Data Structures in Java

Session 22

Instructor: Bert Huang

http://www.cs.columbia.edu/~bert/courses/3134

Announcements

- Homework 5 solutions posted
- Homework 6 to be posted this weekend
- Final exam Thursday, Dec. 17th, 4-7 PM, Hamilton 602 (this room)
  - same format as midterm (open book/notes)
- Distinguished Lecture: Barbara Liskov, MIT.
  Turing Award Winner 2009.
  11 AM Monday. Davis Auditorium, CEPSR/Schapiro

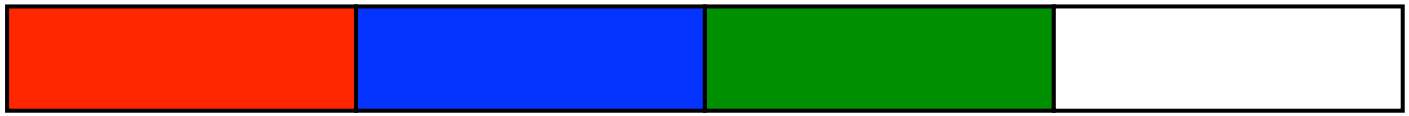
Review

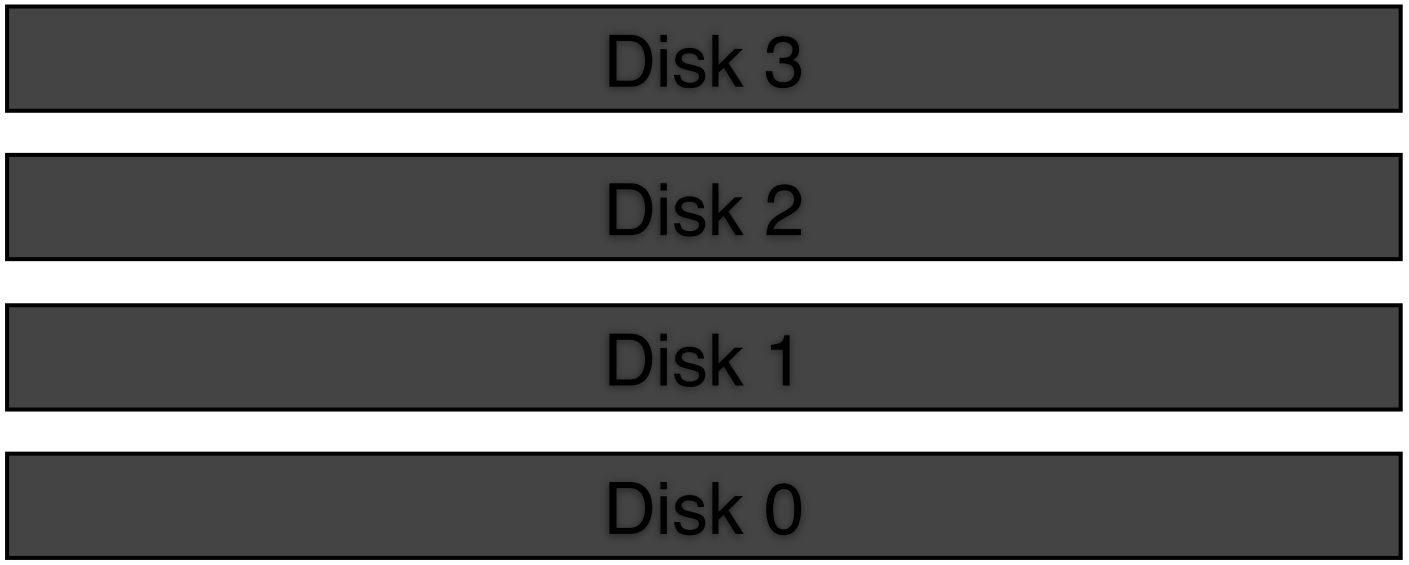
- Sorting Algorithm Examples
- QuickSort space clarification
- External sorting*

Today's Plan

- Correction on external sorting
- Complexity Classes
  - P, NP, NP-Complete, NP-Hard

External Sorting



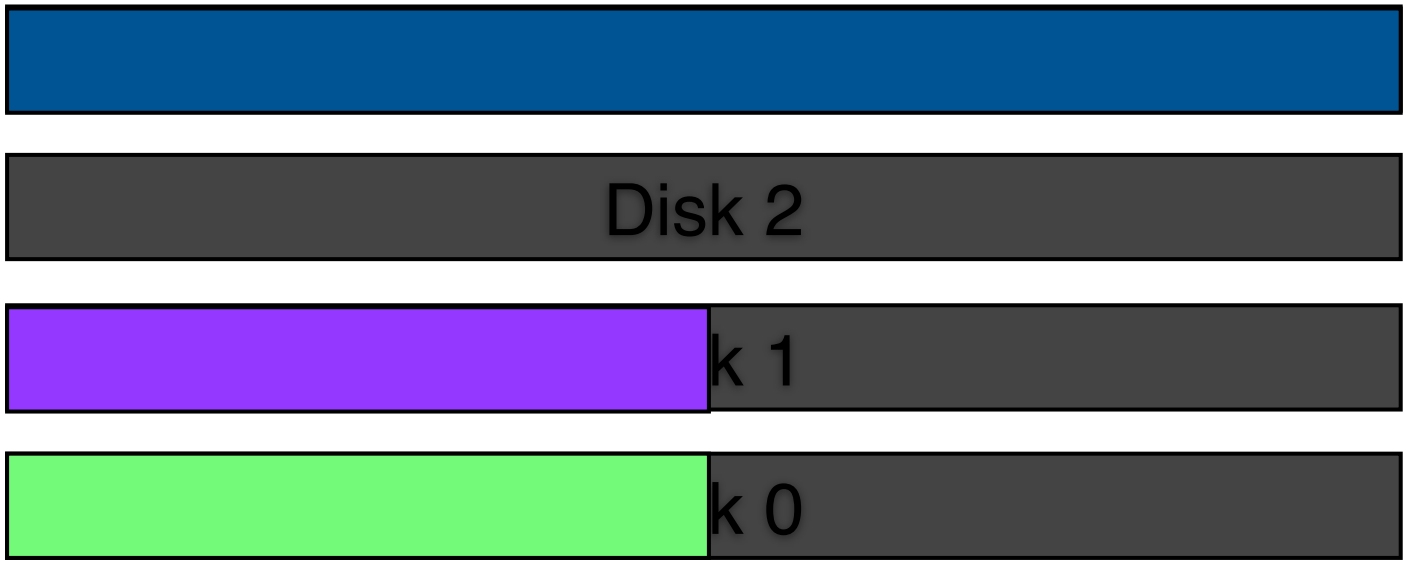


External Sorting

External Sorting

External Sorting

External Sorting



Complexity of Problems

- We've been concerned with the complexity of *algorithms*
- It is important to also consider the complexity of *problems*
- Understanding complexity is important for theory, and also for practice
  - understanding the hardness of problems helps us build better algorithms

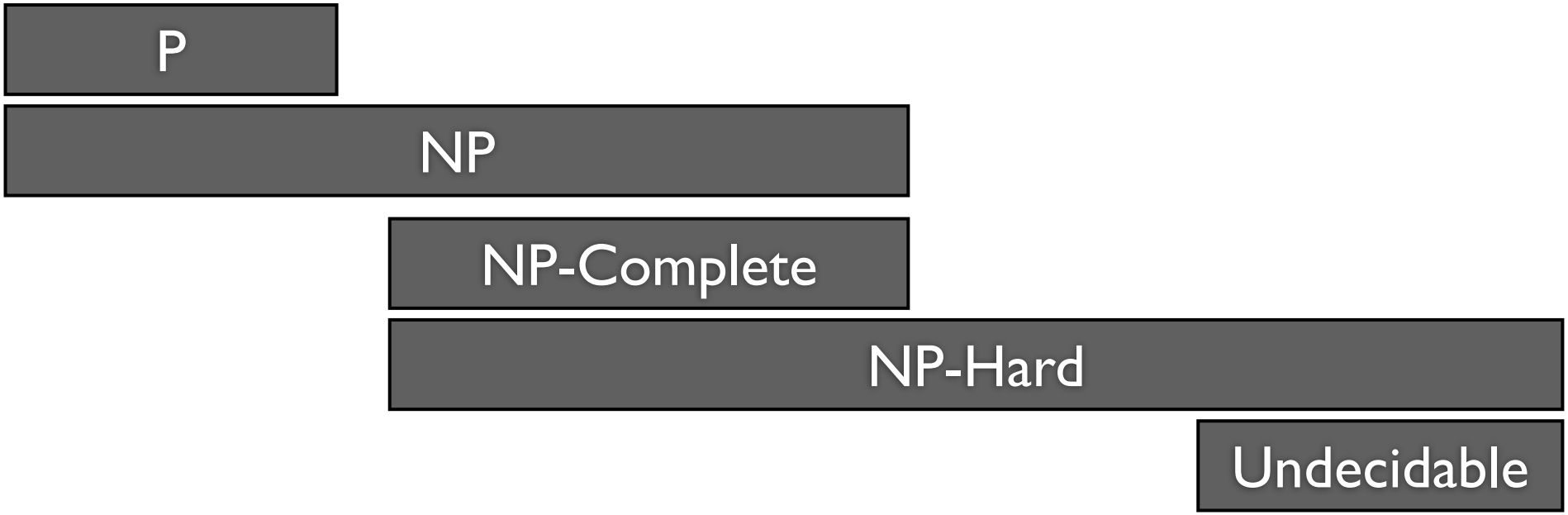
Three Graph Problems

- Euler Path: does a path exist that uses every **edge** exactly once?
- Hamiltonian Path: does a path exist that visits every **node** exactly once?
- Traveling Salesman: find the shortest path that visits every node exactly once?

