Definitions

A **graph** is two related sets: \( V(G) \), the vertex set, and \( E(G) \), the edge set. Each element of \( E(G) \) is a set containing exactly two elements of \( V(G) \).

We often visualize graphs by drawing the vertices as dots and the edges as lines between them. Here is an example of a graph with vertex set \{A, B, C, D, E, F\} and edge set \{\{A, B\}, \{B, C\}, \{A, E\}, \{C, E\}, \{A, D\}, \{C, F\}, \{D, E\}, \{E, F\}\}.

Note that we have defined an edge as a set of cardinality two: this means that a vertex cannot have an edge to itself, and there is no sense of direction in edges. Additionally, since edges are contained in a set, there can be at most one edge between any two vertices.

- If \( u, v \in V(G) \), \( u \) is **adjacent to** \( v \) if \( \{u, v\} \in E(G) \). (Note that by this definition a vertex is not adjacent to itself)
- The **degree** of a vertex is a count of the number of vertices it is adjacent to. Formally, for a vertex \( v \), \( \deg(v) = |\{u \mid \{v, u\} \in E(G)\}| \).
- The **empty graph** on \( n \) vertices has an empty edge set.
- The **complete graph** on \( n \) vertices \( K_n \) has all possible edges between vertices, that is \( E(G) = \{(u, v) \mid u, v \in V(G), u \neq v\} \). This means every pair of vertices is adjacent.
- A **path** is a sequence of vertices such that any two vertices that appear subsequently in the sequence are adjacent in the graph. For instance, in the graph above, \( (A, B, C, E) \) is a path from \( A \) to \( E \). A path is simple if no vertices are repeated.
- Two vertices are **connected** if there exists a path between them (or if they are the same vertex).
- A **subgraph** \( G' \) of a graph \( G \) is a graph such that \( V(G') \subseteq V(G) \) and \( E(G') \subseteq E(G) \). Since \( G' \) is a graph, each edge in \( E(G') \) must be between two vertices in \( G' \).
Warmup

Let $G$ be a graph such that $V(G) = \{a, b, c, d, e, f\}$ and $E(G) = \{\{b, c\}, \{a, b\}, \{d, e\}, \{c, d\}, \{d, b\}\}$

1. Draw $G$.
2. What is the degree of each vertex in $G$?
3. What are the possible simple paths from $a$ to $e$?

1.

2. $d(a) = 1$, $d(b) = 3$, $d(c) = 2$, $d(d) = 3$, $d(e) = 1$, $d(f) = 0$

3. $(a, b, d, e)$ and $(a, b, c, d, e)$

How many distinct graphs can be made on $n$ vertices?

For each pair of vertices, there exists a possible edge. Thus, there are $\binom{n}{2}$ possible edges. Each can be either in or not in the edge set of a graph, so there are $2^{\binom{n}{2}}$ possible graphs.

Prove that in any graph there exist two vertices with the same degree. *Hint*: What are the possible degrees of a vertex in a graph with $n$ nodes?

A vertex can have degree from 0 to $n - 1$. However, if a vertex has degree $n - 1$, there can’t be a vertex with degree 0 because that vertex is adjacent to every other. So, in this case, there are $n - 1$ possible values for the degree of a vertex, and $n$ vertices: by the pigeonhole principle, two must have the same degree. On the other hand, if there is no vertex of degree $n - 1$, then vertices can have degree 0 through $n - 2$: still $n - 1$ options, so by the pigeonhole principle two have the same degree.

Prove that for any graph $G$,

$$\sum_{v \in V(G)} \deg(v) = 2|E(G)|$$

The degree of a vertex is a count of the number of vertices it is adjacent to; hence, the number of edges which contain the vertex. This means that when we add up all the degrees, we are counting each edge twice since each edge is a set of two vertices.

PVDonuts is having an event where two customers that purchase a donut to split receive 10% off their order. The 35 CS22 TAs are planning on taking advantage of this offer by having each TA split donuts with 3 other TAs. Prove that this is impossible.
Let each TA be a vertex such that there exists an edge between two TA if and only if they are splitting a donut. Each of the 35 vertices has degree 3, so the sum of the degrees is 105. We just proved that this equals 2 times the number of edges, which means there are 52.5 edges. This isn’t an integer, which is impossible.

