Math 3322: Graph Theory

Chapters 1–4

Mikhail Lavrov (`mlavrov@kennesaw.edu`)

January 11–February 17, 2021

An example problem

Suppose that you decide to go on a tour of the United States, by car. (You'll skip Alaska and Hawaii.)

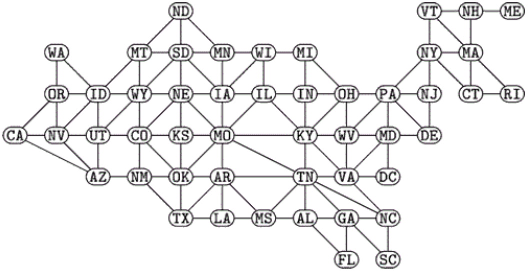
You decide that the perfect tour must visit each of the remaining 48 states **exactly once**. Once you've visited a state you can't come back!

Questions:

1. Is this possible? And is it possible specifically starting in Georgia?
2. What sort of mathematical machinery do we need to talk about questions like this one?

Representing the problem

Rather than a map of the US, it's easier to solve the problem with a diagram like this one:



The locations of the states in the diagram don't matter. We could even represent the problem with a list of adjacencies: "AL is adjacent to FL, GA, TN, MS; AZ is adjacent to CA, NM, NV, UT; . . ."

Definition of a graph

A **graph** is a mathematical object designed to model situations like this one. Formally, a graph G is a pair (V, E) where:

- V is a set of objects called **vertices**;
- E is a set of **edges**; each edge is a pair $\{v, w\}$ of vertices $v, w \in V$ and tells us that v and w are **adjacent**.

  (Sometimes when it's not ambiguous we write vw instead of $\{v, w\}$. This is unordered: wv or $\{w, v\}$ is the same edge.)

It's often convenient to represent a graph by a diagram like the one on the previous slide, where edges are drawn as lines connecting the vertices.

What a graph is not

Here are some things to keep in mind about graphs:

- We can represent a graph by a diagram, but the graph is not the diagram!

  Facts such as "Vertex FL is below vertex GA" or "The edge between MO and TN is the longest" are (usually) irrelevant.

- Extra information such as names of vertices also shouldn't matter.

  Technically we'll get a different object if we rename the vertices from "AL, AZ, AR, CA, ..." to "1, 2, 3, 4, ..." but all properties we care about (such as the existence of a tour) will stay the same.

- We usually forbid multiple edges between the same pair of vertices, or "loop" edges between a vertex and itself.

The Facebook graph

We can represent friendships on Facebook by a graph. The vertices are accounts, with an edge between every pair of Facebook friends.

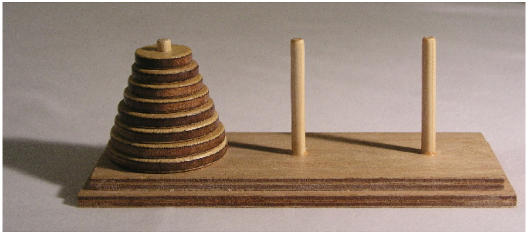
Some graph-theoretic questions about this graph:

- "Six degrees of separation:" can we get from any vertex to any other in six steps between adjacent vertices?
- How cliquish is Facebook? What fraction of a typical person's friends are each other's friends?

If we try to represent Twitter in the same way, we don't get a graph. We get a "directed graph": edges are not symmetric!

The Tower of Hanoi puzzle

In this puzzle, we have 8 disks of different sizes on 3 rods. We're allowed to move the top disk on any rod and put it on a smaller disk.



We can represent this problem by a graph whose vertices are valid states of disks, and whose edges represent moves. "Can we move all the disks to a different rod?" is a question about paths in this graph, which we'll introduce later this week.

Circuit board layouts

A circuit board is a plastic sheet with many components on it, some of which are connected by conductive copper tracks.

The circuit we want to embed on the circuit board is a graph: its vertices are components, and the conductor tracks are edges.

- From the point of view of the graph, it does not matter where the vertices and edges are physically located. (This is how the circuit is often initially designed.)
- But when we print the circuit board, we don't want the conductor tracks to cross!

  This is the problem of finding a **plane embedding** of a graph.

Teaching assignments

The math department faculty have different teaching preferences. I'd be happy teaching graph theory or linear programming, but I'm not prepared for numerical analysis or differential equations.

In the graph representation, people and classes are both vertices. Each person has an edge to every class they're willing to teach.

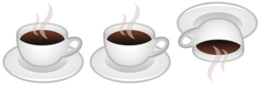
The department wants to decide who teaches which class next semester: to select a subset of the edges that will satisfy constraints.

In the simplest case, when every person teaches one class, and every class is taught once, we are looking for a **matching** in this graph.

The three cup puzzle

You have three cups in a row, upside down.

In one step, you're allowed to flip any two adjacent cups (changing the orientation of both).



Can any number of steps turn all three cups right side up?

The graph model

We have 8 vertices for the possible states of the cups, and we can put edges between states that are one step apart:

$$\begin{tikzcd}
\text{UUU} \arrow[r, dash] \arrow[d, dash] & \text{DDU} \arrow[d, dash] & \text{DDD} \arrow[r, dash] \arrow[d, dash] & \text{UUD} \arrow[d, dash] \\
\text{UDD} \arrow[r, dash] & \text{DUD} & \text{DUU} \arrow[r, dash] & \text{UDU}
\end{tikzcd}$$

Especially if we draw the picture right, it's easy to see that there's no way to get from "DDD" to "UUU": the puzzle has no solution!

Today's goal: make this idea formal… so that we can generalize to other problems.

Walks in a graph

A **walk** in a graph G is a sequence of vertices $(v_0, v_1, v_2, \dots, v_k)$ such that for each $i = 0, 1, 2, \dots, k-1$, $v_i v_{i+1}$ is an edge of G.

It "starts" at v_0 and "ends" at v_k and if we want to emphasize this, we call it a "$v_0 - v_k$ walk".

In our graph on the previous slide, a "UUU – DDD walk" would be a solution to the cup puzzle: a sequence of states, starting at UUU and ending at DDD, so that we can get from each state to the next in one valid step.

(There is no "UUU – DDD walk" in the graph: the puzzle has no solution!)

Walks have lengths

The **length** of the walk $(v_0, v_1, v_2, \ldots, v_k)$ is k: the number of edges used. (In a puzzle like the three cup puzzle, it's the number of steps taken.)

For any vertex v in any graph, there is always a "trivial" $v - v$ walk of length 0: the sequence (v). There might be other $v - v$ walks.

In the three cups puzzle graph,

- there is a $\mathsf{UUU} - \mathsf{UUU}$ walk of length 2: $(\mathsf{UUU}, \mathsf{UDD}, \mathsf{UUU})$.
- there is a $\mathsf{UUU} - \mathsf{UUU}$ walk of length 4: for example, $(\mathsf{UUU}, \mathsf{UDD}, \mathsf{DUD}, \mathsf{DDU}, \mathsf{UUU})$.

There are many more $\mathsf{UUU} - \mathsf{UUU}$ walks of all even lengths!

What we want

The three cups puzzle is easy, but finding walks in graphs comes up in:

- pathfinding in video games (or in Google Maps);
- step-by-step planning task in operations research or robotics;
- solving harder puzzles like Rubik's cubes.

What sort of questions might we have?

- Given a graph and two vertices u and v, does there exist a $u - v$ walk?
- Can we find a $u - v$ walk?
- What is the shortest (minimum length) $u - v$ walk, and can we find it?

What we want connected components to be

The three cups puzzle graph has two halves. We can group the vertices as $V_1 = \{\mathrm{UUU}, \mathrm{UDD}, \mathrm{DUD}, \mathrm{DDU}\}$ and $V_2 = \{\mathrm{DDD}, \mathrm{DUU}, \mathrm{UDU}, \mathrm{UUD}\}$.

There are $u - v$ walks when $u, v \in V_1$ or when $u, v \in V_2$, but not when $u \in V_1$ and $v \in V_2$ or vice versa.

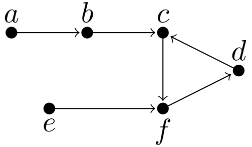
Intuition says that for more complicated graphs, there should always be one or more pieces like this: we can get from any vertex to any other vertex in the same piece, but not to the other pieces.

(Maybe we're Google Maps, and can't give driving directions from America to Europe.)

I will try to convince you that this is Not Obvious and Needs Proof.

Aside: directed graphs

Consider the non-graph example where our edges have directions:



Sometimes you can get from one vertex to another, but you can't get back. (And sometimes you can.)

Sometimes you can't go between two vertices at all, but there's a third vertex they can both reach.

The point is that here, there's no nice division of the vertices into several parts that tells the whole story about connectedness.

Equivalence relations

You may have seen this definition before, but probably nobody has told you what the point is.

If we have a set S and some relation $\sim$ that holds between some elements of S, then $\sim$ is a **equivalence relation** if:

1. it is **reflexive**: for any $x \in S$, $x \sim x$.
2. it is **symmetric**: for any $x, y \in S$, if $x \sim y$, then $y \sim x$.
3. it is **transitive**: for any $x, y, z \in S$, if $x \sim y$ and $y \sim z$, then $x \sim z$.

The whole point of checking these three properties is that they're what we need to know that S has a partition into **equivalence classes**. . .

Equivalence classes

Every equivalence relation $\sim$ on any set S can be summarized as "we give every element of S a label, and $\sim$ is checking if two elements have the same label".

We can also talk about this in terms of equivalence classes. These are sets $S_1, S_2, \ldots, S_k$ such that:

1. every element of S is in exactly one of S_1 or S_2 or $\ldots$ or S_k.
2. for any $x, y \in S$, $x \sim y$ exactly when x and y are in the same S_i.

Disclaimer: actually, there don't have to be finitely many equivalence classes, or even countably many. But in this course, everything will be finite.

Walks and equivalence relations

If u, v are vertices in a graph G, we say "u is connected to v" if there is a $u - v$ walk in G.

Theorem. This is an equivalence relation.

Proof. We check all three conditions:

- It is reflexive: (v) is always a $v - v$ walk for any vertex v.
- It is symmetric: reversing a $u - v$ walk gives a $v - u$ walk.
- It is transitive: joining a $u - v$ walk and a $v - w$ walk together (deleting the duplicate v) gives a $u - w$ walk.

This completes the proof. $\square$

Connected components

This theorem tells us that we can **always** split up the vertices of G into one or more sets $V_1, V_2, \dots, V_k$ (the equivalence classes) such that there is a $u - v$ walk if and only if u and v are in the same V_i.

The **connected components** of G are, depending on who you ask, either the sets $V_1, V_2, \dots, V_k$, or the smaller graphs $G_1, \dots, G_k$ representing what G does on one of those sets.

The graph G is **connected** if there is only one connected component: there is a walk from any vertex to any other.

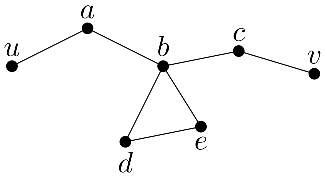
Path and shortest walks

In most graph problems where you want to get from one vertex to another, it would be wasteful to have a $u - v$ walk that comes back to a vertex over and over.

