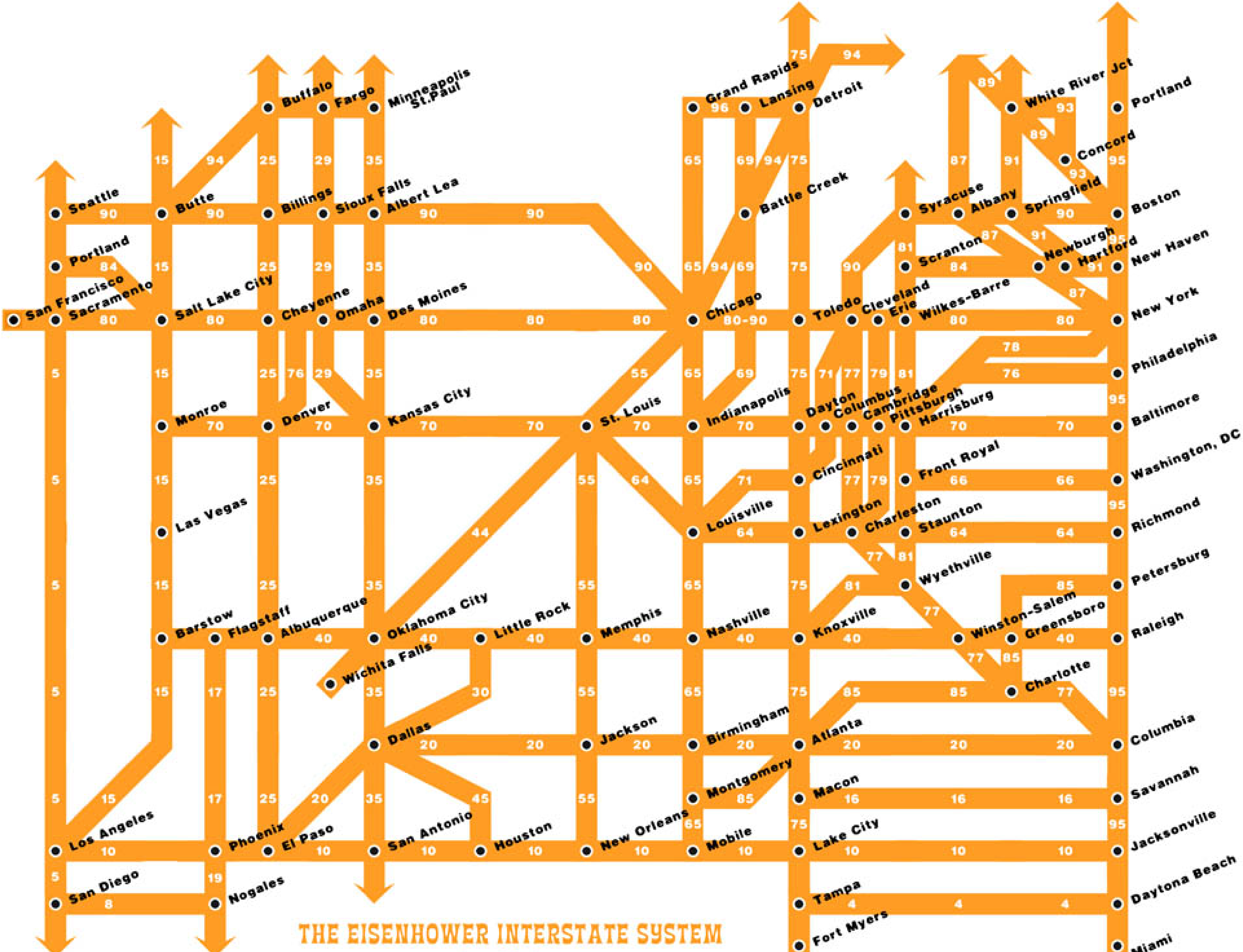
Graph Theory

Part One

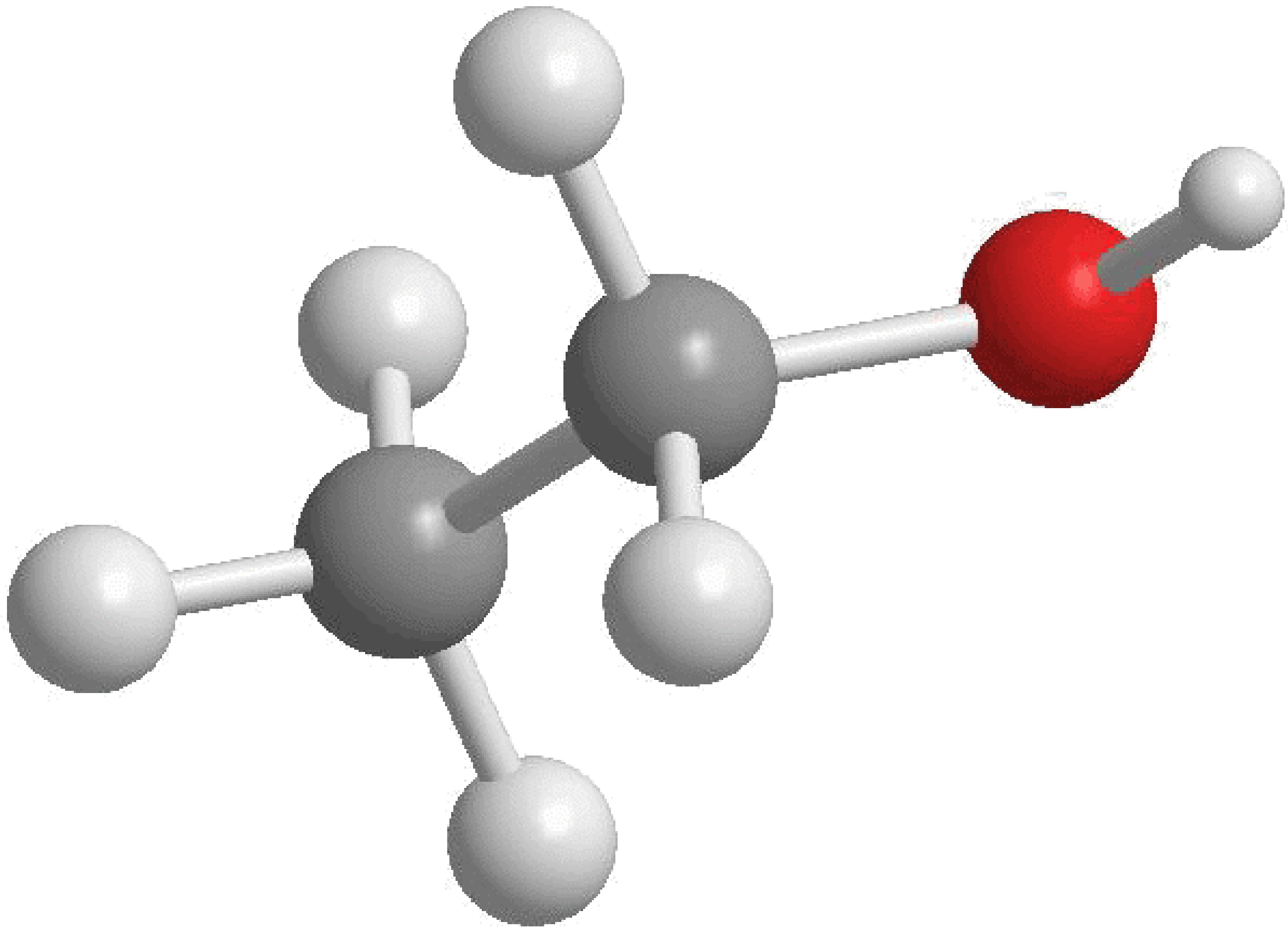


THE EISENHOWER INTERSTATE SYSTEM

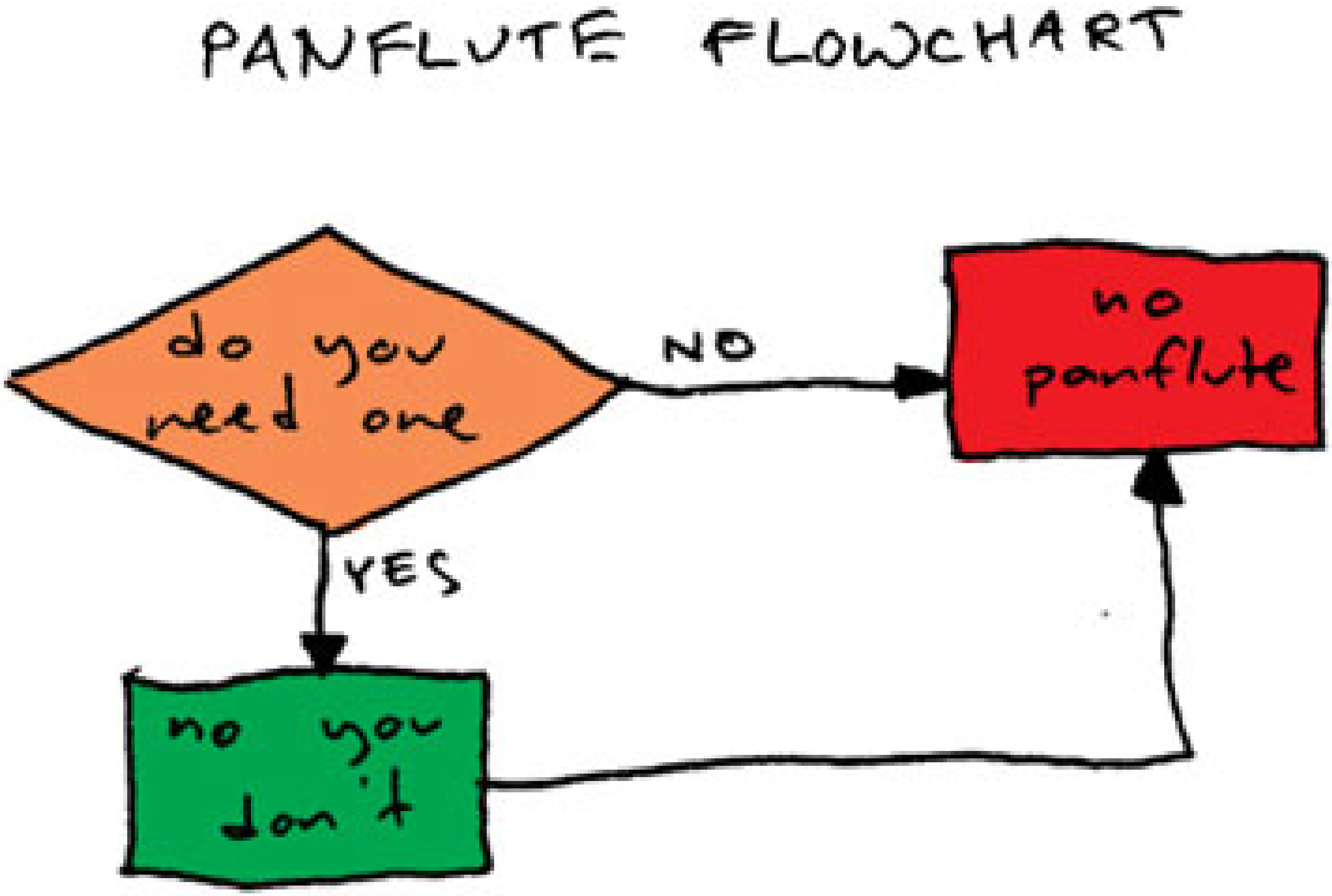
(simplified)

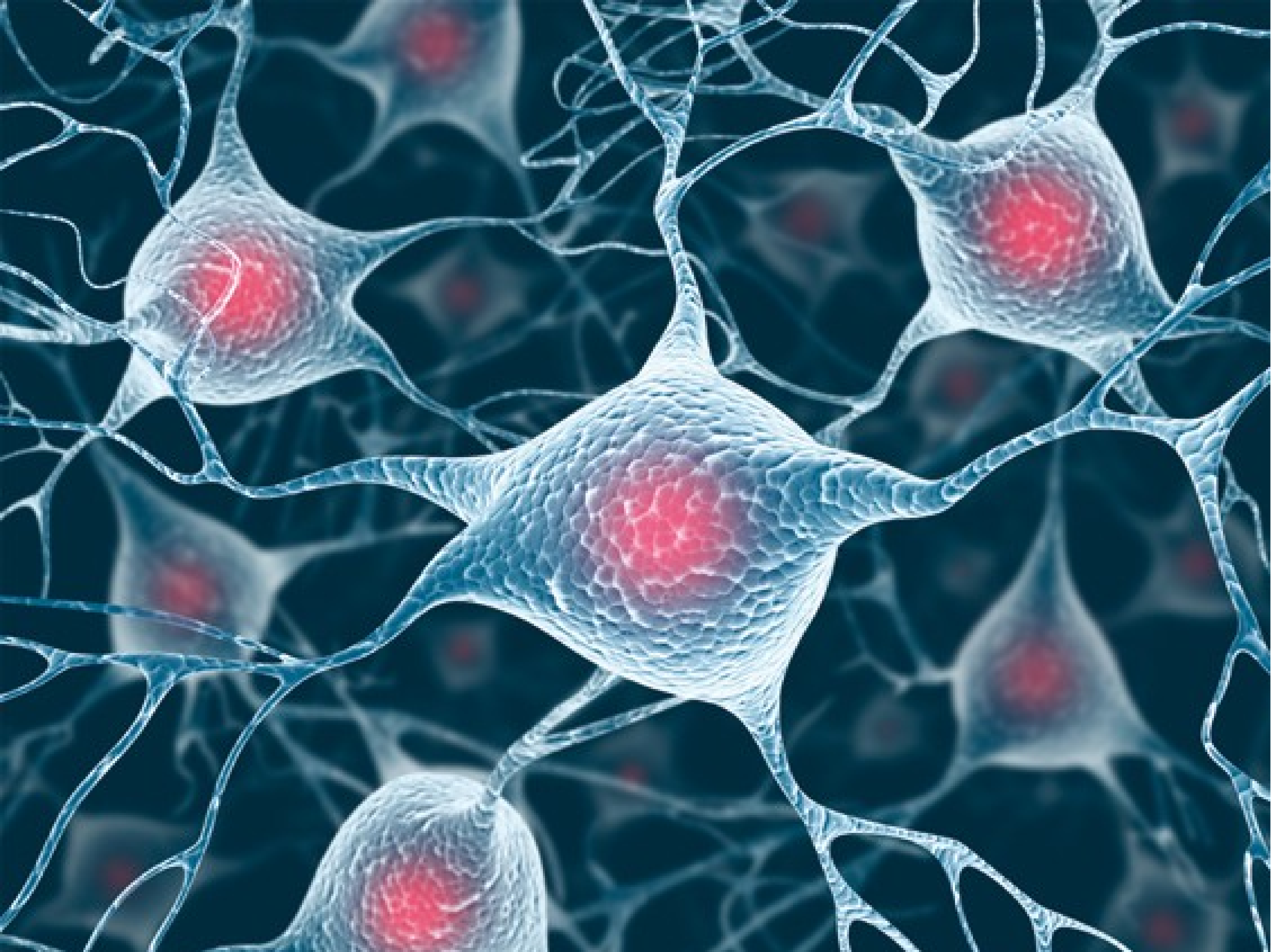
CHRIS YATES 2007

Chemical Bonds



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


PANFLUTE FLOWCHART
do you need one
NO
no panflute
YES
no you don't



facebook®

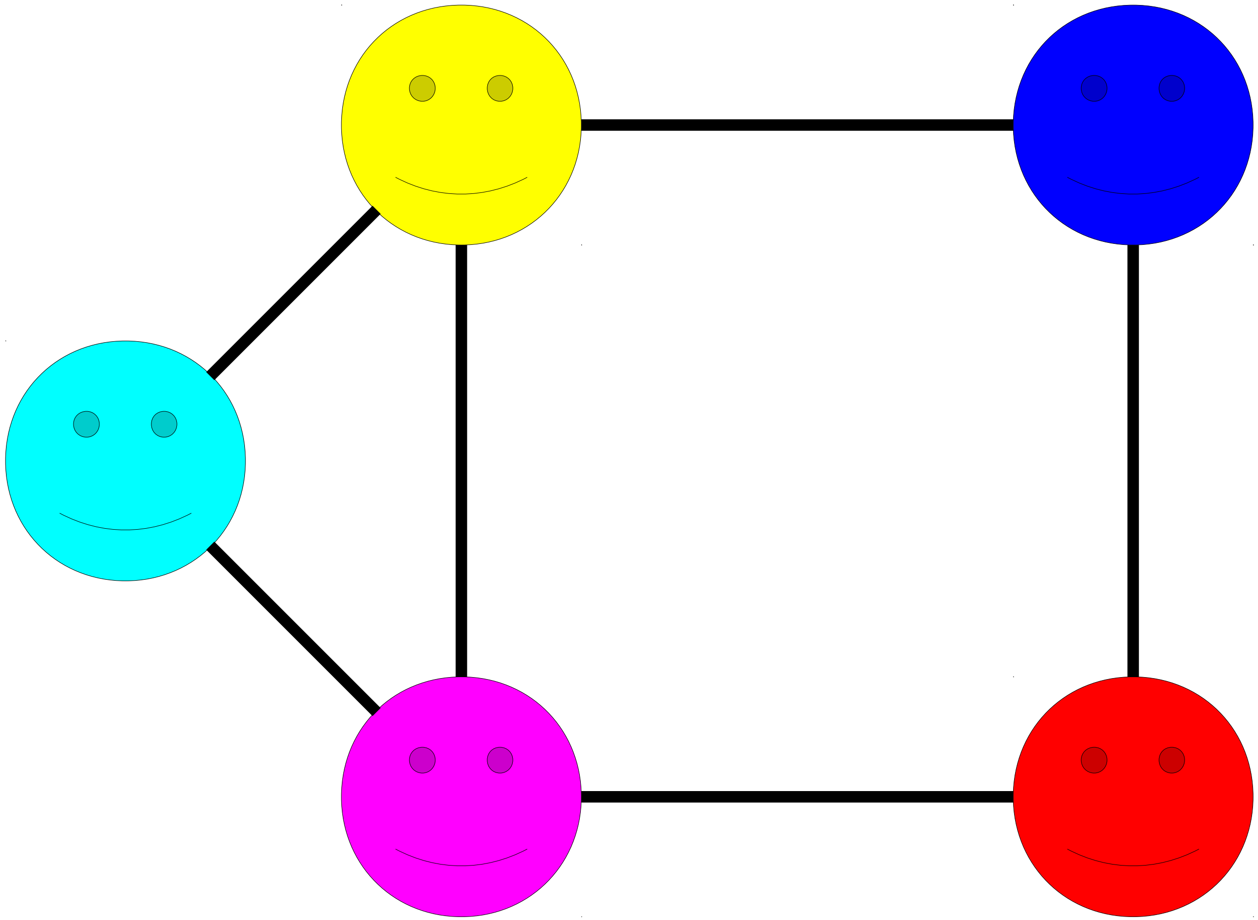
facebook

Me too!

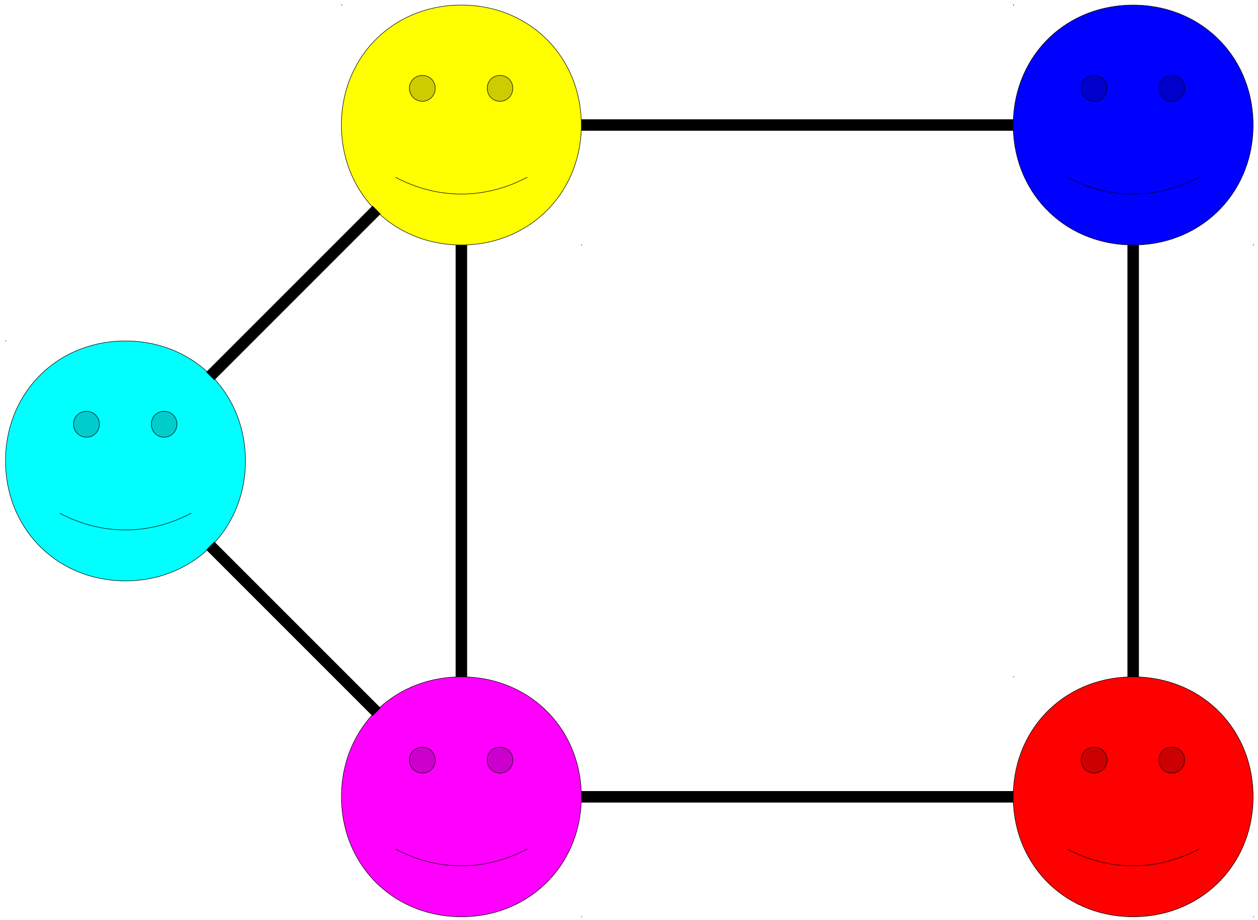
What's in Common

- Each of these structures consists of
  - a collection of 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.

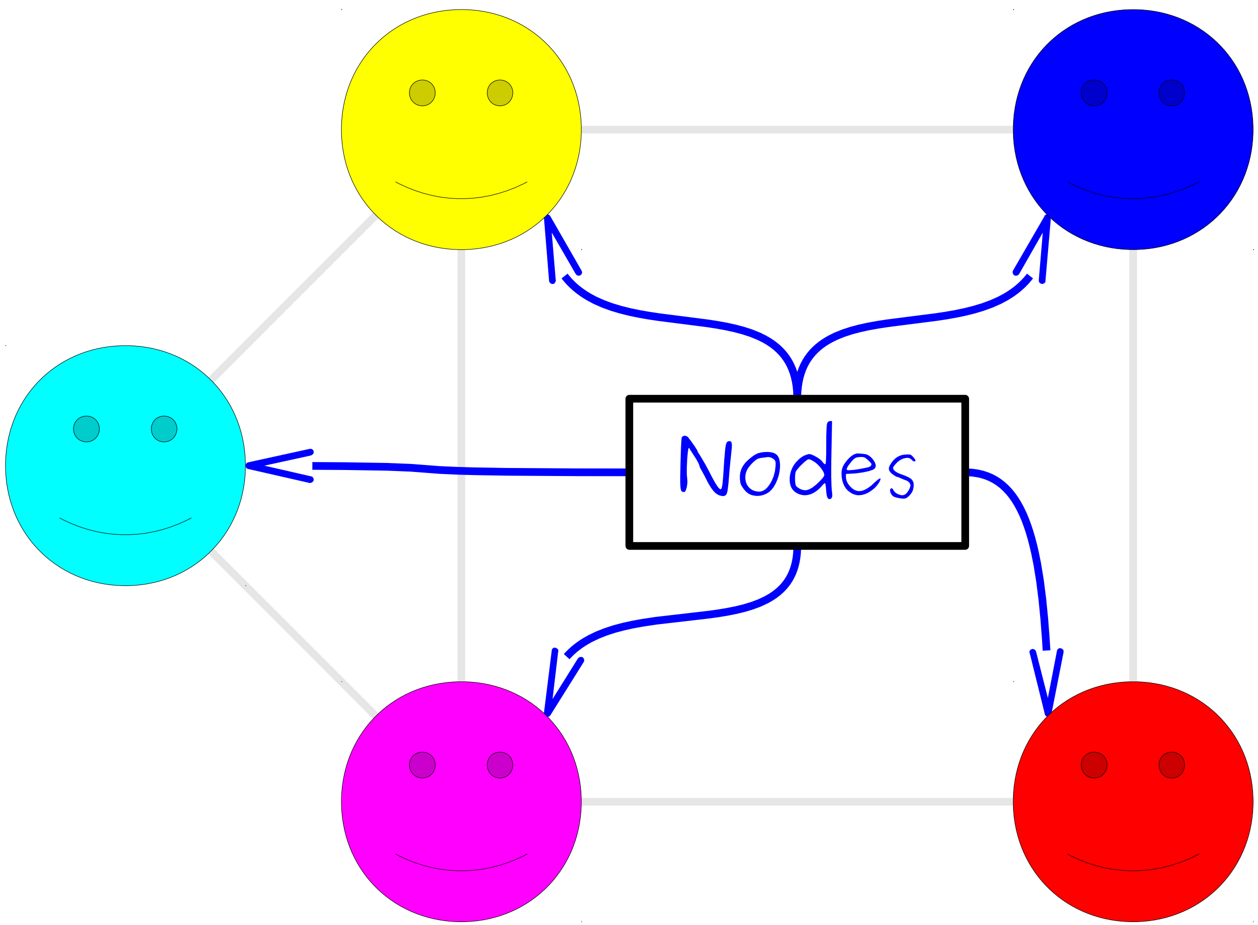


A ***graph*** is a mathematical structure for representing relationships.



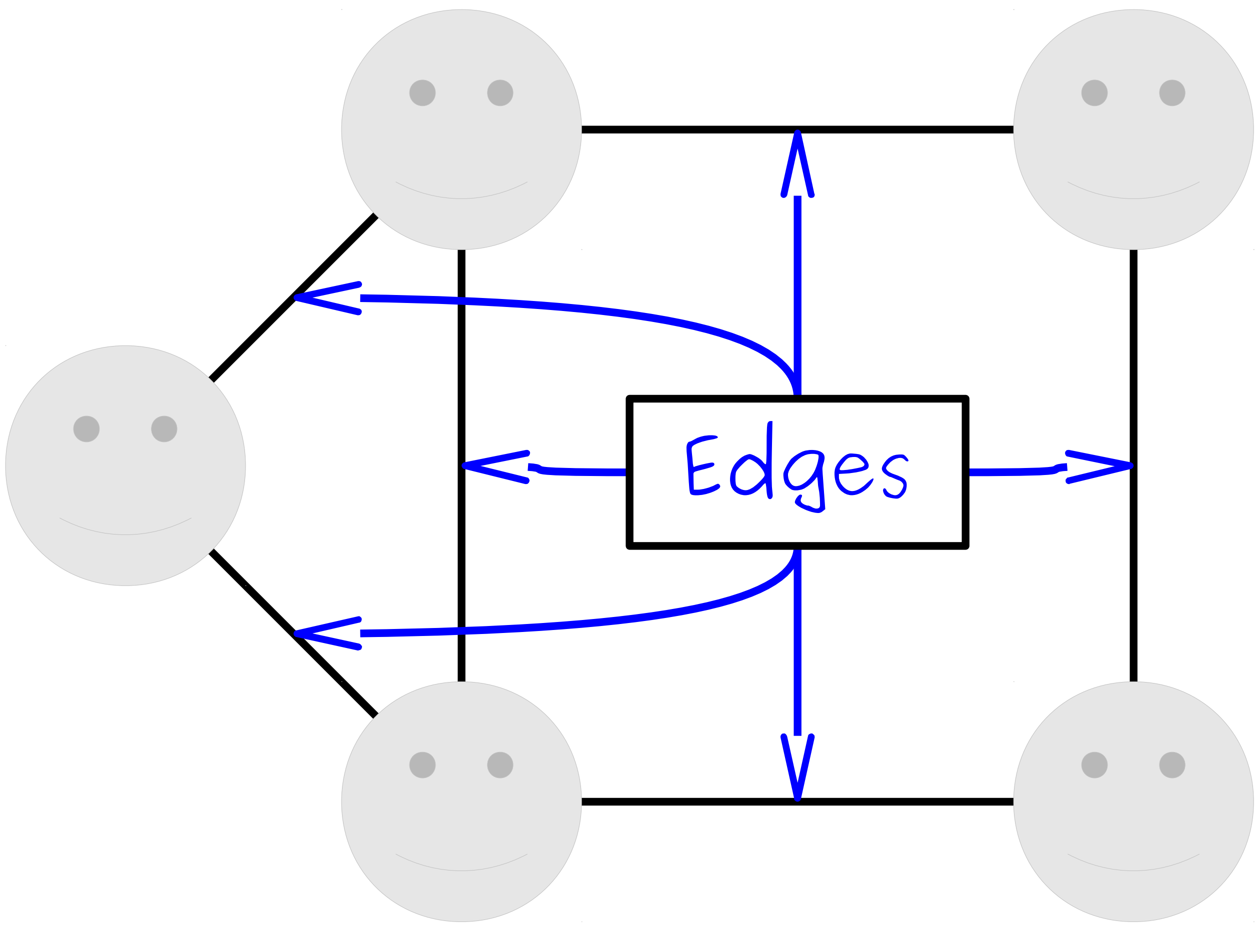
A graph consists of a set of ***nodes*** (or ***vertices***) connected by ***edges*** (or ***arcs***)

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



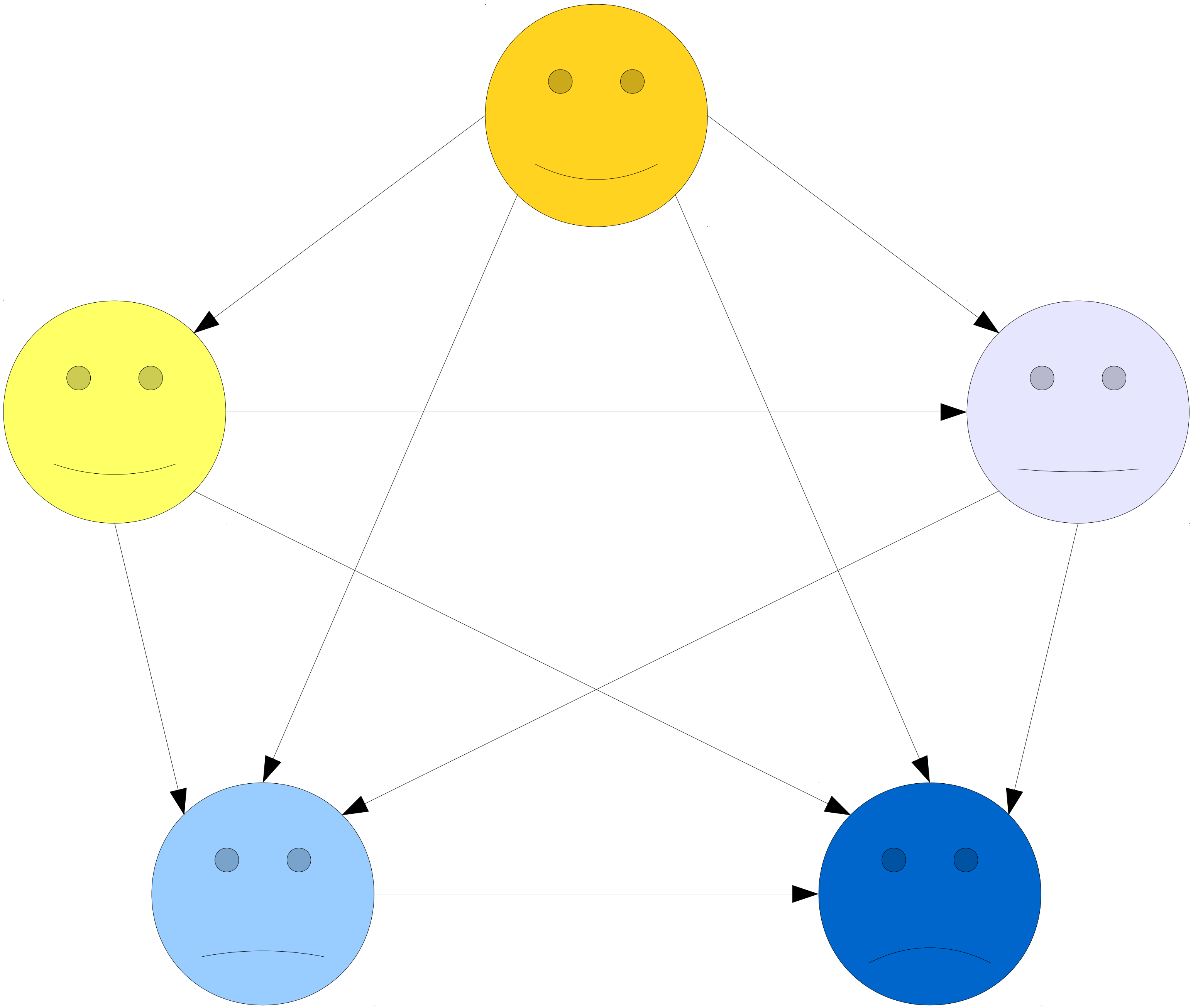
A graph consists of a set of ***nodes*** (or ***vertices***) connected by ***edges*** (or ***arcs***)

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

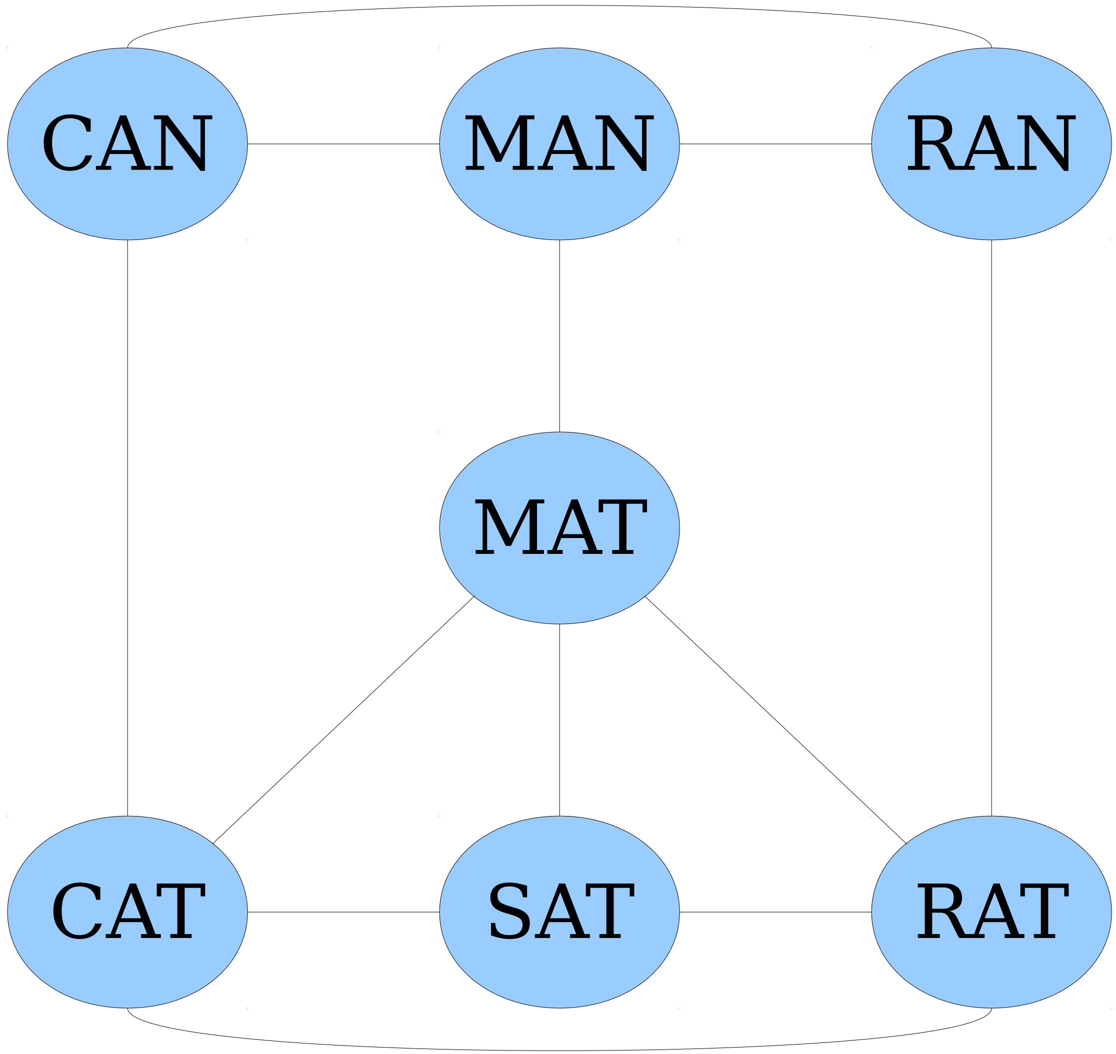


A graph consists of a set of ***nodes*** (or ***vertices***) connected by ***edges*** (or ***arcs***)

Some graphs are ***directed***.



Some graphs are ***undirected***.



Going forward, we're primarily going to focus on undirected graphs.

The term “graph” generally refers to undirected graphs unless specified otherwise.

Formalizing Graphs

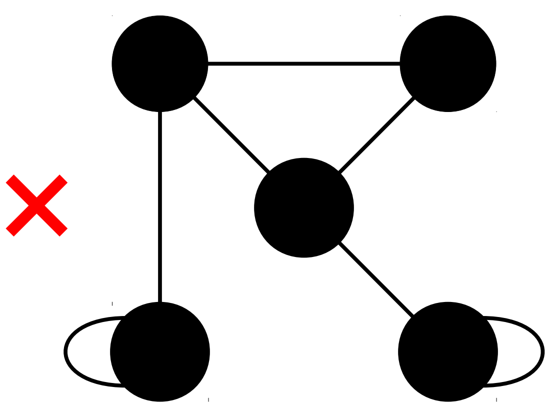
- How might we define a graph mathematically?
- We need to specify
  - what the nodes in the graph are, and
  - which edges are in the graph.
- The nodes can be pretty much anything.
- What about the edges?

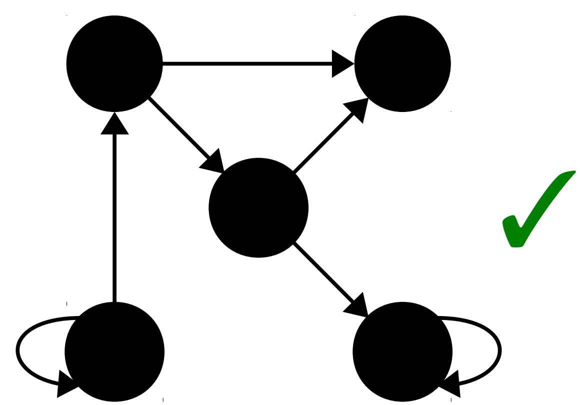
Formalizing Graphs

- An ***unordered pair*** is a set $\{a, b\}$ of two elements (remember that sets are unordered).
  - $\{0, 1\} = \{1, 0\}$
- An ***undirected graph*** is an ordered pair $G = (V, E)$, where
  - V is a set of nodes, which can be anything, and
  - E is a set of edges, which are unordered pairs of nodes drawn from V.
- A ***directed graph*** is an ordered pair $G = (V, E)$, where
  - V is a set of nodes, which can be anything, and
  - E is a set of edges, which are *ordered* pairs of nodes drawn from V.

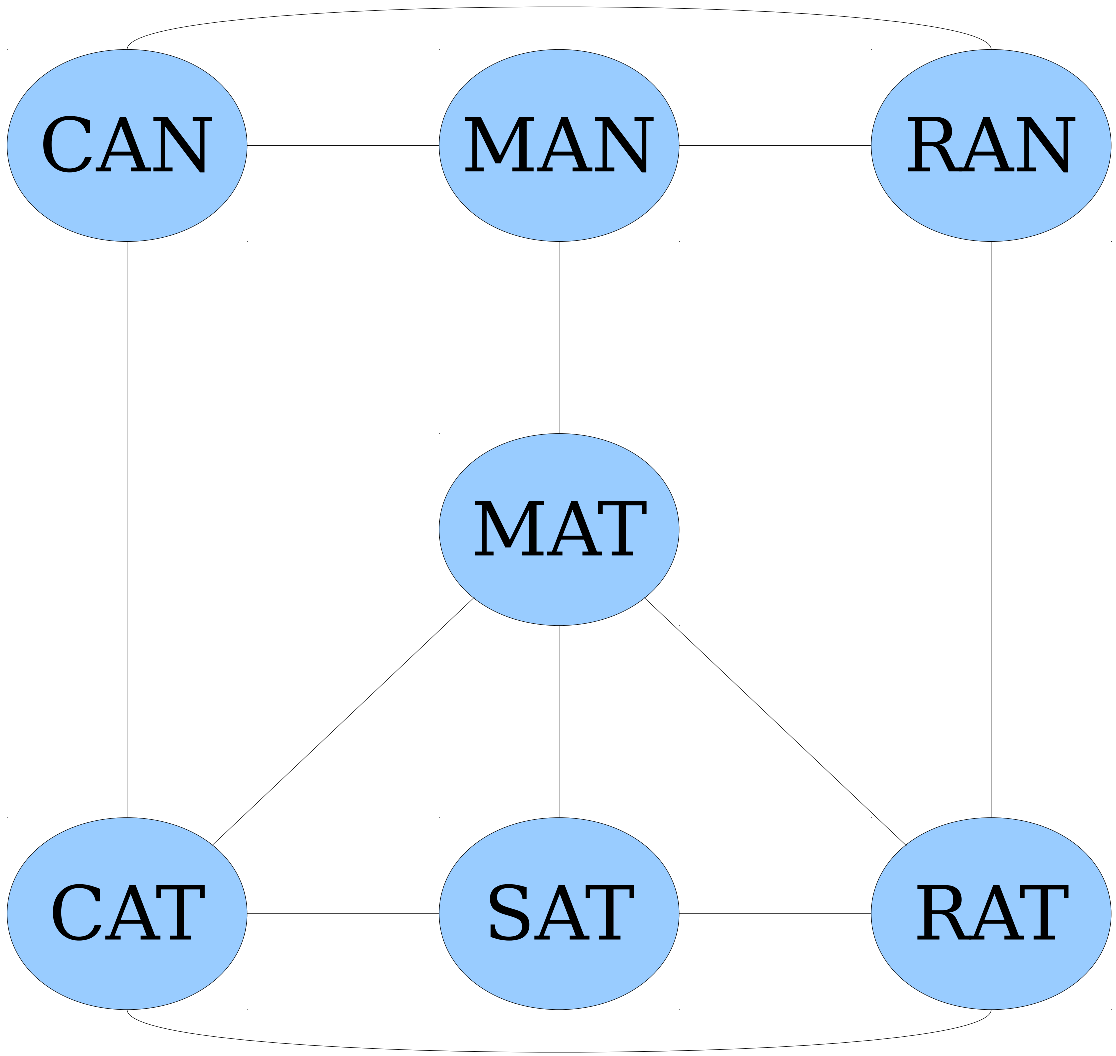
Self-Loops

- An edge from a node to itself is called a ***self-loop***.
- In undirected graphs, self-loops are generally not allowed unless specified otherwise.
  - This is mostly to keep the math easier. If you allow self-loops, a lot of results get messier and harder to state.
- In directed graphs, self-loops are generally allowed unless specified otherwise.