Complexity Classes

- **P -** solvable in polynomial time
- **NP -** solvable in polynomial time by a nondeterministic computer
  - i.e., you can check a solution in polynomial time
- **NP-complete** - a problem in NP such that any problem in NP is polynomially reducible to it
- **Undecidable -** no algorithm can solve the problem

Probable Complexity Class Hierarchy



Polynomial Time **P**

- All the algorithms we cover in class are solvable in polynomial time
- An algorithm that runs in polynomial time is considered **efficient**
- A problem solvable in polynomial time is considered **tractable**

Nondeterministic Polynomial Time **NP**

- Consider a magical nondeterministic computer
  - infinitely parallel computer
- Equivalently, to solve any problem, check every possible solution in parallel
  - return one that passes the check

NP-Complete

- Special class of NP problems that can be used to solve any other NP problem
- Hamiltonian Path, Satisfiability, Graph Coloring etc.
- NP-Complete problems can be **reduced** to other NP-Complete problems:
  - polynomial time algorithm to convert the input and output of algorithms

NP-Hard

- A problem is NP-Hard if it is at least as complex as all NP-Complete problems
- NP-hard problems may not even be NP

Undecidable

- No algorithm can solve undecidable problems
- **halting problem -** given a computer program, determine if the computer program will finish
- Sketch of undecidability proof: Assume we have an algorithm that detects infinite loops
  - Write program LOOP(P) that runs P on itself
  - If P(P) halts, infinite loop, otherwise output

Halting Problem

```
LOOP(P):
  If P(P) will halt:    infinite loop
  If P(P) runs forever: output
```

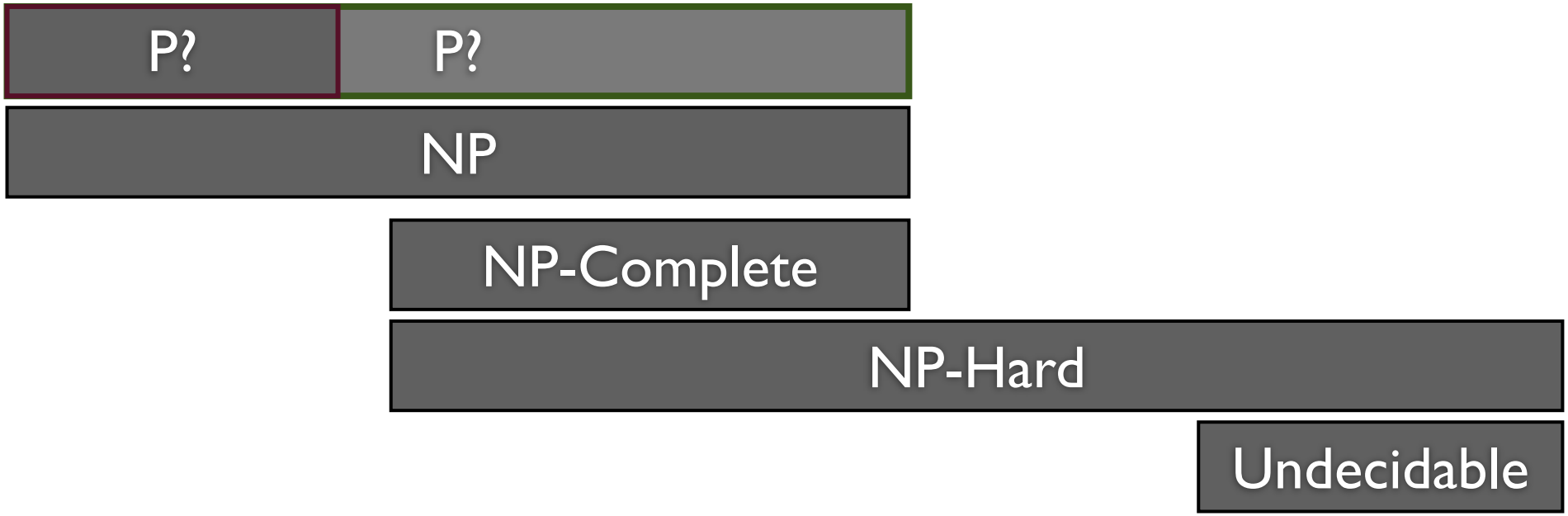
- What happens if we call LOOP(LOOP)?

```
LOOP(LOOP):
  If LOOP(LOOP) will halt:    infinite loop
  If LOOP(LOOP) runs forever: output
```

P vs. NP

- So far, nobody has proven a super-polynomial lower bound on any NP-Complete problems,
- nor has anyone found a polynomial time algorithm for an NP-complete problem
- Therefore no problem is proven to be in one class but not the other;
  - it is unknown if P and NP are the same

Complexity Class Hierarchy



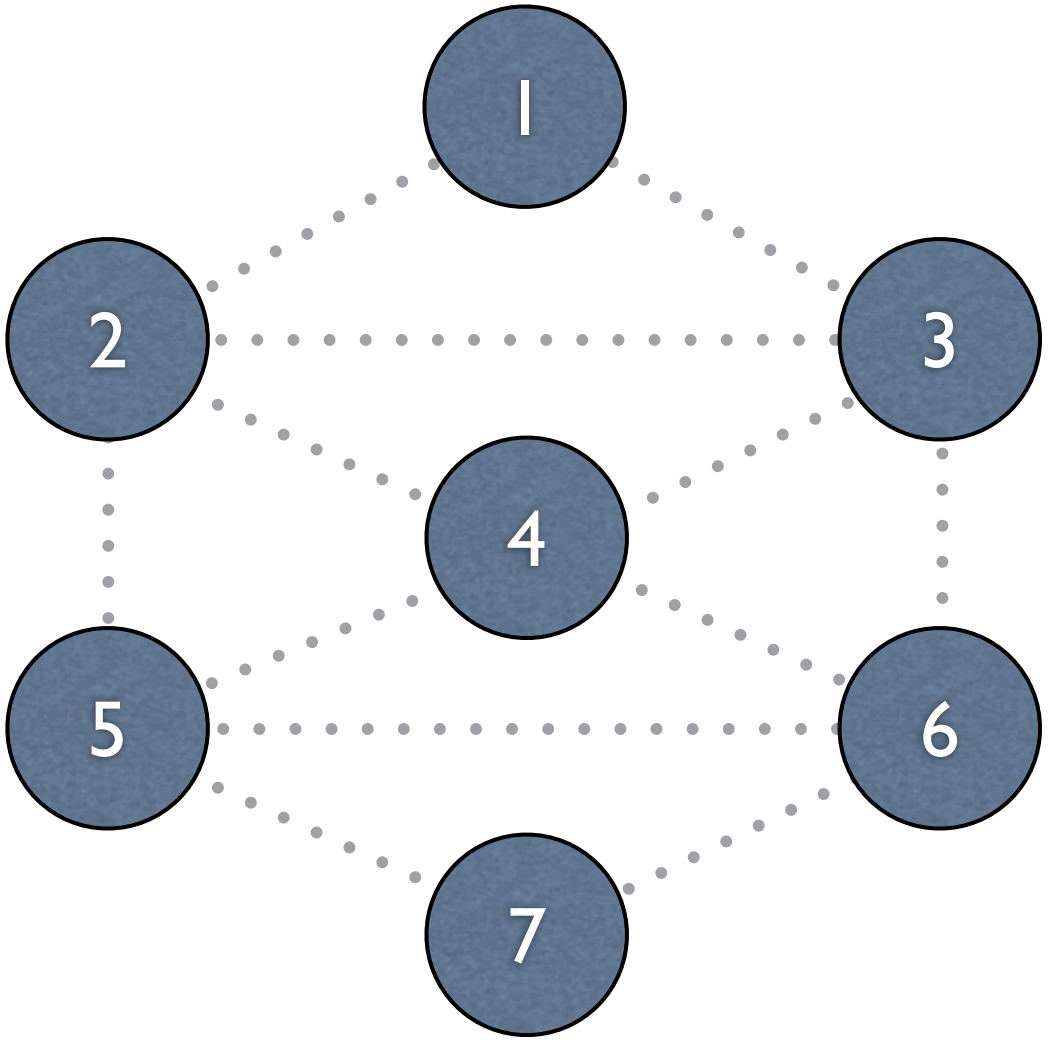
P Problems

- Most problems we've solved
- Proving a polynomial (or logarithmic) bound on any algorithm proves a problem is in **P**
- Euler paths - does a path exist that uses every edge?
  - Return if there are zero or two nodes with odd degree.

