15.082 and 6.855J Fall 2010

Network Optimization J.B. Orlin

WELCOME!

- **Welcome to 15.082/6.855J**
- **Introduction to Network Optimization**
- **Instructor: James B. Orlin**
- **TA: David Goldberg**

- **Textbook: *Network Flows: Theory, Algorithms, and Applications* by Ahuja, Magnanti, and Orlin referred to as AMO**

Quick Overview

- Next: The Koenigsberg Bridge Problem
  - Introduces Networks and Network Algorithms
- Some subject management issues
- Network flows and applications
- Computational Complexity
- Overall goal of today's lecture: set the tone for the rest of the subject
  - provide background
  - provide motivation
  - handle some class logistics

On the background of students

- **Requirement for this class**
  - **Either Linear Programming (15.081J)**
  - **or Data Structures**

- **Mathematical proofs**
  - **The homework exercises usually call for proofs.**
  - **The midterms will not require proofs.**
  - **For those who have not done many proofs before, the TA will provide guidance**

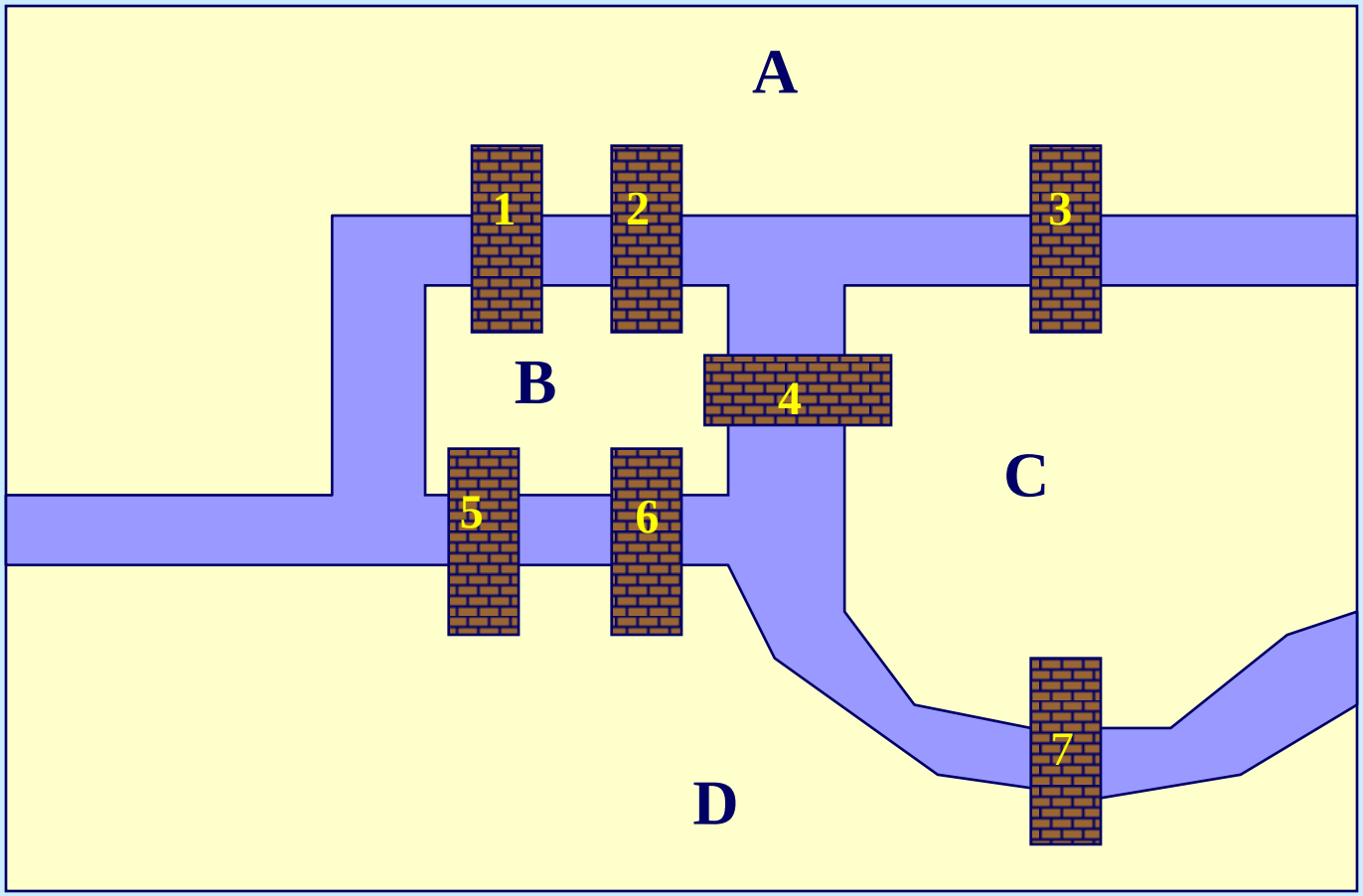
Some aspects of the class

- **Fondness for Powerpoint animations**
- **Cold-calling as a way to speed up learning of the algorithms**
- **Talking with partners (the person next to you in in the classroom.)**
- **Class time: used for presenting theory, algorithms, applications**
  - **mostly outlines of proofs illustrated by examples (not detailed proofs)**
  - **detailed proofs are in the text**