We say that a **path** is a walk in which no vertex is repeated.

Theorem. If there is a $u - v$ walk in a graph G, there is always a $u - v$ path. Moreover, any shortest $u - v$ walk is always a path.

The intuition: we can always simplify walks that aren't paths.



$u - v$ walk: u, a, b, d, e, b, c, v
$u - v$ path: u, a, b, c, v

Distance

Recall: the **length** of a walk $(v_0, v_1, \ldots, v_n)$ is n: the number of edges taken, or one less than the number of vertices.

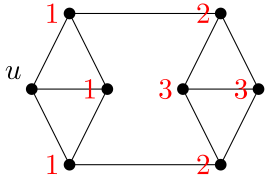
In a graph G with at least one $u - v$ walk, we say that the **distance** $d(u, v)$ is the shortest possible length of a $u - v$ walk. (By the theorem from the previous slide, this is also the shortest possible length of a $u - v$ path.)

Related definitions:

- A $u - v$ **geodesic** is a $u - v$ path of length $d(u, v)$. (There can be several!)
- If G is connected, the **diameter** $\operatorname{diam}(G)$ is the largest distance between any two vertices.

Example of distances

Consider the following graph as an example:



Let's work out the distances from u to all other vertices.

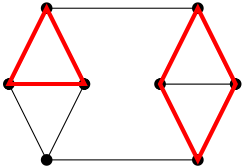
1. All vertices adjacent to u are at distance 1 from u.
2. All unexplored vertices adjacent to them are at distance 2.
3. All unexplored vertices adjacent to those are at distance 3. (And so on, in general.)

Closed walks and cycles

A walk that begins and ends at the same vertex is called a **closed walk**.

(These can be very boring, and look like (a, b, c, d, c, b, a).)

A **cycle** is a closed walk that does not repeat any vertices, except for the start being the same as the end.



Cycles are important for many reasons; for example, because they tell us there's more than one way to get from one vertex to another.

Trails and circuits

The textbook also gives you two more definitions:

- A **trail** is a walk that does not use an edge more than once.

  That is, a walk $(v_0, v_1, \dots, v_n)$ is a trail if the edges $v_0v_1, v_1v_2, \dots, v_{n-1}v_n$ are all distinct.
- A **circuit** is a trail that starts and ends at the same vertex.

These are not usually interesting. We'll encounter them again when we talk about Eulerian trails and Eulerian circuits in a graph, but don't worry about them for now.

What is a subgraph?

A graph H is a **subgraph** of a graph G if:

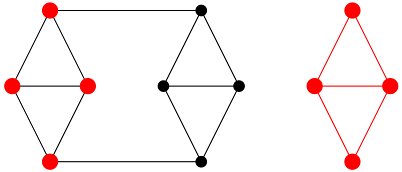
- All vertices of H are also vertices of G ($V(H) \subseteq V(G)$).
- All edges of H are also edges of G ($E(H) \subseteq E(G)$).

Intuitively, we are focusing our attention on just part of G.

By this definition, G itself is a subgraph of G. If H is not all of G (it has fewer vertices, or fewer edges, or both) then we call it a **proper** subgraph.

Induced subgraphs

Pick a graph G and a subset of its vertices $S \subseteq V(G)$. Which subgraphs of G have exactly these vertices?

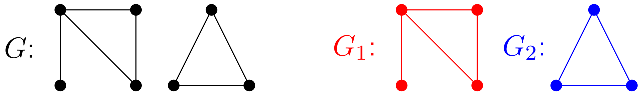


There are many options. The restriction is that only edges of G with **both** endpoints in S can be part of the subgraph.

We say that "the subgraph of G induced by S", denoted $G[S]$, is the subgraph whose vertices are S and whose edges are **all** the edges of G with both endpoints in S.

Connected components as subgraphs

The **connected components** of a graph G are the maximal connected subgraphs of G (connected subgraphs that cannot be made any bigger).

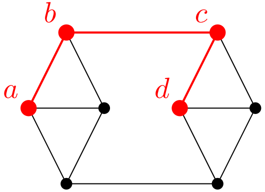


Earlier, we saw that we can split up the vertices of G into subsets $V_1, V_2, \ldots, V_k$ such that there is only a $u - v$ walk when u, v are in the same subset V_i.

The connected components $G_1, G_2, \ldots, G_k$ are exactly the induced subgraphs $G[V_1], G[V_2], \ldots, G[V_k]$.

Paths and cycles as subgraphs

We can think of a path (or cycle) as a subgraph instead of a sequence of vertices.



The $a - d$ path above is either the sequence (a, b, c, d) or the subgraph with vertices $\{a, b, c, d\}$ and edges $\{ab, bc, cd\}$.

There is one difference! The subgraph does not remember direction. In the case of a cycle, it also doesn't remember where we started and ended.

Unpacking statements

Graph theory has a lot of definitions. The first skill to keep in mind when you write a proof is how to unpack the definitions to get to the statement you want to prove.

Some details:

- If you're proving "all X are Y" or "if something is X, then it is Y" then you want to start with an arbitrary X, and prove Y.
- If you're proving "there exists X" then you want to give a way to construct X.
- Keep equivalent definitions in mind. If you are given "u is connected to v" then you can assume there is a $u - v$ path, but if you want to prove "u is connected to v" then it's enough to construct a $u - v$ walk.

Example of unpacking

Consider our earlier proof that “is connected to” is an equivalence relation.

1. First, we unpack “equivalence relation” into “reflexive, symmetric, and transitive”.
2. Let’s just look at one of these: symmetric. We unpack it further: “if there is a $u - v$ walk, then there is a $v - u$ walk.”
3. We are given that there is a $u - v$ walk, so we can start by saying “let $(u = v_0, v_1, \dots, v_k = v)$ be a $u - v$ walk”.
4. Our goal is to prove there is a $v - u$ walk, so we need to use what we have to construct a walk starting at v and ending at u.

Proofs and counterexamples

One of these statements is true, and one is false.

- If a $u - v$ path is a geodesic, then the subgraph of its vertices and edges is induced.
- If the subgraph of a path's vertices and edges is induced, then that path is a geodesic.

How do we prove such a statement, and how do we disprove it?

For a proof, we need to take an arbitrary $u - v$ path in an arbitrary graph. We **assume** that the first property holds, and use it to **prove** the second property.

To disprove such a statement, all we need is one graph with one $u - v$ path where the first property holds, but the second does not.

Geodesics and induced subgraphs: a proof

Let $(v_0, v_1, \ldots, v_k)$ be a geodesic. We want to show that the subgraph of its vertices and edges is induced: the edges $\{v_0v_1, v_1v_2, \ldots, v_{k-1}v_k\}$ are **all** the edges in G between $\{v_0, v_1, v_2, \ldots, v_k\}$.

Neither "there is no shorter path" and "these are all the edges" let us make any assumptions. Let's do a proof by contradiction so that we can make assumptions!

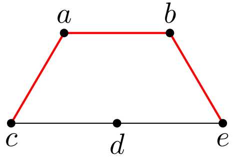
Assume the subgraph is **not induced**: there is another edge v_iv_j that is not one of the edges of the path.

Without loss of generality, $i < j$. "We could write the same exact proof in the case $i > j$." Then $(v_0, v_1, \ldots, v_i, v_j, \ldots, v_k)$ is a shorter $v_0 - v_k$ path: contradiction!

Geodesics and induced subgraphs: a counterexample

To disprove “If the subgraph of a path’s vertices and edges is induced, then that path is a geodesic,” all we need is. . .

. . . **one** graph and a path in it such that the subgraph is induced, but the path is not a geodesic. Here’s one example:



Here, (c, a, b, e) is a $c - e$ path and the path subgraph with vertices c, a, b, e and edges ca, ab, be is induced.

But it’s not a geodesic: (c, d, e) is a shorter $c - e$ path!

Walks and paths: two approaches

Theorem. If there is a $u - v$ walk, there is a $u - v$ path of at most the same length.

We have an intuition that we can "shorten" walks to eliminate redundancy, and try to get a path.

There are two ways to make this intuition rigorous:

1. The extremal principle. Pick the $u - v$ walk of shortest length, and prove that it must be a path.
2. Algorithmic proof. Given a $u - v$ walk, show how to construct the $u - v$ path we get when we eliminate all loops.

The extremal principle

Let $(u = v_0, v_1, \ldots, v_k = v)$ be the $u - v$ walk of shortest length.

Our assumption "there is no shorter walk" and goal "there are no repeated vertices" don't give us anything to work with. It makes sense to proceed by contradiction: suppose that there is a repeated vertex, and prove that there is a shorter walk.

(The extremal principle often leads to proofs by contradiction.)

Suppose that $v_i = v_j$ for some $i < j$. Then

$$(v_0, v_1, \ldots, v_i, v_{j+1}, v_{j+2}, \ldots, v_k)$$

is a $u - v$ walk of length $k - (i - j) < k$. This contradicts our assumption that we took the shortest walk! Therefore there cannot be repeated vertices. $\square$

Algorithmic proof

Given a $u - v$ walk $(u = v_0, v_1, \ldots, v_k = v)$, let's try to construct a $u - v$ path. Our intuition "if there's a loop, remove it" doesn't **immediately** give a path; it must be repeated. We'd like a recipe that immediately gives the path.

Key insight: after visiting a vertex $w = v_i$, go to the vertex after **the last** occurrence of w in the walk. We must prove:

1. The result is still a walk. That's because we go from w to a vertex that follows w in the original walk, so there's still an edge there.
2. The result is not longer. That's because we only skip ahead.
3. The result is a path. We can't see w twice, because the first time we see it, we skip past all other instances of w. $\square$

Challenge question

Lemma. If a graph contains a **closed walk** of odd length, then it contains a **cycle** of odd length.

We will prove this on Friday! If you want a challenge in the meantime, think about how you'd adapt what we've done just now to prove this lemma.

Optimization problems

Many properties of graphs have optimization problems built into them: they're the "minimum" or "maximum" value of something.

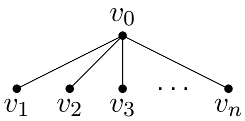
We've already seen one: $\mathrm{diam}(G)$ is the largest possible value of $d(u, v)$ for $u, v \in V(G)$. (In fact, $d(u, v)$ is the smallest possible length of a path, so there's nested optimization!)

Suppose we want to prove that a graph G has diameter k. What do we need to do?

1. Prove that $\mathrm{diam}(G) \leq k$: show that there is a path of length at most k between any two vertices.
2. Prove that $\mathrm{diam}(G) \geq k$: give an example of two vertices u, v for which $d(u, v) \geq k$, and prove that there is no shorter path.

Diameter example

Theorem. The graph with vertices $v_0, v_1, v_2, \ldots, v_n$ and an edge v_0v_i for $i = 1, \ldots, n$ has diameter 2 when $n \geq 2$.



Proof. First, we prove that $\operatorname{diam}(G) \leq 2$.

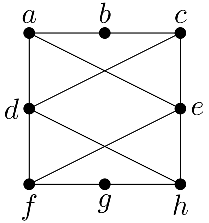
- There's a $v_0 - v_i$ path of length 1 for any $i \neq 0$: (v_0, v_i).
- There's a $v_i - v_j$ path of length 2 for any $i, j \neq 0$: (v_i, v_0, v_j).