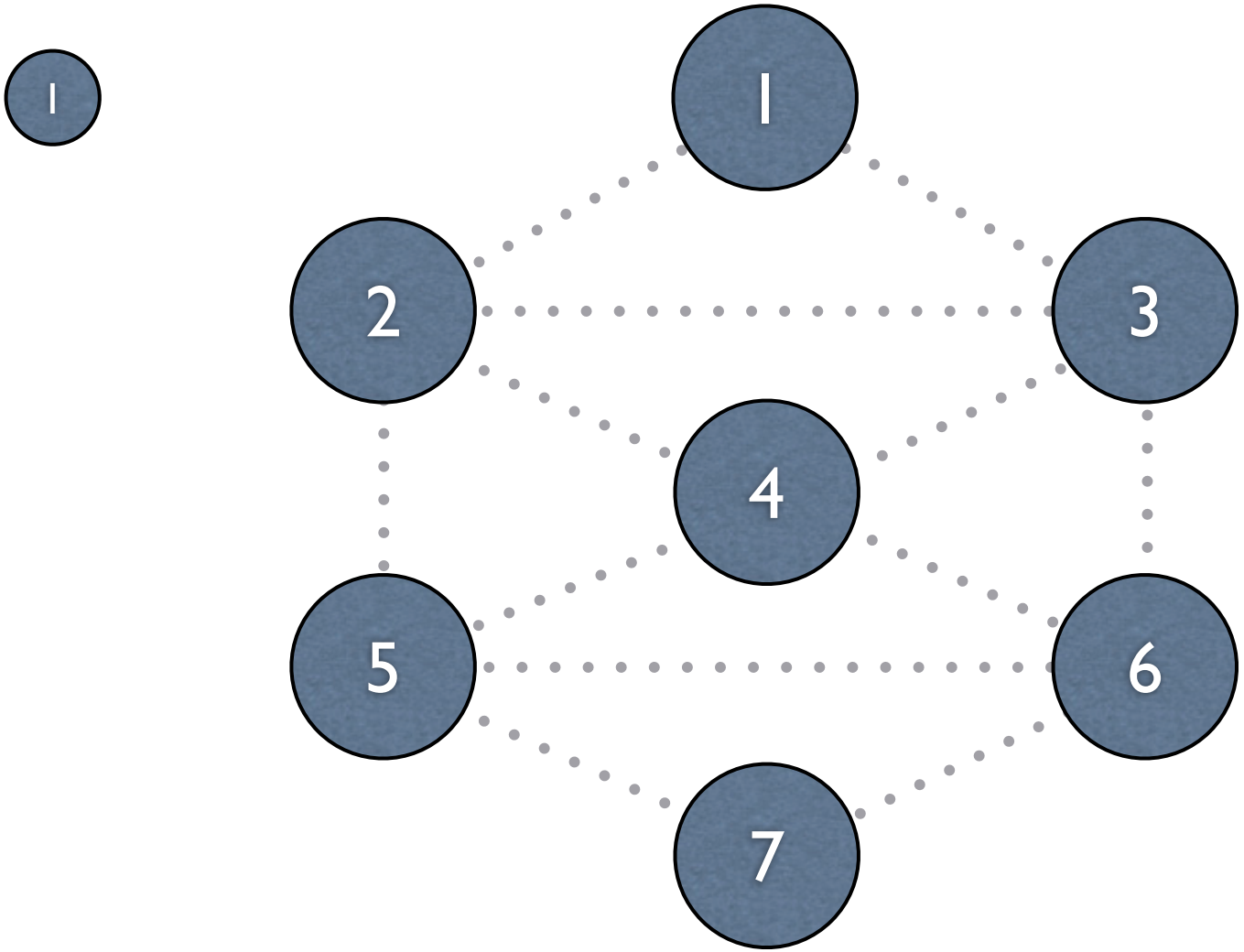
Finding an Euler Circuit

- Run a partial DFS; search down a path until you need to backtrack (mark edges instead of nodes)
- At this point, you will have found a circuit
- Find first node along the circuit that has unvisited edges; run a DFS starting with that edge
- Splice the new circuit into the main circuit, repeat until all edges are visited

