1 Induction

Review: **Induction** as a proof method. 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 **induction step**. The basis step might utilize $P(0)$ or multiple k for $P(k)$.

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

2 More on Walks and Cycles

Use induction to prove that every u, v-walk W of length l contains a u, v-path.

Basis Step: $P(l = 1)$: A u, v-walk of length 1 is a u, v-path of length 1 if $u \neq v$. If $u = v$, then the length 1 u, v-walk contains a path of length 0.
Induction Step: $P(l = k + 1)$: We use the induction hypothesis to assume walks of length k have a path linking their endpoints. If $u \neq v$, we have some final walk edge e in W. Removing e creates W', which we know by our induction hypothesis contains a path P' since it's of length k. If P' does not contain v then we can extend P' by a single edge to create a u, v-path. If P' does contain v, then P is simply a subpath of P'.

An **even** walk/path/trail/cycle has an even length, or number of edges. Likewise, an **odd** walk/path/trail/cycle has an odd length, or number of edges. An **even graph** has all vertex degrees even. A vertex is even if it has an even degree or odd when it has an odd degree.

Prove: Every closed odd walk contains an odd cycle.

Prove: A graph is bipartite iff it has no odd cycle.

3 Eulerian Circuits

A graph is **Eulerian** if it has a closed trail (no repeated edges, start and end vertices are the same) containing all edges in the graph. A closed trail is a **circuit** when there isn't any start/end vertex specified. An **Eulerian circuit** 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.

Prove: If every vertex in G has at least a degree of 2, then G has a cycle.

Prove: A graph is Eulerian iff it has at most one nontrivial component and is an even graph.

How might we find an Eulerian circuit?
Fleury's algorithm:

```
E ← ∅                                                        ▷ Initialize Eulerian circuit
G' ← G
Start at any vertex v
while G' ≠ ∅ do
    Select at edge e to travel along, where (G' − e) is not disconnected
    E ← e
    G' ← (G' − e)
return E
```

Hierholzer's algorithm:

```
E ← ∅                                                        ▷ Initialize Eulerian circuit
Select at any vertex v
E ← randomly traverse unvisited edges until you arrive back at v
G' ← G − E
while G' ≠ ∅ do
    Select any vertex u in E that has incident edges remaining in G'
    P ← randomly traverse unvisited edges in G' until you arrive back at u
    E ← P                                                  ▷ Insert new path into circuit
    G' ← G − P
return E
```

Which is the computational complexity of each approach?