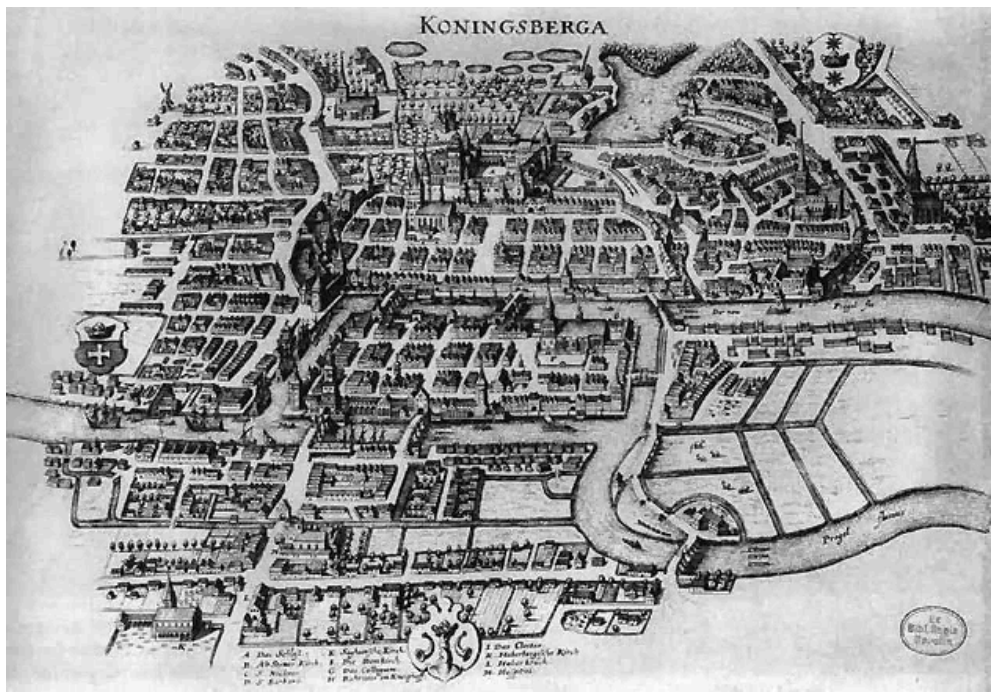
Module 01: Königsberg Bridge Puzzle

Advanced Topics in Network Science

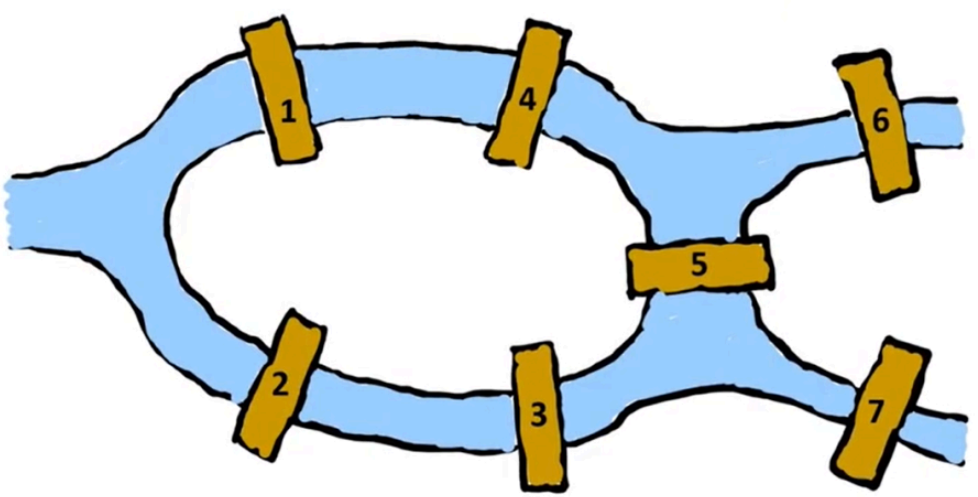
Sadamori Kojaku

skojaku@binghamton.edu



Quiz

1. Explain in your own words what is the conditions for a graph to have an Euler path, and why?
2. What is the minimum number of bridges required to make the Königsberg bridge puzzle have an Euler circuit? Explain your reasoning.

