle.

We will then work on making that cycle bigger until it's Hamiltonian...

Camion's theorem proof: case 1

We have a cycle $C = (v_1, v_2, \ldots, v_k, v_1)$.

Case 1. Suppose there is a vertex w with arcs both **to** and **from** vertices in C.



Find a place where the direction switches: v_i has an arc **to** w, and v_{i+1} has an arc **from** w.

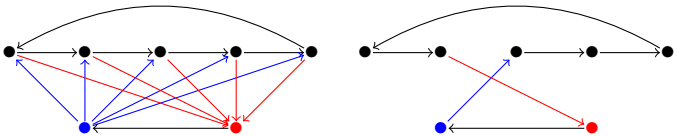
This is a place where we can insert w into the cycle, getting $(v_1, v_2, \ldots, v_i, w, w_{i+1}, \ldots, v_k, v_1)$.

Camion's theorem proof: case 2

Case 2. All vertices not on C can be split into a set A with only arcs **to** C, and a set B with only arcs **from** C.

Note: the tournament is strong, so $|A|, |B| > 0$.

Not **all** arcs between A and the rest of the graph can be pointed out of A. So there must be an arc (b, a) with $a \in A$, $b \in B$.



We can insert b and a between any two vertices on the cycle, extending it to $(v_1, v_2, \ldots, v_i, b, a, v_{i+1}, \ldots, v_k, v_1)$. □

Generalizations

More is true:

1. We can show that every strong tournament has a cycle of length 3. (The argument is similar to Case 2 of the proof.)
2. In Case 1, the cycle's length increases by 1.

   In Case 2, the length increases by 2, but we can modify the argument to make it less efficient: to make the length only go up by 1, again.
3. So if we start the induction with a cycle of length 3, and increase by length 1 at each step, we get:

Theorem (Moon.) Every strong n-vertex tournament has a cycle of every length from 3 to n.

Kings in tournaments

A **king** in a tournament is a vertex u with the following property. For every vertex w, either:

- there is an arc (u, w), or
- there is a third vertex v with arcs (u, v) and (v, w).

In other words, any vertex can be reached from a king in at most 2 steps.

Theorem. Every tournament has a king.

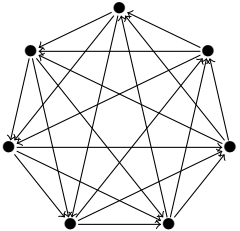
Proof. Homework exercise: if u cannot reach w in at most 2 steps, then $\deg^+(u) < \deg^+(w)$.

So choose u to maximize $\deg^+(u)$. Then $\deg^+(u) < \deg^+(w)$ never holds, so u must be a king. □

Everyone is a king

Kings are not necessarily unique! In fact, for odd n, there is the following construction.

Arrange n vertices in a circle; each vertex has an arc to the $\frac{n-1}{2}$ vertices counterclockwise after it.



Every vertex v in this tournament has $\deg^+(v) = \frac{n-1}{2}$. So every vertex has maximum outdegree, and every vertex is a king.

Regular tournaments

The construction is the previous slide is one example (of many) of a regular tournament.

A **regular tournament** is one in which all vertices have the same outdegree. Since the outdegrees must sum to $\binom{n}{2} = \frac{n(n-1)}{2}$, every outdegree must be $\frac{n-1}{2}$, so these exist only if n is odd.

- Since every vertex is a king, in particular every vertex can reach every other vertex, so all regular tournaments are strongly connected.
- Every Eulerian tour of K_n gives a regular tournament: just orient each edge in the direction that the Eulerian tour visits it.
- For even n, there exist "nearly regular" tournaments in which every vertex is still a king.

Doubly regular tournaments

A **doubly regular** n-vertex tournament is one in which:

1. Every vertex has outdegree $\frac{n-1}{2}$, and
2. For every two vertices u, v, there are exactly $\frac{n-3}{4}$ vertices w with arcs (u, w) and (v, w).

Why $\frac{n-3}{4}$? It's the only possible value if we want this number to be the same for every pair $\{u, v\}$.

For $\frac{n-3}{4}$ to be an integer, n has to be one less than a multiple of 4.

Open research problem. Do doubly regular $(4k-1)$-vertex tournaments exist for all k?