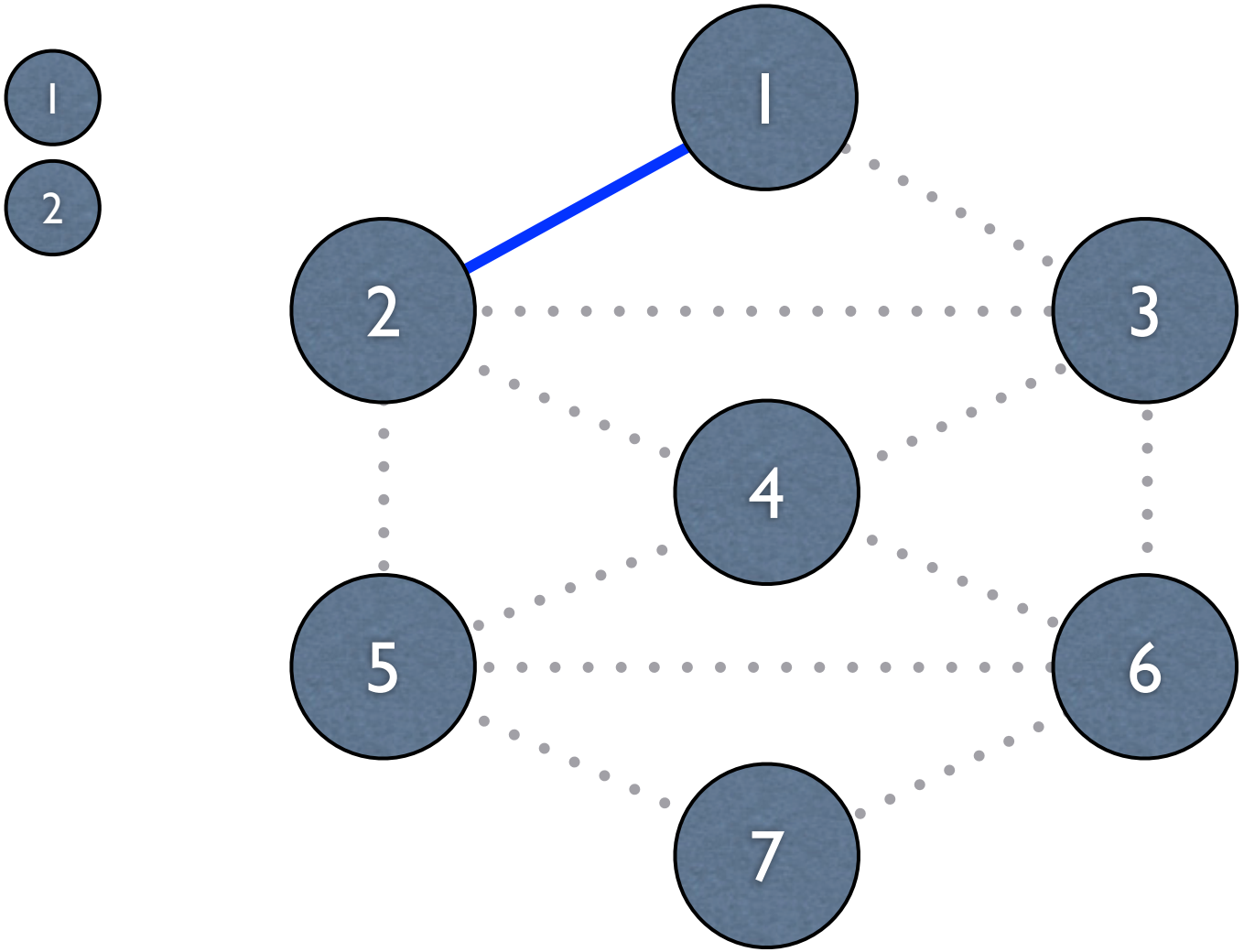
Euler Circuit Example



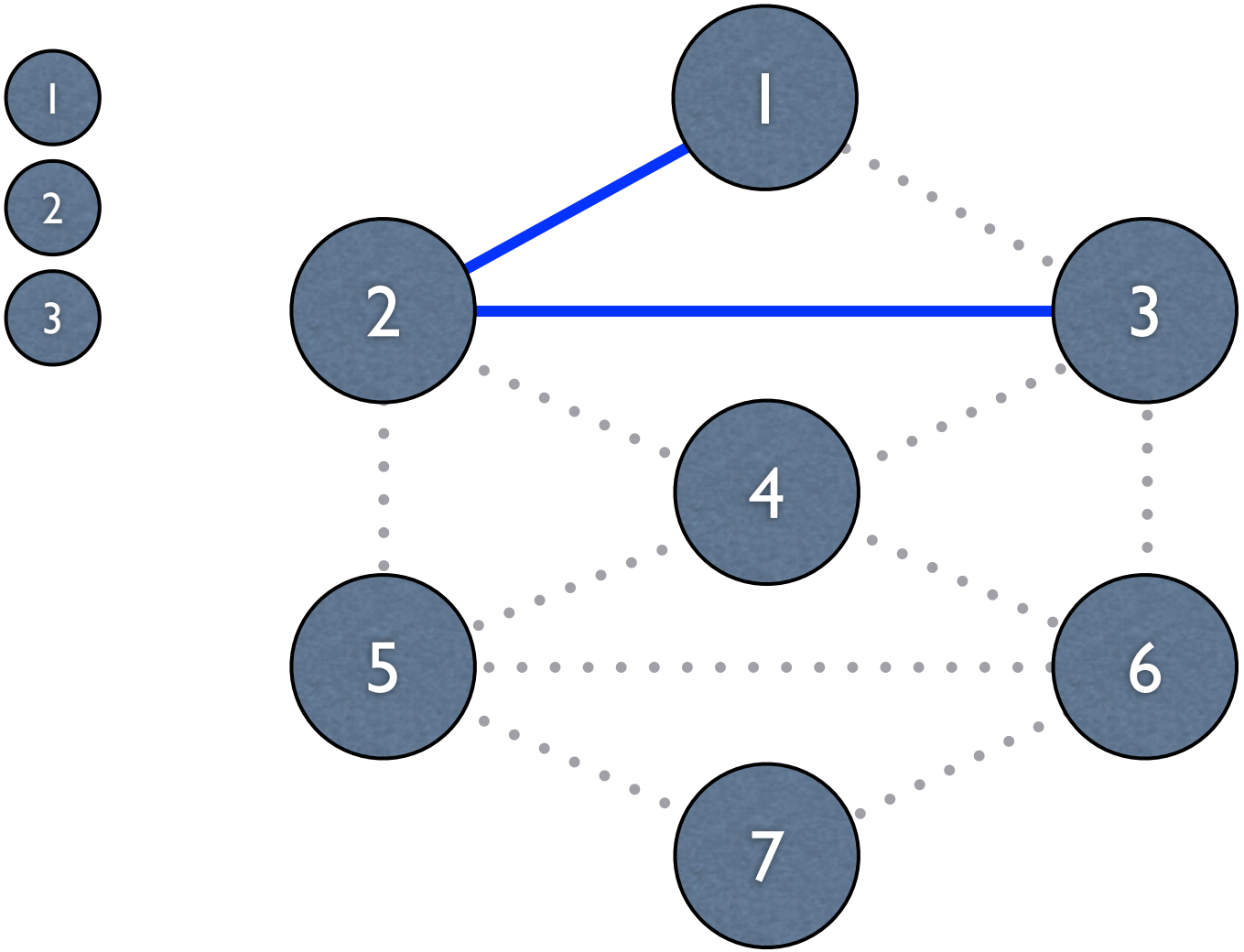
Euler Circuit Example



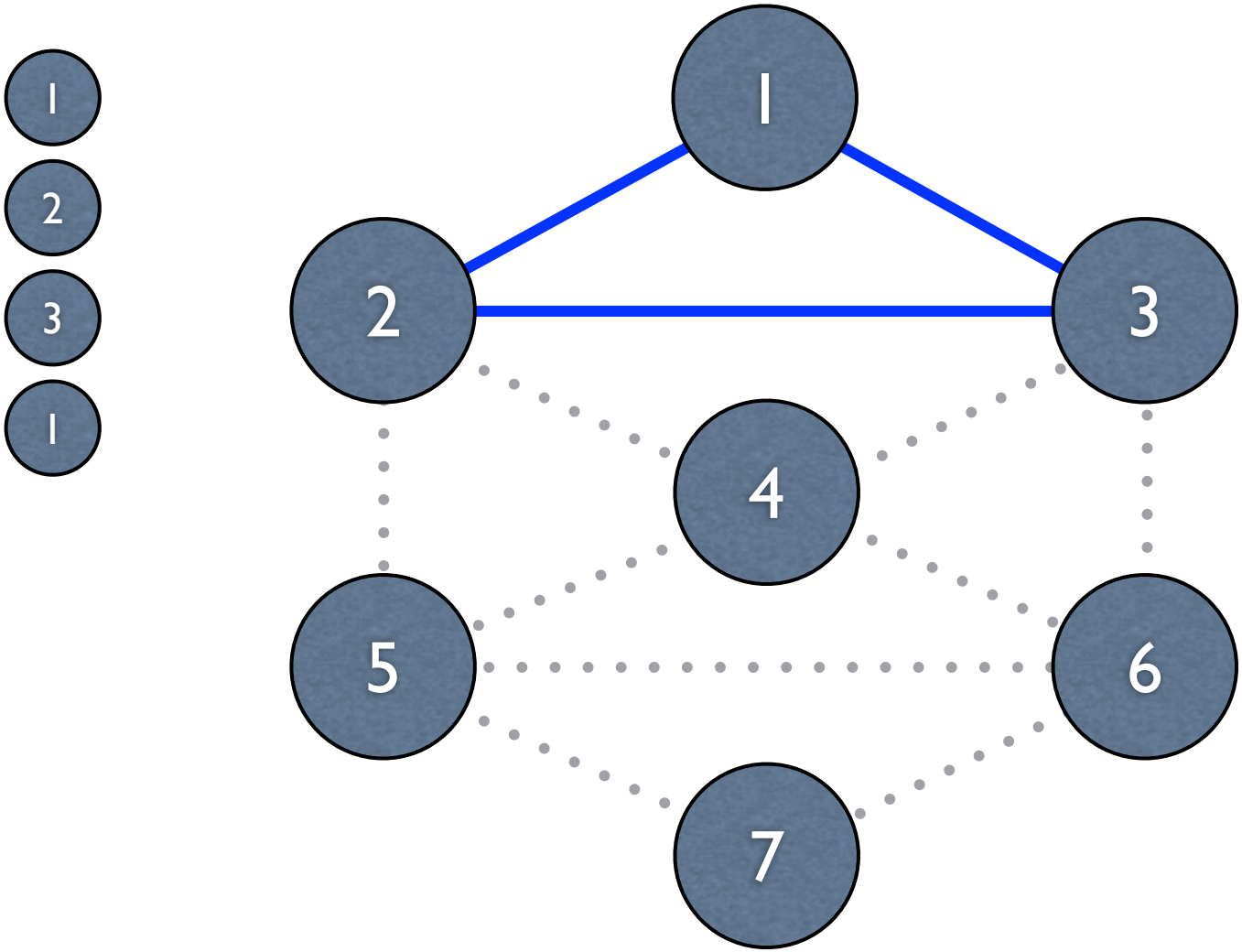
Euler Circuit Example



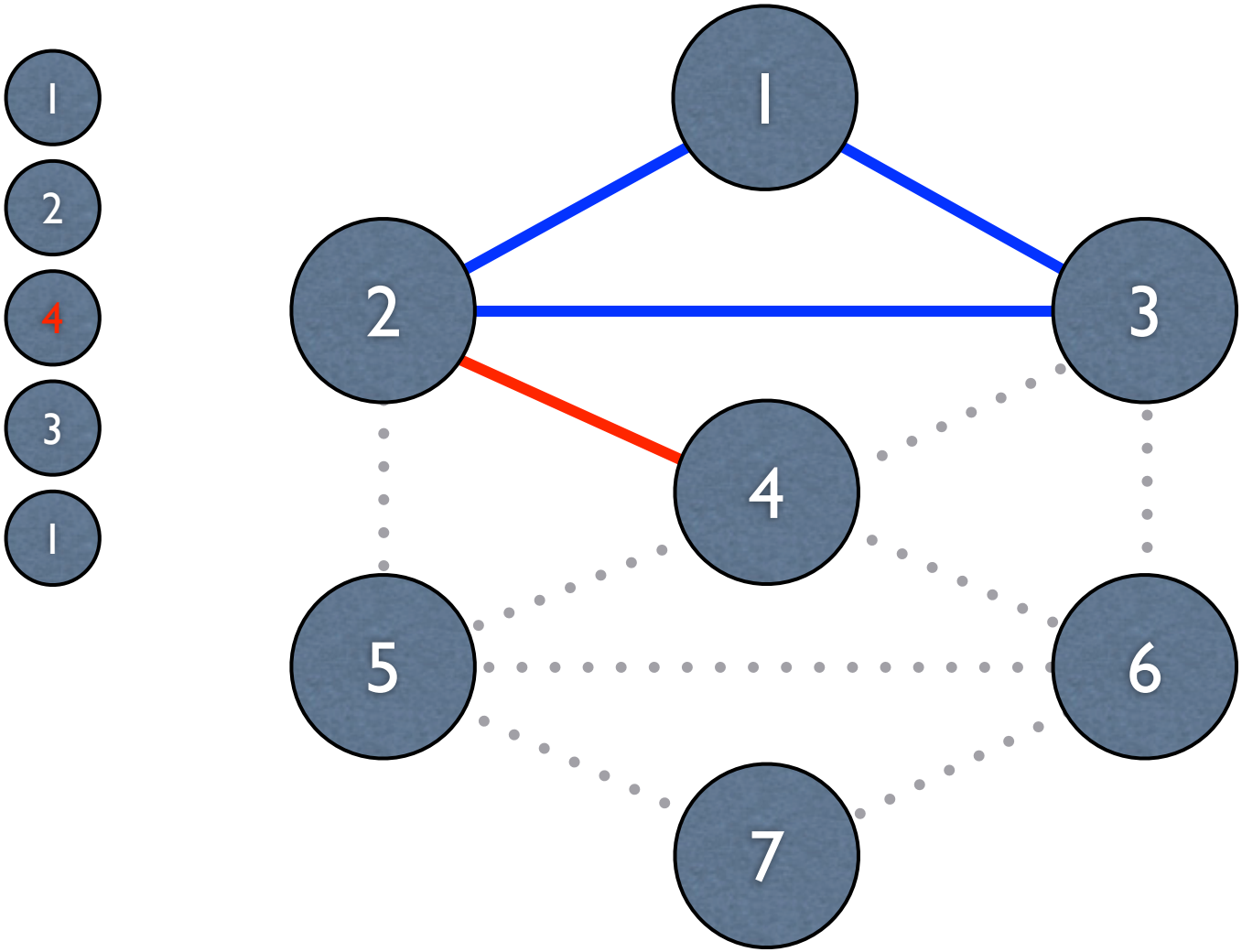
Euler Circuit Example



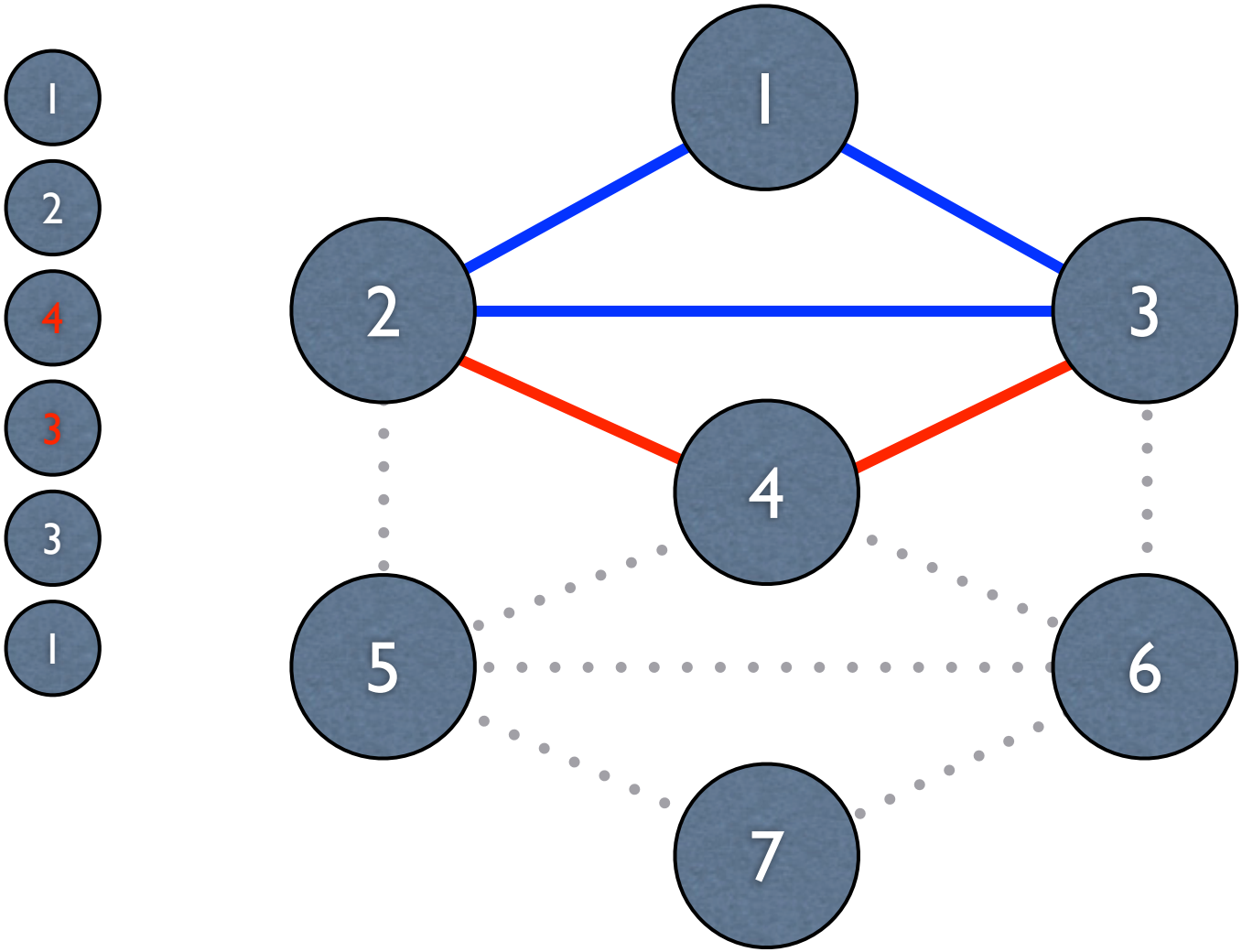
Euler Circuit Example



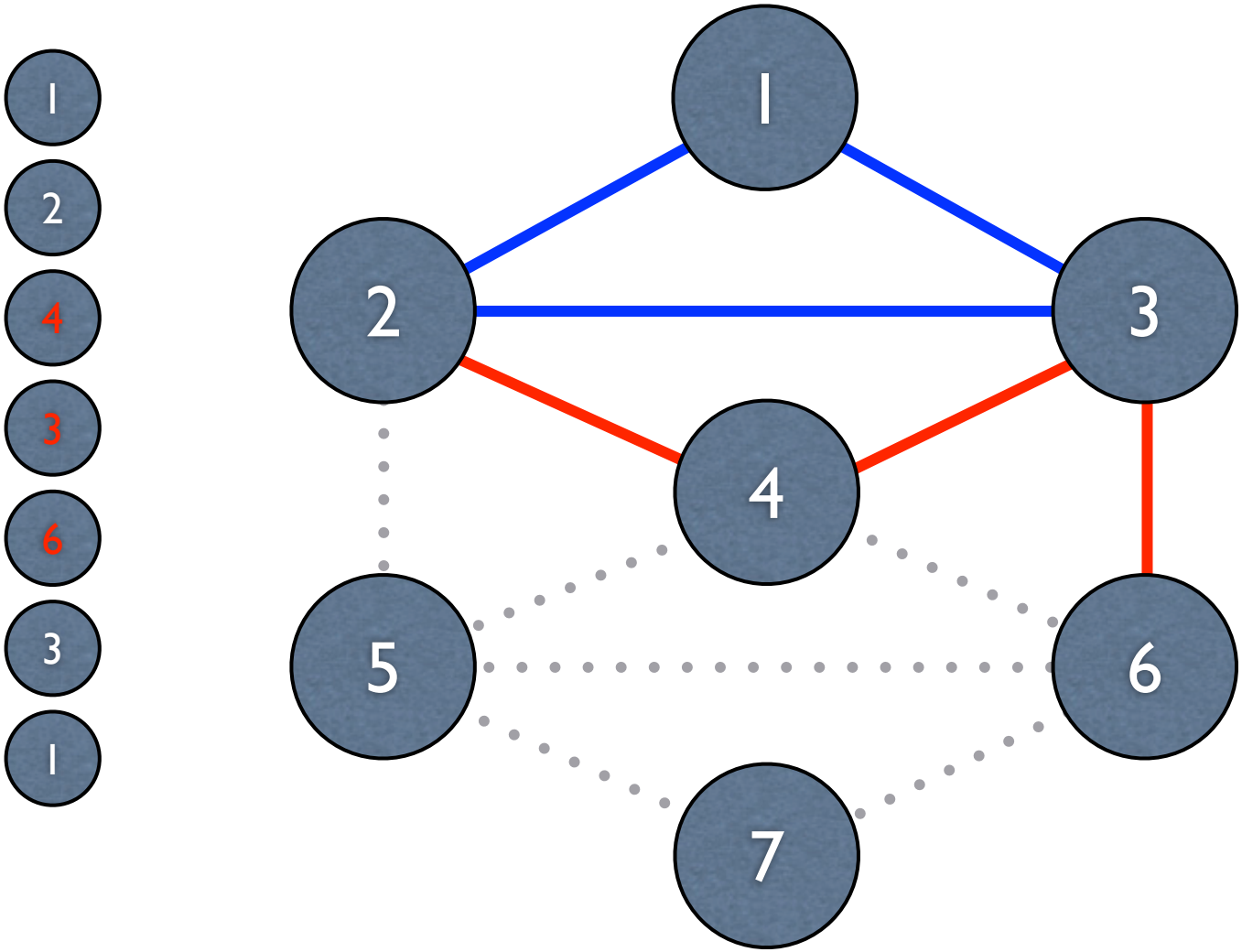
Euler Circuit Example



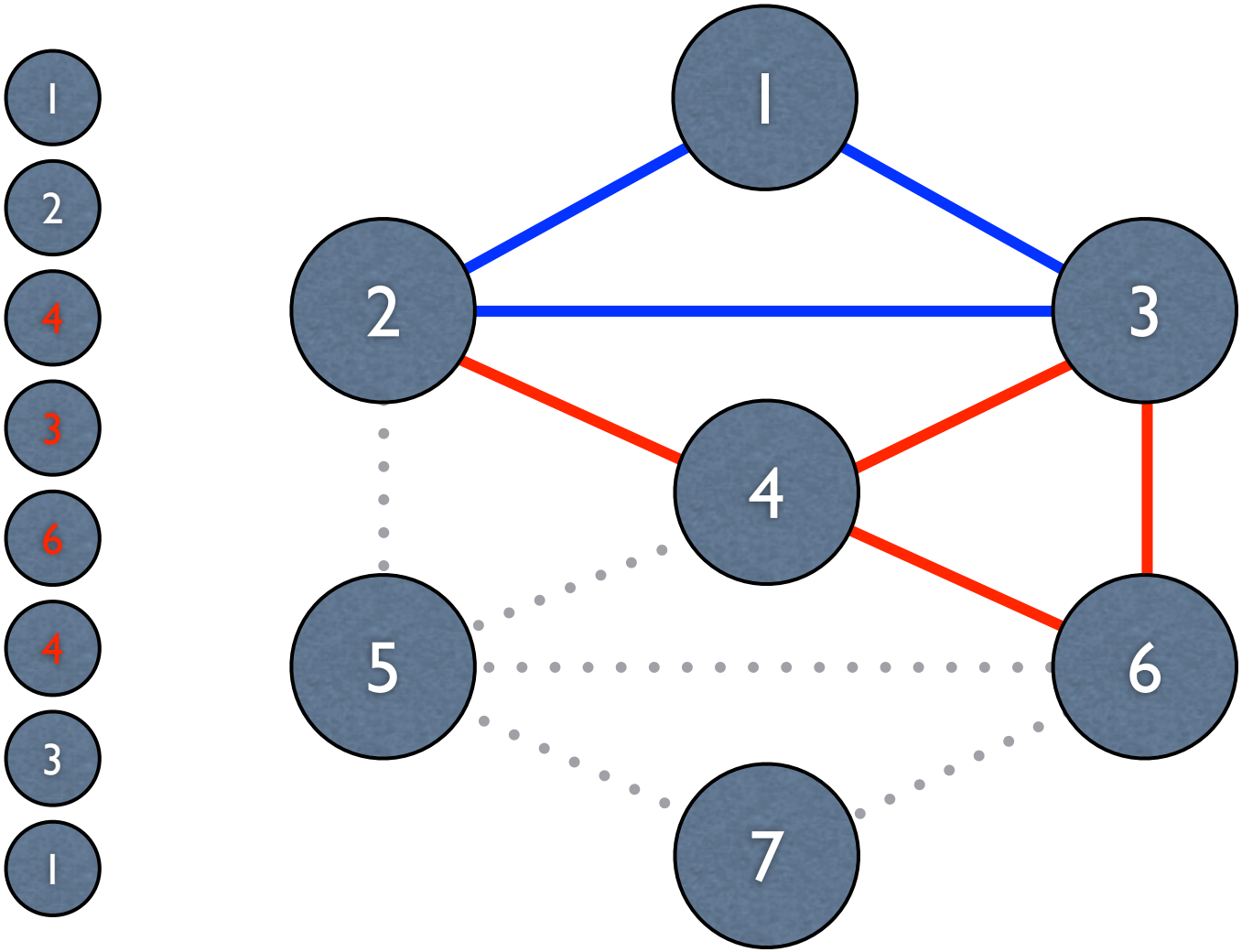
Euler Circuit Example



