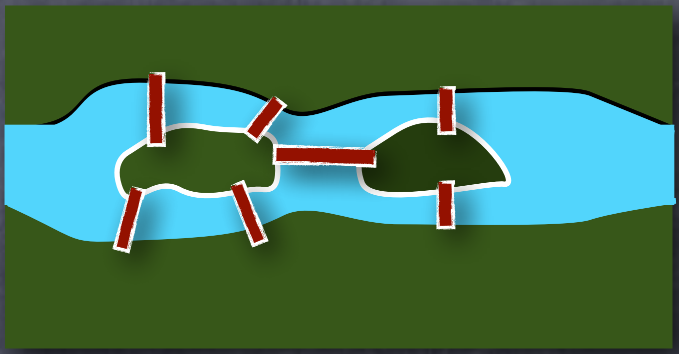
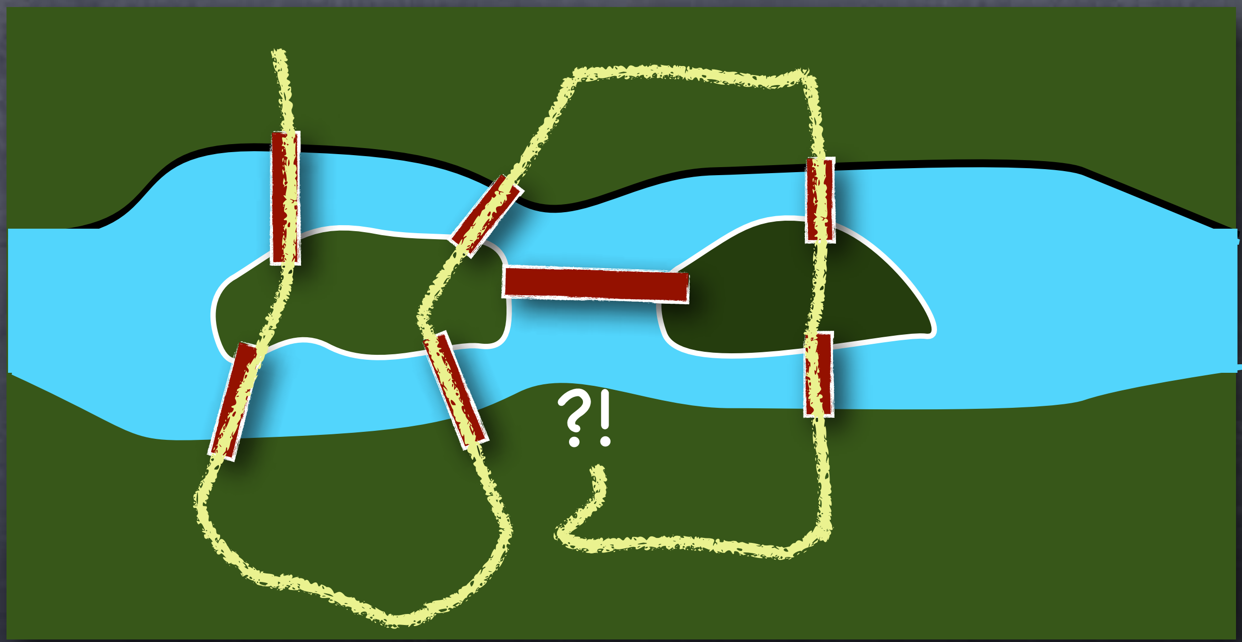
Graphs

Walks and Paths



Bridges of Königsberg

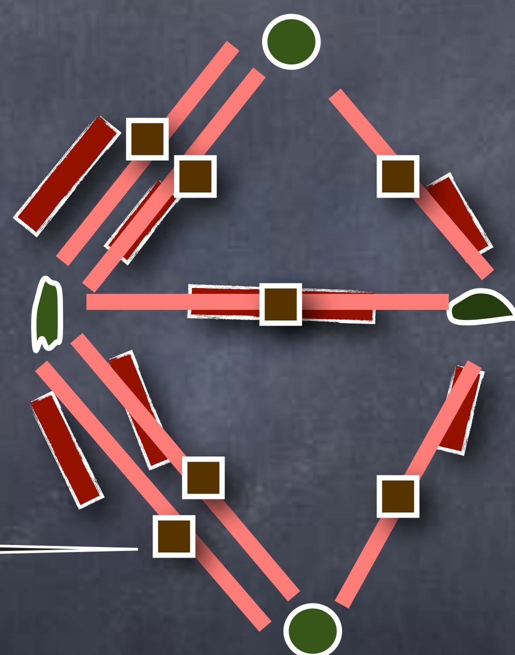
- Cross each bridge exactly once



- Impossible! But how do we know for sure?

Bridges of Königsberg

- Cross each bridge exactly once



Add a node for each bridge too, if we want it to be a simple graph

- Impossible! But how do we know for sure?

