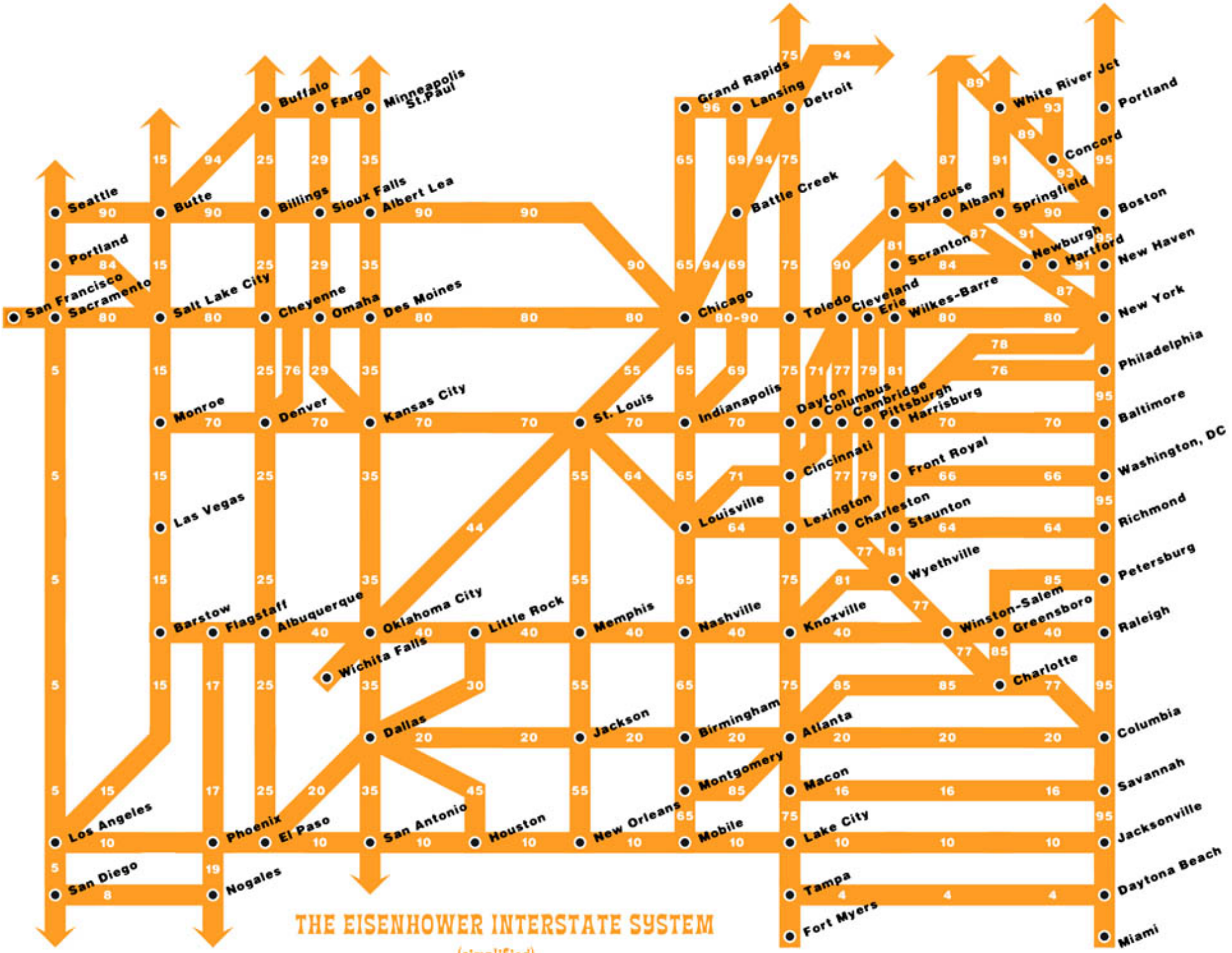
Graph Theory

Part One

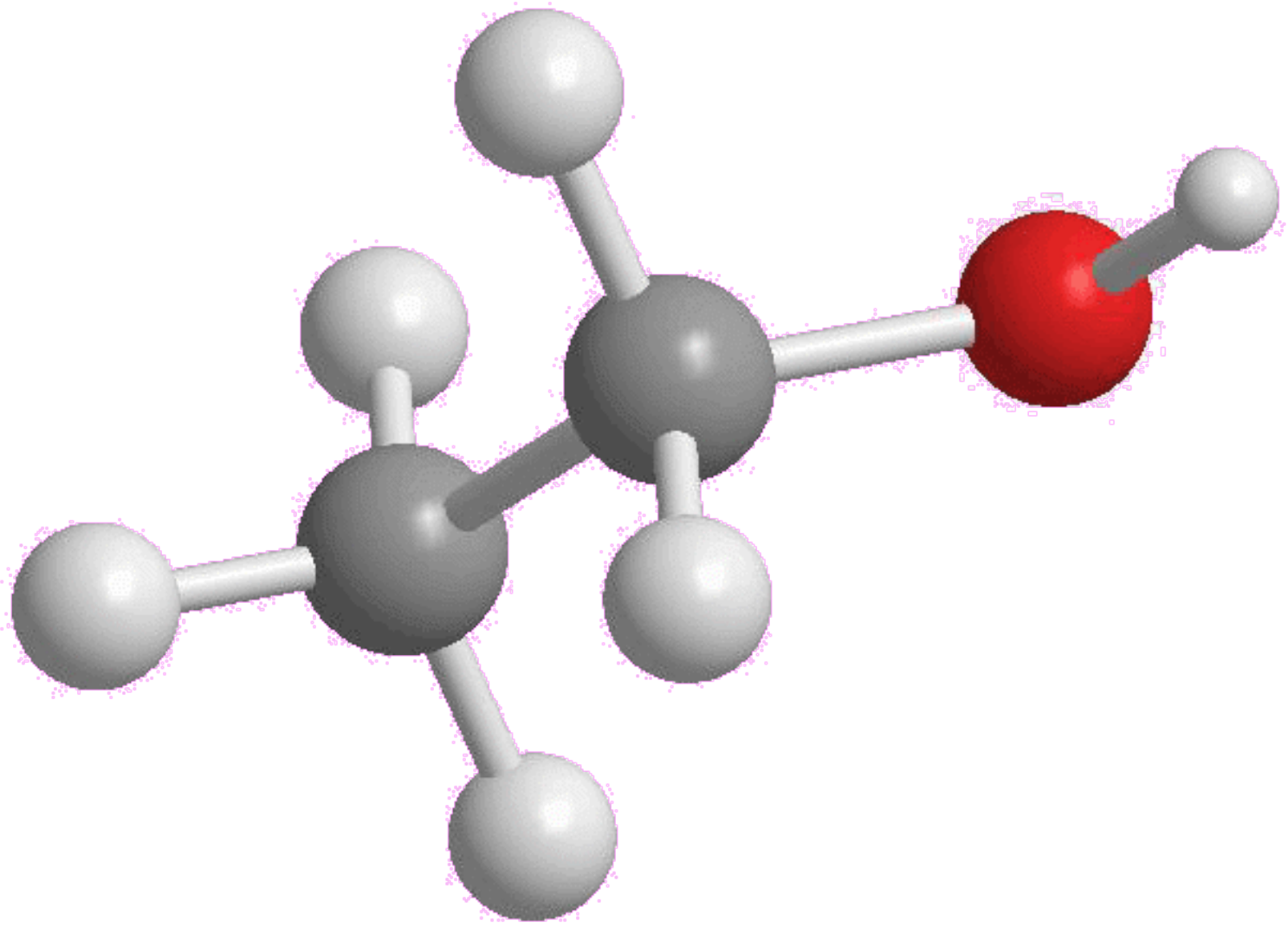
Graph Theory

For those of you who have already completed CS106B/X:

