Fun with Graphs!

Tandy Warnow

warnow@Illinois.edu

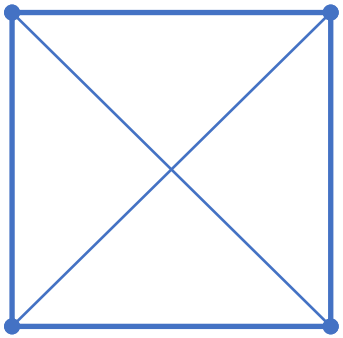
http://tandy.cs.Illinois.edu

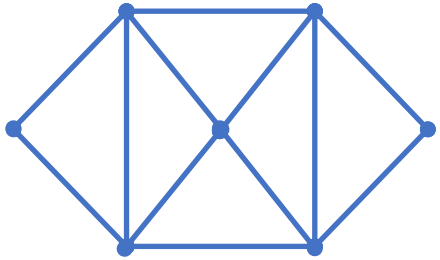
Graphs

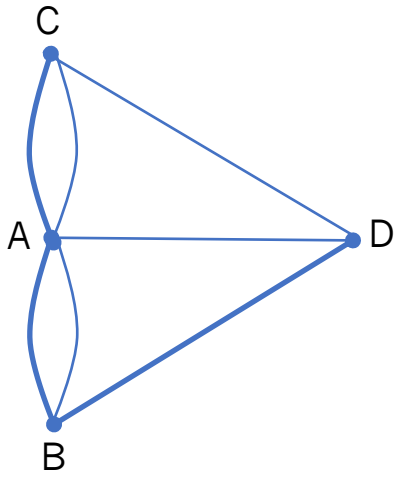
- Vertices!
- Edges!

Graphs!

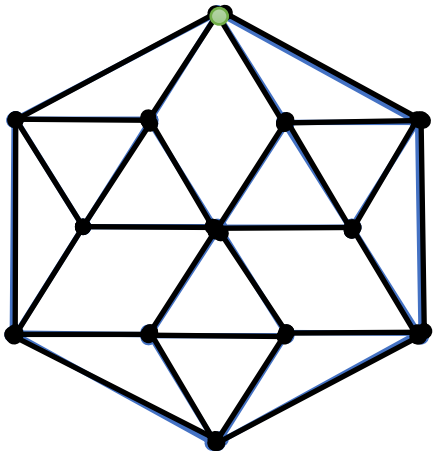
- How many vertices in each graph?
- What are the degrees of each vertex?







More Graphs!



More graphs

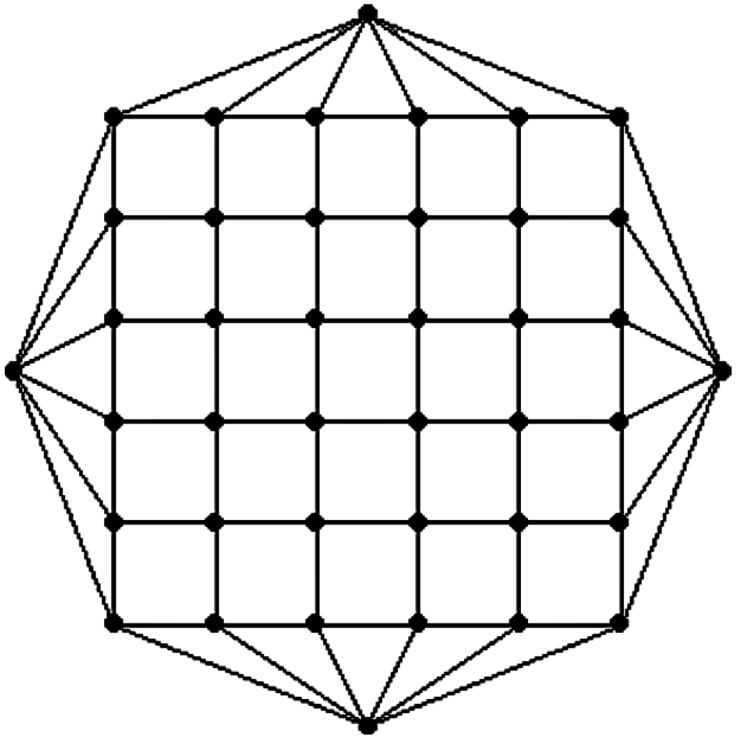


Image from: http://www.cs.sunysb.edu/~skiena/combinatorica/animations/euler.html

Graphs

- Vertices!
- Edges!