The Bridges of Koenigsberg: Euler 1736

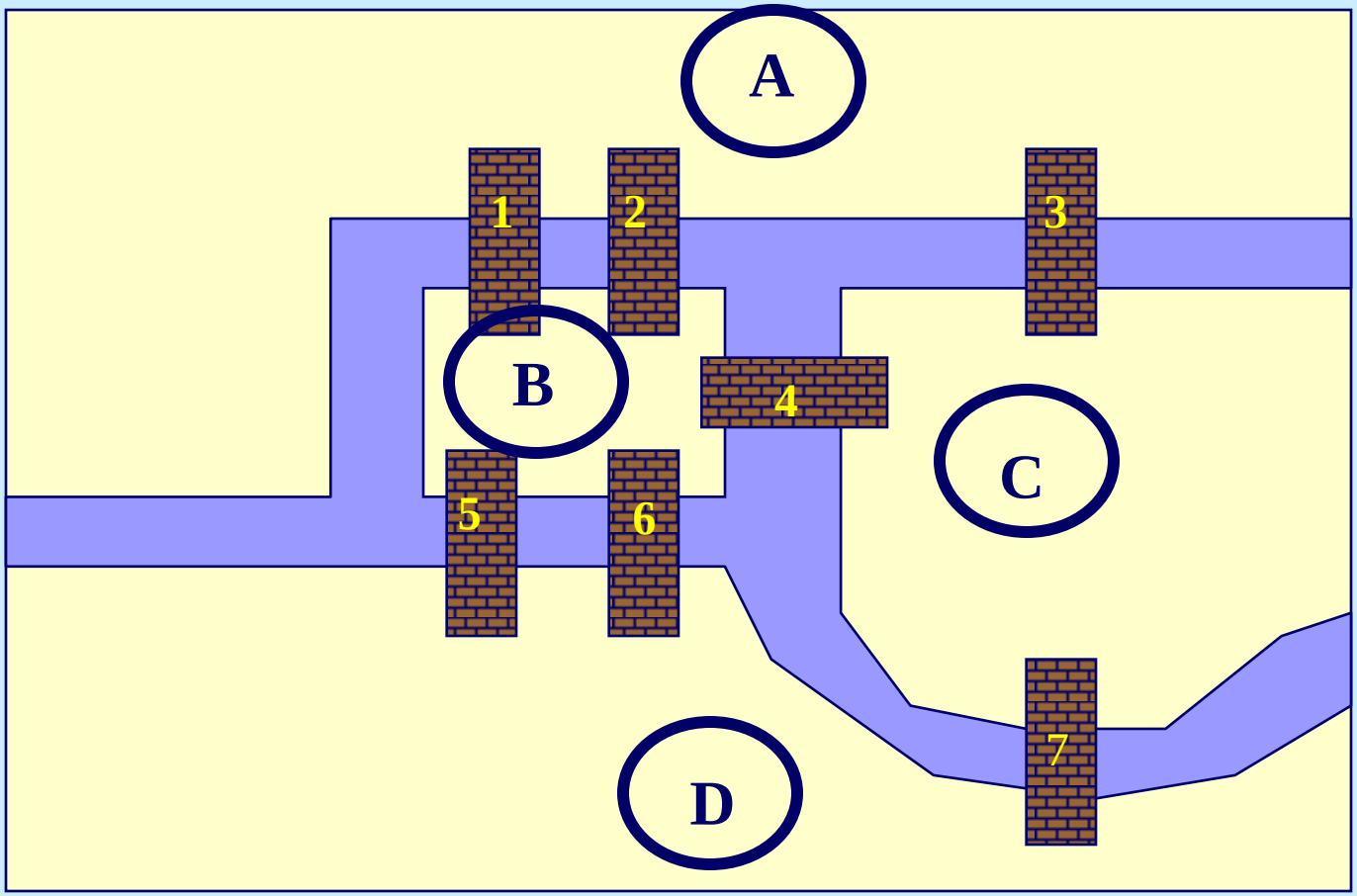
- "Graph Theory" began in 1736
- Leonard Eüler
  - Visited Koenigsberg
  - People wondered whether it is possible to take a walk, end up where you started from, and cross each bridge in Koenigsberg exactly once
  - Generally it was believed to be impossible

The Bridges of Koenigsberg: Euler 1736



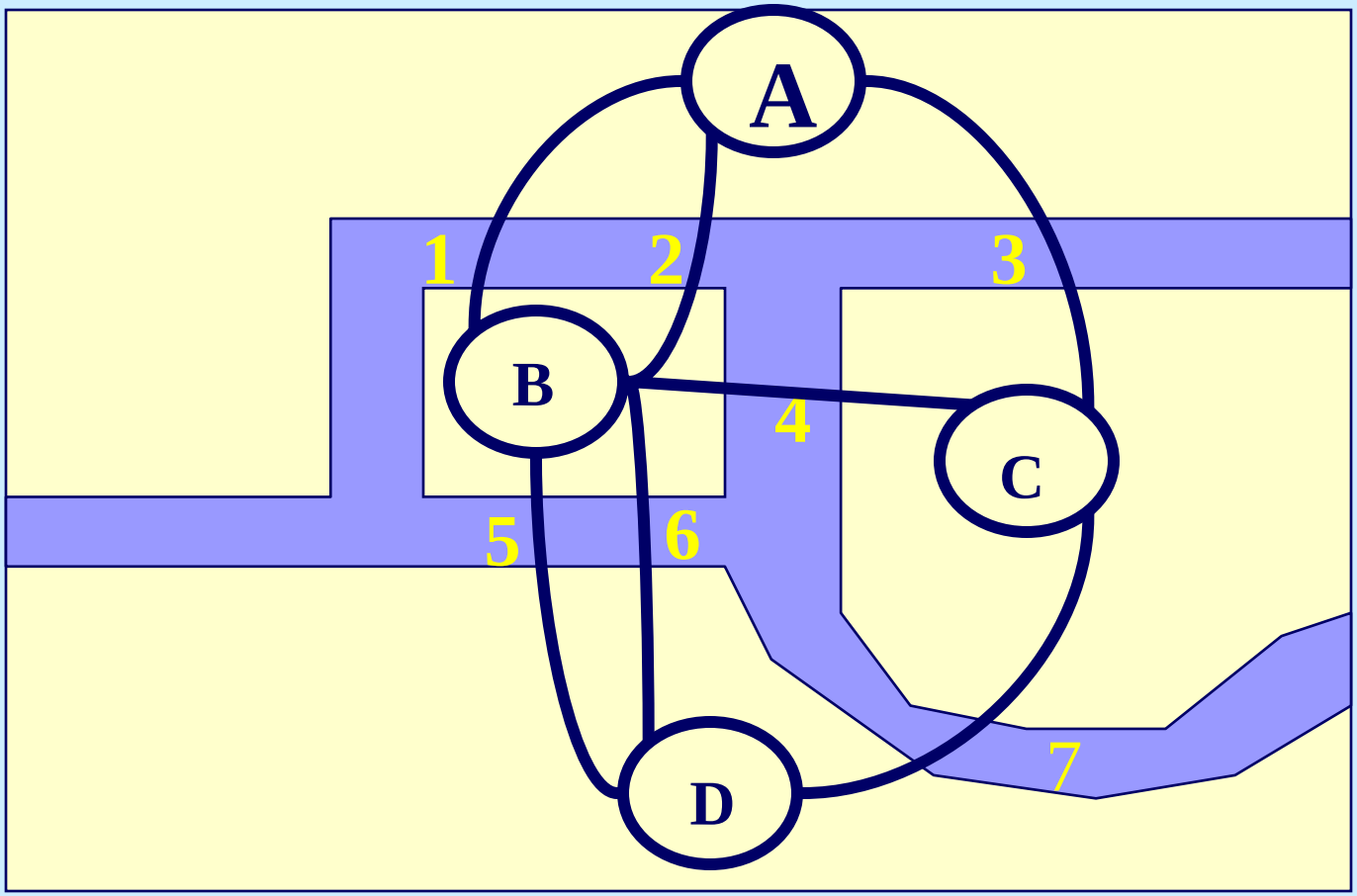
Is it possible to start in A, cross over each bridge exactly once, and end up back in A?