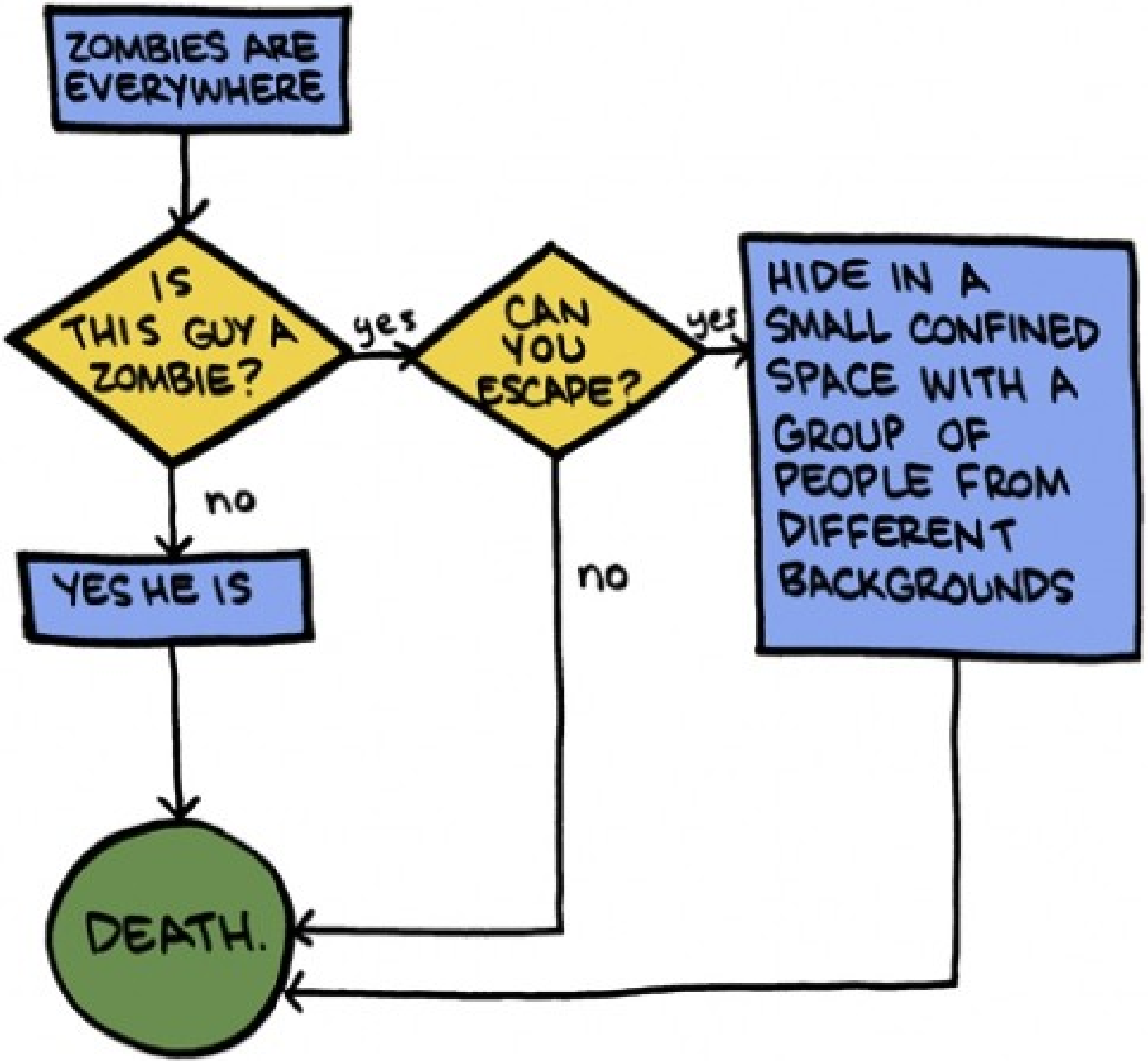


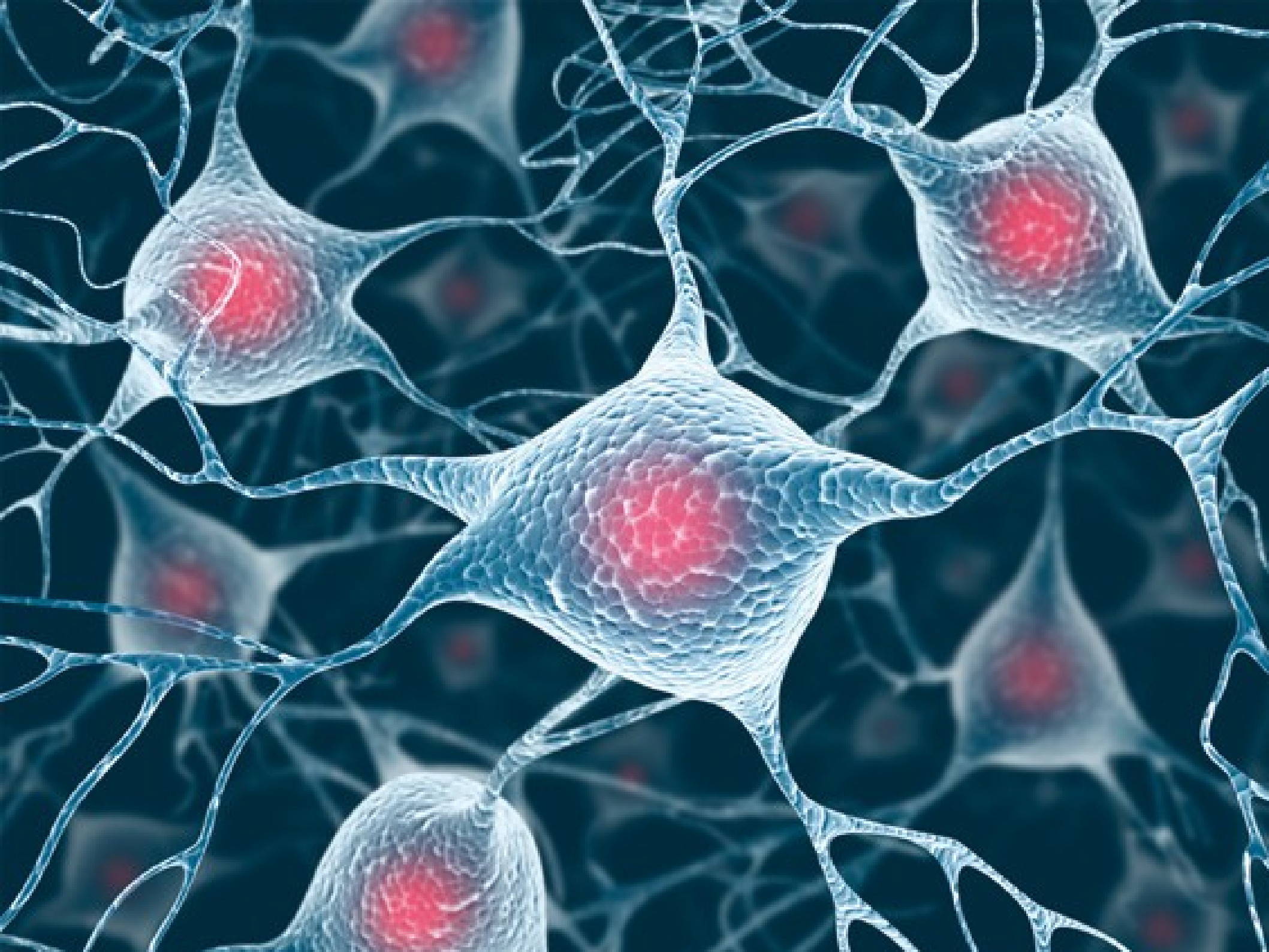
http://strangemaps.files.wordpress.com/2007/02/fullinterstatemap-web.jpg