- Graphs are used in computer science to solve problems.
- For example:
  - Finding the largest number of people in a group, all of whom like each other.
  - Finding a shortest path between two cities (using highways)
  - Inviting friends to parties (and avoiding fights)

Graphs: Inviting people to parties

- I have a lot of friends, but not everyone likes everyone else.
- I want to invite them all to a party, but I don’t want to have any fights.
- So I need more than one party. I don’t want to have too many parties.
- What should I do?

(Suppose I know who likes who.)

Graphs: Can I do this with 2 parties?

- I have a lot of friends, but not everyone likes everyone else.
- I want to invite them all to a party, but I don't want to have any fights.
- Can I do this with two parties?

Draw a graph, and try to figure it out.

Graphs: Can I do this with 3 parties?

- I have a lot of friends, but not everyone likes everyone else.
- I want to invite them all to a party, but I don't want to have any fights.
- Suppose I can't do this with 2 parties, so I try 3.

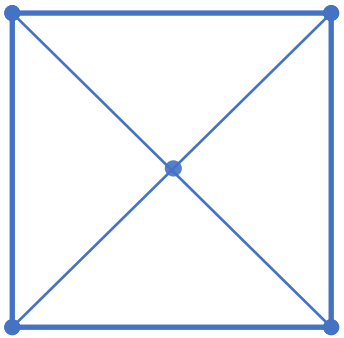
Draw a graph, and try to figure it out.

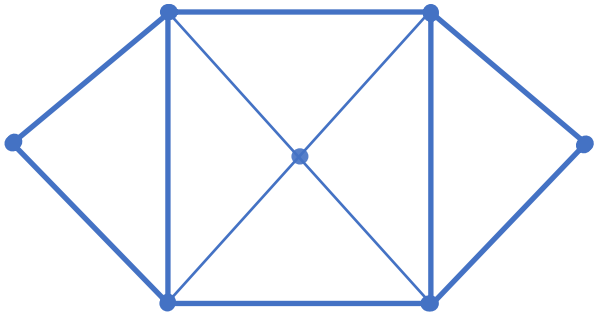
Graphs!

- We make this a graph problem!
  - Every person is a vertex
  - Two people are connected by an edge if they don't like each other
  - What am I looking for?

How many parties do I need?

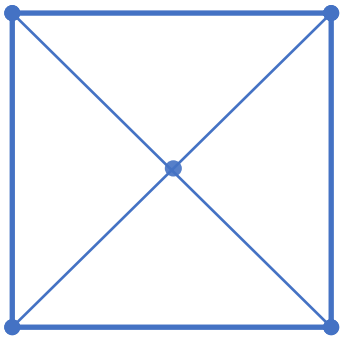
- Each vertex is a person
- Each edge means they don't like each other
- How many parties do I need?

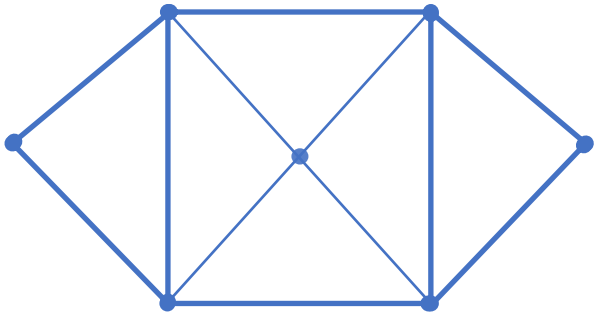




Vertex coloring: How many parties do I need?

- Each vertex is a person
- Each edge means they don't like each other
- Color the vertices so that no two adjacent vertices get the same color.
- The number of colors you need is the number of parties!