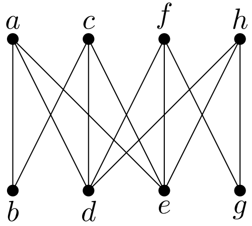
Second, we prove $\operatorname{diam}(G) \geq 2$. Well, $d(v_1, v_2) > 1$ because a path of length 1 would require an edge v_1v_2. So $\operatorname{diam}(G) \geq d(v_1, v_2) \geq 2$. □

Bipartite graphs

A graph G is **bipartite** there is a partition $V(G) = A \cup B$ (with $A \cap B = 0$) such that all edges of G have one endpoint in A and one endpoint in B. The pair (A, B) is called the **bipartition**.

Example: the graph below is a bipartite graph. . .





. . . because we can set $A = \{a, c, f, h\}$ and $B = \{b, d, e, g\}$.

Examples of bipartite graphs

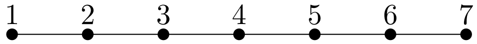
There are two ways a graph can end up being bipartite.

- Graphs that are about connections between two types of vertices.

  Example: the graph whose vertices are teachers and classes, with an edge between a each teacher and each class they're willing to teach.

- Graphs that just happen to be bipartite because of their structure.

  Example: a path graph is bipartite because all edges join an even-numbered vertex to an odd-numbered vertex.



Characterizing bipartite graphs

Theorem. The following are equivalent for a graph G:

1. G is bipartite.
2. G has no closed walks of odd length.
3. G has no cycles of odd length.

There's lots of flexibility in proving "the following are equivalent" (TFAE) statements. We just have to make sure there's a chain of implications from any statement to any other.

Here, $2 \implies 3$ because all cycles are closed walks. $1 \implies 2$ is the motivation for this theorem: if every edge goes A to B or B to A, a walk that starts at A must take an even number of steps to return to A.

We will prove $3 \implies 2$ and $2 \implies 1$.

Proving $2 \implies 1$

We will prove: "if G has no closed walks of odd length, it is bipartite".

To prove a graph is bipartite, construct a bipartition! The idea is:

1. Pick a vertex u; let $A = \{v : d(u, v) \text{ is even}\}$ and $B = \{v : d(u, v) \text{ is odd}\}$.
2. If the graph is not connected, do this in each component.

If vw is an edge and v, w are in the same part, then take a $u - v$ geodesic, edge vw, and a $w - u$ geodesic. This is a closed walk of odd length.

- If $v, w \in A$: length is even $+ 1 +$ even $=$ odd.
- If $v, w \in B$: length is odd $+ 1 +$ odd $=$ odd.

Proving $3 \implies 2$

We will prove: "if G has a closed walk of odd length, it has a cycle of odd length".

Let's use **the extremal principle**, and take the shortest closed walk of odd length

$$(v = v_0, v_1, v_2, \dots, v_k = v).$$

Is it possible for this not to be a cycle?

Suppose $v_i = v_j$ with $i < j$.

- If $j - i$ is even, then we get a shorter closed walk of odd length: $(v = v_0, v_1, \dots, v_i, v_j, \dots, v_k = v)$. Contradiction!
- If $j - i$ is odd, then we get a shorter closed walk of odd length: $(v_i, v_{i+1}, \dots, v_j)$. Contradiction! $\square$

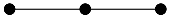
Some graphs that show up over and over

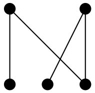
Here are some common graphs that we've given simple names:

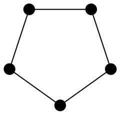
- The **path graph** P_n has vertices $\{v_1, v_2, \dots, v_n\}$ and edges $\{v_1v_2, v_2v_3, \dots, v_{n-1}v_n\}$.
- The **cycle graph** C_n has vertices $\{v_1, v_2, \dots, v_n\}$ and edges $\{v_1v_2, v_2v_3, \dots, v_{n-1}v_n, v_nv_1\}$.
- The **complete graph** K_n has vertices $\{v_1, v_2, \dots, v_n\}$ and all possible edges v_iv_j.
- The **complete bipartite graph** $K_{s,t}$ has vertices $\{v_1, \dots, v_s, w_1, \dots, w_t\}$ and all edges of the form v_iw_j.

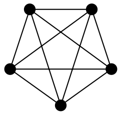
If another graph (especially a subgraph) has the same structure but different vertex names, we call it "a copy of P_n", etc.

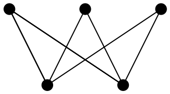
Some pictures

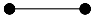

P_3

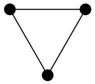

P_5

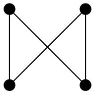

C_5


K_5


$K_{2,3} \cong K_{3,2}$


$P_2 \cong K_2 \cong K_{1,1}$


$C_3 \cong K_3$

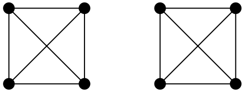

$C_4 \cong K_{2,2}$

Graph union

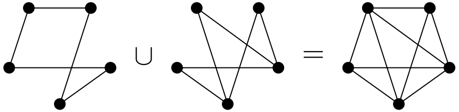
We can define new graphs from existing graphs. One of the simplest operations is the **graph union** $G \cup H$ of two graphs G and H.

- Often, we assume $V(G) \cap V(H) = \varnothing$ and just combine them.

$K_4 \cup K_4$ or $2K_4$:



- We can also define $G \cup H$ as the graph with $V(G \cup H) = V(G) \cup V(H)$ and $E(G \cup H) = E(G) \cup E(H)$.



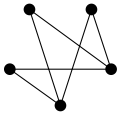
Complement

The **complement** $\overline{G}$ of a graph G is the graph with all the same vertices, but exactly the edges that don't appear in G.

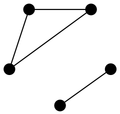
(For every $v, w \in V(G)$ with $v \neq w$, $vw \in E(\overline{G})$ if and only if $vw \notin E(G)$.)

Example:

$K_{3,2}$:



$\overline{K_{3,2}} \cong K_3 \cup K_2$:



Special case: the complement of the complete graph $\overline{K_n}$ is called the **empty graph**. It has n vertices and no edges.

Graph products

There are several notions of **graph products**. A graph product of G and H should be a graph with vertex set $V(G) \times V(H)$. *(Your textbook defines one, which it writes $G \times H$; notation varies.)*

Motivation: walking around G and H simultaneously. (Maybe you're solving two puzzles at once, or a robot is moving two robot arms.)

The question is: what should count as a single step?

- Is it "take a step in one of the graphs, and stay put in the other?" (Your textbook's choice.)
- Is it "take a step in each graph?"
- Or do you get to pick how many graphs to take a step in?

Depends on the problem we want to solve!

Proofs by induction

Proofs by induction are a way to deal with a hard proof a little at a time:

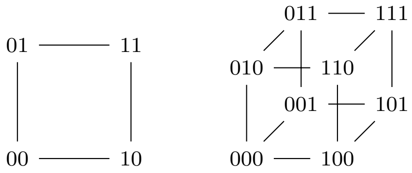
1. We prove some easy cases of the proof called base cases directly.
2. For each other case of the proof, we prove it assuming some simpler case.
3. For this to work, going from each case to its simpler case to its simpler case and so on must lead to a base case.

The most common case: our statement depends on a parameter n. We prove $n = 0$ or maybe $n = 1$ as the base case. For each other value of n, we prove the case assuming the $n - 1$ case.

For graphs, the parameter is often the number of vertices, or edges.

Example: diameter of hypercubes

The **hypercube graph** Q_n is the graph with with vertices $\{0,1\}^n$: n-tuples of 0 or 1. Two vertices are adjacent if they differ in only one coordinate.

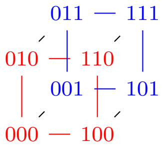


Theorem. $\operatorname{diam}(Q_n) = n$.

Note that this has two parts. We need to prove that there is a $u - v$ path of length $\leq n$ between any two vertices, and we need to find two vertices that are at least n steps apart.

An upper bound on diameter

The hypercube graph Q_n has a recursive structure: Q_n is made up of two copies of Q_{n-1}, with an extra edge between each vertex and the corresponding vertex of the other copy. We'll use this.



Proof that $\operatorname{diam}(Q_n) \le n$. Check this for Q_1 by hand, then induct on n. Let u, v be any two vertices of Q_n. If they're in the same copy of Q_{n-1}, then $d(u,v) \le n-1$ by induction.

Otherwise, let u' be u's neighbor in the other copy of Q_{n-1}; then $d(u', v) \le n-1$ by induction, so $d(u,v) \le n$. $\square$

The matching lower bound

Proof that $\operatorname{diam}(Q_n) \geq n$. We'll prove that if u, v differ in all coordinates, then $d(u, v) \geq n$ by induction.

Take any $u - v$ path $(u = v_0, v_1, \ldots, v_k = v)$. Imagine taking its "shadow" in Q_{n-1} as follows:

1. Drop the last coordinate of each vertex of the path.
2. If two consecutive vertices are now equal (which must happen at least once), delete the duplicate.

Example: $(000, 010, 110, 111) \rightsquigarrow (00, 01, 11, 11) \rightsquigarrow (00, 01, 11)$.

The new path connects vertices in Q_{n-1} that differ in all coordinates, so it has length at least $n - 1$ by induction. Therefore the original path had length at least n. $\square$

Induction on graphs

There are two common kinds of proof by induction for statements about all graphs (or all graphs with some property).

1. Induction on the edges. Here, we start with an empty graph that has as many vertices as we like, but no edges. Usually, things are easy to prove about it.

   Then, we check that if we add an edge to our graph, the statement remains true.

2. Induction on the vertices. Here, we want to go from n-vertex graphs to $(n+1)$-vertex graphs. It's important to be careful, because there's a common mistake that occurs here. . .

An incorrect example

False theorem. If G is a graph on $n \geq 3$ vertices, and each vertex is the endpoint of at least 2 edges, then G has at least $2n - 3$ edges.

Bad proof. True for $n = 3$: when $n = 3$, G must be K_3.

Assume it's true for n. Take an n-vertex graph, and add a new vertex v to it; to satisfy the condition, we need to add at least two edges with endpoint v.

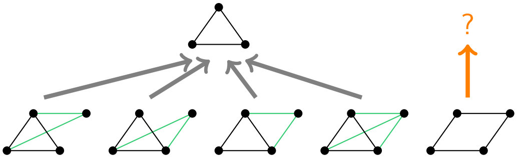
We started with $2n - 3$ edges, and added at least 2, so we have at least $2n - 1 = 2(n + 1) - 3$. By induction, the claim holds for all n. □

Counterexample: take the n-cycle C_n. This has n vertices, satisfies the condition, but has only n edges! For $n > 3$, n is less than $2n - 3$.

The induction trap

The problem with this example is the way it builds graphs up starting from the base case. Some graphs (such as the counterexample C_n) **can't be reached** by starting with a 3-vertex graph and building up with the condition satisfied at each step.

Imagine a "family tree" of cases, where every case of a theorem has a "parent case" it comes from.



Every case should be descended from a base case. Induction doesn't work if there are orphans!

