3.1 Decomposition and Special Graphs

The **complement** $\overline{G}$ of a graph G has $V(G)$ as its vertex set. Two vertices are adjacent in $\overline{G}$ iff they are not adjacent in G. A graph G is called **self-complementary** if G is isomorphic to $\overline{G}$. A **decomposition** of a graph is a list of subgraphs such that each edge appears in only a single subgraph in the list.

There are a couple specific names for certain graphs that we might talk about (and have already talked about this semester) repeatedly:
Triangle: A three vertex cycle C_3 or clique K_3
Claw: The complete bipartite graph $K_{1,3}$
Note: the claw is also a **star graph**, which are the class of complete bipartite graphs $K_{1,n}$. The claw is star graph S_4
Also note: the book gives several other examples in 1.1.35; we probably won't be talking much specifically about the other ones.

3.2 Walks and Connectivity

A **walk** is a list of of vertices and edges (e.g., v_0, e_5, v_6, e_1, v_2) such that each listed edge connects the preceding and proceeding listed vertices. The list begins and ends with unique vertices. A **trail** is a walk with no repeated edges. A **path** has no repeated edges or vertices. A u, v-walk and u, v-trail begin with vertex u and end with vertex v. A u, v-path is a path with endpoint vertices u and v having degree 1 and all other vertices being internal. The **length** of a walk/trail/path is the number of contained edges. A walk is **closed** if the start and end vertices are the same. *Random walks* are performed by starting at a given vertex, moving to an adjacent vertex selected at random, then iteratively continuing this procedure from the newly selected vertex. Random walks have a number of interesting uses and properties; we'll talk a little more about these later.

A graph G is **connected** if for every $u, v \in V(G)$ there is a path connecting u and v. Otherwise G is disconnected. A **connected component** of G is a *maximal* connected subgraph. We say a component is **trivial** when it consists of a single vertex and no edges. Otherwise, the component is **nontrivial**. A **cut-edge** or **cut-vertex** are the edges or vertices that, when removed from G, increase the number of connected components.

We will also talk more about connectivity later in the course.

3.3 Induction

In this class, we'll often be proving properties about graphs using **induction**.

Review: **Weak Induction** as a proof method. You have probably seen this in an earlier class. Consider a natural number n, let $P(n)$ be a mathematical statement. If properties 1 and 2 below hold, then $P(n)$ is true for all $n \in \mathbb{N}$.

1. $P(1)$ is true
2. for $k \in \mathbb{N}$, if $P(k)$ is true, then $P(k+1)$ is true

(1.) is the **basis step** and (2.) is the **inductive step**. The inductive step includes our **induction hypothesis**, which is the assumption that our current step of $P(k)$ is true. The basis step might utilize $P(0)$ or some other integer.

Weak inductive proofs then work to show that if e.g. $P(1)$ is true, and $P(k) \implies P(k+1)$ is true, then $P(1) \implies P(1+1)$, $P(1+1) \implies P(1+1+1)$, etc. So $P(k)$ is true for all natural numbers greater than or equal to the basis.

Prove that: $2^1 + 2^2 + \ldots + 2^n = 2^{n+1} - 2$

Basis Step: $P(n=1) = 2^1 = 2^2 - 2 = 2$ ✓
Induction Step: $P(n=k+1) = 2^1 + 2^2 + \ldots + 2^k + 2^{k+1}$
$= [2^1 + 2^2 + \ldots + 2^k] + 2^{k+1}$
$= 2^{k+1} - 2 + 2^{k+1}$
$= [2^{k+1} + 2^{k+1}] - 2$
$= 2 \times 2^{k+1} - 2$
$= 2^{k+2} - 2$
$= 2^{(k+1)+1} - 2$
$= 2^{n+1} - 2$ ✓

However, a number of graph theoretic proofs require **Strong Induction**:

1. $P(1)$ is true
2. for $k > 1$, if $P(k)$ is true, then $P(n)$ is true for $1 \leq k < n$

Instead of limiting ourselves to $P(n = k+1)$ in our inductive step, we assume that all $P(k)$ less than our n are true. When we're working with graphs, we often do induction on the number of vertices/edges in some graph G. With strong induction, we can establish a relation between $G(k)$ and $G(n)$ as the difference of a larger subgraph beyond just a single vertex or a single edge. This is often required, as a key aspect of induction is showing how $P(k)$ being true implies $P(n)$ is true. It is not always possible to do so with simple edge or vertex deletion.