The Bridges of Koenigsberg: Euler 1736



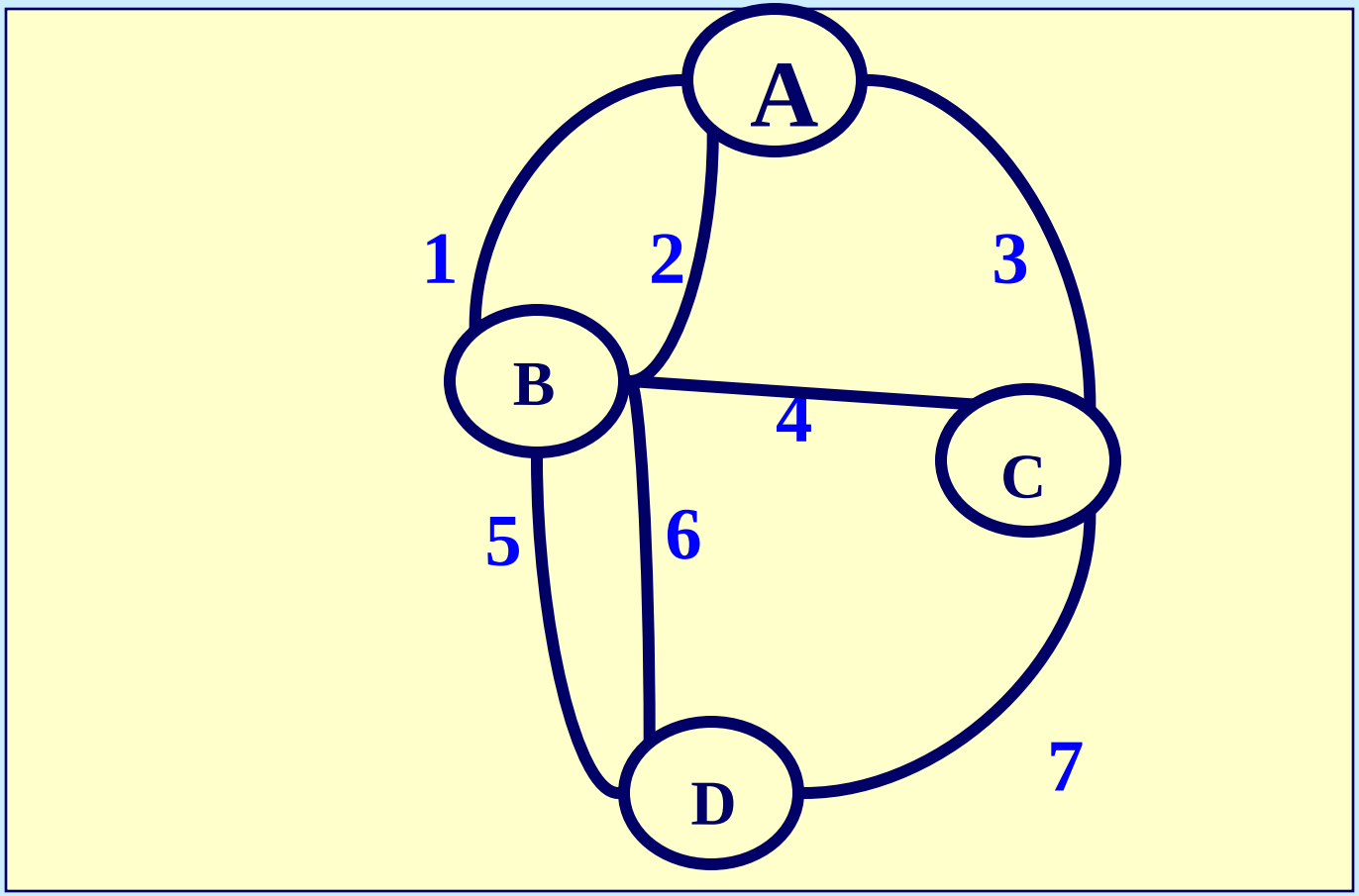
Conceptualization: Land masses are “nodes”.

The Bridges of Koenigsberg: Euler 1736



Conceptualization: Bridges are "arcs."

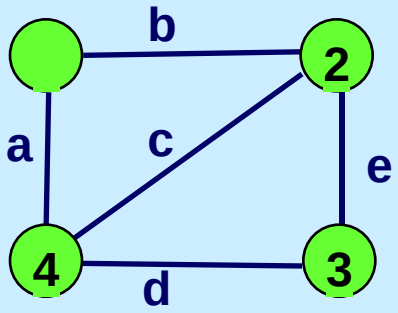
The Bridges of Koenigsberg: Euler 1736



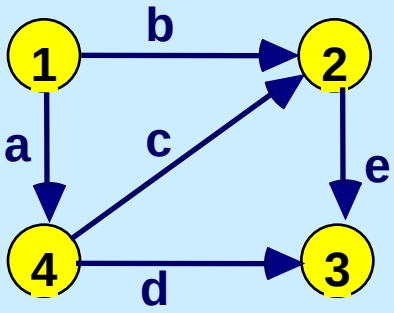
Is there a "walk" starting at A and ending at A and passing through each arc exactly once?

Notation and Terminology

Network terminology as used in AMO.