Walks, Paths & Cycles

- A **walk** (of length k, $k \geq 0$) from node a to node b is a sequence of nodes $(v_0, v_1, \ldots, v_k)$ such that
  - $v_0 = a$, $v_k = b$
  - for all $i \in \{0,\ldots,k-1\}$, the edge $\{v_i, v_{i+1}\} \in E$
- Length is the number of edges in a walk. Could be 0.
- If a walk has no node repeating, then it is called a **path**
- If a walk of length $k \geq 3$ has $v_0 = v_k$, but no other two nodes are equal, then it is called a **cycle**
  - Note: we require a cycle to be of length at least 3
- A graph is acyclic if it has no cycles (i.e., no C_k is a subgraph of G)

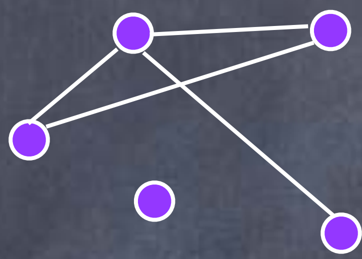
Connectivity

- Given a graph G, whether there is a path between two nodes u and v is an important question regarding G
  - u is said to be **connected to** v if there is such a path
  - u connected to v iff there is a walk from u to v
- Relation Connected(u,v) is an equivalence relation
  - Reflexive, Symmetric and Transitive

    Walks can be spliced together to get walks
  - Equivalence classes of this relation are called the connected components of G

Degree of a node

- Given a simple graph G = (V,E), for each node $v \in V$, the degree of v is the number of edges incident on v



- Formally, $deg(v) = | \{ u : \{u,v\} \in E \} |$

Note: Definition restricted to simple graphs

- Counting edges in two different ways: $2 \cdot |E| = \Sigma_{v \in V} deg(v)$
- Degree sequence: sorted list of degrees. (e.g.: 0,1,2,2,3)
- Degree sequence invariant under isomorphism

Eulerian Trail & Circuit

- Eulerian trail: a walk visiting every edge exactly once
  - Eulerian trail exists → at most 2 odd degree nodes
    - Enter(v) = { $\{v_{i-1},v_i\}$ | v_i = v }, Exit(v) = { $\{v_i,v_{i+1}\}$ | v_i = v } partition all the edges incident on v. |Enter(v)|=|Exit(v)| for all v except the start and end nodes of the walk.
- Eulerian circuit: a closed walk visiting every edge exactly once
  - Eulerian circuit exists → no odd degree nodes
- If no odd degree nodes and all edges in one connected component, then must have an Eulerian circuit!
  - Proof sketch: Must be cyclic [Why?] Remove a cycle: still no odd degree node. Inductively obtain Eulerian circuits in each connected component in the remaining graph. Can stitch them all onto the removed cycle into one circuit.

Hamiltonian Cycle

- Eulerian circuit: a closed walk visiting every edge exactly once
  - Eulerian circuit exists $\longleftrightarrow$ all edges in the same connected component and no odd degree nodes
  - Can efficiently find one if they exist
- Hamiltonian Cycle: a cycle that contains all the nodes in the graph
  - No efficient algorithm known to check if a graph has a Hamiltonian cycle!
    - An "NP-hard" problem. Widely believed that no efficient algorithm exists!
      - (cf. Graph Isomorphism: It is believed to be hard, but also believed to be not NP-hard)

Distance

Prove via contradiction

Hence, $\exists$ walk $\rightarrow$ $\exists$ path

In many applications, the edges on the graph will have "lengths". In simple graphs, all edges are of length 1.

- Shortest walk between nodes u and v is always a path
- Shortest path is of great interest in many applications
  - e.g., nodes correspond to locations on a map and edges are roads, optic fibers etc.
  - Also, graph can be used to model probabilistic processes, with shortest path indicating the most likely outcome
- Length of the shortest path between u and v is called the distance between u and v (∞ if no path)

$$\min_{W:\ u\text{-}v \text{ walk}} \text{Length}(W)$$

- Diameter is the largest distance in a graph (can be ∞)

$$\max_{u,v} \text{Distance}(u,v) = \max_{u,v} \min_{W:\ u\text{-}v \text{ walk}} \text{Length}(W)$$

Graphs in action

Shortest Paths in Action

- Obvious example: nodes correspond to locations on a map and edges are roads, optic fibers etc.
  - Weighted edges: each edge has its own "length" (instead of 1)
- But also over more abstract graphs
  - e.g., Graph-based models in AI/machine-learning for modeling probabilistic systems
    - e.g., a graph, modeling speech production: nodes correspond to various "states" the vocal chords/lips etc. could be in while producing a given a sound sequence. Edges show transitions (next state) over time. Shortest path in this graph gives the "most likely" word that was spoken.