Chemical Bonds



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



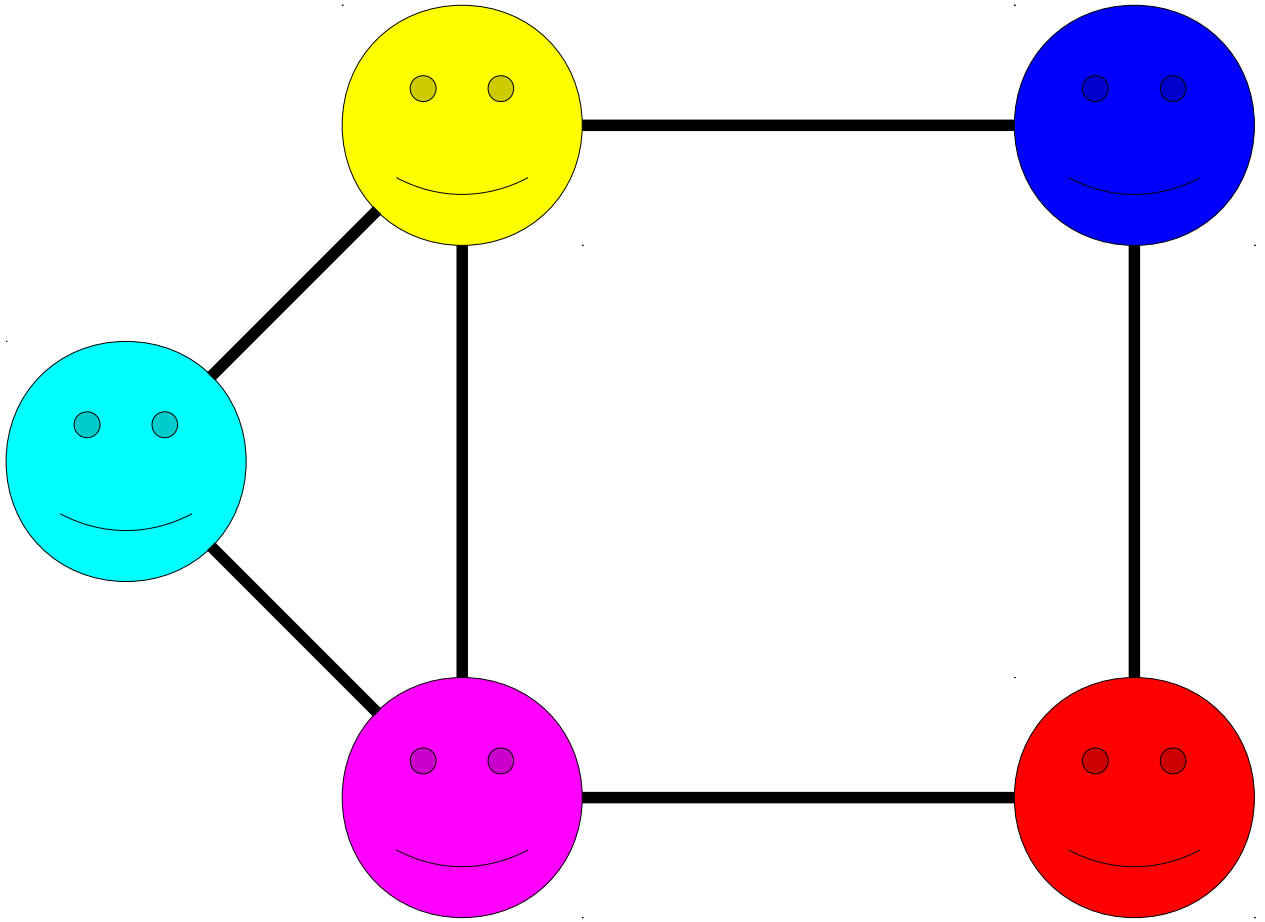



facebook®

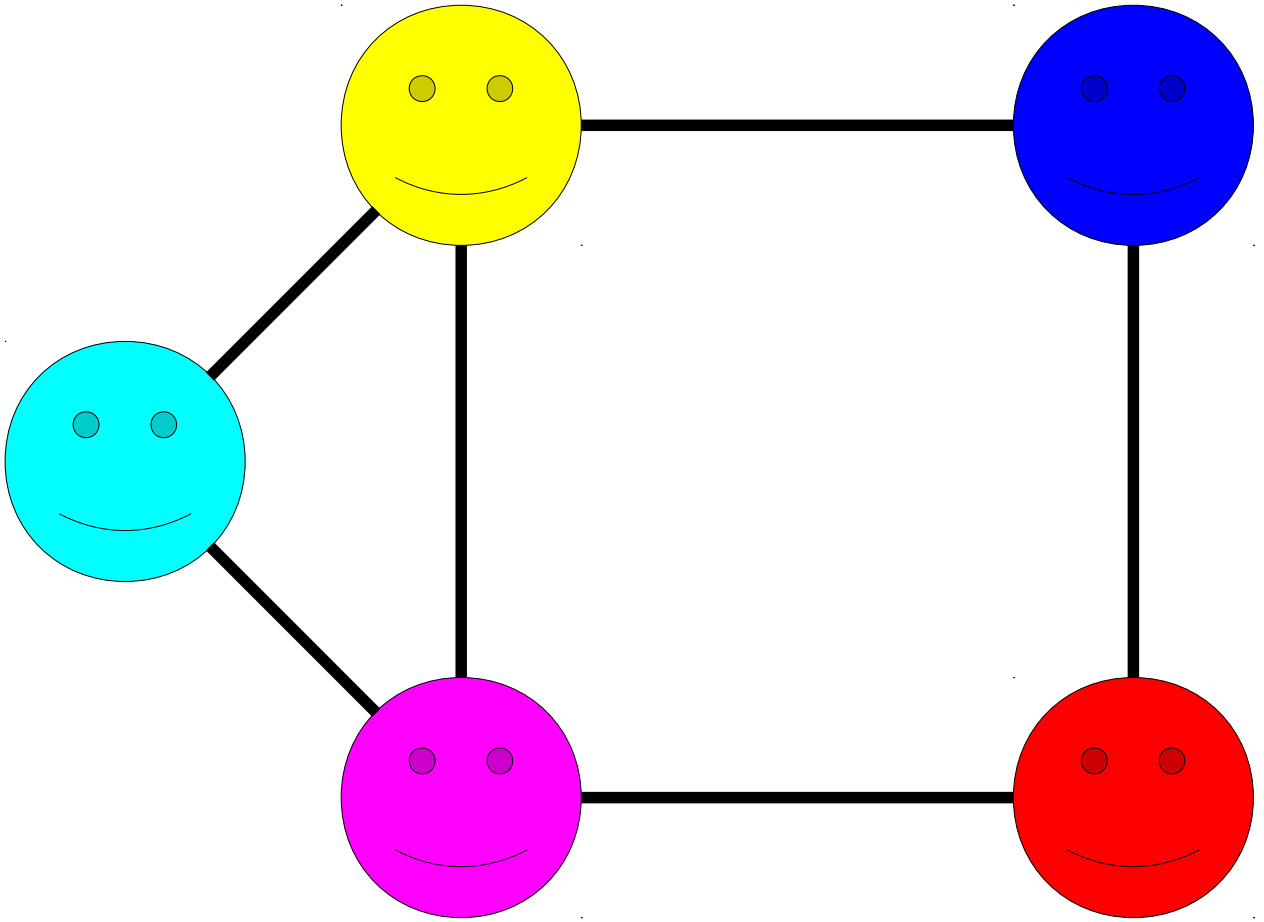
What's in Common

- 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.

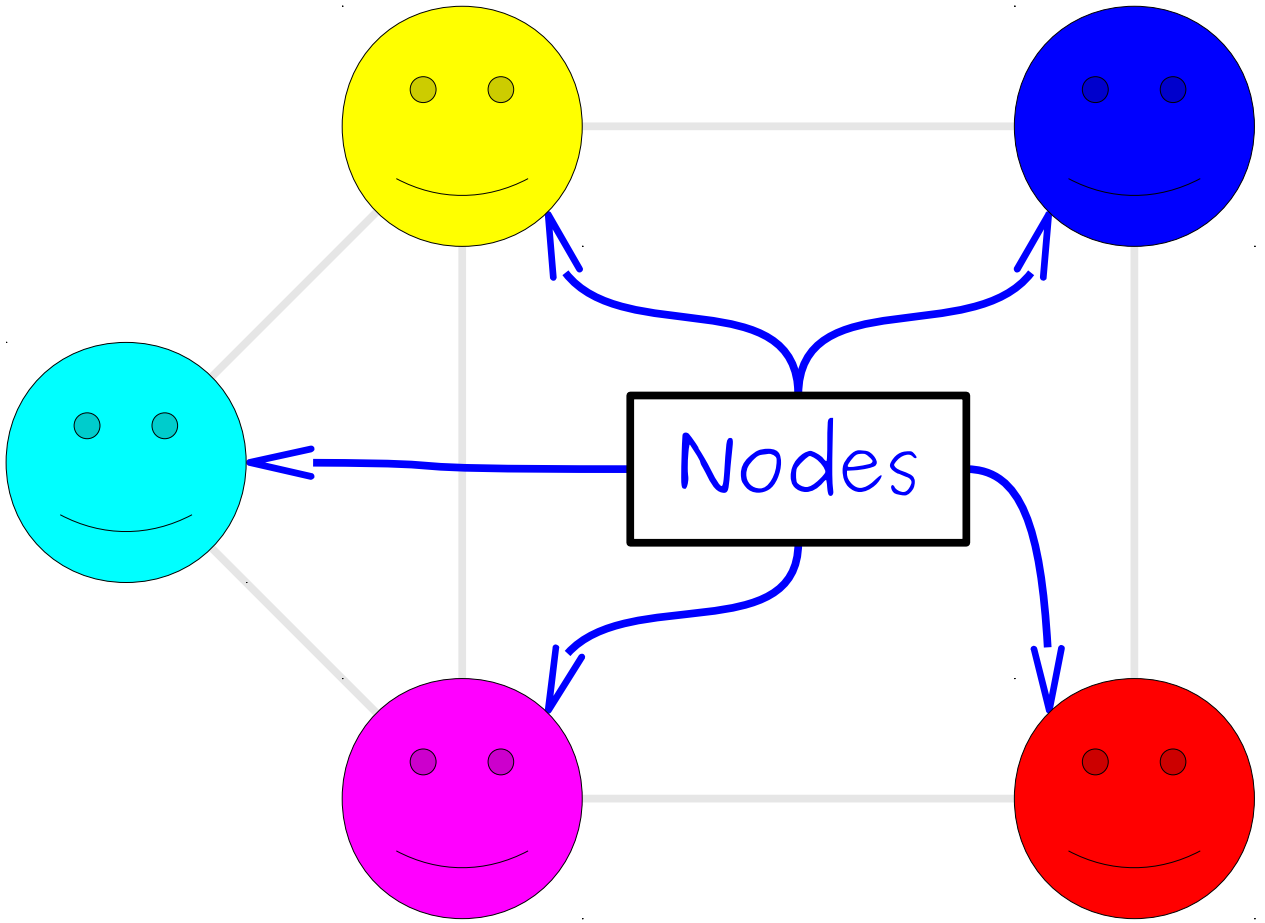


A ***graph*** is a mathematical structure for representing relationships.



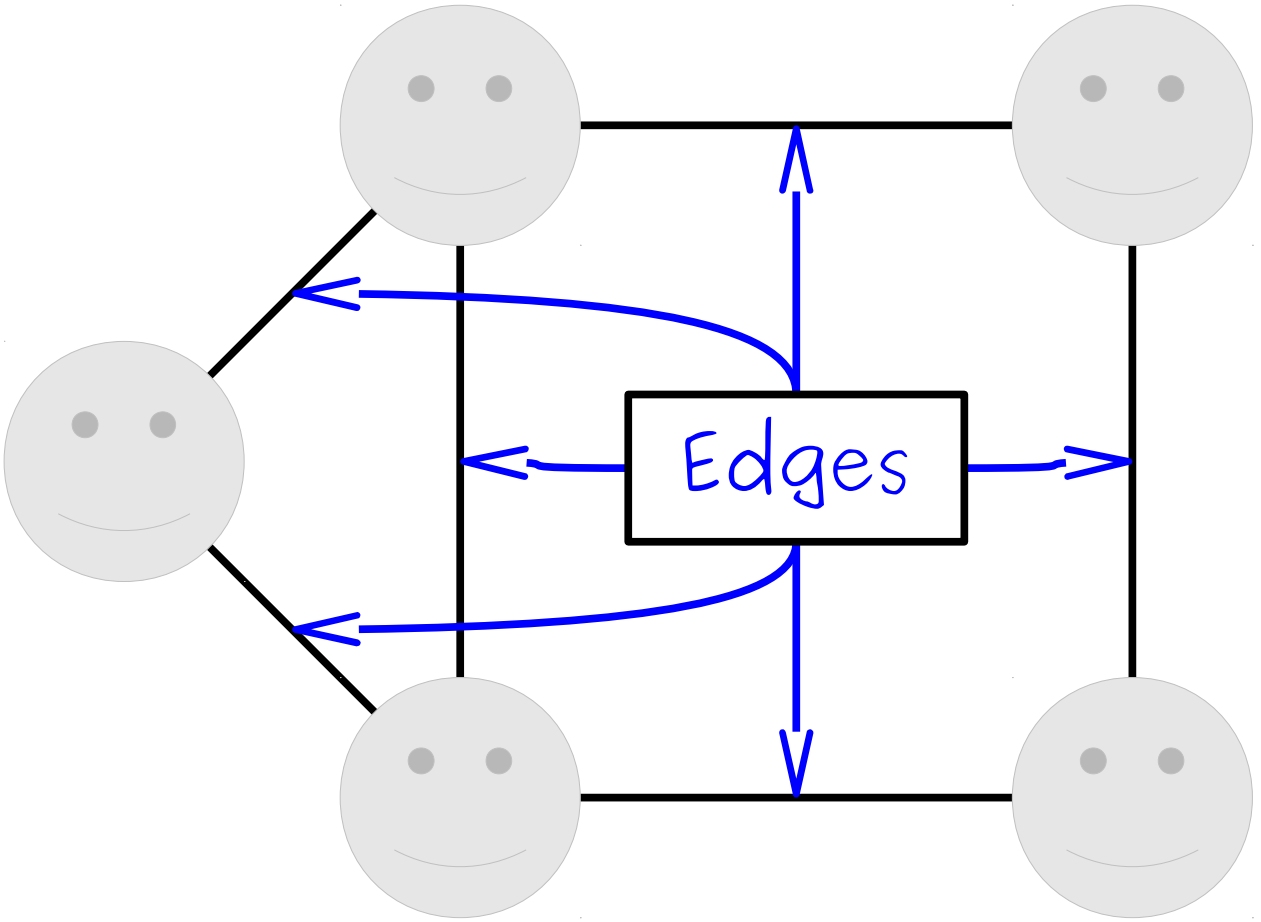
A graph consists of a set of ***nodes*** (or ***vertices***) connected by ***edges*** (or ***arcs***)