An Undirected Graph or Undirected Network



A Directed Graph or Directed Network

Network **G = (N, A)**

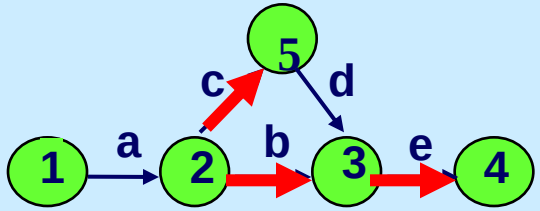
Node set **N = {1, 2, 3, 4}**

Arc Set **A = {(1,2), (1,3), (3,2), (3,4), (2,4)}**

In an undirected graph, (i,j) = (j,i)

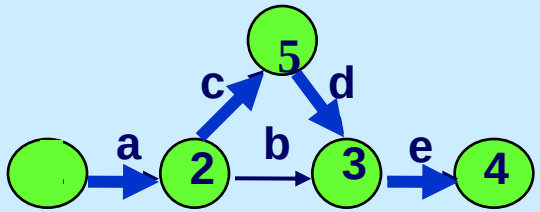
Path: **Example: 5, 2, 3, 4.**
(or 5, c, 2, b, 3, e, 4)

- **No node is repeated.**
- **Directions are ignored.**



Directed Path **. Example: 1, 2, 5, 3, 4**
(or 1, a, 2, c, 5, d, 3, e, 4)

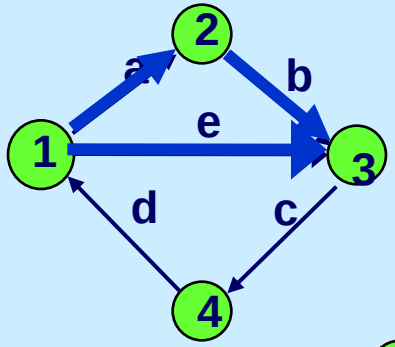
- **No node is repeated.**
- **Directions are important.**



Cycle (or circuit or loop)

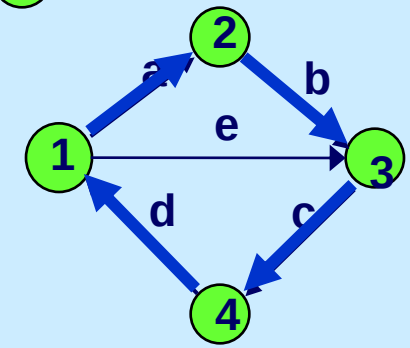
1, 2, 3, 1. (or 1, a, 2, b, 3, e)

- **A path with 2 or more nodes, except that the first node is the last node.**
- **Directions are ignored.**

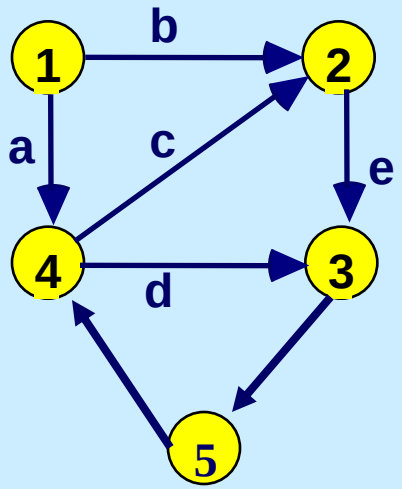


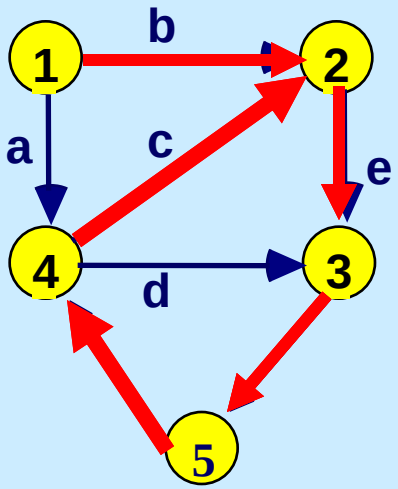
Directed Cycle: (1, 2, 3, 4, 1) or
1, a, 2, b, 3, c, 4, d, 1

