Graphs

Outline for Today

- **Final Thoughts on FOL**
- **Motivating Graphs**
- **Defining Graphs**
- **Undirected Connectivity**
- **Planar Graphs**
- **Graph Coloring**
- **An Overarching Question**
  - How exactly do you “do” math?

Final Thoughts on First-Order Logic

Quantifying Over Sets

- The notation

$$\forall x \in S.\ P(x)$$

  means "for any element x of set S, $P(x)$ holds."

- This is not technically a part of first-order logic; it is a shorthand for

$$\forall x.\ (x \in S \to P(x))$$

- How might we encode this concept?

$$\exists x \in S.\ P(x)$$

Answer: $\exists x.\ (x \in S \wedge P(x))$.

Note the use of ∧ instead of → here.

Quantifying Over Sets

- The syntax

$$\forall x \in S.\ \varphi$$

$$\exists x \in S.\ \varphi$$

  is allowed for quantifying over sets.
- In CS103, please do not use variants of this syntax.
- Please don't do things like this:

$$\forall x \text{ with } P(x).\ Q(x)$$

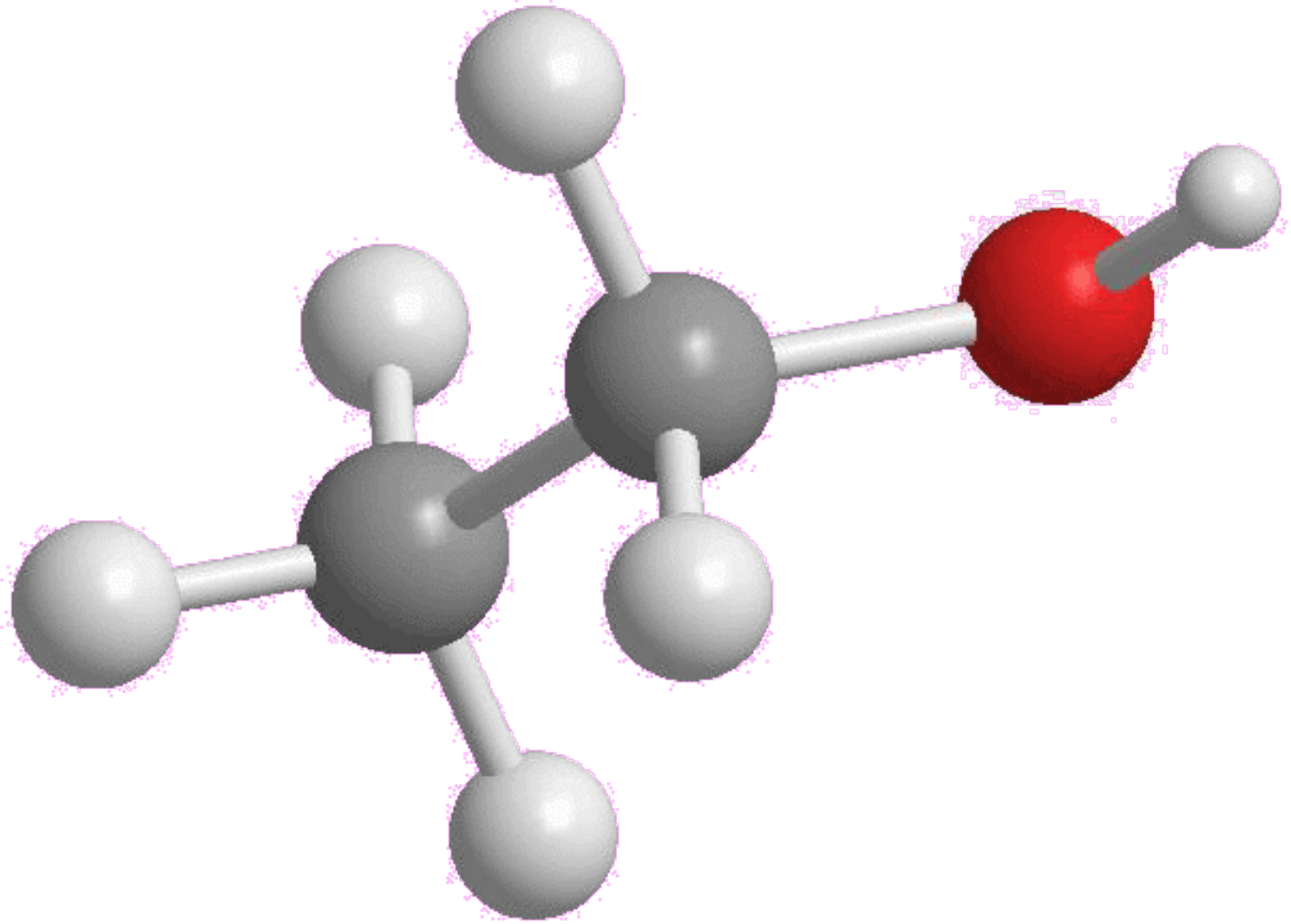
$$\forall y \text{ such that } P(y) \wedge Q(y).\ R(y).$$

Graphs

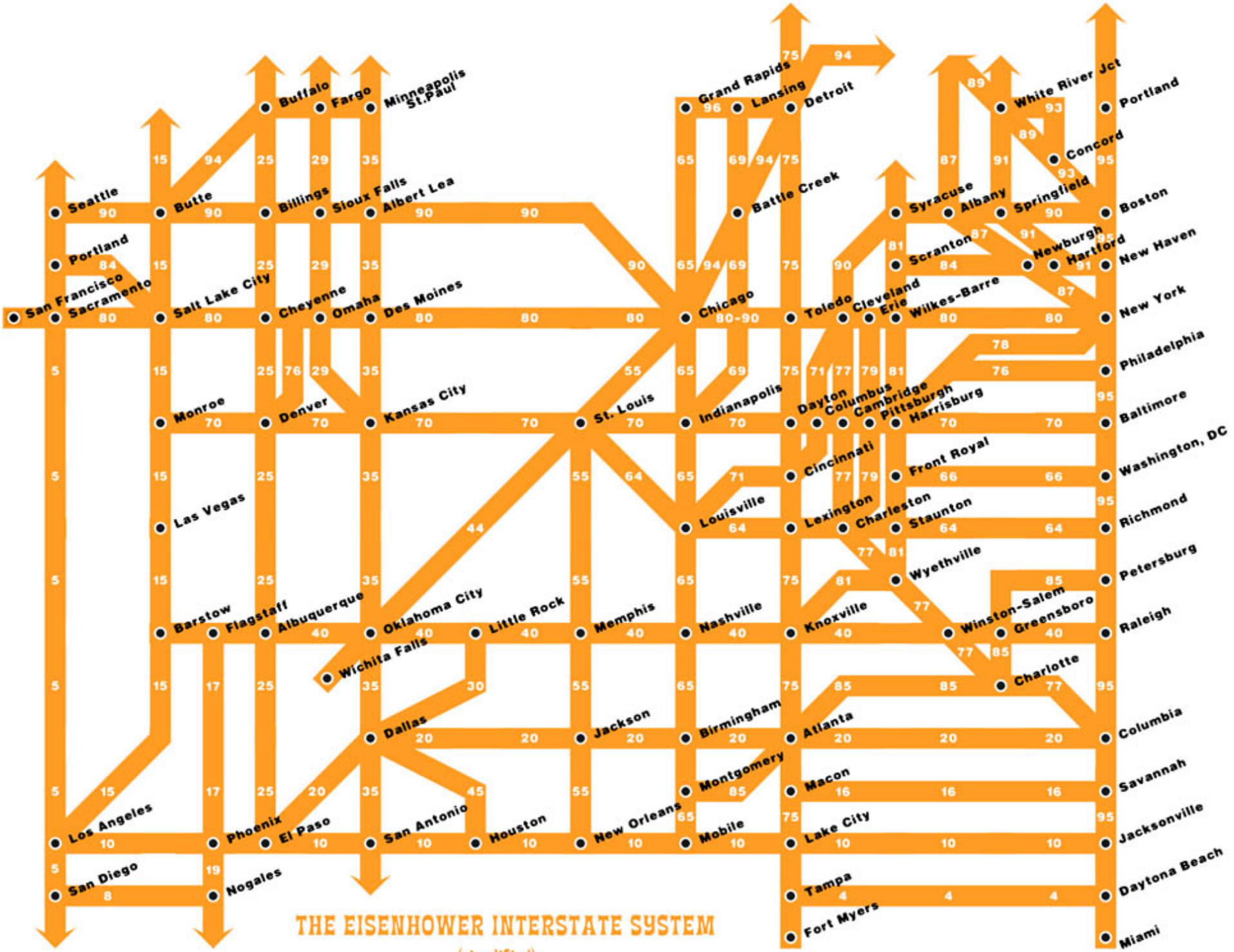
Mathematical Structures

- Just as there are common data structures in programming, there are common mathematical structures in discrete math.
- So far, we've seen simple structures like sets and natural numbers, but there are many other important structures out there.
- Over the next few weeks, we'll explore several of them.

Chemical Bonds

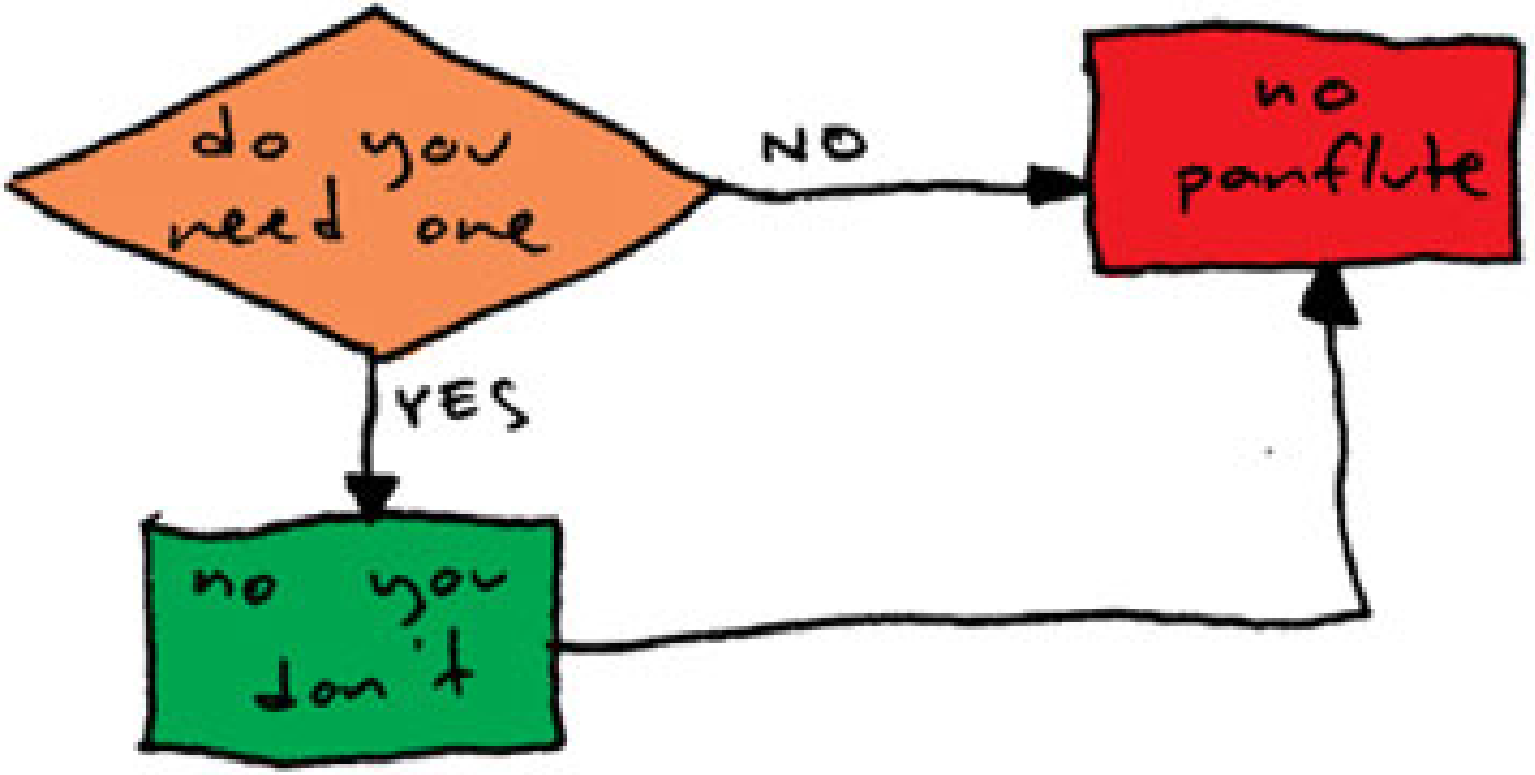


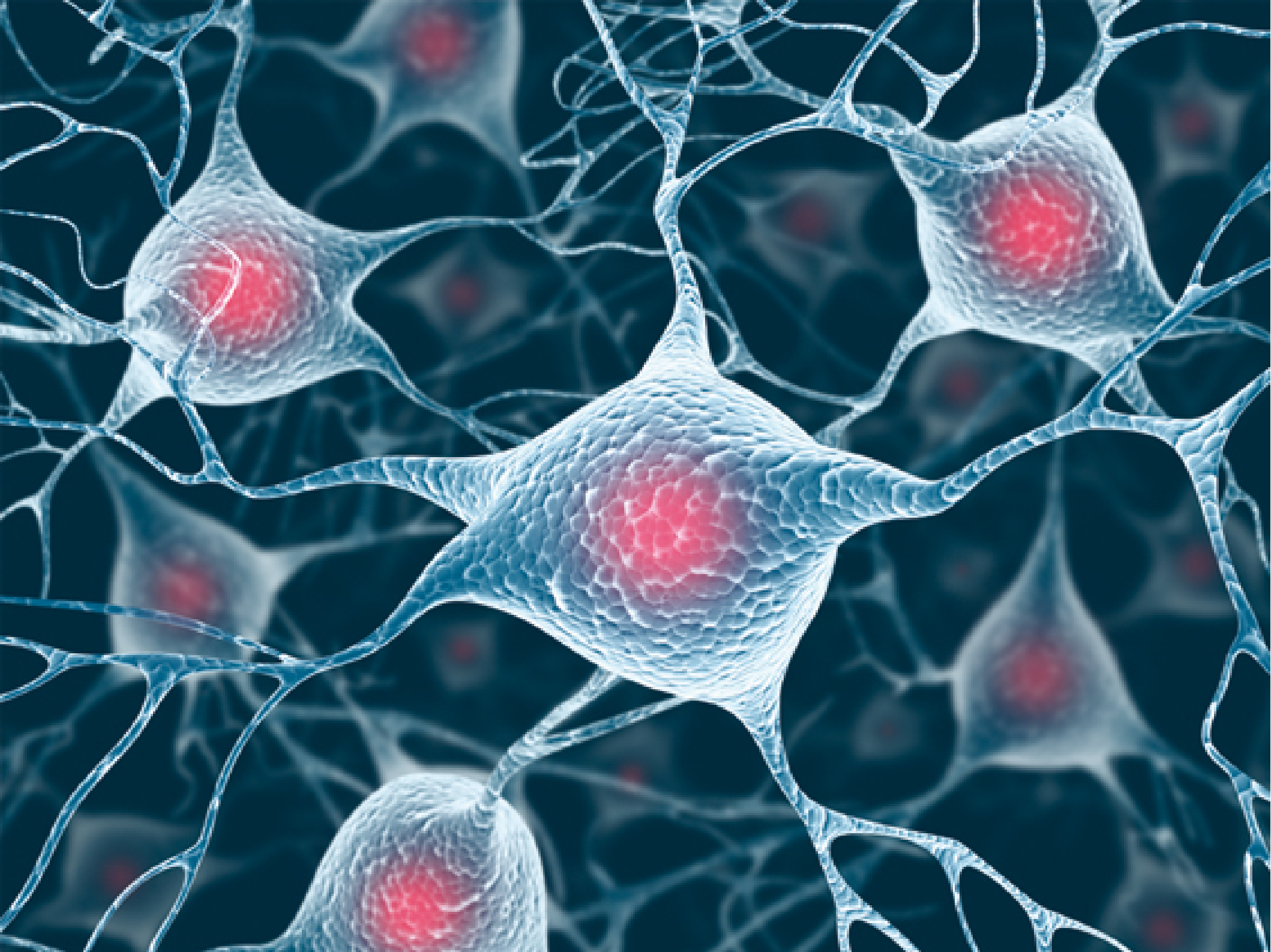
http://4.bp.blogspot.com/-xCtBJ8lKHqA/Tjm0BONWBRI/AAAAAAAAAK4/-