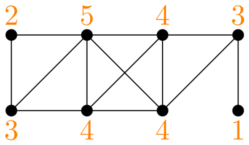
Avoiding the induction trap

There are two approaches to avoiding the induction trap. I'll assume here that we're proving a claim about n-vertex graphs of some special kind.

1. When you start with an n-vertex graph of the special kind, and add a vertex, prove that **all** $(n+1)$-vertex graphs of the special kind can be obtained.
2. "Go from $n-1$ to n". Take an arbitrary n-vertex graph of the special kind, and delete a vertex (maybe a carefully chosen vertex) to reduce it to an $(n-1)$-vertex graph of that kind. Apply the induction hypothesis to that graph.

Many definitions

The **degree** $\deg(v)$ of a vertex v is the number of edges that have v as an endpoint: the number of edges **incident** on v.



Some special terminology:

- A vertex v is called **isolated** if $\deg(v) = 0$ and a **leaf** if $\deg(v) = 1$.
- A graph G has **minimum degree** $\delta(G)$ and **maximum degree** $\Delta(G)$ equal to the smallest/largest degree among its vertices.

What's the point?

Why do we care about vertex degrees?

- It's a simple property that's easy to measure, so it comes up a lot in otherwise unrelated problems.
- It's a primitive measure of connectivity: how resilient a graph is to being disconnected by removing edges.

  (It's not very good: it takes very high degree to even guarantee connectedness.)
- Vertex degrees, and especially their degree distributions, are used in network analysis to make guesses about the process that created a network.

  (They might also appear for many other miscellaneous reasons.)

The degree sum formula

Also known as the handshake lemma, or the “first theorem of graph theory.”

Theorem. If G has m edges and vertices $v_1, \dots, v_n$, then

$$\sum_{i=1}^{n} \deg(v_i) = 2m.$$

Proof. Induct on m. If G has no edges, both sides are 0.

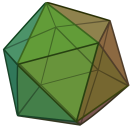
If we add an edge $v_i v_j$, then $\deg(v_i)$ and $\deg(v_j)$ each go up by 1, and $2m$ goes up by 2, so the equation continues to hold. $\square$

Three quick applications

Q1. Is there a 2021-vertex graph in which all vertices have degree 3?

A1. No: the sum of degrees is $3 \cdot 2021 = 6063$, which cannot be $2m$.

Q2. An icosahedron has 12 vertices. How many edges?



A2. The sum of degrees is $12 \cdot 5 = 60$, so there are $\frac{60}{2} = 30$ edges.

Q3. If G is a 9-vertex, 20-edge graph, what can $\delta(G)$ be?

A3. The degree sum is $2m = 40$, so $\delta(G) \leq 4$: if all vertices have degree 5 or more, that adds up to 45. $\delta(G) = 0, 1, 2, 3, 4$ are all possible.

Odd and even vertices

When we're lazy, we say that a vertex v is **odd** if $\deg(v)$ is odd; v is **even** if $\deg(v)$ is even.

Corollary of the handshake lemma. Every graph has an even number of odd vertices.

(This generalizes our earlier claim that there's no 2021-vertex graph where every vertex has degree 3.)

Proof. The sum of degrees is always even: it's twice the number of edges.

Even vertices don't change whether the sum is even or odd. If we add an odd number of odd degrees, we get an odd number, which is impossible; we must be adding up an even number of odd degrees. $\square$

From degrees to cycles

Theorem. Every graph G with $\delta(G) \geq 2$ contains a cycle.

Intuition: start at any vertex and keep walking to adjacent vertices. Every time you reach a new vertex you haven't seen before, you can use a different edge from the one you came in by, because that vertex has at least 2 incident edges.

At some point, you will reach a vertex you **have** seen before. The first time that happens, the portion of our walk starting and ending at that vertex is a cycle!

Whenever you find yourself saying "If we're not done yet, we can keep going," it's a good place to apply the extremal principle.

Proof using the extremal principle

Theorem. Every graph G with $\delta(G) \geq 2$ contains a cycle.

Proof. Pick the biggest/smallest/longest. . . longest path in G. Call it

$$(v_0, v_1, v_2, \dots, v_k).$$

We know that $\deg(v_k) \geq 2$. One of the edges incident to v_k is $v_{k-1}v_k$, but there is at least one other. Where does it go?

- If the edge is $v_k w$, where $w \notin \{v_0, v_1, \dots, v_{k-2}\}$, then we can make our path longer: $(v_0, v_1, v_2, \dots, v_k, w)$.

  This contradicts our choice of path, so it's impossible.

- So the edge is $v_j v_k$ for some $j \leq k-2$. This gives us a cycle:

$$(v_j, v_{j+1}, \dots, v_k, v_j). \quad \square$$

Generalizations

There are several ways to generalize this theorem.

- Every graph G with $\delta(G) \geq k$ contains a cycle **of length at least** $k + 1$.

  I'm assigning a version of this statement for homework. Think about our proof, how the cycle could end up being too short, and how we can avoid that with a larger minimum degree.

- Every n-vertex graph G with $\delta(G) \geq \frac{n}{2}$ contains a cycle through all n vertices.

  Note that this is twice as long as we expect! We'll prove this generalization much later in the course.

Tight results

Whenever we prove a theorem, we'd like it to be the best theorem possible.

For example, suppose we prove "Every graph G with $\delta(G) \geq k$ contains a cycle of length at least $k+1$". Was this the best we could do, or can we get a cycle of length $k+2$? Of length $2k$?

No! We can construct a counterexample. K_n, or a graph consisting of many copies of K_n, has minimum degree $n-1$, but no cycles longer than n.

We say that the result is **tight** if it cannot be improved. In this case, the value of $k+1$ in the statement is tight: without additional assumptions, we cannot prove anything better.

Degree and connectedness

We'd like to think that the higher the degrees of vertices are in a graph, the better connected it is.

This is not quite true. Being connected and very resilient (as we'll see later in the course) requires high vertex degrees, but a graph can have pretty high minimum degree without being connected.

But we can prove, for an n-vertex graph G:

- If $\delta(G) \geq \frac{n-1}{2}$, then G is connected.
- What's more, if $\deg(u) + \deg(v) \geq n - 1$ whenever $uv \notin E(G)$, then G is connected.
- What's more, G is not just connected but $\operatorname{diam}(G) \leq 2$.

Proving the theorem

Theorem. If G has n vertices and $\deg(u) + \deg(v) \geq n - 1$ whenever $uv \notin E(G)$, then $\operatorname{diam}(G) \leq 2$.

Proof. Take two arbitrary vertices u, v; we'll prove $d(u, v) \leq 2$.

If uv is an edge, then there's a length-1 path (u, v), and we're done.

Otherwise, there are $n - 2$ other vertices in the graph, and the condition is that they have at least $n - 1$ edges to u or to v. At least one of them must have an edge to both!

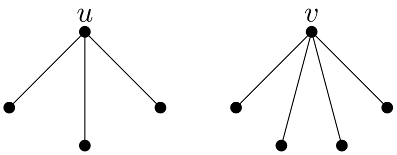
Call that common neighbor w. Then (u, w, v) is a path of length 2, so $d(u, v) \leq 2$. Therefore $\operatorname{diam}(G) \leq 2$. □

Is this theorem tight?

We already discussed that whenever we prove a theorem, it's nice to be able to give an example to show that the bound it gives is tight.

The proof of the theorem is actually a good place to start looking for an example. We can look at it and ask: if we replace $n-1$ by $n-2$, what would it take for this argument to fail, and for G not to be connected?

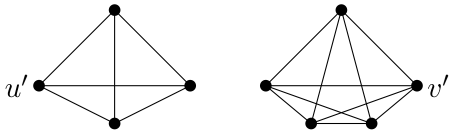
Vertices u, v must have $n-2$ edges to the rest of the graph, but no common vertices:



Note that this is already just two connected components.

This theorem is tight!

Suppose our graph looks like this, and we'd like it not to be connected.



Pick u' on the left and v' on the right. Again, we want $\deg(u') + \deg(v') \geq n - 2$, but without connecting the graph. This can only happen if everything on the left is adjacent to u' and everything on the right to v'.

By the same argument, we get all the other edges in both components. This gives us the **extremal construction**: G must be a disjoint union $K_s \cup K_t$ with $s + t = n$.

A glimpse of random graphs

Suppose we generate a graph G with vertices $v_1, \dots, v_n$ by flipping a coin for each pair $\{v_i, v_j\}$ to see if $v_i v_j \in E(G)$.

Theorem. With probability tending to 1 as $n \to \infty$, G is connected and has $\operatorname{diam}(G) \leq 2$.

Proof. For each pair $\{v_i, v_j\}$, the probability that there is no path (v_i, v_k, v_j) for some $k \neq i, j$ is $(\frac{3}{4})^{n-2}$. (Any particular v_k works with probability $\frac{1}{4}$; if there is no path, then all $n-2$ possible v_k have to fail.)

There are $\frac{n(n-1)}{2}$ pairs $\{v_i, v_j\}$. An upper bound for the probability that any of them are too far apart is $\frac{n(n-1)}{2}(\frac{3}{4})^{n-2}$. This goes to 1 as $n \to \infty$.

Average degree

If G is a graph with vertices $v_1, v_2, \dots, v_n$, then its **average degree** is the average

$$\frac{\deg(v_1) + \deg(v_2) + \cdots + \deg(v_n)}{n}.$$

By the handshake lemma, if G has m edges, it has average degree $\frac{2m}{n}$.

It's often convenient to be able to say that $\delta(G) \le \frac{2m}{n} \le \Delta(G)$: there must be a vertex whose degree is at most average, and a vertex whose degree is at least average.

Unfortunately, we can't put a lower bound on $\delta(G)$ from the average degree alone: maybe there's an isolated vertex that's an outlier.

Subgraphs with high minimum degree

Theorem. If G has average degree d, it has a subgraph H with $\delta(H) > \frac{d}{2}$.

Why $\frac{d}{2}$? It's the best value for which we can say: delete this vertex, and the average degree of the remainder will still be at least d. (We'll prove this on the next slide.)

This suggests two strategies for proving the theorem:

- The extremal principle: go to the end of "delete vertices like this" by saying "let H be the smallest subgraph of G with average degree d.
- We will see how to turn the same idea into a proof by induction, for practice with induction on graphs.

The low-degree vertex lemma

Lemma. If G has average degree d, and v is a vertex with $\deg(v) \leq \frac{d}{2}$, then $G - v$ has average degree at least d.

Proof. Saying that G has average degree d means that $\frac{2|E(G)|}{|V(G)|} = d$, or $|E(G)| = \frac{d}{2}|V(G)|$.

In $G - v$, there are $\deg(v)$ fewer edges, and 1 fewer vertex. So

$$|E(G-v)| = \frac{d}{2}|V(G)| - \deg(v) \qquad |V(G-v)| = |V(G)| - 1.$$

Since $\deg(v) \leq \frac{d}{2}$, we get

$$|E(G-v)| \geq \frac{d}{2}(|V(G)| - 1) = \frac{d}{2}|V(G-v)|,$$

proving that $G - v$ has average degree at least d. $\square$

Proving the theorem by induction

Theorem. If G has average degree at least d, it has a subgraph H with $\delta(H) > \frac{d}{2}$.