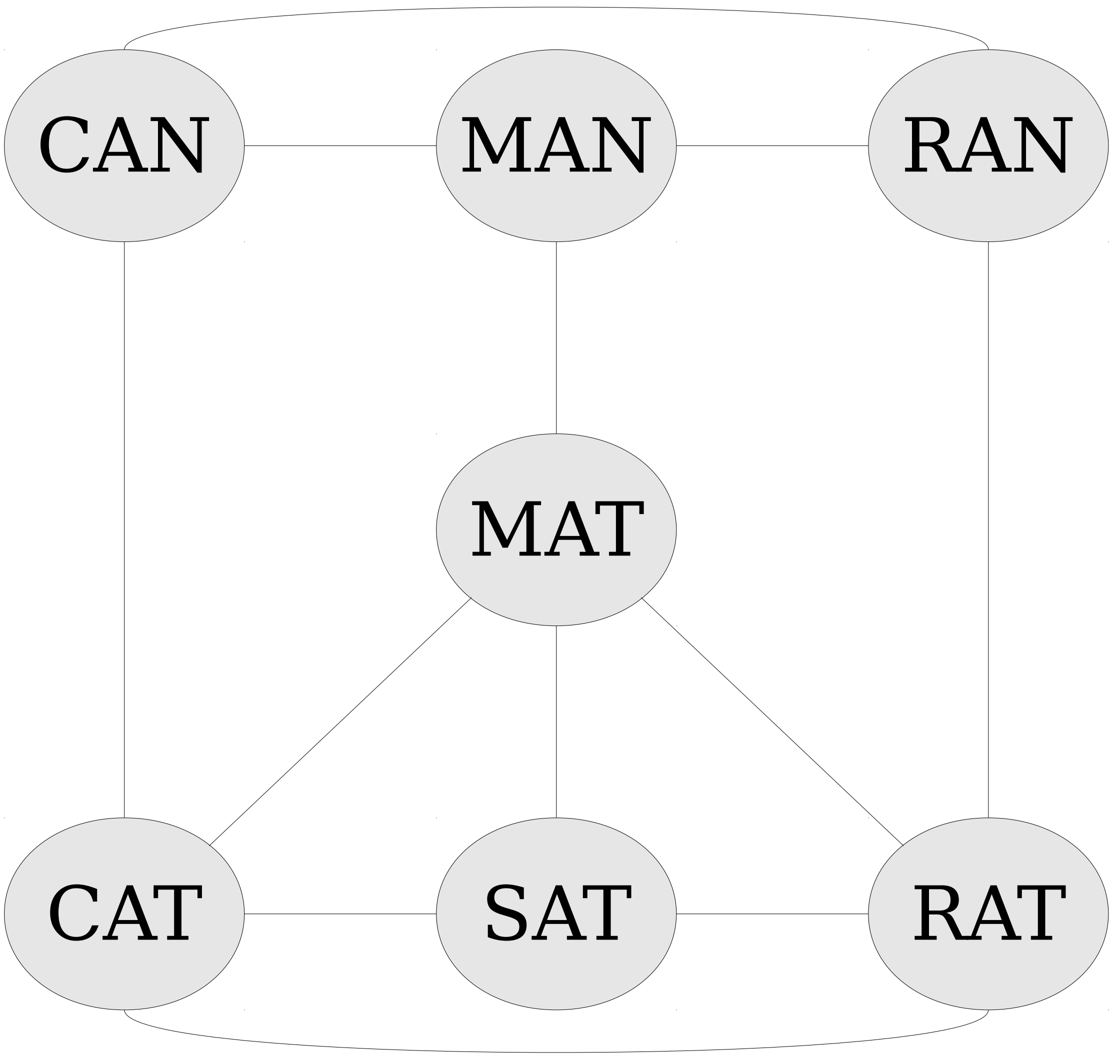




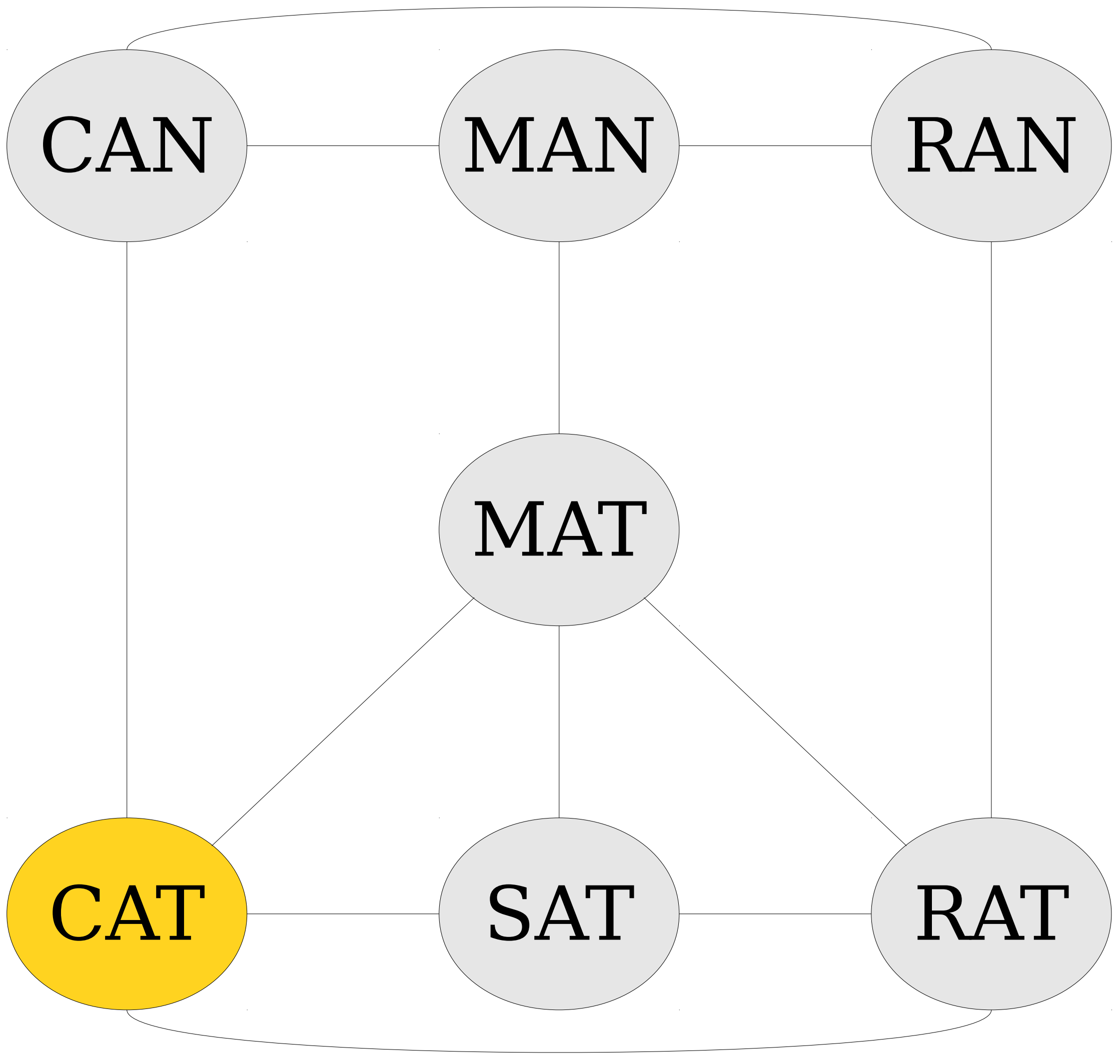
Standard Graph Terminology



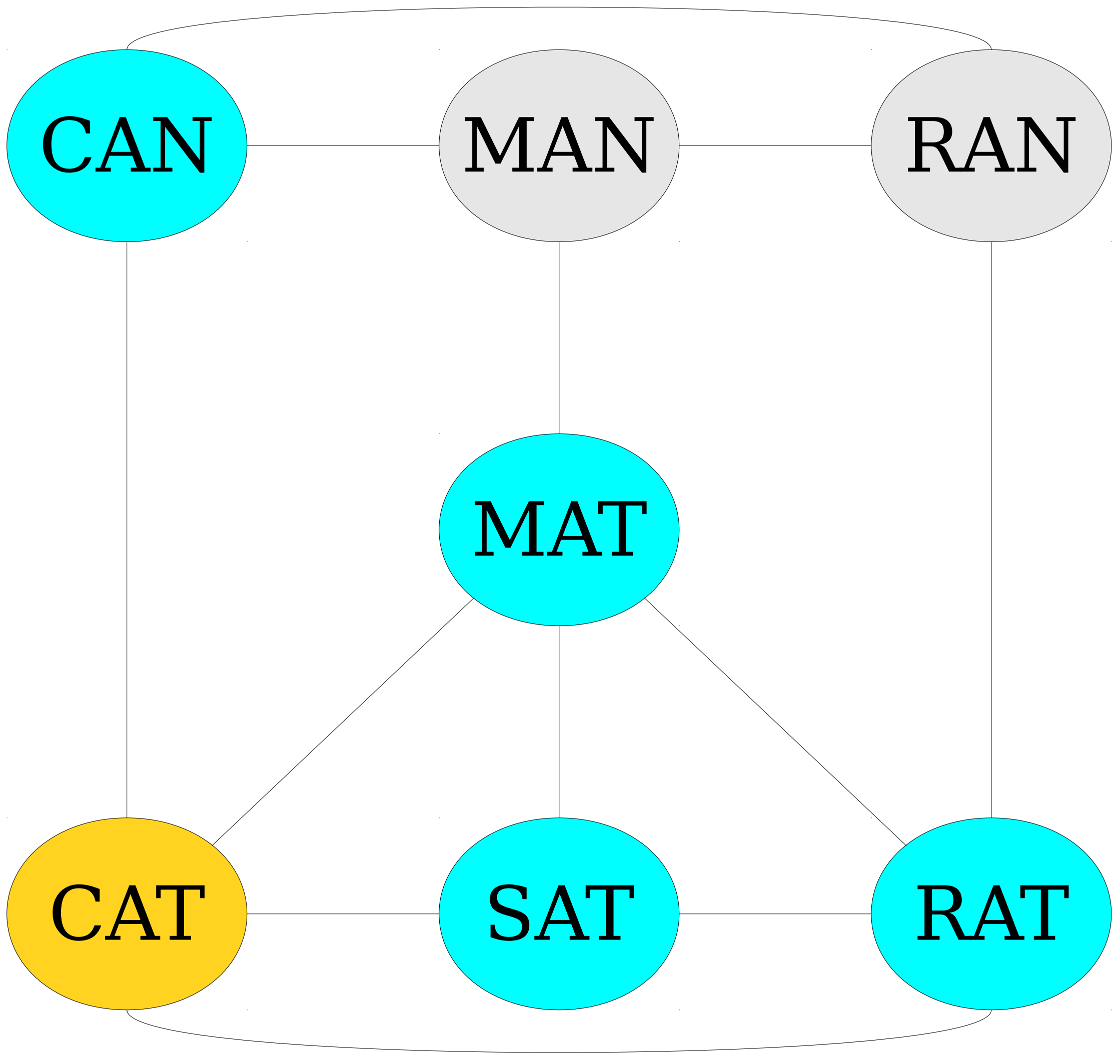
Two nodes are called ***adjacent*** if there is an edge between them.



Two nodes are called ***adjacent*** if there is an edge between them.



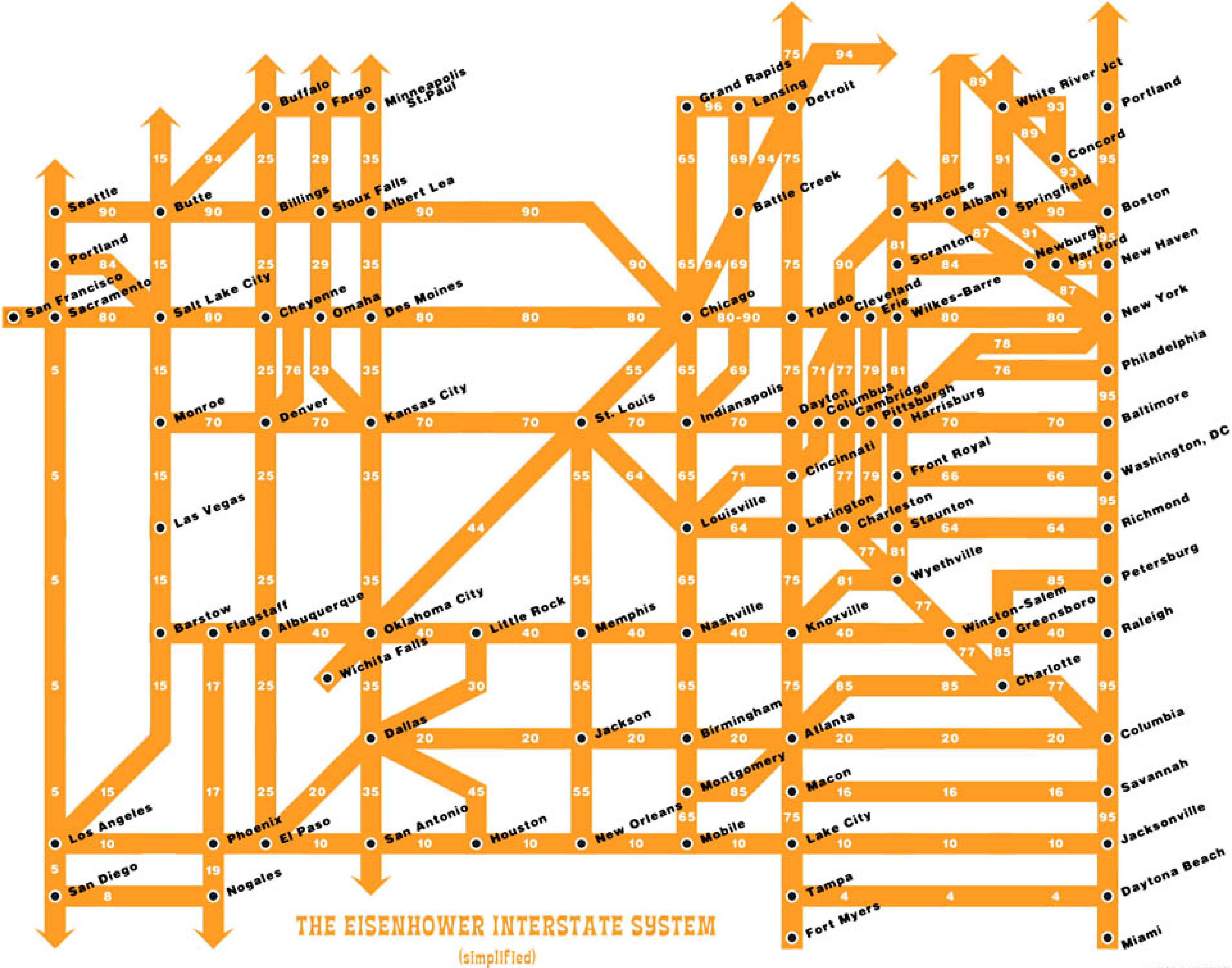
Two nodes are called ***adjacent*** if there is an edge between them.



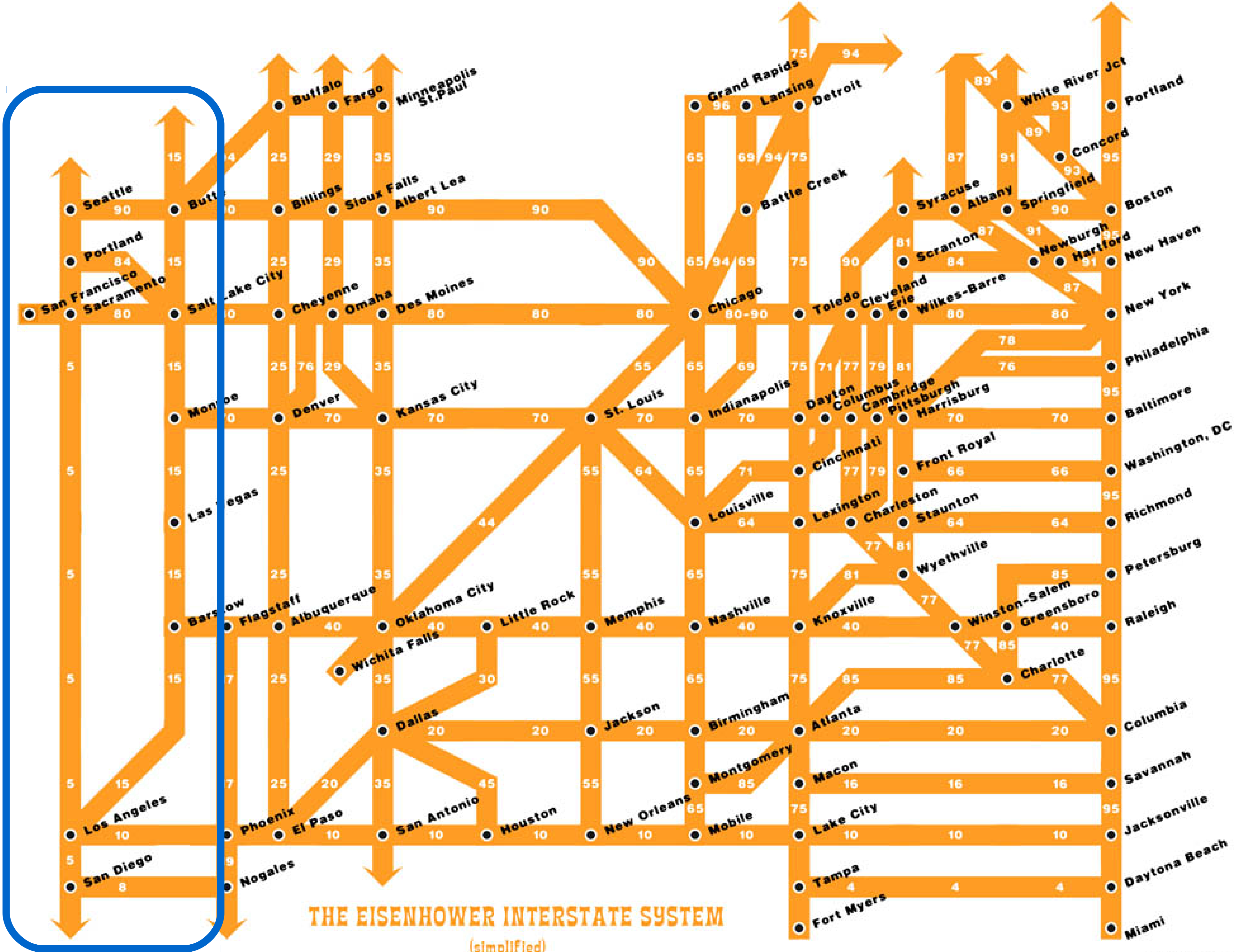
Two nodes are called ***adjacent*** if there is an edge between them.

Using our Formalisms

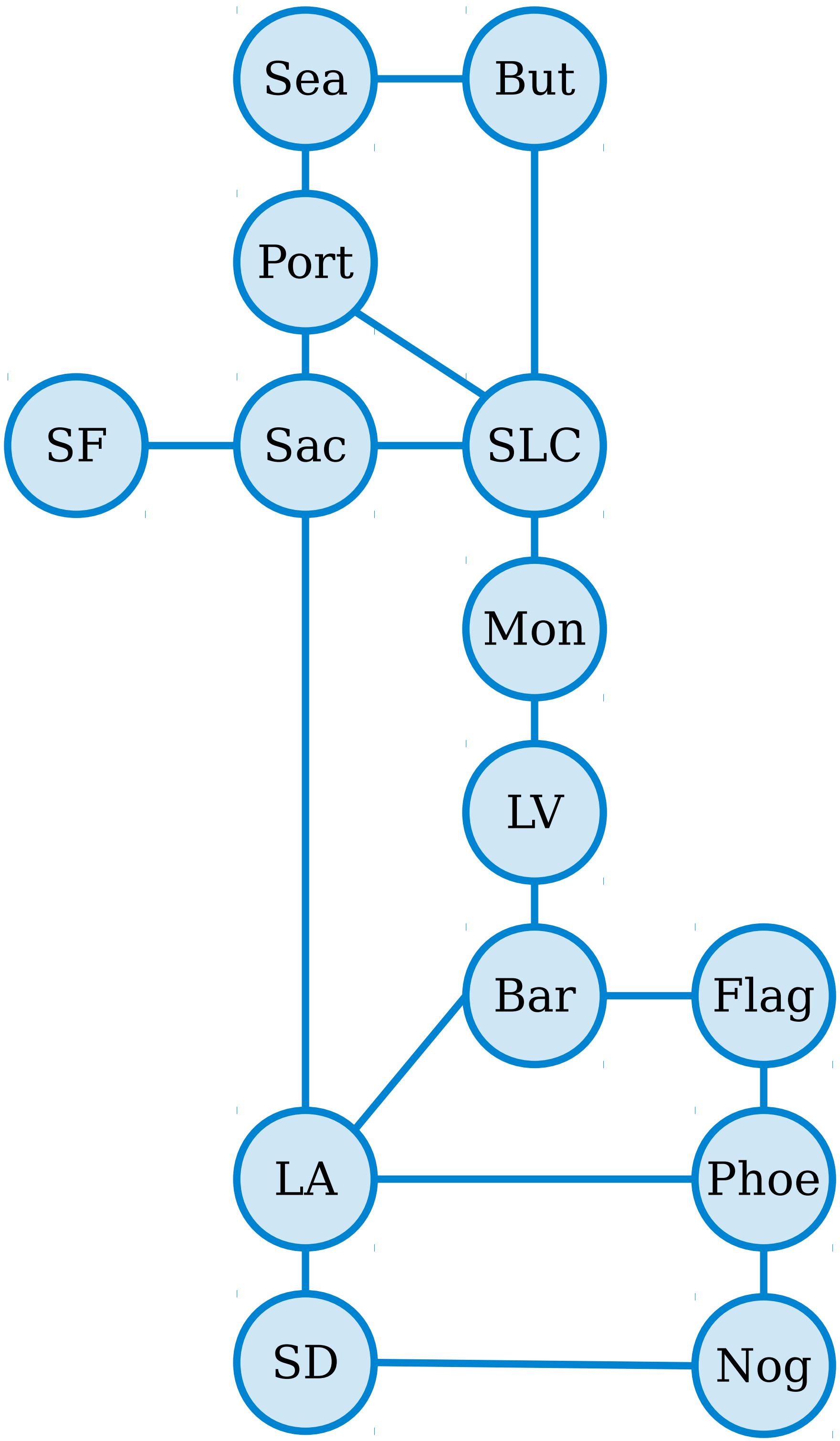
- Let $G = (V, E)$ be a graph.
- Intuitively, two nodes are adjacent if they're linked by an edge.
- Formally speaking, we say that two nodes $u, v \in V$ are ***adjacent*** if $\{u, v\} \in E$.

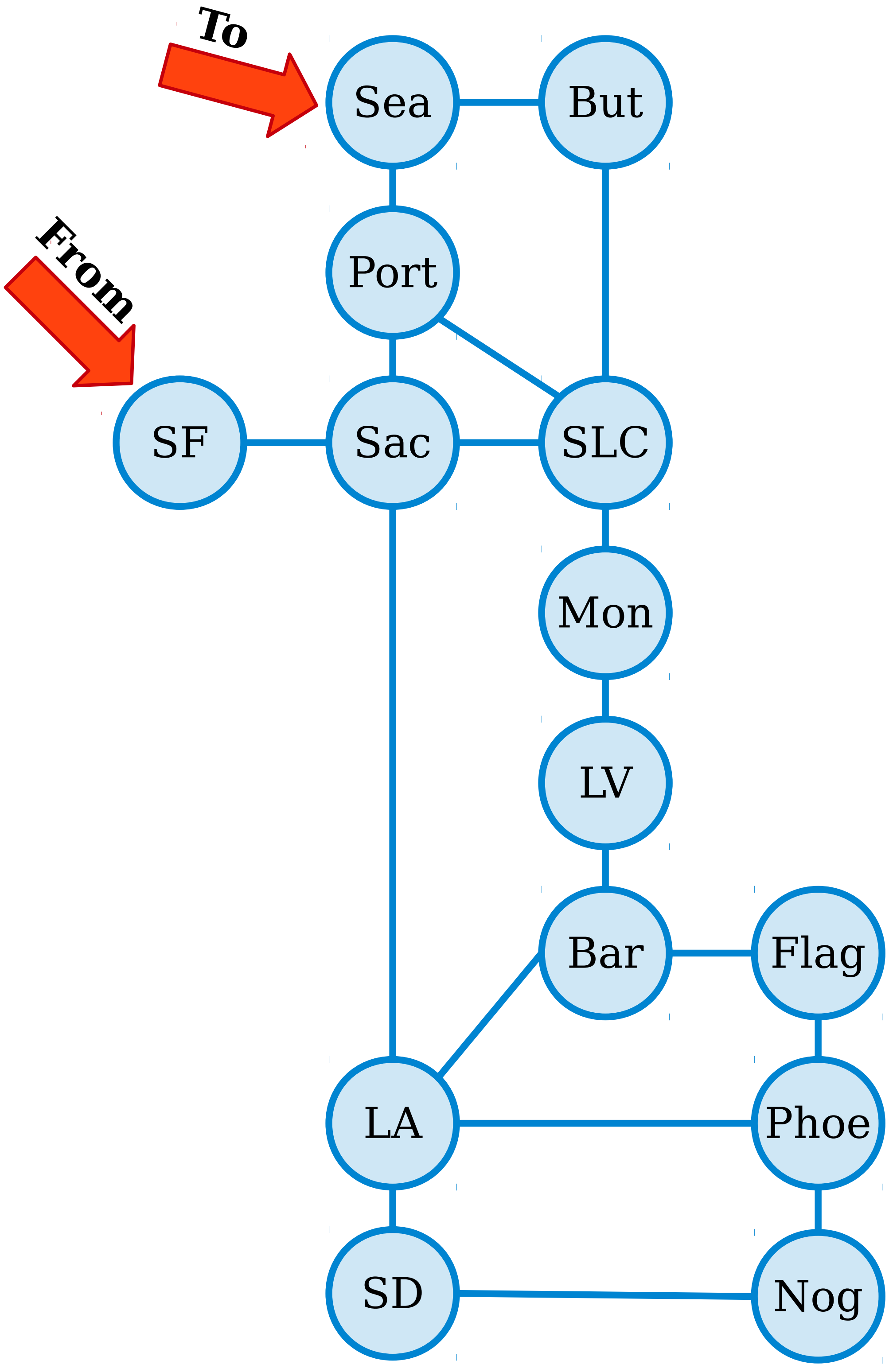


http://strangemaps.files.wordpress.com/2007/02/fullinterstatemap-web.jpg

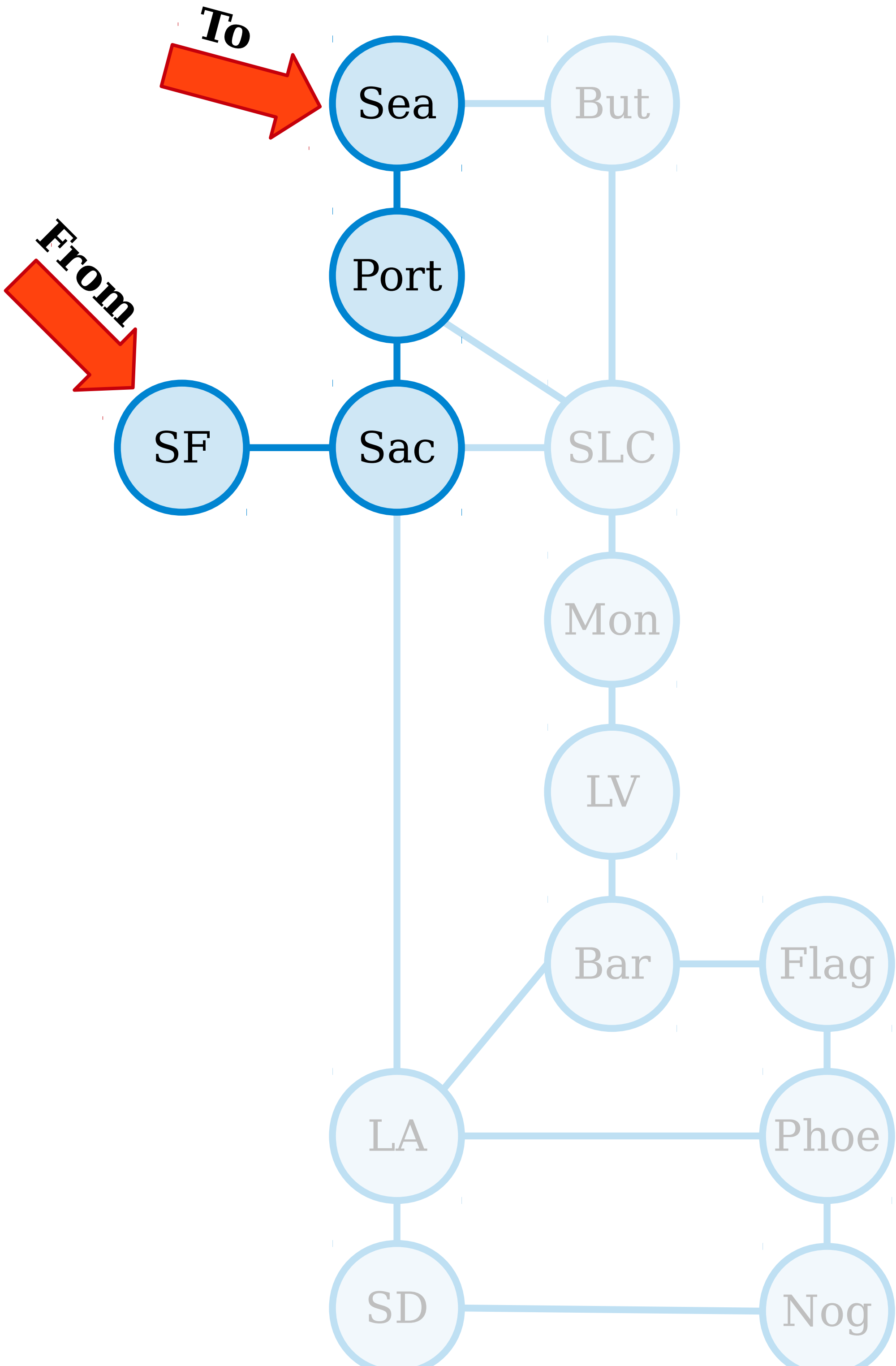


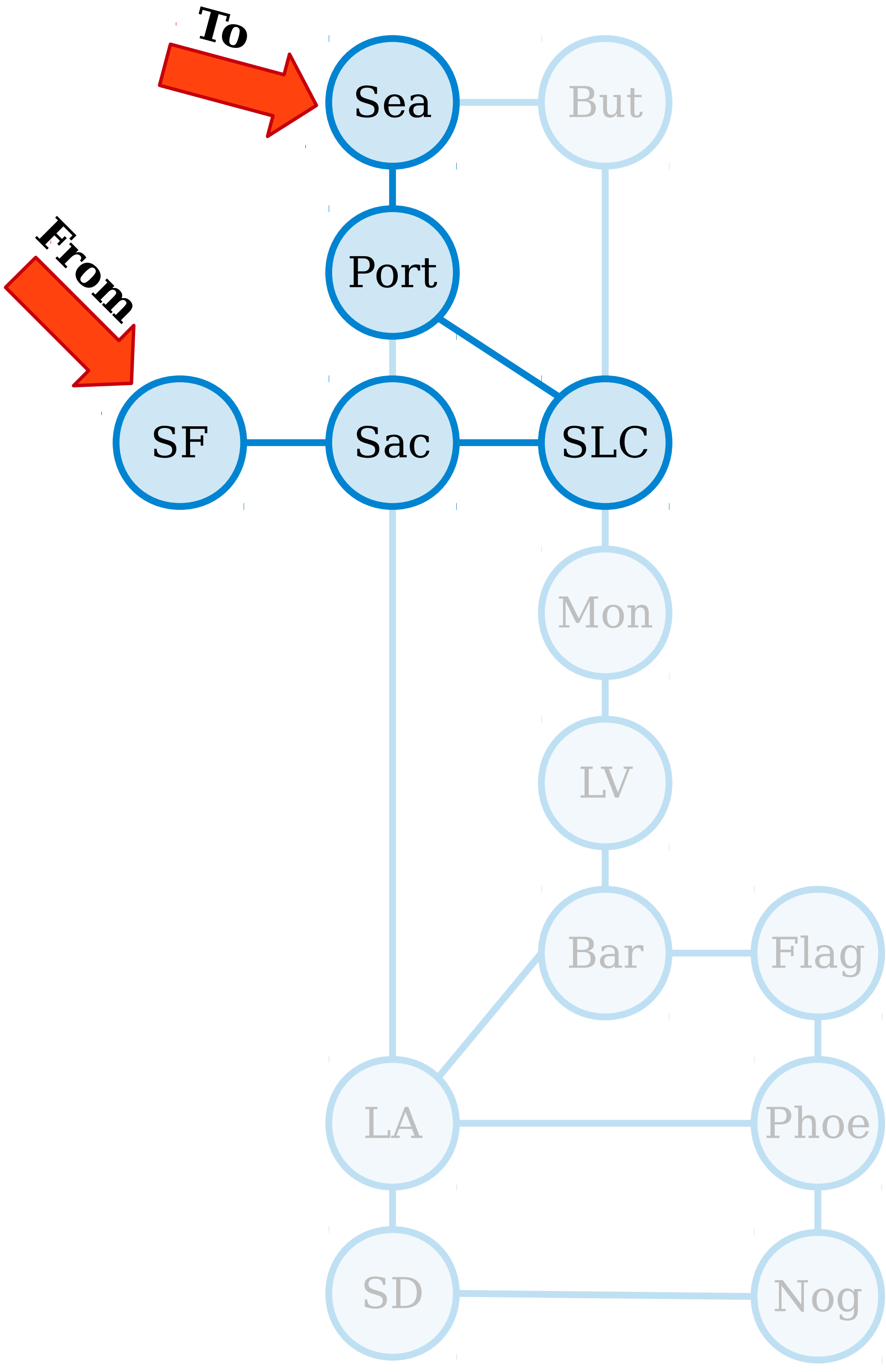
http://strangemaps.files.wordpress.com/2007/02/fullinterstatemap-web.jpg

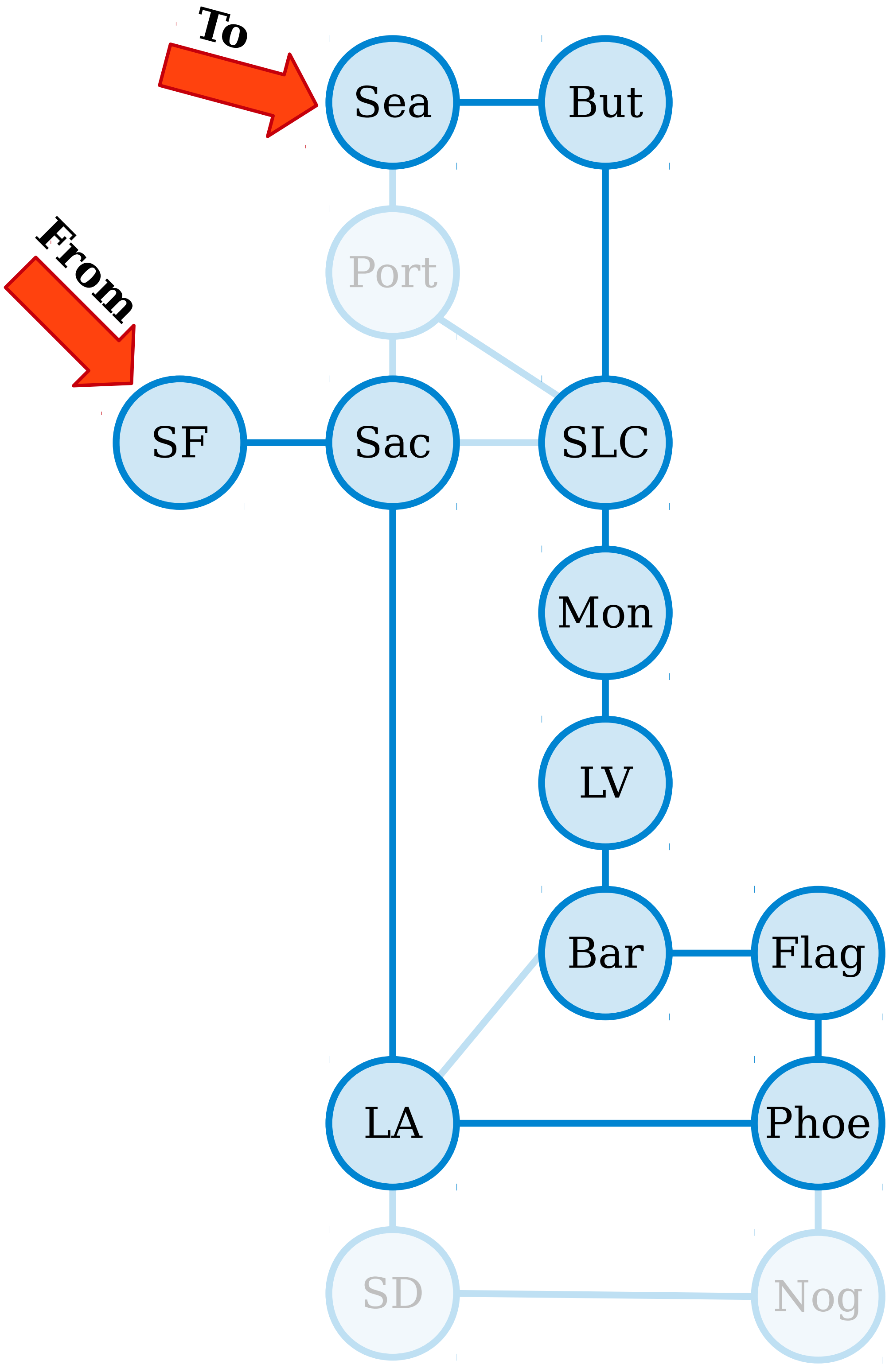

Sea
But
Port
SF
Sac
SLC
Mon
LV
Bar
Flag
LA
Phoe
SD
Nog



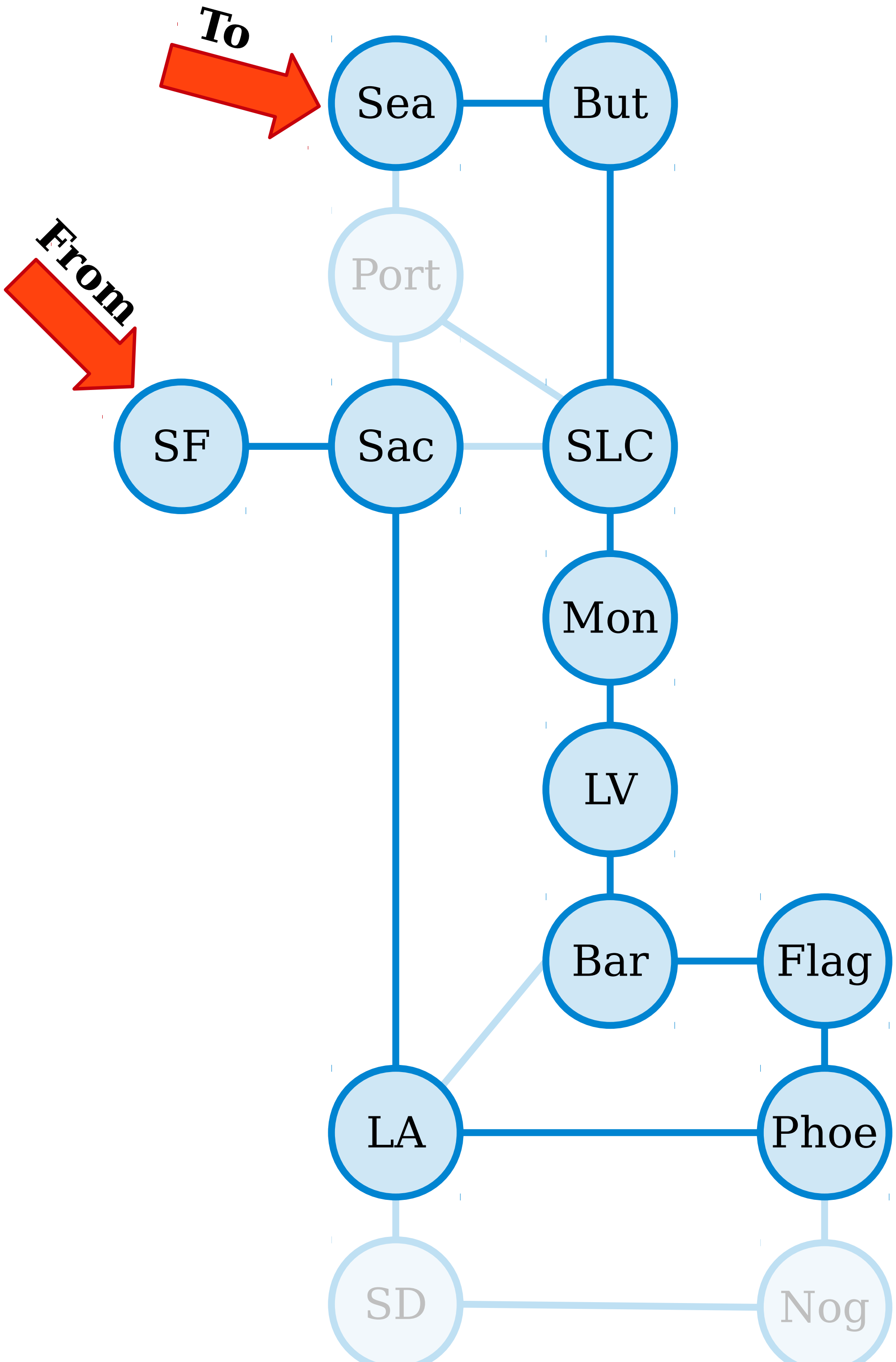
To
From
Sea
But
Port
SF
Sac
SLC
Mon
LV
Bar
Flag
LA
Phoe
SD
Nog