http://strangemaps.files.wordpress.com/2007/02/fullinterstatemap-web.jpg

PANFLUTE FLOWCHART



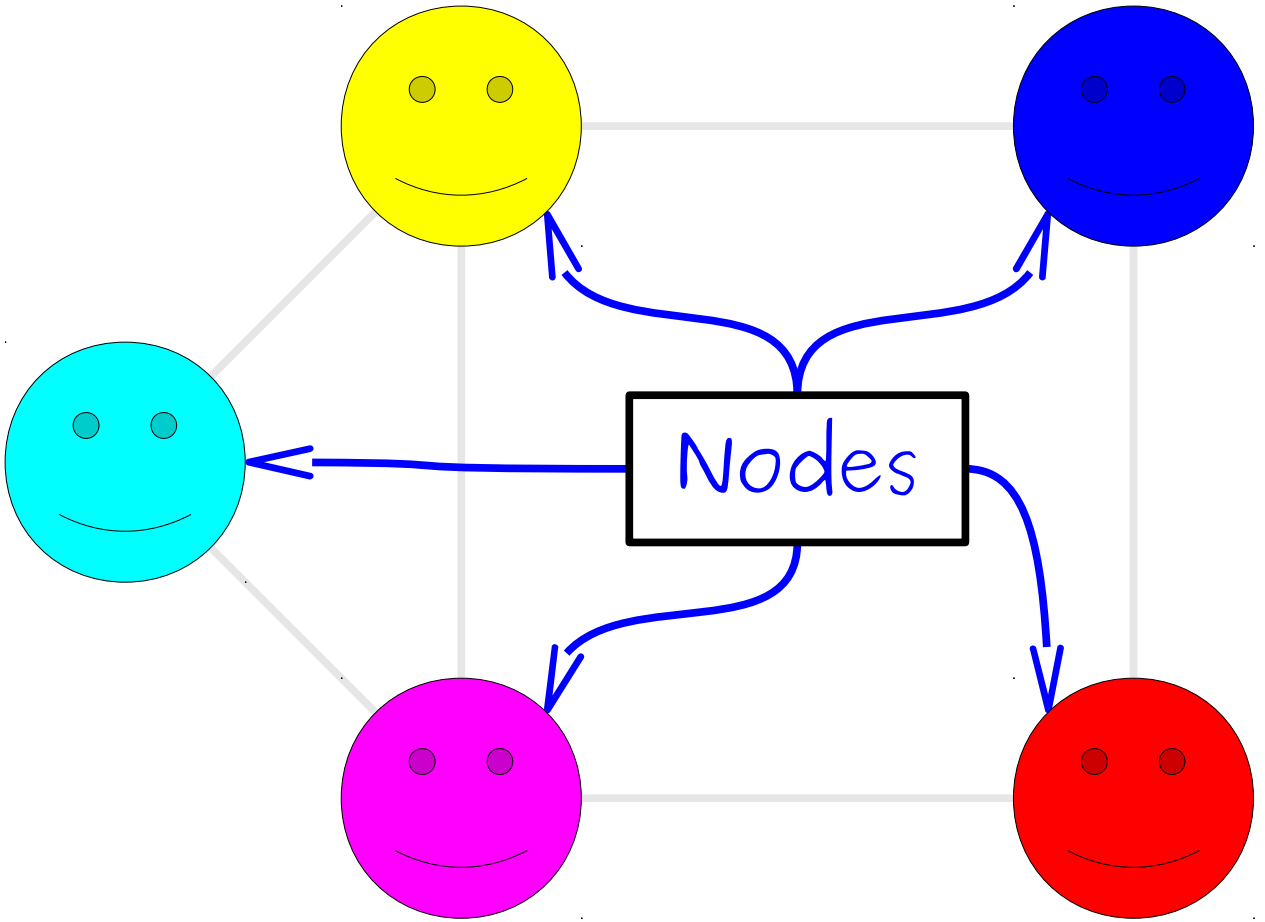


Me too!

What's in Common

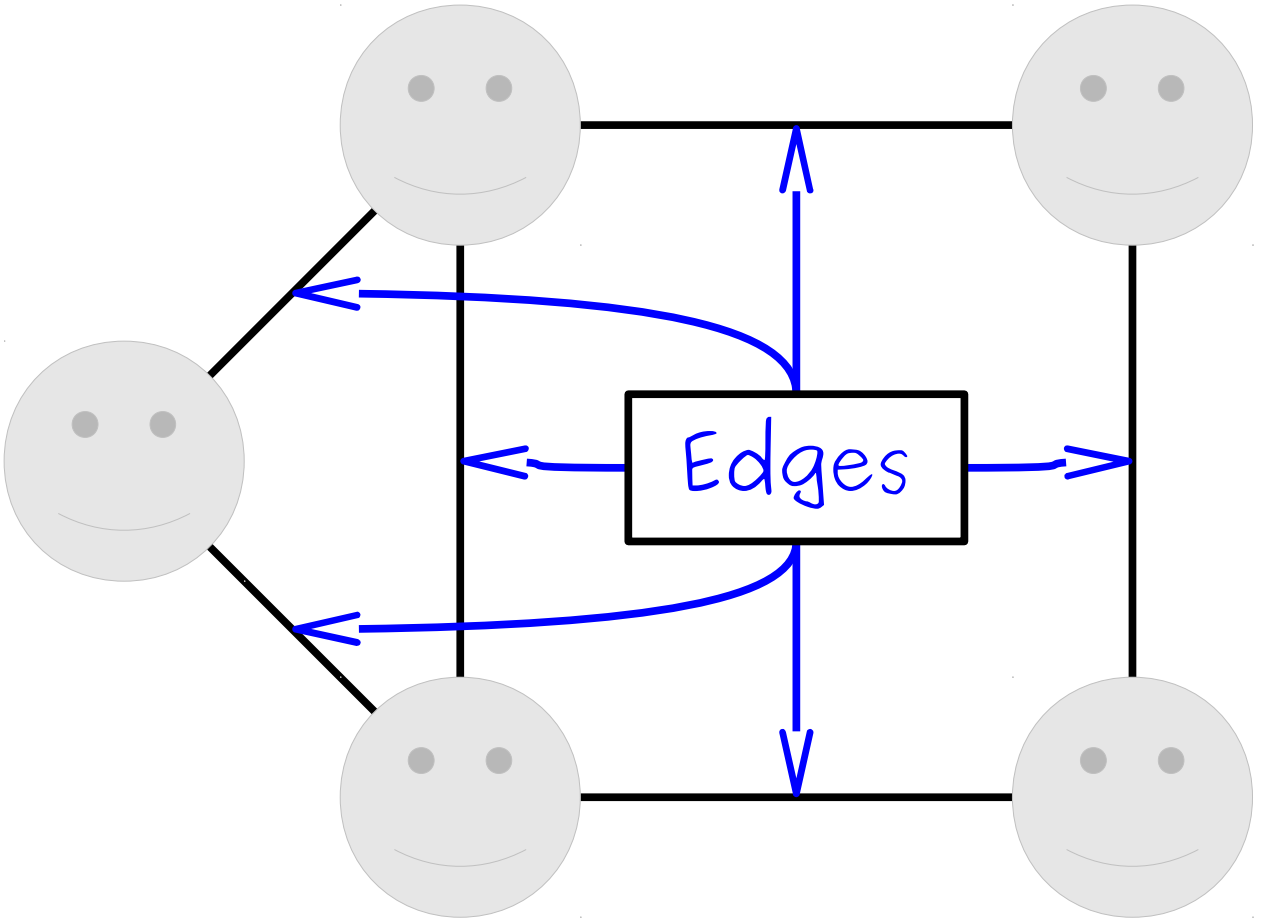
- Each of these structures consists of
  - Individual objects and
  - Links between those objects.
- Goal: find a general framework for describing these objects and their properties.

A ***graph*** is a mathematical structure for representing relationships.



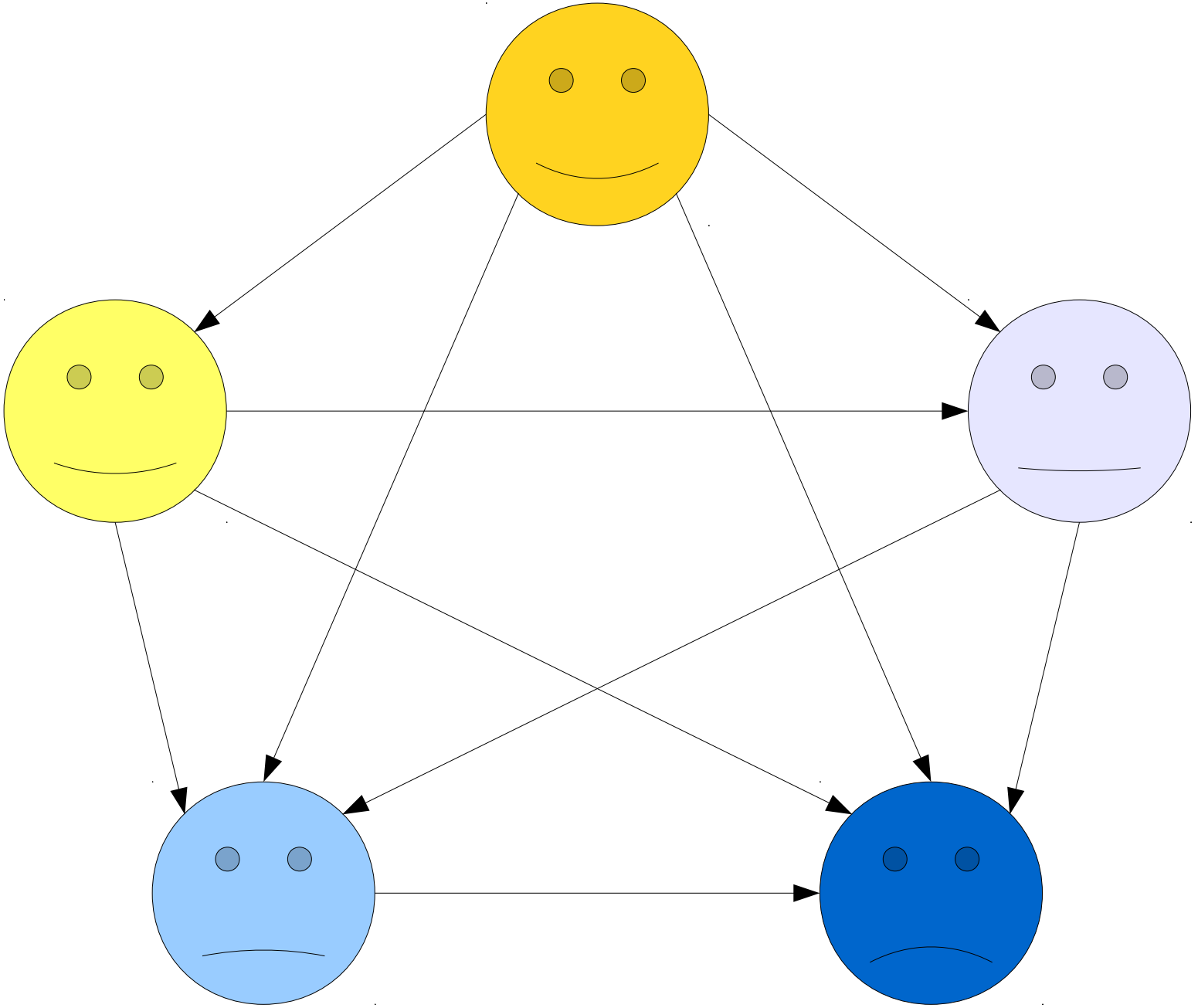
A graph consists of a set of ***nodes*** (or ***vertices***) connected by ***edges*** (or ***arcs***)

A ***graph*** is a mathematical structure for representing relationships.

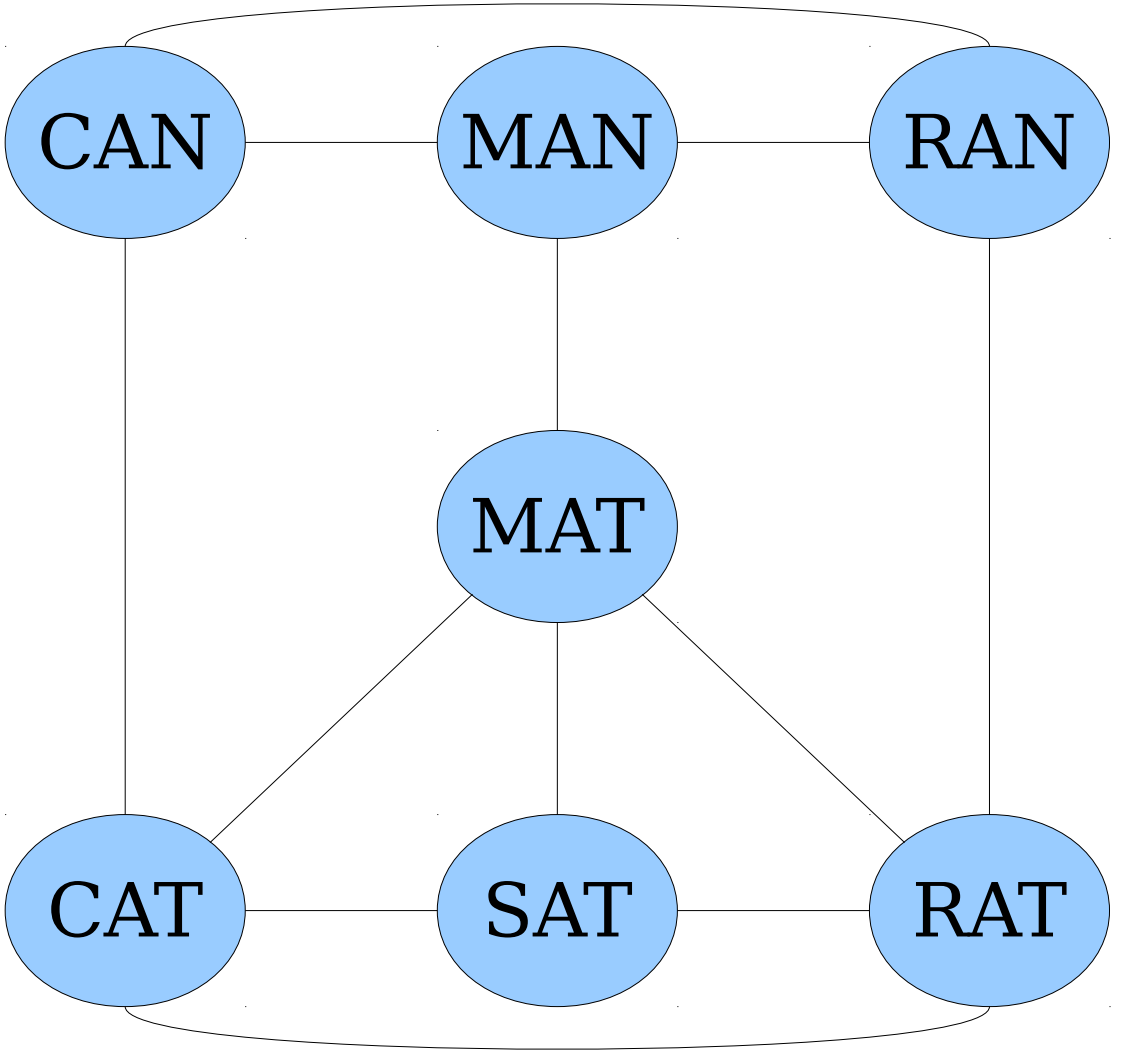


A graph consists of a set of ***nodes*** (or ***vertices***) connected by ***edges*** (or ***arcs***)

Some graphs are ***directed***.