- **No node is repeated.**
- **Directions are important.**



Walks



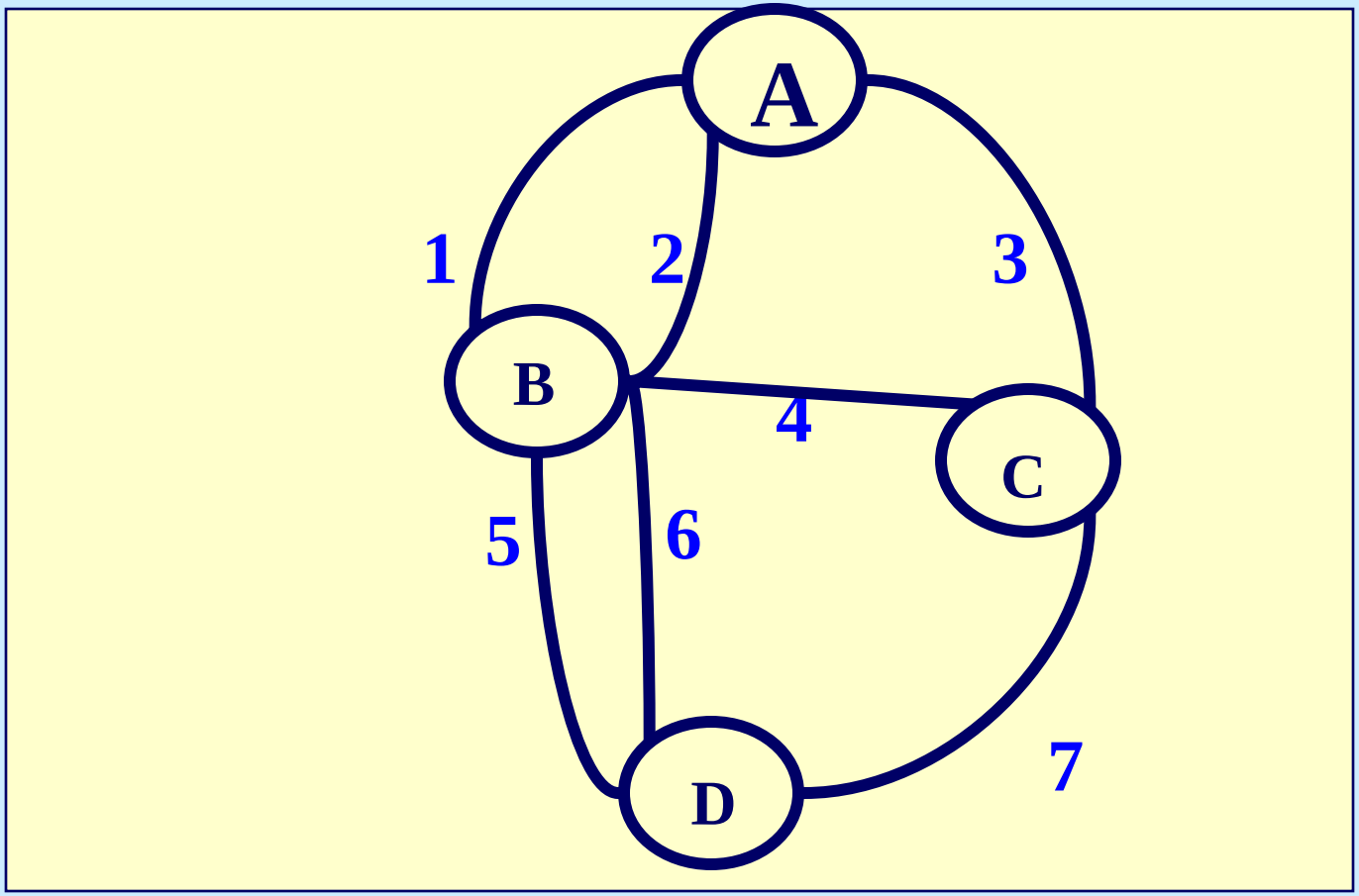


Walks **are paths that can repeat nodes and arcs**

Example of a *directed walk*: 1-2-3-5-4-2-3-5

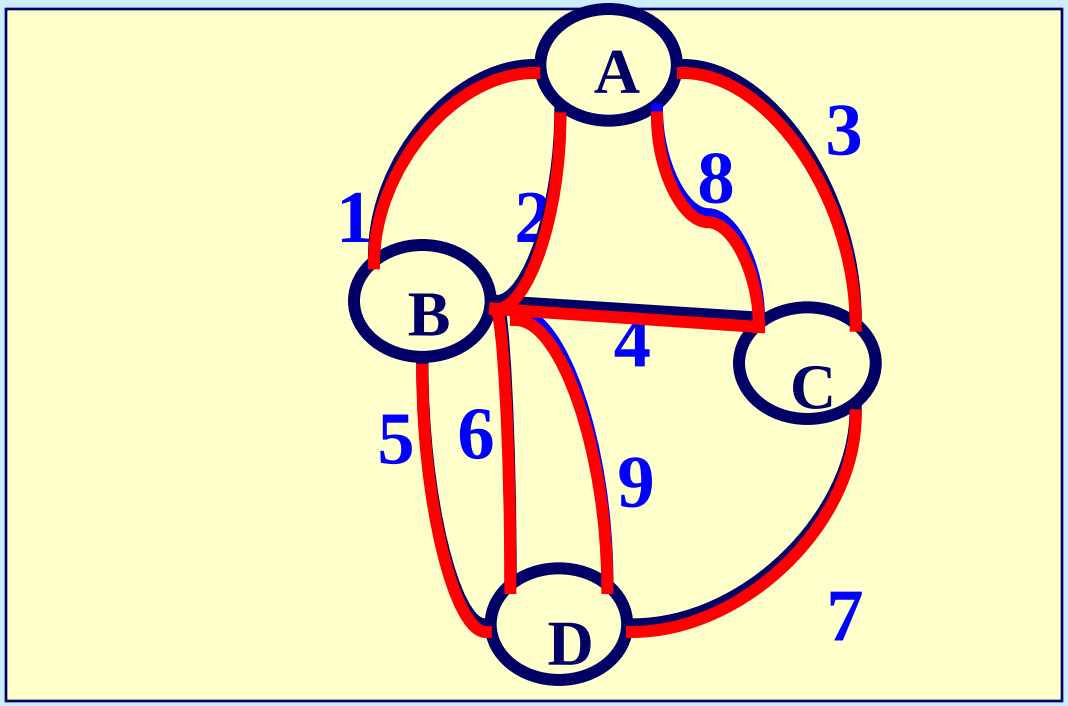
A walk is *closed* if its first and last nodes are the
A closed walk is a cycle except that it can repeat nodes and arcs.

The Bridges of Koenigsberg: Euler 1736



Is there a "walk" starting at A and ending at A and passing through each arc exactly once?
Such a walk is called an *eulerian cycle*.