Proof. Induct on n, the number of vertices in G. Our base case can be $n = 1, \dots, d$ (where there are no such vertices) or $n = d + 1$ (where $G = K_n$ is the only possibility).

Now assume we know the theorem for $(n-1)$-vertex graphs. If $\delta(G) > \frac{d}{2}$, then G itself is the subgraph we want, so we can assume G has a vertex v with $\deg(v) \le \frac{d}{2}$.

By the lemma, $G - v$ has average degree at least d. By the inductive hypothesis, $G - v$ has a subgraph H with $\delta(H) > \frac{d}{2}$. This is also a subgraph of G, which was what we wanted. $\square$

Definitions

A graph G is **regular** if all of its vertices have the same degree. (We call a graph r-regular if all of its vertices have degree r.)

Among the graphs we have seen:

- C_n is 2-regular and K_n is $(n-1)$-regular.
- $K_{s,t}$ is not in general regular, but $K_{n,n}$ is n-regular.
- The hypercube graph Q_n is n-regular (with 2^n vertices).
- If G is an r-regular graph with n vertices, then $\overline{G}$ is $(n-r-1)$-regular.

Key things to know

What kind of r-regular graphs are out there?

- By the handshake lemma, there is no r-regular graph on n vertices if r and n are both odd.
- Provided that either r or n is even, and $0 \leq r \leq n-1$, an r-regular graph always exists.
- In most cases, there are many possible r-regular graphs.

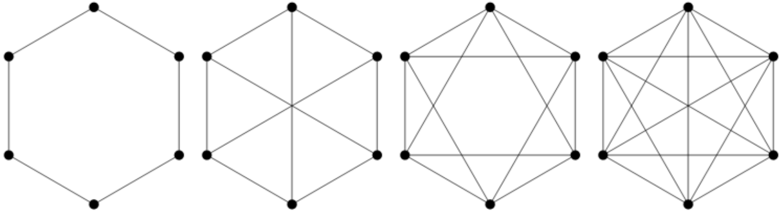
  (But for $r \leq 2$ or $r \geq n-3$, we can classify them all.)
- Graphs that are highly symmetric tend to be regular, but regular graphs don't have to be symmetric at all.

Harary graphs

Theorem. If $0 \leq r \leq n-1$ and rn is even, there is an r-regular graph on n vertices.

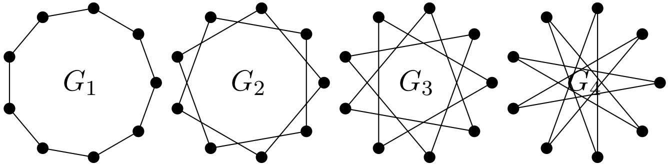
The best way to prove a theorem like this is to describe how to construct such a graph.

One very nice family of graphs of this type is called the Harary graphs.



Harary graphs for even r

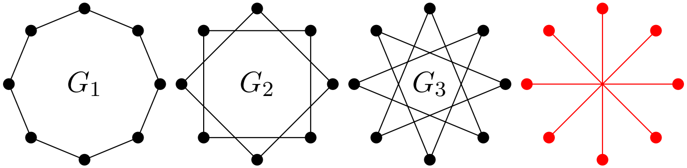
Arrange n vertices in a circle. For $k = 1, \dots, \lfloor \frac{n-1}{2} \rfloor$, let G_k be the graph where vertices are adjacent if they're "k steps apart" around the circle.



These graphs are 2-regular, and don't share any edges. The Harary graph $H_{r,n}$ is the union $G_1 \cup G_2 \cup \cdots \cup G_{r/2}$ (all with the same vertex set), and is an r-regular graph.

Harary graphs for odd r, and even n

When r is odd, and n is even, include an extra graph H where opposite vertices are joined together.



Again, if we assume all the vertices are the same in each case, they don't have any edges in common.

To get a Harary graph $H_{r,n}$ for odd r, include $\frac{r-1}{2}$ of the 2-regular graphs, together with the 1-regular graph H.

The really boring cases

Theorem. When r is one of $0, 1, n-2, n-1$, there is "at most one" r-regular graph on n vertices.

("At most one" really means "at most one, up to isomorphism"; we won't discuss this rigorously until later.)

Proof. When $r = 1$, if you take a vertex v and its only neighbor w, they cannot be adjacent to anything else. The graph must have $\frac{n}{2}$ connected components, each a copy of K_2.

When $r = 0$, there are no edges, so the only possibility is the empty graph.

When $r = n-2$ or $r = n-1$, the complement of the graph must have $r = 1$ or $r = 0$ respectively. $\square$

2-regular graphs

Theorem. The “only” connected 2-regular graph on n vertices is C_n.

Proof. A 2-regular graph has minimum degree 2; earlier, we proved that this means it must have a cycle.

If that cycle is not the whole graph, we have a problem: there can’t be edges between it and any other vertices, so the graph would not be connected.

So the cycle must pass through all n vertices. $\square$

This means that in general, a 2-regular graph is the disjoint union of several components that are cycles. The complements of such graphs give us the $(n-3)$-regular graphs.

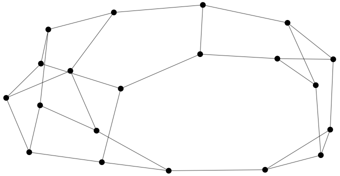
3-regular graphs

Once we get to 3-regular graphs (also known as "cubic graphs"), we have a lot more variety. The number of "different" connected cubic graphs on n vertices is

$$0, 0, 0, 0, 1, 0, 2, 0, 5, 0, 19, 0, 85, 0, 509, 0, 4060, \ldots$$

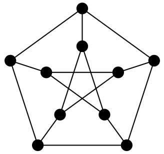
See `https://oeis.org/A002851` for this sequence.

Although the Harary graphs give a very symmetric picture, almost all of these graphs (for large n) look very very asymmetric.



The Petersen graph

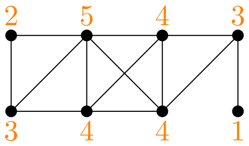
The Petersen graph can be defined as follows: take all 2-element subsets of $\{1, 2, 3, 4, 5\}$ as its vertices, $V = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \ldots, \{3, 5\}, \{4, 5\}\}$, and say that two of these are adjacent if they have no intersection, such as $\{1, 2\}$ and $\{3, 4\}$. Here is a diagram:



The graph is primarily known for being the simplest or sometimes only counterexample to many conjectures.

Degree sequences

The **degree sequence** of a graph is just the finite sequence of degrees of all its vertices.



$2, 5, 4, 3, 3, 4, 4, 1$

Often, we look at a sorted degree sequence to tell us information about the graph as a glance: $5, 4, 4, 4, 3, 3, 2, 1$ or $1, 2, 3, 3, 4, 4, 4, 5$.

It's a good first-pass "signature" to distinguish graphs, but different graphs can have the same degree sequence. (For example, 19 different graphs have the degree sequence $3, 3, 3, 3, 3, 3, 3, 3, 3, 3$.)

Graphical sequences

A sequence of numbers is called **graphical** if it is the degree sequence of some graph. Finding the degree sequence of a graph is easy; going the other way is harder.

Common questions we ask:

- Given a sequence of numbers, is it graphical?
- Given a graphical sequence, how many graphs have it as their degree sequence?
- Given a graphical sequence, can we pick a graph with that degree sequence at random?

Several examples

Here are some ways we can try to tell if a sequence is graphical or not.

1. Is the sequence $2, 2, 3, 3, 3$ graphical?

   No: the number of odd-degree vertices cannot be odd.

2. Is the sequence $1, 2, 3, 4, 5, 6, 7$ graphical?

   No: the value 7 is too big. In a 7-vertex graph, all degrees are at most 6.

3. Is the sequence $0, 1, 2, 3, 4$ graphical?

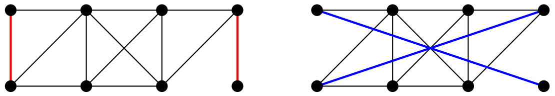
   No. In a 5-vertex graph, a degree-4 vertex would be adjacent to every other vertex. But then we can't have a vertex of degree 0.

But these are just three possible reasons; they don't generalize!

Edge swap

An **edge swap** is an operation we can do on a graph to get a different graph with the same degree sequence.

The idea is this: we delete edges uv and xy and add edges ux and vy. We only do this if those edges did not already exist in the graph.



A new question: if we have two graphs with the same degree sequence, can we turn one into the other by doing edge swaps?

The edge swap lemma

Lemma. Let G be a graph with vertices $v_1, v_2, \dots, v_n$ sorted so that

$$\Delta(G) = \deg(v_1) \ge \deg(v_2) \ge \cdots \ge \deg(v_n) = \delta(G).$$

A sequence of edge swaps can turn G into a graph where v_1 is adjacent to exactly the vertices $\{v_2, v_3, \dots, v_{\Delta(G)+1}\}$.

Why prove this lemma?

1. It's the first step to showing that two graphs with the same degree sequence are connected via edge swaps: we first get v_1 to do the same thing in both graphs, then repeat on the remainders.
2. After the edge swaps, $G - v_1$ has a predictable degree sequence. We will later use this to test if a sequence is graphical recursively.

Proving the lemma: planning

Let S be the set of vertices $\{v_2, v_3, \ldots, v_{\Delta(G)+1}\}$. Our goal is to perform edge swaps to get v_1 to be adjacent to all the vertices in S (and no others).

In a single edge swap, what can we hope to do? Since we only care about the neighbors of v_1, say we swap edges $\{v_1v_i, v_jv_k\}$ to get $\{v_1v_j, v_iv_k\}$ (For some i, j, k.)

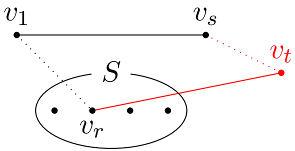
- After this swap, v_1 is adjacent to v_j but it stops being adjacent to v_i. This is good for us if $v_j \in S$, but $v_i \notin S$.
- If we're not done, then v_i and v_j must exist; the trick is finding v_k such that **edge** v_jv_k **exists** but **edge** v_iv_k **does not**.
- If that's always possible, then every swap we do increase's v_1's neighbors in S, until it's adjacent to all of them.

Proving the lemma: step 1

Proof. Let G be a graph as described earlier, where v_1 is **not yet** adjacent to **all** the vertices in S.

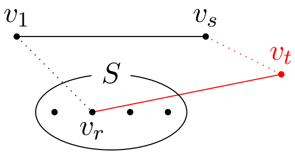
Then there is some vertex in S not adjacent to v_1: let v_r be any such vertex. Also, since $\deg(v_1) = |S|$ but v_1 is not adjacent to all the vertices in S, there must be a vertex outside S adjacent to v_1: let v_s be such a vertex.

Our goal: find a third vertex v_t adjacent to v_r but not to v_s.



Proving the lemma: step 2

Why must the vertex v_t exist?



We know that $\deg(v_r) \geq \deg(v_s)$, because of how we chose S. What's more, v_s has a neighbor that v_r doesn't: the vertex v_1.

This must be made up for by a vertex $v_t \notin \{v_1, v_r, v_s\}$ adjacent to v_r, but not v_s. (The edge $v_r v_s$, if it exists, would help v_r and v_s equally.)