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



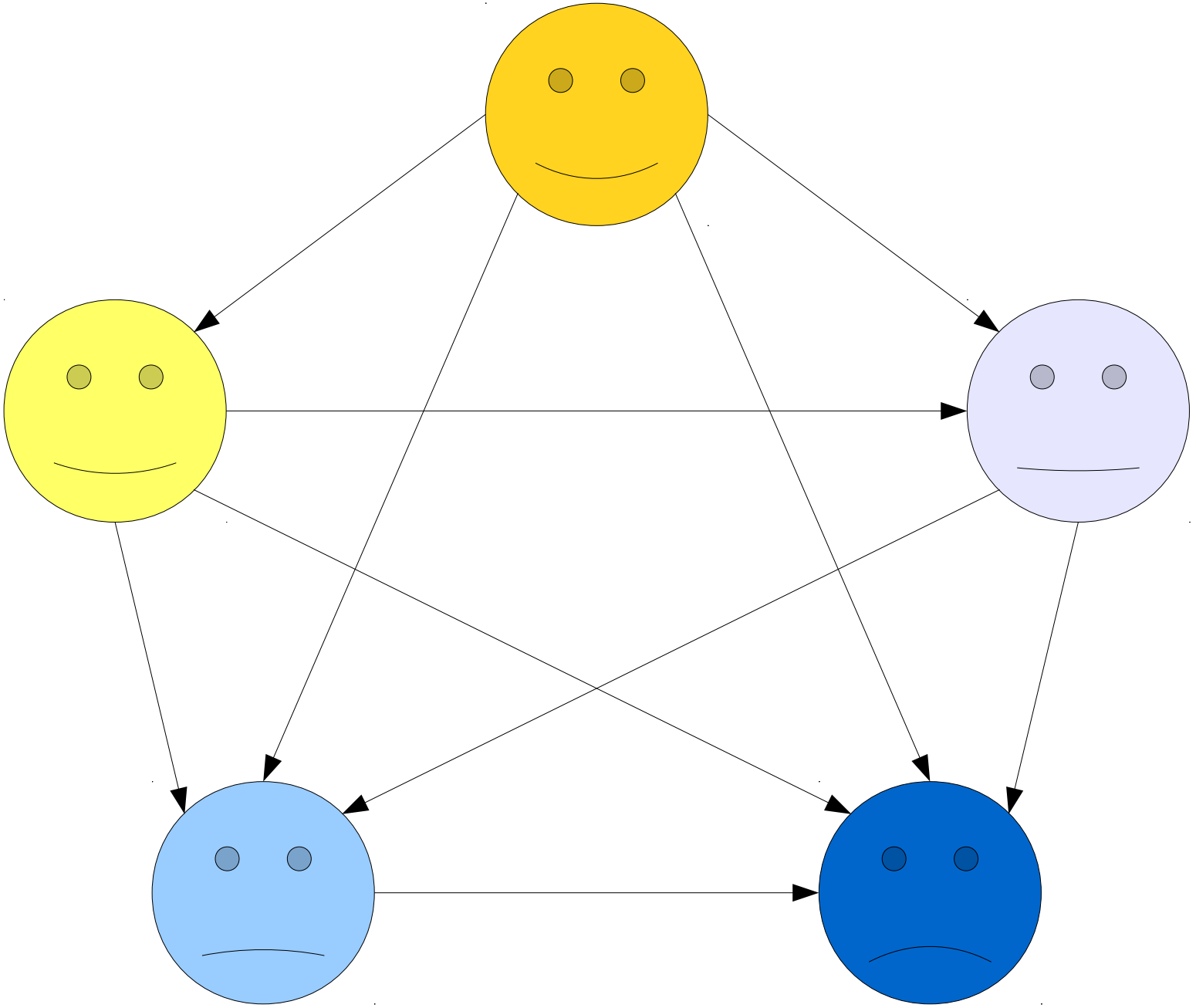
A graph consists of a set of ***nodes*** (or ***vertices***) connected by ***edges*** (or ***arcs***)

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

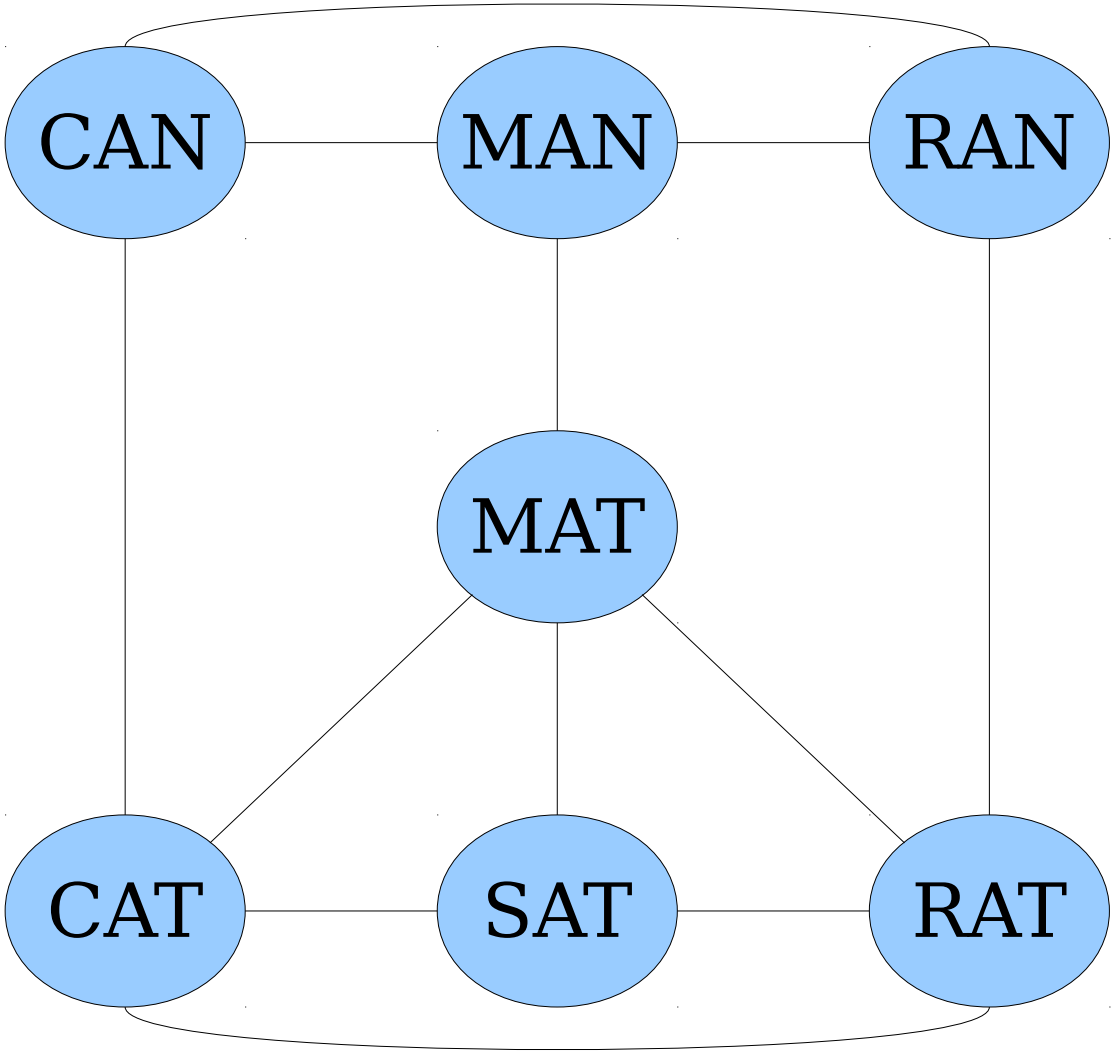


A graph consists of a set of ***nodes*** (or ***vertices***) connected by ***edges*** (or ***arcs***)

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 with a finite number of nodes, unless specified otherwise.

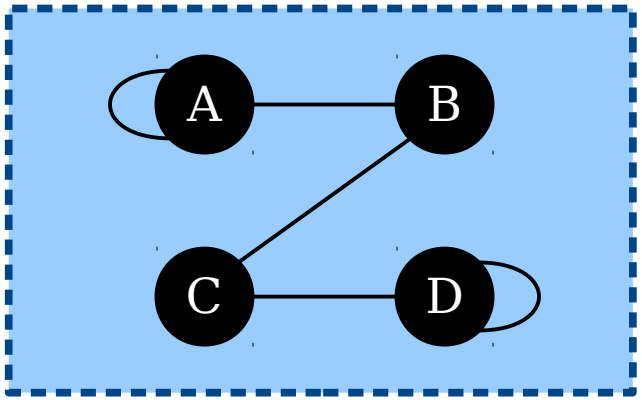
Formalizing Graphs

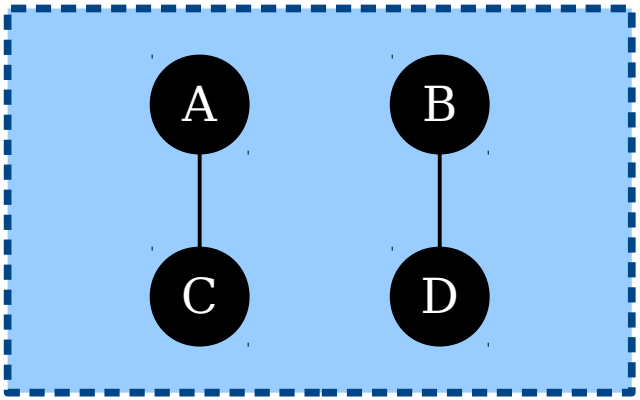
- How might we define a graph mathematically?
- We need to specify
  - what the nodes in the graph are, and
  - which edges are in the graph.
- The nodes can be pretty much anything.
- What about the edges?