To
From
Sea
But
Port
SF
Sac
SLC
Mon
LV
Bar
Flag
LA
Phoe
SD
Nog

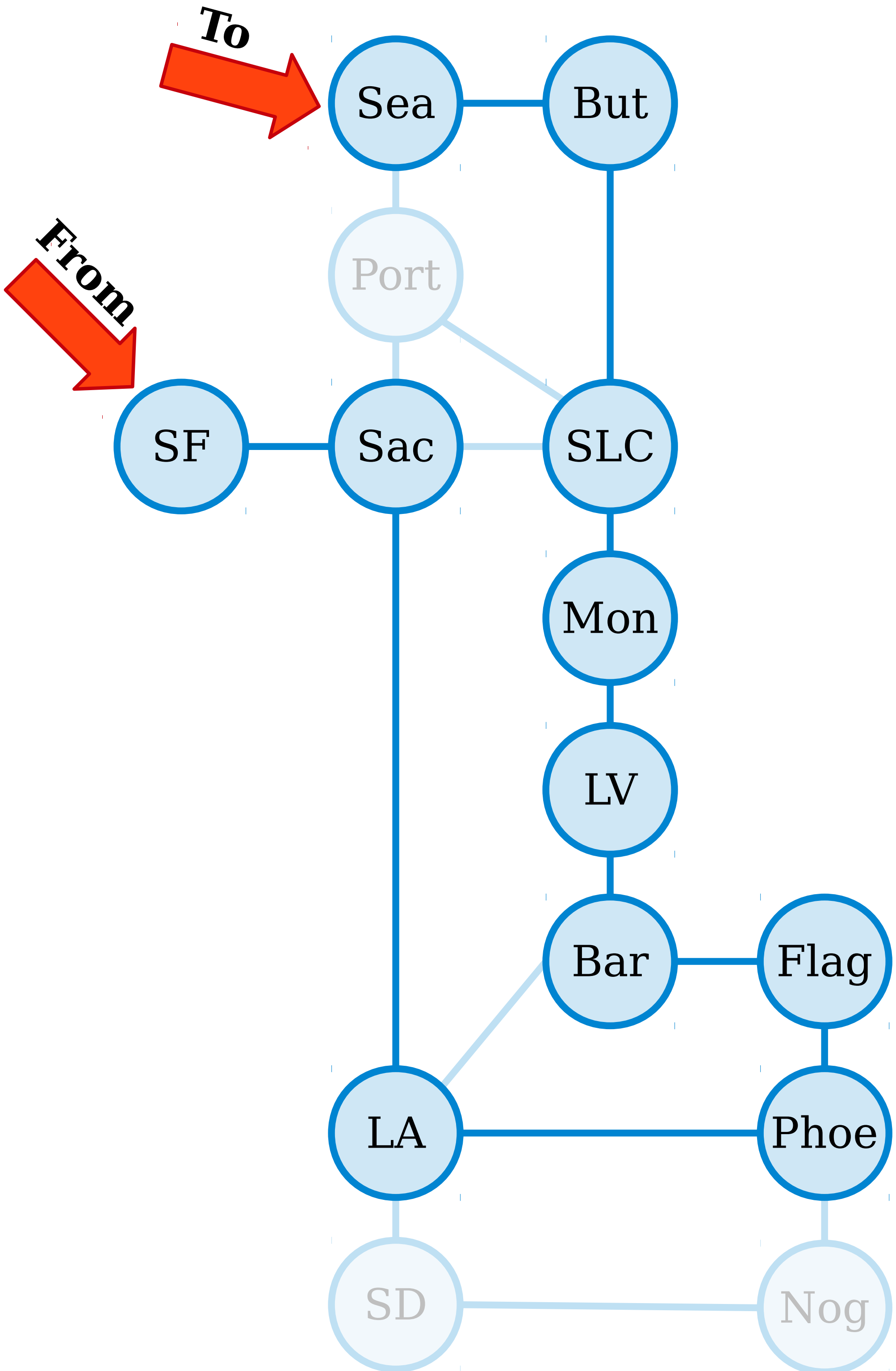

To
From
Sea
But
Port
SF
Sac
SLC
Mon
LV
Bar
Flag
LA
Phoe
SD
Nog



To
From
Sea
But
Port
SF
Sac
SLC
Mon
LV
Bar
Flag
LA
Phoe
SD
Nog

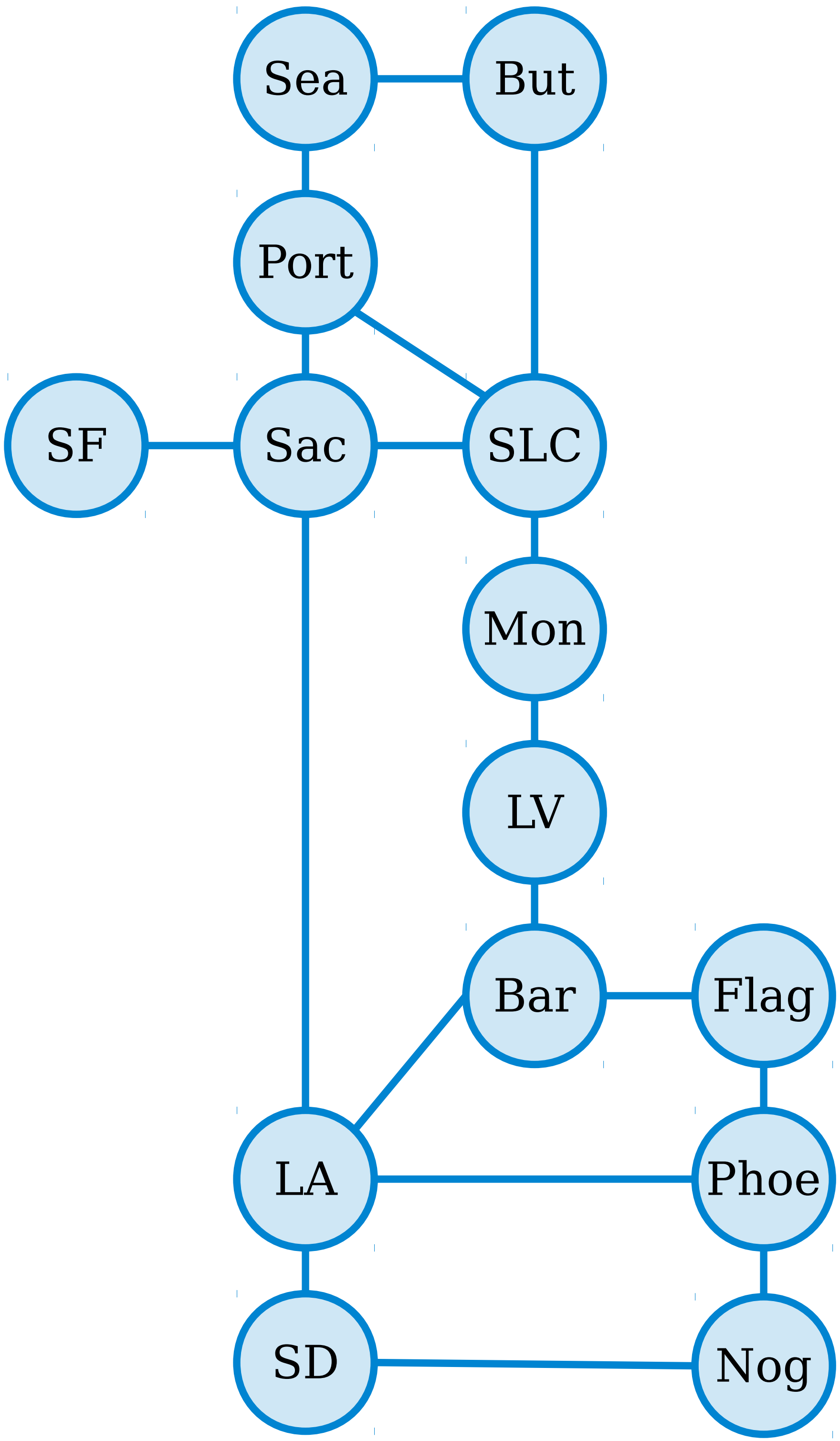


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.



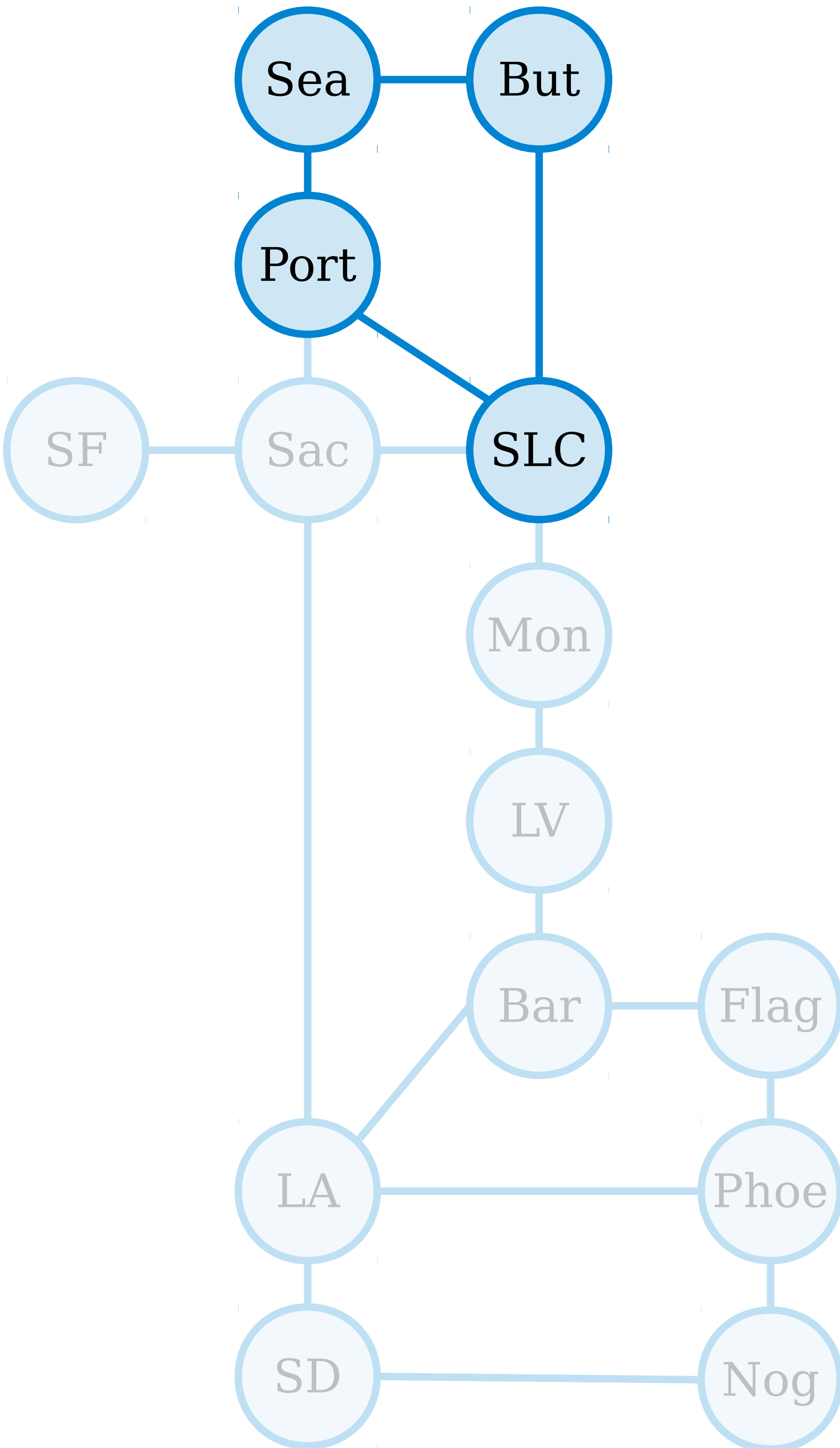
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.



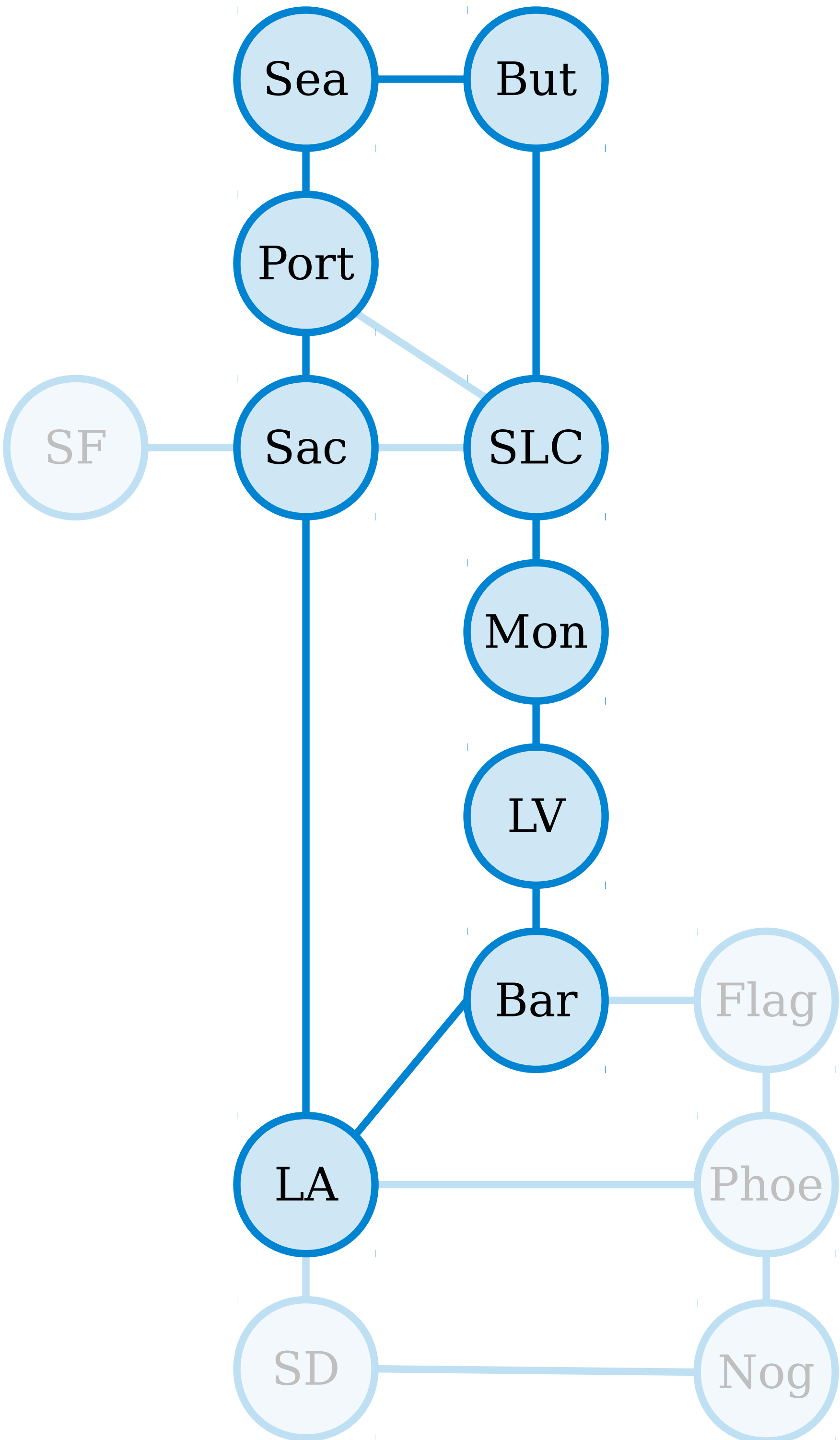
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.



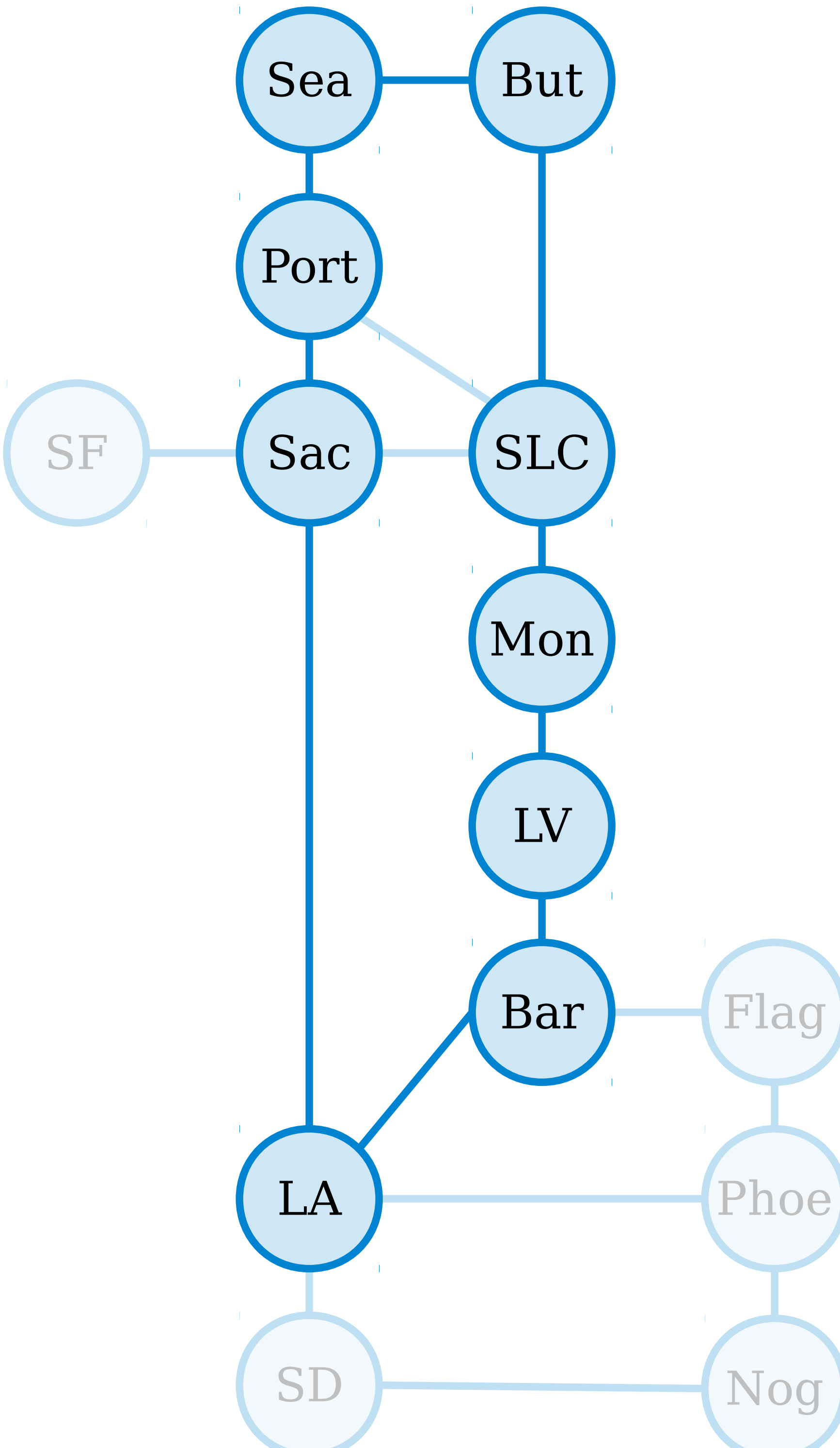
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

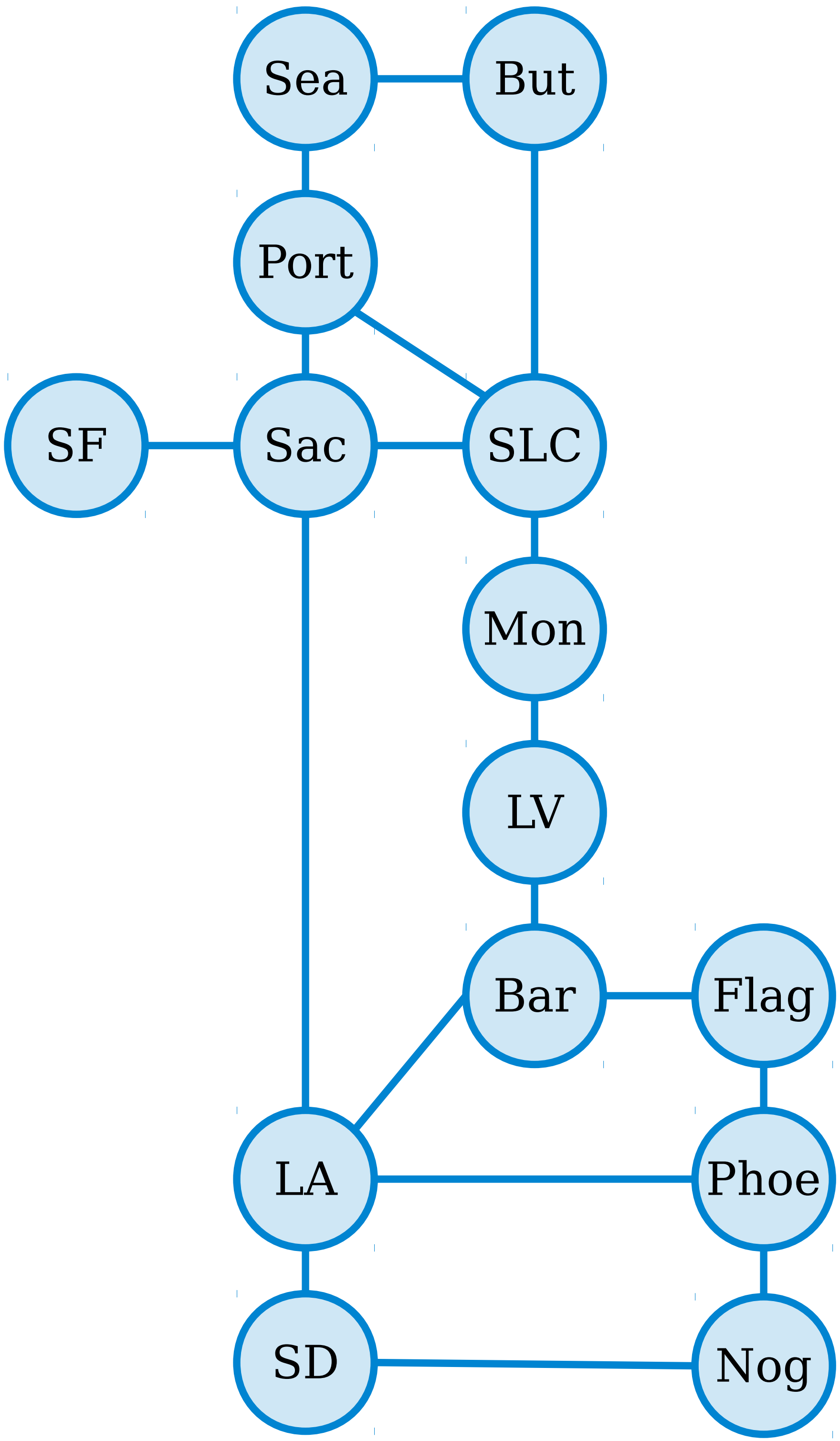
The ***length*** of a path is the number of edges in it.



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

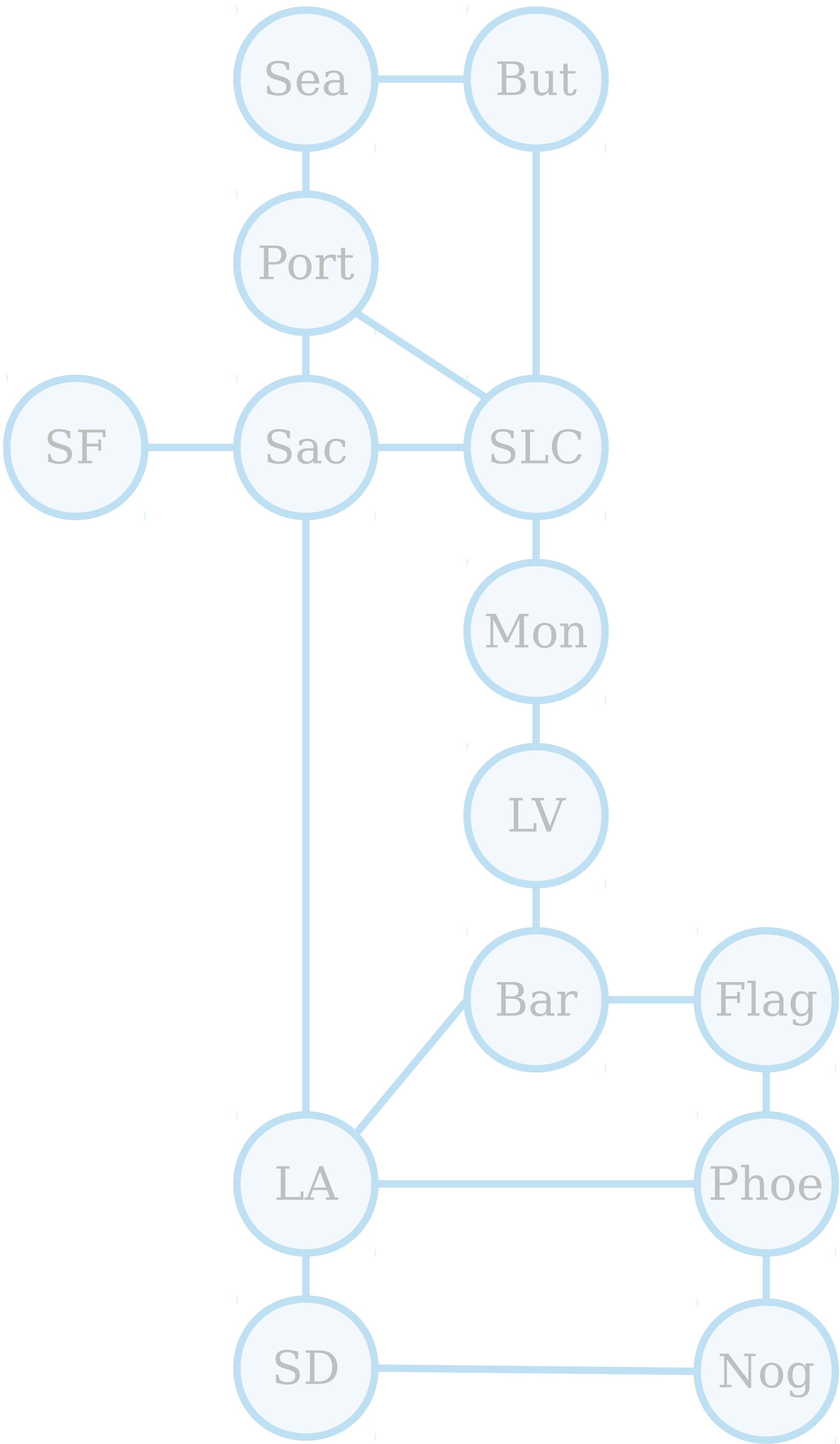
A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

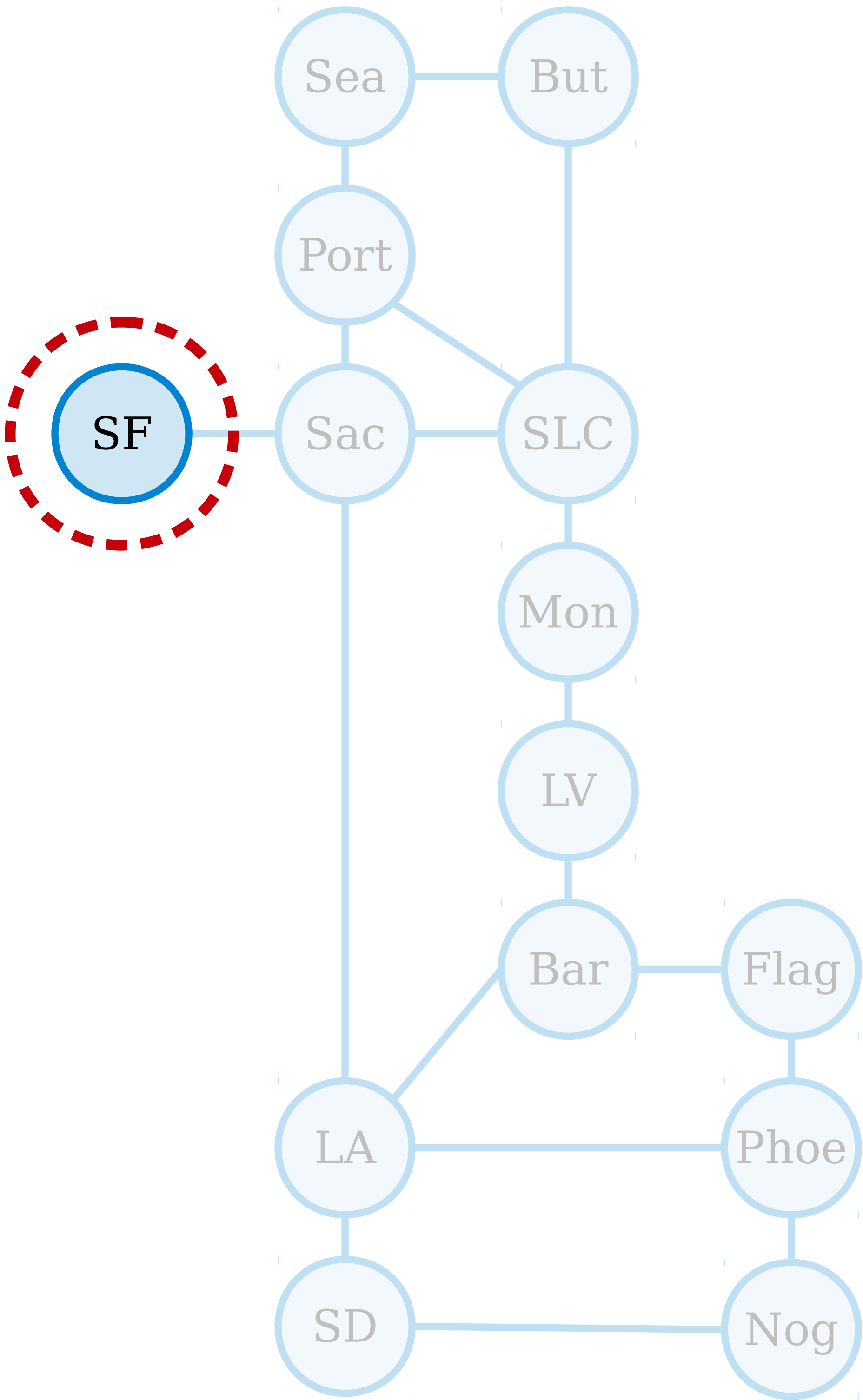
A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

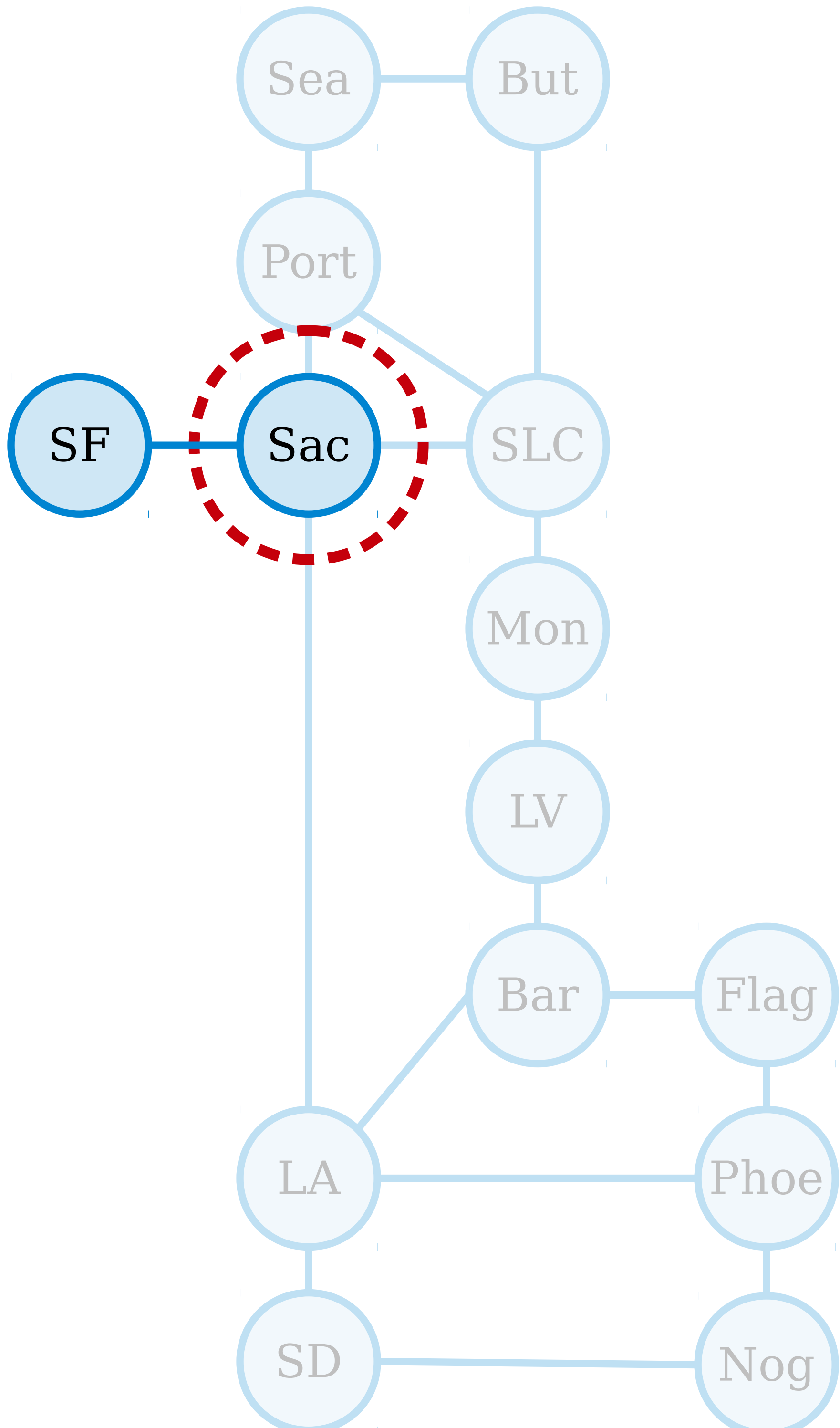
A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

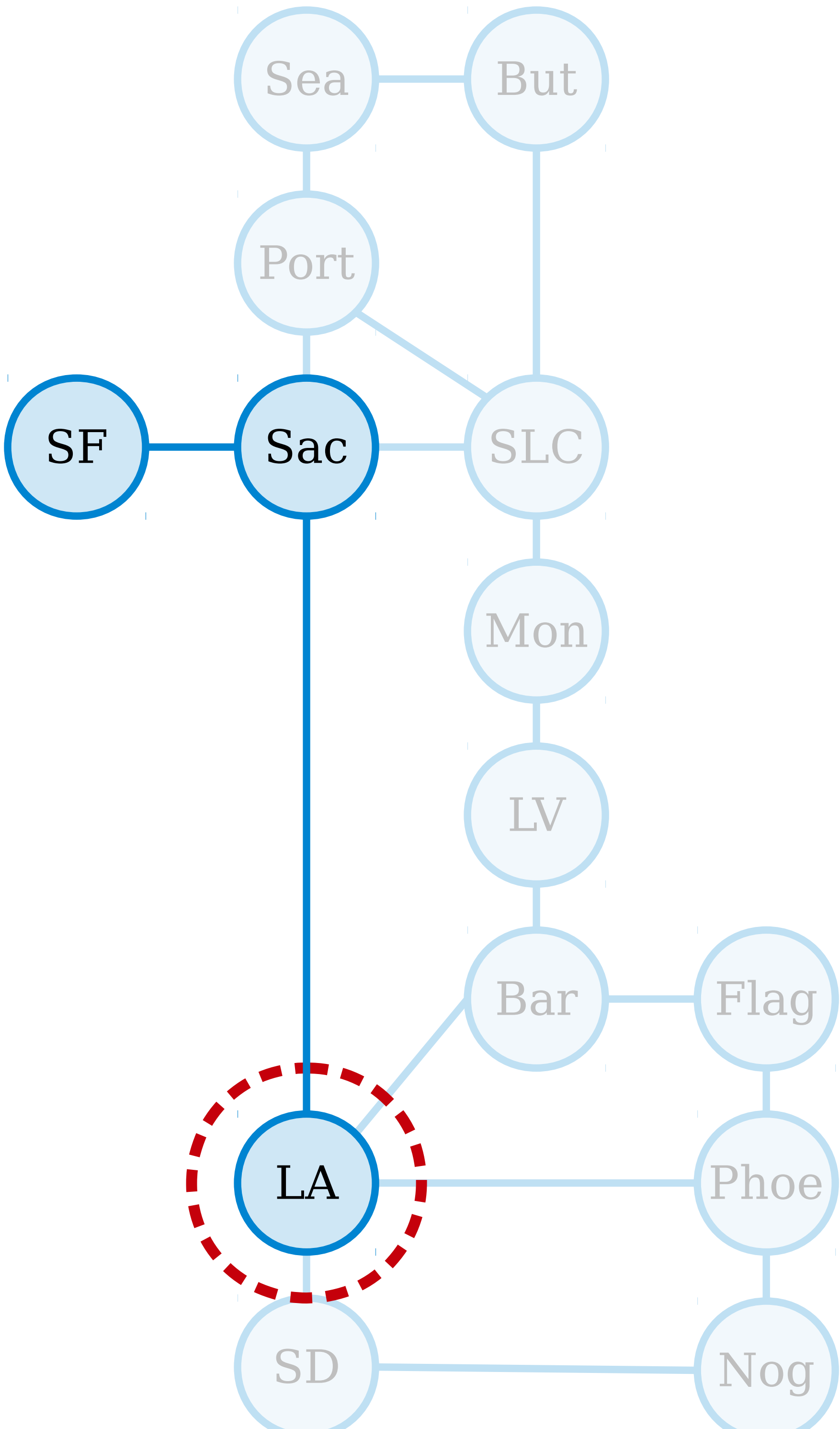
A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

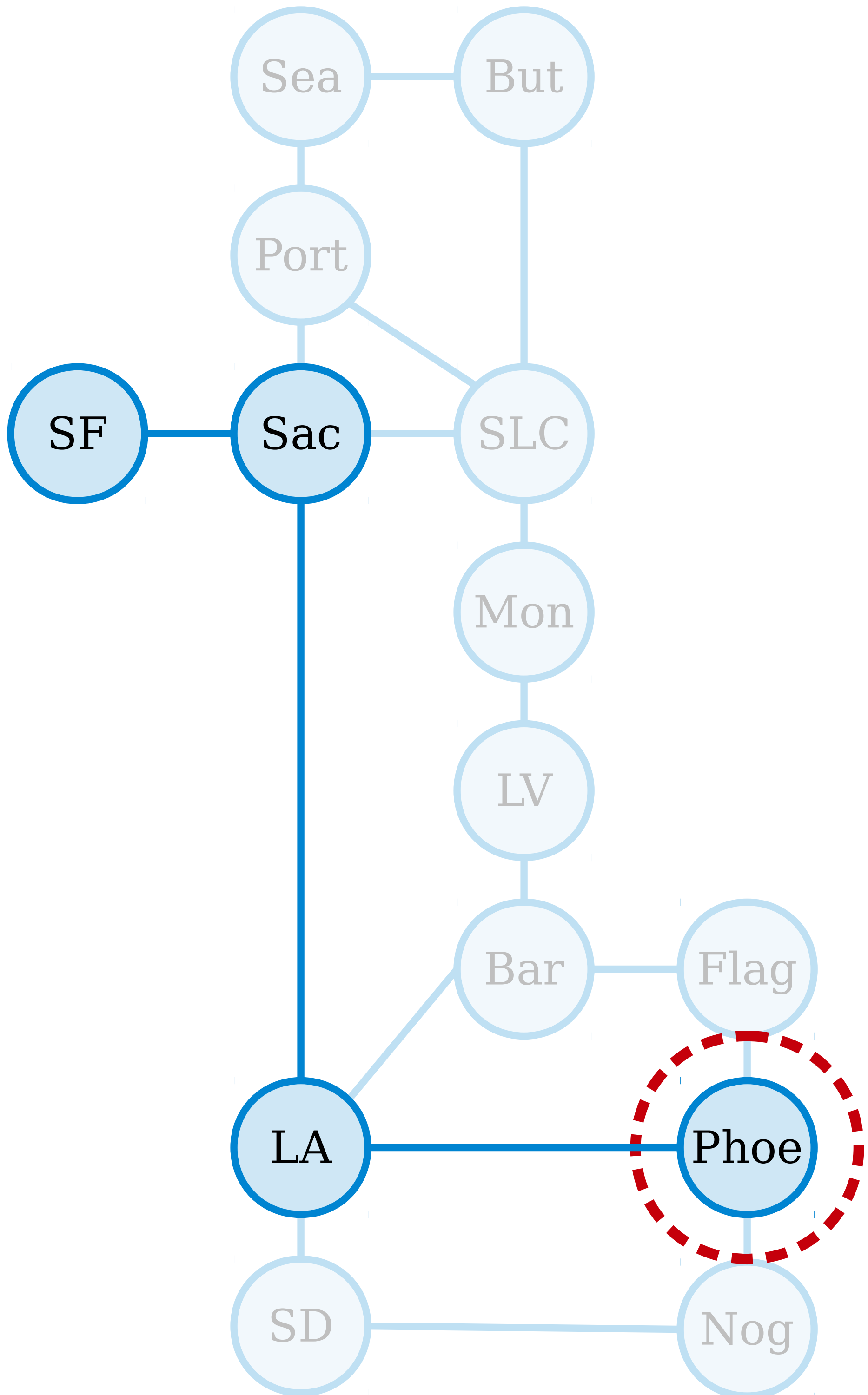
A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