Replacing edges $\{v_1 v_s, v_r v_t\}$ by $\{v_1 v_r, v_s v_t\}$ increases the number of edges from v_1 to S. We can make progress to our goal! □

The Havel–Hakimi theorem

Theorem (Havel 1957, Hakimi 1963). A sequence $(d_1, d_2, \dots, d_n)$ sorted in decreasing order is graphic if and only if the sequence

$$(\underbrace{d_2 - 1, d_3 - 1, \dots, d_{d_1+1} - 1}_{d_1 \text{ terms}}, d_{d_1+2}, \dots, d_n)$$

is graphic.

This lets us reduce the problem to a smaller problem, giving us a recursive algorithm to test if a sequence is graphic. (We'll see examples.)

Proof of $\Longleftarrow$. If the second sequence is graphic, take a graph on vertices $v_2, v_3, \dots, v_n$ with that degree sequence.

Then add a vertex v_1 adjacent to $v_2, v_3, \dots, v_{d_1+1}$. □

Using the edge swap lemma

Proof of $\implies$. Recall the edge swap lemma: if G is a graph with degree sequence $(d_1, d_2, \dots, d_n)$, then we can perform edge swaps to make v_1 adjacent to $v_2, v_3, \dots, v_{d_1+1}$.

In particular: if $(d_1, d_2, \dots, d_n)$ is graphic, then there is a graph G with that degree sequence in which v_1 is adjacent to $v_2, v_3, \dots, v_{d_1+1}$.

Deleting vertex v_1 from that graph produces exactly the degree sequence

$$(\underbrace{d_2 - 1, d_3 - 1, \dots, d_{d_1+1} - 1}_{d_1 \text{ terms}}, d_{d_1+2}, \dots, d_n)$$

that we wanted.

A non-graphic example

Let's test the sequence $(5,5,5,4,2,1,1,1)$ for being graphic.

1. Delete 5 and subtract 1 from the next 5 numbers, getting $(5-1,5-1,4-1,2-1,1-1,1,1)$.
2. Simplify to $(4,4,3,1,0,1,1)$ and sort to $(4,4,3,1,1,1,0)$.
3. Delete 4 and subtract 1 from the next 4 numbers, getting $(4-1,3-1,1-1,1-1,1,0)$.
4. Simplify to $(3,2,0,0,1,0)$ and sort to $(3,2,1,0,0,0)$.
5. Delete 3 and subtract 1 from the next 3 numbers, getting $(2-1,1-1,0-1,0,0)$.
6. We got a negative number: the sequence isn't graphic.

A graphic example

Let's test the sequence $(3, 3, 3, 3, 2)$ for being graphic.

1. Delete 3 and subtract 1 from the next 3 numbers, getting $(3-1, 3-1, 3-1, 2)$.
2. Simplify to $(2, 2, 2, 2)$.
3. Delete 2 and subtract 1 from the next 2 numbers, getting $(2-1, 2-1, 2)$.
4. Simplify to $(1, 1, 2)$ and sort to $(2, 1, 1)$.
5. Delete 2 and subtract 1 from the next 2 numbers, getting $(1-1, 1-1)$.
6. Simplify to $(0, 0)$ and realize that this is graphic. We can work backwards to find a realization.

Edge swaps can do anything

Theorem. If G and H have the same degree sequence, then we can turn G into H via edge swaps.

Proof. Enough to show: we can turn both G and H into the same third graph K via edge swaps. We'll prove this by induction on n, the number of vertices.

1. Label vertices of G and H as $v_1, \dots, v_n$ with decreasing degrees.
2. Do swaps on both to make v_1 adjacent to $v_2, v_3, \dots, v_{\Delta+1}$.

   Call the resulting graphs G' and H'.
3. By induction, we can do swaps to turn $G' - v_1$ and $H' - v_1$ to turn them into the same graph.

   The same swaps can be done to turn G', H' into the same graph.

Generating graphs with a degree sequence

Cultural aside: how do we generate a random graph with degree sequence $(d_1, d_2, \ldots, d_n)$?

1. Begin with **some** graph with this degree sequence.
2. Do lots of random edge swaps.

To a first approximation, we can just say that every graph is about equally likely to be generated, if we've done enough edge swaps. At the very least, every graph is **possible** to generate.

There are fancier methods that can be more precise, which are beyond us in this course.

Erdős–Gallai theorem

There is another way to test if a sequence is graphic.

Theorem (Erdős–Gallai, 1961). A sequence of nonnegative integers $d_1 \geq d_2 \geq \cdots \geq d_n$ is graphic if $d_1 + \cdots + d_n$ is even and

$$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min\{d_i, k\}$$

for $k = 1, 2, \ldots, n$.

For example, $(5, 5, 5, 4, 2, 1, 1, 1)$ is not graphic because, when we take $k = 4$, we get

$$5 + 5 + 5 + 4 > 4(3) + 2 + 1 + 1 + 1,$$

which does not satisfy the inequality.

Erdős–Gallai intuition

Where does the inequality

$$\sum_{i=1}^{k} d_i \le k(k-1) + \sum_{i=k+1}^{n} \min\{d_i, k\}$$

come from? We can rewrite it as

$$\sum_{i=1}^{k} (d_i - (k-1)) \le \sum_{i=k+1}^{n} \min\{d_i, k\}.$$

Count edges between $\{v_1, \dots, v_k\}$ and $\{v_{k+1}, \dots, v_n\}$:

- For $i \le k$, v_i has at least $d_i - (k-1)$ edges to $v_{k+1}, \dots, v_n$.
- For $i > k$, v_i has at most $\min\{d_i, k\}$ edges to $v_1, \dots, v_k$.

Note: we have not proved that this condition is sufficient!

When are graphs isomorphic?

Sometimes we want to say that two graphs have **the same structure**. For instance, even though there are many possibly connected 2-regular graphs on n vertices, they all “look like C_n”.

Intuitively, we can “relabel” any of these graphs to get C_n.

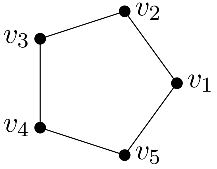
To make this a rigorous statement we can prove or disprove, we define graph isomorphism.

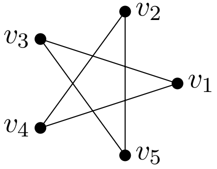
We say that a graph G is **isomorphic** to another graph H if

1. there is a bijection $\phi : V(G) \to V(H)$ (the “relabeling”)...
2. ...that preserves edges: $uv \in E(G)$ if and only if $\phi(u)\phi(v) \in E(H)$.

Example: C_5 is isomorphic to $\overline{C_5}$

For example, let $G = C_5$ and $H = \overline{C_5}$: the complement of C_5.





(We often don't label our vertices, but here, the labels are important!)

These are isomorphic. Define $\phi : V(G) \to V(H)$ by $\phi(v_1) = v_1$, $\phi(v_2) = v_3$, $\phi(v_3) = v_5$, $\phi(v_4) = v_2$, and $\phi(v_5) = v_4$.

This is a bijection, and the edges $v_1v_2, v_2v_3, v_3v_4, v_4v_5, v_5v_1$ of G are mapped to v_1v_3, v_3v_5, v_5v_2, v_2v_4, v_4v_1: exactly the edges of H.

Graph properties

Most properties of graphs we've discussed are preserved by graph isomorphism. (Philosophically, we can say that graph theory is all about studying properties preserved by graph isomorphism.)

For example, suppose G and H are isomorphic. Then:

- G and H have the same number of vertices, the same number of edges, the same diameter.
- H is connected iff G is connected; bipartite iff G is bipartite; regular iff G is regular; etc.
- G and H have the same degree sequence (up to reordering).
- G contains the same number of copies of K_3 (subgraphs isomorphic to K_3!) as H does; the same for any other subgraph.

Example: connectedness is a graph property

These claims all need proof. The proofs are all very similar; let's just do a few. (Good exercise: think about some others!)

Theorem. Suppose G and H are isomorphic. If G is connected, then H is connected.

Proof. Let $\phi : V(G) \to V(H)$ be an isomorphism. Pick any vertices $u, v \in V(H)$. Because G is connected, there is a walk from $\phi^{-1}(u)$ to $\phi^{-1}(v)$: a walk $(v_0, v_1, \ldots, v_k)$ with $v_0 = \phi^{-1}(u)$ and $v_k = \phi^{-1}(v)$.

Because ϕ is an isomorphism, the walk $(\phi(v_0), \phi(v_1), \ldots, \phi(v_k))$ is a walk in H: for each i, $v_i v_{i+1} \in E(G)$, so $\phi(v_i)\phi(v_{i+1}) \in E(H)$.

This walk starts at $\phi(v_0) = u$ and ends at $\phi(v_k) = v$!

Therefore there is a $u - v$ walk in H. Therefore H is connected. $\square$

Example: the number of edges is a graph property

Theorem. Suppose G and H are isomorphic. Then $|E(G)| = |E(H)|$.

Proof. The most common way to prove that two sets are the same size is to find a bijection between them. Define $f : E(G) \to E(H)$ by $f(vw) = \phi(v)\phi(w)$.

We know that this definition is okay, because if $vw \in E(G)$, $\phi(v)\phi(w) \in E(H)$ by the definition of graph isomorphism.

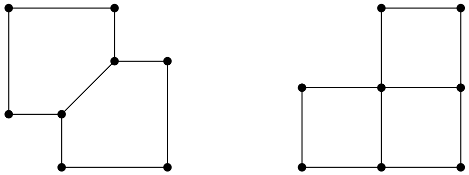
To show f is a bijection, we find an inverse. That inverse is $g : E(H) \to E(G)$ defined by $g(xy) = \phi^{-1}(x)\phi^{-1}(y)$.

(We can check that $g(f(vw)) = vw$ and $f(g(xy)) = xy$.)

Therefore f is a bijection, so $|E(G)| = |E(H)|$. □

Our first example

Are these two graphs isomorphic?

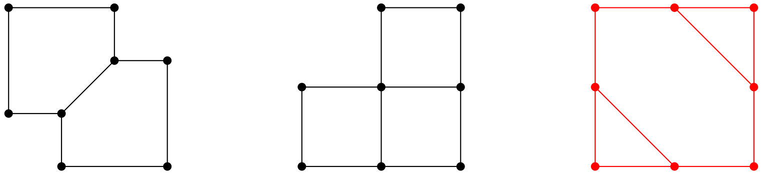


If the answer were “yes”, we could try to prove it by finding an isomorphism. After we try it for a while and it doesn’t work, we suspect the answer is “no”, but that’s not a proof!

We can **prove** the answer is isomorphic by finding a graph property that distinguishes them. In this case, the two graphs are distinguished by the number of edges: the left one has 9, and the right one has 10.

A third graph

Is the third graph isomorphic to either of the first two?