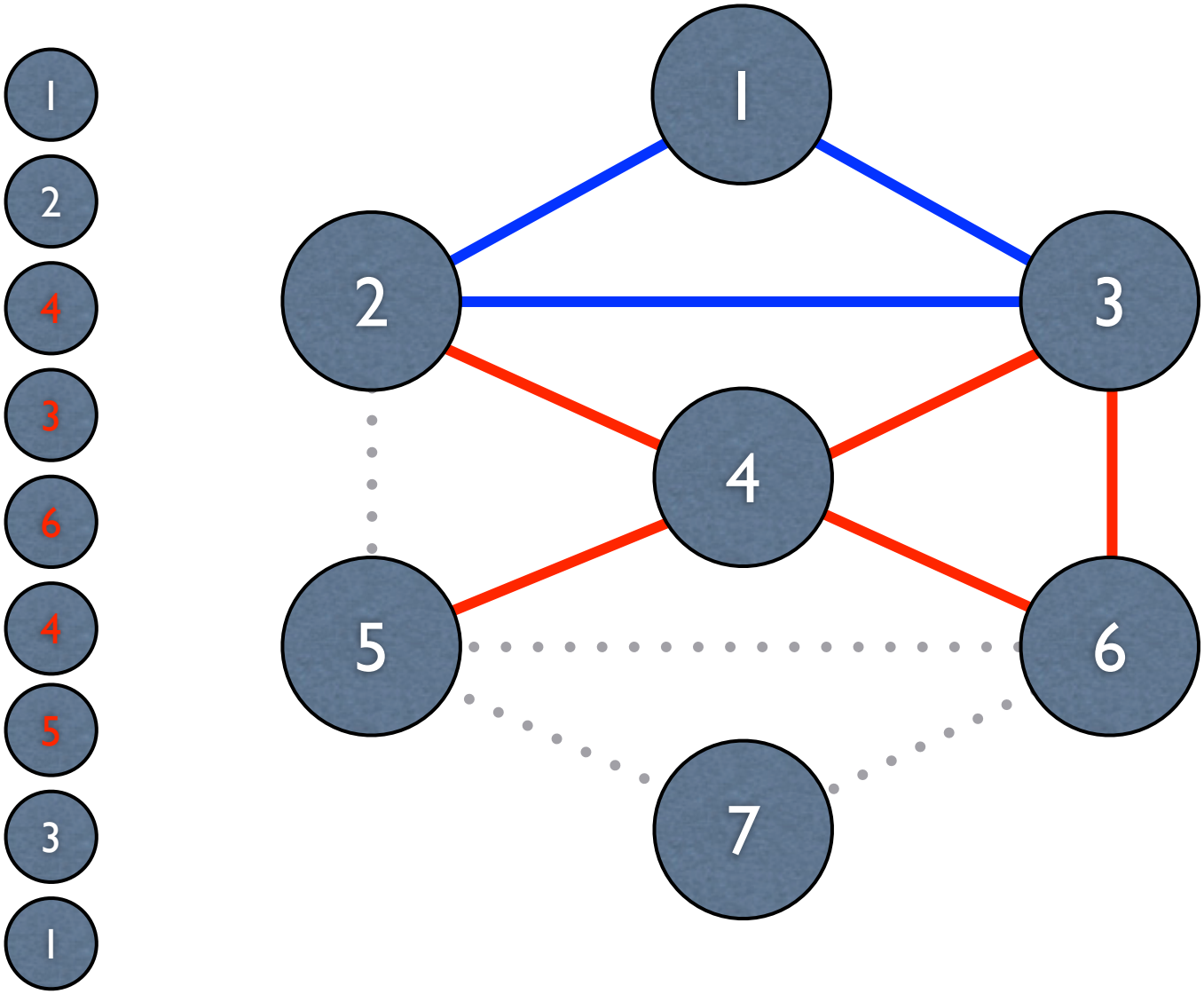
Euler Circuit Example



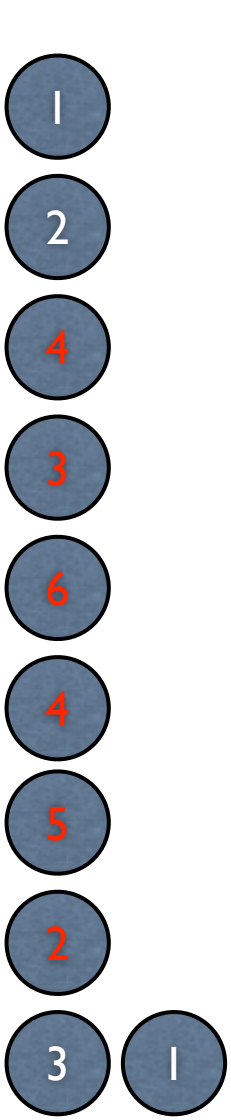
Euler Circuit Example

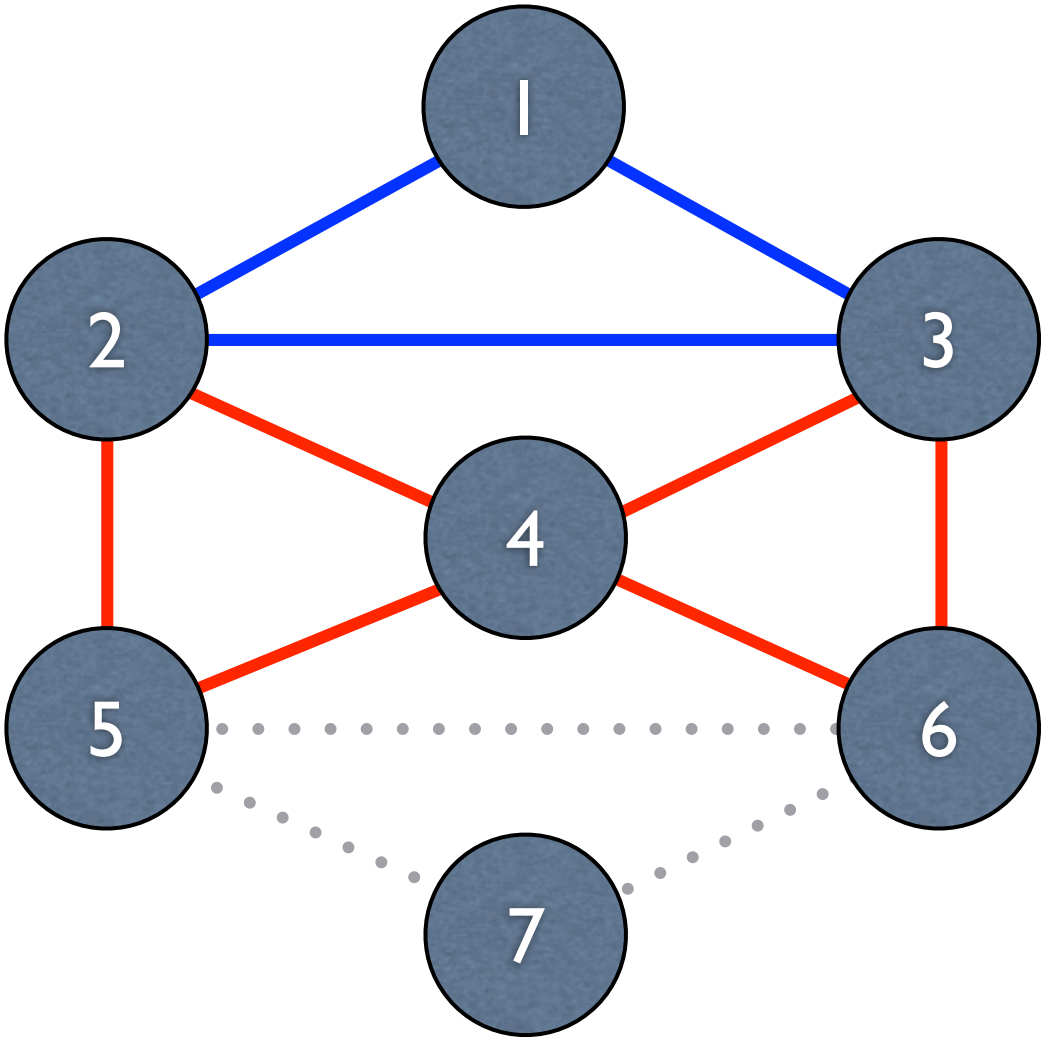


Euler Circuit Example

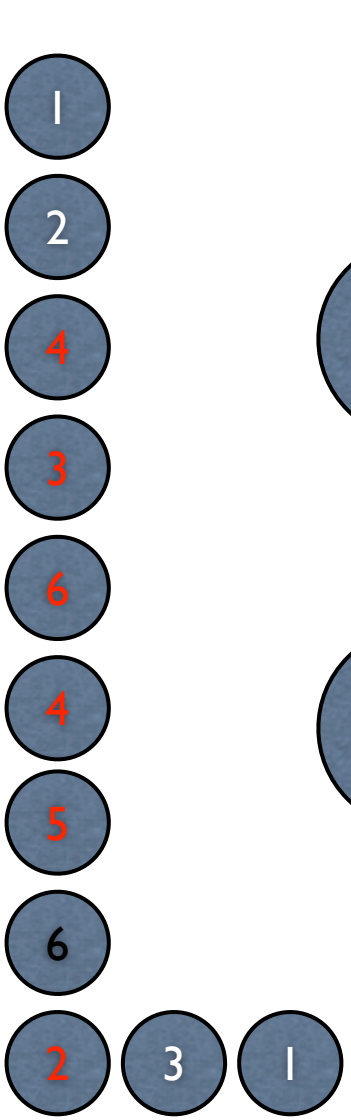


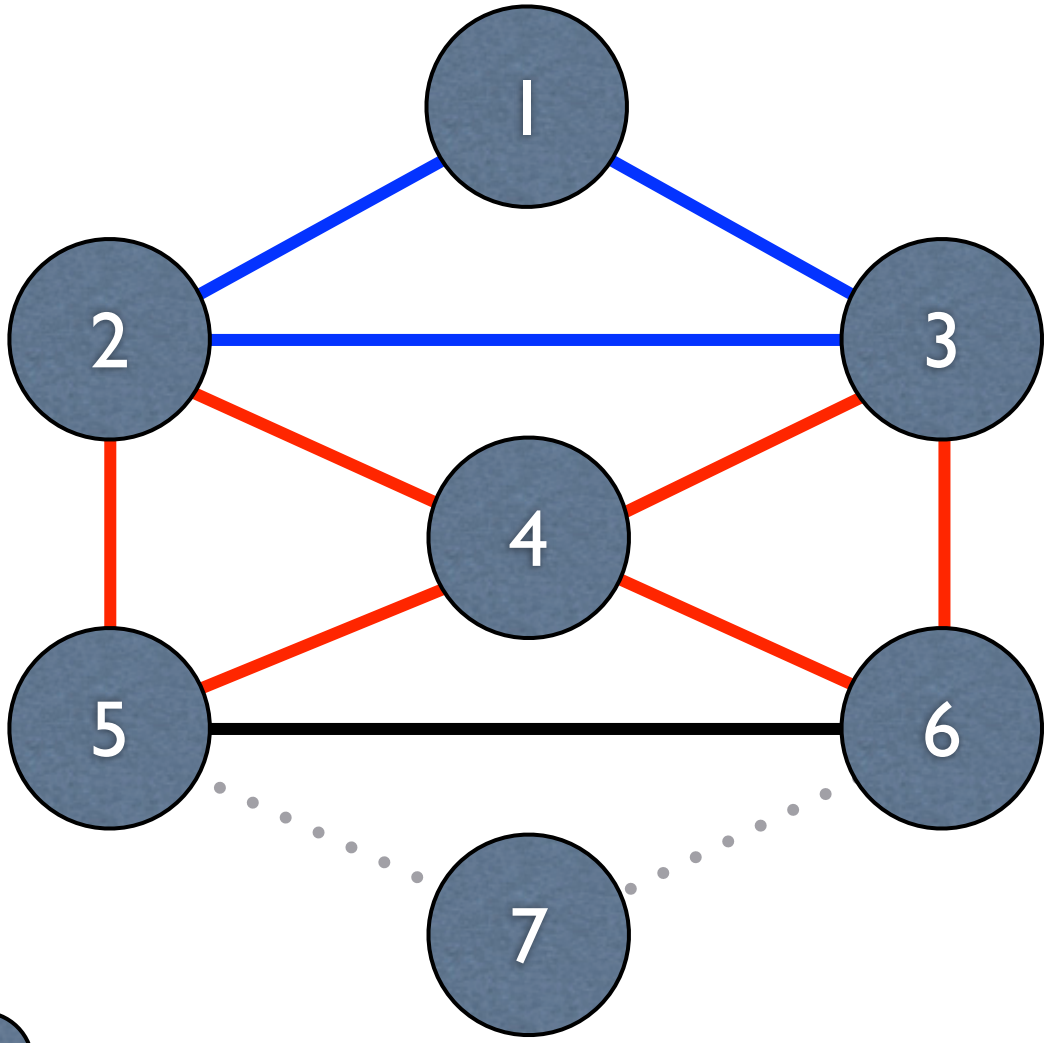
Euler Circuit Example



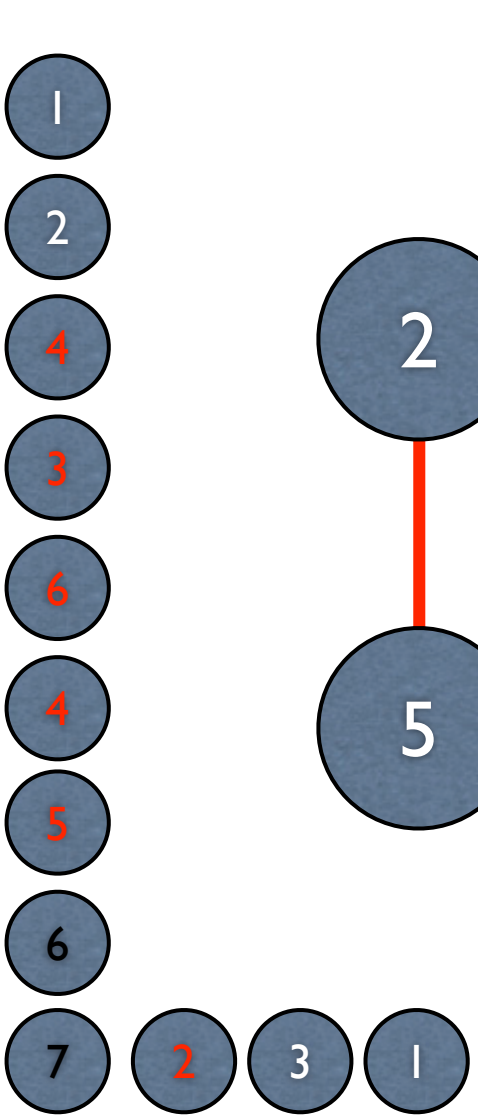


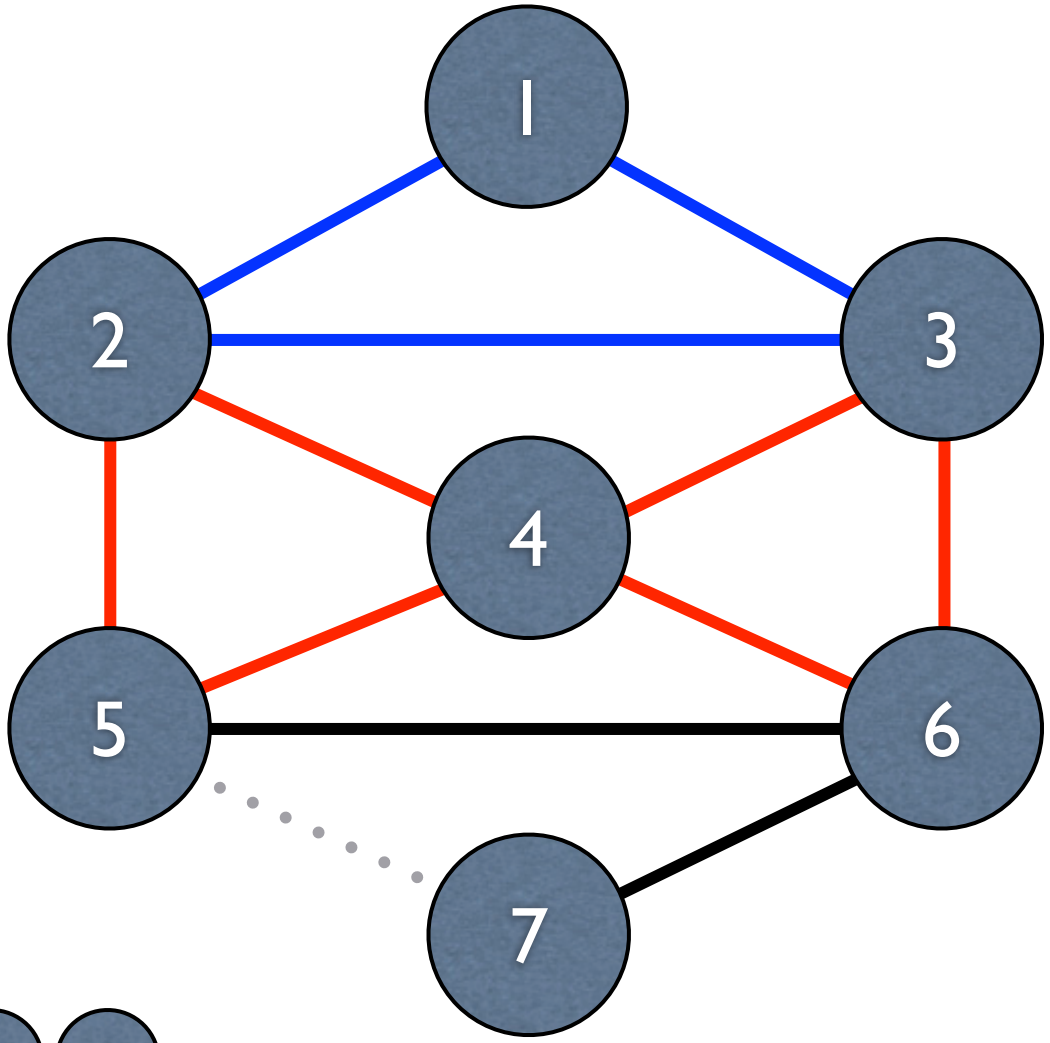
Euler Circuit Example



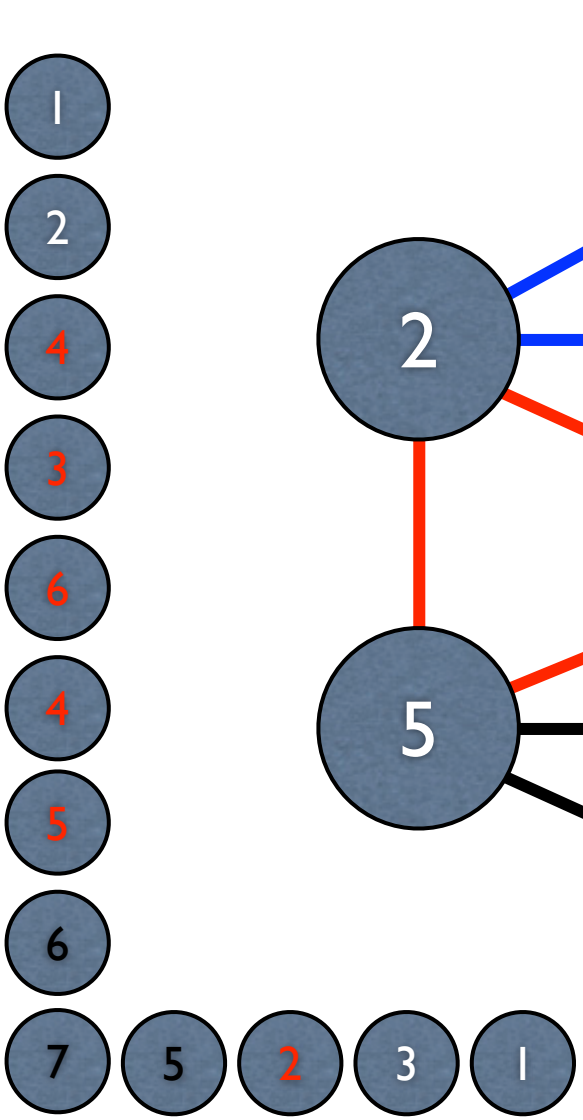


