5.1 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.

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?

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** $T$
```

Hierholzer's algorithm:

```
$T \leftarrow \emptyset$                                         ▷ Initialize Eulerian circuit
Select at any vertex $v$
$T \leftarrow$ randomly traverse unvisited edges until you arrive back at $v$
$G' \leftarrow G - T$
while $G' \neq \emptyset$ do
    Select any vertex $u$ in $T$ that has incident edges remaining in $G'$
    $P \leftarrow$ randomly traverse unvisited edges in $G'$ until you arrive back at $u$
    $T \leftarrow P$                                             ▷ Insert new path into circuit
    $G' \leftarrow G - P$
return $T$
```

You aren't going to be required to know or use these algorithms. But think about the computational complexity of each approach and why one might make for a better implementation than the other.

5.2 Degrees

As mentioned previously, we're going to use variables n and m regularly as:

$$n = |V(G)|, m = |E(G)|$$

As we've discussed, the **degree** of a vertex is the number of incident edges. We write degree of vertex v as $d(v)$ or sometimes d_v. For a graph G, the maximum degree is $\Delta(G)$ and the minimum degree is $\delta(G)$. A graph is **regular** if $\Delta(G) = \delta(G)$. A graph is k-**regular** if $k = \Delta(G) = \delta(G)$.

The degree sum formula shows that the sum of the degrees of all vertices in a graph is always even:

$$\sum_{v \in V(G)} d(v) = 2m$$

So it follows that there can only be an even number of vertices of odd degree in G.

The average degree of a graph G is $\frac{2m}{n}$. Therefore:

$$\delta(G) \leq \frac{2m}{n} \leq \Delta(G)$$

For directed graphs, we've already seen that we consider both **out degree** ($d^+(v)$) or **in degree** ($d^-(v)$) separately. We likewise have minimum and maximum out and in degrees:

$$\delta^-(v), \delta^+(v), \Delta^+(v), \Delta^-(v)$$

And our degree sum formula for digraphs:

$$\sum_{v \in V(G)} d^+(v) = |E(G)| = \sum_{v \in V(G)} d^-(v)$$

5.3 Extremal Problems

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

Prove the minimum number of edges in a connected graph is $(n-1)$. Note: You already proved this on the last quiz, but there are many more ways to prove it than just using induction.

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 a proof technique through the course. Recall from the first proof we worked through today – "Let P be a *maximal* path in G". We'll consider many other minimal or maximal graphs, subgraphs, and properties as methods to solve various proofs.

5.4 Graphic Sequences

The **degree sequence** of a graph is the list of vertex degrees, usually in non-increasing order: $d_1 \geq d_2 \geq \ldots \geq d_n$.

A **graphic sequence** is a list of nonnegative numbers that is the degree sequence of a simple graph. A simple graph G with degree sequence S *realizes* S.

A sequence $S = \{d_1, d_2, \ldots, d_n\}$ is a graphic sequence iff sequence $S' = \{d_2 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n\}$ is a graphic sequence, where $d_1 \geq d_2 \geq \ldots \geq d_n$ and $n \geq 2$ and $d_1 \geq 1$. This is called the **Havel-Hakimi Theorem**. We can use this general idea to also create (*realize*) a graph using a given graphic sequence.

For time consideration, we're not going to go over the proof in class, so go through the book or use other online resources to understand it. A couple relevant youtube videos are also listed below if you're interested:

`https://www.youtube.com/watch?v=aNKO4ttWmcU`
`https://www.youtube.com/watch?v=iQJ1PFZ4gh0`