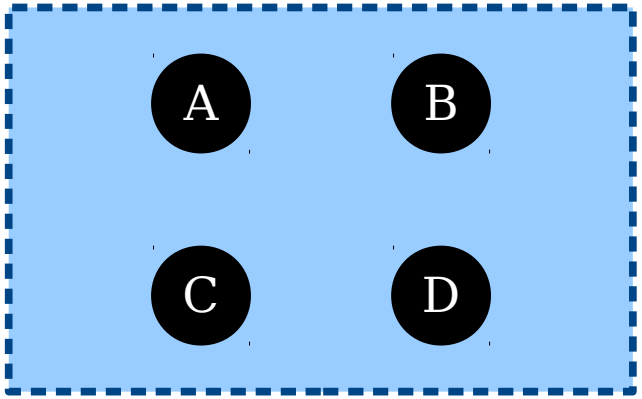
Formalizing Graphs

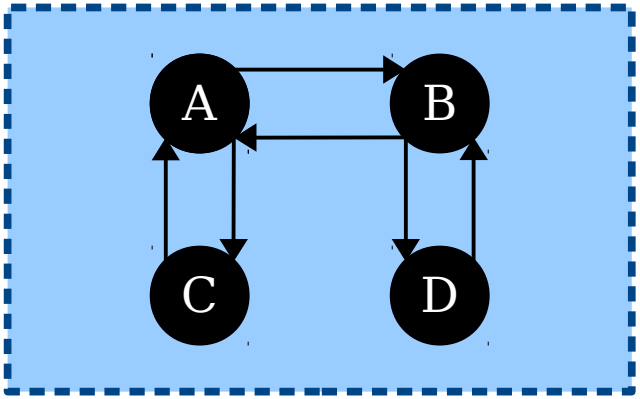
- An ***unordered pair*** is a set $\{a, b\}$ of two elements $a \neq b$. (Remember that sets are unordered).
  - $\{0, 1\} = \{1, 0\}$
- An ***undirected graph*** is an ordered pair $G = (V, E)$, where
  - V is a set of nodes, which can be anything, and
  - E is a set of edges, which are unordered pairs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

- An ***unordered pair*** is a set $\{a, b\}$ of two elements $a \neq b$.
- 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.







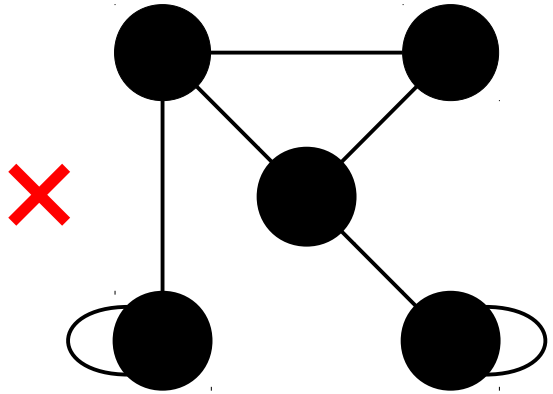


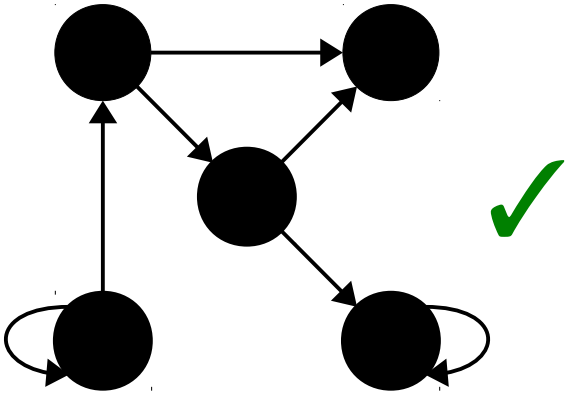
How many of these drawings are of valid undirected graphs?

Answer at **PollEv.com/cs103** or
text **CS103** to **22333** once to join, then a number.

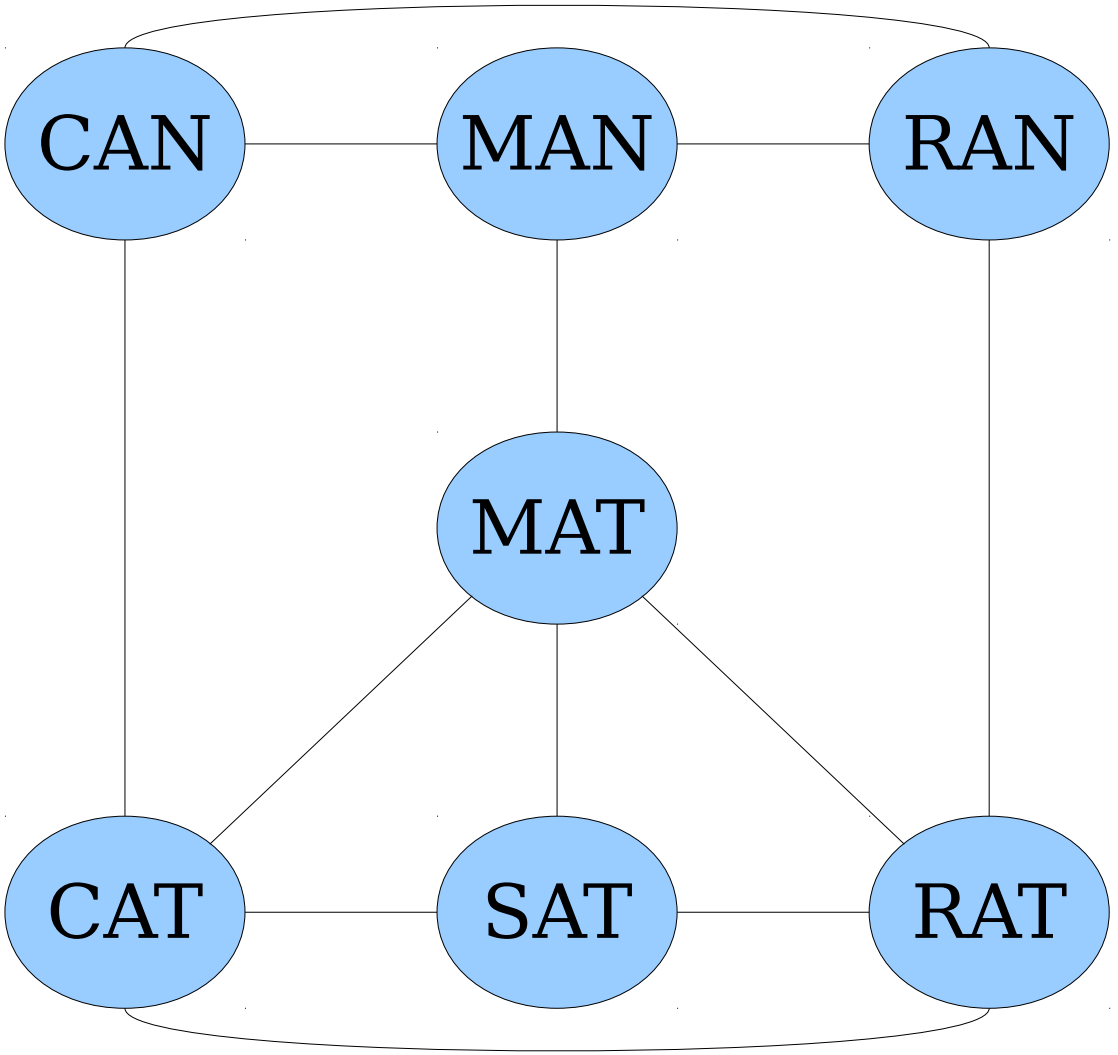
Self-Loops

- An edge from a node to itself is called a ***self-loop***.
- In undirected graphs, self-loops are generally not allowed.
  - Can you see how this follows from the definition?
- In directed graphs, self-loops are generally allowed unless specified otherwise.