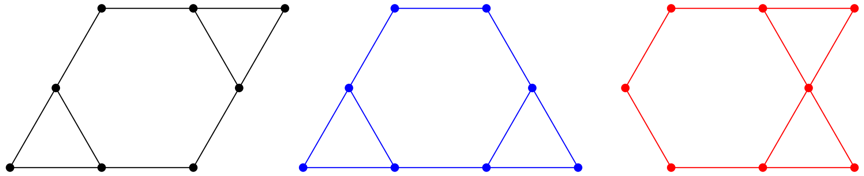


It has 10 edges, so if it's isomorphic to any of them, it's isomorphic to the second one. But there are a few ways to distinguish it from the second one as well:

- The second graph has a vertex of degree 4, but the third doesn't.
- The second graph is bipartite, but the third graph isn't. (It contains an odd cycle!)

Degrees and isomorphisms

What about these three graphs?



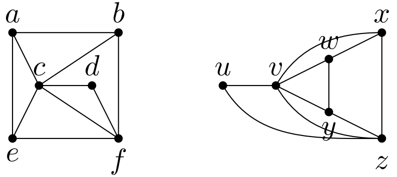
The third graph is different because it has a vertex of degree 4. The first and second have the same degree sequence. . .

If the first two graphs are isomorphic, the degrees hint at how: degree-2 vertices in one graph should go to degree-2 vertices in the other.

This is impossible! In the blue graph, two degree-2 vertices are adjacent. No such vertices exist in the black graph.

Finding an isomorphism

These two graphs are isomorphic. Can we find how?



Degrees help us get started: we must have $\phi(c) = v$ and $\phi(d) = u$. Then $\phi(f)$ must be u's only other neighbor: z.

What's left is matching a, b, e to w, x, y. We must have $\phi(a) = w$ since a is adjacent to b and e, and w is adjacent to x and y.

At the end, b and e are indistinguishable, and so are x and y. We can pick $\phi(b) = x$ and $\phi(e) = y$, or vice versa.

Resilience of graphs

One of the big questions we tackle with graph theory is connectivity: can we get from one vertex to another by walking along the edges?

The next step is to understand how vulnerable graphs are to being disconnected. When will deleting an edge disconnect the graph?

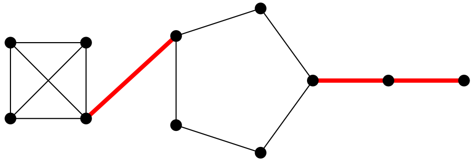
Some applications:

- Road networks in a city, and how connected they are after a traffic accident. (More realistically, railways.)
- Computers in a network, which might lose the ability to communicate if a single line of communication is cut.
- Noticing that a step in solving a puzzle (or writing a proof?) is critical and cannot be avoided.

Bridges

We say that an edge e of a connected graph G is a **bridge** if $G - e$ (the graph we get when we delete e) is disconnected.

(More generally, if G is not connected, the bridges are the edges that increase the number of components when deleted.)



The highlighted edges in the graph above are bridges.

Note that whenever $\deg(v) = 1$, the edge out of v is a bridge.

Bridges and cycles

Theorem. An edge in a graph is a bridge if and only if it is not part of any cycles.

Proof. Suppose G is connected (otherwise, look at a connected component). Let uv be an edge of G.

$$\begin{aligned} & \text{The edge } uv \text{ is not a bridge} \\ \iff \quad & G - uv \text{ is connected} \\ \iff \quad & \text{There is a } u - v \text{ path } (v_0, v_1, \ldots, v_k) \text{ in } G - uv \\ \iff \quad & \text{There is a cycle } (v_0, v_1, \ldots, v_k, v_0) \text{ in } G \text{ with } v_0 v_k = uv \\ \iff \quad & \text{There is a cycle in } G \text{ containing edge } uv. \qquad \square \end{aligned}$$

The hardest step is noticing that $G - uv$ is connected provided that there is a $u - v$ path. For any vertices x, y, there is an $x - y$ walk in G; if it contains edge uv, we can replace it by the $u - v$ path.

Bridges and degrees

We know that if we add up all degrees in a graph G with m edges, their sum is $2m$. What if we sum degrees over a subset $S \subseteq V(G)$?

Each edge with 2 endpoints in S is counted twice. Edges between S and $V(G) - S$ are counted once. So:

$$\sum_{v \in S} \deg(v) = 2(\#\text{ edges in } G[S]) + (\#\text{ edges with } 1 \text{ endpoint in } S).$$

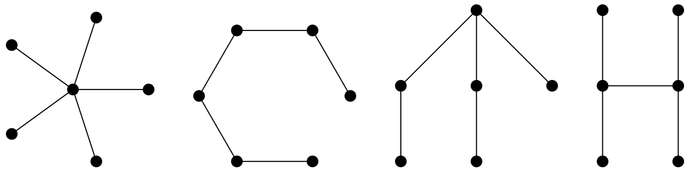
Corollary. If a bridge separates S from $V(G) - S$, then $\sum_{v \in S} \deg(v)$ is odd.

Corollary. If all degrees in G are even, G has no bridges.

Trees

A **tree** is a "minimally connected" graph: a connected graph which becomes disconnected if any edge is deleted. In other words, a tree is a graph in which all edges are bridges.

Some examples (these are 4 out of 6 possible 6-vertex trees):



Note that the path graph P_n is always a tree. So is $K_{1,n}$, which is also called a **star**. (There are many more increasingly silly terms.)

Characterizations of trees

There are many other characterizations of trees: alternative definitions, which hold for trees and no other graphs. Here are some:

- A tree is a connected graph with no cycles (an **acyclic** graph).

  (A cycle would prevent some edges from being bridges. On the other hand, if there are no cycles, every edge must be a bridge.)

- A tree is a maximally acyclic graph: it has no cycles, but adding any edge would create a cycle.

  (Adding edge uv creates a cycle if and only if there is already a $u - v$ path, so "adding any edge would create a new cycle" is another way of saying "connected".)

Unique paths

Here's another characterization of trees.

Claim. A graph G is a tree if and only if there is exactly one path between any two vertices.

We will only prove the hardest part of this claim: if there are two $u - v$ paths for some u and v, then G has a cycle: it isn't a tree.

Let $(u = v_0, v_1, \dots, v_k = v)$ be one of these paths. There must be some edge $v_i v_{i+1}$ that is not an edge of the other path.

Then there is a $v_i - v_{i+1}$ walk in $H = G - v_i v_{i+1}$. To get it, combine $(v_i, v_{i-1}, \dots, v_0 = u)$, the other $u - v$ path, and $(v = v_k, \dots, v_{i+1})$. Therefore there is a $v_i - v_{i+1}$ path in H, which together with $v_i v_{i+1}$ makes a cycle in G. $\square$

Why study trees?

The first reason to study trees is that they are the simplest kind of connected graph. If it is hard to solve a problem for all connected graphs, we can begin by solving it for trees.

A second reason is that often, solving the problem for trees is enough. Here is an example of a claim we won't prove yet:

Claim. Any connected graph G has a vertex v such that $G - v$ is connected.

We will soon be able to prove this from two facts:

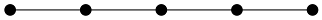
- Any connected graph has a **spanning tree**: a tree connecting all its vertices.
- Any tree T has a vertex v such that $T - v$ is connected.

How many edges in a tree?

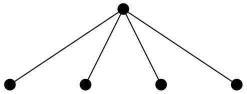
How many edges can a tree with n vertices have?

We can look at some examples we know already:

- A path P_n has n vertices and $n-1$ edges.



- A star $K_{1,n-1}$ has n vertices and $n-1$ edges.



This leads us to guess (and we will now prove) that all trees with n vertices have exactly $n-1$ edges.

Edges and components

We start with the following result:

Theorem. A graph with n vertices and m edges has at least $n - m$ connected components.

Proof. Induct on the number of edges. When $m = 0$, the theorem is clearly true.

Why does the theorem continue to hold when we add an edge uv?

Because the only effect the edge can have is join u's connected component and v's connected component into one connected component (if that wasn't the case already). $\square$

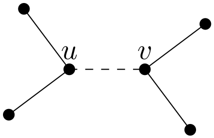
Next, we'll want to know when this minimum number of connected components is achievable.

Edges, components, and cycles

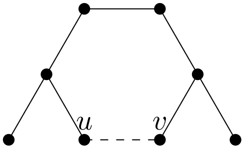
If we look at our proof just now more carefully, we get:

Claim. A graph with n vertices and m edges has **exactly** $n - m$ connected components if and only if it has no cycles.

This goes back to our earlier claims about bridges and cycles. Intuitively, when we add an edge uv, we're in one of two cases:



I. We join two components.



II. We create a cycle.

In particular: n-vertex trees, which have 1 component and no cycles, must have exactly $n - 1$ edges.

Trees have leaves

A **leaf** (or "end-vertex") is a vertex of degree 1.

Theorem. Every tree on $n \geq 2$ vertices has at least 2 leaves.

Proof. We know that in a tree, $\sum_{v \in V} \deg(v) = 2(n-1)$.

However, if a tree had minimum degree 2, $\sum_{v \in V} \deg(v)$ would be at least $2n$, so that can't happen.

Even a single vertex of degree 1 would only get us down to $2n-1$. We need at least two vertices of degree 1... or a vertex of degree 0, which can only exist in a 1-vertex tree. $\square$

This may seem like a trivial observation. But we're about to see that this theorem is actually incredibly useful...

Growing a tree

We've already seen the right way to structure a proof by induction on the number of vertices in a graph:

1. Assume that the statement holds for $(n-1)$-vertex graphs.
2. Given an n-vertex graph G, pick a vertex v to remove.
3. Apply the assumption to $G - v$, and use it to prove the statement for G.

We now know that every tree T has a leaf: a vertex v with $\deg(v) = 1$. This means that $T - v$ is a smaller tree! So we can induct on trees by removing a leaf v and applying the inductive hypothesis to $T - v$.

Another way to put it: all n-vertex trees can be grown from a 1-vertex tree by adding a leaf $n - 1$ times.

Example: trees are bipartite

Theorem. All trees are bipartite.

Proof. This holds for a 1-vertex tree: take the bipartition with the vertex on one side, and nothing on the other.

Assume all $(n-1)$-vertex trees are bipartite. Take an n-vertex tree T and let v be a leaf of T.

By the inductive hypothesis, $T - v$ has a bipartition (A, B). Without loss of generality, v's only neighbor is an element of A. Then $(A, B \cup \{v\})$ is a bipartition of T, proving that T is bipartite.

By induction, the statement holds for trees of any order. □

(Easier proof: bipartite graphs are those with no odd cycles, and trees don't have any cycles.)

Example: finding trees in graphs

Theorem. A graph G with $\delta(G) \geq k-1$ has a copy of every k-vertex tree T as a subgraph.

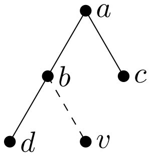
Proof idea. We will induct on k (certainly when $k = 1$, any graph G contains a 1-vertex subgraph.)

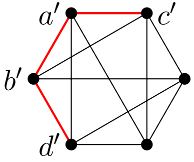
What will the induction look like? We pick a graph G and a tree T, which has a leaf v.

We can apply the induction hypothesis to find a copy of $T - v$ inside G. Now, to do the induction step, all we have to do is "grow" the copy of $T - v$: show that we can extend it to a copy of T.

Finding trees in graphs: an illustration

Here is what this situation can look like.





We've found a copy of $T - v$ inside G. To extend it to a copy of T, we need to find a neighbor of b' that we haven't already used.

How can we fail? If b' has no neighbors that aren't already playing a role in the copy of $T - v$ we found.