When using weak induction, we would need to consider all possible "structural" ways that adding a single vertex/edge to get from $P(k)$ to $P(n)$ might impact the property we're trying to prove. Because of the *combinatorial explosion*[1] of possible graph configurations, this becomes quite unwieldy quite quickly. In addition, it might not be possible to realize all possible graphs that fit our assumptions by only adding a single vertex or edge, preventing us from fully proving the statement.

With strong induction, we can often specifically select a vertex or edge (or subgraph) to remove from $P(n)$ to get to $P(k)$; we then only need to consider the impact of that specific structure within our proof. This might be confusing, but we'll be doing a number of proofs throughout the rest of the class to differentiate and emphasize this key difference between *weak* and *strong* induction. I will refer to the selection of vertices/edges for deletion/addition during an inductive step as our inductive **construction**.

Note that there are several other variations of induction as a proof technique[2]. You might have seen them in your other classes. Discussion of these is generally beyond the scope or relevance of this course.

3.4 More on Walks and Cycles

As defined earlier, the **length** of a walk is the number of edges that the walk traverses. We might also equivalently say that a walk takes some number of **hops**. An **even** walk/path/trail/cycle has an even length, or an even number of edges. Likewise, an **odd** walk/path/trail/cycle has an odd length, or takes an odd number of hops. An **even graph** has all vertex degrees even. An **even vertex** has an even degree. Likewise, we will also use the terms **odd graph** and **odd vertex**.

Prove with strong induction: Every closed odd walk contains an odd cycle.

Basis Step: $P(n = 1)$: a closed length 1 walk is a single self loop, hence is an odd cycle of length 1
Induction Step: $P(n > 1)$: We use the *induction hypothesis* to assume walks of length $k < n$ have an odd cycle. Consider walk W of length n, where n is odd. If W is odd and has no repeated vertices, then W is an odd cycle by itself. Otherwise, some vertex v is repeated. Consider breaking W into walks W_1 and W_2 originating from v. As the length of W is odd, then W_1 must be odd and W_2 even. We say that the length of W_1 is $k < n$,

[1] 💣💣💣💣💣💣💣

[2] `https://en.wikipedia.org/wiki/Mathematical_induction`

which by our inductive hypothesis must have some odd cycle. As W_1 is a subpath of W, then W must also contain an odd cycle.

This is the power of strong induction in action!!!!!!!!!

3.5 Breaking it Down

We can break down the proof process a little more explicitly. First, we can note that we essentially want to prove equivalence between the two classes of {closed odd walks} and {closed odd walks containing an odd cycle}. The question of how to approach this proof is one of determining any consistent relationship between closed odd walks and cycles.

We can intuit a few things in this regard:

1. A closed walk that doesn't repeat any vertices is simply a cycle, so an odd closed walk without repeats is an odd cycle. Note that if any edge is repeated, the endpoint vertices of that edge will be repeated. And since we can have a vertex repeated without edges repeated, the {vertex repeated} property is *stronger*, meaning that the class of walks having the {edge repeated} property is a subset of the class of walks having the {vertex repeated} property.
2. Following the above, we therefore only can further restrict ourselves to the class {closed odd walks with some vertex repeated}. Note how we make sure to consider all possible graph configurations within the {closed odd walks} class; often, more than a simple binary spit is needed.
3. From here, we might next need to consider all the possible ways that an odd cycle could be embedded in a closed odd walk with at least one repeated vertex. However, we have the power of induction at our disposal. Note how if we split the walk in twain, we have two subwalks, one of which must be odd due to the power of parity (we'll cover parity-based arguments in more detail later).
4. Since we have a subgraph of our original graph (or walk) that falls into the same property class as our original graph, we can then effectively recurse back to our first intuited item with our smaller graph (or walk). Explicitly repeating this process will eventually, due to the finite size of our original walk, result in a closed odd walk that does not have any repeating vertices or is itself a closed odd walk of length 1 (our basis case). Hence, either way it is an odd cycle. This is the basic recursive logic of all inductive proofs.

Thankfully, our inductive proof step+hypothesis allows us to avoid explicitly considering a full recursion back to our base case. We just need to be sure that we consider *all possible cases* within our given graph property class and that our inductive *construction* results in a graph that falls into the exact same graph property class.