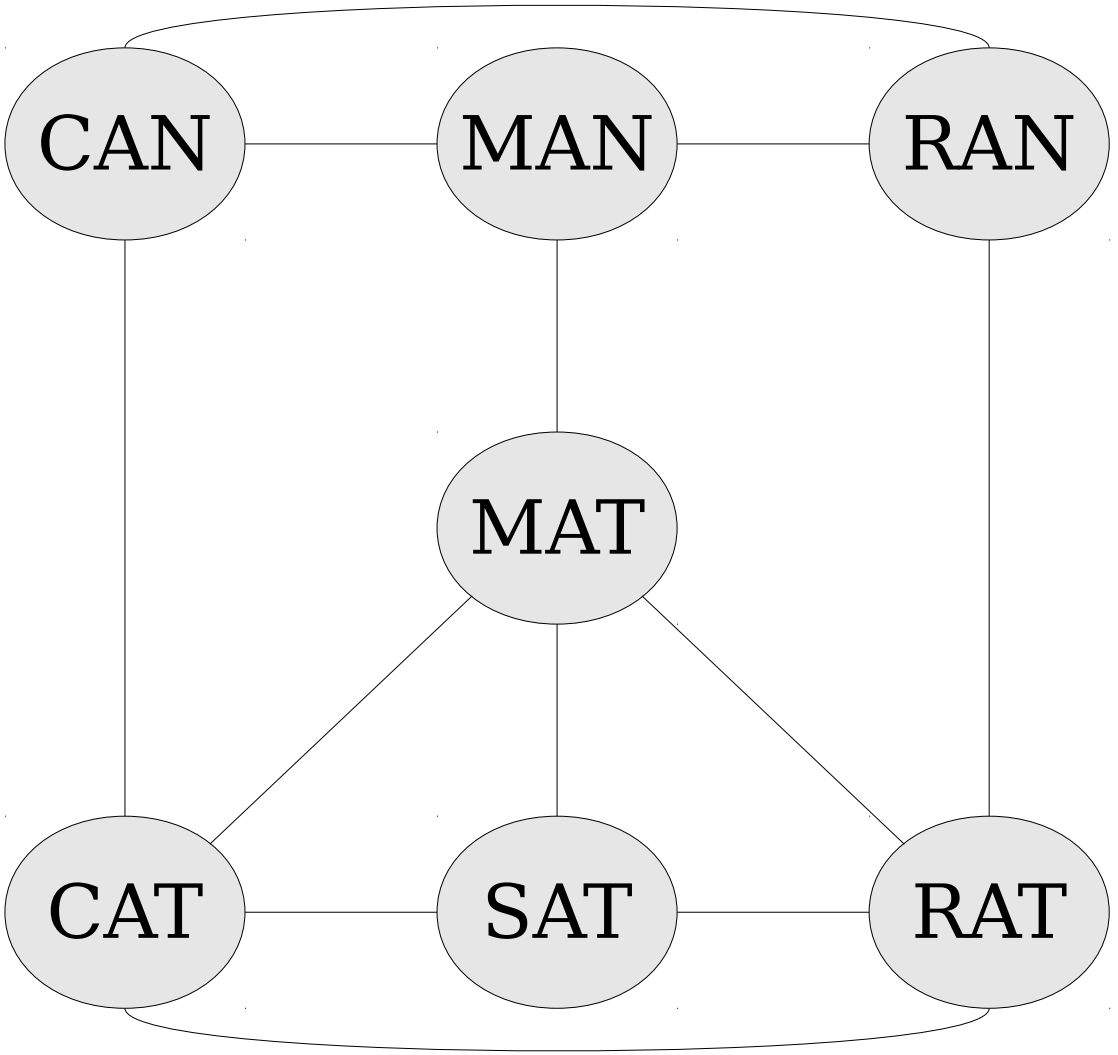




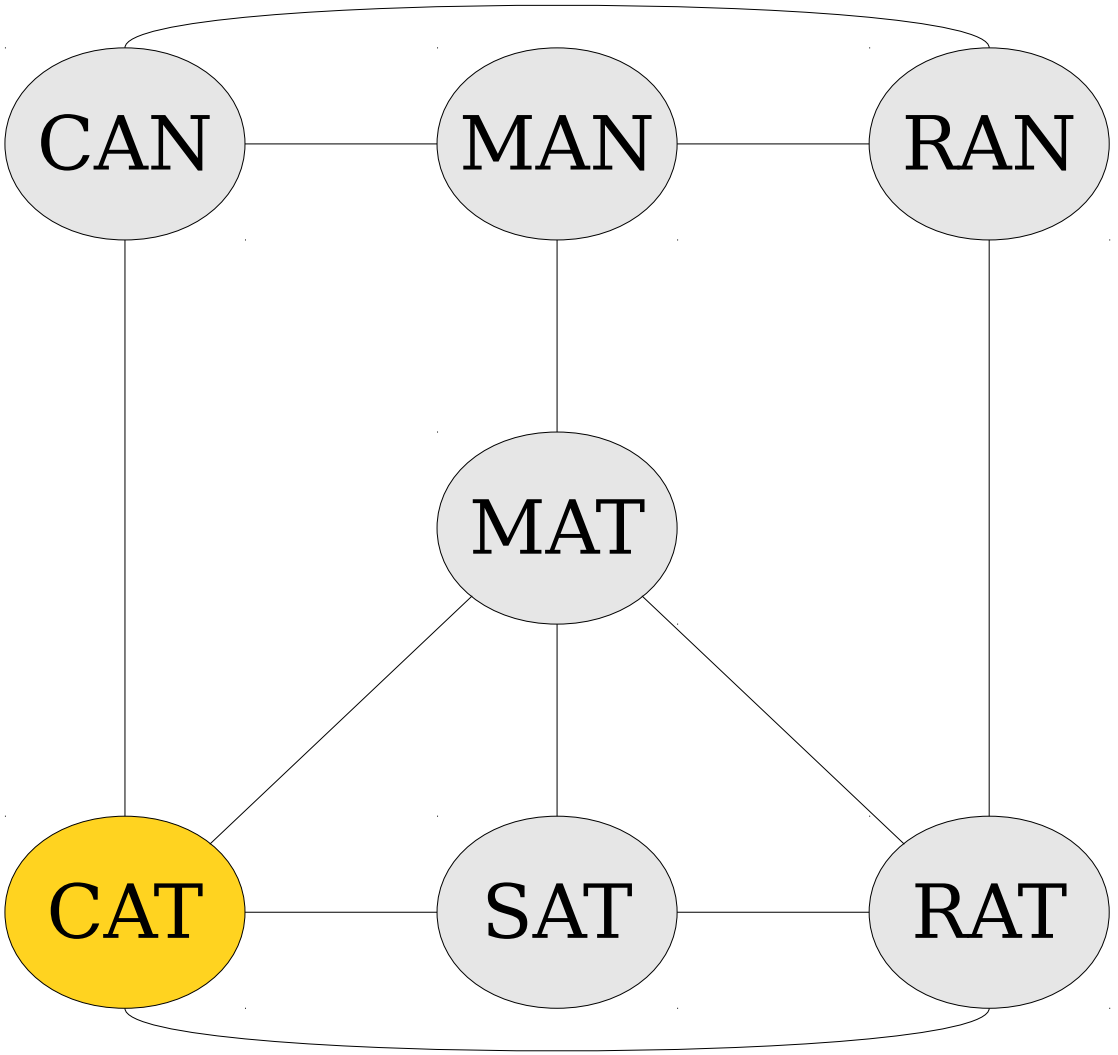
Standard Graph Terminology



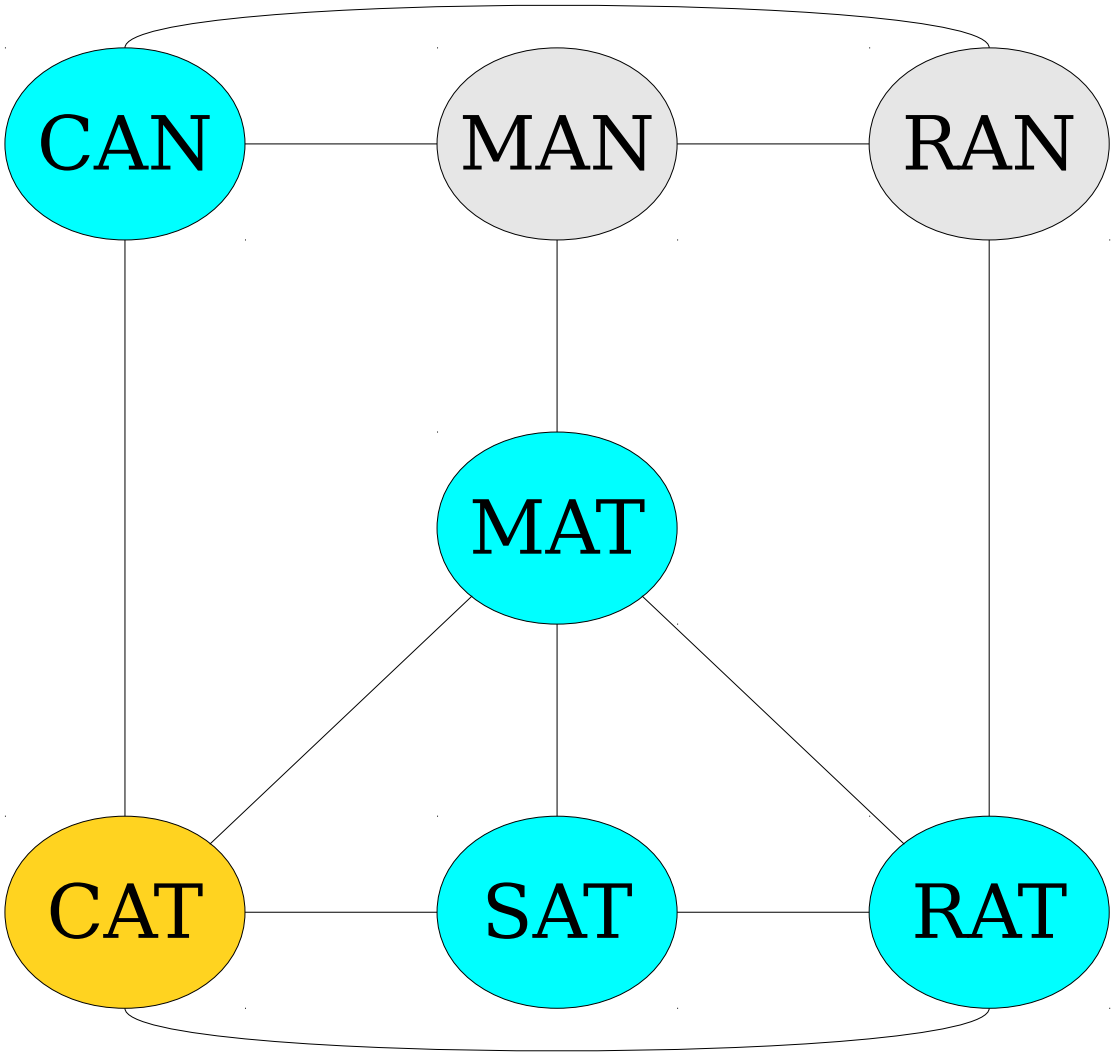
Two nodes are called ***adjacent*** if there is an edge between them.