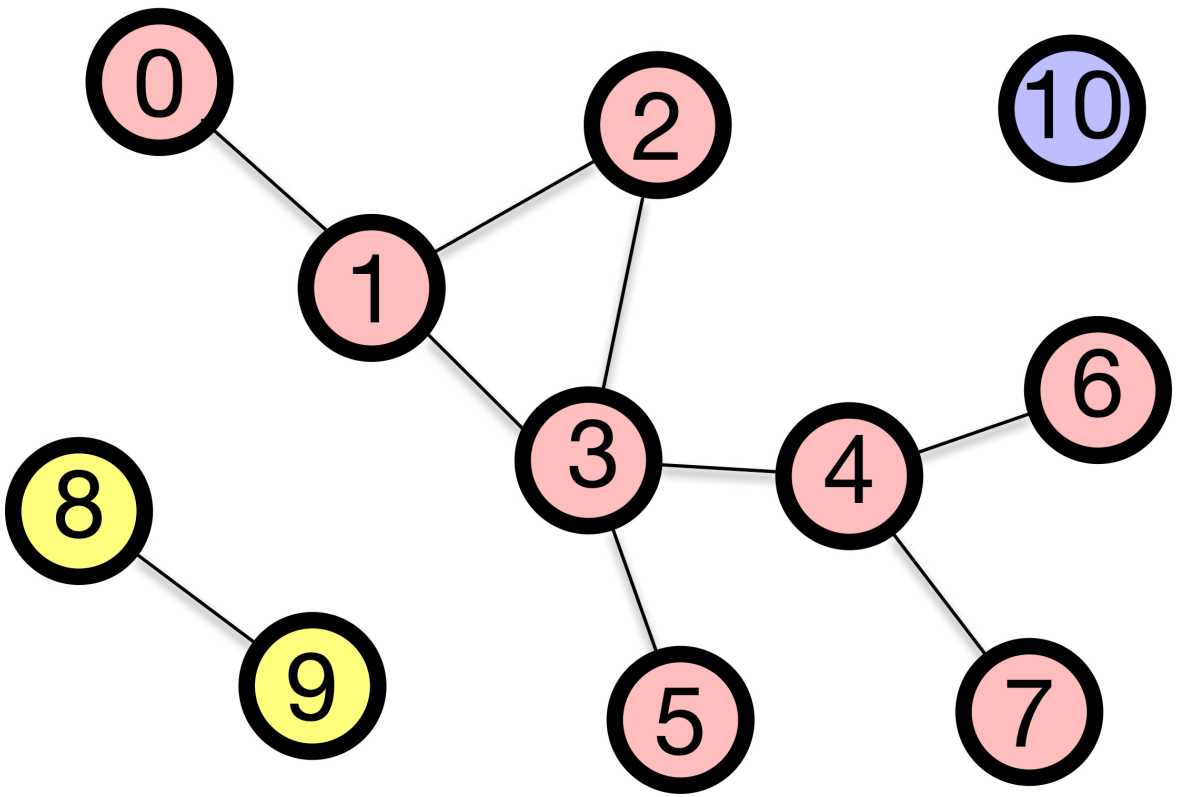


Menue

1. Connected components
2. Representation of networks
3. Coding exercise: set up
4. Assignment set up

Connected Component

Definition: A connected component is a maximal set of nodes where every node can reach every other node within that set.



Question: Is a single node a connected component?

In large real-world networks, what would you expect 🤔?

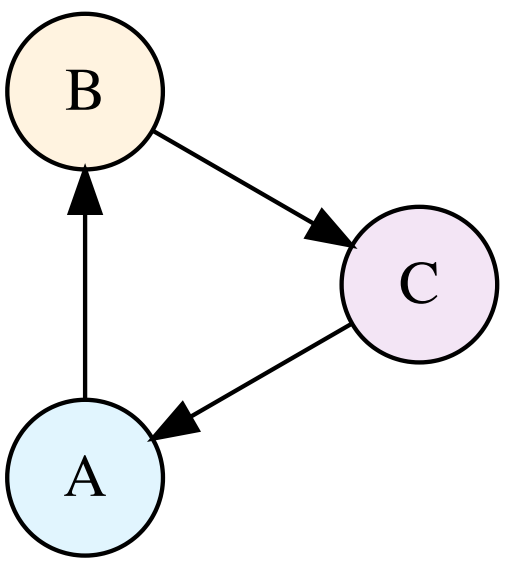
- Many tiny components of 2-3 nodes?
- One huge component containing most nodes?
- All components roughly equal size?

- Many networks contain a giant component that contains a significant fraction of all nodes in the network
- **Definition:** A giant component is a connected component where almost every node in the network is reachable from any other node in the component.