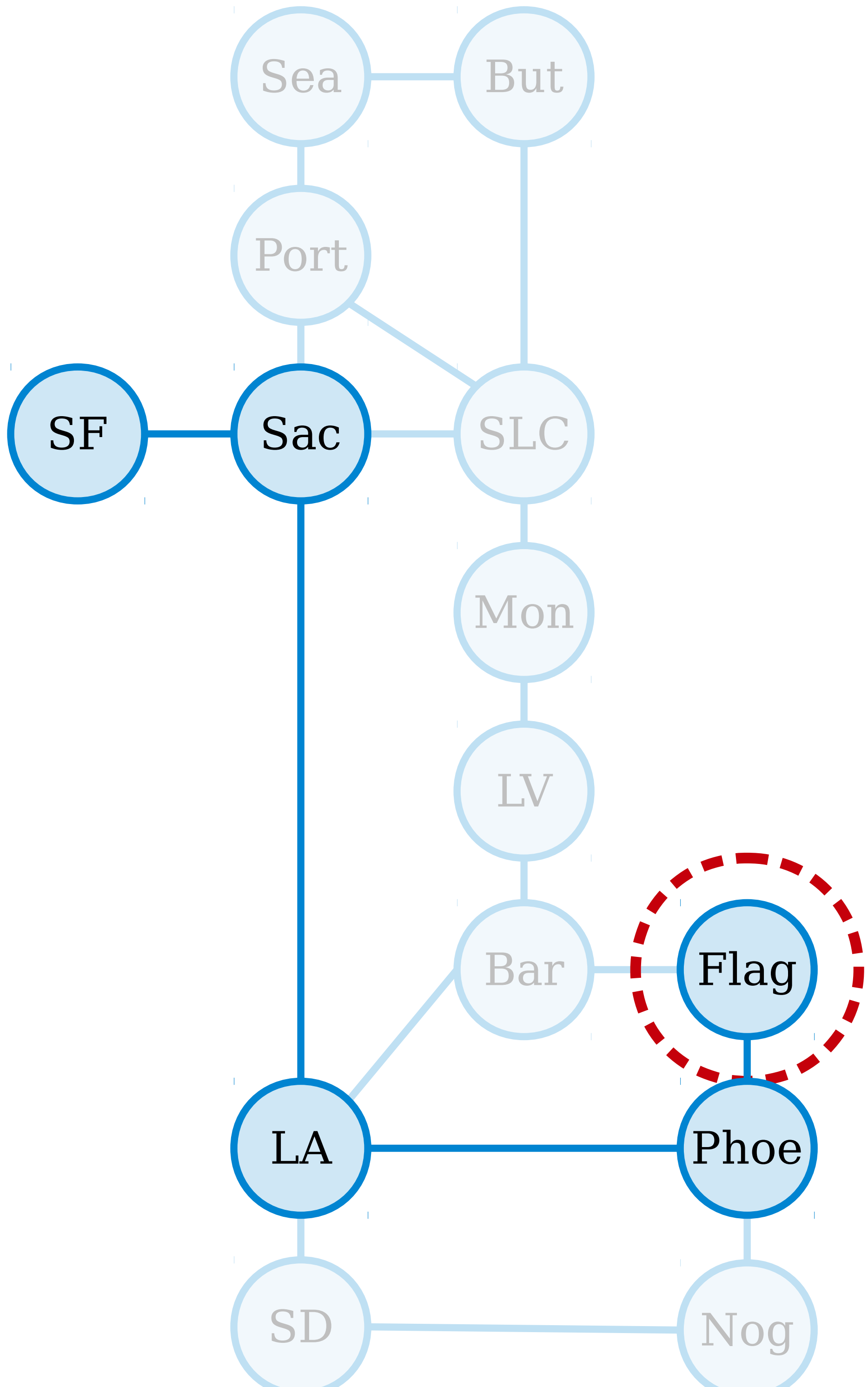
A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

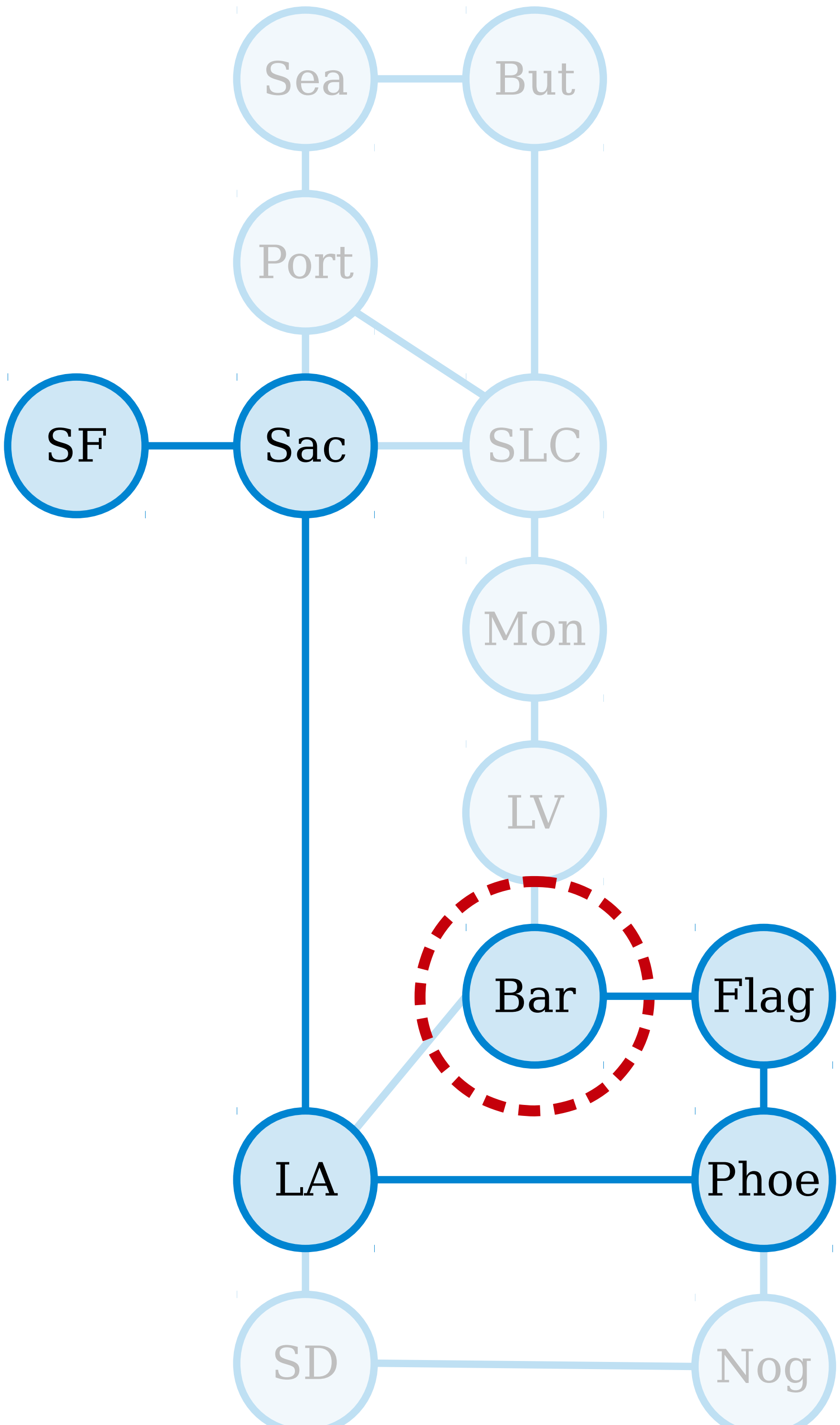
A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

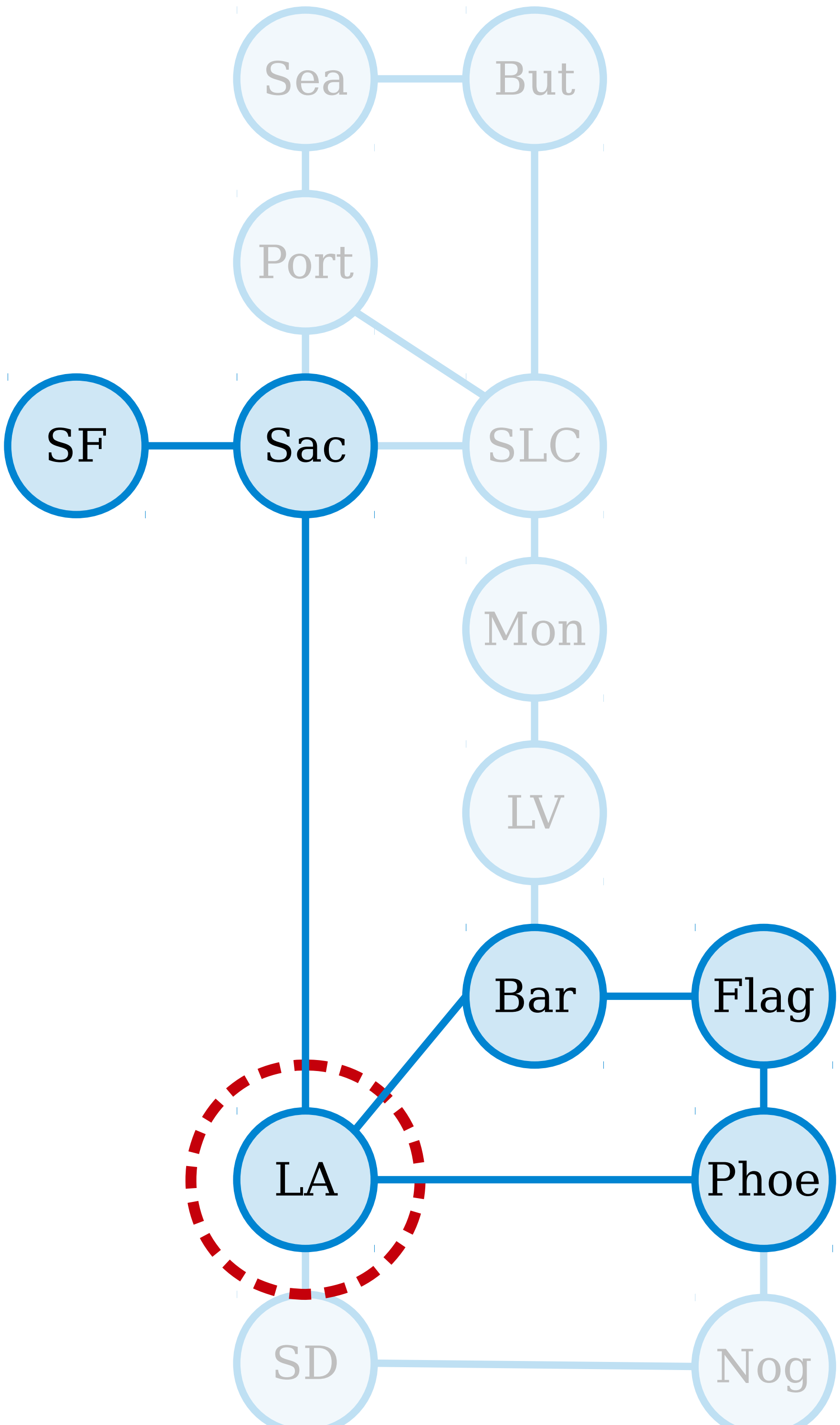
A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

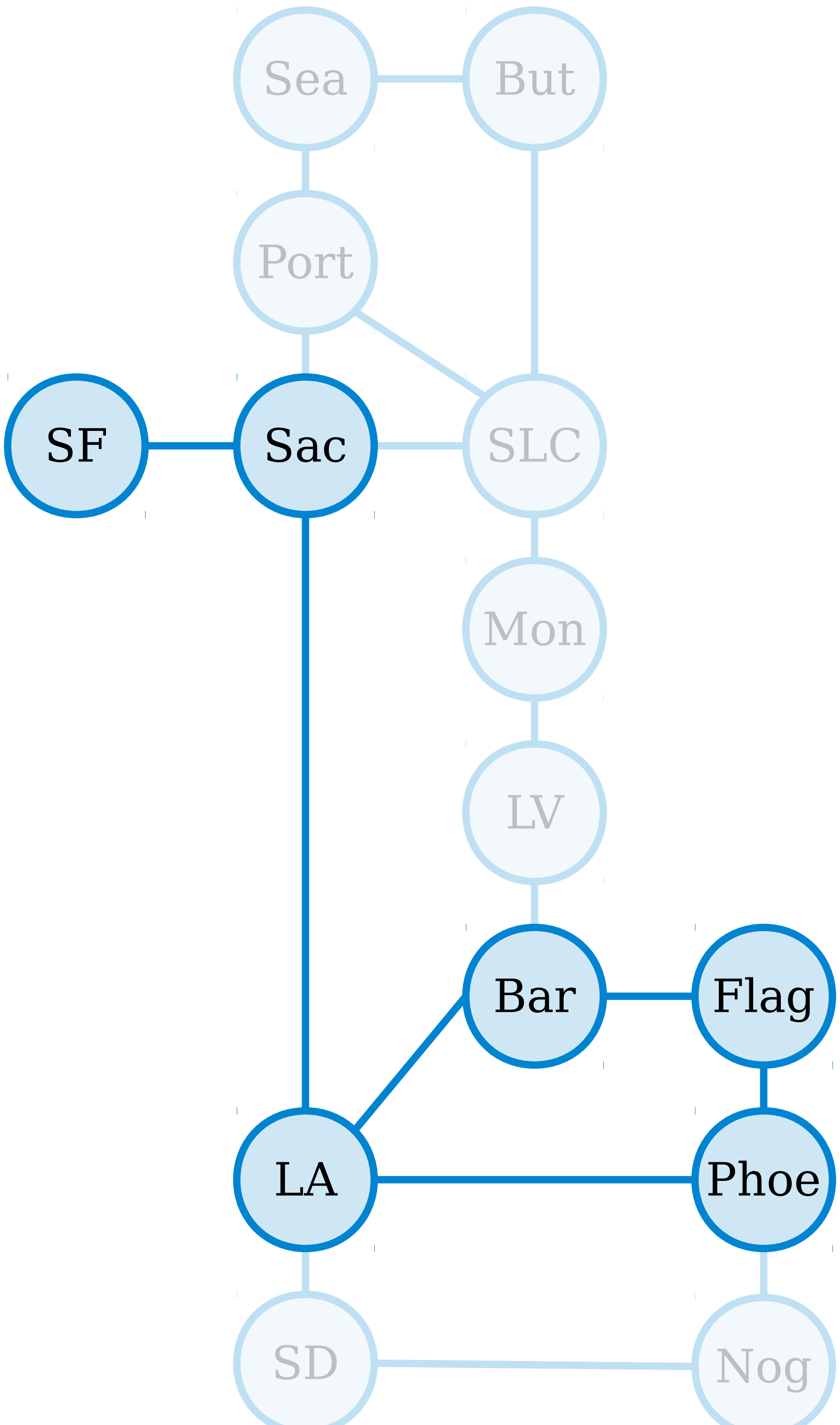
A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

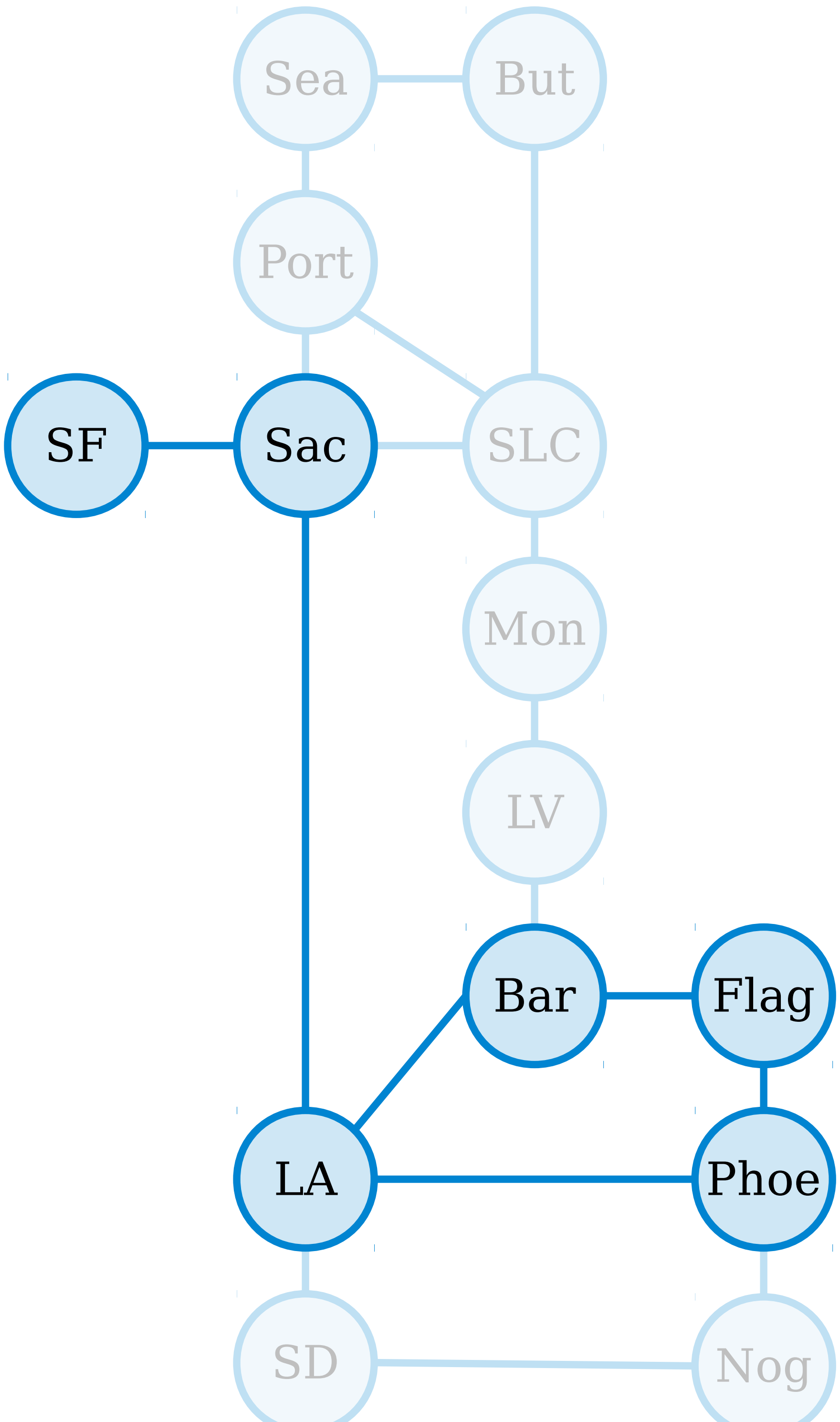
A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)

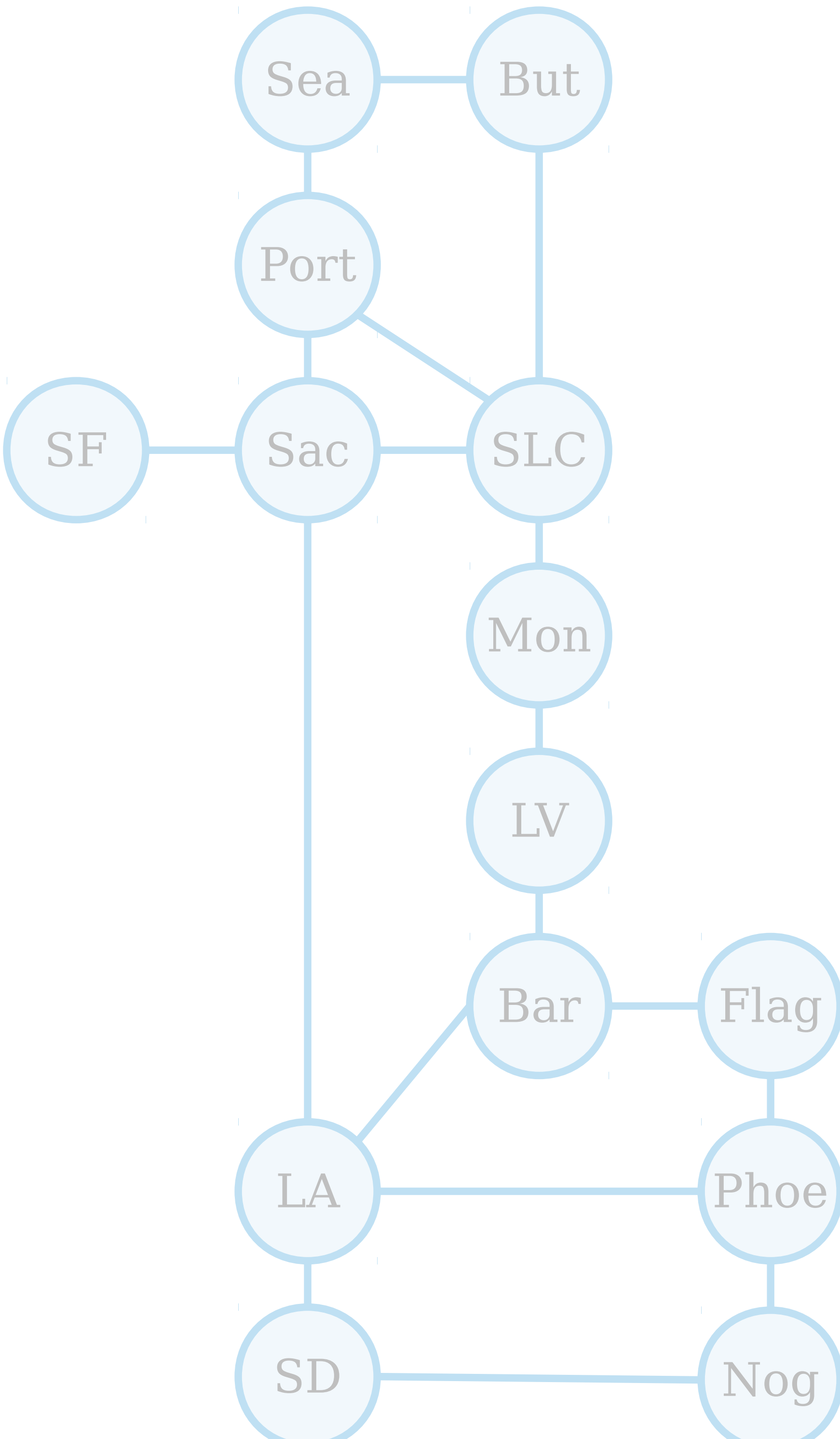


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)

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

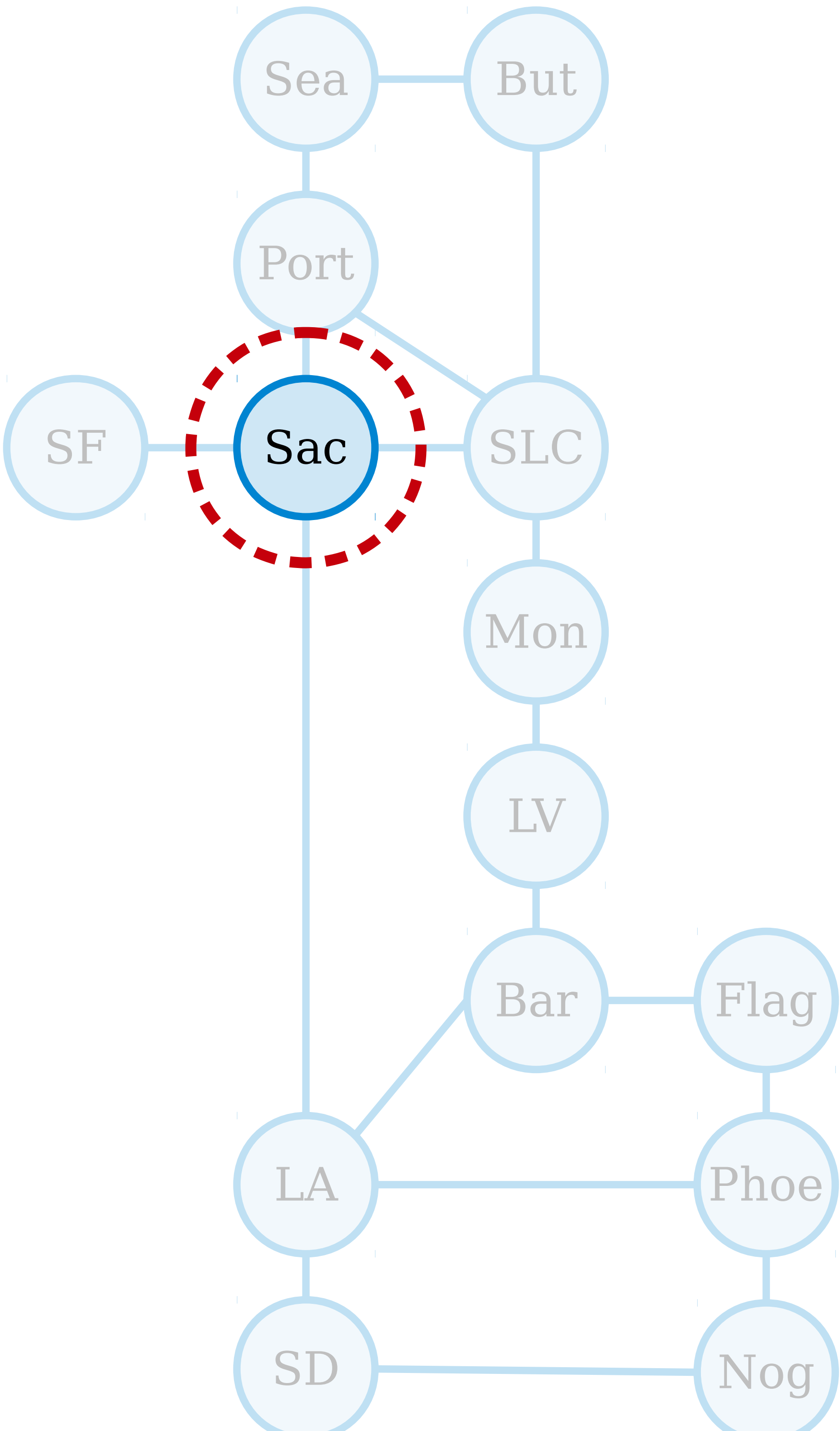


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)

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

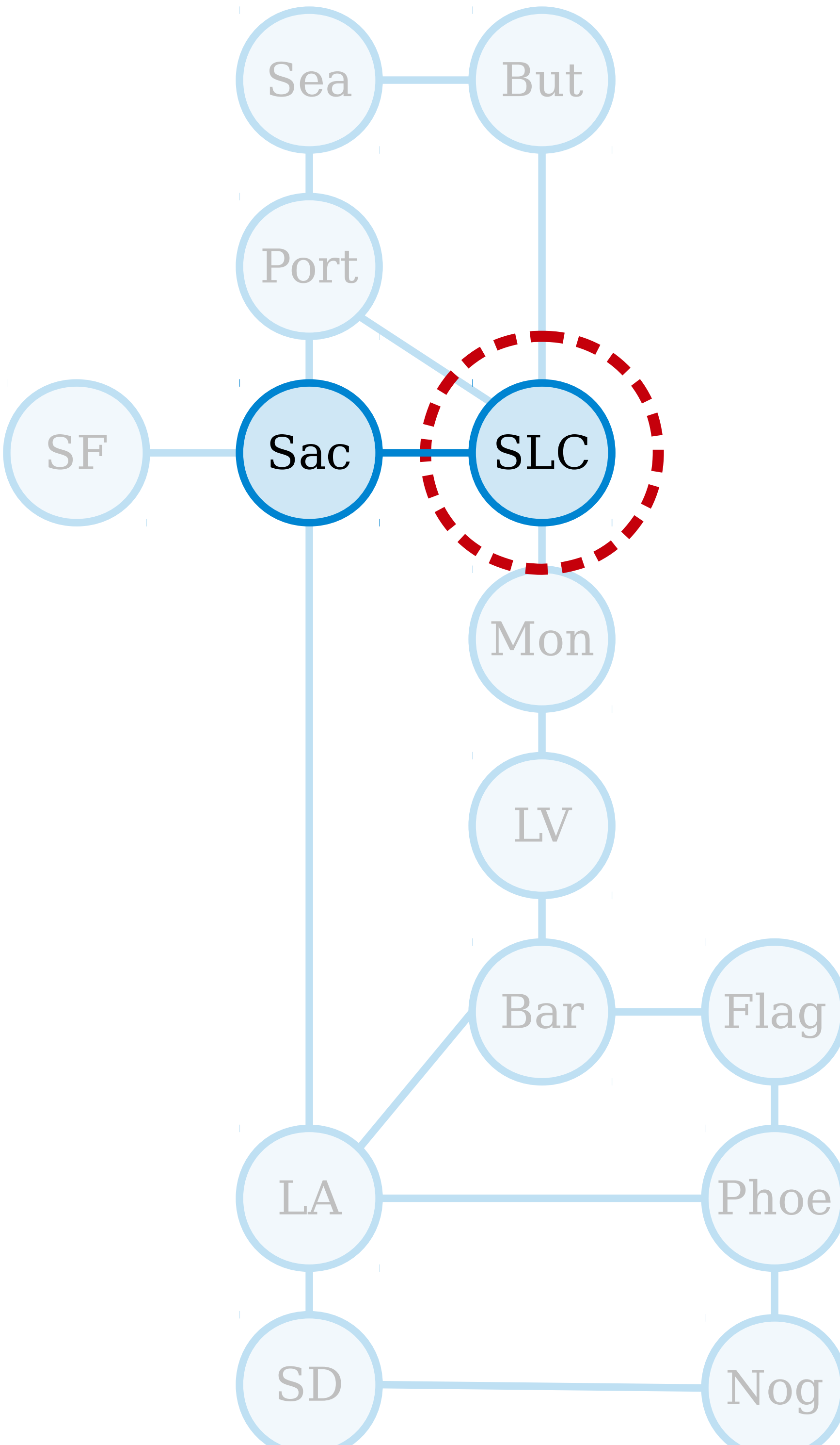


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)

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

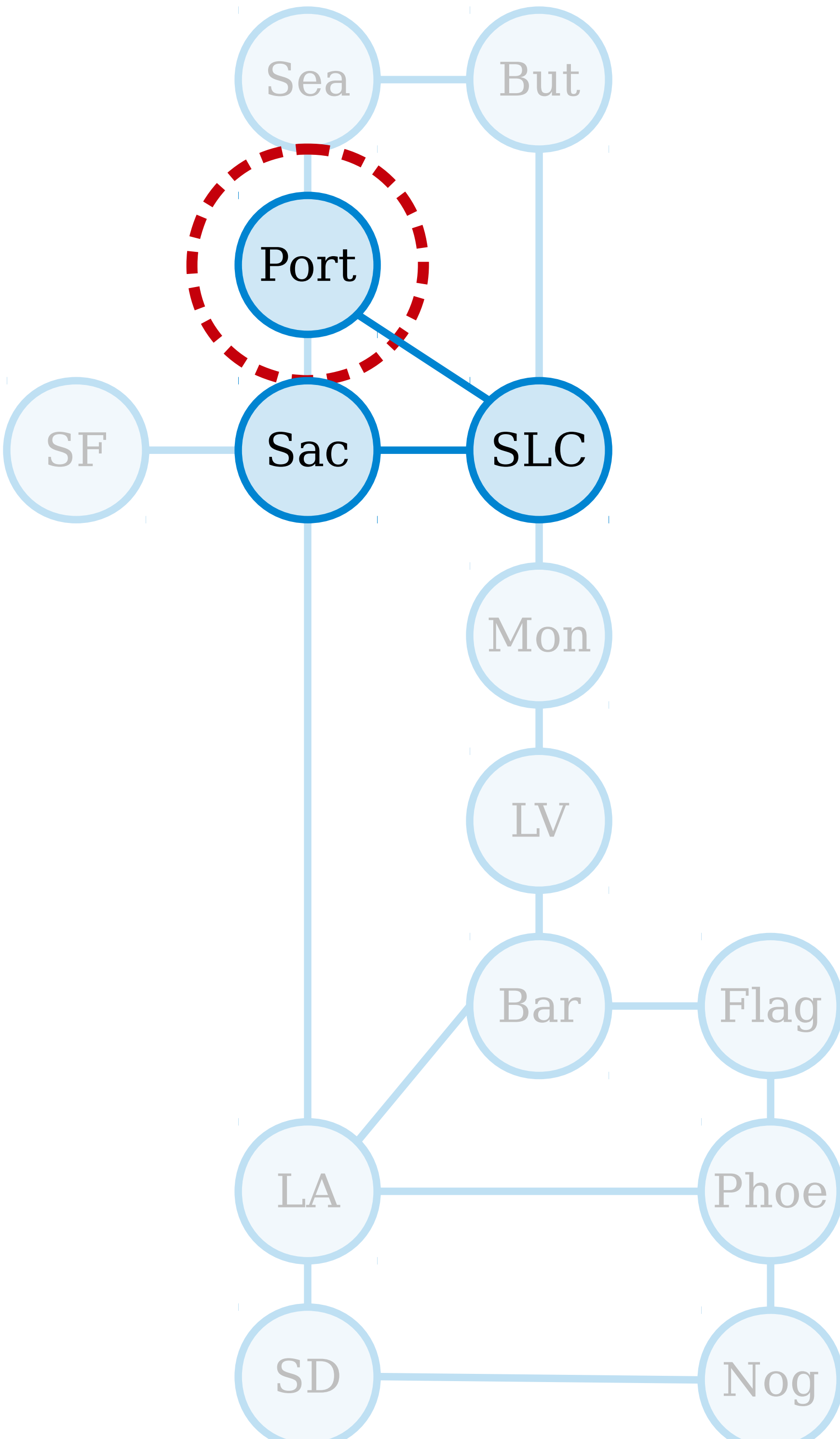


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)

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

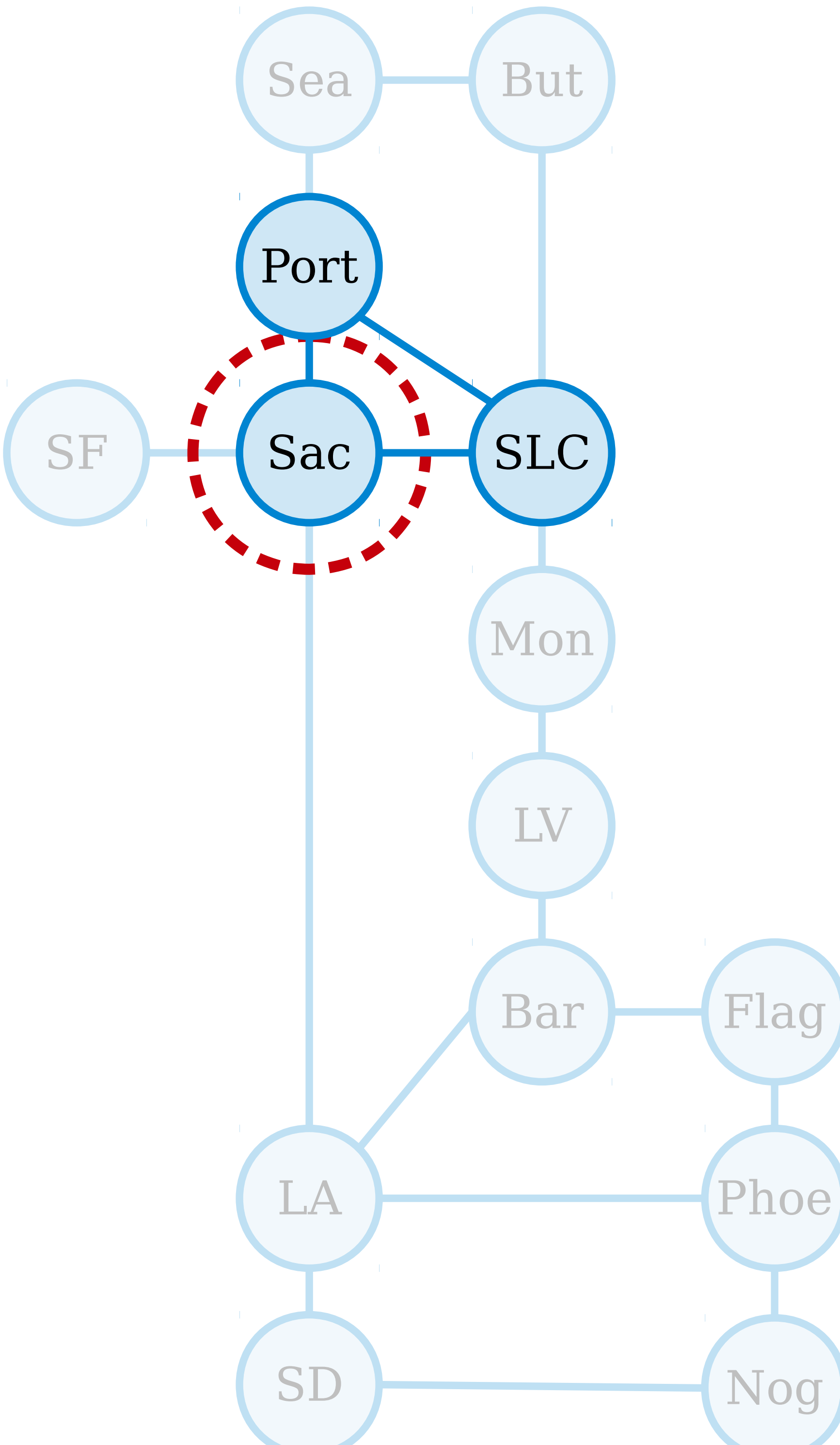


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)

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

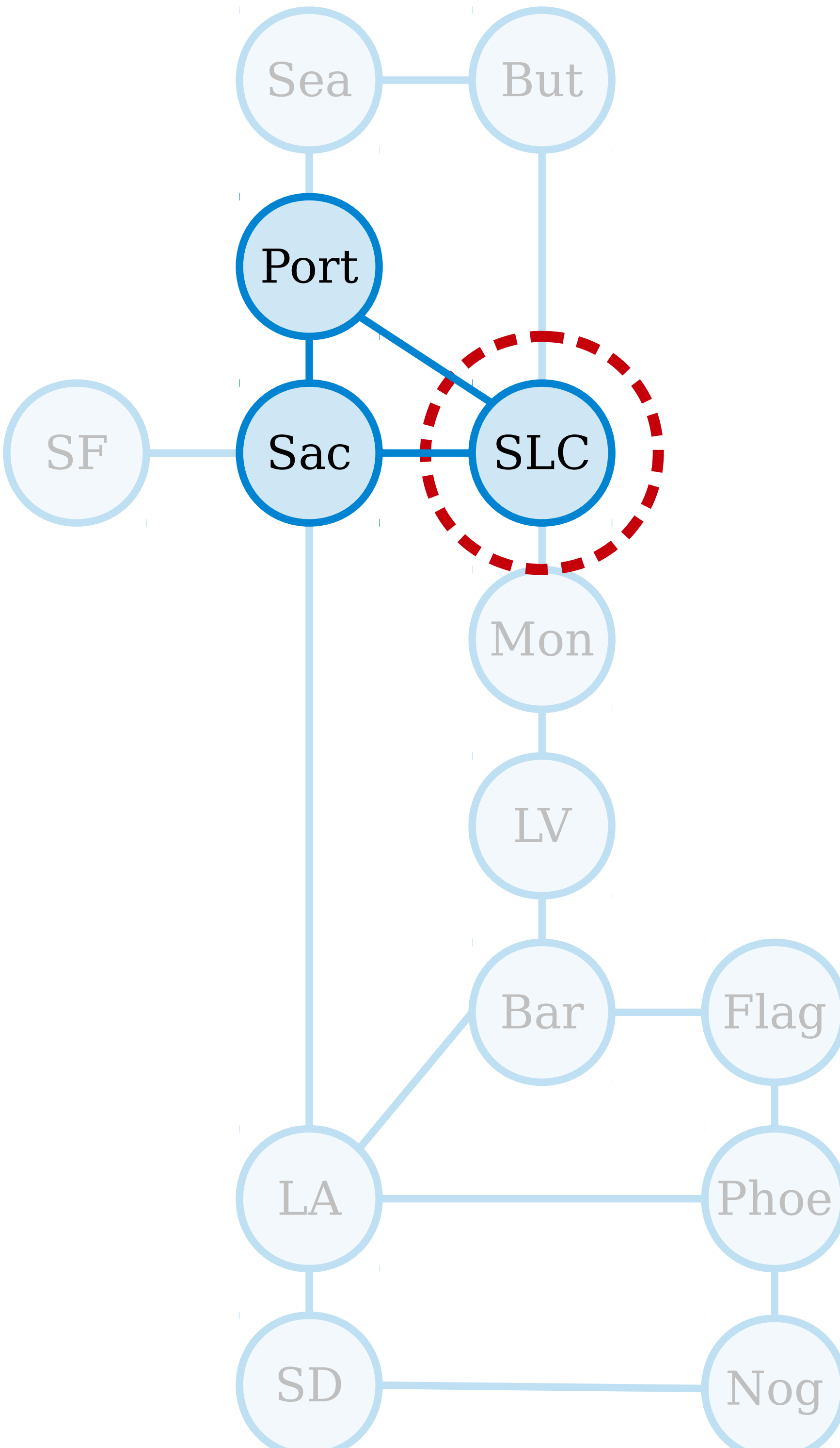


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)

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

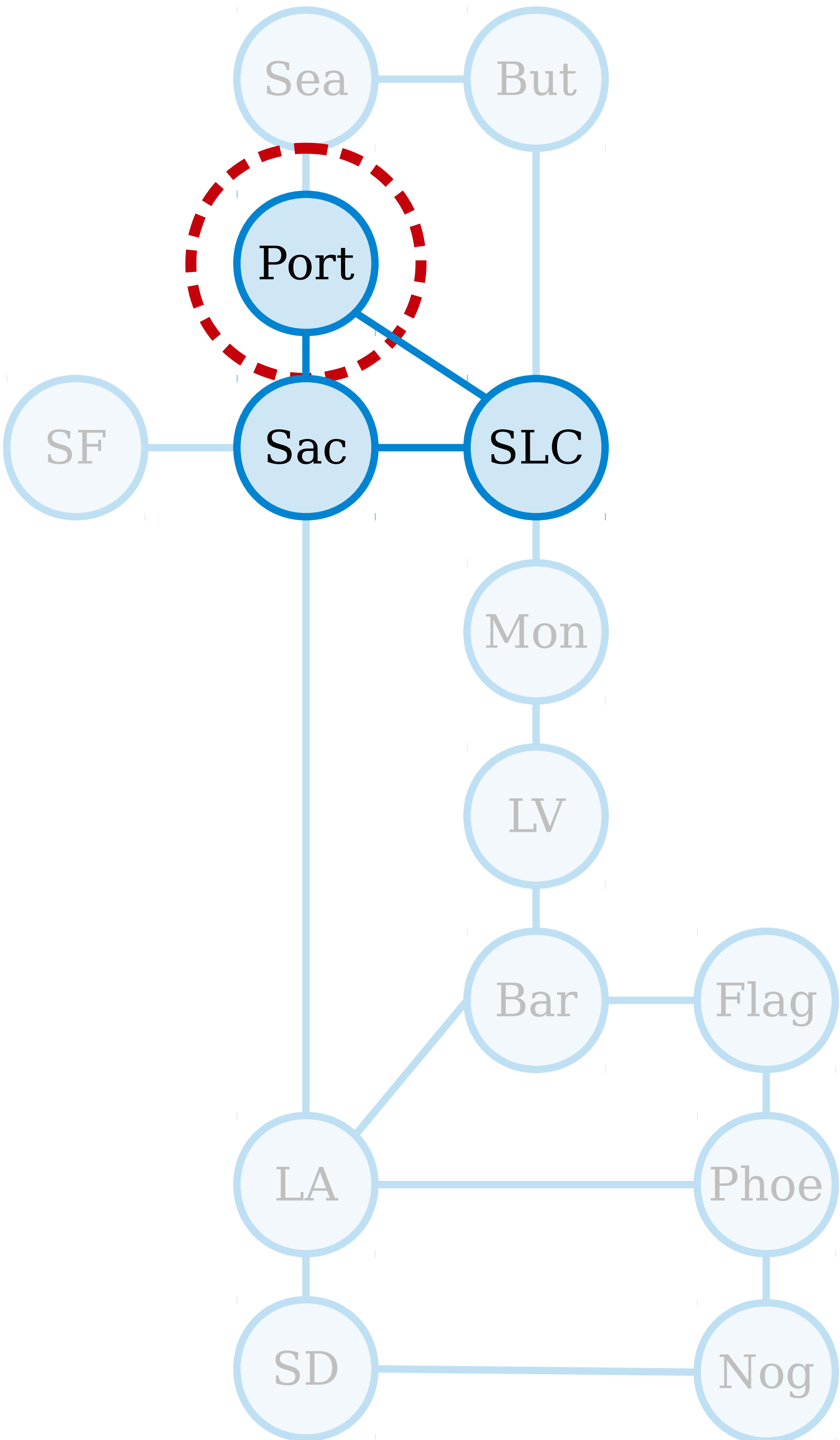


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)