Some graphs are ***undirected***.



Formalizing Graphs

- How might we define a graph mathematically?
- Need to specify
  - What the nodes in the graph are, and
  - What the edges are in the graph.
- The nodes can be pretty much anything.
- What about the edges?

Ordered and Unordered Pairs

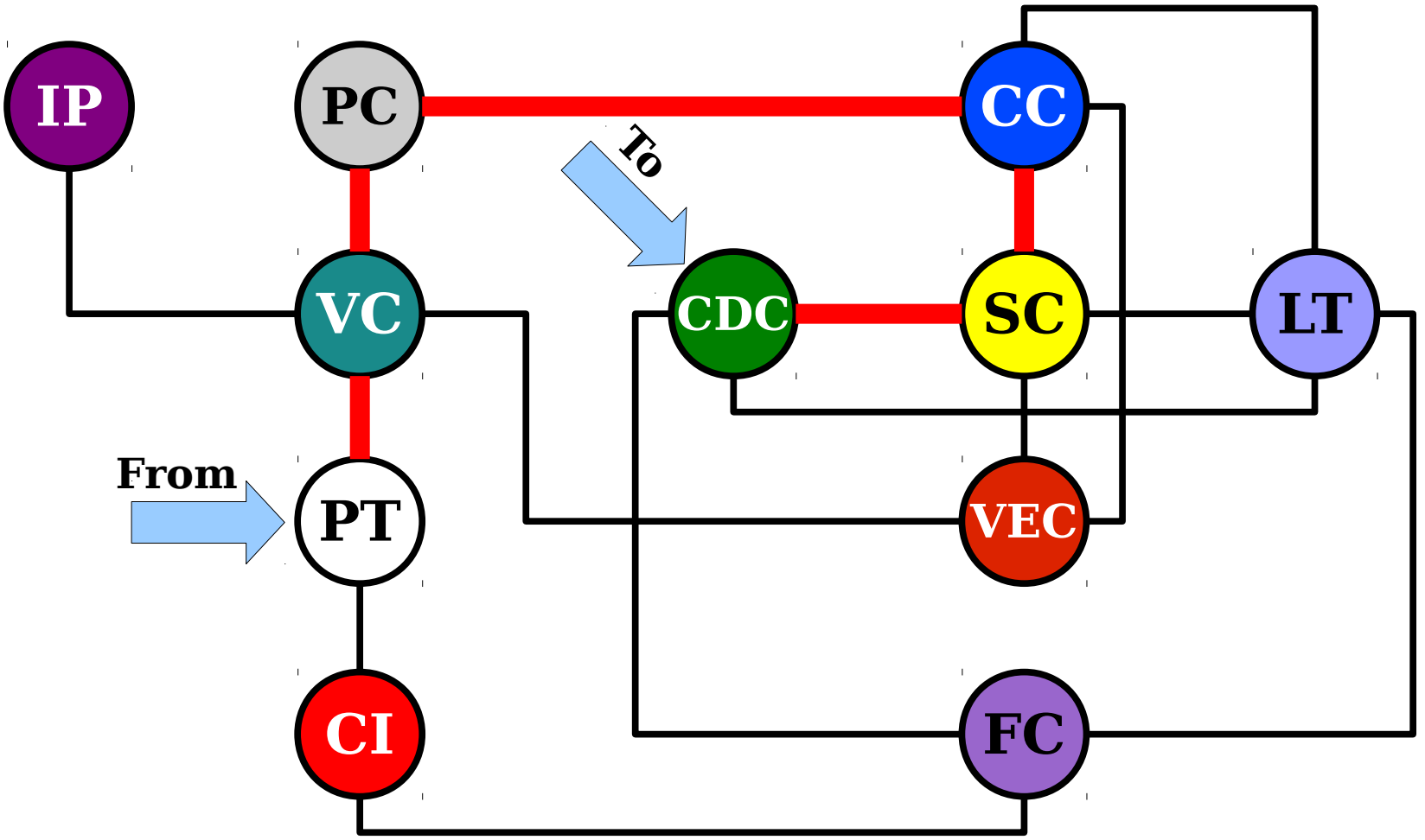
- An ***unordered pair*** is a set $\{a, b\}$ of two elements (remember that sets are unordered).
  - $\{0, 1\} = \{1, 0\}$
- An ***ordered pair*** (a, b) is a pair of elements in a specific order.
  - $(0, 1) \neq (1, 0)$.
  - Two ordered pairs are equal iff each of their components are equal.

Formalizing Graphs

- Formally, a ***graph*** is an ordered pair $G = (V, E)$, where
  - V is a set of nodes.
  - E is a set of edges.
- G is defined as an *ordered* pair so it's clear which set is the nodes and which is the edges.
- V can be any set whatsoever.
- E is one of two types of sets:
  - A set of *unordered* pairs of elements from V.
  - A set of *ordered* pairs of elements from V.

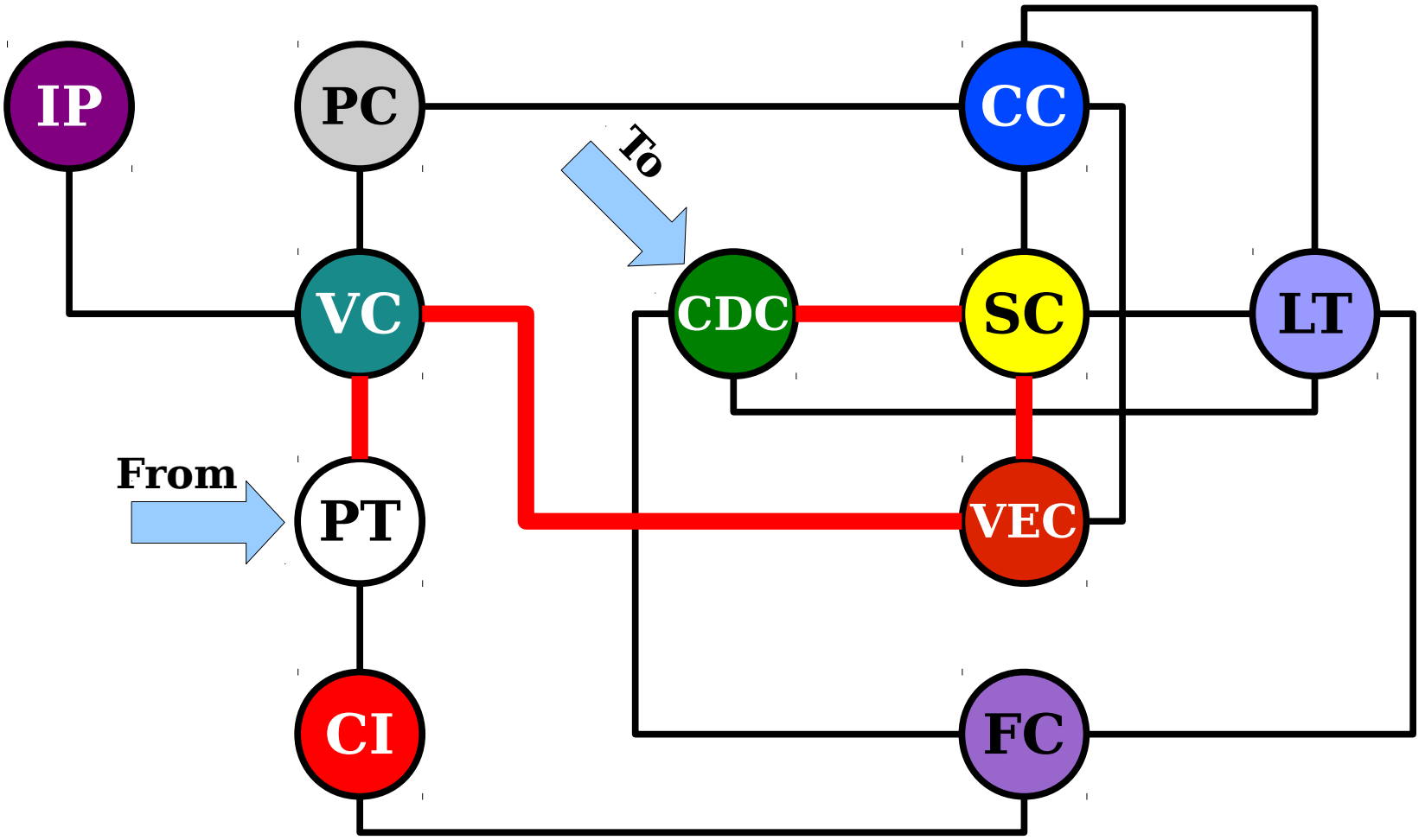
Undirected Connectivity

Navigating a Graph



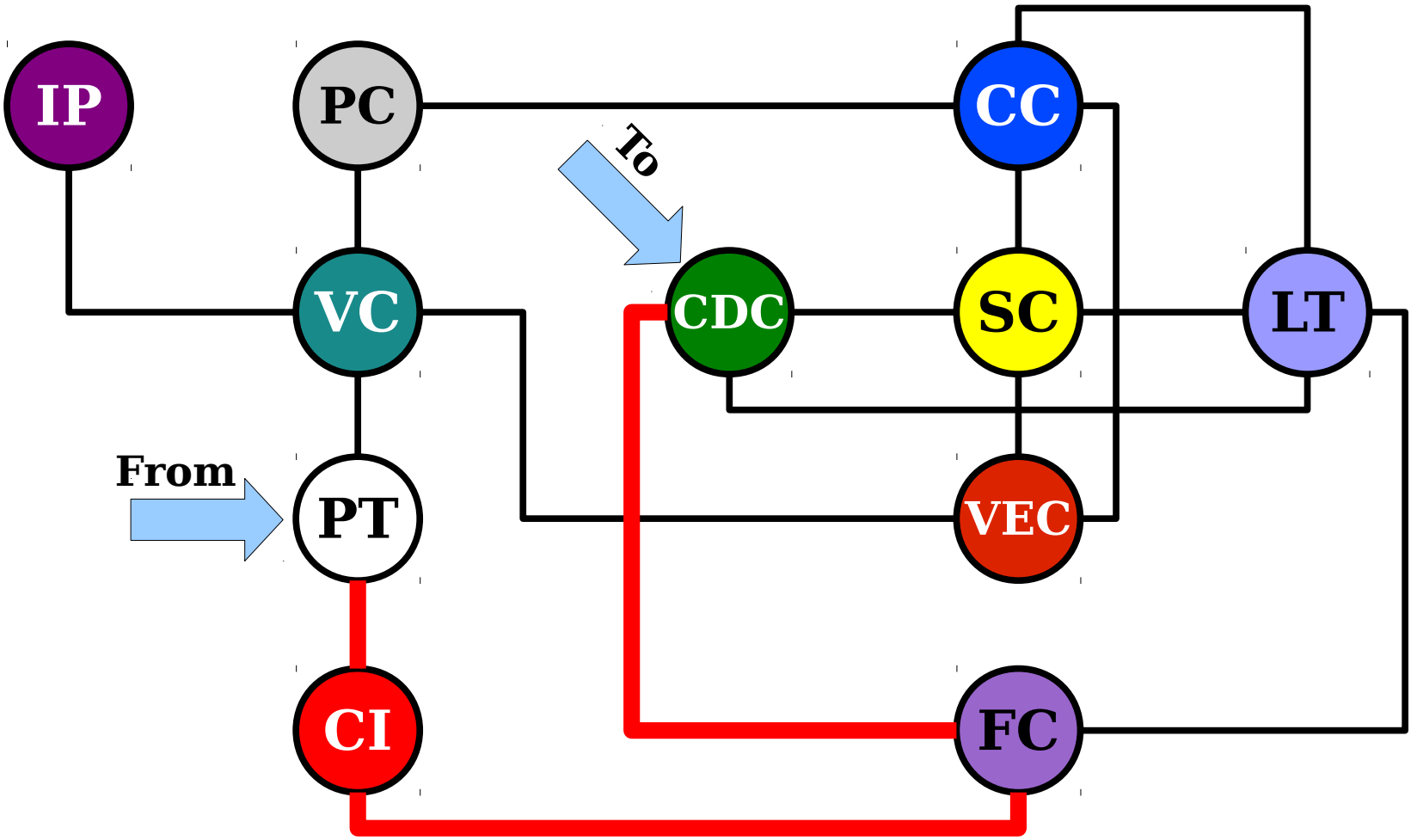
PT → VC → PC → CC → SC → CDC

Navigating a Graph



PT → VC → VEC → SC → CDC

Navigating a Graph



PT → CI → FC → CDC

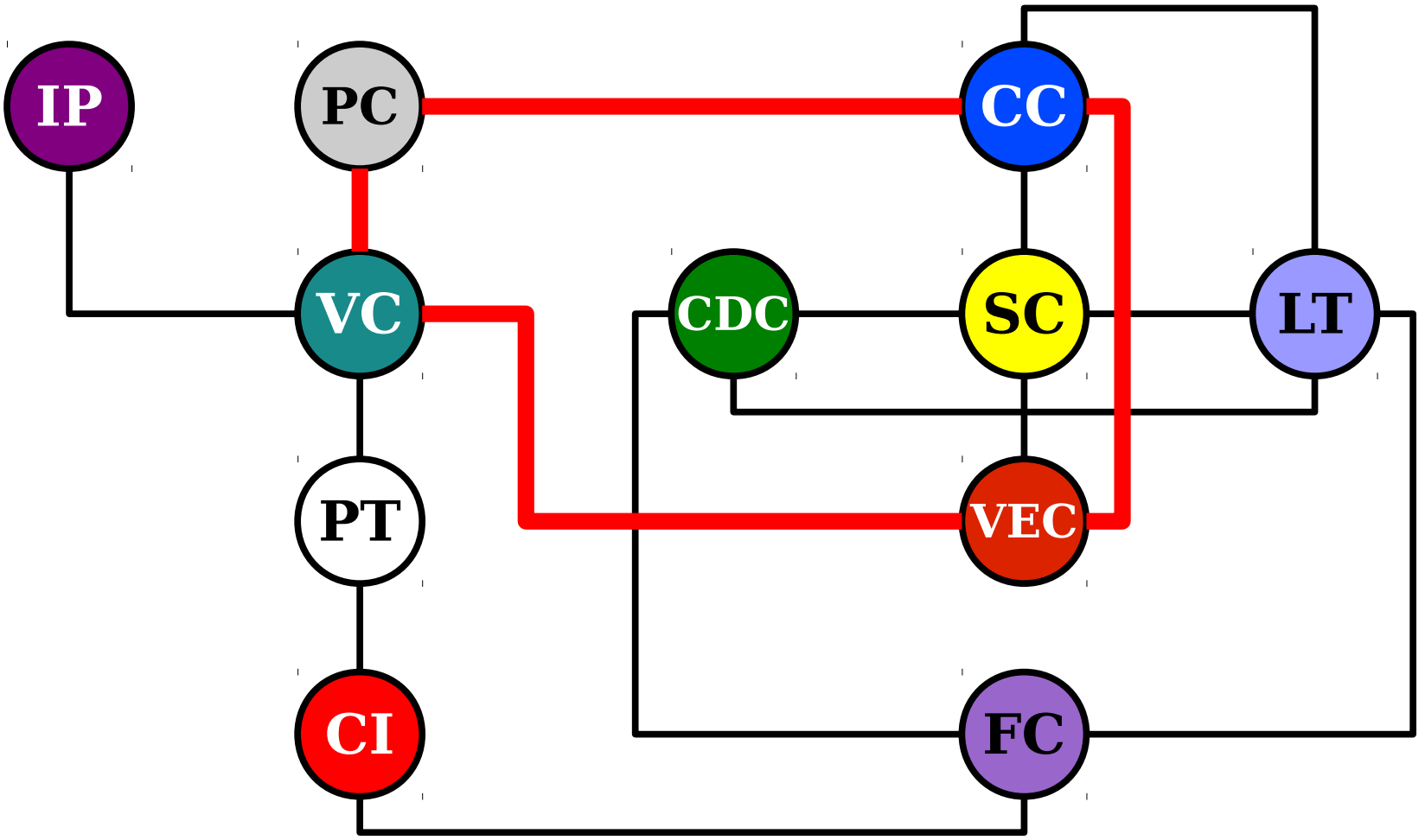
A ***path*** from v_1 to v_n is a sequence of nodes $v_1, v_2, \ldots, v_n$ where $(v_k, v_{k+1}) \in E$ for all natural numbers in the range $1 \leq k \leq n - 1$.