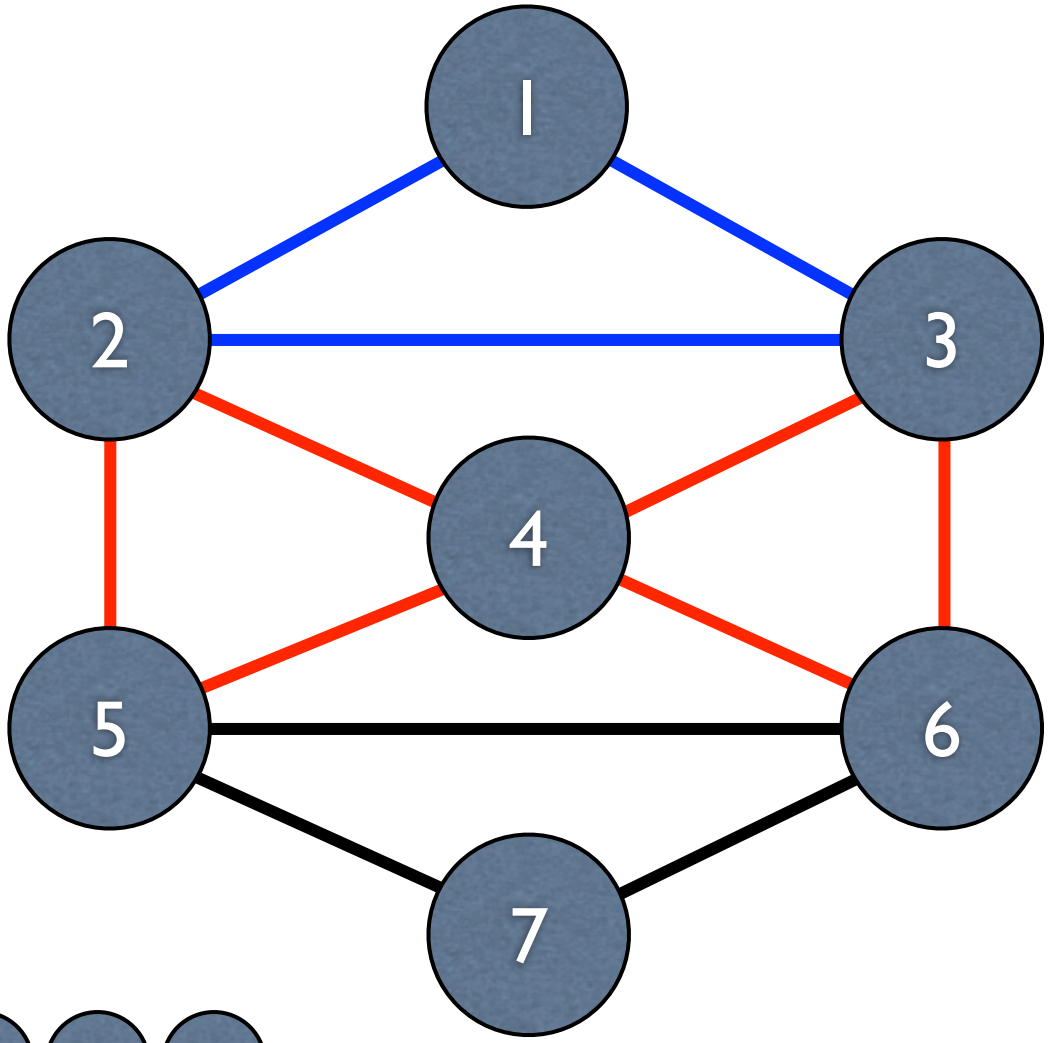
Euler Circuit Example





Euler Circuit Example





NP-Complete Problems
Satisfiability

- Given Boolean expression of N variables, can we set variables to make expression true?
- First NP-Complete proof because Cook's Theorem gave polynomial time procedure to convert any NP problem to a Boolean expression
- I.e., if we have efficient algorithm for Satisfiability, we can efficiently solve any NP problem