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

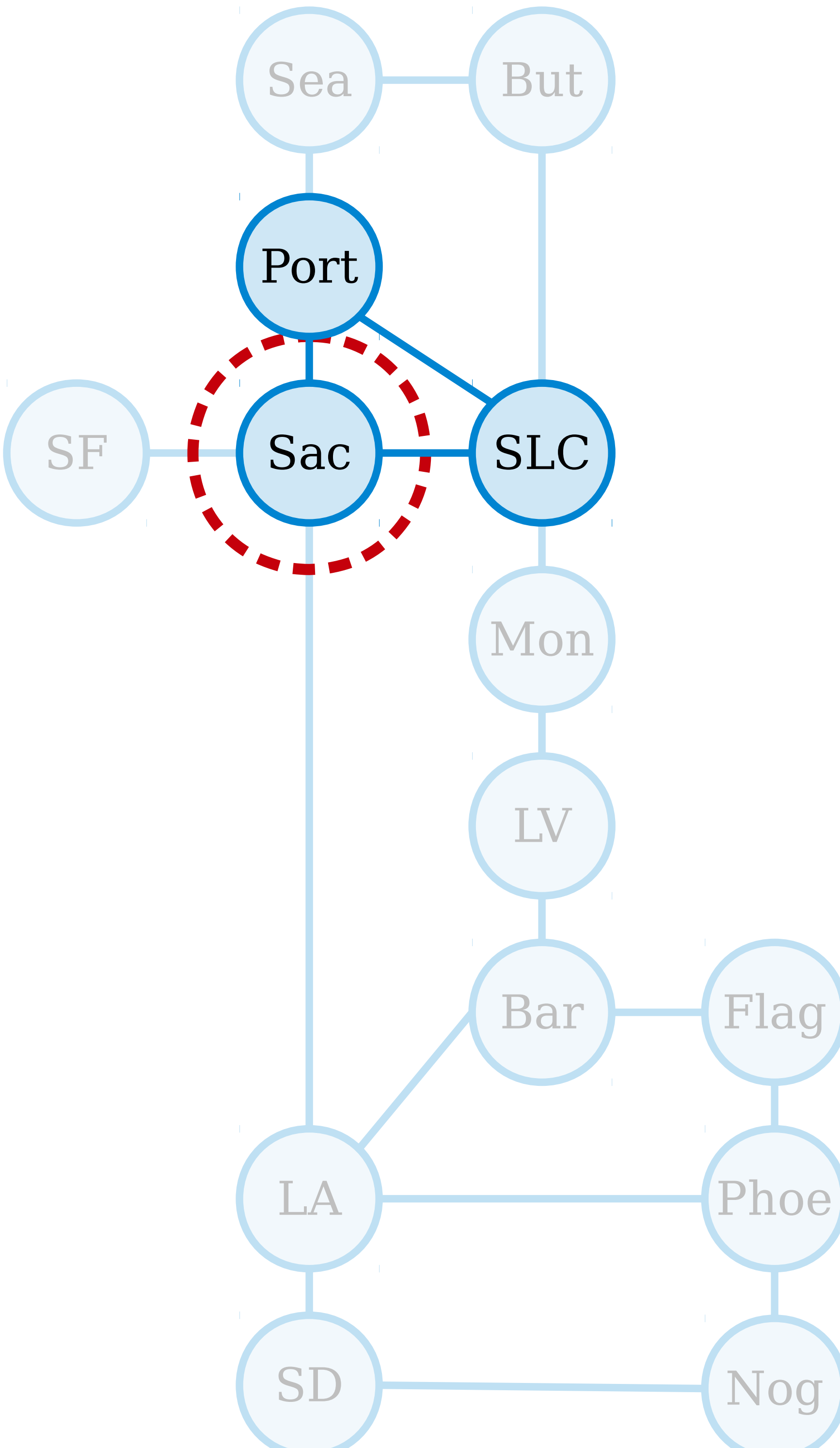


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)

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

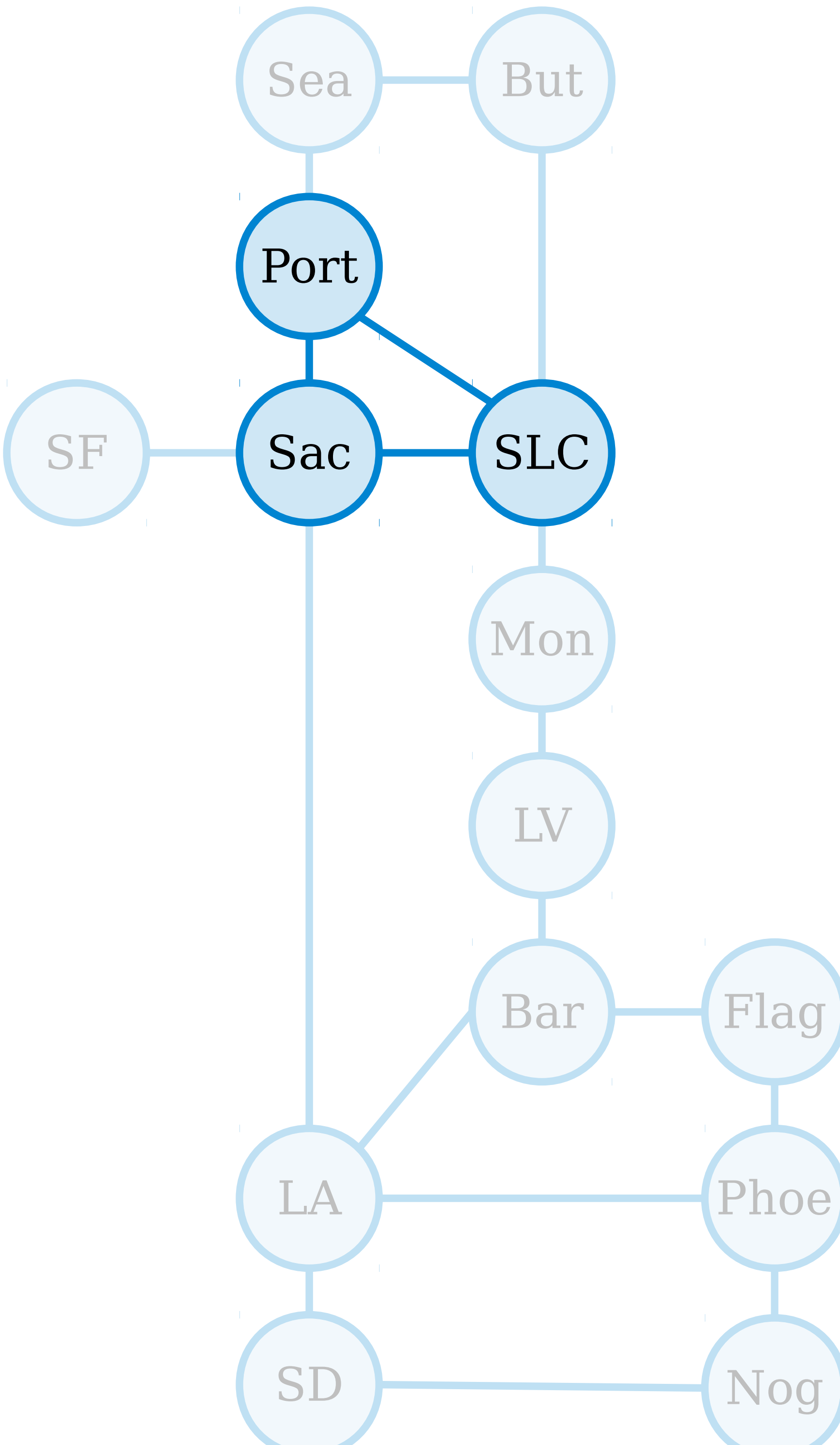


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)

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



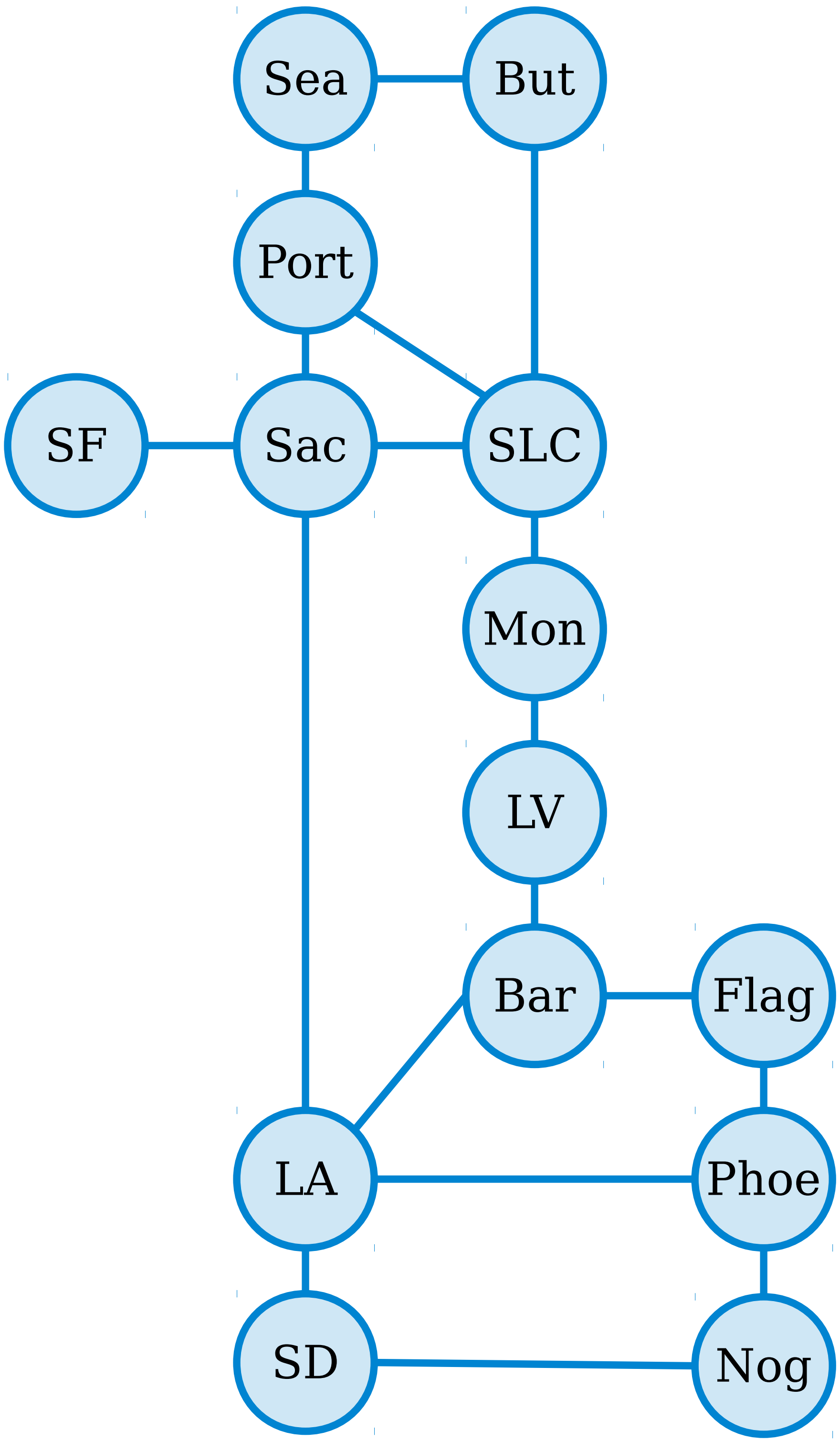
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of a path is the number of edges in it.

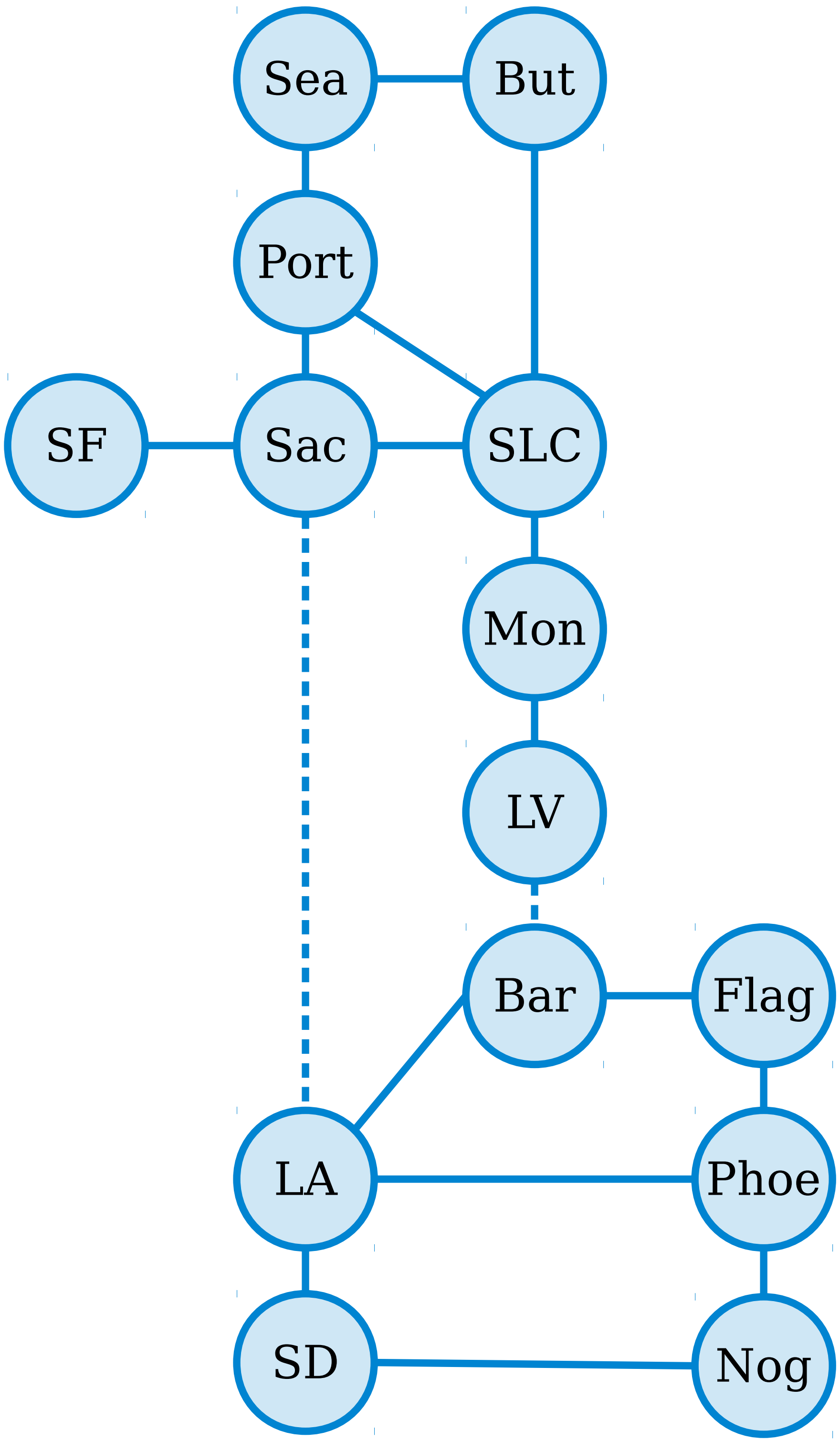
A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)

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

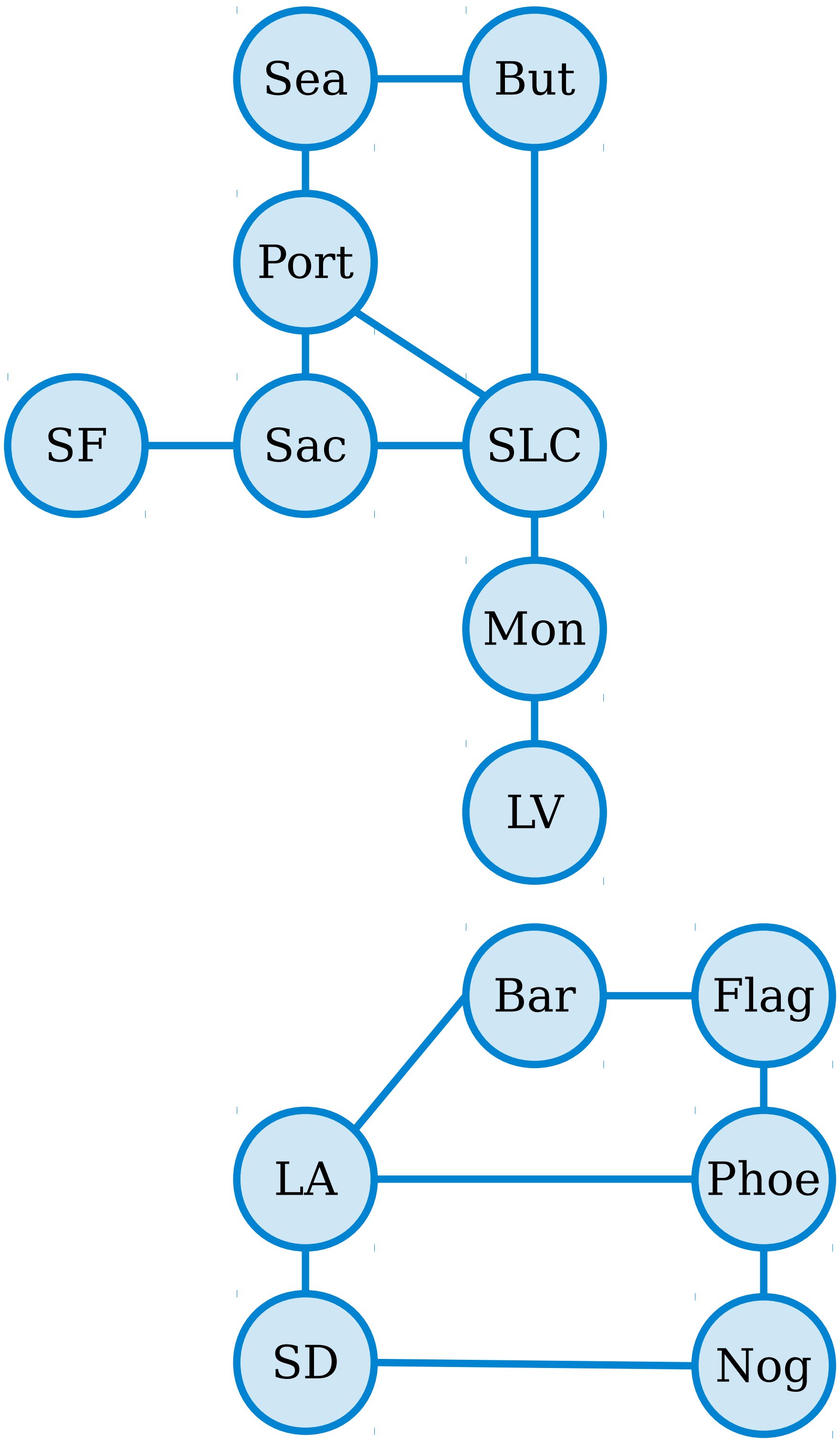
A ***simple cycle*** in a graph is cycle that does not repeat any nodes or edges except the first/last node.



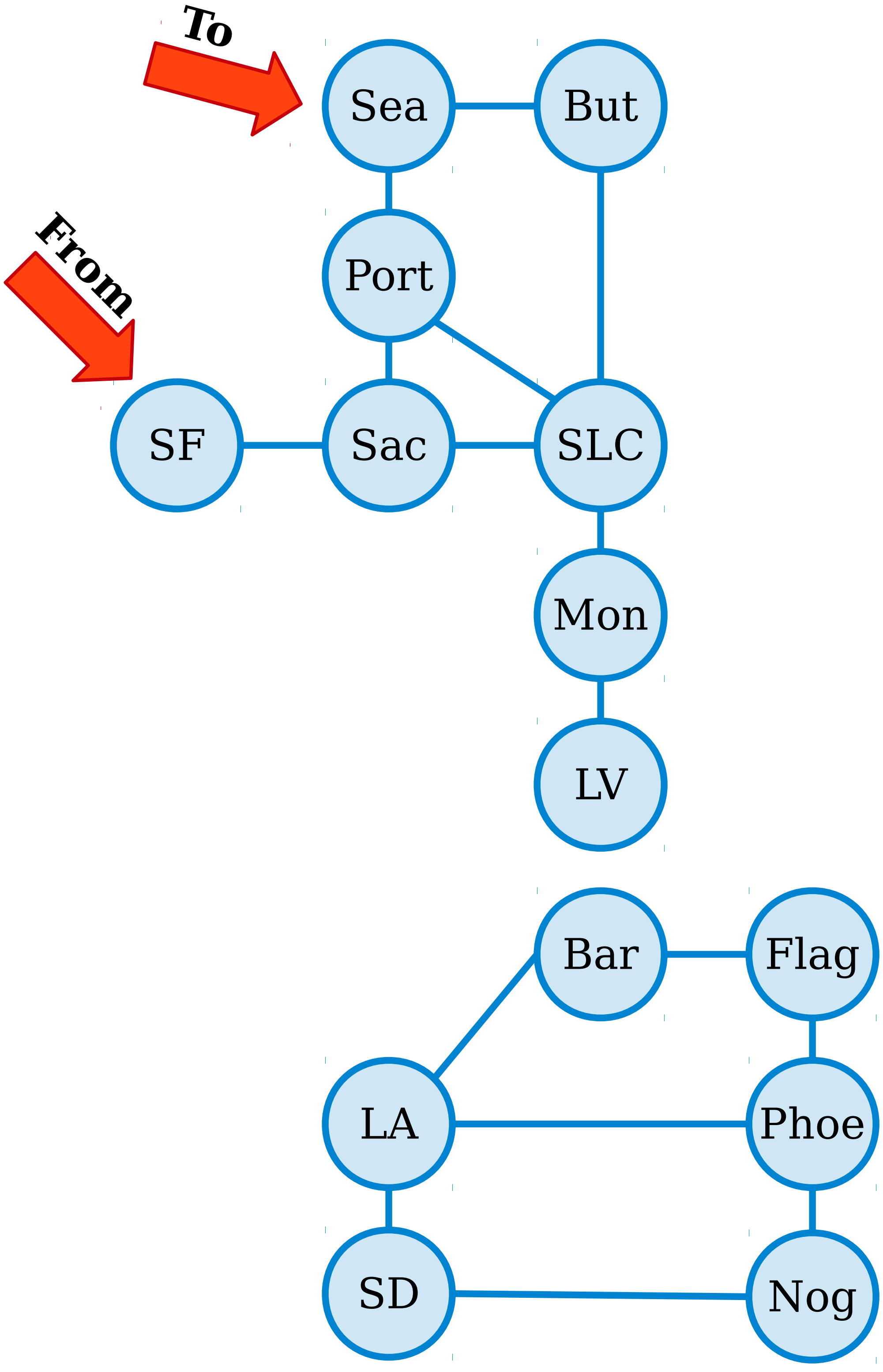
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.



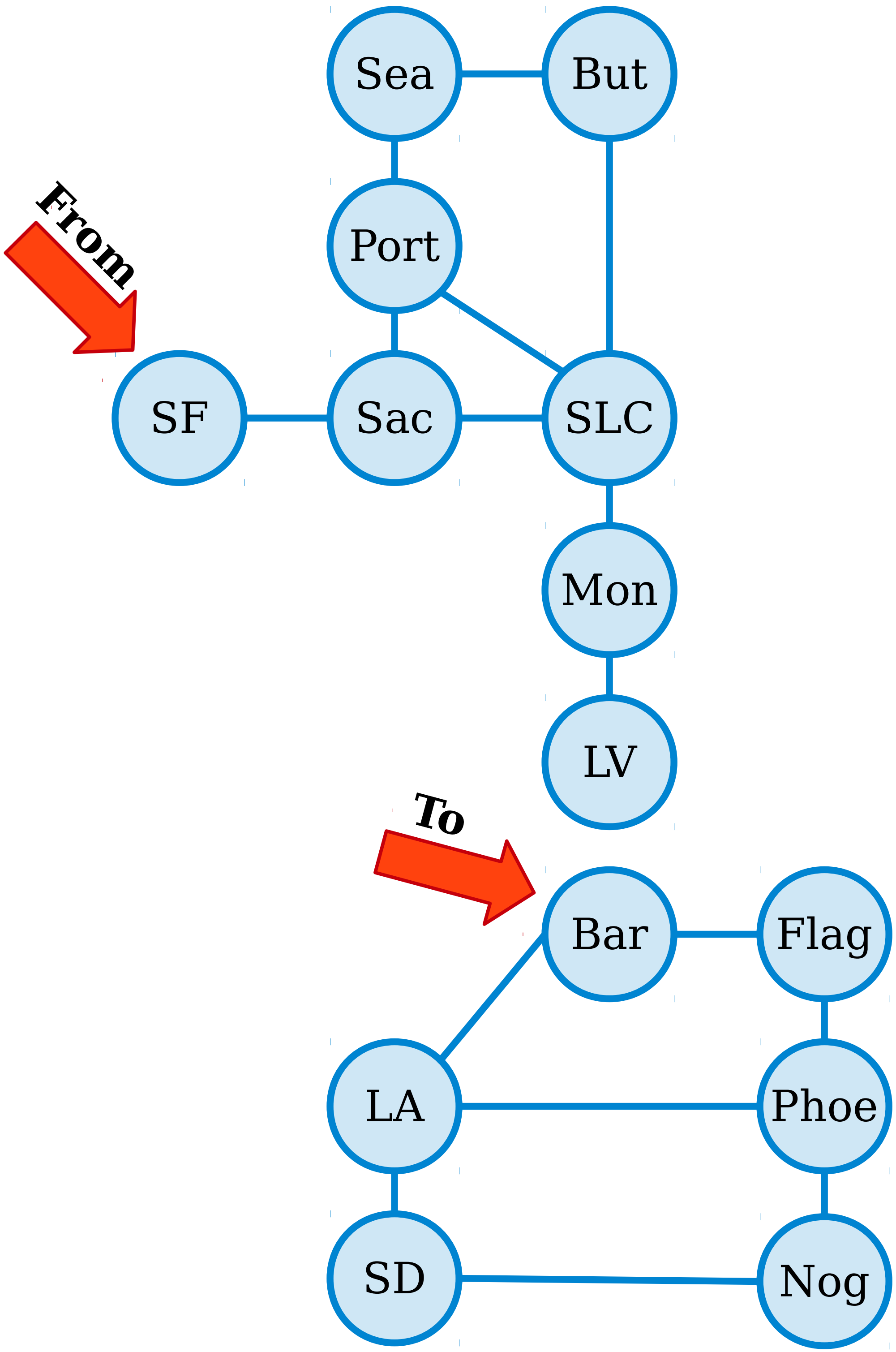
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.



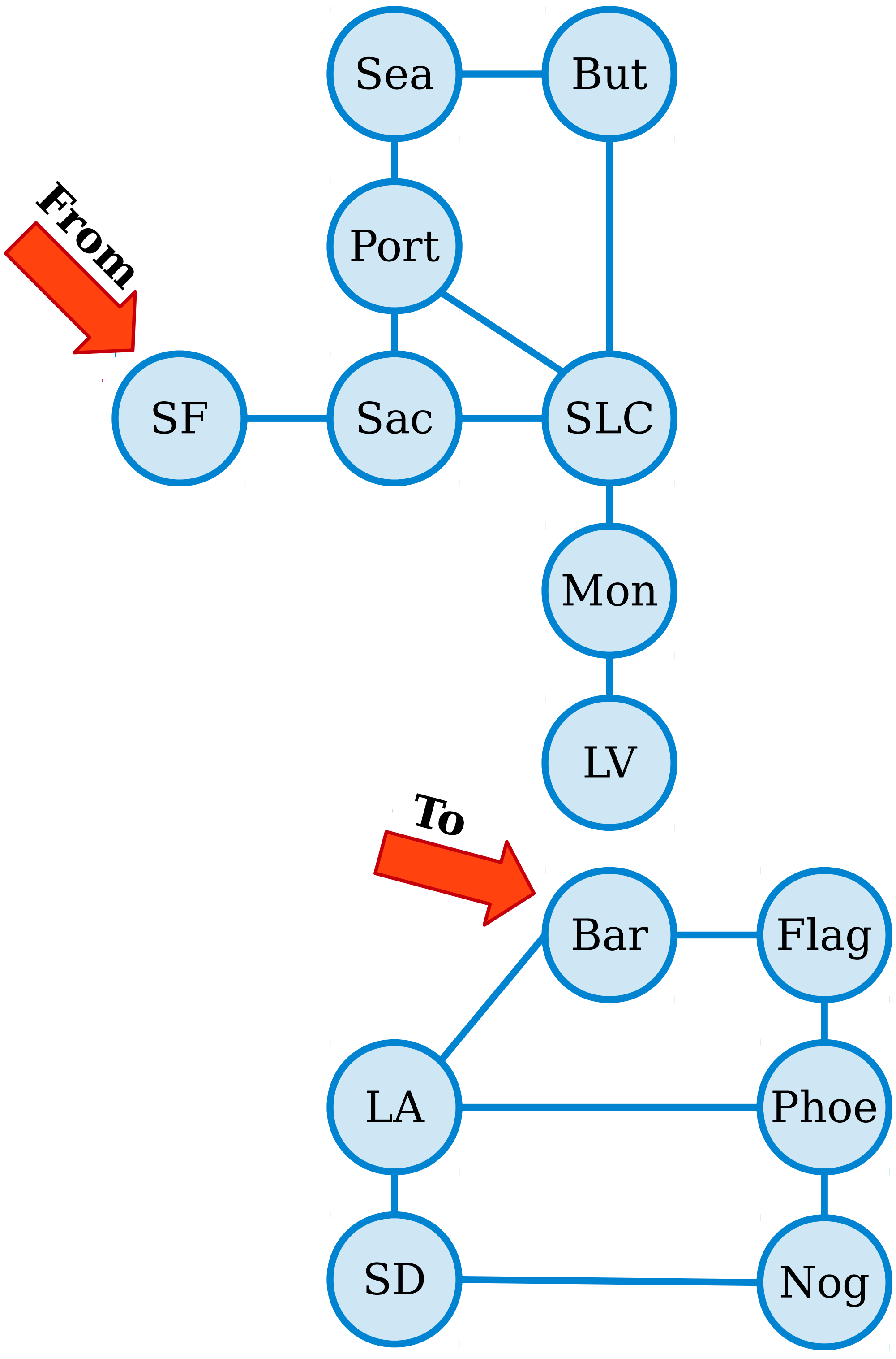
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

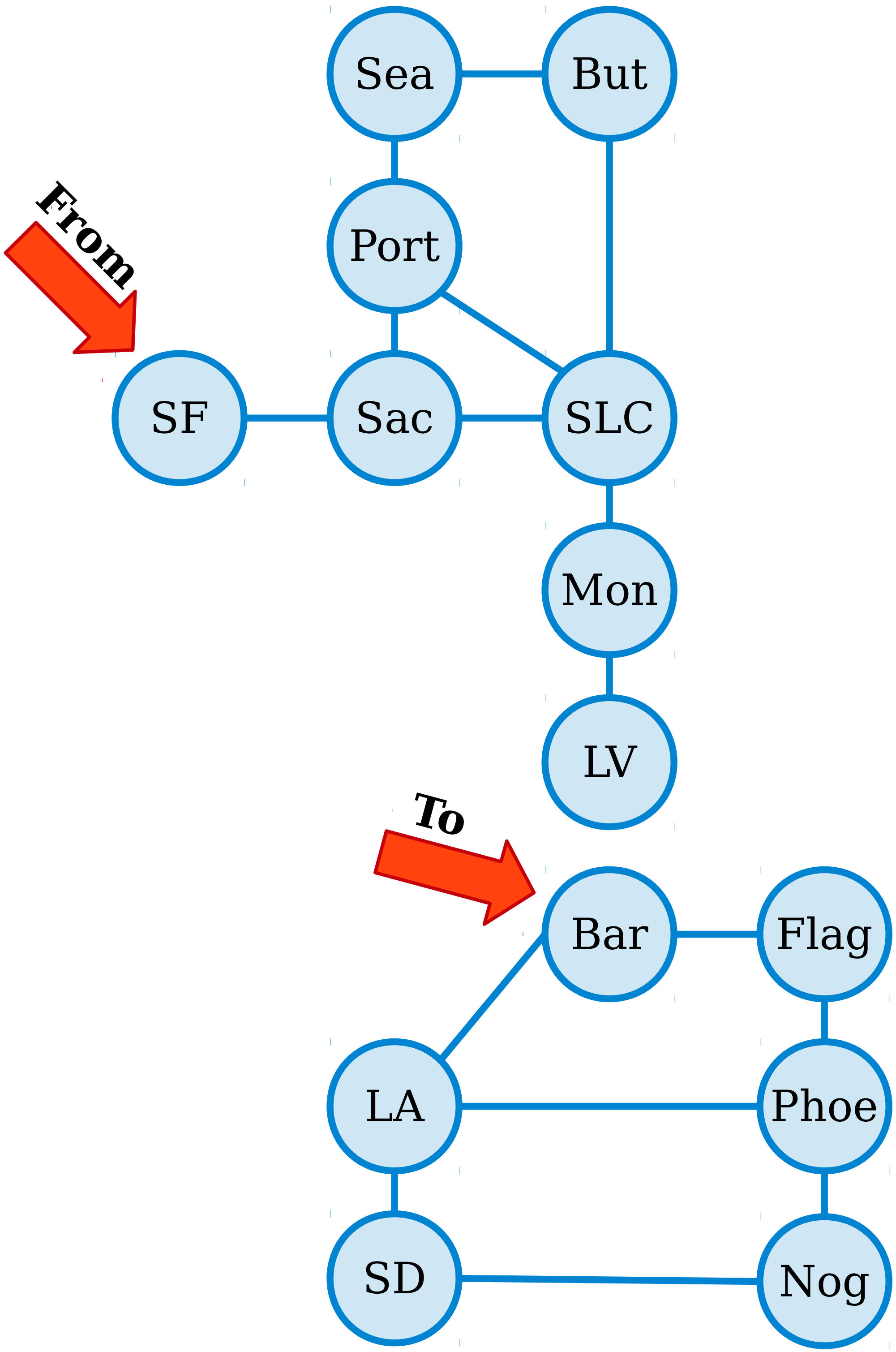


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

Two nodes in a graph are called ***connected*** if there is a path between them



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

Two nodes in a graph are called ***connected*** if there is a path between them

A graph G as a whole is called ***connected*** if all pairs of nodes in G are connected.

Time-Out for Announcements!

Midterm Exam

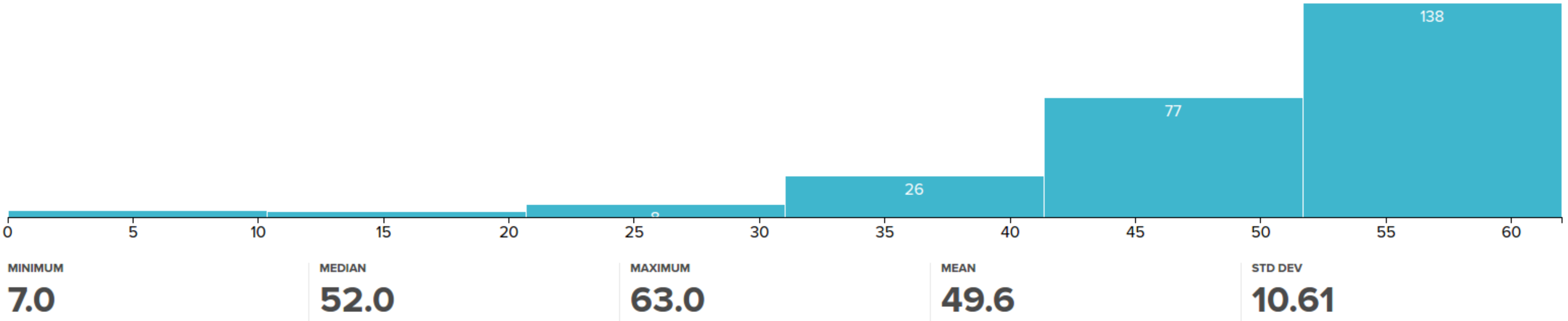
- The first midterm exam is next ***Tuesday, May 2nd***, from ***7:00PM – 10:00PM***. Locations are divvied up by last (family) name:
  - `Abb – Niu`: Go to Hewlett 200.
  - `Nor – Vas`: Go to Hewlett 201.
  - `Vil – Yim`: Go to Hewlett 102.
  - `You – Zuc`: Go to Hewlett 103.
- You're responsible for Lectures 00 – 05 and topics covered in PS1 – PS2. Later lectures and problem sets won't be tested.
- The exam is closed-book, closed-computer, and limited-note. You can bring a double-sided, 8.5" × 11" sheet of notes with you to the exam, decorated however you'd like.

Midterm Practice

- We've just uploaded
  - the practice midterm from last night, with solutions;
  - another practice midterm exam;
  - solutions to Extra Practice Problems 1; and
  - another set of review problems, EPP2.
- We'll release EPP3 on Friday along with solutions to EPP2 and the additional practice midterm exam.
- Need more practice? Let us know!

Problem Set Two

- Problem Set Two has been graded and feedback has been release.
  - ***Please read over your feedback*** – the skills tested on this problem set are the same skills tested on the midterm.
- Score distribution is shown here:



Problem Set Two: Common Mistakes

“Someone has two pet kittens
and no other pets.”

“Someone has two pet kittens
and no other pets.”

$$
\begin{aligned}
&\exists p.\ (Person(p)\ \land \\
&\quad \exists k_1.\ (Kitten(k_1)\ \land\ HasPet(p, k_1)\ \land \\
&\quad\quad \exists k_2.\ (Kitten(k_2)\ \land\ HasPet(p, k_2)) \\
&\quad) \\
&)
\end{aligned}
$$

"Someone has two pet kittens
and no other pets."

$$
\begin{aligned}
&\exists p.\ (\mathit{Person}(p) \land \\
&\quad \exists k_1.\ (\mathit{Kitten}(k_1) \land \mathit{HasPet}(p, k_1) \land \\
&\qquad \exists k_2.\ (\mathit{Kitten}(k_2) \land \mathit{HasPet}(p, k_2)) \\
&\quad) \\
&)
\end{aligned}
$$

“Someone has two pet kittens
and no other pets.”

$$
\begin{aligned}
&\exists p.\ (Person(p)\ \land \\
&\quad \exists k_1.\ (Kitten(k_1)\ \land\ HasPet(p, k_1)\ \land \\
&\qquad \exists k_2.\ (Kitten(k_2)\ \land\ k_1 \neq k_2\ \land\ HasPet(p, k_2)) \\
&\quad) \\
&)
\end{aligned}
$$

“Someone has two pet kittens
and no other pets.”

$\exists p.\ (Person(p)\ \land$
$\quad \exists k_1.\ (Kitten(k_1)\ \land\ HasPet(p, k_1)\ \land$
$\quad\quad \exists k_2.\ (Kitten(k_2)\ \land\ k_1 \neq k_2\ \land\ HasPet(p, k_2))$
$\quad)$
$)$

“Someone has two pet kittens
and no other pets.”

$$
\begin{array}{l}
\exists p.\ (Person(p)\ \land \\
\quad \exists k_1.\ (Kitten(k_1)\ \land\ HasPet(p, k_1)\ \land \\
\quad\quad \exists k_2.\ (Kitten(k_2)\ \land\ k_1 \neq k_2\ \land\ HasPet(p, k_2)\ \land \\
\quad\quad\quad \forall q.\ (HasPet(p, q) \to q = k_1 \lor q = k_2) \\
\quad\quad) \\
\quad) \\
)
\end{array}
$$

“Someone has two pet kittens
and no other pets.”

$$
\begin{aligned}
&\exists p.\ (Person(p) \land \\
&\quad \exists k_1.\ (Kitten(k_1) \land HasPet(p, k_1) \land \\
&\quad\quad \exists k_2.\ (Kitten(k_2) \land k_1 \neq k_2 \land HasPet(p, k_2) \land \\
&\quad\quad\quad \forall q.\ (HasPet(p, q) \to q = k_1 \lor q = k_2) \\
&\quad\quad) \\
&\quad) \\
&)
\end{aligned}
$$

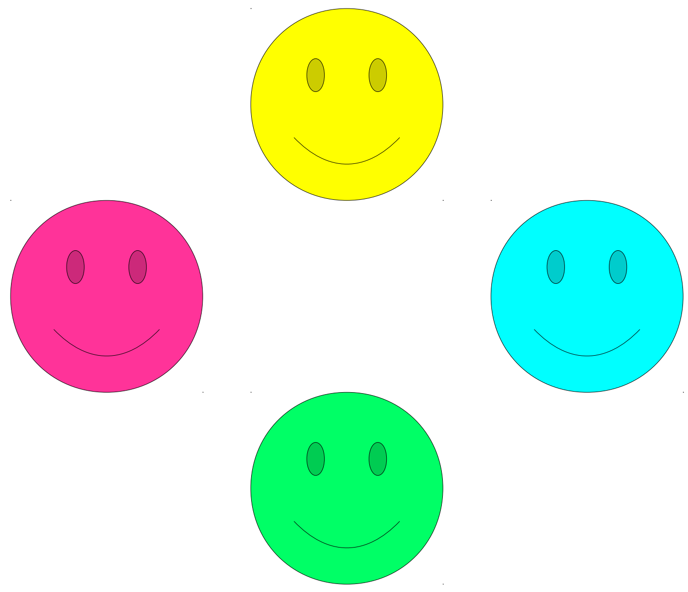
1. Remember that multiple quantifiers can range over the same objects!

2. To express “and nothing else does,” show that anything matching the property must be equal to something you already know.