Context: What if edges have direction? (Think Twitter follows, webpage links)

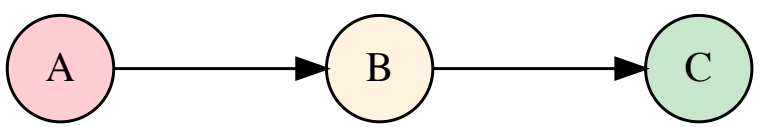
Strongly connected 💪

Every node can reach every other node following edge directions



Weakly connected 🤝

Connected if we ignore edge directions



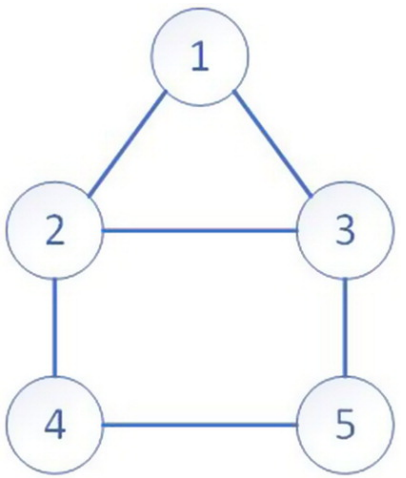
Question: Is every strongly connected component also weakly connected?

Coding Networks in Python 

Given any network, how would you represent it in a computer?

Three ways to represent the same network:

1. **Edge Table** - List of connections
2. **Adjacency List** - Each node's neighbors
3. **Adjacency Matrix** - Grid of 1s and 0s



5 nodes, 6 edges

Edge Table: The Direct Approach 📋

Simply list every connection:

```
edges = [
    (0, 1),  # Node 0 connects to Node 1
    (0, 2),  # Node 0 connects to Node 2
    (1, 2),  # Node 1 connects to Node 2
    (1, 3),  # Node 1 connects to Node 3
    (2, 4),  # Node 2 connects to Node 4
    (3, 4)   # Node 3 connects to Node 4
]
```

- *How would you count the degree of node 1 from this list?*
- *How would you find the neighbors of node 1?*

Adjacency List: Neighborhood Map 

Each node knows its neighbors:

```
neighbors = {
    0: [1, 2],      # Node 0 connects to nodes 1,2
    1: [0, 2, 3],   # Node 1 connects to nodes 0,2,3
    2: [0, 1, 4],   # Node 2 connects to nodes 0,1,4
    3: [1, 4],      # Node 3 connects to nodes 1,4
    4: [2, 3]       # Node 4 connects to nodes 2,3
}
```

- *How would you count the degree of node 1 from this list?*
- *How would you find the neighbors of node 1?*

Adjacency Matrix: The Math Way 📐

Grid where entry $(i, j) = 1$ if connected:

```
import numpy as np

matrix = np.array([
    [0, 1, 1, 0, 0],  # Node 0: connects to 1,2
    [1, 0, 1, 1, 0],  # Node 1: connects to 0,2,3
    [1, 1, 0, 0, 1],  # Node 2: connects to 0,1,4
    [0, 1, 0, 0, 1],  # Node 3: connects to 1,4
    [0, 0, 1, 1, 0]   # Node 4: connects to 2,3
])
```

- *How would you count the degree of node 1 from this matrix?*
- *How would you find the neighbors of node 1?*

Implementing Euler's Theorem 🧮

```python
def has_euler_path(adjacency_matrix):
    # Calculate degrees
    degrees = adjacency_matrix.sum(axis=1)

    # Count odd degrees
    odd_count = sum(1 for d in degrees if d % 2 == 1)

    # Euler's condition
    return odd_count == 0 or odd_count == 2
```

Do you agree with this?

The Missing Piece: Connectivity ⚖️

Revisit

An Euler path exists if and only if:

1. **The graph is connected** ← We forgot this!
2. **Exactly 0 or 2 nodes have odd degree**

Module 01 Review 🎯

- **Euler's legacy** (1736)
  - Abstraction over physical details
  - Focus on relationships → Birth of graph theory
- **Euler's theorem**
  - Euler path exists if and only if the graph is connected and exactly 0 or 2 nodes have odd degree
- **Key concepts**:
  - Path, walk, trail, circuit, cycle
  - Connected component, giant component, degree
- **Computational Representation**
  - Edge table, adjacency list, adjacency matrix

- **Walk**
  - Any sequence of connected nodes
- **Trail**
  - Walk without repeated edges
- **Path**
  - Walk without repeated nodes
- **Circuit/Cycle:**
  - Closed versions that return to start
- Is a path always a trail 🤔?
  - Yes. Path does not repeat edges.

- **Connected**
  - A network where there is a path between every pair of nodes
- **Connected component**
  - A maximal set of nodes where every node can reach every other node within that set
- **Giant component**
  - A connected component where almost every node in the network is reachable from any other node in the component
- **Strongly connected**
  - A network where every node can reach every other node following edge directions
- **Weakly connected**

- **Euler path**
  - A path that visits each edge exactly once
- **Euler circuit**
  - An Euler path that starts and ends at the same node
- What is the condition for the existence of an Euler circuit?
  - Graph is connected and all nodes have even degree.