NP-Complete Problems Graph Coloring

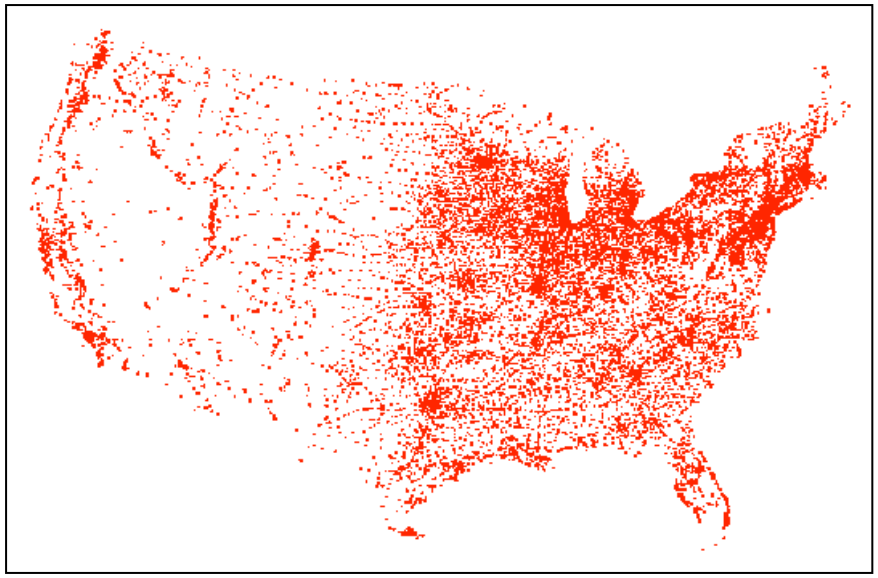
- Given a graph is it possible to color with **k** colors all nodes so no adjacent nodes are the same color?
- Coloring countries on a map
- Sudoku is a form of this problem. All squares in a row, column and blocks are connected. **k** = 9

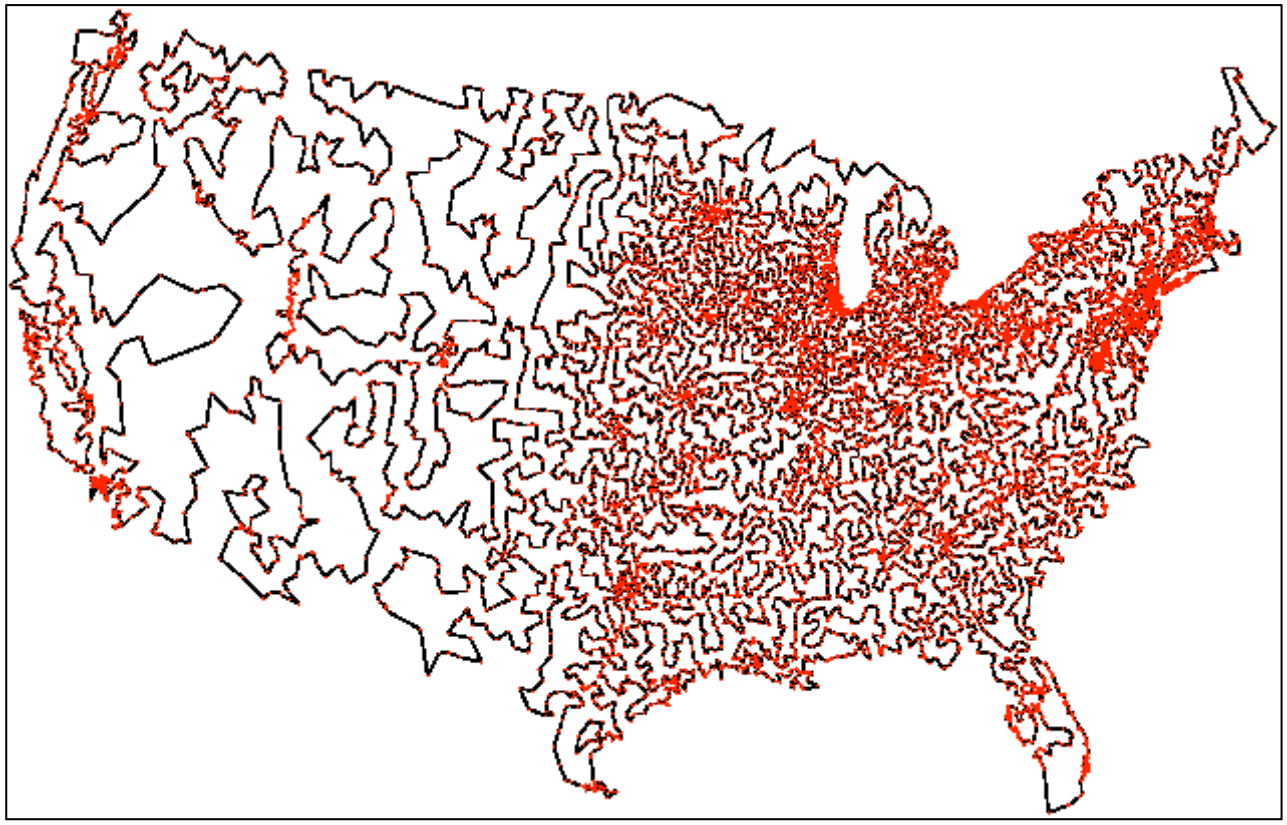
NP-Complete Problems
Hamiltonian Path

- Given a graph with N nodes, is there a path that visits each node exactly once?

NP-Hard Problems Traveling Salesman

- Closely related to Hamiltonian Path problem
- Given complete graph **G**, find the shortest path that visits all nodes
- If we are able to solve TSP, we can find a Hamiltonian Path; set connected edge weight to constant, disconnected to infinity
  - TSP is NP-hard





http://www.tsp.gatech.edu/gallery/itours/usa13509.html

Reading

- Weiss 9.7
- http://www.tsp.gatech.edu/