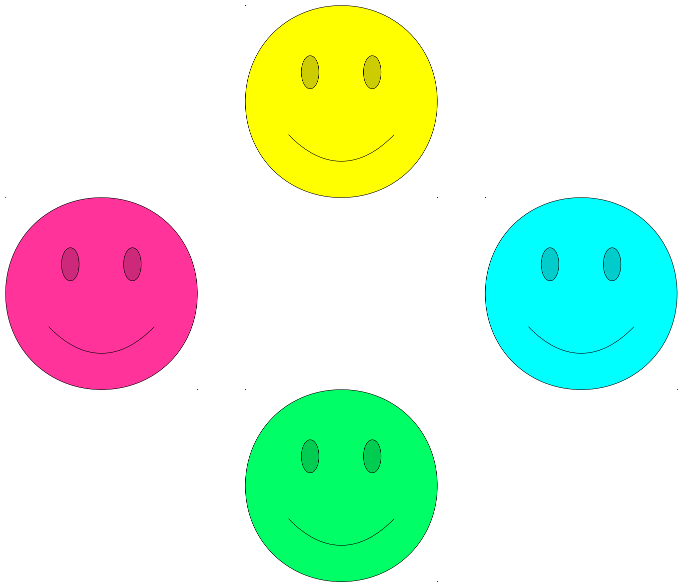
Problem Set Three

- The Problem Set Three checkpoint has been graded.
  - ***Please, please, please review your feedback!*** That problem was tricky and a lot of people had a lot of trouble with it.
- Remaining problems are due on Friday. Be strategic about taking late days.

PS3 Checkpoint: Common Mistakes

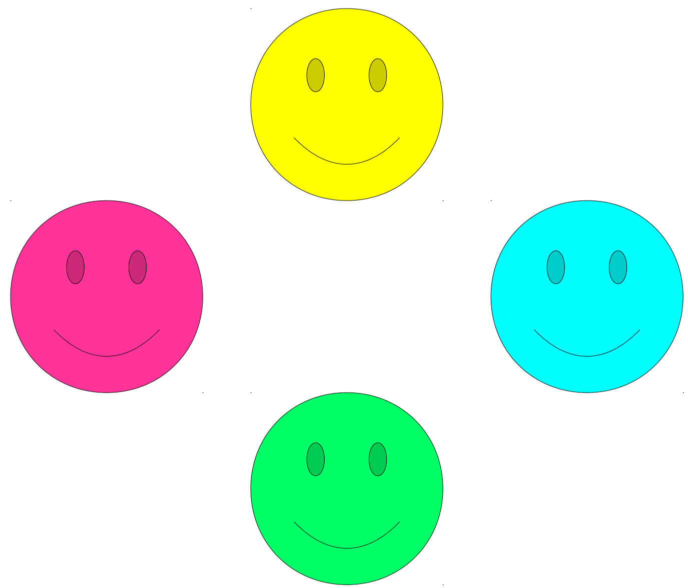


Is this relation symmetric?



Is this relation symmetric?

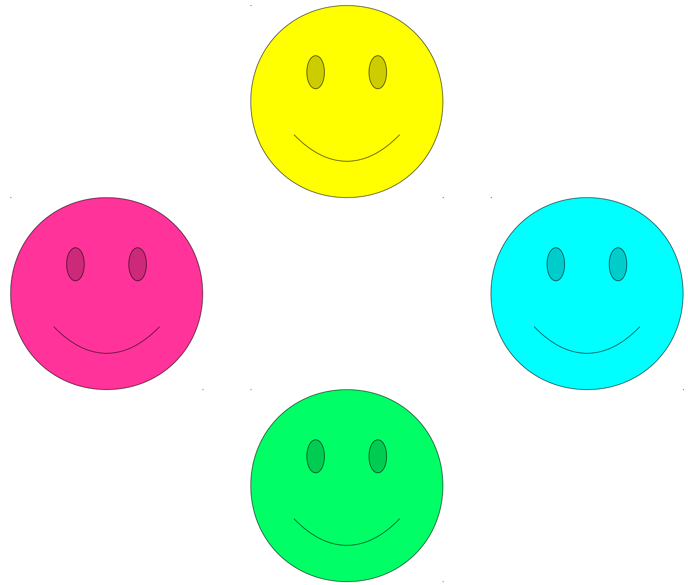
$$\forall a \in A.\ \forall b \in A.\ (aRb \rightarrow bRa)$$



Is this relation symmetric?

$$\forall a \in A.\ \forall b \in A.\ (aRb \rightarrow bRa)$$

Is this relation asymmetric?



Is this relation symmetric?

$$\forall a \in A.\ \forall b \in A.\ (aRb \rightarrow bRa)$$

Is this relation asymmetric?

$$\forall a \in A.\ \forall b \in A.\ (aRb \rightarrow b \not R a)$$

Theorem: A binary relation R is a strict order if and only if it's irreflexive and transitive.

Proof: Let R be an arbitrary relation over a set A that is irreflexive and transitive. We will prove that R is a strict order by proving that it is also asymmetric.

Assume for the sake of contradiction that R is not asymmetric. That means that there exist some $x, y \in A$ where $xRy \to yRx$.

We know that R is irreflexive, so for any $a \in A$, we know that $a \not R a$. Plugging in $x = a$ and $y = a$ into $xRy \to yRx$, tells us that aRa. However, this contradicts the fact that $a \not R a$.

We have reached a contradiction, so our assumption must have been wrong. Therefore, R is asymmetric, so R is a strict order. ■

Theorem: A binary relation R is a strict order if and only if it's irreflexive and transitive.

Proof: Let R be an arbitrary relation over a set A that is irreflexive and transitive. We will prove that R is a strict order by proving that it is also asymmetric.

Assume for the sake of contradiction that R is not asymmetric. That means that there exist some $x, y \in A$ where $xRy \to yRx$.

We know that R is irre

that $a \not R a$. Plugging in

us that aRa. However,

We have reached a co

must have been wron

is a strict order. ■

What is the negation of the statement

$$\forall x \in A.\ \forall y \in A.\ (xRy \to y \not R x)?$$

Theorem: A binary relation R is a strict order if and only if it's irreflexive and transitive.

Proof: Let R be an arbitrary relation over a set A that is irreflexive and transitive. We will prove that R is a strict order by proving that it is also asymmetric.

Assume for the sake of contradiction that R is not asymmetric. That means that there exist some $x, y \in A$ where $xRy \to yRx$.

We know that R is irre
that $a \not R a$. Plugging in
us that aRa. However,

We have reached a co
must have been wron
is a strict order. ■

What is the negation of the statement

$$\forall x \in A.\ \forall y \in A.\ (xRy \to y \not R x)?$$

It's

$$\exists x \in A.\ \exists y \in A.\ (xRy \land yRx).$$

Theorem: A binary relation R is a strict order if and only if it's irreflexive and transitive.

Proof: Let R be an arbitrary relation over a set A that is irreflexive and transitive. We will prove that R is a strict order by proving that it is also asymmetric.

Assume for the sake of contradiction that R is not asymmetric. That means that there exist some $x, y \in A$ where $xRy \to yRx$.

We know that R is irreflexive, so for any $a \in A$, we know that $a \not R a$. Plugging in $x=a$ and $y=a$ into $xRy \to yRx$, tells us that aRa. ... at $a \not R a$.

We have reac... tion must have be... ric, so R is a strict order. ■

Remember: ***don't use first-order logic notation in your proofs!***

Theorem: A binary relation R is a strict order if and only if it's irreflexive and transitive.

Proof: Let R be an arbitrary relation over a set A that is irreflexive and transitive. We will prove that R is a strict order by proving that it is also asymmetric.

Assume for the sake of contradiction that R is not asymmetric. That means that there exist some $x, y \in A$ where $xRy \to yRx$.

We know that R is irreflexive, so for any $a \in A$, we know that $a \not R a$. Plugging in $x = a$ and $y = a$ into $xRy \to yRx$, tells us that aRa. However, this contradicts the fact that $a \not R a$.

We have reached a contradiction, so our assumption must have been wrong. Therefore, R is asymmetric, so R is a strict order. ■

Theorem: A binary relation R is a strict order if and only if it's irreflexive and transitive.

Proof: Let R be an arbitrary relation over a set A that is irreflexive and transitive. We will prove that R is a strict order by proving that it is also asymmetric.

Assume for the sake of contradiction that R is not asymmetric. That means that there exist some $x, y \in A$ where $xRy \to yRx$.

We know that R is irreflexive, so for any $a \in A$, we know that $a \not R a$. Plugging in $x = a$ and $y = a$ into $xRy \to yRx$, tells us that aRa. However, this contradicts the fact that $a \not R a$.

We have reached a co
must have been wron
is a strict order. ■

The x and y discussed earlier are ***existentially-quantified***. That means that we know they exist, but we can't necessarily say what they are. We cannot come back later on and give them specific values!

Theorem: A binary relation R is a strict order if and only if it's irreflexive and transitive.

Proof: Let R be an arbitrary relation over a set A that is irreflexive and transitive. We will prove that R is a strict order by proving that it is also asymmetric.

Assume for the sake of contradiction that R is not asymmetric. That means that there exist some $x, y \in A$ where $xRy \to yRx$.

We know that R is irreflexive, so for any $a \in A$, we know that $a \not R a$. Plugging in $x = a$ and $y = a$ into $xRy \to yRx$, tells us that aRa. However, this contradicts the fact that $a \not R a$.

We have reached a contradiction, so our assumption must have been wrong. Therefore, R is asymmetric, so R is a strict order. ■

Theorem: A binary relation R is a strict order if and only if it's irreflexive and transitive.

Proof: Let R be an arbitrary relation over a set A that is irreflexive and trans... order by proving tha...

Remember to prove both directions of implication!

Assume for the sake... asymmetric. That m... where $xRy \to yRx$.

We know that R is irreflexive, so for any $a \in A$, we know that $a \not R a$. Plugging in $x=a$ and $y=a$ into $xRy \to yRx$, tells us that aRa. However, this contradicts the fact that $a \not R a$.

We have reached a contradiction, so our assumption must have been wrong. Therefore, R is asymmetric, so R is a strict order. ■

To Summarize

- Call back to definitions. If something isn’t asymmetric, it doesn’t mean it’s symmetric (or vice-versa).
- Watch your negations! There’s a reason we had you practice this on PS2 and why we spent time in lecture last week going over it.
- Don’t use first-order logic notation in your proofs.
- Watch how you introduce variables. Existentially-quantified variables have values that you can’t pick. Universally-quantified variables should be chosen arbitrarily (except in rare circumstances).
- Make sure to prove both directions of implication.

Your Questions

“What are your favorite books?”

Guns, Germs, and Steel by Jared Diamond
Brilliant thesis about the development of human civilizations.

Whistling Vivaldi by Claude Steele
Should be required reading at Stanford.

The Omnivore’s Dilemma by Michael Pollan
A peek into our food supply.

Command and Control by Eric Schlosser
A study in institutional dysfunction… with nuclear weapons.

The Trial by Franz Kafka
Especially the “Before the Law” section.

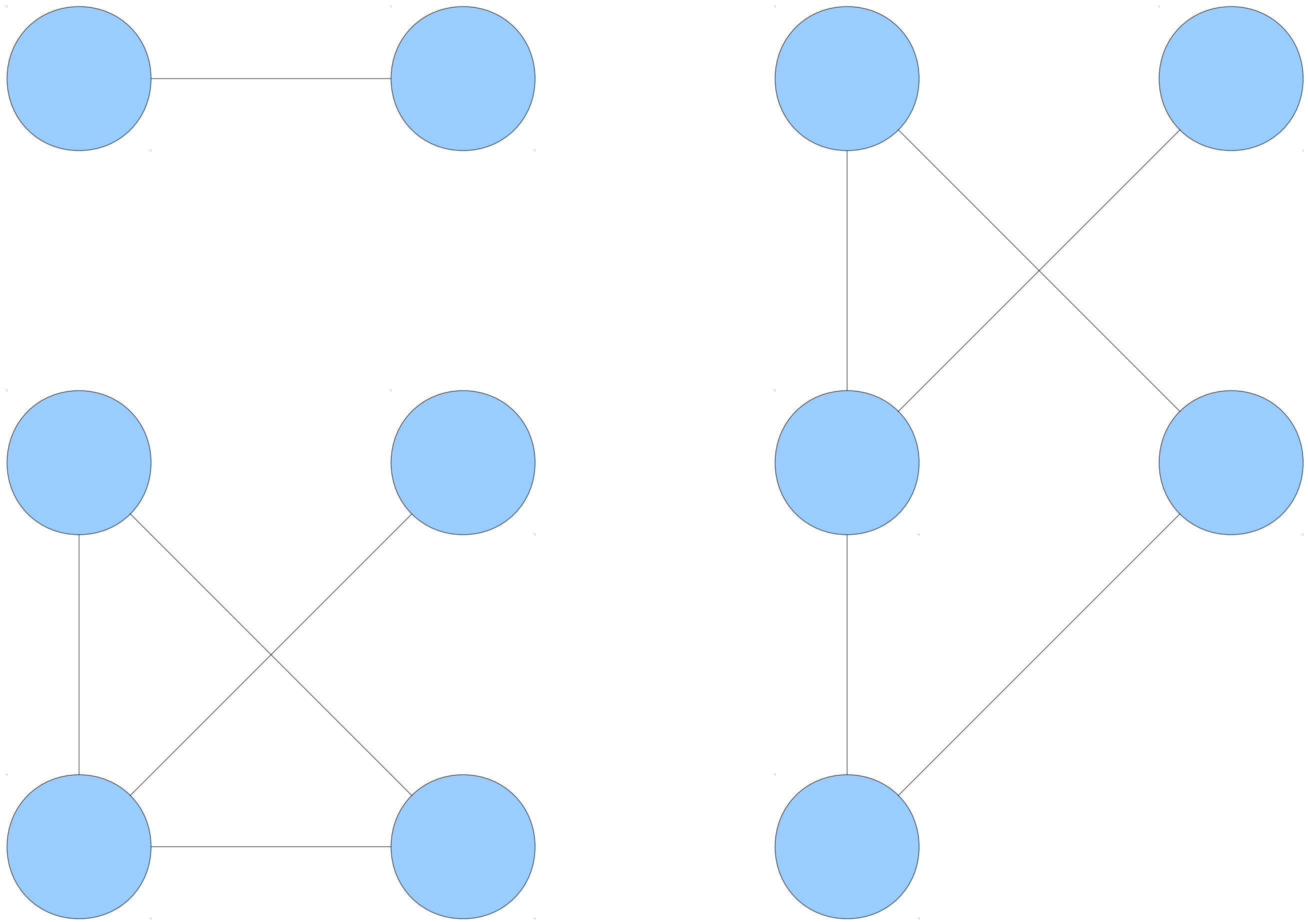
Radetsky March by Joseph Roth
A portrait of the twilight of the Hapsburg empire.

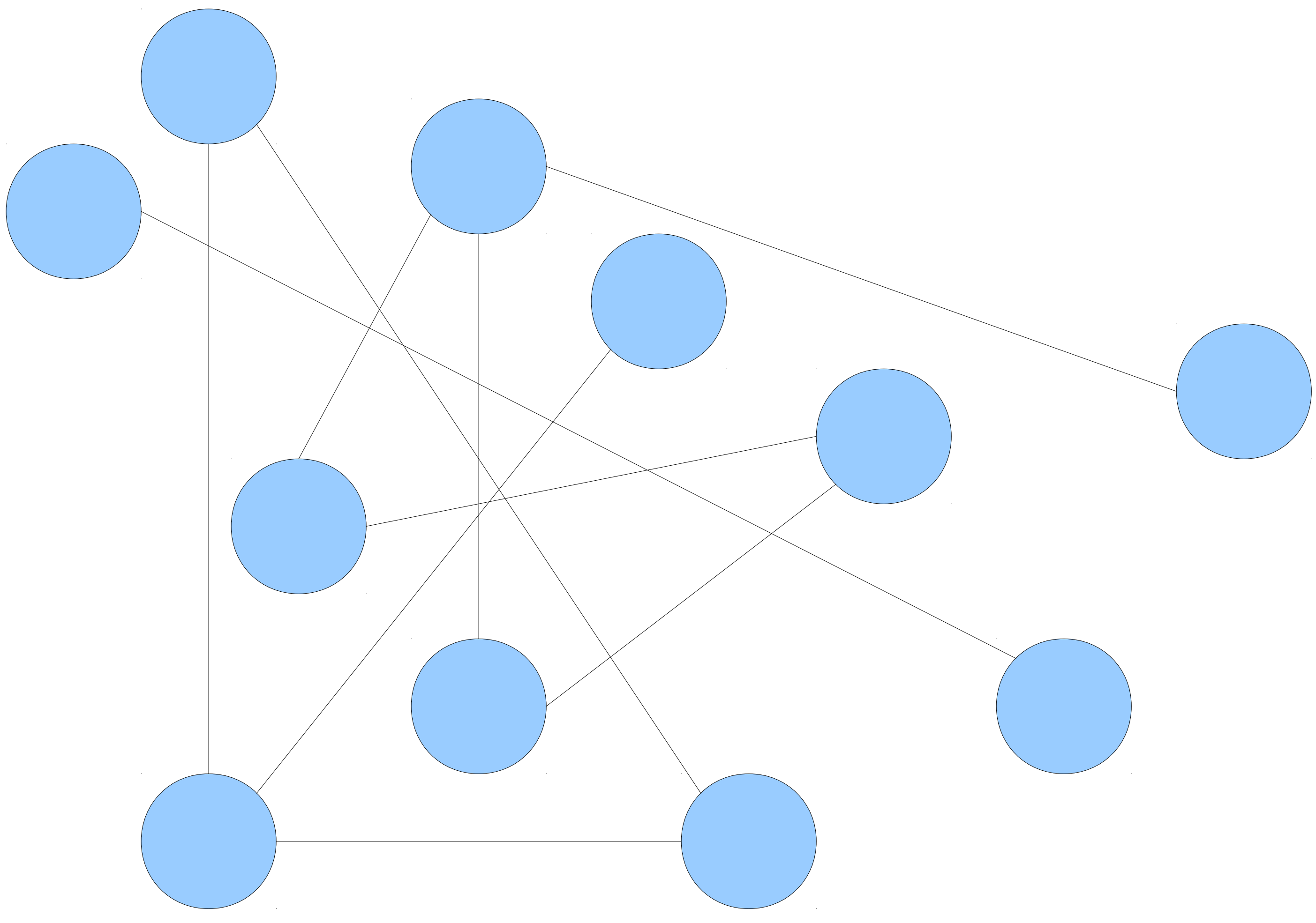
Stories of Your Life and Others by Ted Chiang
Excellent speculative fiction.

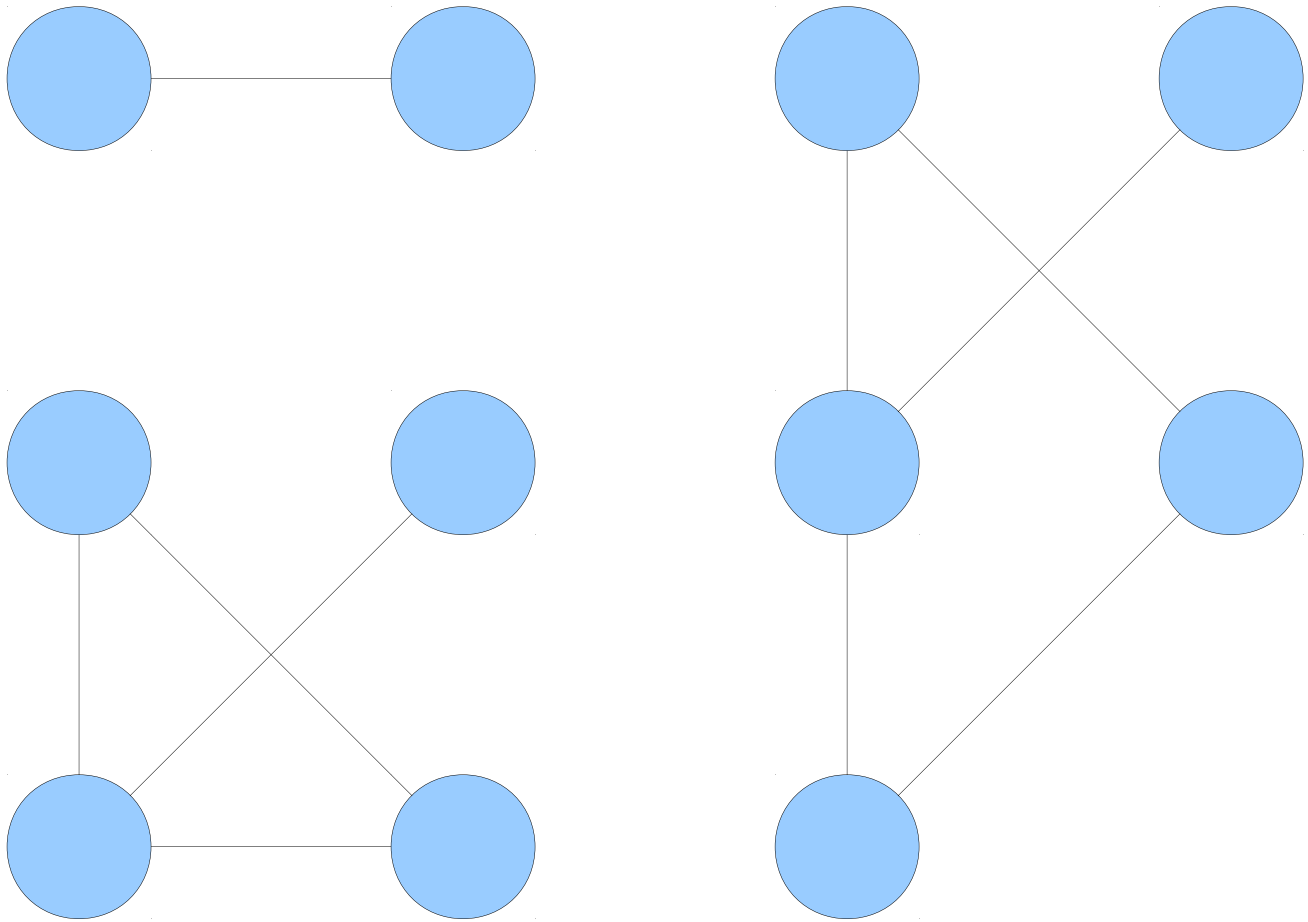
Longitude by Dava Sobel
Beautiful narrative history of how science is actually done.

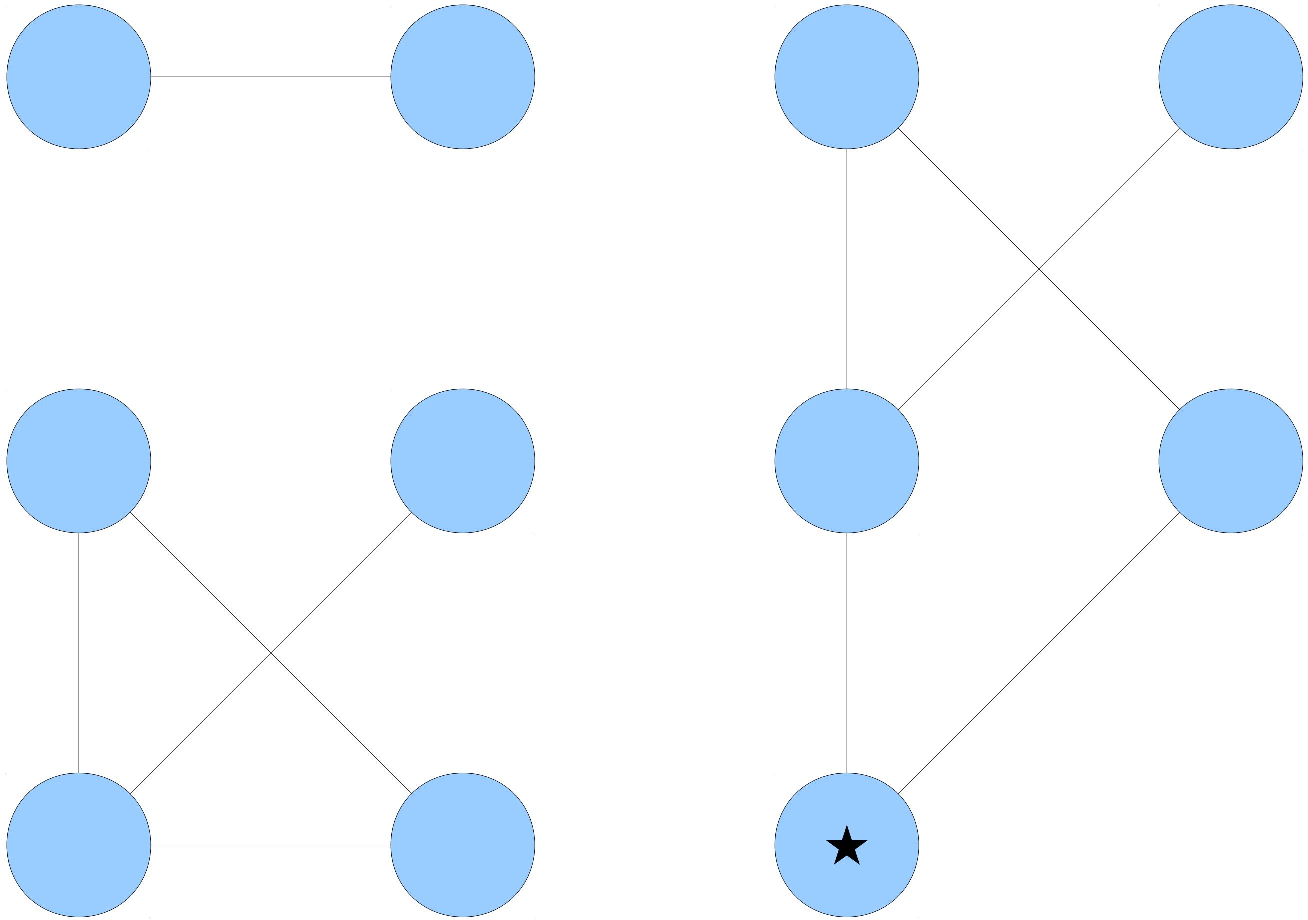
Back to CS103!

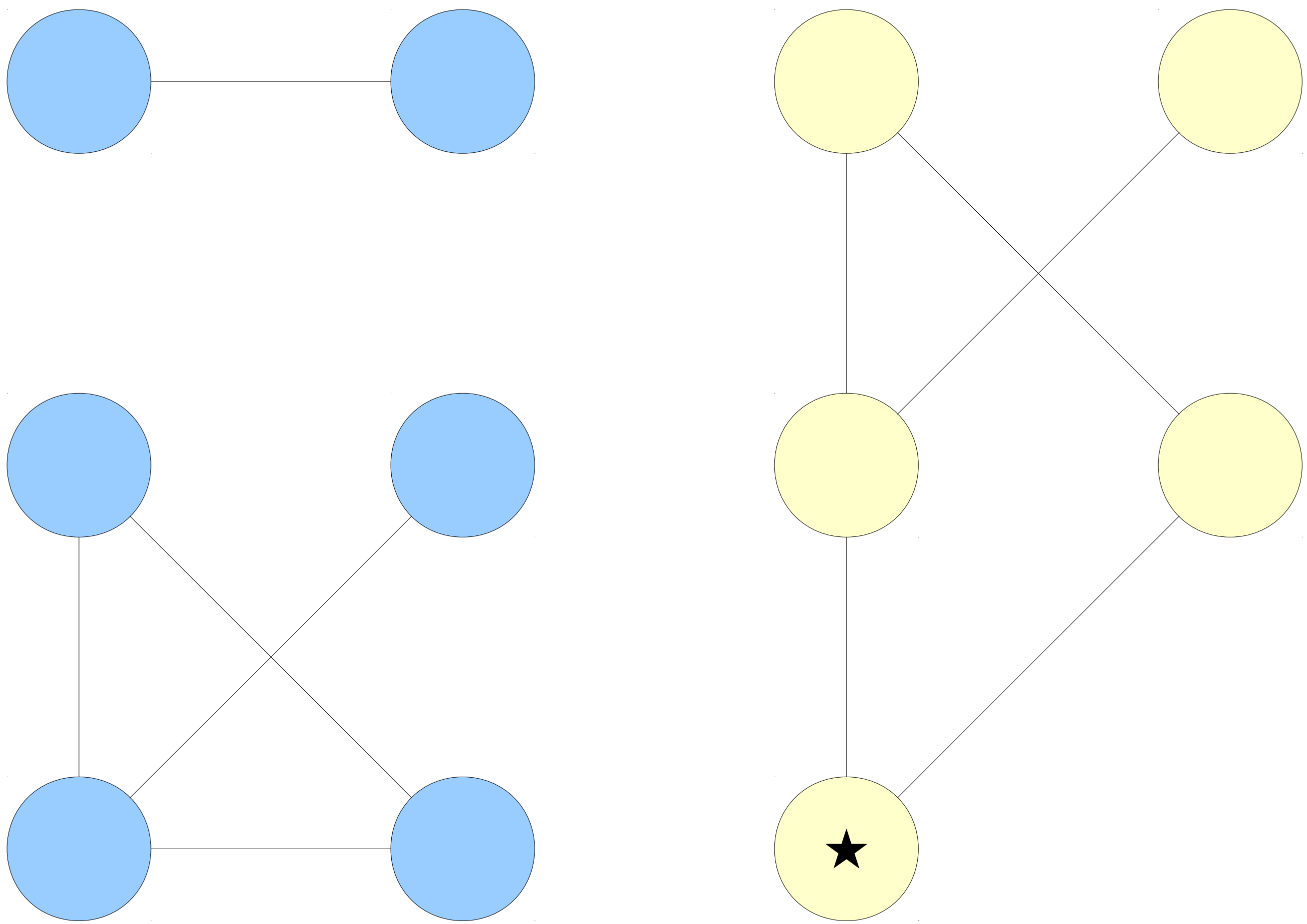
Connected Components











Connected Components

- Let $G = (V, E)$ be a graph. For each $v \in V$, the ***connected component*** containing v is the set

$$[v] = \{ x \in V \mid v \text{ is connected to } x \}$$

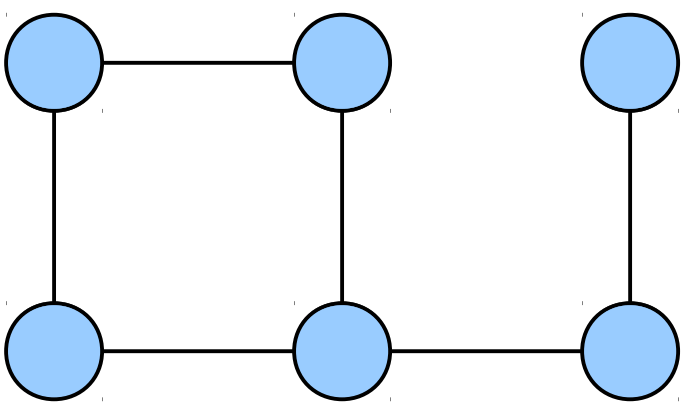
- Intuitively, a connected component is a “piece” of a graph in the sense we just talked about.
- ***Question:*** How do we know that this particular definition of a “piece” of a graph is a good one?
- ***Goal:*** Prove that any graph can be broken apart into different connected components.

We’re trying to reason about some way of partitioning the nodes in a graph into different groups.

What structure have we studied that captures the idea of a partition?

Connectivity

- ***Claim:*** For any graph *G*, the "is connected to" relation is an equivalence relation.
  - Is it reflexive?
  - Is it symmetric?
  - Is it transitive?



Connectivity

Claim: For any graph G, the "is connected to" relation is an equivalence relation.

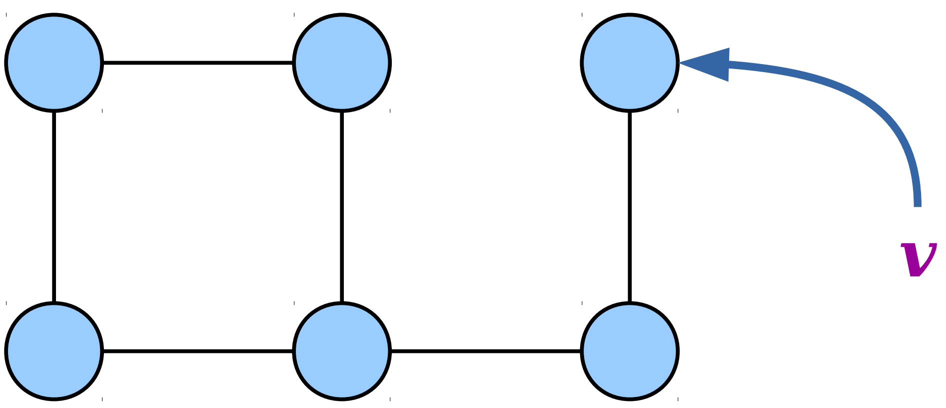
- Is it reflexive?

  Is it symmetric?

  Is it transitive?

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

A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.



Connectivity

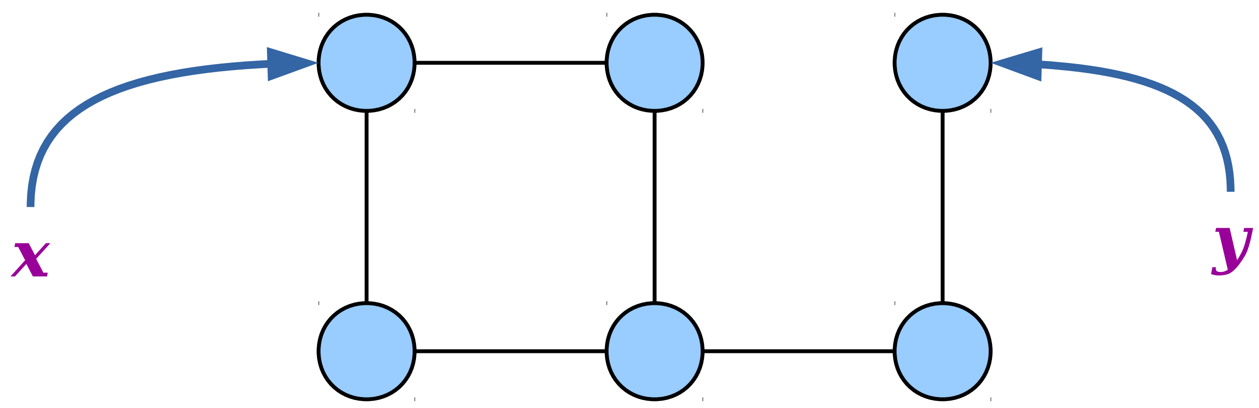
Claim: E
connecte
relation.

$$\forall x \in V.\ \forall y \in V.\ (Conn(x, y) \rightarrow Conn(y, x))$$

Is it reflexive?

- Is it symmetric?

Is it transitive?



Connectivity

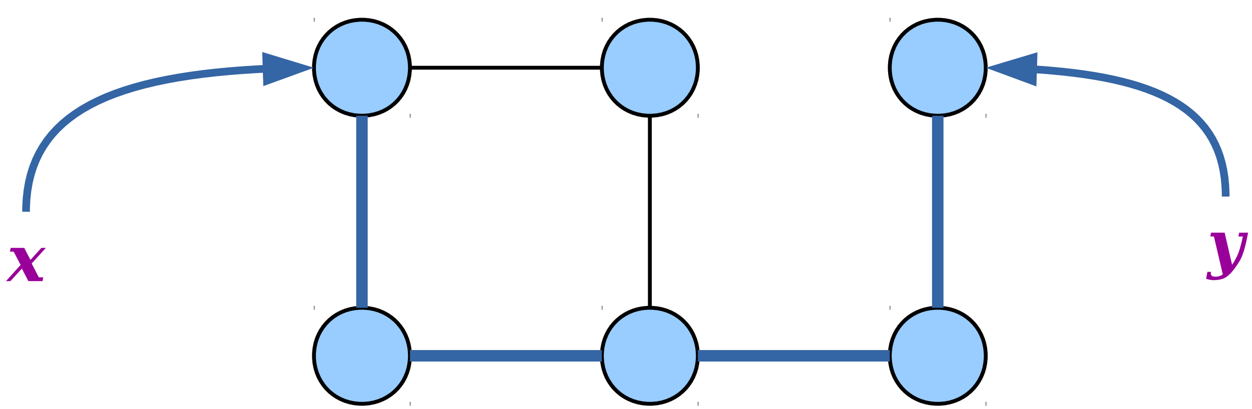
Claim: E
connecte
relation.

$$\forall x \in V.\ \forall y \in V.\ (Conn(x, y) \rightarrow Conn(y, x))$$

Is it reflexive?

- Is it symmetric?

Is it transitive?



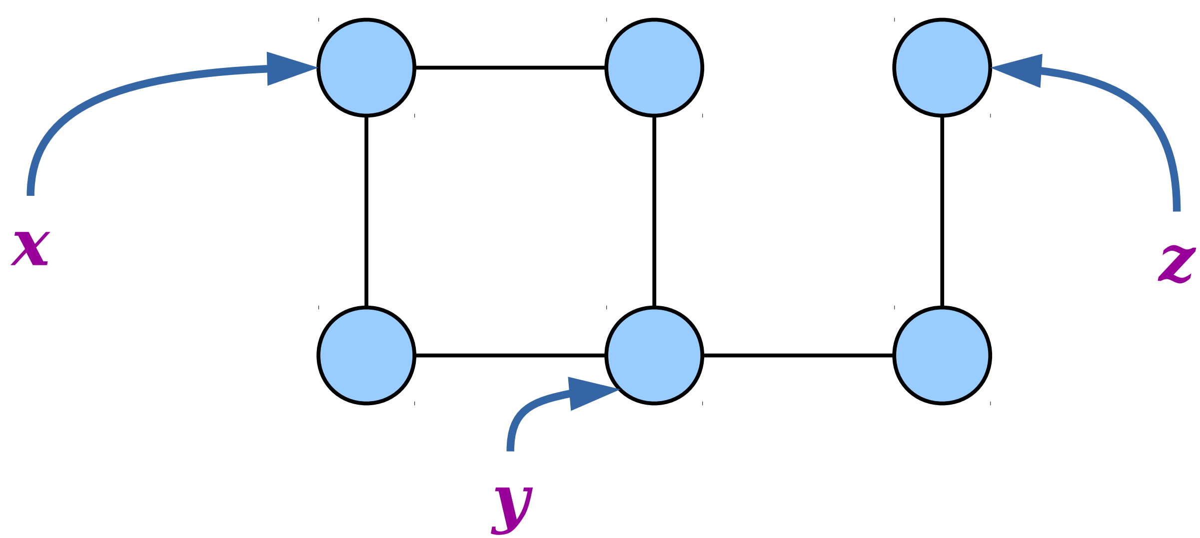
Connectivity

$$\forall x \in V.\ \forall y \in V.\ \forall z \in V.\ (Conn(x, y) \land Conn(y, z) \to Conn(x, z))$$

Is it reflexive?

Is it symmetric?

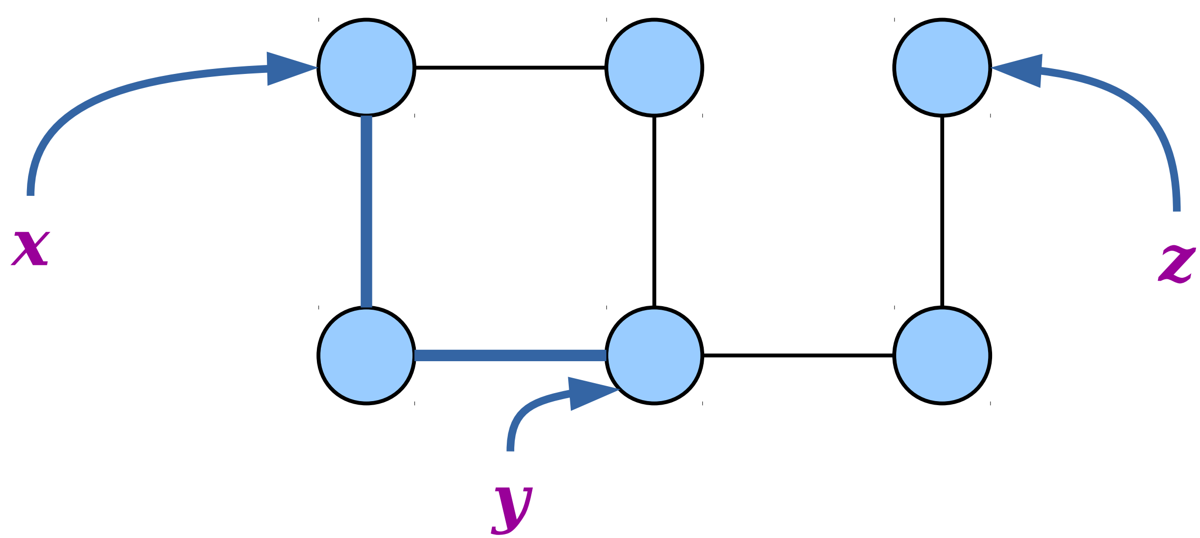
- Is it transitive?



Connectivity

$$\forall x \in V.\ \forall y \in V.\ \forall z \in V.\ (Conn(x, y) \land Conn(y, z) \rightarrow Conn(x, z))$$

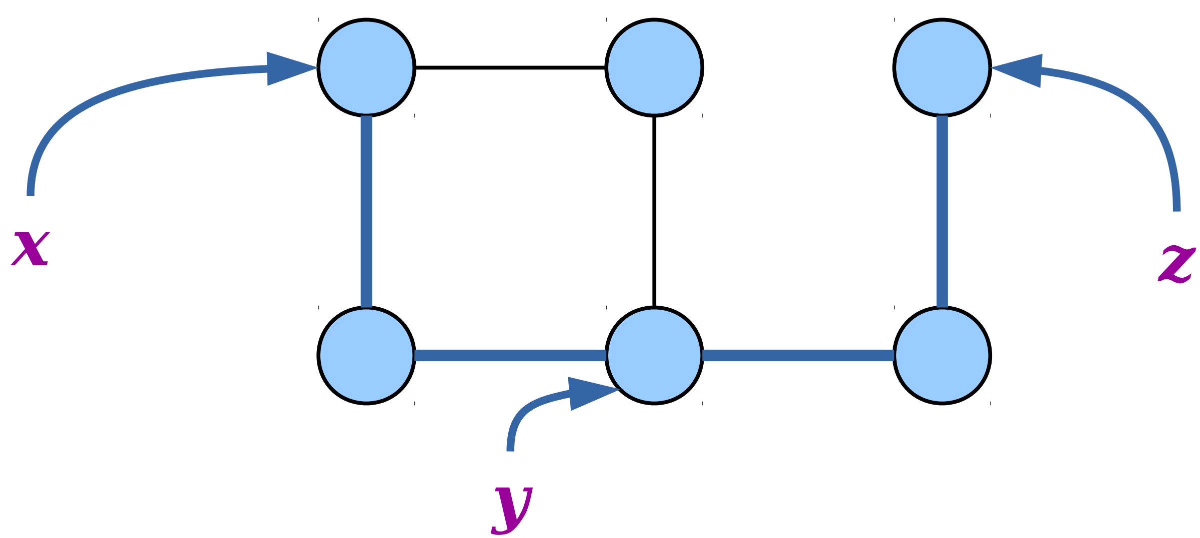
- Is it reflexive?
- Is it symmetric?
- Is it transitive?



Connectivity

$$\forall x \in V.\ \forall y \in V.\ \forall z \in V.\ (Conn(x, y) \land Conn(y, z) \to Conn(x, z))$$

- Is it reflexive?
- Is it symmetric?
- Is it transitive?



Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof:

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$. Then the singleton path v is a path from v to itself.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$. Then the singleton path v is a path from v to itself. Therefore, v is connected to itself, as required.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$. Then the singleton path v is a path from v to itself. Therefore, v is connected to itself, as required.

To show that connectivity is symmetric, consider any $x, y \in V$ where x is connected to y.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$. Then the singleton path v is a path from v to itself. Therefore, v is connected to itself, as required.

To show that connectivity is symmetric, consider any $x, y \in V$ where x is connected to y. We need to show that y is connected to x.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$. Then the singleton path v is a path from v to itself. Therefore, v is connected to itself, as required.

To show that connectivity is symmetric, consider any $x, y \in V$ where x is connected to y. We need to show that y is connected to x. Since x is connected to y, there is some path $x, v_1, \ldots, v_n, y$ from x to y.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$. Then the singleton path v is a path from v to itself. Therefore, v is connected to itself, as required.

To show that connectivity is symmetric, consider any $x, y \in V$ where x is connected to y. We need to show that y is connected to x. Since x is connected to y, there is some path $x, v_1, \ldots, v_n, y$ from x to y. Then $y, v_n, \ldots, v_1, x$ is a path from y back to x, so y is connected to x.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$. Then the singleton path v is a path from v to itself. Therefore, v is connected to itself, as required.

To show that connectivity is symmetric, consider any $x, y \in V$ where x is connected to y. We need to show that y is connected to x. Since x is connected to y, there is some path $x, v_1, \ldots, v_n, y$ from x to y. Then $y, v_n, \ldots, v_1, x$ is a path from y back to x, so y is connected to x.

Finally, to show that connectivity is transitive, let $x, y, z \in V$ be arbitrary nodes where x is connected to y and y is connected to z.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$. Then the singleton path v is a path from v to itself. Therefore, v is connected to itself, as required.

To show that connectivity is symmetric, consider any $x, y \in V$ where x is connected to y. We need to show that y is connected to x. Since x is connected to y, there is some path $x, v_1, \ldots, v_n, y$ from x to y. Then $y, v_n, \ldots, v_1, x$ is a path from y back to x, so y is connected to x.

Finally, to show that connectivity is transitive, let $x, y, z \in V$ be arbitrary nodes where x is connected to y and y is connected to z. We will prove that x is connected to z.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$. Then the singleton path v is a path from v to itself. Therefore, v is connected to itself, as required.

To show that connectivity is symmetric, consider any $x, y \in V$ where x is connected to y. We need to show that y is connected to x. Since x is connected to y, there is some path $x, v_1, \ldots, v_n, y$ from x to y. Then $y, v_n, \ldots, v_1, x$ is a path from y back to x, so y is connected to x.

Finally, to show that connectivity is transitive, let $x, y, z \in V$ be arbitrary nodes where x is connected to y and y is connected to z. We will prove that x is connected to z. Since x is connected to y, there is a path $x, u_1, \ldots, u_n, y$ from x to y.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$. Then the singleton path v is a path from v to itself. Therefore, v is connected to itself, as required.

To show that connectivity is symmetric, consider any $x, y \in V$ where x is connected to y. We need to show that y is connected to x. Since x is connected to y, there is some path $x, v_1, \ldots, v_n, y$ from x to y. Then $y, v_n, \ldots, v_1, x$ is a path from y back to x, so y is connected to x.

Finally, to show that connectivity is transitive, let $x, y, z \in V$ be arbitrary nodes where x is connected to y and y is connected to z. We will prove that x is connected to z. Since x is connected to y, there is a path $x, u_1, \ldots, u_n, y$ from x to y. Since y is connected to z, there is a path $y, v_1, \ldots, v_k, z$ from y to z.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$. Then the singleton path v is a path from v to itself. Therefore, v is connected to itself, as required.

To show that connectivity is symmetric, consider any $x, y \in V$ where x is connected to y. We need to show that y is connected to x. Since x is connected to y, there is some path $x, v_1, \ldots, v_n, y$ from x to y. Then $y, v_n, \ldots, v_1, x$ is a path from y back to x, so y is connected to x.

Finally, to show that connectivity is transitive, let $x, y, z \in V$ be arbitrary nodes where x is connected to y and y is connected to z. We will prove that x is connected to z. Since x is connected to y, there is a path $x, u_1, \ldots, u_n, y$ from x to y. Since y is connected to z, there is a path $y, v_1, \ldots, v_k, z$ from y to z. Then the path $x, u_1, \ldots, u_n, y, v_1, \ldots, v_k, z$ goes from x to z.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$. Then the singleton path v is a path from v to itself. Therefore, v is connected to itself, as required.

To show that connectivity is symmetric, consider any $x, y \in V$ where x is connected to y. We need to show that y is connected to x. Since x is connected to y, there is some path $x, v_1, \ldots, v_n, y$ from x to y. Then $y, v_n, \ldots, v_1, x$ is a path from y back to x, so y is connected to x.

Finally, to show that connectivity is transitive, let $x, y, z \in V$ be arbitrary nodes where x is connected to y and y is connected to z. We will prove that x is connected to z. Since x is connected to y, there is a path $x, u_1, \ldots, u_n, y$ from x to y. Since y is connected to z, there is a path $y, v_1, \ldots, v_k, z$ from y to z. Then the path $x, u_1, \ldots, u_n, y, v_1, \ldots, v_k, z$ goes from x to z. Thus x is connected to z, as required.

Theorem: Let $G = (V, E)$ be a graph. Then the connectivity relation over V is an equivalence relation.

Proof: Consider an arbitrary graph $G = (V, E)$. We will prove that the connectivity relation over V is reflexive, symmetric, and transitive.

To show that connectivity is reflexive, consider any $v \in V$. Then the singleton path v is a path from v to itself. Therefore, v is connected to itself, as required.

To show that connectivity is symmetric, consider any $x, y \in V$ where x is connected to y. We need to show that y is connected to x. Since x is connected to y, there is some path $x, v_1, \ldots, v_n, y$ from x to y. Then $y, v_n, \ldots, v_1, x$ is a path from y back to x, so y is connected to x.

Finally, to show that connectivity is transitive, let $x, y, z \in V$ be arbitrary nodes where x is connected to y and y is connected to z. We will prove that x is connected to z. Since x is connected to y, there is a path $x, u_1, \ldots, u_n, y$ from x to y. Since y is connected to z, there is a path $y, v_1, \ldots, v_k, z$ from y to z. Then the path $x, u_1, \ldots, u_n, y, v_1, \ldots, v_k, z$ goes from x to z. Thus x is connected to z, as required. ■

Putting Things Together

- Earlier, we defined the connected component of a node v to be

$$[v] = \{ x \in V \mid v \text{ is connected to } x \}$$

- Connectivity is an equivalence relation! So what's the equivalence class of a node v with respect to connectivity?

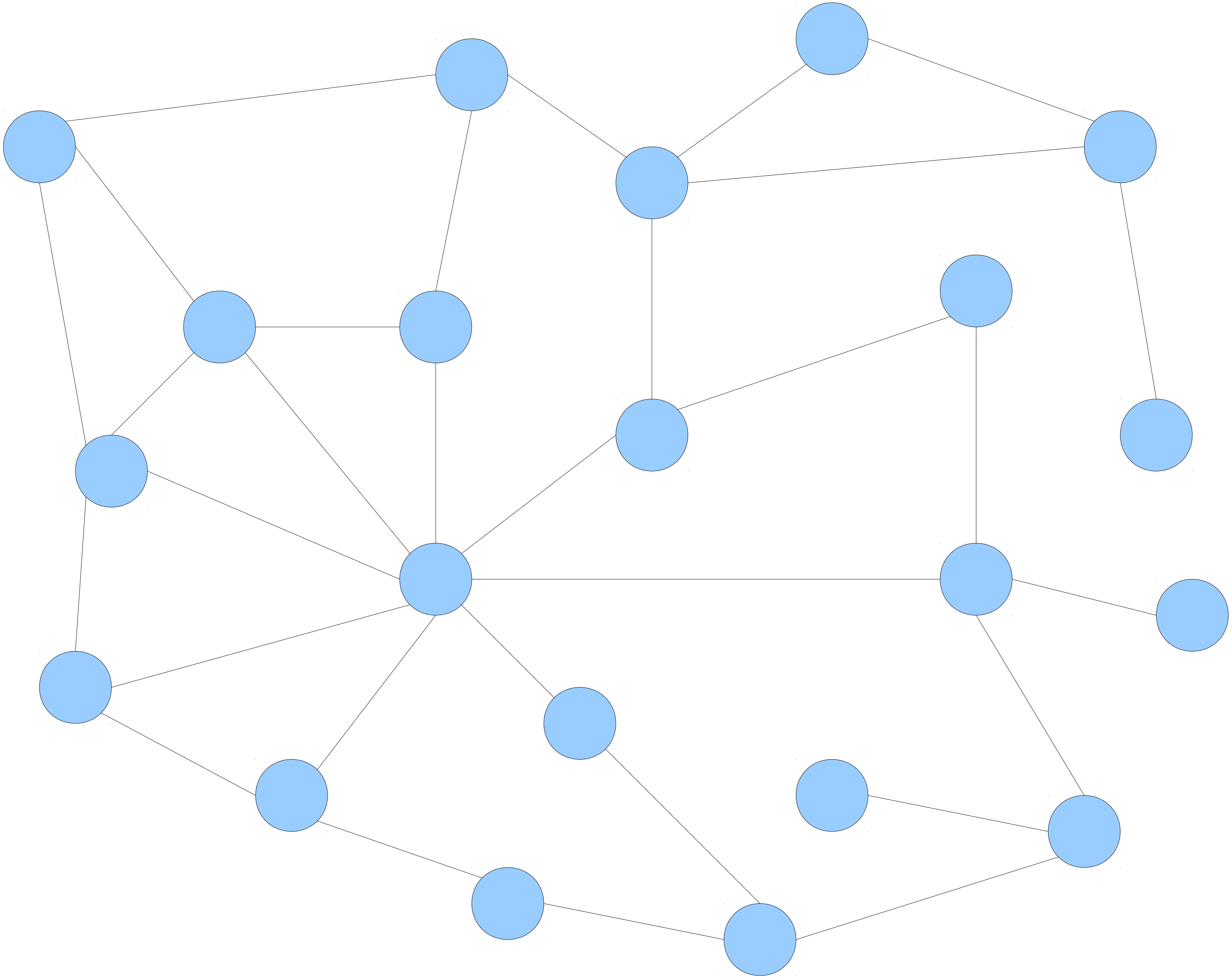
$$[v] = \{ x \in V \mid v \text{ is connected to } x \}$$

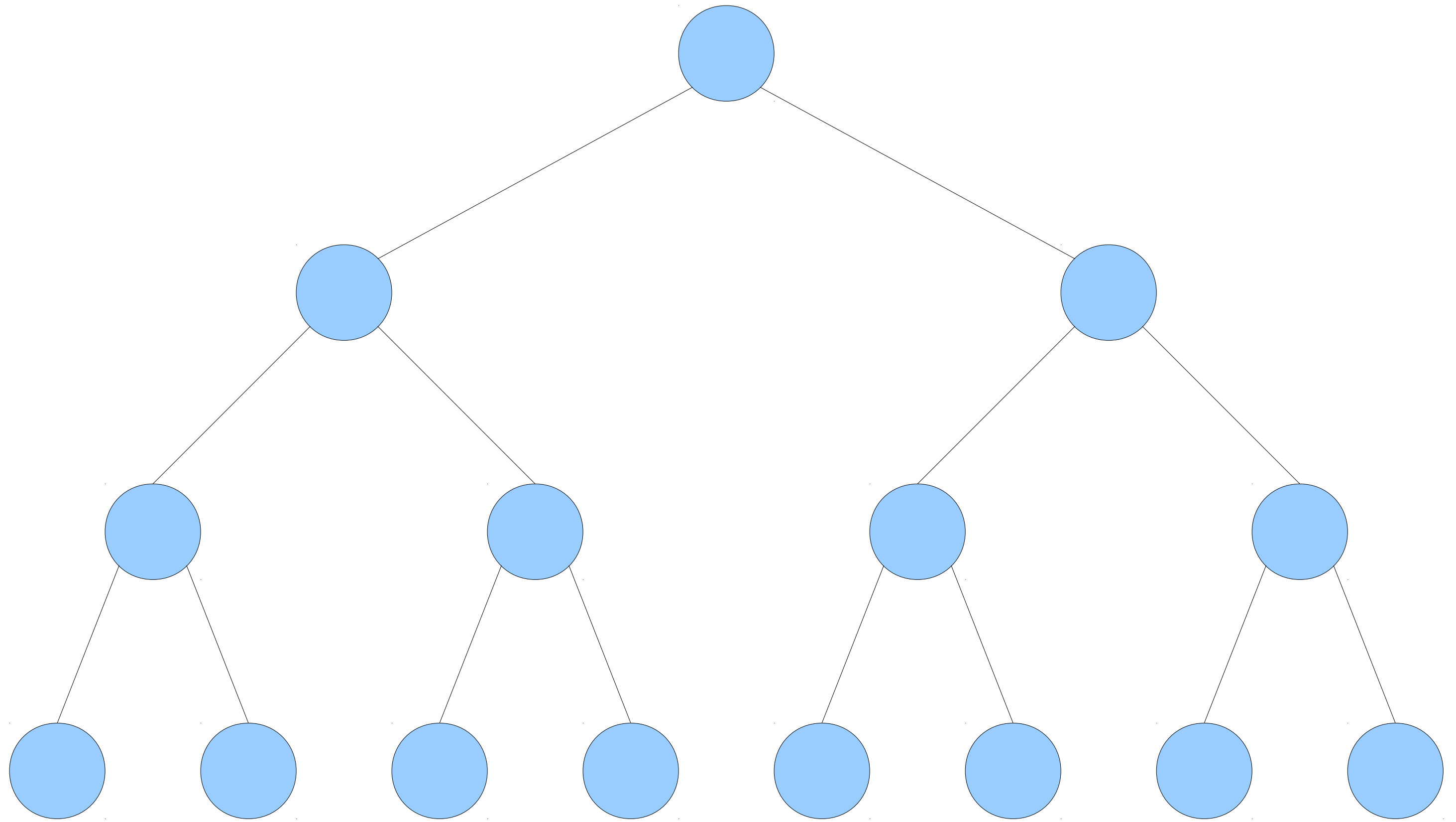
- ***Connected components are equivalence classes of the connectivity relation!***

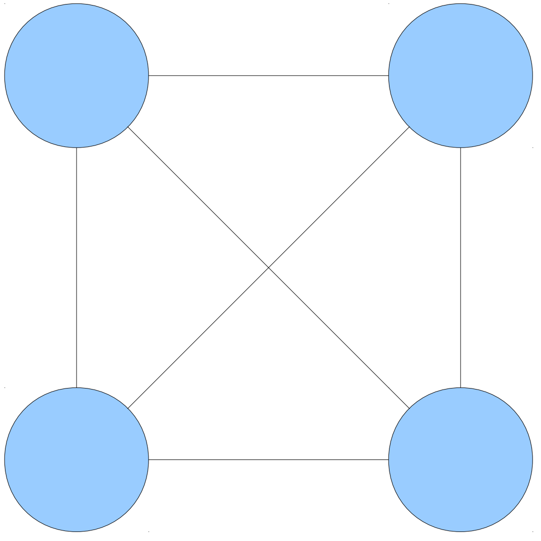
Theorem: If $G = (V, E)$ is a graph, then every node in G belongs to exactly one connected component of G.

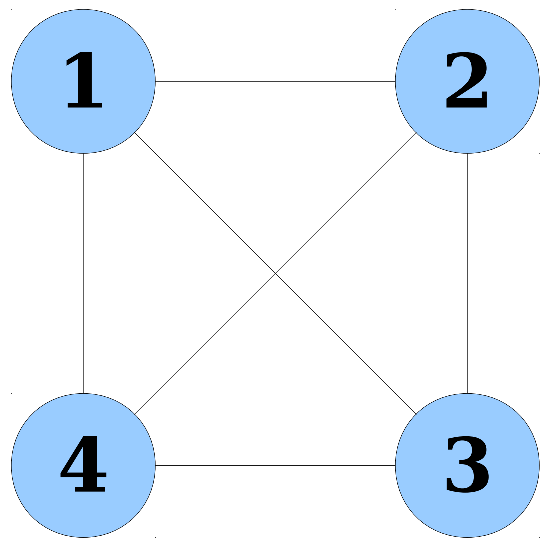
Proof: Let $G = (V, E)$ be an arbitrary graph and let $v \in V$ be any node in G. The connected components of G are just the equivalence classes of the connectivity relation in G. The Fundamental Theorem of Equivalence Relations guarantees that v belongs to exactly one equivalence class of the connectivity relation. Therefore, v belongs to exactly one connected component in G. ■

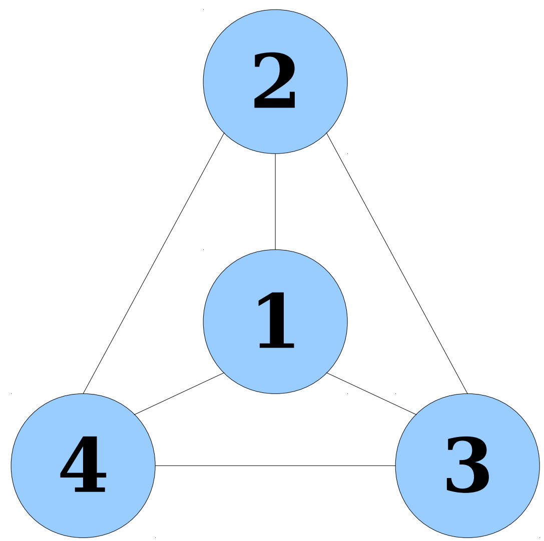
Planar Graphs

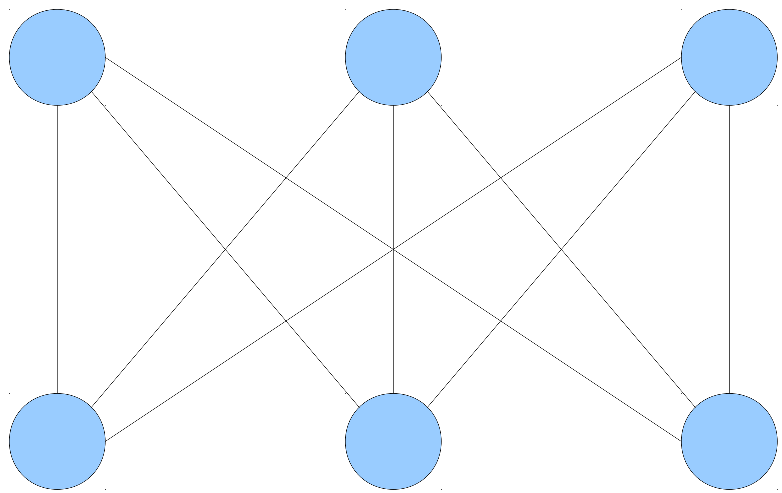


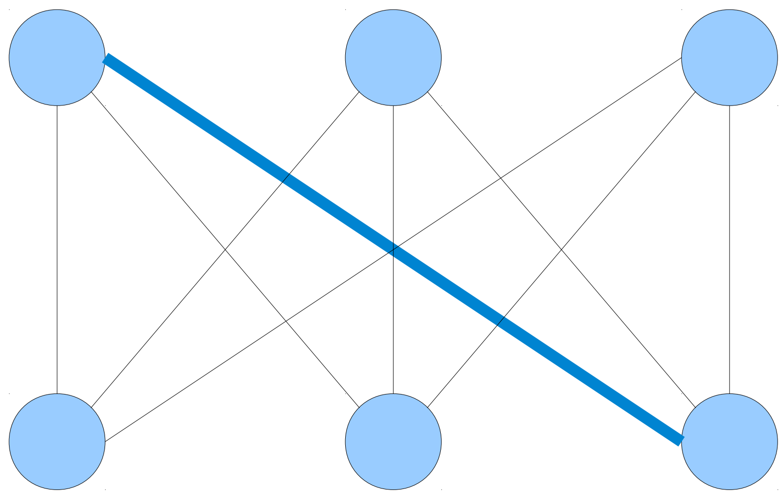


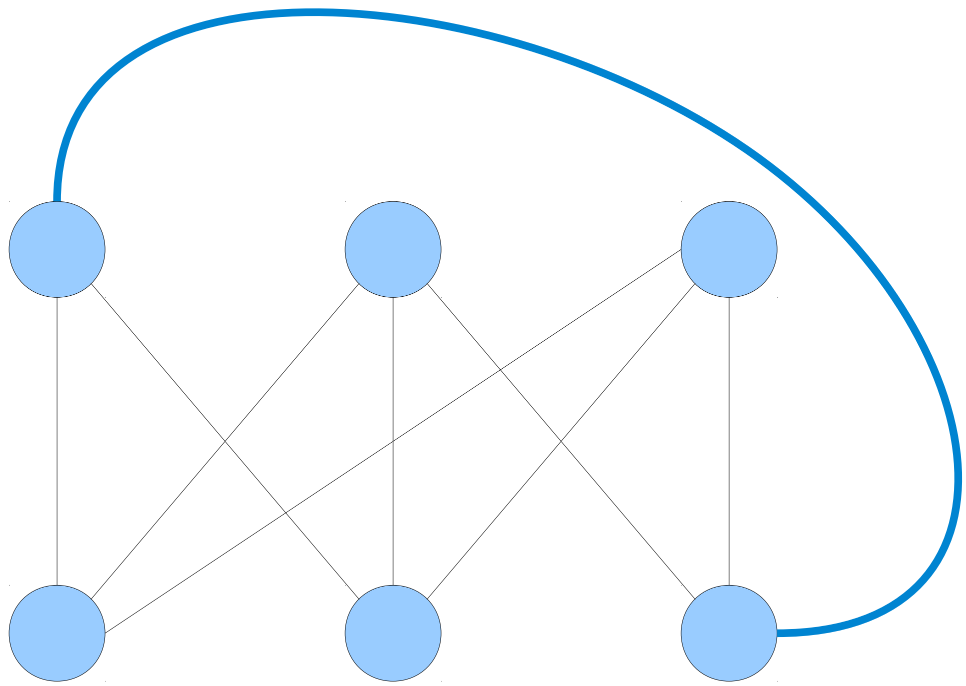


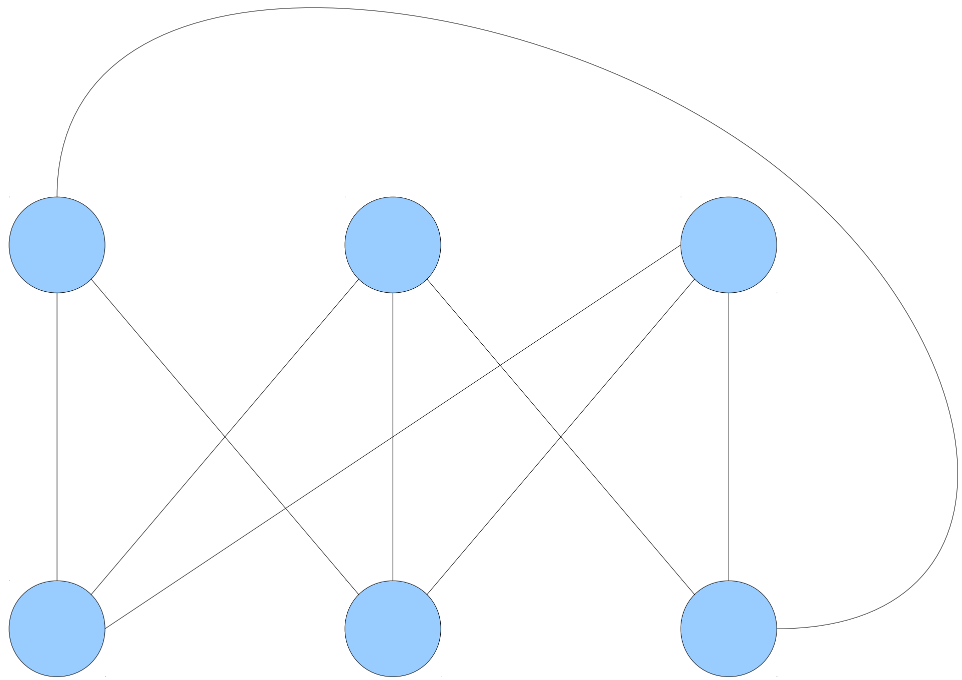

1
2
4
3

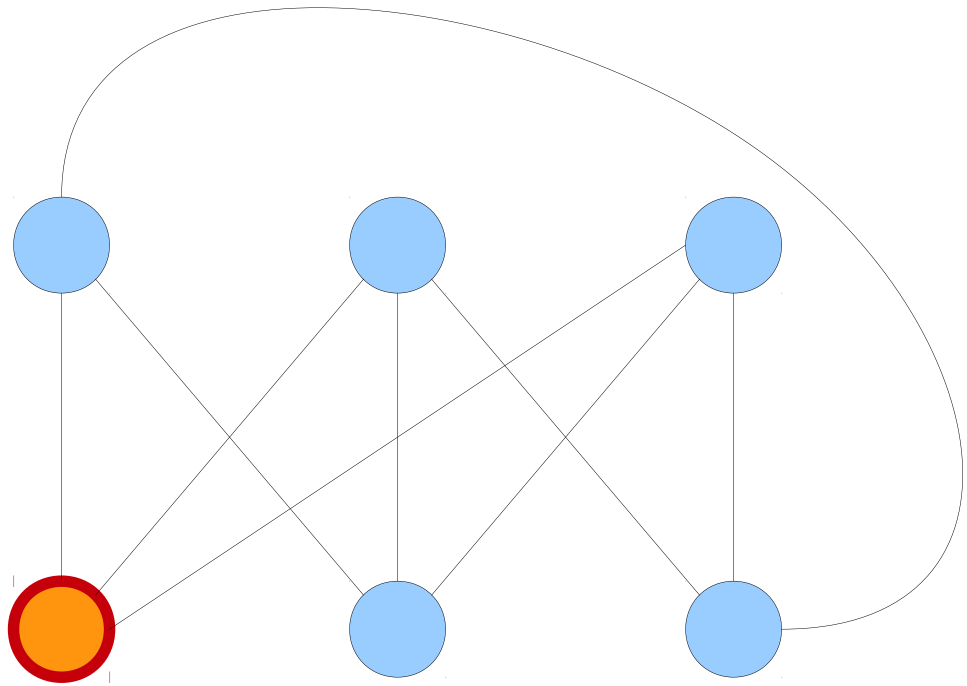

2
1
4
3

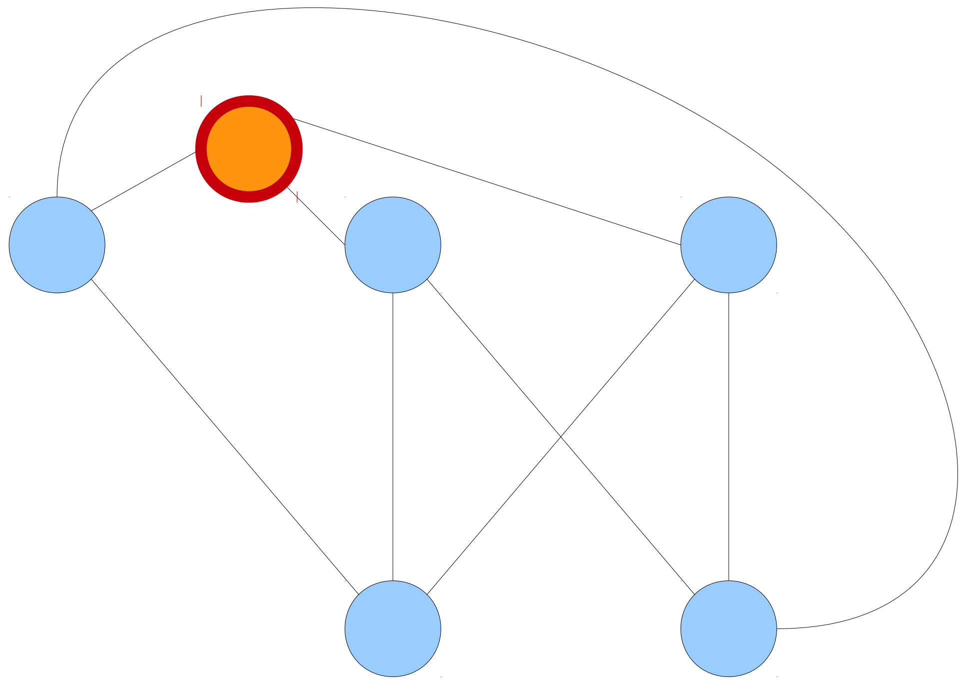


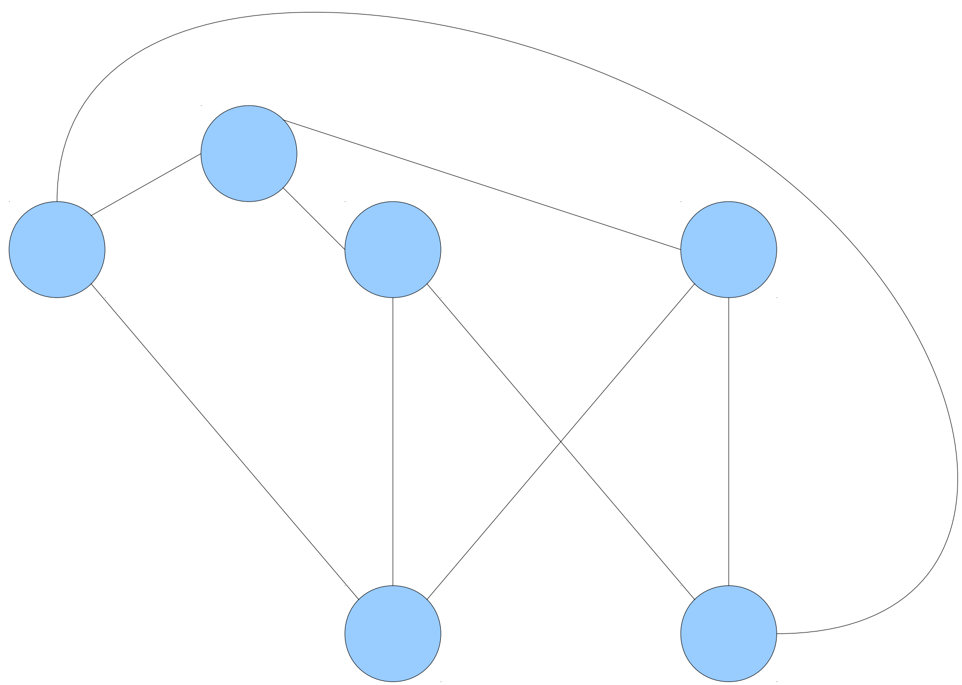


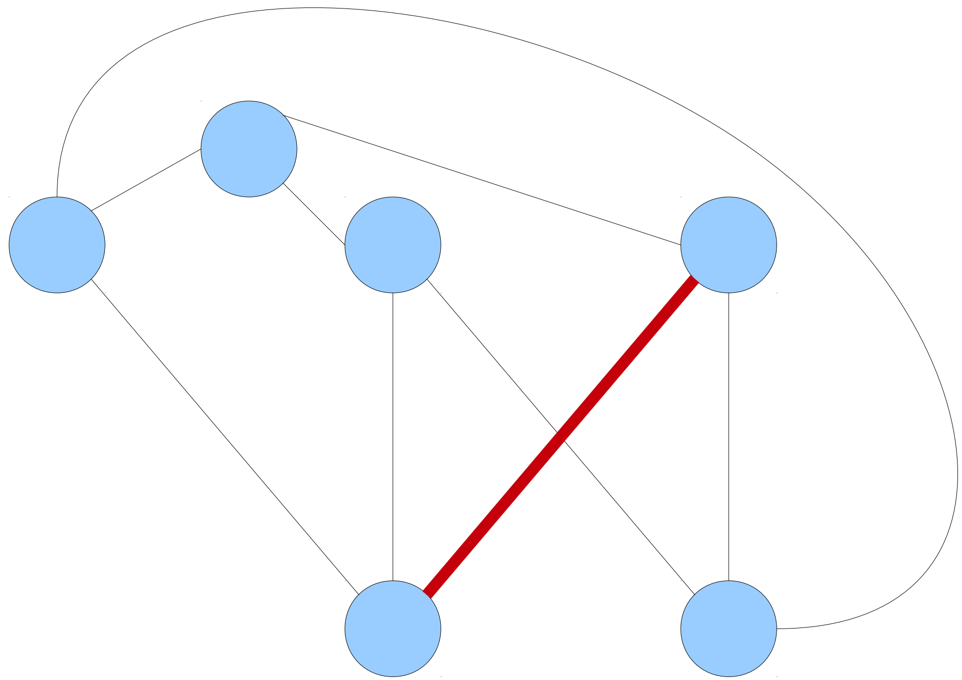


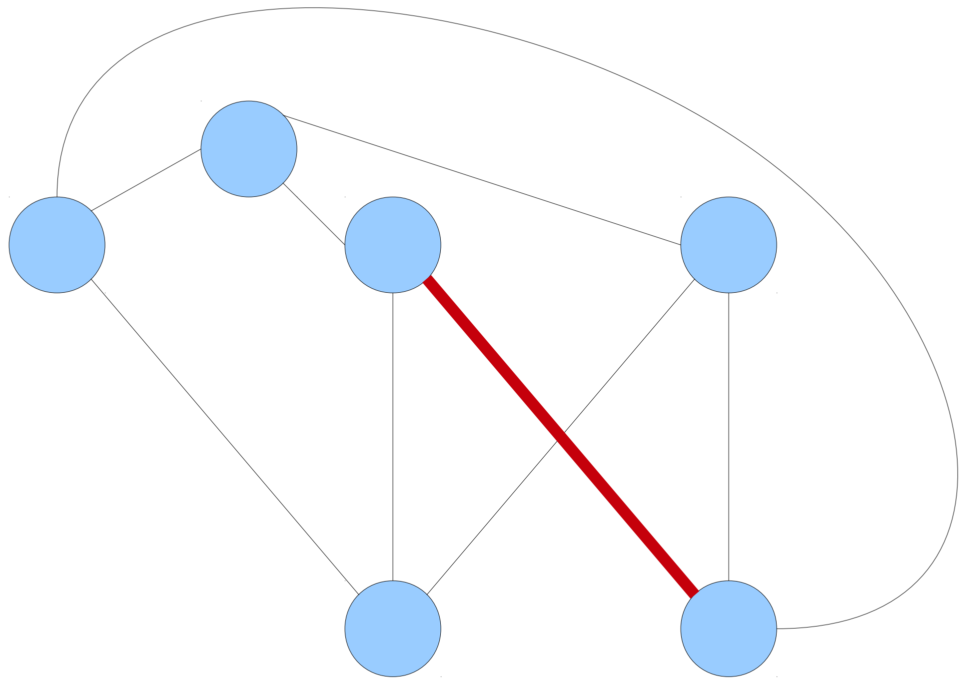


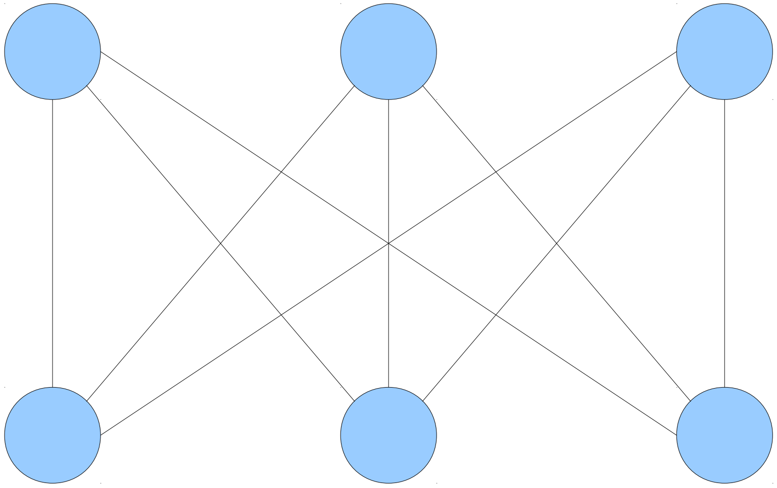










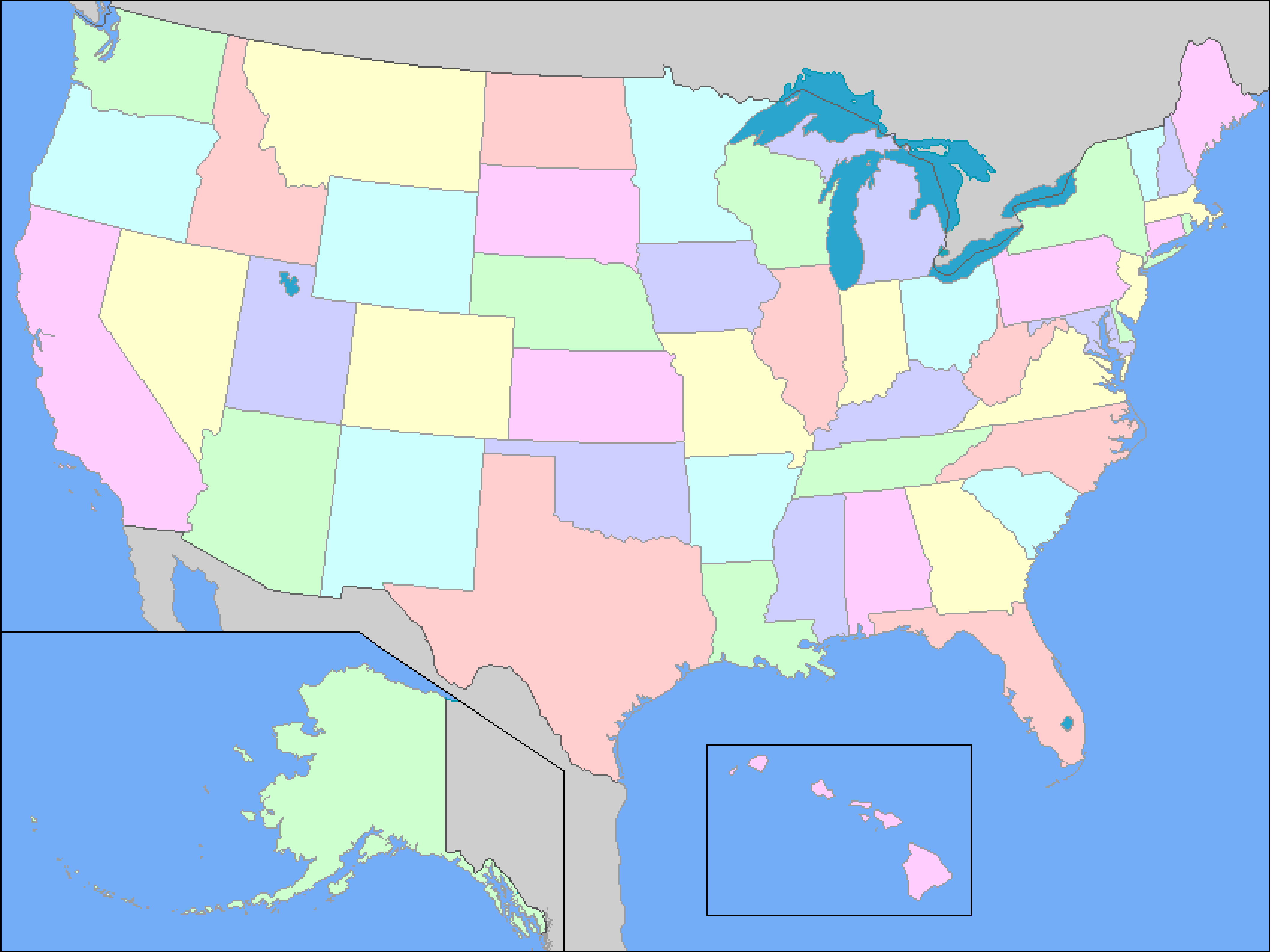


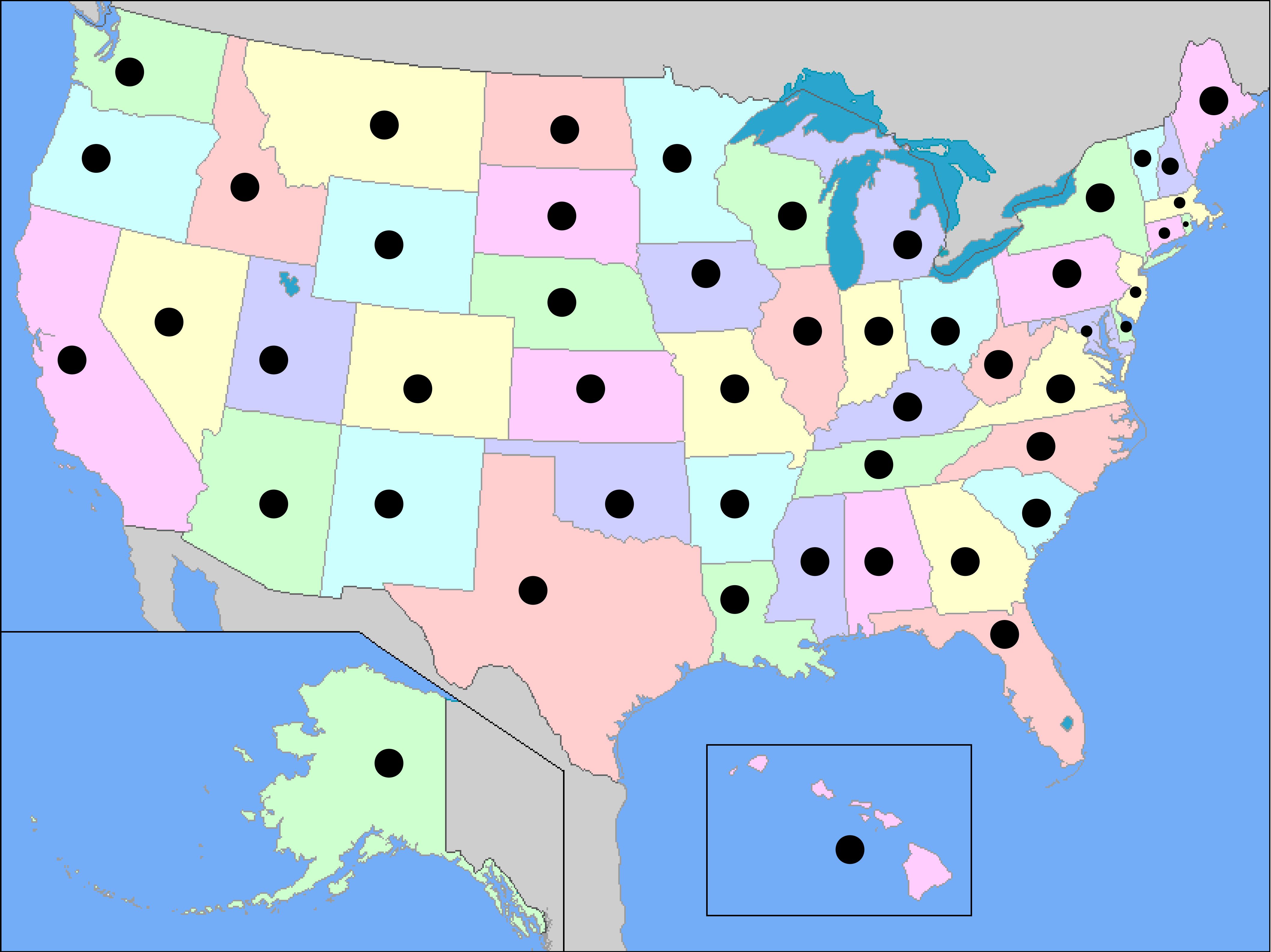
This graph is called the ***utility graph***. There is no way to draw it in the plane without edges crossing.