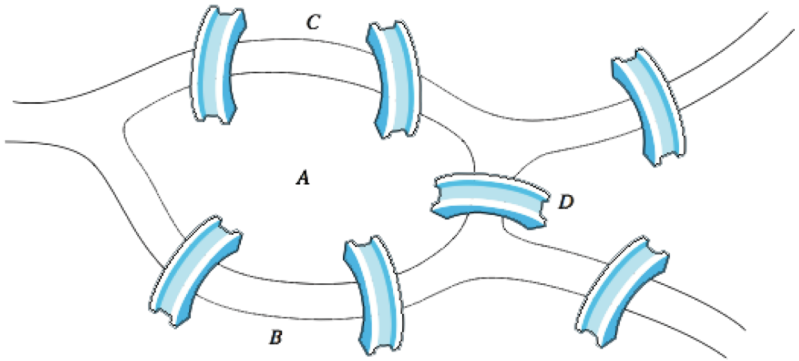


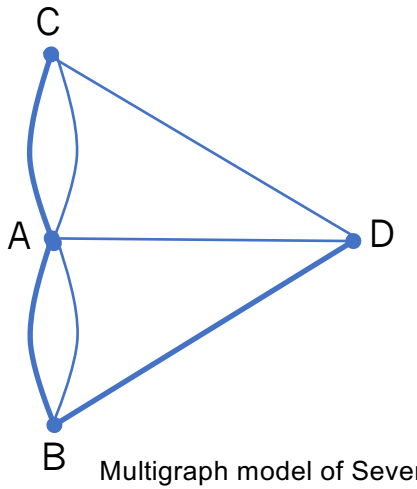
Graphs: Inviting people to parties

- I have a lot of friends, but not everyone likes everyone else.
- I want to invite them all to a party, but I don't want to have any fights. So I need more than one party!
- Think about just answering:
  - Can I do this with 2 parties?
  - Can I do this with 3 parties?
- One problem (2-colorability) is easy, one problem (3-colorability) seems hard.

The Seven Bridges of Königsberg

Question: Is it possible to start at some location in town, travel across all seven bridges without crossing any bridge twice, and return to the same starting point?

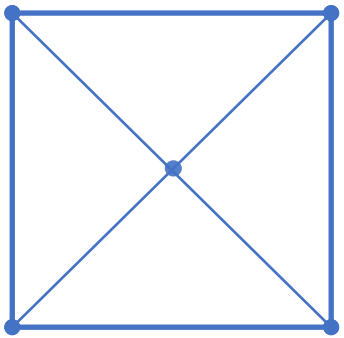


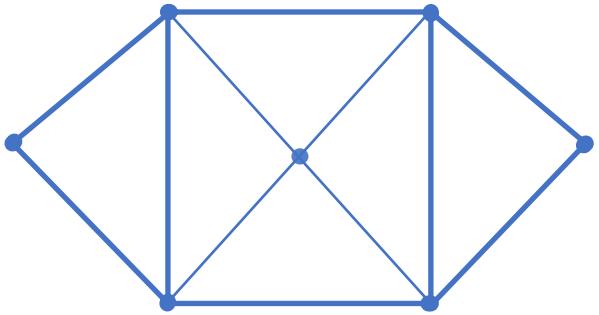


Multigraph model of Seven Bridges of Konigsberg.

Eulerian Graphs

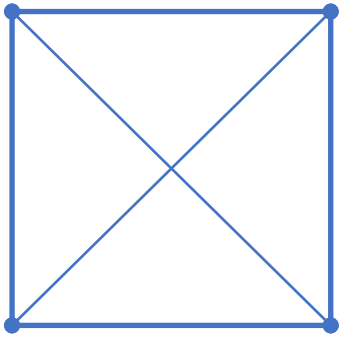
- Can you draw these graphs – starting and ending at the same vertex – and never cover any edge twice?

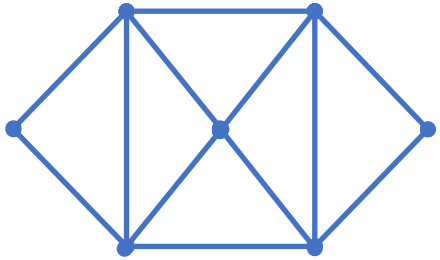


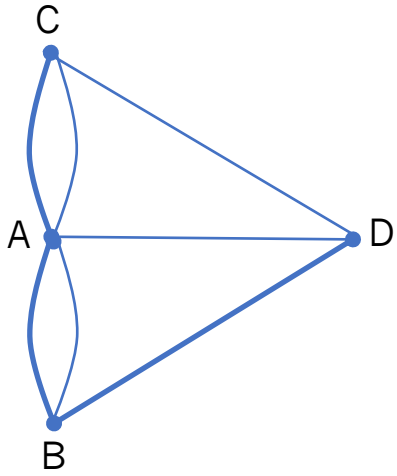


Is it easy to figure out if a graph is Eulerian?