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



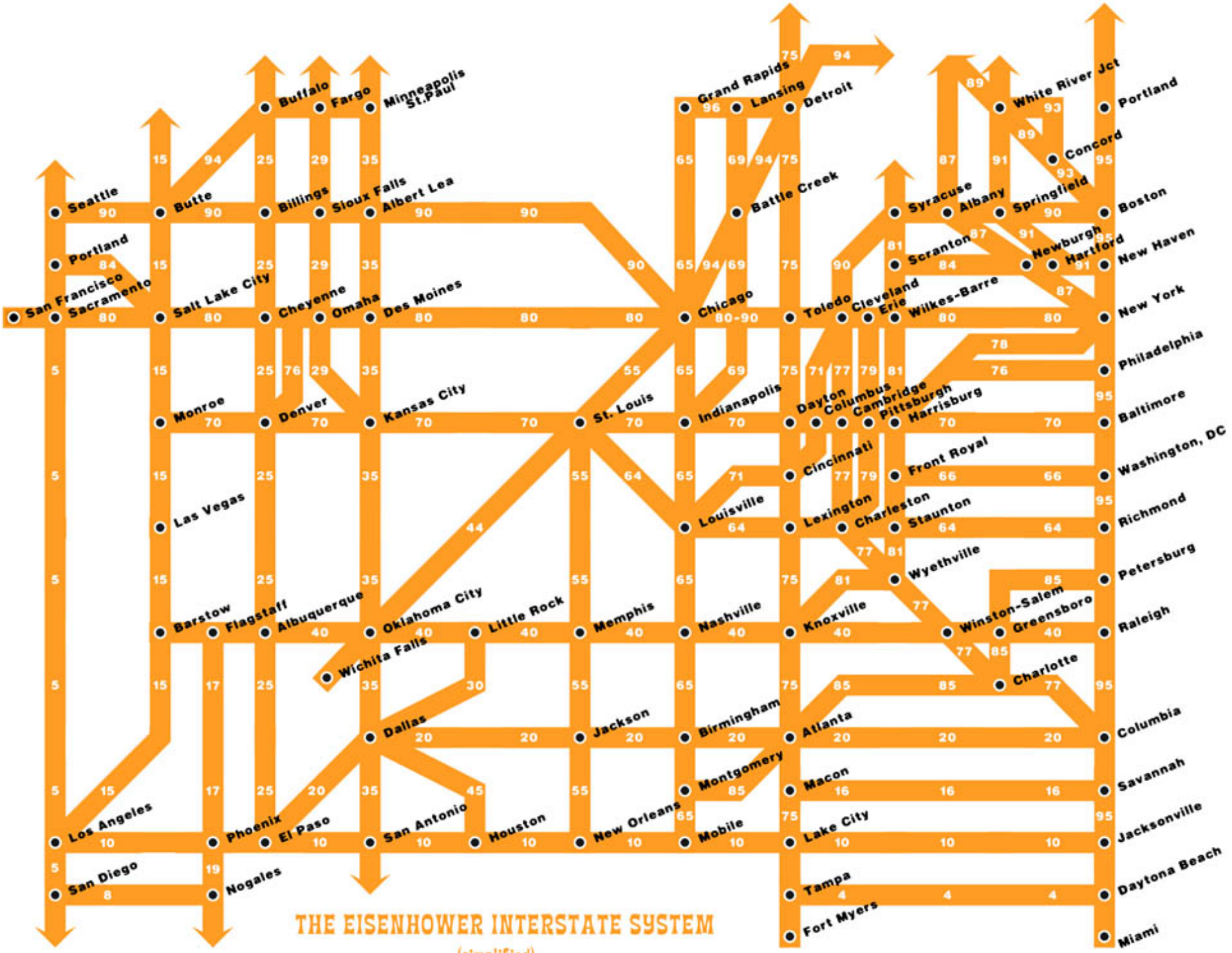
Two nodes are called ***adjacent*** if there is an edge between them.



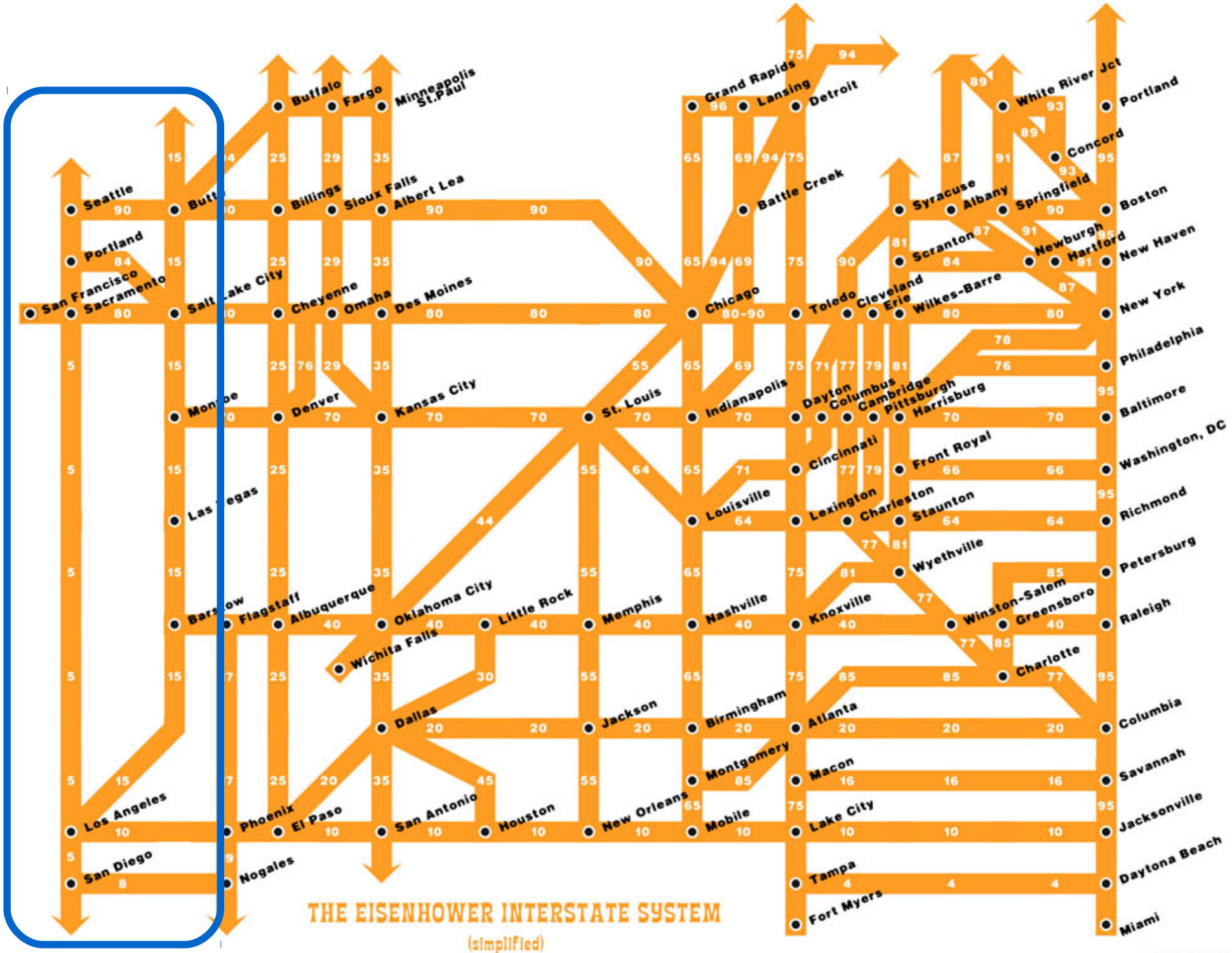
Two nodes are called ***adjacent*** if there is an edge between them.