Adding two bridges creates such a walk

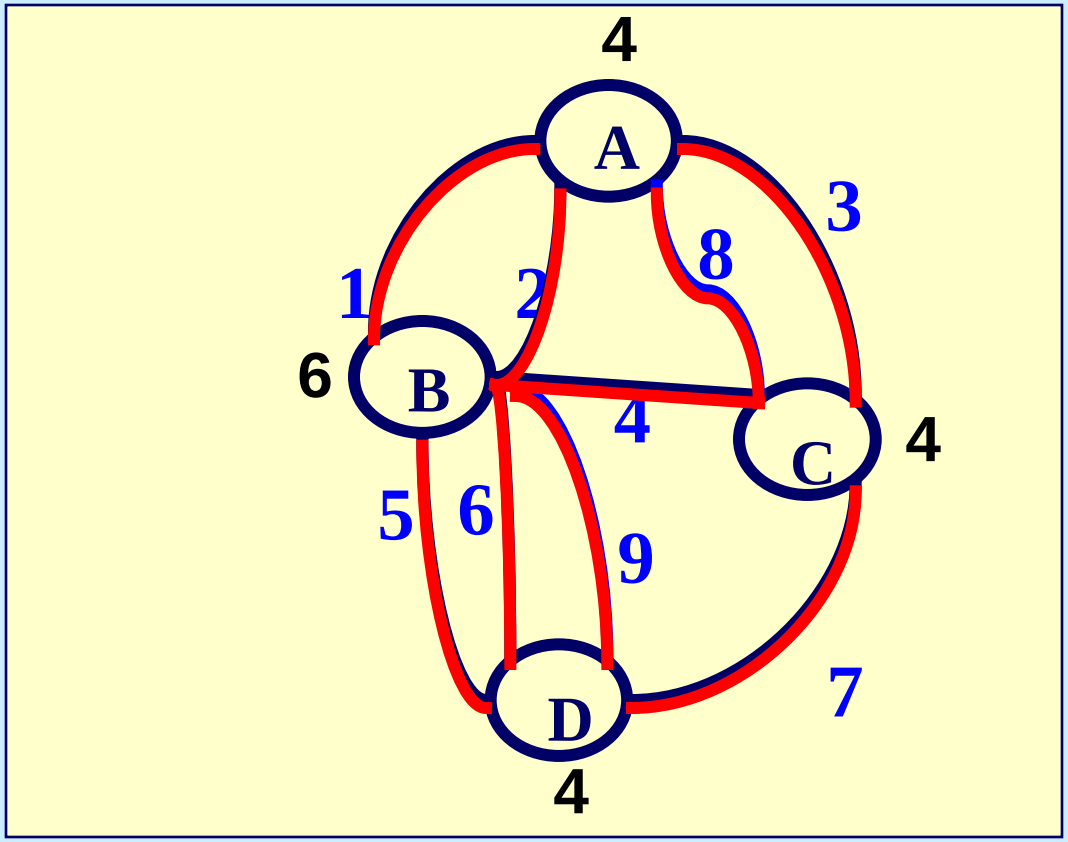


Here is the walk.

A, 1, B, 5, D, 6, B, 4, C, 8, A, 3, C, 7, D, 9, B, 2, A

Note: the number of arcs incident to B is twice the number of times that B appears on the walk.

On Eulerian Cycles



The degree of a node in an undirected graph is the number of incident arcs

Theorem. An undirected graph has an eulerian cycle if and only if

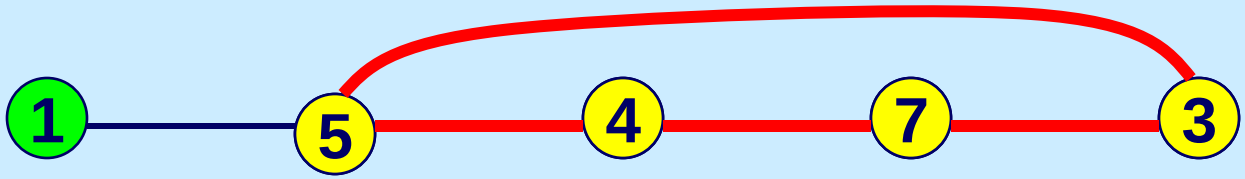
(1) every node degree is even and

(2) the graph is connected (that is, there is a path from each node to each other node).

More on Euler's Theorem

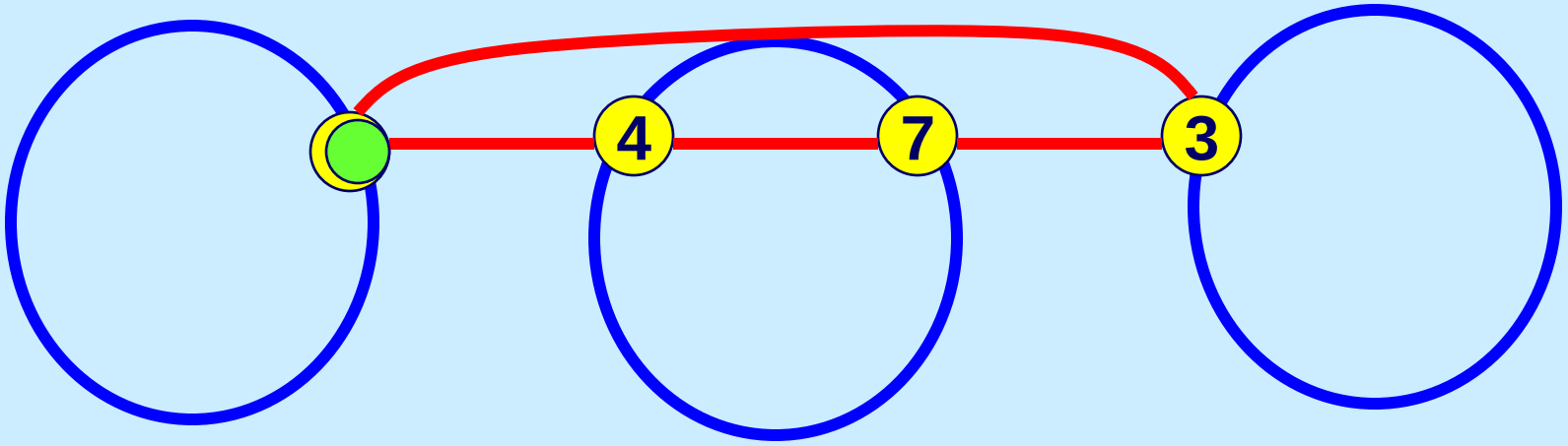
- Necessity of two conditions:
  - Any eulerian cycle "visits" each node an even number of times
  - Any eulerian cycle shows the network is connected
    - caveat: nodes of degree 0

- Sufficiency of the condition
  - Assume the result is true for all graphs with fewer than |A| arcs.
  - Start at some node, and take a walk until a cycle C is found.



More on Euler's Theorem

- Sufficiency of the condition
  - Start at some node, and take a walk until a cycle C is found.
  - Consider G' = (N, A\C)
    - the degree of each node is even
    - each component is connected
- So, G' is the union of Eulerian cycles
- Connect G' into a single eulerian cycle by adding C.



Comments on Euler's theorem

1. **It reflects how proofs are done in class, often in outline form, with key ideas illustrated.**

2. **However, this proof does not directly lead to an efficient algorithm. (More on this in two lectures.)**

3. **Usually we focus on efficient algorithms.**

15.082/6.855J Subject Goals:

1. To present students with a knowledge of the state-of-the art in the theory and practice of solving network flow problems.
   - A lot has happened since 1736

2. To provide students with a rigorous analysis of network flow algorithms.
   - computational complexity & worst case analysis

3. To help each student develop his or her own intuition about algorithm development and algorithm analysis.

Homework Sets and Grading

- **Homework Sets**
  - **6 assignments**
  - **4 points per assignment**
  - **lots of practice problems with solutions**

- **Grading**
  - **homework: 24 points**
  - **Project 16 points**
  - **Midterm 1: 30 points**
  - **Midterm 2: 30 points**

Class discussion

- **Have you seen network models elsewhere?**
- **Do you have any specific goals in taking this subject?**

Mental break

Which nation gave women the right to vote first?

New Zealand.