The ***length*** of a path is the number of edges it contains, which is one less than the number of nodes in the path.

A ***path*** from v_1 to v_n is a sequence of nodes v_1, v_2, …, v_n where $\{v_k, v_{k+1}\} \in E$ for all natural numbers in the range $1 \leq k \leq n - 1$.

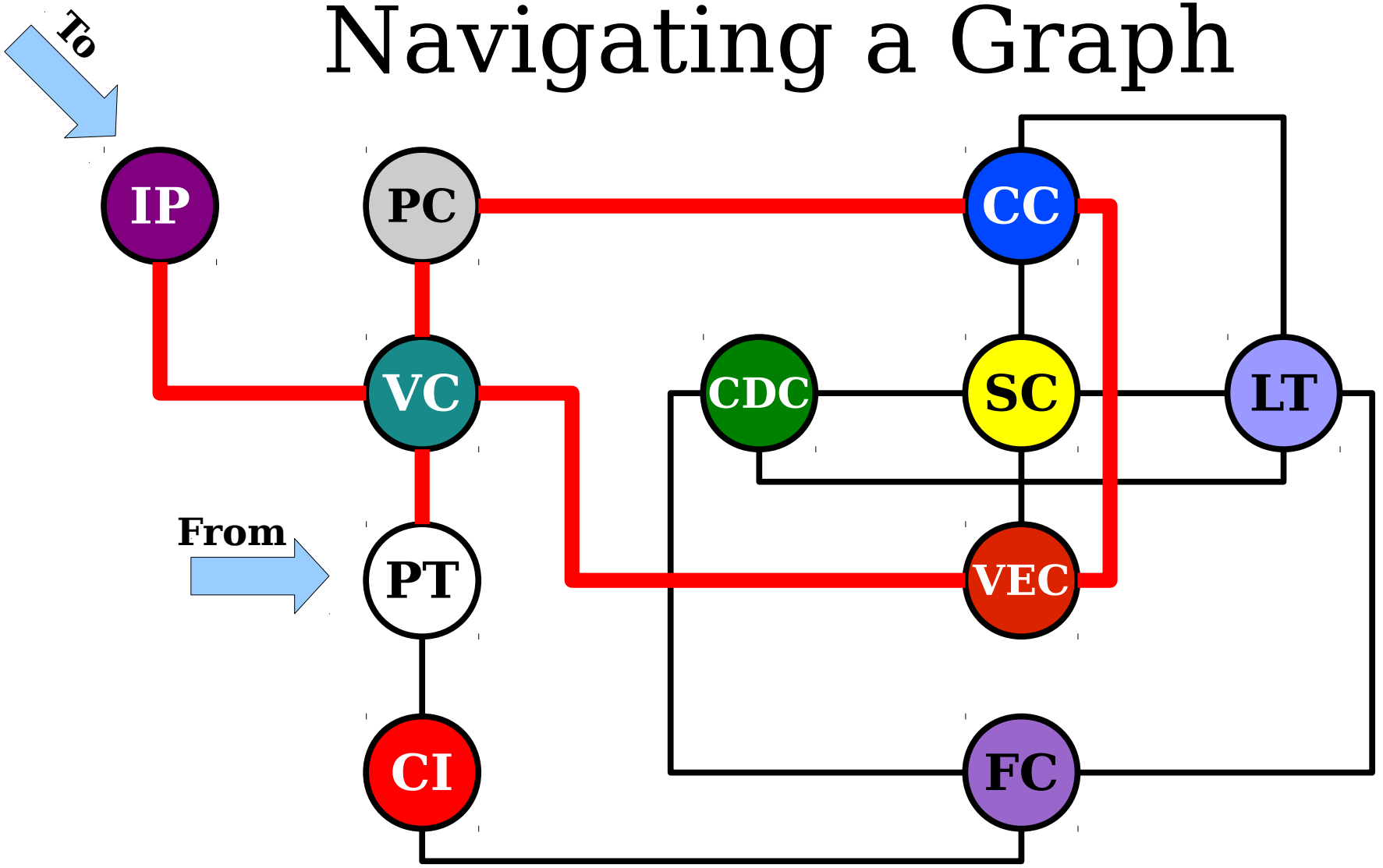
The ***length*** of a path is the number of edges it contains, which is one less than the number of nodes in the path.

Navigating a Graph



PC → CC → VEC → VC → PC

Navigating a Graph



PT → VC → PC → CC → VEC → VC → IP

A ***cycle*** in a graph is a path from a node to itself.

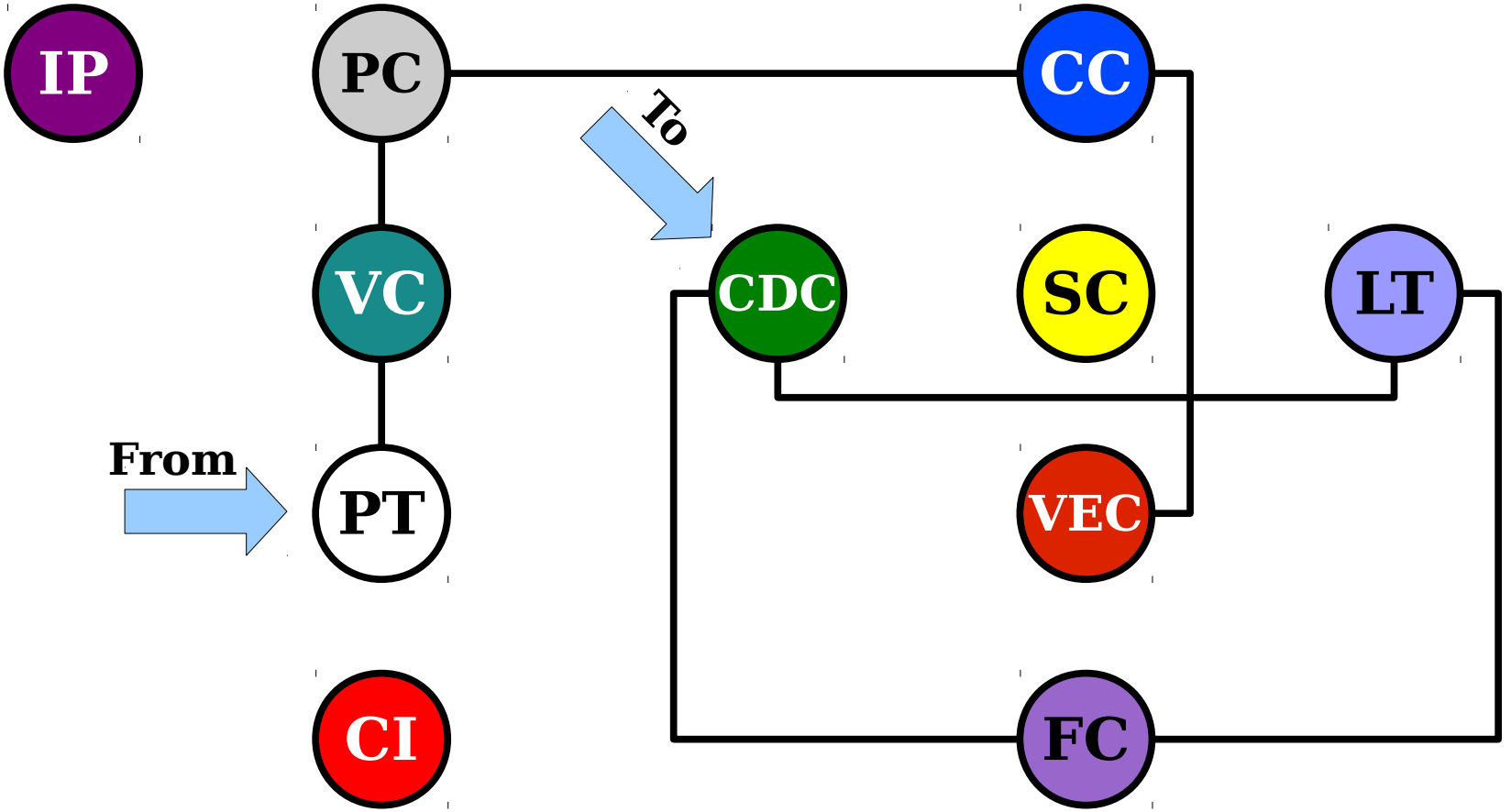
The ***length*** of a cycle is the number of edges in that cycle.

A ***simple path*** in a graph is a path that does not revisit any nodes or edges.

A ***simple cycle*** in a graph is a cycle that does not revisit any nodes or edges (except the start/end node).

Usually, the ***empty path*** starting and ending at a node and containing no edges is considered a simple path, but not a simple cycle.

Navigating a Graph



In an undirected graph, two nodes u and v are called ***connected*** if there is a path from u to v.

Properties of Connectivity

- ***Theorem:*** For any graph $G = (V, E)$, the following properties hold for the connectivity relation:

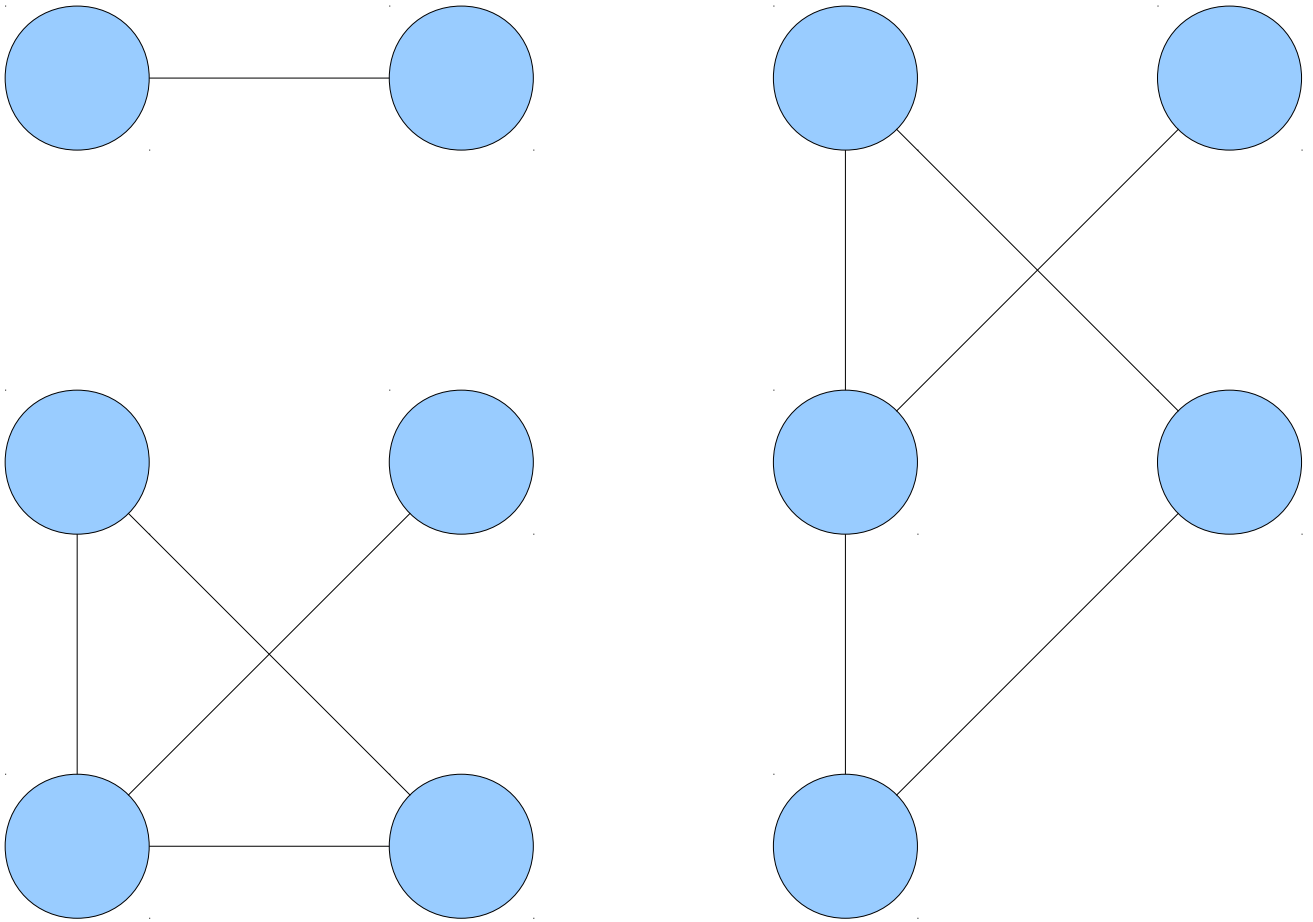
  $\forall v \in V.\ Connected(v, v)$

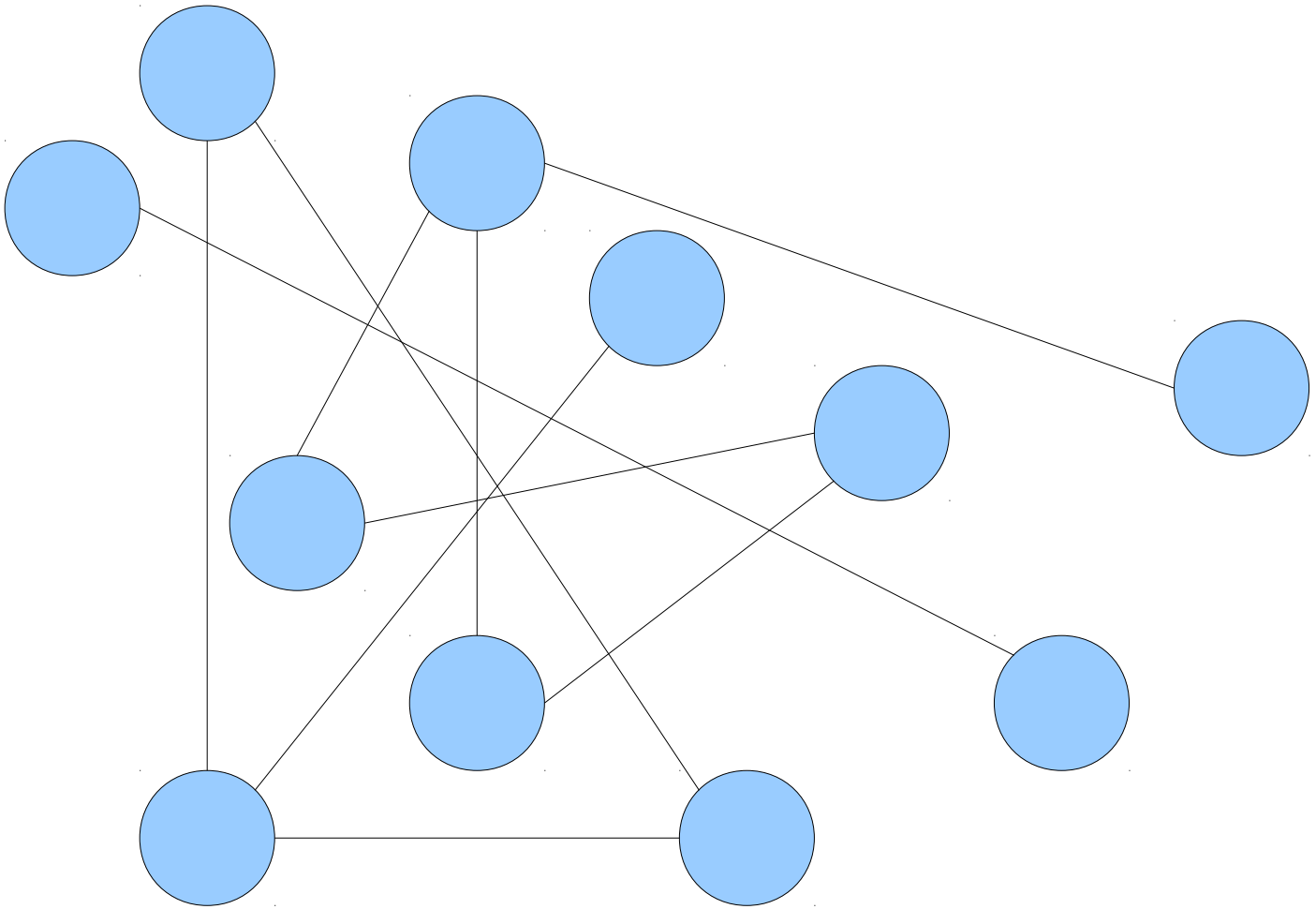
  $\forall u \in V.\ \forall v \in V.\ ($
  $\quad Connected(u, v) \to Connected(v, u)$
  $)$

  $\forall u \in V.\ \forall v \in V.\ \forall w \in V.\ ($
  $\quad Connected(u, v) \wedge Connected(v, w) \to$
  $\quad Connected(u, w)$
  $)$

- Can prove by thinking about the paths that are implied by each.

Connected Components





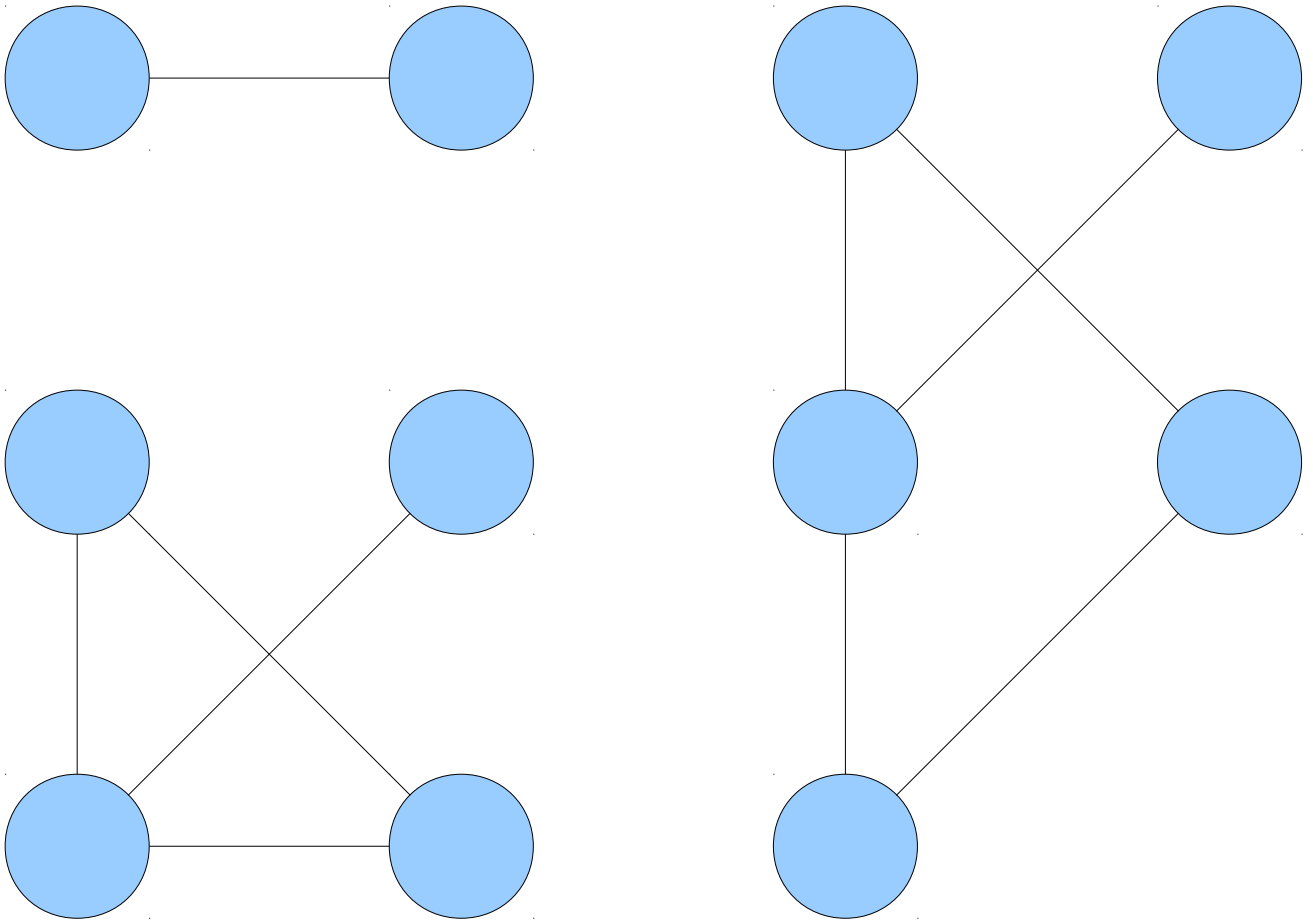
An Initial Definition

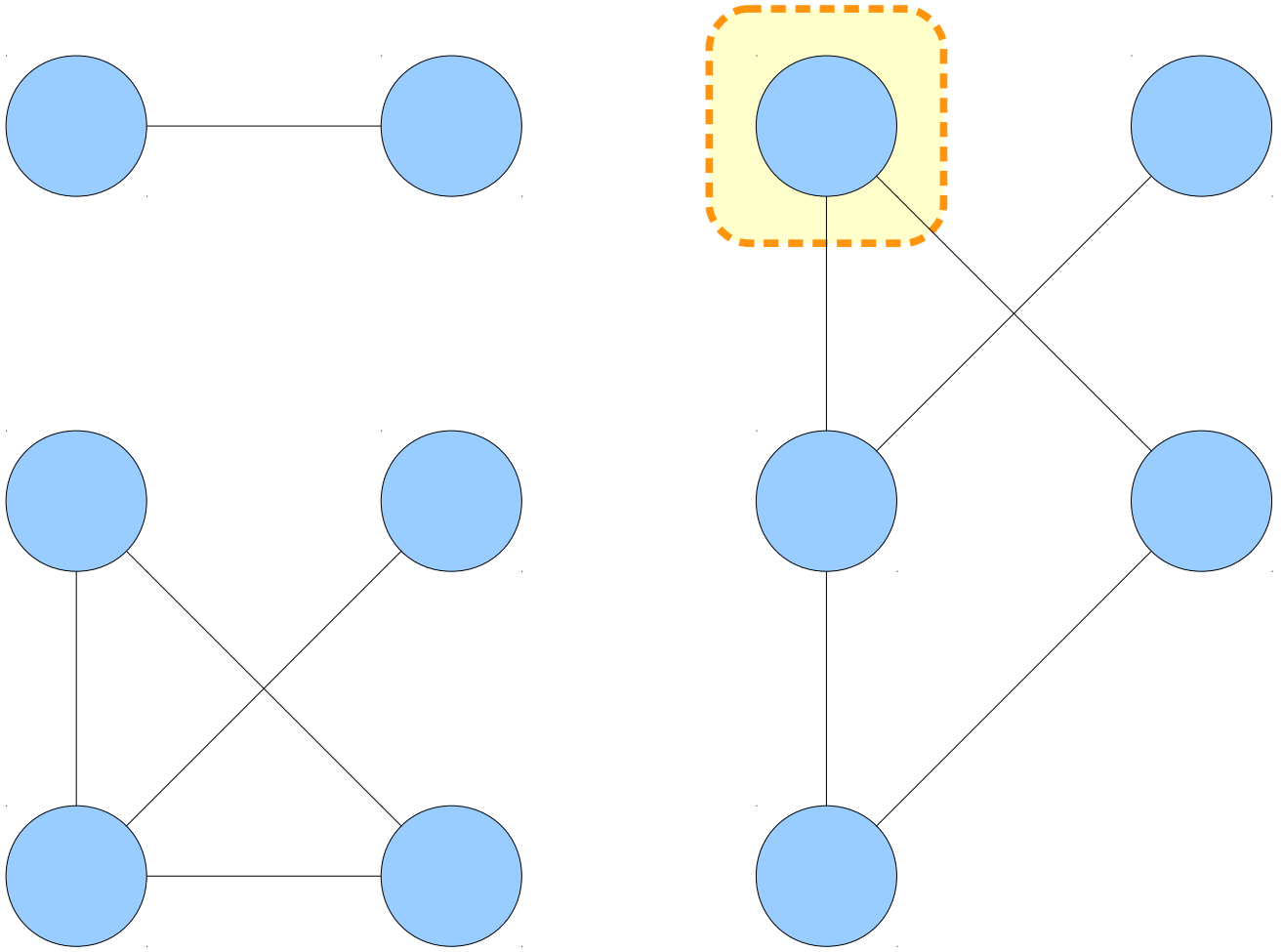
- **Attempted Definition #1:** A *piece* of an undirected graph $G = (V, E)$ is a set $C \subseteq V$ where

$$\forall u \in C.\ \forall v \in C.\ \mathit{Connected}(u, v)$$

- Intuition: a piece of a graph is a set of nodes that are all connected to one another.

⚠ *This definition has some problems; please don't use it as a reference.* ⚠

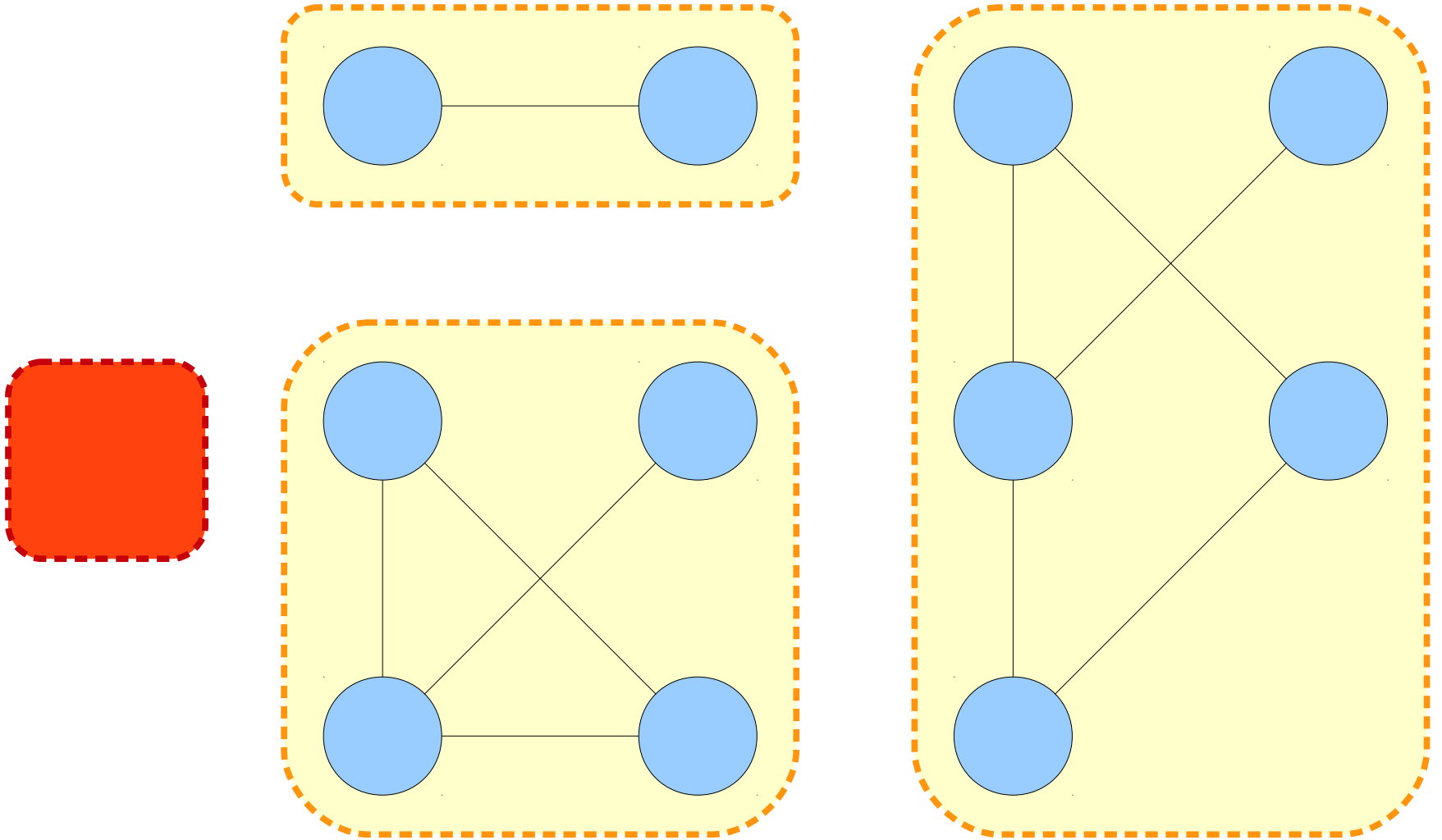




An Updated Definition

- **Attempted Definition #2:** A *piece* of an undirected graph $G = (V, E)$ is a set $C \subseteq V$ where
  - $\forall u \in C.\ \forall v \in C.\ Connected(u, v)$
  - $\forall u \in C.\ \forall v \in V - C.\ \neg Connected(u, v)$
- Intuition: a piece of a graph is a set of nodes that are all connected to one another that doesn't “miss” any nodes.

⚠ *This definition has some problems;* ⚠
please don't use it as a reference.



A Final Definition

- **Definition:** A ***connected component*** of an undirected graph $G = (V, E)$ is a *nonempty* set $C \subseteq V$ where
  - $\forall u \in C.\ \forall v \in C.\ Connected(u, v)$
  - $\forall u \in C.\ \forall v \in V - C.\ \neg Connected(u, v)$
- Intuition: a connected component is a nonempty set of nodes that are all connected to one another that includes as many nodes as possible.

