## 4.1 Proof Techniques

Let's go over in more detail some additional proof techniques. Recall our proof from last class, where we used induction to prove that all odd walks contain an odd cycle.

Note that in the proof 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. Usually, we do this to determine broad structural subclasses within our given graph property class. We can then make simpler proof arguments related to each of these subclasses. As we saw, the 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 sub-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. Quickly figuring out which structural properties to consider generally comes with practice.

Our next technique, **Parity Arguments**, utilizes the notion of *parity* to prove or disprove some statement. Note how in the above proof we also 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.

While not as widely used in graph theory as some other 'proof techniques', the **pigeonhole principle** still occasionally makes an appearance. Basically, the pigeonhole principle states that if we have $x$ containers in which we wish to place $y$ items, where $x < y$, then at least one container will contain at least two items.

**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"). Proving equivalences basically means that were proving two graph classes defined by differing properties actually contain the same set of possible graph configurations; i.e., the graph classes are equivalent and fully overlap in terms of all possible graph realizations.

To generally prove an equivalence relationship, we can show that the given properties or conditions ($A$ iff $B$) are both necessary and sufficient; i.e., by proving that if property $A$ implies property $B$ and if property $B$ implies property $A$, we prove their equivalence. For any equivalence relationship, knowledge about one property class gives us knowledge about the other property class – 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 direction of the equivalence being quite easy to prove[1]

- We can 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.

The **contrapositive** of a logical implication can often be used to simplify an approach for a proof. Consider the logical implication $P \implies Q$. This is actually equivalent to $\neg Q \implies \neg P$. E.g., ($G$ has no odd cycle) $\implies$ ($G$ is bipartite) is logically equivlant to ($G$ is not bipartite) $\implies$ ($G$ has an odd cycle). When used in conjuction with proving an equivalence, we can actually say that $P \iff Q$ is equivalent to $\neg P \iff \neg Q$.

## 4.2 Eulerian Circuits

Recall the Königsberg bridge problem we discussed in the first class. The problem essentially reduces to whether or not its possible to begin at some vertex, traverse every edge exactly once, and return to that starting vertex. In other words, a closed trail exists on $G$ that contains all $e \in E(G)$.

A graph is **Eulerian** if such a trail exists. A closed trail is a **circuit** when there isn't any specific start/end vertex specified. An **Eulerian circuit** (or an **Euler Tour**) in a graph is the circuit or trail containing all edges. An **Eulerian path** in a graph is a path containing all edges, but isn't closed, i.e., doesn't start or end at the same vertex. We'll focus discussion on Eulerian circuits today. The following two proofs will let us demonstrate a characterization of Eulerian graphs.

Prove: If every vertex in $G$ has at least a degree of 2, then $G$ has a cycle. For this proof, we can construct an argument using the **extremal principal**. We'll talk more about this later.

[1]The ol' "easy one way but harder the other", as Slota terms it. Other people might call it TONCAS - The Obvious Necessary Conditions are Also Sufficient, but they are wrong.

Prove: A graph is Eulerian iff it has at most one nontrivial component and is an *even graph*. An even graph contains vertices which all have an even degree.

How might we find an Eulerian circuit?

One approach is Fleury's algorithm:

---

$T \leftarrow \emptyset$ ▷ Initialize Eulerian circuit
$G' \leftarrow G$
Start at any vertex $v$
**while** $G' \neq \emptyset$ **do**
    Select at edge $e$ to travel along, where $(G' - e)$ is not disconnected
    $T \leftarrow e$
    $G' \leftarrow (G' - e)$
**return** $T$

---

Note that you aren't going to be required to know or use this algorithm. But think about potential proofs to prove the *correctness* of the algorithm. How would you show whether the algorithm returns a full tour for every valid input?

## 4.3 Extremal Problems

An **extremal problem** asks for the maximum or minimum value of a function over a class of objects. Consider possible proofs for the below extremal problems related to degrees and connectivity.

- Prove the minimum number of edges in a connected graph is $(n - 1)$.
- Prove a graph must be connected if $\delta(G) \geq \frac{(n-1)}{2}$.

We'll often use *extremal arguments* (commonly called the **extremal principle**) as another proof technique throughout the course. The extremal principle states that within some set of countable and orderable values, there exists some item(s) with a maximum value and some item(s) with a minimum value.

Recall from the first proof we worked through today – "Let $P$ be a *maximal* path in $G$". By selecting an item with the most extreme property, we can often use information inferred from that item as a starting point to understand the more general case.

We'll consider many other minimal or maximal graphs, subgraphs, and other properties as methods to solve various proofs.