Which Ocean goes to the deepest depths?

Pacific Ocean

What is northernmost land on earth?

Cape Morris Jessep in Greenland

Where is the Worlds Largest Aquarium?

Epcot Center in Orlando, FL

Mental break

What country has not fought in a war since 1815?

Switzerland

What does the term Prima Donna mean in Opera?

The leading female singer

What fruit were Hawaiian women once forbidden by law to eat?

The coconut

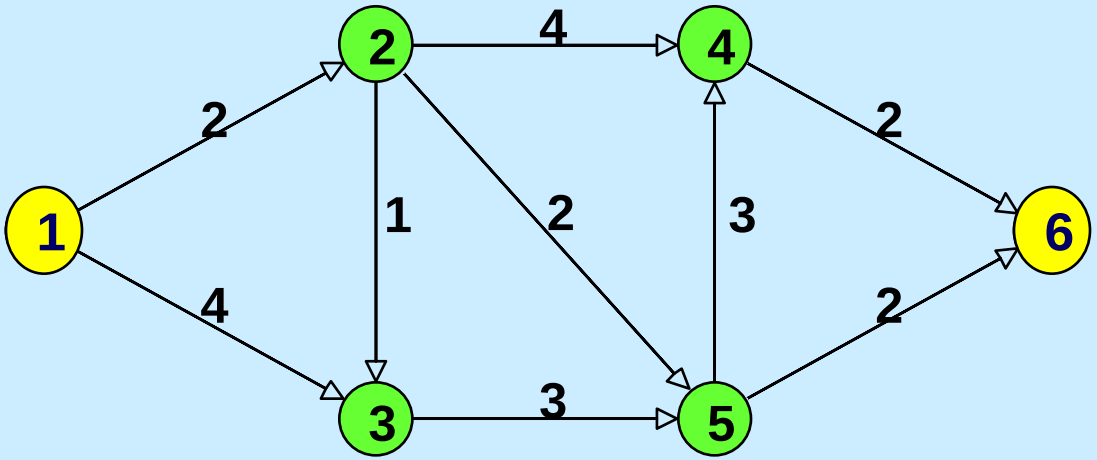
What's the most common non-contagious disease in the world?

Tooth decay

Three Fundamental Flow Problems

- The shortest path problem
- The maximum flow problem
- The minimum cost flow problem

The shortest path problem



Consider a network G = (N, A) in which there is an origin node s and a destination node t.

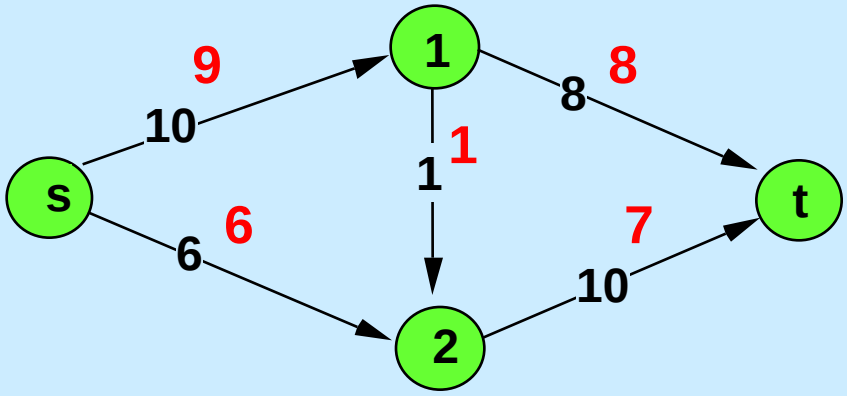
standard notation: n = |N|, m = |A|

What is the shortest path from s to t?

The Maximum Flow Problem

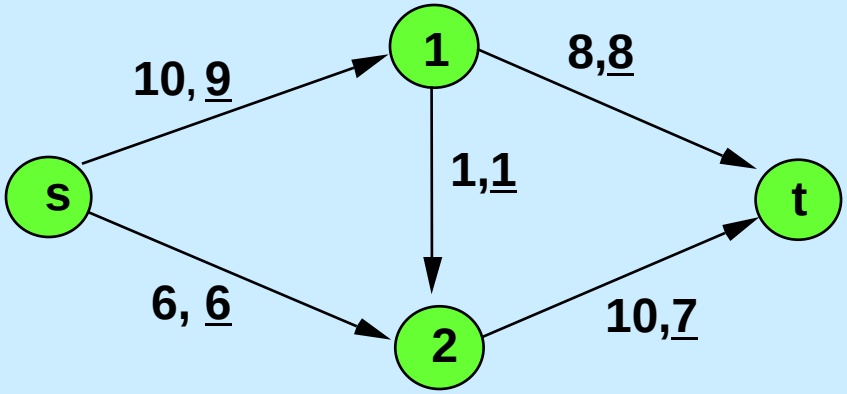
- Directed Graph G = (N, A).
  - Source s
  - Sink t
  - Capacities u_{ij} on arc (i,j)
  - Maximize the flow out of s, subject to

- Flow out of i = Flow into i, for i ≠ s or t.



A Network with Arc Capacities (and the maximum flow)

Representing the Max Flow as an LP



Flow out of i - Flow into i = 0 for i ≠ s or t.