Time-Out for Announcements!

Problem Set Three

- Problem Set Two was due at 12:50PM today.
- Problem Set Three goes out today.
  - Checkpoint problem due at the start of class on Monday.
  - Remaining problems due at the start of class on Friday.
- Explore propositional logic, first-order logic, and graph theory!
- ***Start this problem set early!***

Understanding Cuba:

Social, Economic and Technological Landscape (1 unit)

Gain the knowledge you need to implement a possible tech project in Cuba through Interdisciplinary Exchange of the Americas (IdEA), a new Stanford Organization affiliated with Haas Center for Public Service.

Join us for the second class! Syllabus at tinyurl.com/idea-syllabus

Date: Friday, April 17, 2015 @ 4 - 5:30pm
Location: Old Union Rm. 215

Your Questions

“When picking classes, applying to internships, etc., I often find myself torn between trying something completely new or deepening my expertise in some topic/field with which I'm already familiar. As a student, should I prioritize breadth or depth?”

My advice would be to think about things in the long term. You want enough breadth to have a sense of what's out there, but at some point you need to commit to something.

“What track did you choose in the CS major when you were at Stanford? Can you tell us a little about each one/how to choose one?”

“What goes on in the different CS fields? How can a potential CS major or beginning programmer learn about all those specializations without taking a class in each one? The intro series deals with Systems a lot, but is that a big part of CS overall?”

I don't think I can fully answer these questions in the time we have, but I can try!

Back to CS103!

Manipulating our Definition

Proving the Obvious

- ***Theorem:*** If $G = (V, E)$ is a graph, then every node $v \in V$ belongs to exactly one connected component.
- How exactly would we prove a statement like this one?
- Use an ***existence and uniqueness proof***:
  - Prove there is *at least* one object of that type.
  - Prove there is *at most* one object of that type.
- These are usually separate proofs.

Proving the Obvious

- There are two parts to this theorem, but due to time constraints we'll just prove this one:

Every node belongs to at least one connected component.

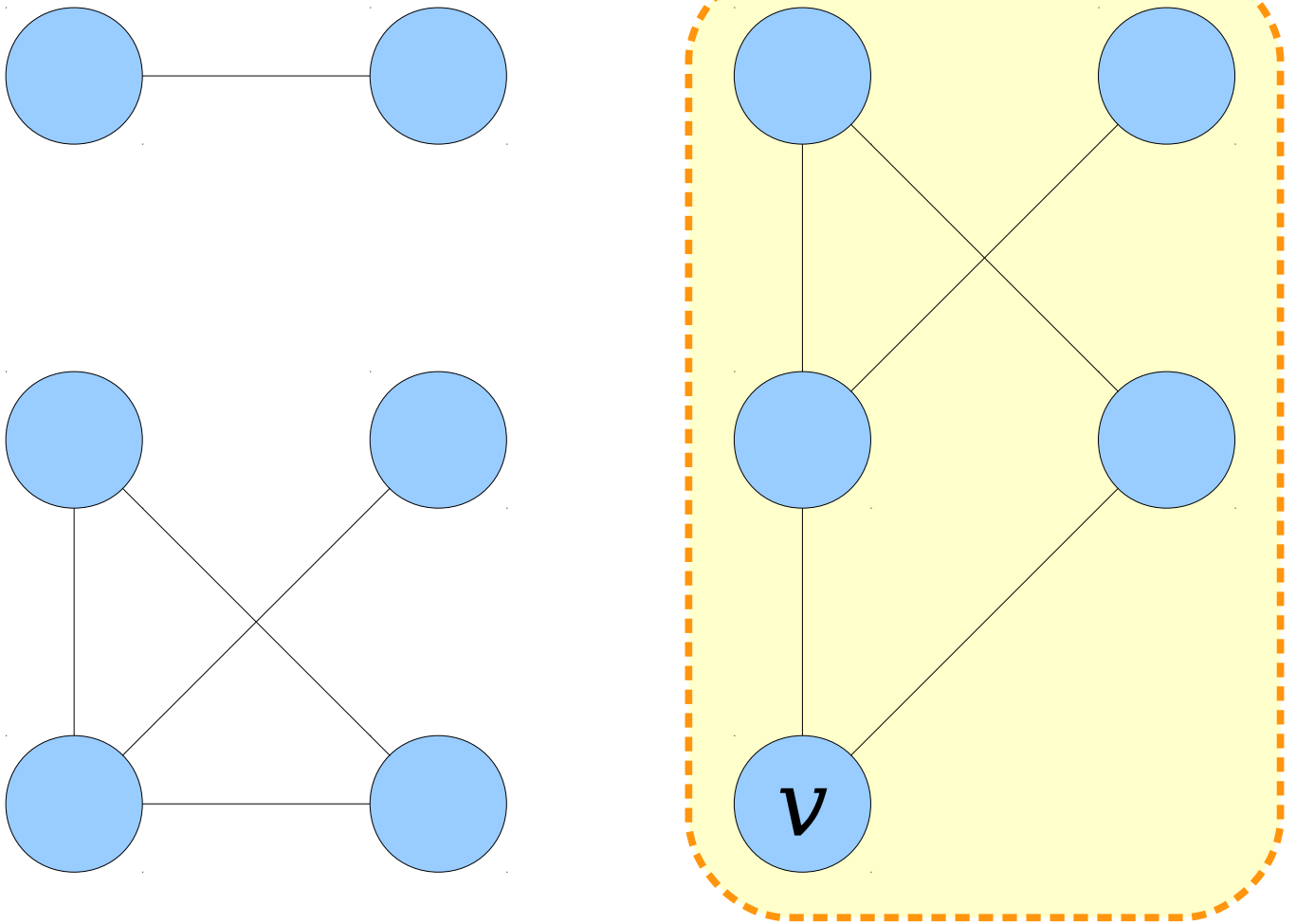
- This is trickier to prove than it might initially seem. We need to make sure to adhere to the definition we've set up for ourselves.

Proving Existence

- Given an arbitrary graph $G = (V, E)$ and an arbitrary node $v \in V$, we need to show that there exists some connected component C where $v \in C$.
- The key part of this is the existential statement

  There exists a connected component C such that $v \in C$.

- The challenge: how can we find the connected component that v belongs to given that v is an arbitrary node in an arbitrary graph?


v

The Conjecture

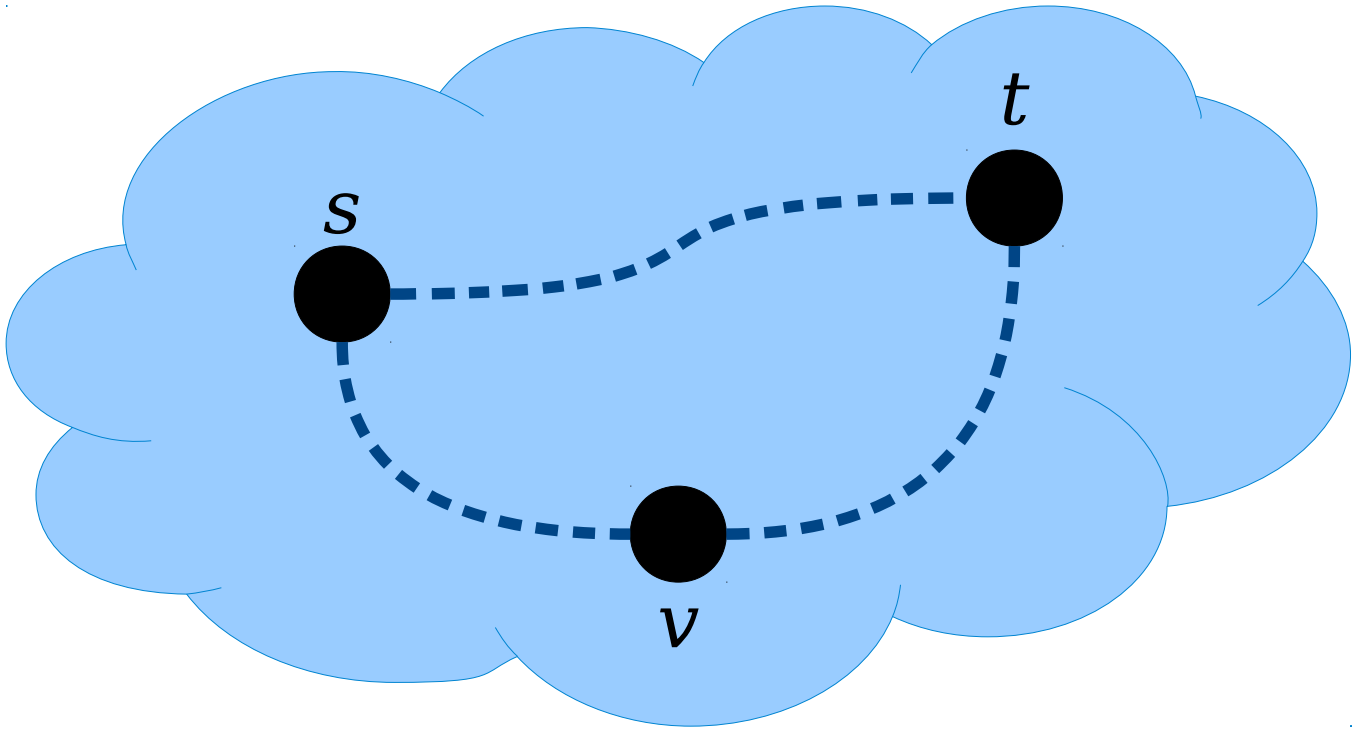
- **Conjecture:** Let $G = (V, E)$ be an undirected graph. Then for any node $v \in V$, the set $\{ x \in V \mid v \text{ is connected to } x \text{ in } G \}$ is a connected component and it contains v.
- If we can prove this, we have shown *existence*: at least one connected component contains v.

The Tricky Part

- We need to show for any $v \in V$ that the set $C = \{ x \in V \mid v \text{ is connected to } x \text{ in } G \}$ is a connected component and that it contains v.
- Therefore, we need to show
  - $C \subseteq V$,
  - $v \in C$,
  - $C \neq \emptyset$,
  - $\forall s \in C.\ \forall t \in C.\ \mathit{Connected}(s, t)$, and
  - $\forall s \in C.\ \forall t \in V - C.\ \neg \mathit{Connected}(s, t)$.

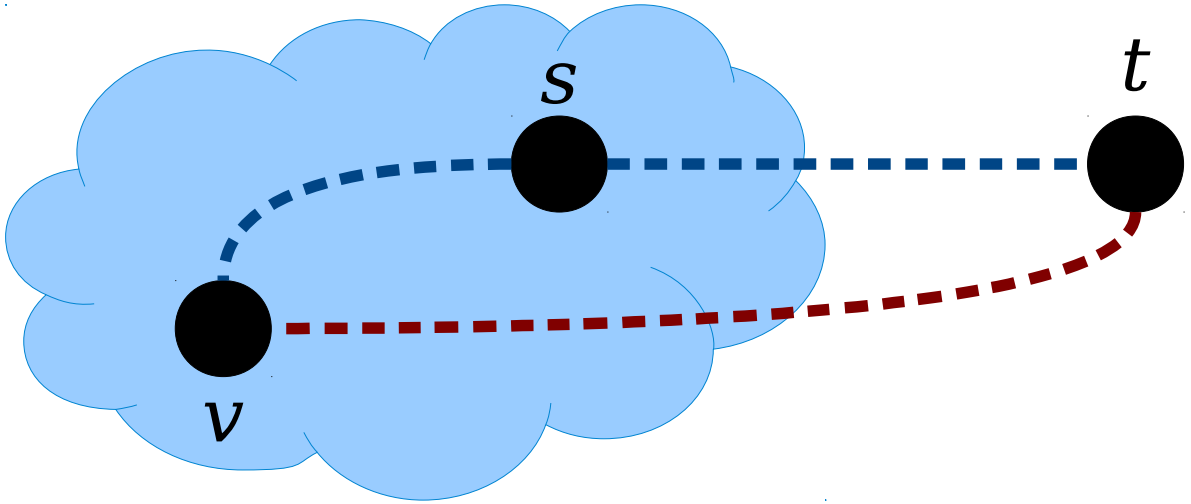
Lemma 1: Let $G = (V, E)$ be an undirected graph. For any node $v \in V$, the set $C = \{ x \in V \mid v \text{ is connected to } x \text{ in } G \}$ contains v.

Proof: By our earlier theorem, we know that any node v in a graph G is connected to itself. Therefore, by definition of C, we see that $v \in C$. ■



Lemma 2: Let $G = (V, E)$ be an undirected graph and $v \in V$ be an arbitrary node in G. Let $C = \{ x \in V \mid v \text{ and } x \text{ are connected in } G \}$. Then for any $s, t \in C$, the nodes s and t are connected in G.

Proof: Choose any arbitrary $s, t \in C$. By definition of C, we know that v and s are connected and that v and t are connected in G. From our earlier theorem, since v and s are connected in G, we know that s and v are connected in G. By that same theorem, since s and v are connected in G and v and t are connected in G, we know that s and t are connected in G, which is what we needed to show. ■



Lemma 2: Let $G = (V, E)$ be an undirected graph and $v \in V$ be an arbitrary node in G. Let $C = \{ x \in V \mid v \text{ and } x \text{ are connected in } G \}$. Then for any $s \in C$ and $t \in V - C$, the nodes s and t are not connected in G.

Proof: Consider any $s \in C$ and $t \in V - C$. Assume for the sake of contradiction that s is connected to t in G. Since $s \in C$, we know that v and s are connected in G. Also, since $t \in V - C$, we know that v and t are not connected in G. Since v and s are connected in G and s and t are connected in G, by our earlier theorem we know that v and t are connected in G. This contradicts our initial assumption that v and t are not connected in G.

We've reached a contradiction, so our assumption must have been wrong. Therefore, if $s \in C$ and $t \in V - C$, we see that s and t are not connected in G. ■

Theorem: Let $G = (V, E)$ be an undirected graph. Then every node $v \in V$ belongs to some connected component of G.