Checkpoint 1: Call over a TA.

A Tree is Nice

- A cycle is a path that starts and ends with the same vertex. A cycle is simple if there are no other repeated vertices.
- A graph is cyclic if it contains a simple cycle and acyclic otherwise.
- A graph is connected if there is a simple path between each pair of vertices (that is, all vertices are connected).
- A tree is a connected, acyclic graph.
- A forest is an acyclic graph. In other words, a forest is a set of trees.
- A leaf of a tree is a vertex with degree 1.
- A spanning tree of a graph $G$ is a subgraph $T$ of $G$ such that $V(T) = V(G)$ and $T$ is a tree.

Prove that in a tree, there is exactly one simple path between any two vertices.

The graph is connected, so there must be at least one path between any two vertices. Assume there were two paths between two vertices: there is a point where they stop being the same path, and a point where they rejoin at the end. If we put together the parts between these points, we get a cycle.

One way of defining a tree is as a minimally connected, maximally acyclic graph. This means that if we remove an edge, our graph will no longer be connected and if we add an edge, the graph will no longer be acyclic. Prove these two statements.

There is only one path between two vertices, so if we remove an edge $\{a, b\}$, there is now no way to get from $a$ to $b$. All vertices are connected, so if we add the edge $\{a, b\}$, there are now two ways to get from $a$ to $b$ and hence a cycle.

Let $T$ be a tree with at least two vertices. Prove that $T$ has at least two leaves. Hint: Start by assuming that the longest simple path in $T$ does not start and end with leaves. What contradiction can we reach?

Assume the longest simple path in $T$ has a non-leaf endpoint. As all non-leaf vertices have degree greater than 1 and our path is simple (so the endpoint only appears once, there is at least one edge incident to the ending vertex which is unused. Extend the path by also traversing this edge. The path will still be simple because if we already used the new vertex, we would now have two ways of getting from the endpoint to the vertex. So, we have a longer simple path, which contradicts our assumption that our original path was longest. So, both
end vertices are leaves.

What are all of the spanning trees of this graph?

Remove either \{1, 2\}, \{1, 4\}, \{2, 3\}, or \{3, 4\}

Checkpoint 2: Call over a TA.

### Induction on Graphs

Many graph theory questions involve induction. When doing induction on graphs, we often use a technique referred to in CS22 as "build-down" induction. This is just regular induction, with a slightly different perspective. To review, here are the steps of an inductive proof:

1. Define the predicate \(P(n)\).
2. Show that the base case is true.
3. Assume the inductive hypothesis is true. If you are using standard induction then you will assume \(P(k)\) is true for some integer \(k\). If you are using strong induction then you will assume \(P(i)\) is true for all \(i \leq k\).
4. Show that \(P(k+1)\) is true given the inductive hypothesis.

   Here is where build down induction differs from our usual perspective. A lot of the time, we start by invoking our inductive hypothesis, and manipulating the \(k\) case to get the \(k+1\) case. In build down induction, we do the following instead:

   i. Consider an arbitrary \(k+1\) case.
   ii. Alter the \(k+1\) case to obtain an \(k\) case.
   iii. Invoke the induction hypothesis, demonstrating that your property holds for the \(k\) case.
   iv. Recover the original \(k+1\) case by undoing your alteration.
   v. Prove that the property still holds despite building back up to the original \(k+1\) case.
This technique is especially useful when it is not clear how to address all \( k + 1 \) cases in general when starting from an \( k \) case. For example, if we started with a graph with \( k \) edges, where do we add an edge to obtain a general graph with \( k + 1 \) edges? There are a lot of ways to add this extra edge that we have to consider. To avoid this, we can start with an arbitrary \( k + 1 \) case. Then, we build-down to an \( k \) case, at which point we can invoke our induction hypothesis. Then, we build back up to the original \( k + 1 \) case.