Think about the degree of each vertex

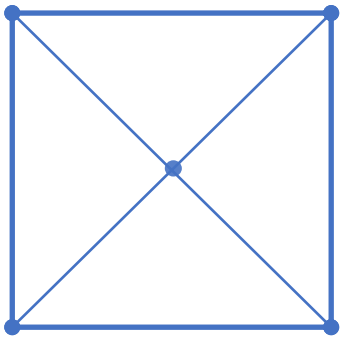


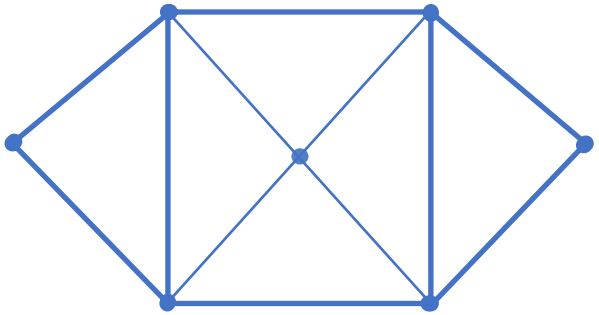




Hamiltonian Graph

- Suppose you can traverse all the vertices in the graph, starting and ending in the same place, and never use an edge twice.
- Such graphs are called Hamiltonian.
- Is it easy or hard to figure out if a graph is Hamiltonian?





Some problems are easy, some seem hard

- No one knows if it is possible to quickly answer (in polynomial time) whether a graph can be 3-colored. This is an “NP-hard” problem.
- No one knows if it is possible to quickly answer (in polynomial time) whether a graph is Hamiltonian. This is also an NP-hard problem.

- But determining if a graph can be 2-colored is easy, and determining if a graph is Eulerian is easy (i.e., both solvable in polynomial time).
- Most computer scientists think it’s IMPOSSIBLE to develop a fast algorithm for any NP-hard problems. But no one has a proof.

Write to me if you want to, at warnow@Illinois.edu

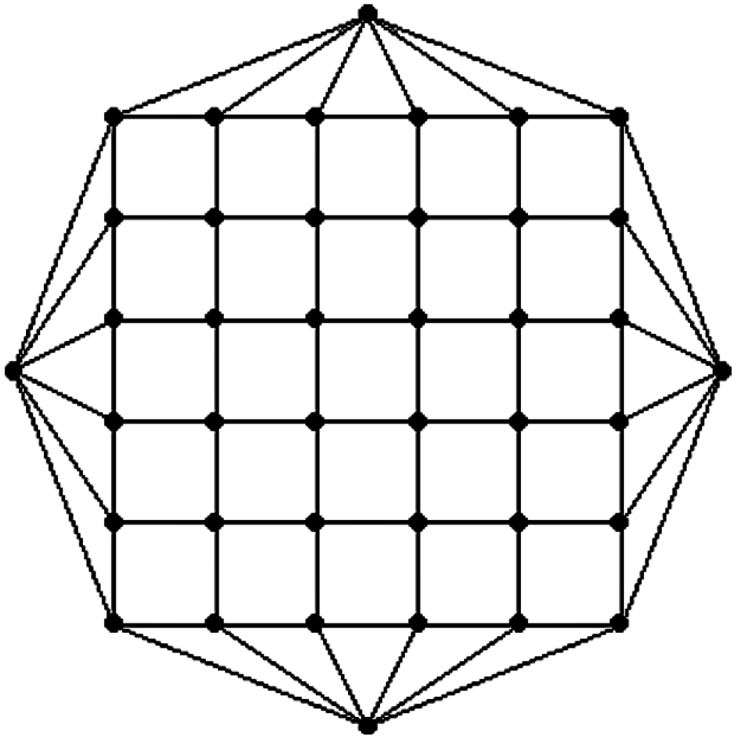


Image from: http://www.cs.sunysb.edu/~skiena/combinatorica/animations/euler.html