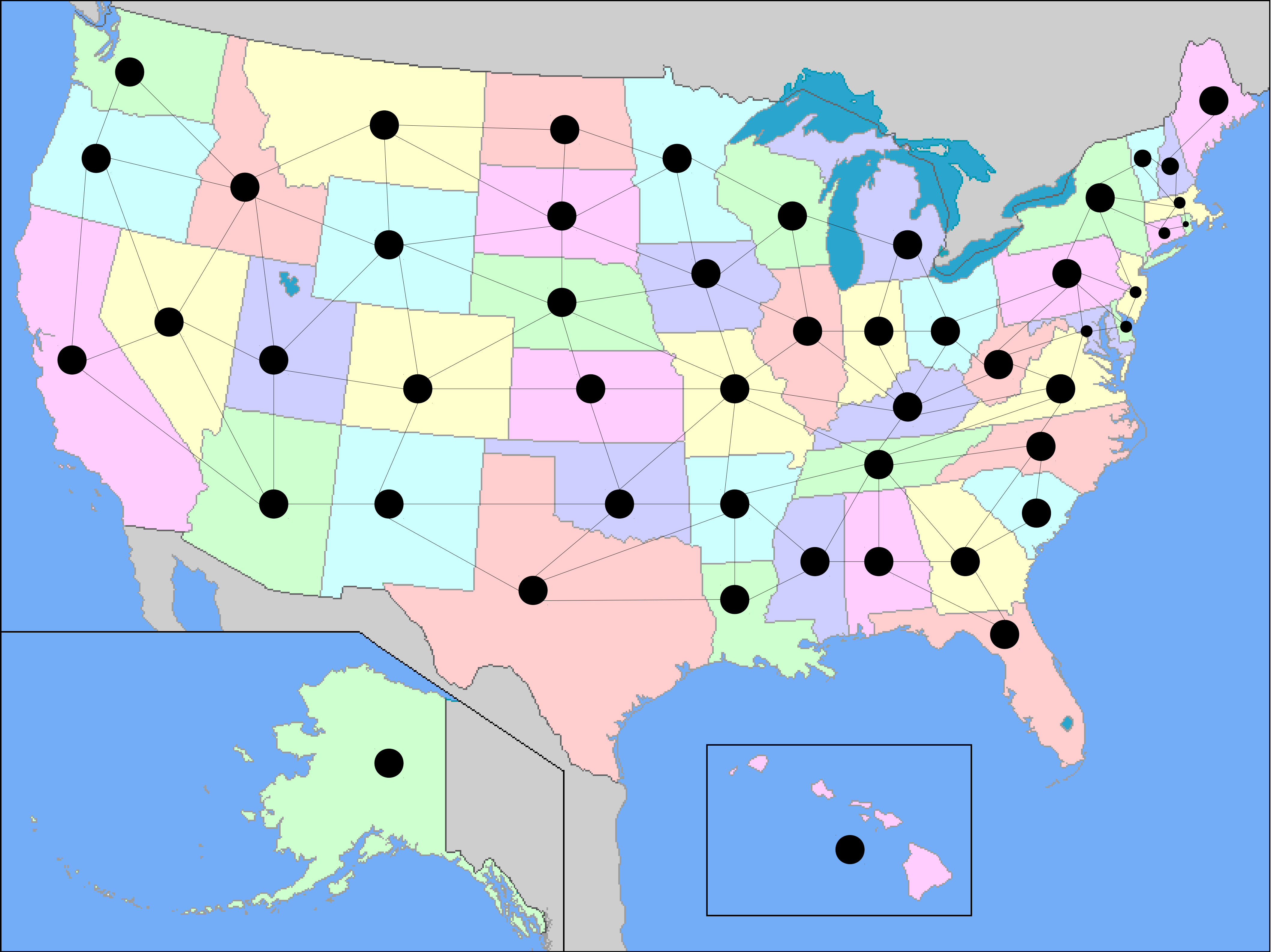
A graph is called a ***planar graph*** if there is some way to draw it in a 2D plane without any of the edges crossing.

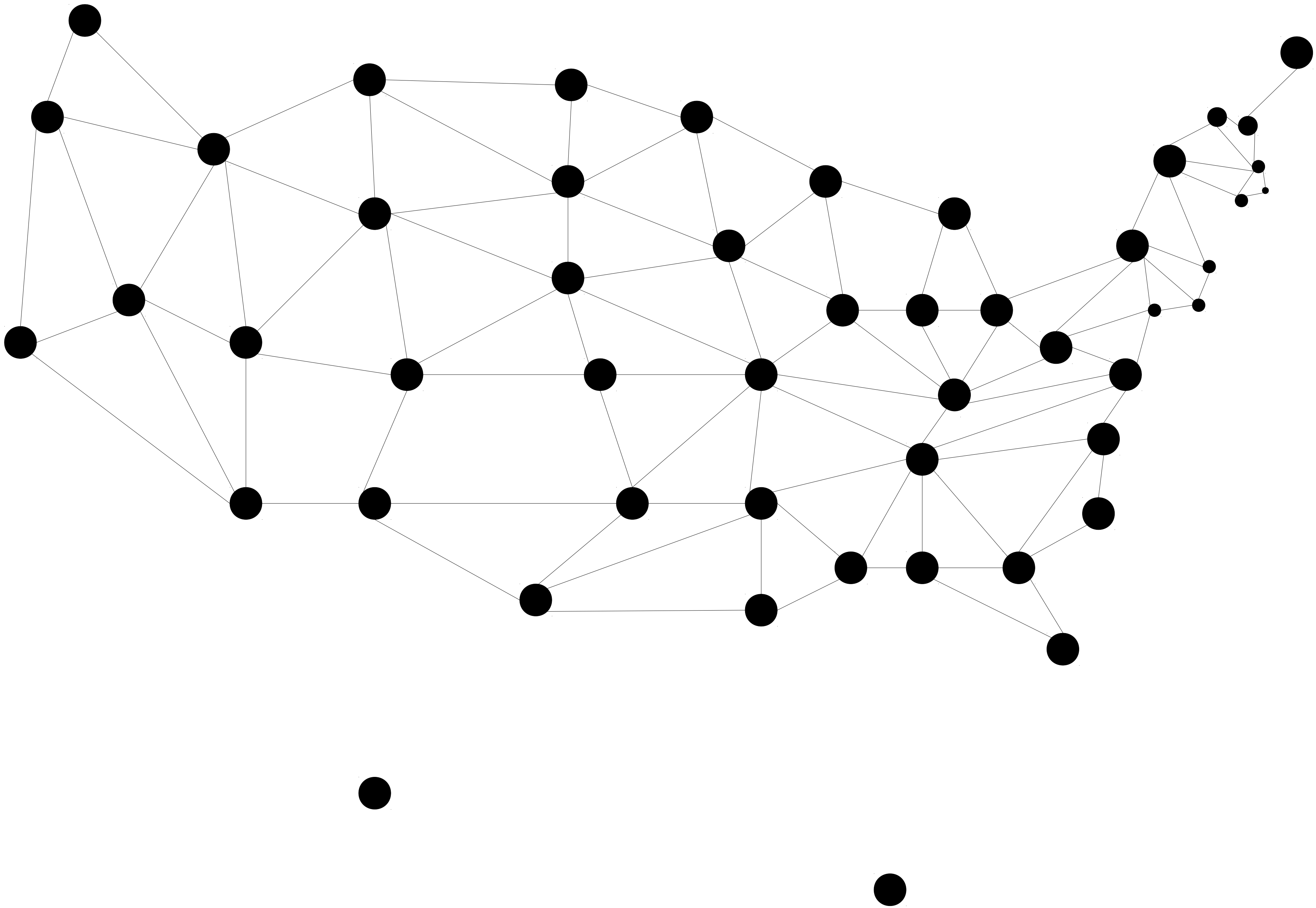
A Fun (And Strangely Addicting) Game:

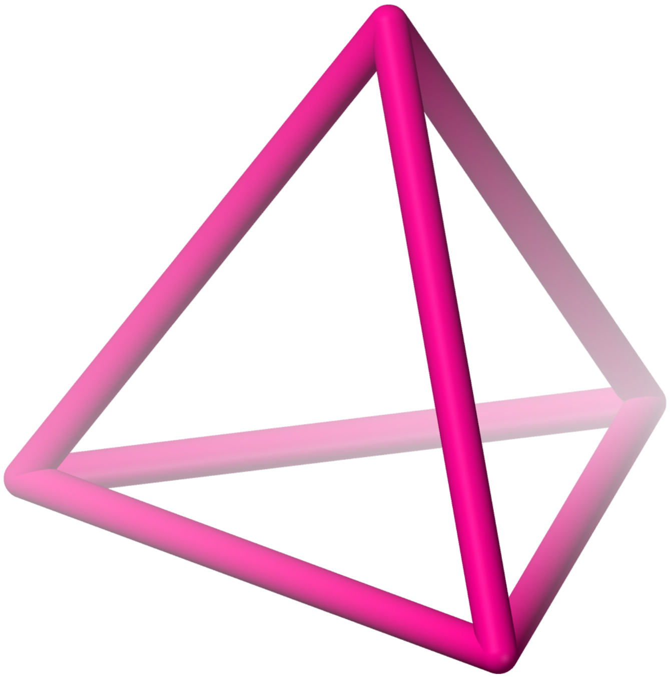
http://planarity.net/

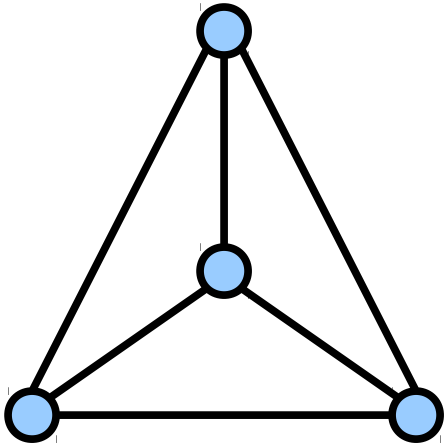


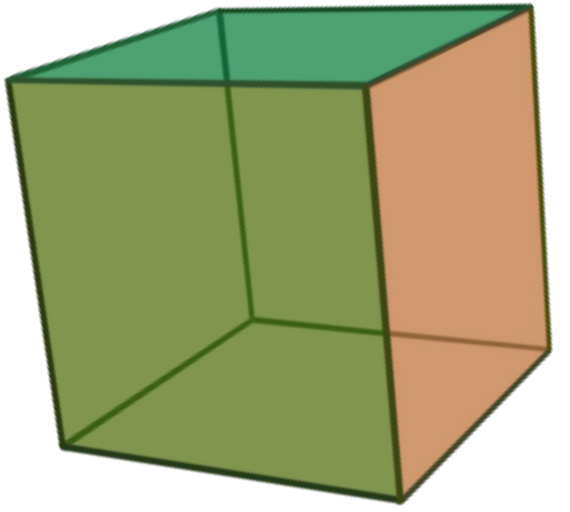


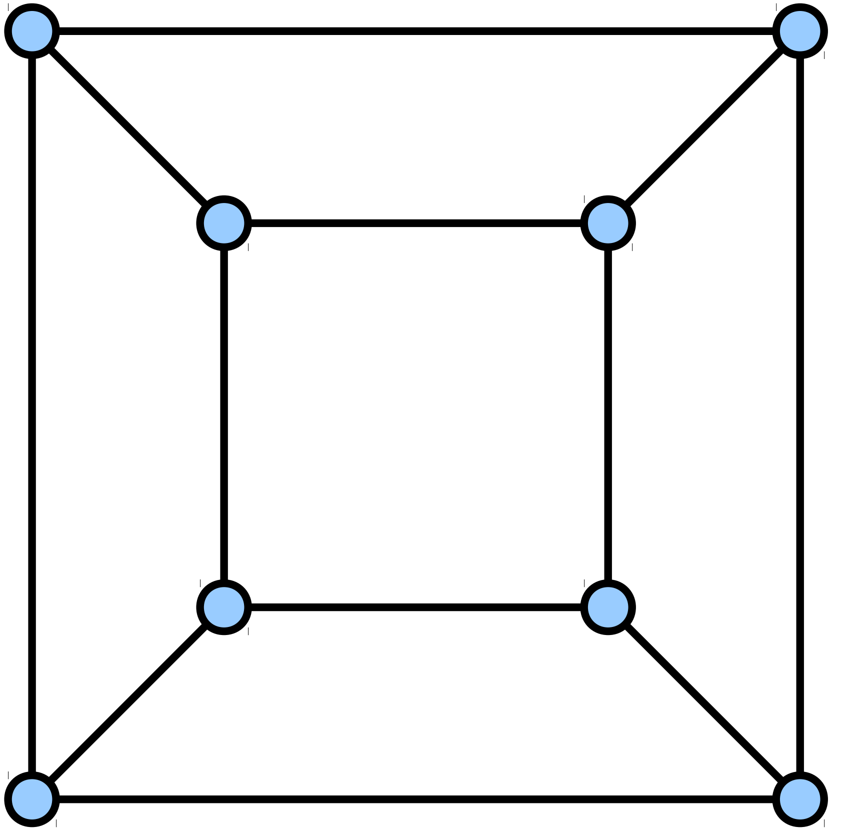


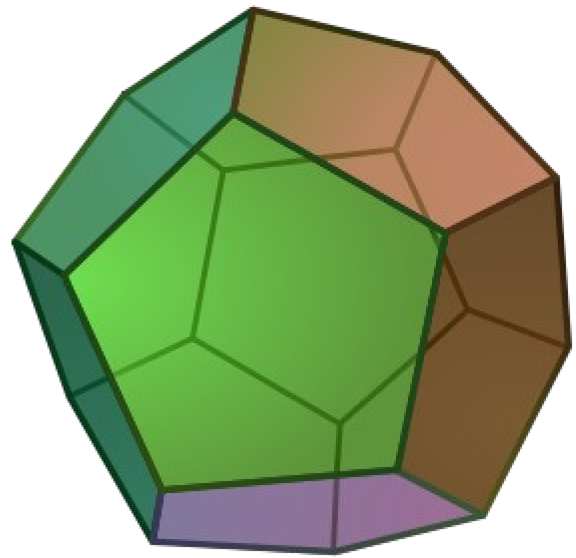


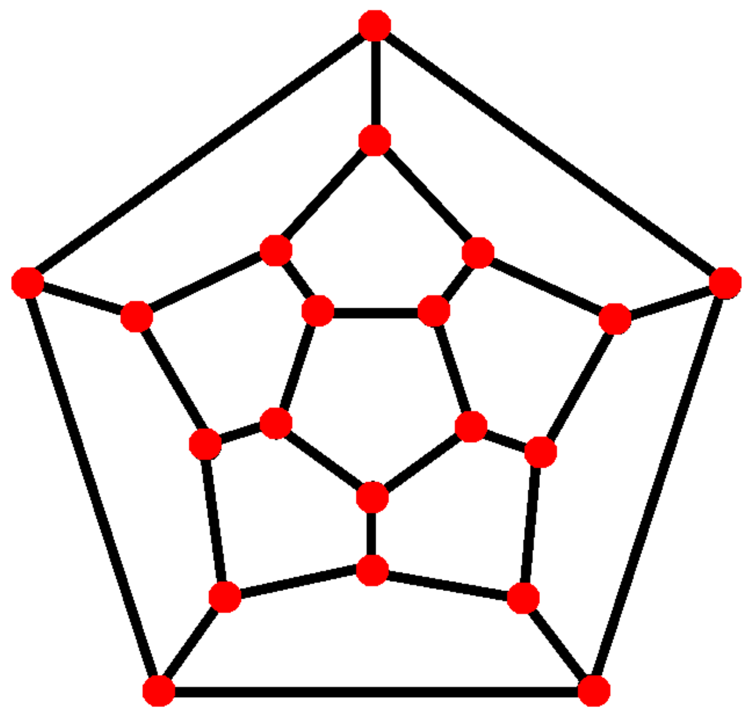




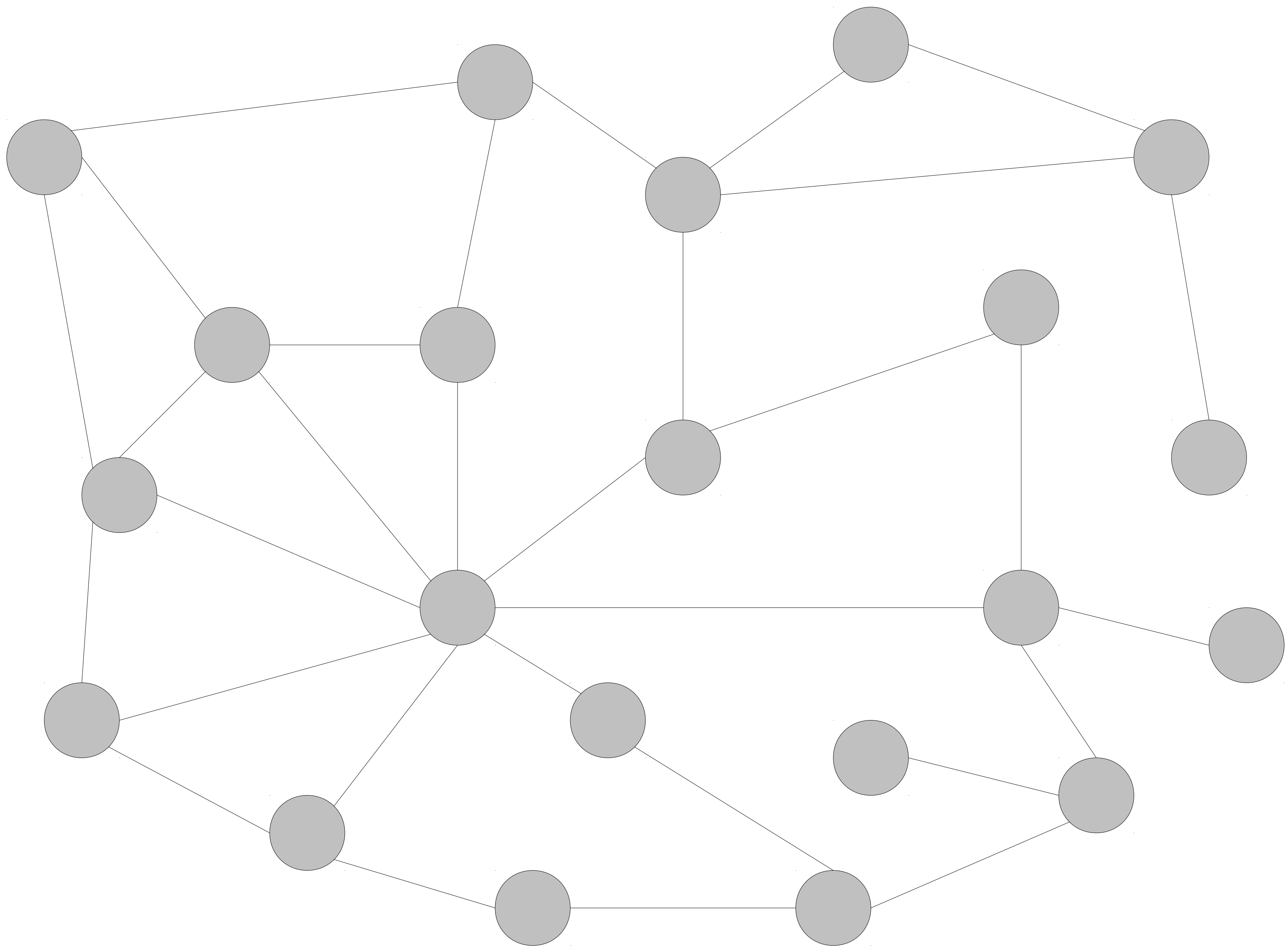




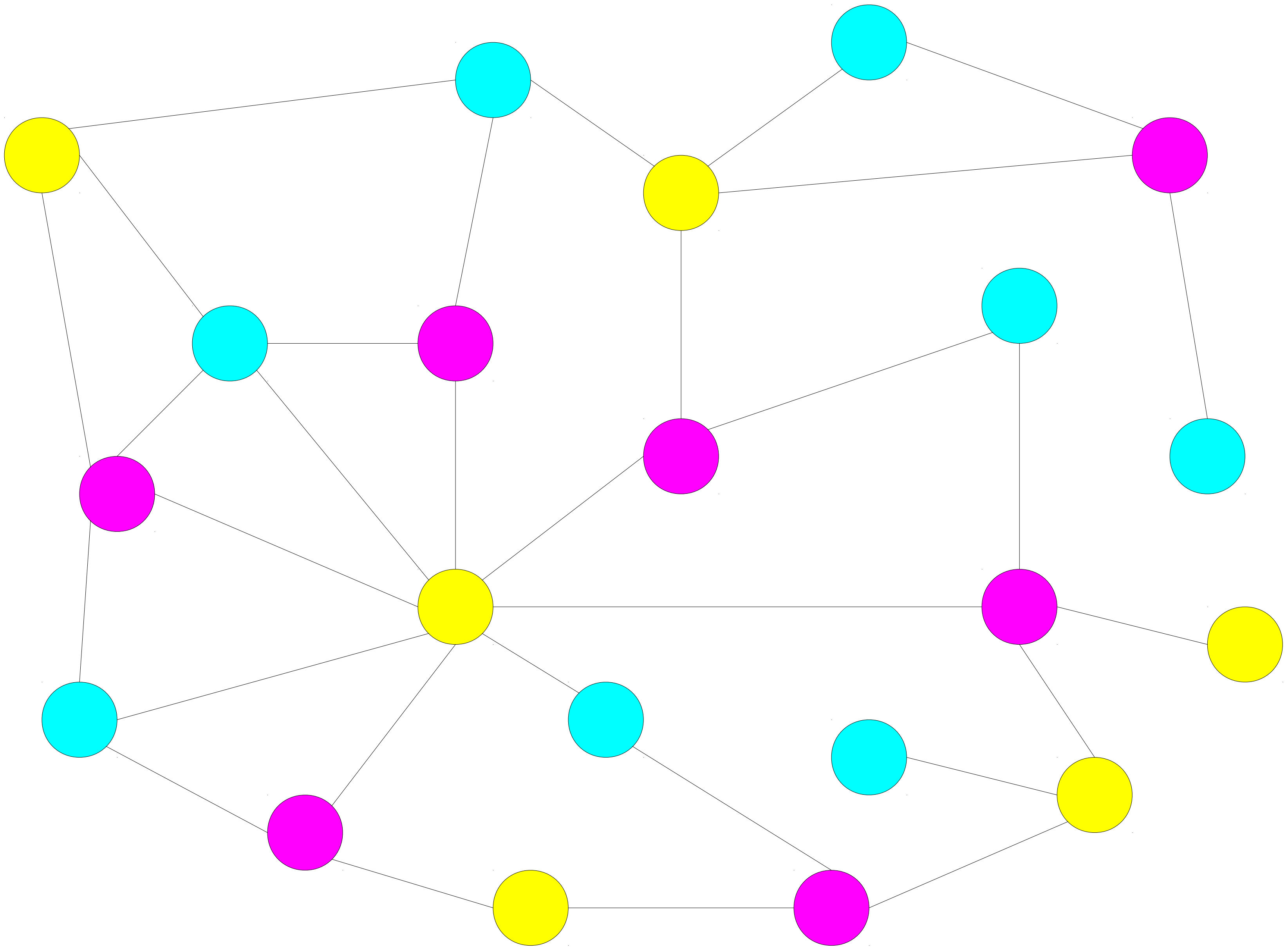


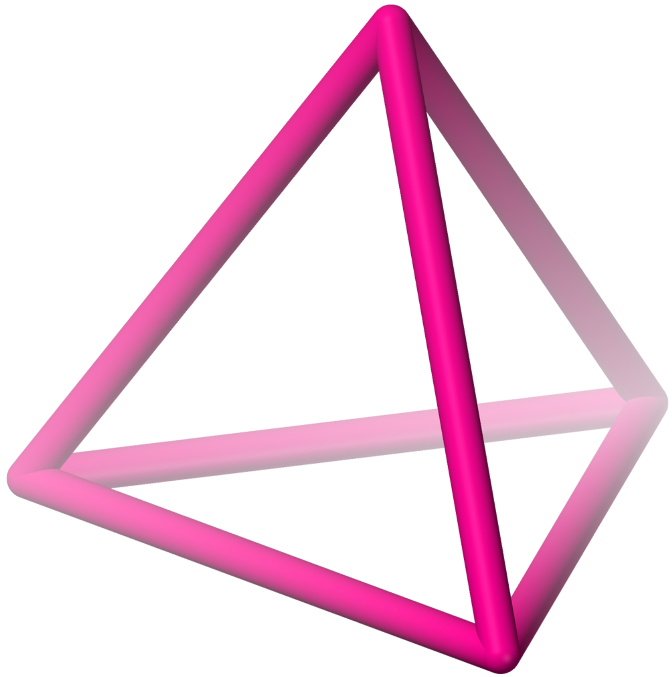


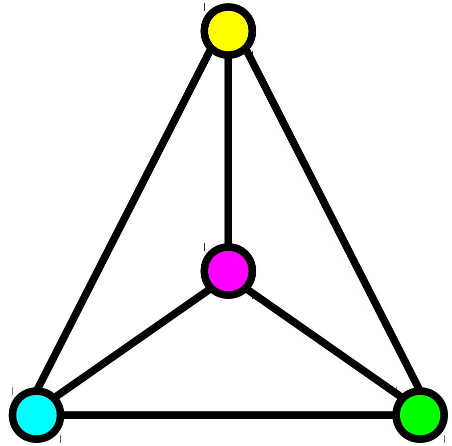
Graph Coloring

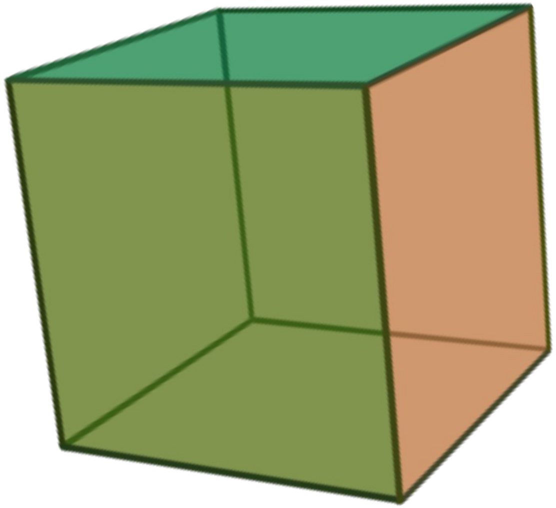


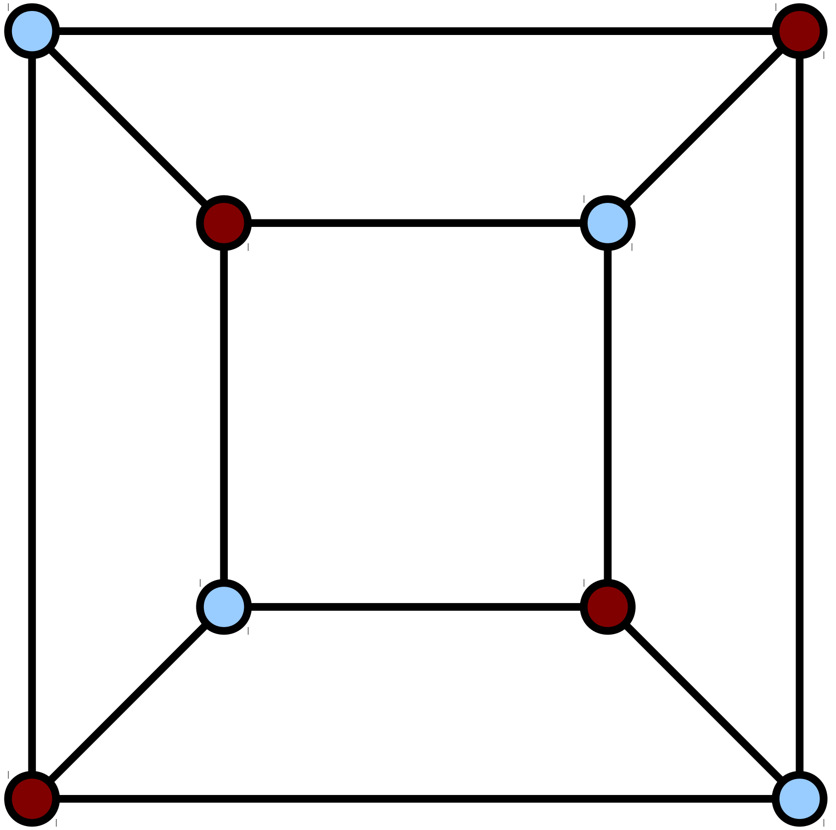
Graph Coloring











A Fantastic Video on a Cool Theorem:

https://youtu.be/-9OUyo8NFZg

Graph Coloring

- Intuitively, a ***k-coloring*** of a graph $G = (V, E)$ is a way to color each node in V one of k different colors such that no two adjacent nodes in V are the same color.

Graph Coloring

- Intuitively, a ***k*-coloring** of a graph $G = (V, E)$ is a way to color each node in V one of k different colors such that no two adjacent nodes in V are the same color.
- Formally, a ***k*-coloring** of a graph $G = (V, E)$ is a function

$$f : V \to \{1, 2, \ldots, k\}$$

such that

$$\forall u \in V. \forall v \in V. (\{u, v\} \in E \to f(u) \neq f(v))$$

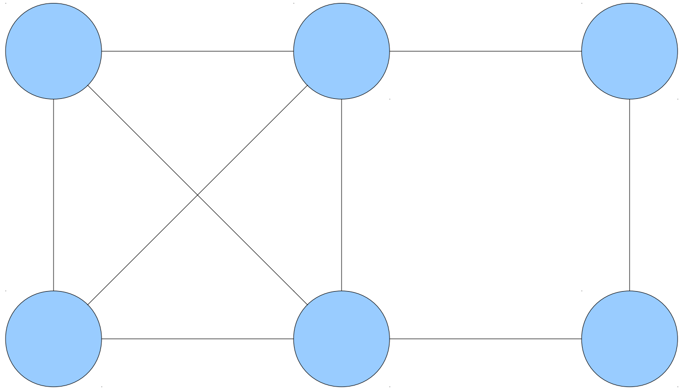
Graph Coloring

- Intuitively, a ***k-coloring*** of a graph $G = (V, E)$ is a way to color each node in V one of k different colors such that no two adjacent nodes in V are the same color.
- Formally, a ***k-coloring*** of a graph $G = (V, E)$ is a function

$$f : V \to \{1, 2, \ldots, k\}$$

such that

$$\forall u \in V.\ \forall v \in V.\ (\{u, v\} \in E \to f(u) \neq f(v))$$



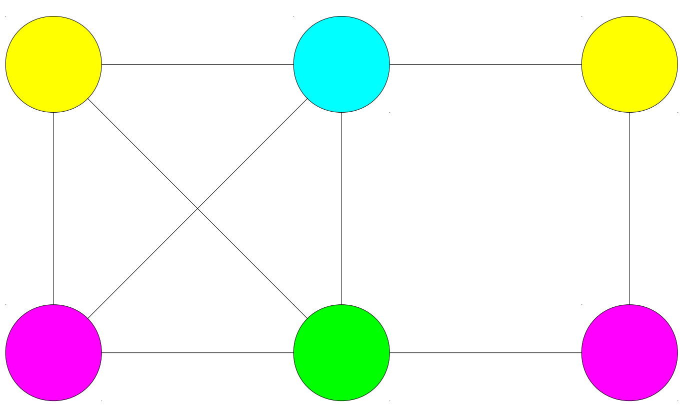
Graph Coloring

- Intuitively, a ***k-coloring*** of a graph $G = (V, E)$ is a way to color each node in V one of k different colors such that no two adjacent nodes in V are the same color.
- Formally, a ***k-coloring*** of a graph $G = (V, E)$ is a function

$$f : V \to \{1, 2, \ldots, k\}$$

such that

$$\forall u \in V. \forall v \in V. (\{u, v\} \in E \to f(u) \neq f(v))$$



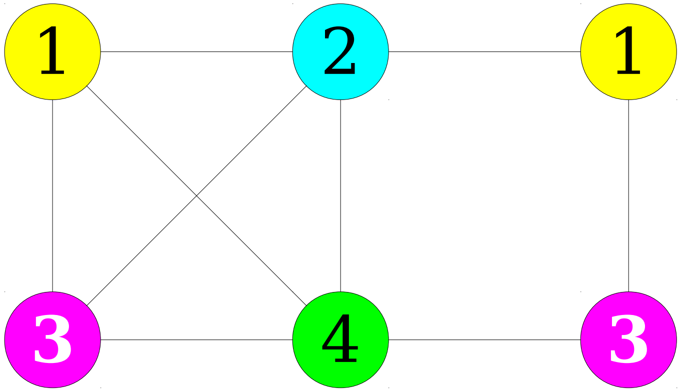
Graph Coloring

- Intuitively, a ***k*-coloring** of a graph $G = (V, E)$ is a way to color each node in V one of k different colors such that no two adjacent nodes in V are the same color.
- Formally, a ***k*-coloring** of a graph $G = (V, E)$ is a function

$$f : V \to \{1, 2, \ldots, k\}$$

such that

$$\forall u \in V.\ \forall v \in V.\ (\{u, v\} \in E \to f(u) \neq f(v))$$



Graph Coloring

- Intuitively, a ***k*-coloring** of a graph $G = (V, E)$ is a way to color each node in V one of k different colors such that no two adjacent nodes in V are the same color.
- Formally, a ***k*-coloring** of a graph $G = (V, E)$ is a function

$$f : V \to \{1, 2, \ldots, k\}$$

such that

$$\forall u \in V.\ \forall v \in V.\ (\{u, v\} \in E \to f(u) \neq f(v))$$

Although this is the formal definition of a *k*-coloring, you rarely see it used in proofs. It's more common to just talk about assigning colors to nodes. However, this definition is super useful if you want to write programs to reason about graph colorings!

Graph Coloring

- Intuitively, a ***k-coloring*** of a graph $G = (V, E)$ is a way to color each node in V one of k different colors such that no two adjacent nodes in V are the same color.
- Formally, a ***k-coloring*** of a graph $G = (V, E)$ is a function

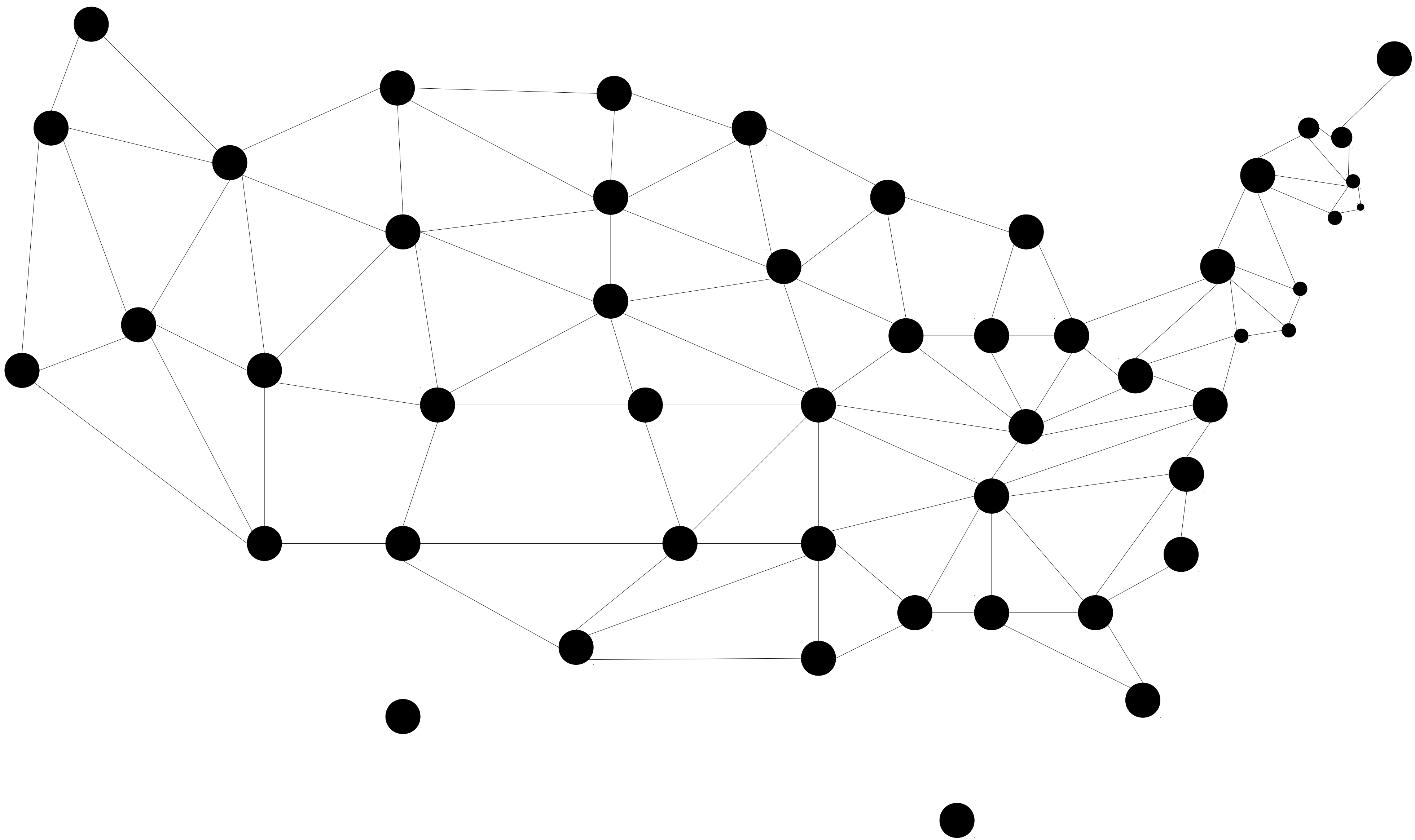
$$f : V \to \{1, 2, \ldots, k\}$$

  such that

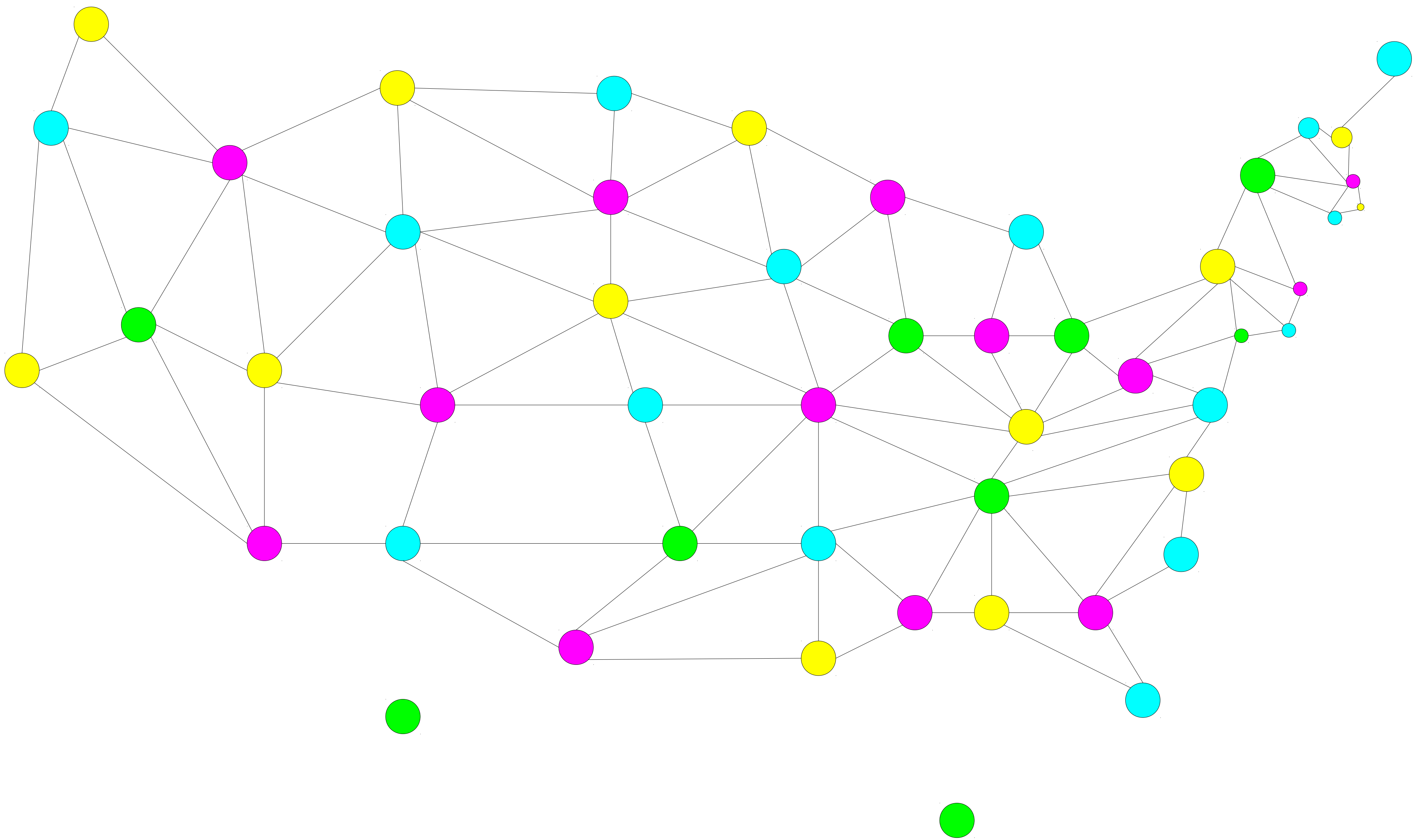
$$\forall u \in V. \forall v \in V. (\{u, v\} \in E \to f(u) \neq f(v))$$

- A graph G is called ***k-colorable*** if a k-coloring of G exists.
- The smallest k for which G is k-colorable is its ***chromatic number***.
  - The chromatic number of a graph G is denoted $\chi(G)$, from the Greek *χρώμα*, meaning “color.”

Graph Coloring



Graph Coloring



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.

Philosophical Question: Is a theorem true if no human has ever read the proof?

Next Time

- ***The Pigeonhole Principle***
  - A simple, powerful, versatile theorem.
- ***Graph Theory Party Tricks***
  - Applying math to graphs of people!
- ***Fair Division Protocols (ITA)***
  - Nifty techniques for divvying things up.