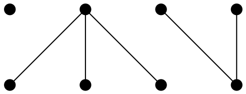
Why can't this occur? Because $\deg(b') \geq \delta(G) = k - 1$, and only $k - 2$ neighbors of b' can be part of the $(k-1)$-vertex tree $T - v$.

Forests

A **forest** is:

- A graph in which every connected component is a tree.
- Equivalently, any acyclic graph (not necessarily connected) is a forest.

A forest with 3 components:

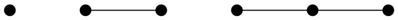


By our previous work, a forest with n vertices can have m edges for any $m \leq n-1$, and it will have exactly $n-m$ components. (In fact, all graphs with n vertices, m edges, and $n-m$ components are forests.)

Some practice with forests

Let's find all the 5-vertex forests.

1. First, we'll need to find all trees on $1, 2, 3, 4, 5$ vertices.
2. Trees on 1, 2, and 3 vertices are unique up to isomorphism:

   Trees on 4 and 5 vertices will require more work.
3. To make sure we find them all, we will systematically try all the possible degree sequences.

Fact we will not prove: Every sequence $(d_1, d_2, \ldots, d_n)$ of positive integers with $d_1 + d_2 + \cdots + d_n = 2(n-1)$ is the degree sequence of at least one tree. (Maybe more than one!)

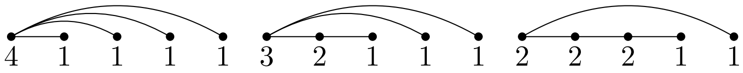
Finding the bigger trees

1. 4-vertex trees can have a sorted degree sequence of $(3, 1, 1, 1)$ or $(2, 2, 1, 1)$.

   The first is (only) a star $K_{1,3}$; the second is (only) a path P_4.

2. 5-vertex trees can have $(4, 1, 1, 1, 1)$ or $(3, 2, 1, 1, 1)$ or $(2, 2, 2, 1, 1)$.

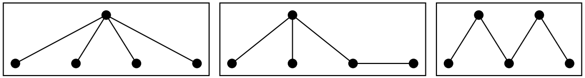
   Each of these also has only one realization:



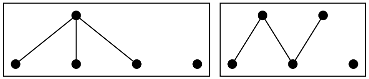
Note that the pattern does not continue. For 6 vertices, the degree sequence $(3, 2, 2, 1, 1, 1)$ corresponds to two different, non-isomorphic trees.

Putting these together

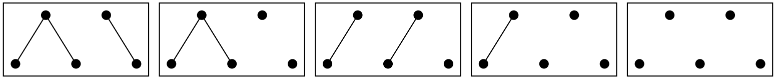
Each 5-vertex tree is a forest:



Each 4-vertex tree, plus an isolated vertex, is a forest:

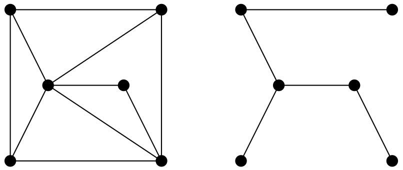


Finally, we can put together the smaller trees in several ways:



Spanning trees

A **spanning tree** of a graph G is a tree subgraph of G that includes all of G's vertices.



What's the point?

- "Simplest" connected structure inside G (for proofs).
- "Cheapest" way to connect all vertices of G (for applications).

Spanning trees exist!

Theorem. Every connected graph G has a spanning tree.

Proof. We proceed by the extremal principle. Let H be the connected subgraph of G with $V(H) = V(G)$ and the fewest edges possible. We will prove that H is a tree.

To do this, we'll show every edge of H is a bridge. Let $uv \in E(H)$. Can $H - uv$ be connected? No: if it were, then $H - uv$ would be a connected subgraph of G with $V(H - uv) = V(G)$ and fewer edges than H.

Therefore uv is a bridge; since uv was arbitrary, all edges of H are bridges; therefore H is a tree. $\square$

The converse also holds: if G has a spanning tree, then it is connected.

Application of spanning trees

Theorem. Every connected graph G with at least 2 vertices has a vertex v such that $G - v$ is connected.

Proof. G has a spanning tree T, and T has a leaf v. $T - v$ is a spanning tree of $G - v$, so $G - v$ is connected. □

In fact, since T has at least two leaves, G has at least two such vertices. (This is proved much more painfully as Theorem 1.9 in the textbook.)

This is a general pattern: if you are proving something about all connected graphs, see if it's enough to prove it for all trees.

Vertex labelings

Theorem. Let G be a connected graph, and let f be a function that gives each vertex $v \in V(G)$ a number $f(v)$. Suppose that whenever uv is an edge, $f(u) = f(v)$. Then f is a constant function: $f(x) = f(y)$ for all vertices $x, y \in V(G)$.

This is a generalization of problem 5 on homework 2, and can be done in the same way!

We can also approach it differently:

1. Prove the theorem for trees (by tree induction).
2. Let T be a spanning tree of G; use the theorem on T to prove the theorem for G.

Vertex labelings: proof

Suppose we have a tree T and a number $f(v)$ for every vertex $v \in V(T)$; whenever uv is an edge, $f(u) = f(v)$. We want to show f is the same for all vertices.

Proof. Induct on the number of vertices in T. When there's just one vertex, f is definitely the same for all vertices!

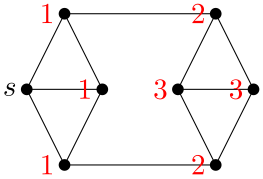
Assume that the result holds for all $(n-1)$-vertex trees. Let T be an n-vertex tree; let v be a leaf, and let w be v's only neighbor.

By the induction hypothesis applied to $T - v$, we know f is constant on the vertices of $T - v$. Also, $f(v) = f(w)$ because vw is an edge, so f has the same value on v as well. Therefore f has the same value on all vertices of T.

By induction, this holds for all trees. $\square$

Finding distances

We have a graph G and a vertex $s \in V(G)$. We want to find the distances $d(s, v)$ for all other vertices v:



We previously described the algorithm as follows: set $d(s, v) = 1$ for all neighbors of s. Set $d(s, v) = 2$ for all their neighbors (that are not neighbors of s). Set $d(s, v) = 3$ for all **their** neighbors, repeat.

When the graph is large, and/or we need to tell a computer to do this, we have to be a bit more precise about how to do this.

Algorithm for finding distances

An algorithm should **keep a list** of the vertices that haven't been processed yet. Initially, the list just has s on it (labeled with distance 0). Repeat the following:

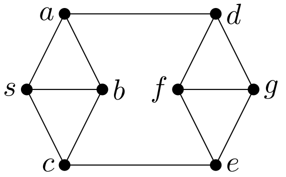
1. Let v be the first vertex on the list; take v off the list.
2. For each neighbor w of v that hasn't been labeled: set $d(s, w) = d(s, v) + 1$, and add w **to the end** of the list.

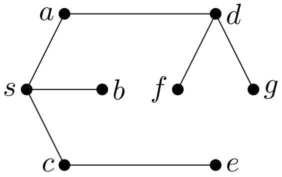
We stop when the list is empty; in a connected graph, this means all vertices have been labeled.

We are careful about the order of the list because this ensures all vertices at distance k get processed before any vertices at distance $k + 1$. This gets us to each vertex in the shortest way possible.

The shortest-path tree

Whenever we set $d(s,w) = d(s,v) + 1$ in the algorithm, let's highlight edge vw. Which edges do we highlight?



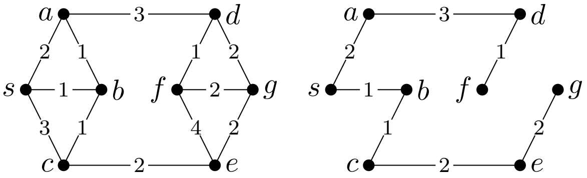


The result is a spanning tree of G. Being a tree, it has a unique $s-v$ path for every vertex v. That $s-v$ path is one of the shortest $s-v$ paths in G!

This spanning tree tree is called a **shortest-path tree** or **BFS tree**.

Weighted graphs

In most applications, not all edges are equal. Each edge has a **weight** or cost. Rather than paths with the fewest edges, we want to find paths with minimum total weight.

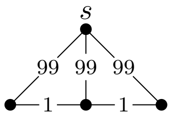


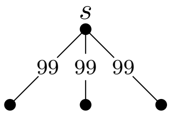
There is still a shortest-path tree in this case. However, it takes a more complicated algorithm to find it. (We might need to use paths that have more edges, but have lower total weight.)

Weight of a spanning tree

The shortest-path tree is the spanning tree to use if you want to make sure all paths from s have the least weight.

This might mean that the total weight of all the edges is much higher than it could be. Consider the example below:





The total weight of the shortest-path tree is $3 \cdot 99 = 297$ to make sure we can get anywhere from s as cheaply as possible.

But there are several spanning trees with total weight 101, instead.

Minimum-weight spanning trees

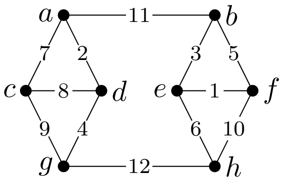
A **minimum-weight spanning tree** (or **MST** for short) of a weighted graph G is a spanning tree of G where the sum of all edge weights is as small as possible. An analogy:

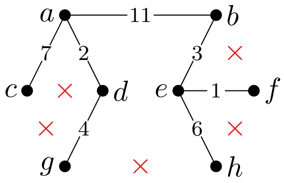
- The shortest-path tree is a railroad network built by a dictator, who wants to be able to get anywhere as quickly as possible from the capital.
- The MST is a railroad network built by a poor but democratic government, which wants to build as little as it takes to connect all cities.

Your textbook gives two algorithms for how to find an MST. We will only discuss one: Kruskal's algorithm.

Kruskal's algorithm

Kruskal's algorithm goes through the edges $e_1, e_2, \ldots, e_m$ of a graph G in order from lowest to highest weight. When we get to edge e_i, we add it to a subgraph T if it does not create a cycle in T.





Assuming G is connected, when we've processed all the edges, T is a minimum-weight spanning tree.

Proof of correctness

Initially, all we know for sure about T is that it doesn't have any cycles, we made sure of that. Why is it even a tree?

It is connected! Consider any subset $S \subseteq V(G)$. There must be at least one edge from S to $V(G) - S$... because the **first** such edge we process can always be added without fear of creating a cycle.

We still have to prove that T has minimum weight. To do this, let T^* be an MST; if there are many MSTs, let T^* be the one that agrees with T in the most edges. We will prove that T and T^* will never disagree...

Proof of minimality

Suppose that T and T^* agree on what to do with edges $e_1, e_2, \dots, e_{n-1}$, but disagree about $e_n = vw$.

If T doesn't want vw, then it would create a cycle, so T^* can't have it either. So suppose that vw is added to T, but **not** to T^*.

Then T^* has some other $v - w$ path. At least one edge e' of that path is not in T: otherwise, T would have two ways to get from v to w.

Since T and T^* disagree on e', edge e' comes after edge vw. In particular, e' has an equal or higher weight than vw.

Delete e' from T^*, and add vw instead. The result is still a tree (check that it's connected!) whose total weight is no higher, and which agrees with T in more edges. Contradiction! □