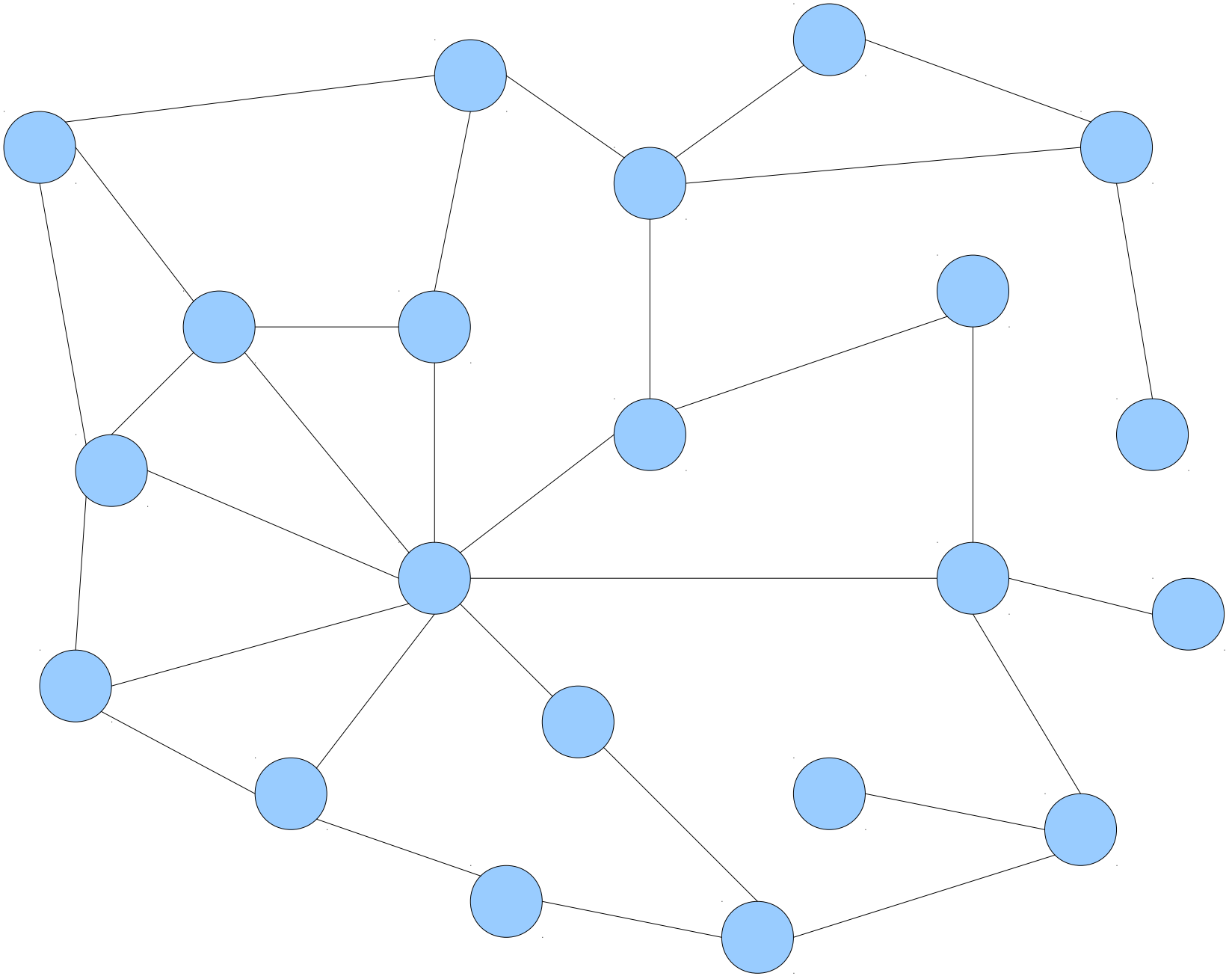
Proof: Take any $v \in V$ and let $C = \{ x \in V \mid v \text{ and } x \text{ are connected in } G \}$. We need to show that $C \subseteq V$, that $C \neq \emptyset$, that any two nodes in C are connected, and that no node in C is connected to any node outside of C.

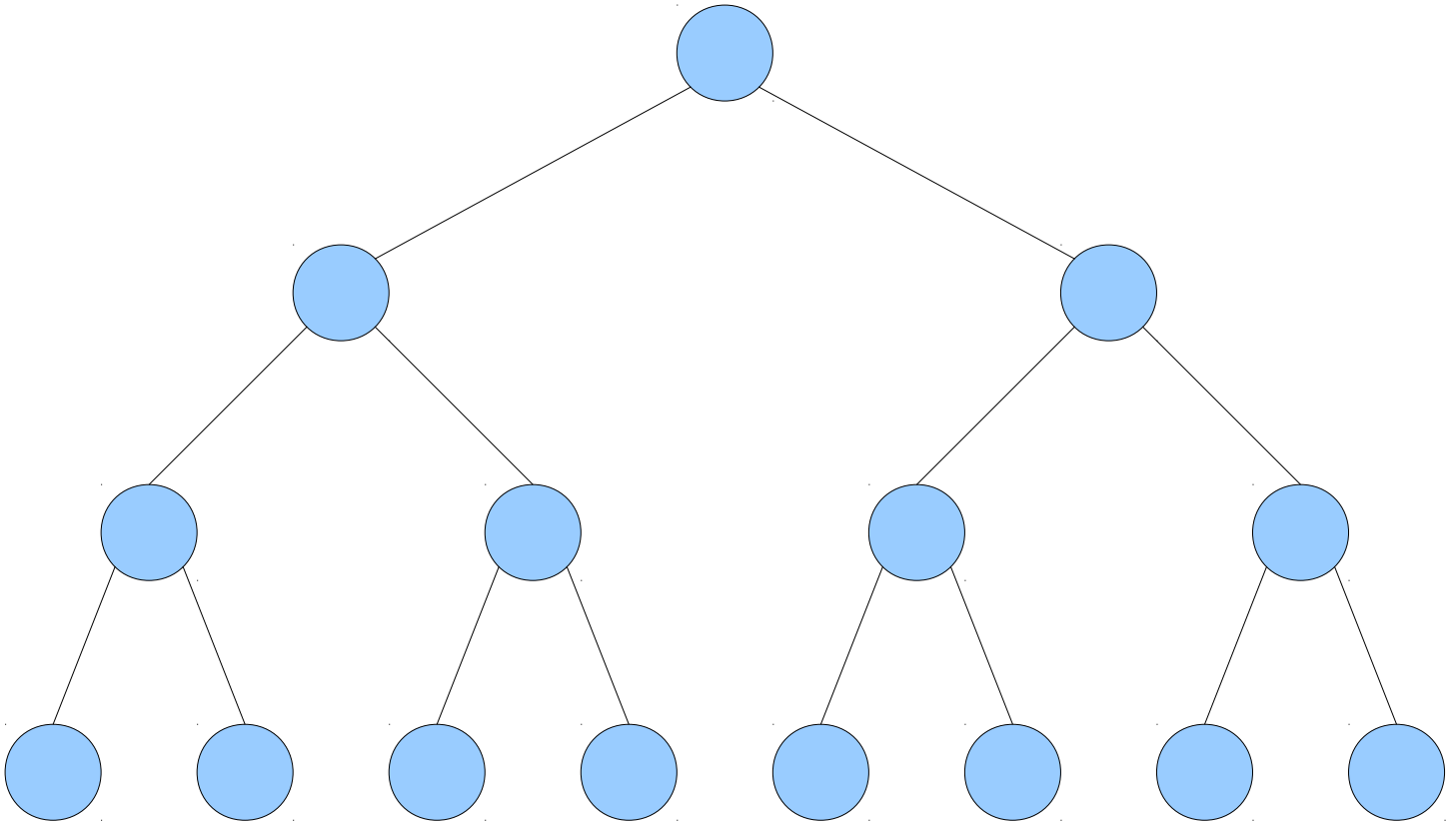
First, note that every element of C is a node in V, so $C \subseteq V$. Next, note by Lemma 1 that $v \in C$, meaning that $C \neq \emptyset$. From Lemma 2, we know that any two nodes in C are connected, and from lemma 3 we know that no node in C is connected to any node outside of C. Therefore, by definition, C is a connected component. Since we also know that $v \in C$, we've shown that there is at least one connected component containing v, as required. ■

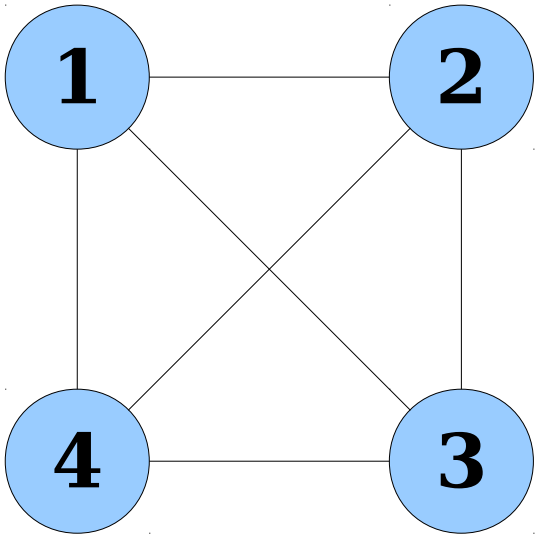
Why All This Matters

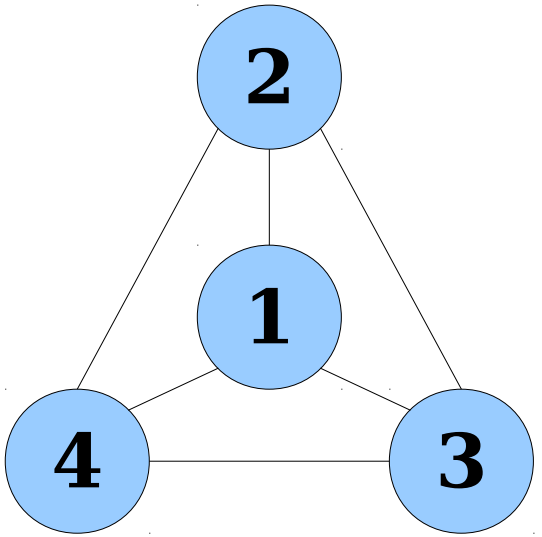
- I chose the example of connected components to
  - describe how to come up with a precise definition for intuitive terms;
  - see how to manipulate a definition once we've come up with one; and
  - explore multipart proofs with several different lemmas.

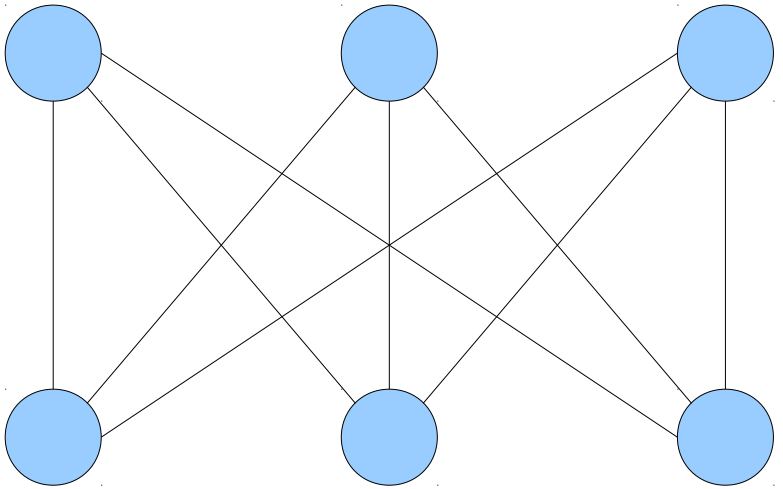
Planar Graphs





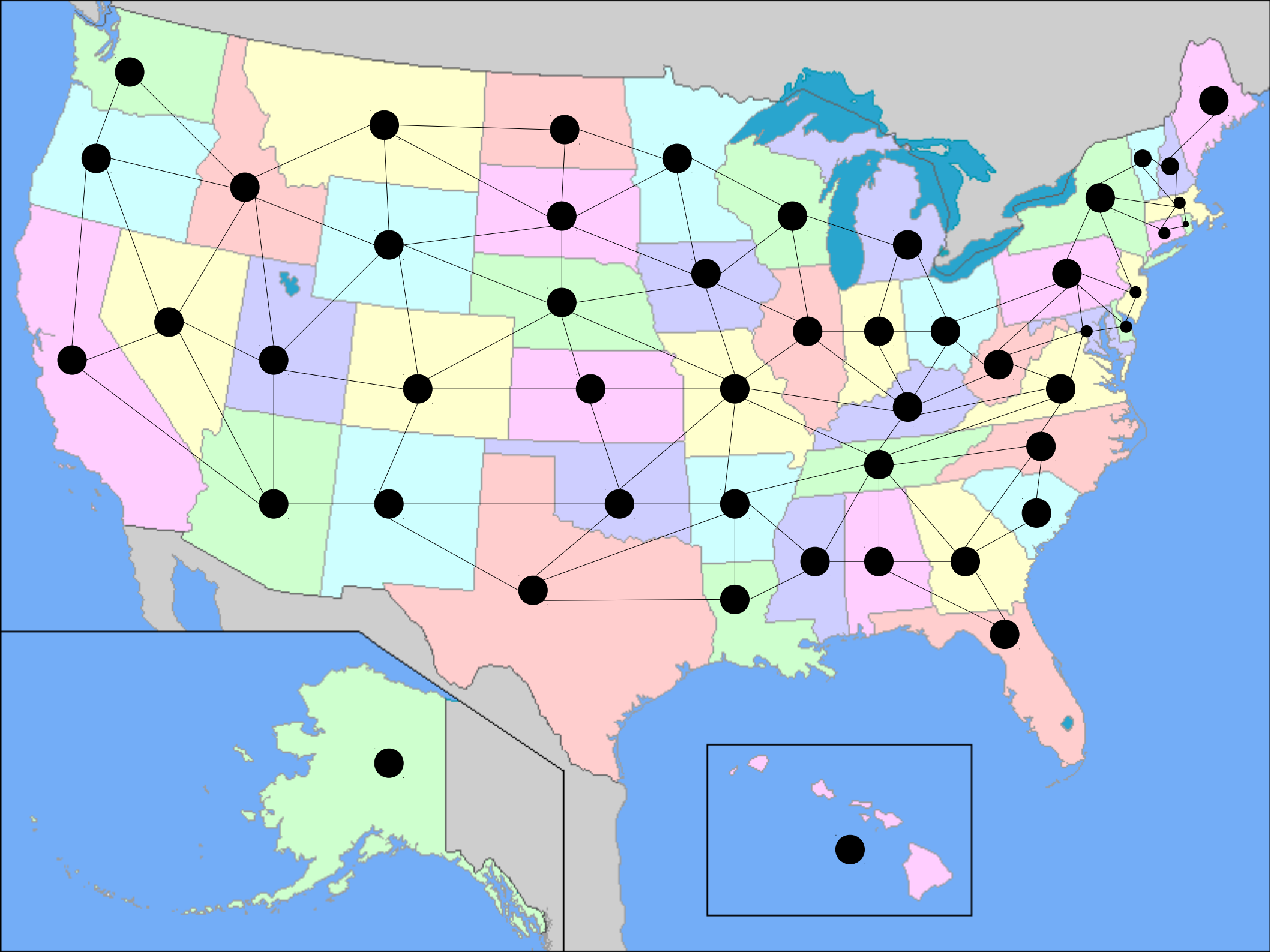

1
2
4
3

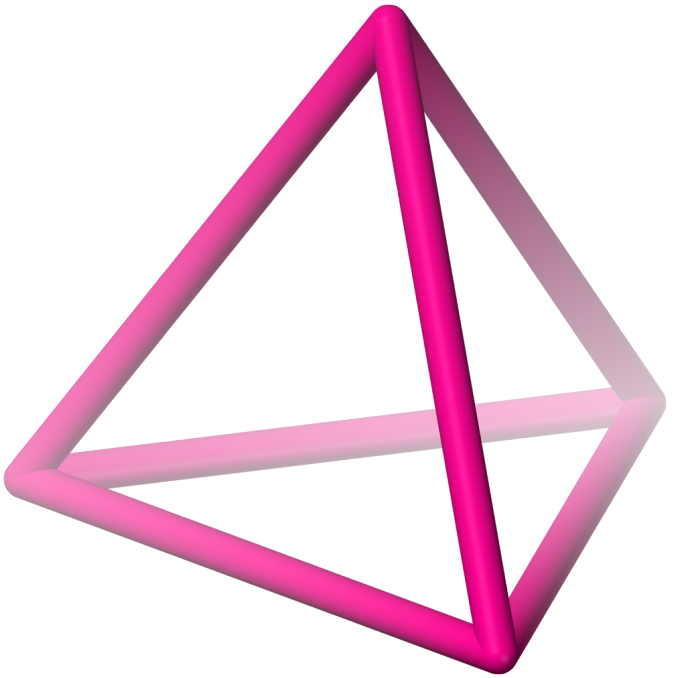

2
1
4
3

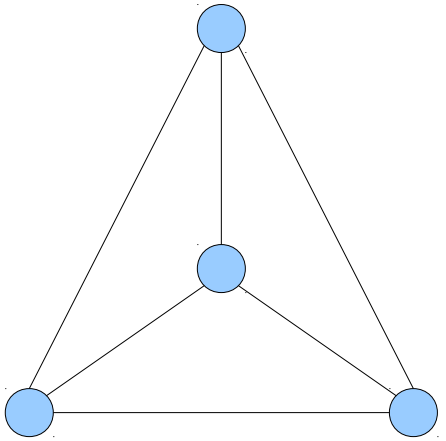


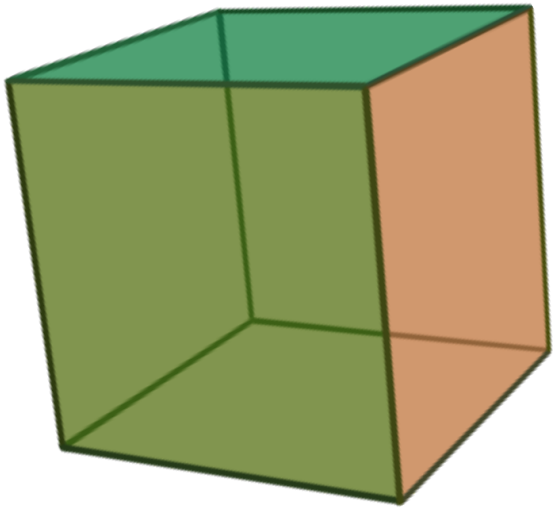
This graph is called the ***utility graph***. There is no way to draw it in the plane without edges crossing.