Representation of Networks

Edge Table

```
edges = [
    (0, 1),
    (1, 2),
    (2, 3)
]
```

Best for: Storage, I/O

Adjacency List

```
neighbors = {
    0: [1],
    1: [0, 2],
    2: [1, 3]
}
```

Best for: Neighbor search

Adjacency Matrix

```
matrix = np.array([
    [0, 1, 0],
    [1, 0, 1],
    [0, 1, 0]
])
```

Best for: Math operations

My recommendation:

Use edge table for saving the network data. Use (sparse) adjacency matrices for analysis.

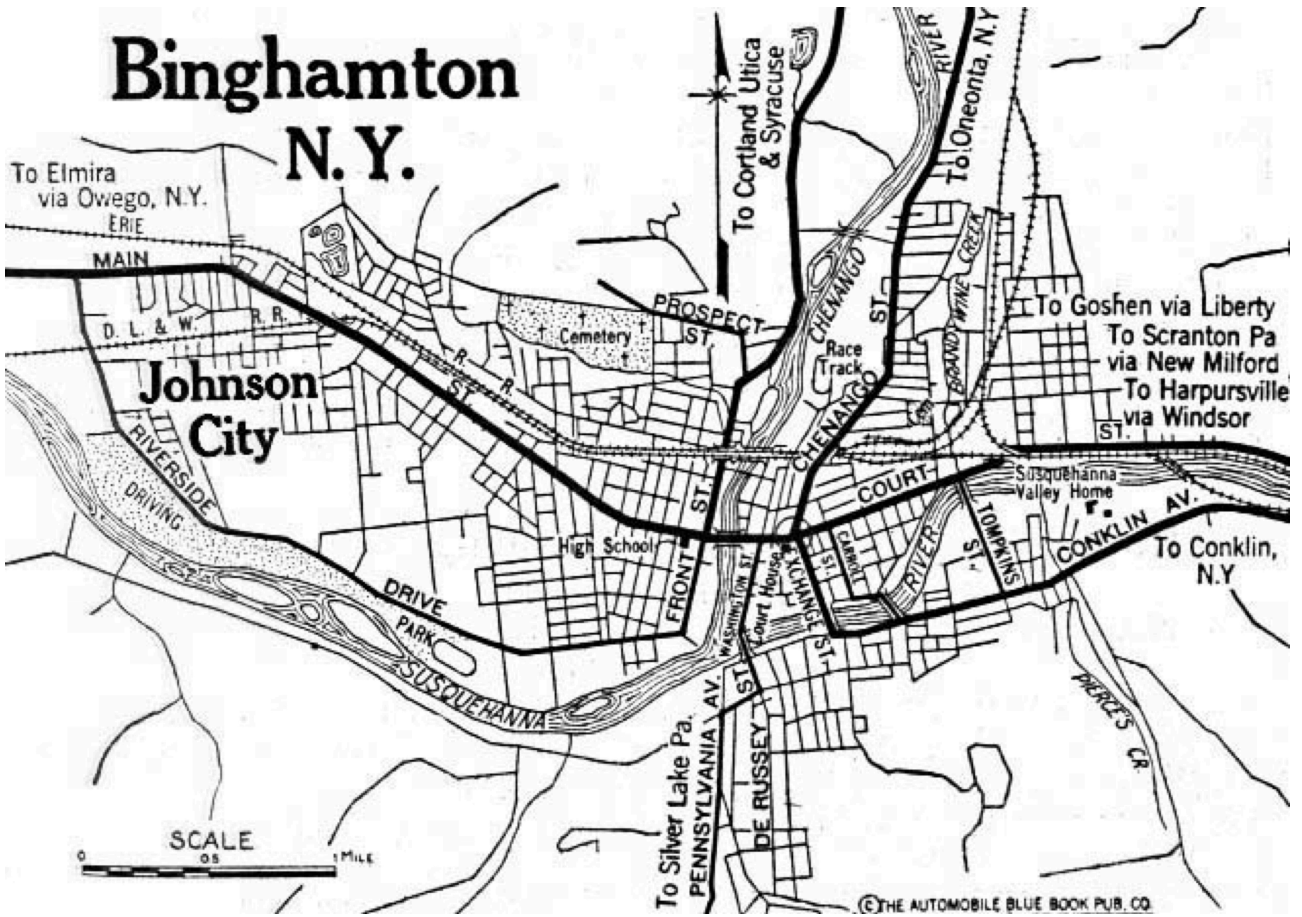
Coming up in Module 02:

Small world networks

Almost all 8 billion people on the planet are your friends of friends of friends of friends of friends of friends.

Coding Exercise

- Represent the bridge network as either an edge table, an adjacency list, or an adjacency matrix in Python.
- Write a function that takes the network data and returns the existence of an Euler path.



Assignment set up

- https://classroom.github.com/a/Sdey2VNh.
- Preparations: Install Docker Desktop and GitHub Desktop
- A simple workflow using GitHub Web UI
- A recommended workflow using Docker and VS Code.