$$\max \quad v$$

$$\text{s.t} \quad x_{s1} + x_{s2} = v$$

$$-x_{s1} + x_{12} + x_{1t} = 0$$

$$-x_{s2} - x_{12} + x_{2t} = 0$$

$$-x_{1t} - x_{2t} = -v$$

$$0 \le x_{ij} \le u_{ij} \text{ for all } (i,j)$$

$$\max \quad v$$

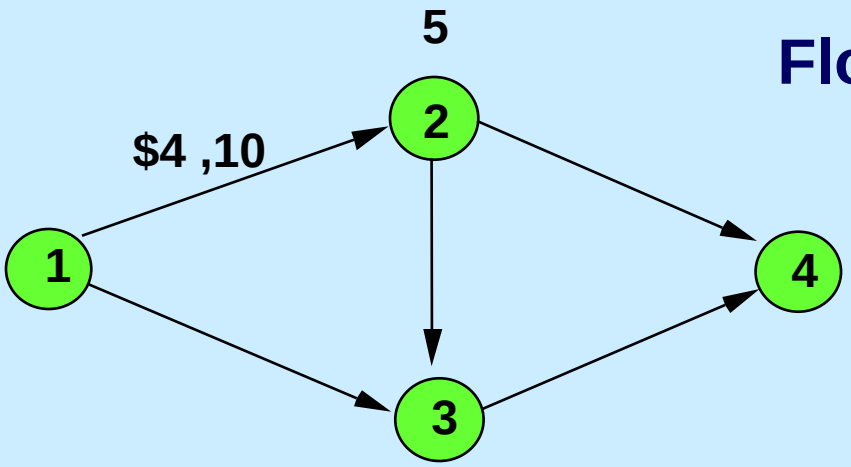
$$\text{s.t.} \quad \sum_j x_{sj} = v$$

$$\sum_j x_{ij} - \sum_j x_{ji} = 0 \text{ for each } i \neq s \text{ or } t$$

$$\text{s.t.} \quad -\sum_i x_{it} = -v$$

$$0 \le x_{ij} \le u_{ij} \text{ for all } (i,j)$$

Min Cost Flows



Flow out of i - Flow into i = b(i)

Each arc has a linear cost and a capacity

$$\min \sum_{i,j} c_{ij}x_{ij}$$

$$\text{s.t} \quad \sum_j x_{ij} - \sum_j x_{ji} = b(i) \quad \text{for each } i$$

$$0 \le x_{ij} \le u_{ij} \text{ for all } (i,j)$$

Covered in detail in Chapter 1 of AMO

Where Network Optimization Arises

- **Transportation Systems**
  - **Transportation of goods over transportation networks**
  - **Scheduling of fleets of airplanes**
- **Manufacturing Systems**
  - **Scheduling of goods for manufacturing**
  - **Flow of manufactured items within inventory systems**
- **Communication Systems**
  - **Design and expansion of communication systems**
  - **Flow of information across networks**
- **Energy Systems, Financial Systems, and much more**

Next topic: computational complexity

- What is an efficient algorithm?
- How do we measure efficiency?
- “Worst case analysis”
- but first …

Measuring Computational Complexity

- **Consider the following algorithm for adding two m × n matrices A and B with coefficients a(,) and b(,).**

```
begin
    for  i = 1 to m  do
    for  j = 1 to n  do  c(i,j) := a(i,j) + b(i,j)
end
```

What is the running time of this algorithm?

- **Let's measure it as precisely as we can as a function of n and m.**
- **Is it 2nm, or 3nm, or what?**

Worst case versus average case

- **How do we measure the running time?**
- **What are the basic steps that we should count?**

Compute the running time precisely.

Operation Number (as a function of m,n)

- **Additions**
- **Assignments**
- **Comparisons**
- **Multiplications**

Towards Computational Complexity

1. We will ignore running time constants.

2. Our running times will be stated in terms of relevant problem parameters, e.g., nm.

3. We will measure everything in terms of worst case or most pessimistic analysis (performance guarantees.)

4. All arithmetic operations are assumed to take one step,
(or a number of steps that is bounded by a constant).

A Simpler Metric for Running Time.

- Operation Number (as a function of m,n)
- Additions $\leq c_1 mn$ for some c_1 and m, n ≥ 1
  - O(mn) steps
- Assignments $\leq c_2 mn$ for some c_2 and m, n ≥ 1
  - O(mn) steps
- Comparisons $\leq c_3 mn$ for some c_3 and m, n ≥ 1
  - O(mn) steps
- TOTAL $\leq c_4 mn$ for some c_4 and m, n ≥ 1
  - O(mn) steps

Simplifying Assumptions and Notation

- **MACHINE MODEL: *Random Access Machine* (RAM).**
  This is the computer model that everyone is used to. It allows the use of arrays, and it can select any element of an array or matrix in O(1) steps.
  - **c(i,j) := a(i,j) + b(i,j).**

- **Integrality Assumption. All numbers are integral (unless stated otherwise.)**

Size of a problem

- The size of a problem is the number of **bits** needed to represent the problem.

- The size of the n × m matrix A is not nm.
  - If each matrix element has K bits, the size is nmK
  - e.g., if max $2^{107} < a_{ij} < 2^{108}$, then K = 108.
  - $K = O(\log (a_{max}))$.

Polynomial Time Algorithms

- **We say that an algorithm runs in *polynomial time* if the number of steps taken by an algorithm on any instance `I` is bounded by a polynomial in the size of `I`.**

- **We say that an algorithm runs in *exponential time* if it does not run in polynomial time.**

- **Example 1: finding the determinant of a matrix can be done in $O(n^3)$ steps.**
  - **This is polynomial time.**

Polynomial Time Algorithms

- **Example 2: We can determine if n is prime by dividing n by every integer less than n.**
  - **This algorithm is exponential time.**
  - **The size of the instance is log n**
  - **The running time of the algorithm is O(n).**
  - **Side note: there is a polynomial time algorithm for determining if n is prime.**
- **Almost all of the algorithms presented in this class will be polynomial time.**
- **One can find an Eulerian cycle (if one exists) in O(m) steps.**
- **There is no known polynomial time algorithm for finding a min cost traveling salesman tour**

On polynomial vs exponential time

- We contrast two algorithm, one that takes 30,000 n^3 steps, and one that takes 2^n steps.

- Suppose that we could carry out 1 billion steps per second.

# of nodes	30,000 n^3 steps	2^n steps
n = 30,	0.81 seconds	1 second
n = 40,	1.92 seconds	17 minutes
n = 50	3.75 seconds	12 days
n = 60	6.48 seconds	31 years

On polynomial vs. exponential time

- **Suppose that we could carry out 1 trillion steps per second, and instantaneously eliminate 99.9999999% of all solutions as not worth considering**

# of nodes	1,000 n^{10} steps	2^n steps
n = 70,	2.82 seconds	1 second
n = 80,	10.74 seconds	17 minutes
n = 90	34.86 seconds	12 days
n = 100	100 seconds	31 years

Overview of today's lecture

- Eulerian cycles
- Network Definitions
- Network Applications
- Introduction to computational complexity

Upcoming Lectures

- **Lecture 2: Review of Data Structures**
  - **even those with data structure backgrounds are encouraged to attend.**
- **Lecture 3. Graph Search Algorithms.**
  - **how to determine if a graph is connected**
  - **and to label a graph**
  - **and more**

15.082J / 6.855J / ESD.78J Network Optimization

Fall 2010