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 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'll talk more about connectivity later in the course.

3.3 Induction

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

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 $P(n = k + 1)$ 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 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's not always possible to do so with simple edge or vertex deletion.

By using only weak induction, we'd 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. With strong induction, we can often specifically select a vertex or edge 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.

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

The **length** of a walk is the number of edges that it 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** or **odd vertex**.

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

Basis Step: $P(l = 1)$: a closed length 1 walk is a single self loop, hence is an odd cycle of length 1
Induction Step: $P(l = n > 1)$: 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 $k < n$, 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!!!!!!!!!

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

Note that in the above we had to **Consider the Cases** of possible configurations of the walk. Generally, we will need to do this for a good portion of the proofs, inductive or not. The above proof has two cases that we needed to explicitly consider:

- **Case 1:** Walk W has no repeated vertices.
- **Case 2:** Walk W has repeated vertices.

Some of the proofs we do will have many more cases and even subcases that need consideration. When approaching any proof, try to come up with all possible configurations, lengths, sizes, etc. of the graph, subgraph, walk, etc. whose properties that you are attempting to prove or disprove.

Parity Arguments utilize the notion of *parity* to prove or disprove some statement. Note how in the above proof we made use of the additive properties of integers such that:
$odd + odd = even$
$even + even = even$
$odd + even = odd$

This is called integer *parity*. In this class, we will use it in the form of *parity arguments*, which utilize parity prove or disprove something on countable properties. We might use it in tandem with induction or solely on its own, usually when considering edges, vertices, or some other countable property of graphs.

Necessity and Sufficiency is used to prove *equivalence relationships*, such as we'll soon show that *a graph is bipartite iff it has no odd cycle* (note: iff $\rightarrow$ "if and only if"). To generally prove an equivalence relationship, we can show that the given conditions (A iff B) are both necessary and sufficient; i.e., by proving that if condition A implies condition B and if conditions B implies condition A, we prove their equivalence. For any equivalence relationship, knowledge about one condition gives us knowledge about the other condition – so if we know a graph has no odd cycles, we also know that it therefore must be bipartite. As you'll find in this class, a lot of equivalence proofs on graphs tend to have one of half of the equivalence being quite easy to prove[3]

Prove with necessity and sufficiency: A graph is bipartite iff it contains no odd cycle.

Necessity: It is *necessary* for a bipartite graph to have no odd cycles.
Graph G is bipartite $\implies$ G contains no odd cycles.

Sufficiency: A graph having no odd cycles is *sufficient* in demonstrating that the graph is bipartite.
Graph G contains no odd cycles $\implies$ graph G is bipartite.

[3]The ol' "easy one way but harder the other", as Slota terms it.