Try it out on a few sample problems!

Prove that a tree with \( n \) vertices has \( n - 1 \) edges.

Base case: 1 vertex tree has 0 edges

Inductive hypothesis: assume all trees with \( k \) vertices have \( k - 1 \) edges.

Inductive step: Consider a tree with \( k + 1 \) vertices. There must be some leaf, which has one edge. Remove the leaf and its edge. The resultant graph will still be acyclic, since removing an edge cannot create a cycle, and is connected because the edge we removed only connected the removed vertex to the rest of the graph. So, we now have a tree with \( k \) vertices, which by the IH has \( k - 1 \) edges. Put the removed vertex and edge back in, and we now have \( k \) edges. As our \( k + 1 \) tree was arbitrary, all trees with \( k + 1 \) vertices have \( k \) edges.

Prove that \( K_n \) has \( \frac{n(n-1)}{2} \) edges. Note that we could do this without build-down induction, but you should use it here for practice.

Base case: \( K_1 \) has 0 edges, which is \( \frac{1 \times 0}{2} \)

IH: Assume \( K_k \) has \( \frac{k(k-1)}{2} \) edges

IS: Consider \( K_{k+1} \). We can get \( K_k \) by removing a vertex and all its incident edges, of which there are \( k \). By IH, \( K_k \) has \( \frac{k(k-1)}{2} \) edges. If we add back in the vertex and edges, we get \( \frac{k(k-1)}{2} + k = \frac{k(k-1)+2k}{2} = \frac{k(k+1)}{2} \) edges as needed.

Checkpoint 3: Call over a TA.

Modeling Relations

In lecture, we briefly looked at bipartite graphs. A graph is bipartite if we can partition its vertices into two sets such that all edges are between the two sets. The example we looked at was the relation diagram from one of the first days of class.
In general, given a relation on sets $A$ and $B$, we can construct a bipartite graph that represents it. We simply make each element in $A$ and each element in $B$ a vertex, and draw an edge between two vertices if those elements are related.

Suppose our relation is a function. In addition to being bipartite, what property will the constructed graph have?

$\forall a \in A, \ deg(a) = 1$

How about an injection? A surjection? A bijection?

| Injection: function property + $\forall b \in B, \ deg(b) \leq 1$. Surjection: function property + $\forall b \in B, \ deg(b) \geq 1$. Bijection: $\forall v \in V(G), \ deg(v) = 1$. |

This representation can make sense for functions since often, the domain and codomain are distinct. However, if we have a relation on just one set, we would need two vertices for each element of the set to make this representation work. A lot of the time, we work with equivalence relations, where the domain and codomain must be the same set.

To fix this, we can use directed graphs. In a directed graph, the edge set is composed of ordered pairs of vertices instead of sets. This means that a pair of vertices might have two edges between them: one in each direction. It also means we can have self-loops: an edge from a vertex to itself.

Here are some example directed graphs: note that bidirectional edges can be represented either with two arrows, like in the first picture, or with a bidirectional arrow, like in the second picture.

Note that we have not define directed graphs in lecture, and all graphs in this class are undirected unless otherwise specified.

Draw a graph representing the following relations. Which of the equivalence relation properties (reflexivity, symmetry, or transitivity) does each have?
a. The relation on the set \(\{1, 2, 3, 4, 5\}\) where two numbers are related if the absolute value of their difference is 2.

b. The "divides" relation on the integers from 2 to 10.

c. The relation on the integers from 1 to 9 where \(a\) and \(b\) are related if \(a \equiv b \pmod{3}\).

What do each of the properties of equivalence relations mean for a graph?

Reflexive: each vertex has a loop, symmetric: all edges are bidirectional, transitive: edge from \(a\) to \(b\) and \(b\) to \(c\) means edge from \(a\) to \(c\)

What does the graph of an equivalence relation look like? What does this tell us about equivalence classes?

There are several components, each of which has an edge in each direction between every two vertices (in addition to every vertex having a self-loop). These components are the equivalence
classes.

Checkpoint 4: Call over a TA.