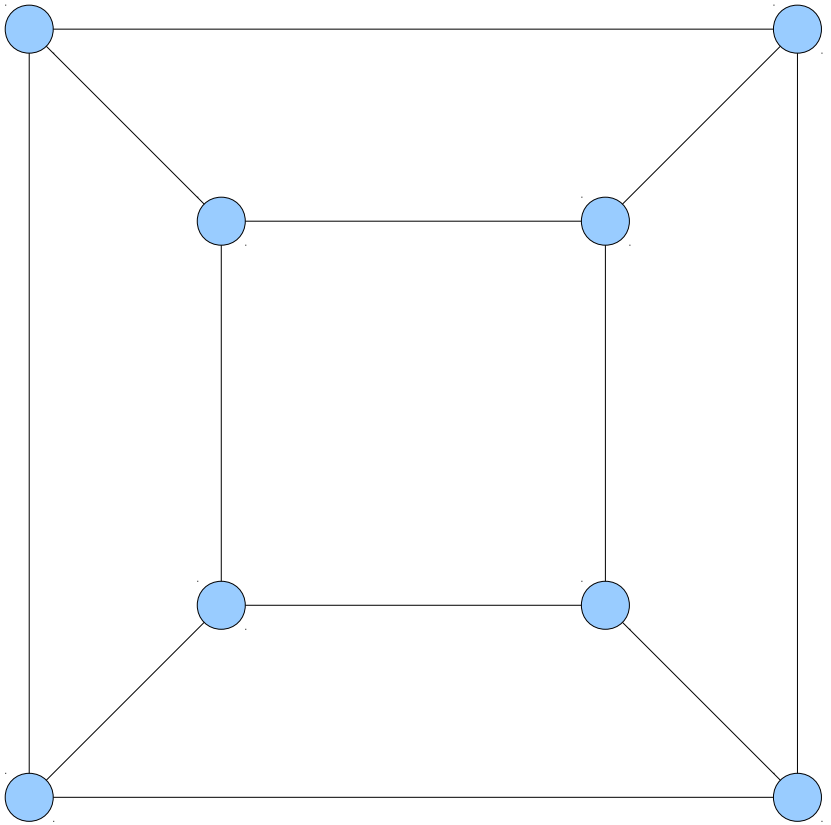
A graph is called a ***planar graph*** if there is some way to draw it in a 2D plane without any of the edges crossing.

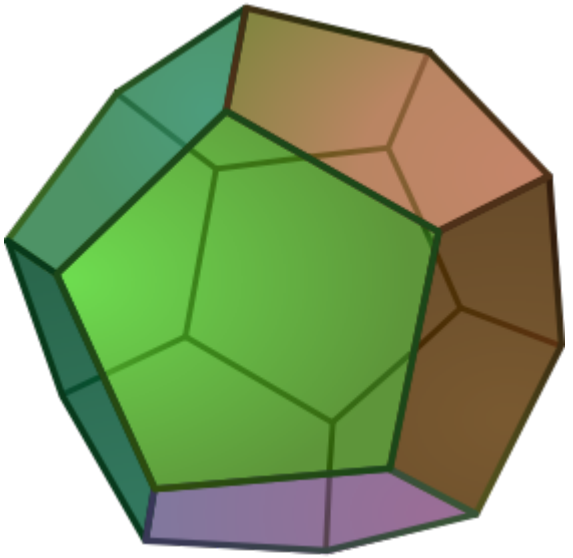


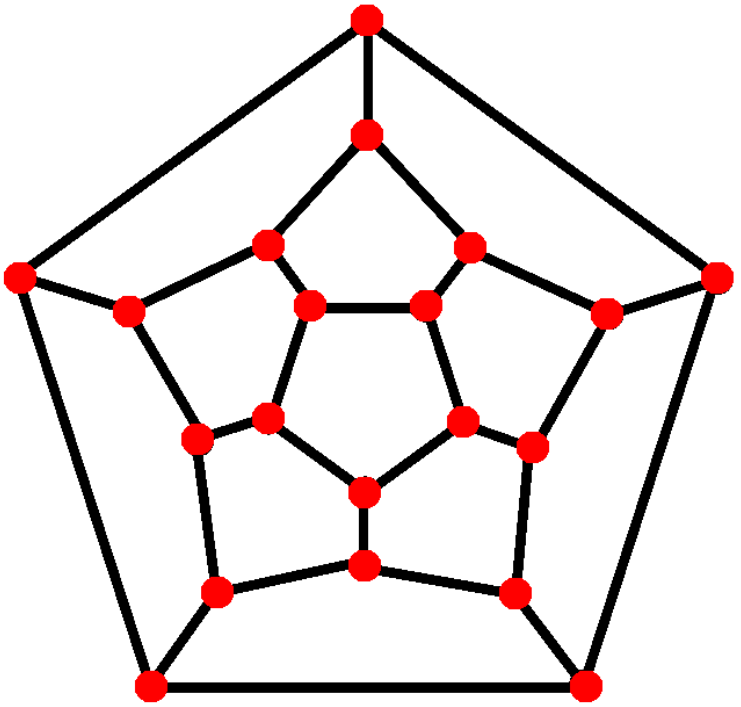




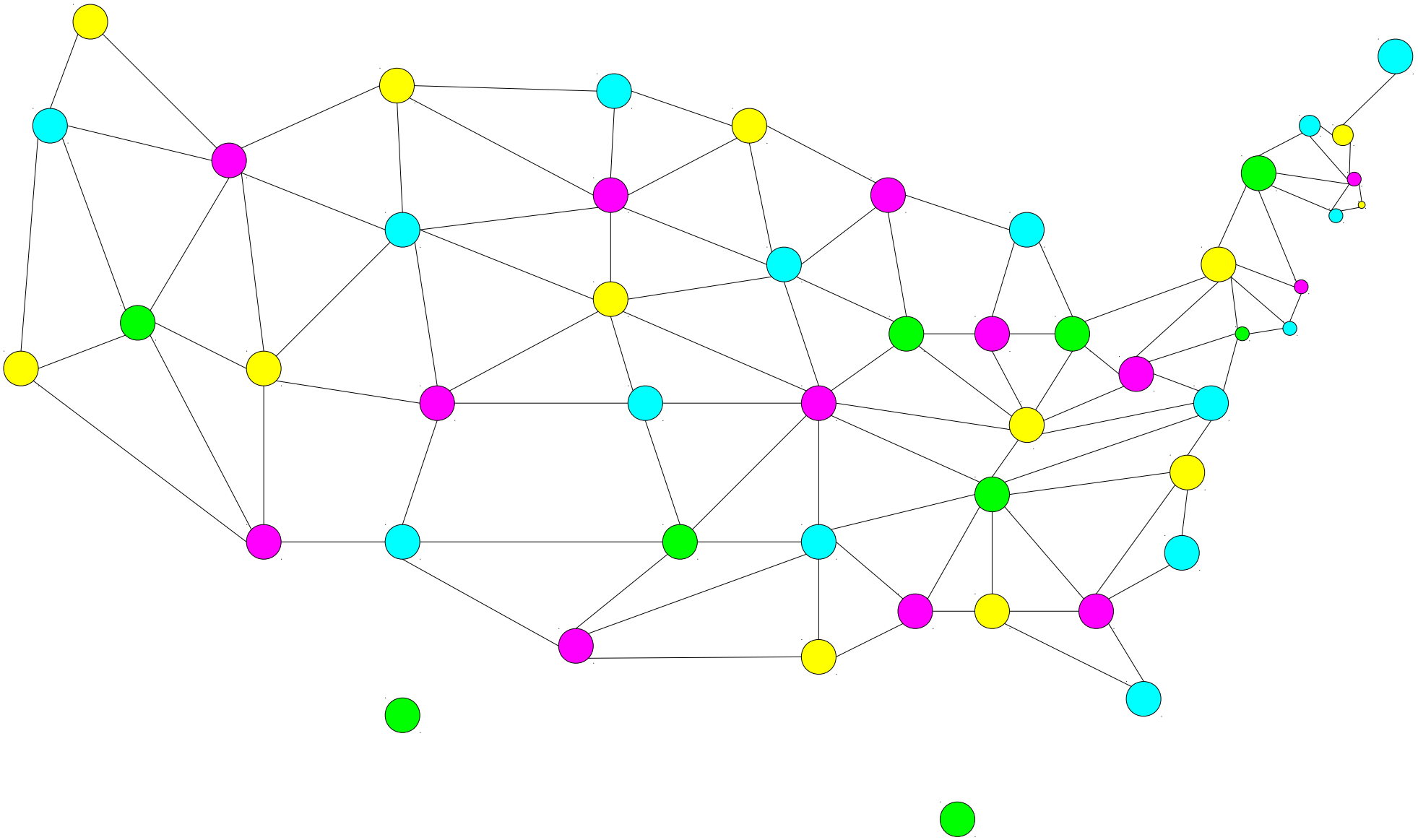




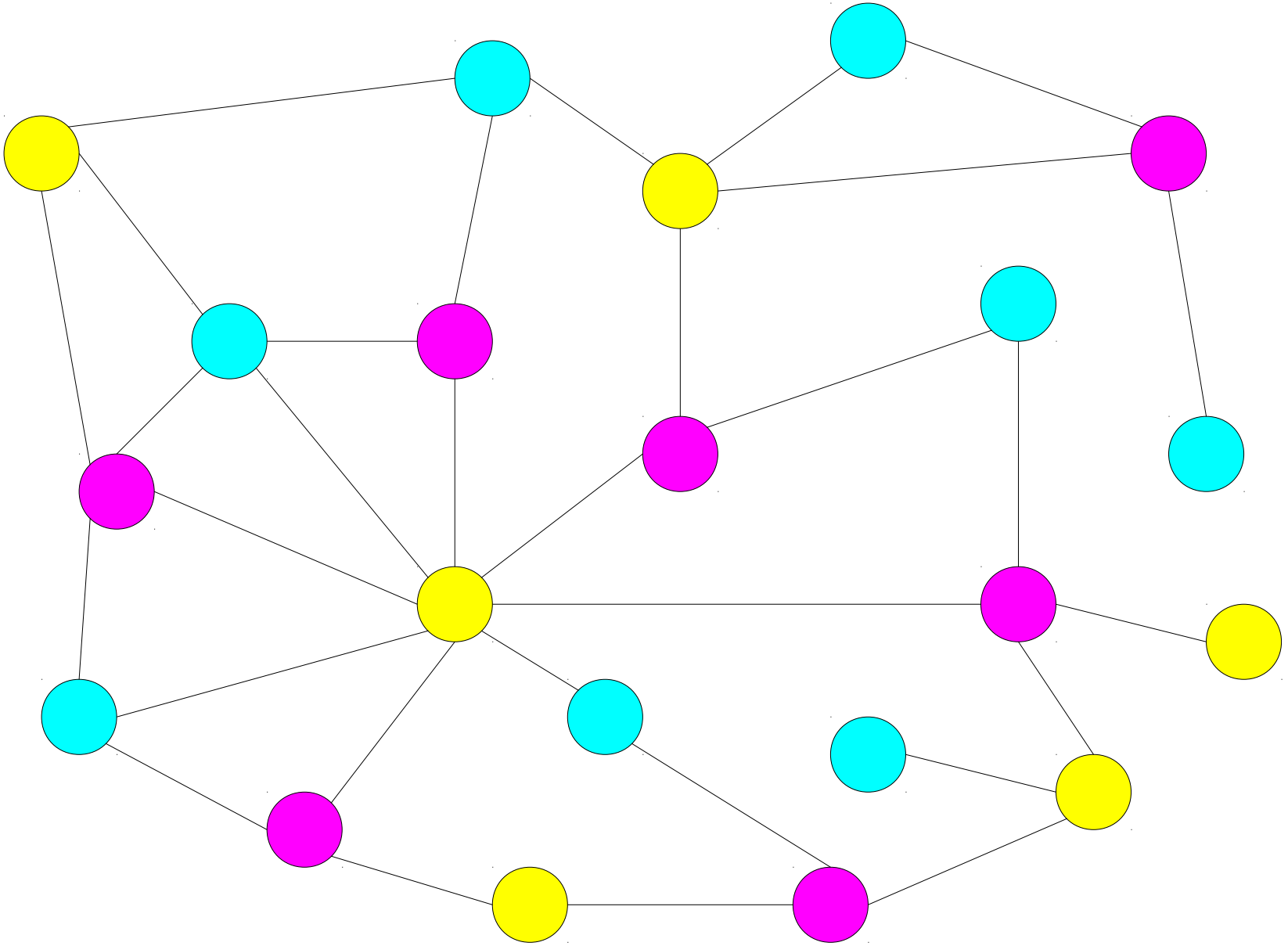




Graph Coloring



Graph Coloring



Graph Coloring

- An undirected graph $G = (V, E)$ with no self-loops (edges from a node to itself) is called ***k-colorable*** if the nodes in V can be assigned one of k different colors such that no two nodes of the same color are joined by an edge.
- The minimum number of colors needed to color a graph is called that graph's ***chromatic number***.
  - The chromatic number of a graph G is usually denoted $\boldsymbol{\chi(G)}$, from the Greek $\chi\rho\acute{\omega}\mu\alpha$ (“color”).

Theorem (Four-Color Theorem): Every planar graph is 4-colorable.

- **1850s:** Four-Color Conjecture posed.
- **1879:** Kempe proves the Four-Color Theorem.
- **1890:** Heawood finds a flaw in Kempe's proof.
- **1976:** Appel and Haken design a computer program that proves the Four-Color Theorem. The program checked 1,936 specific cases that are “minimal counterexamples;” any counterexample to the theorem must contain one of the 1,936 specific cases.
- **1980s:** Doubts rise about the validity of the proof due to errors in the software.
- **1989:** Appel and Haken revise their proof and show it is indeed correct. They publish a book including a 400-page appendix of all the cases to check.
- **1996:** Roberts, Sanders, Seymour, and Thomas reduce the number of cases to check down to 633.
- **2005:** Werner and Gonthier repeat the proof using an established automatic theorem prover (Coq), improving confidence in the truth of the theorem.