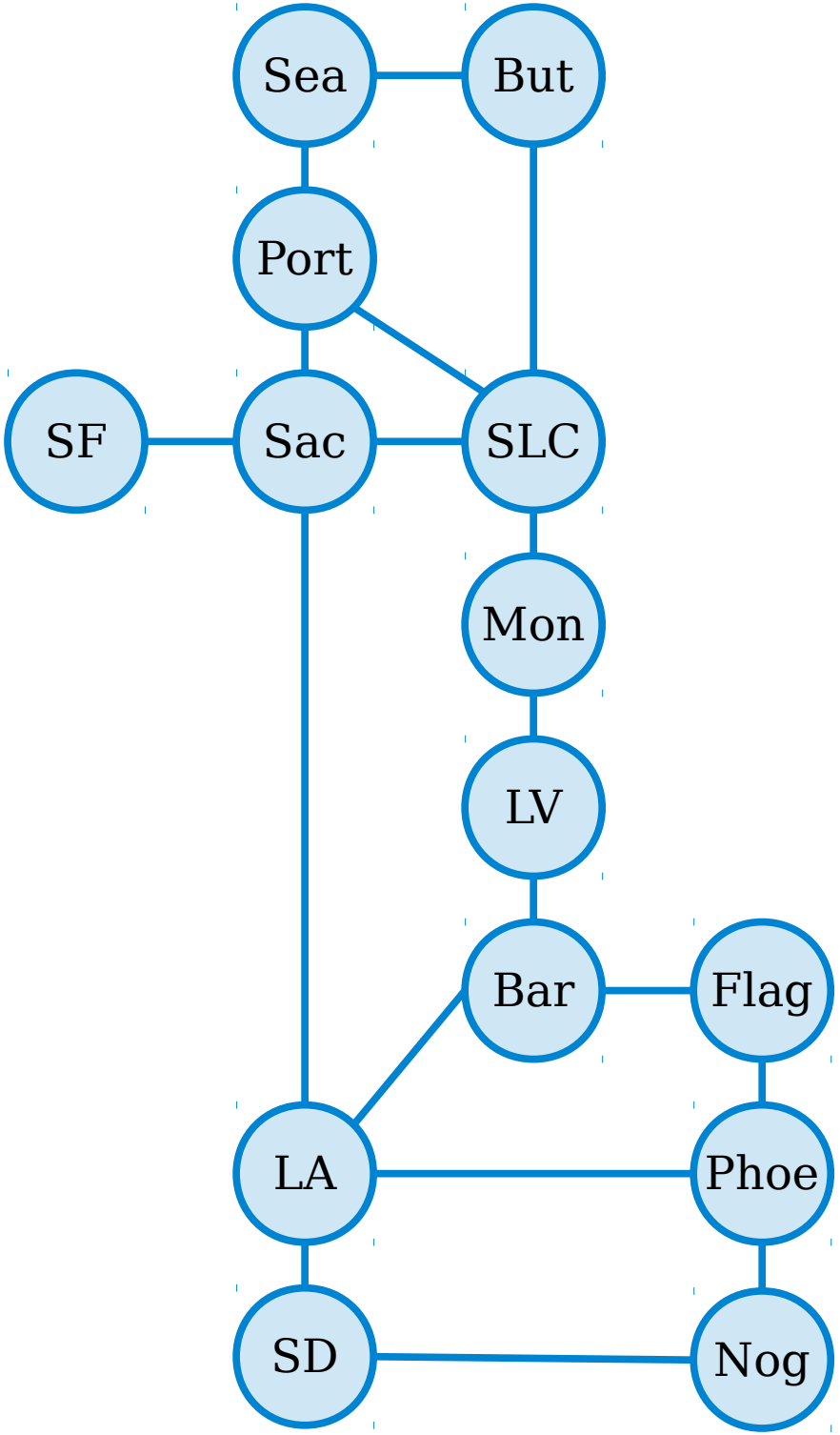
Using our Formalisms

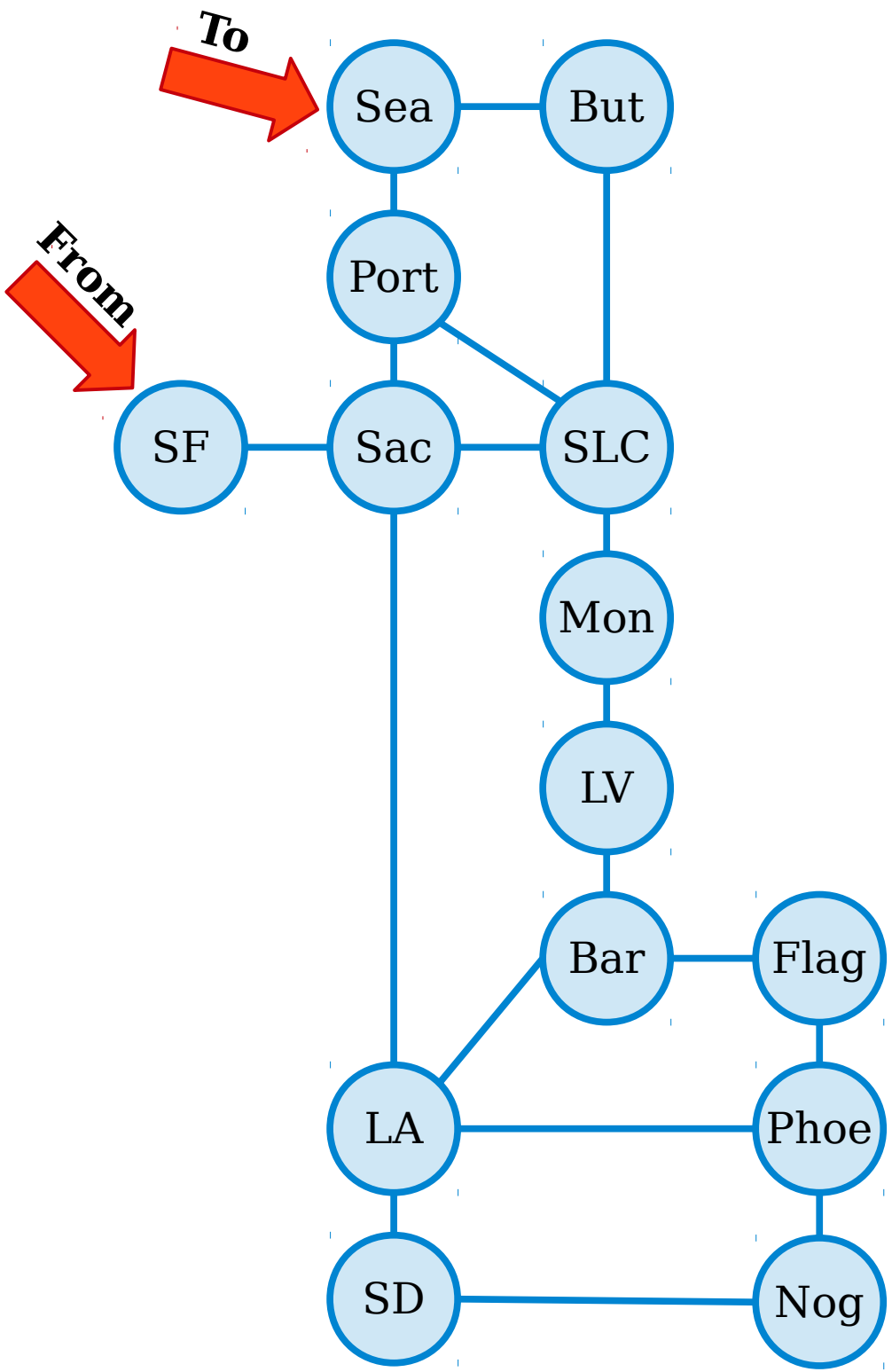
- Let $G = (V, E)$ be a graph.
- Intuitively, two nodes are adjacent if they're linked by an edge.
- Formally speaking, we say that two nodes $u, v \in V$ are ***adjacent*** if $\{u, v\} \in E$.

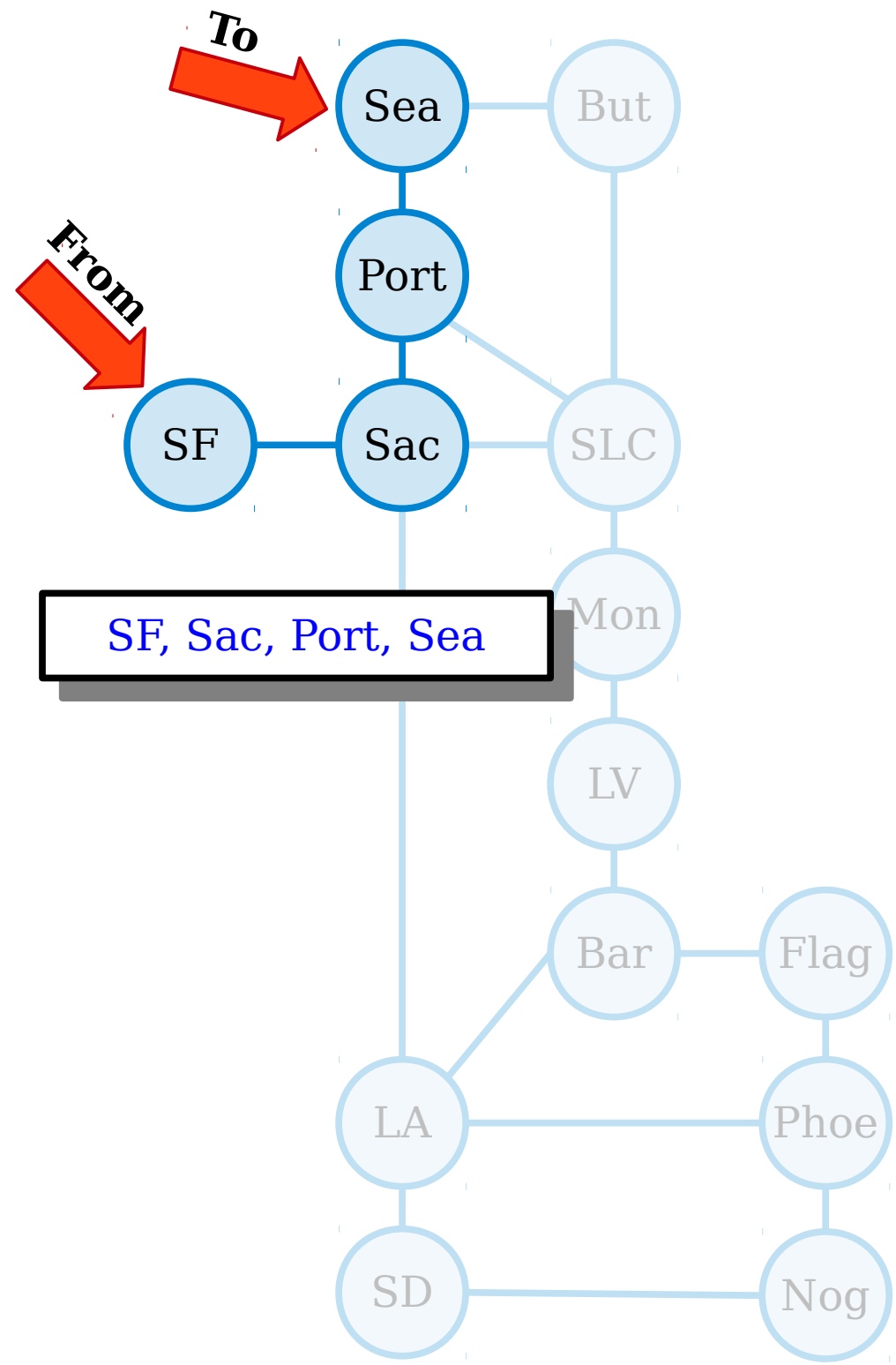


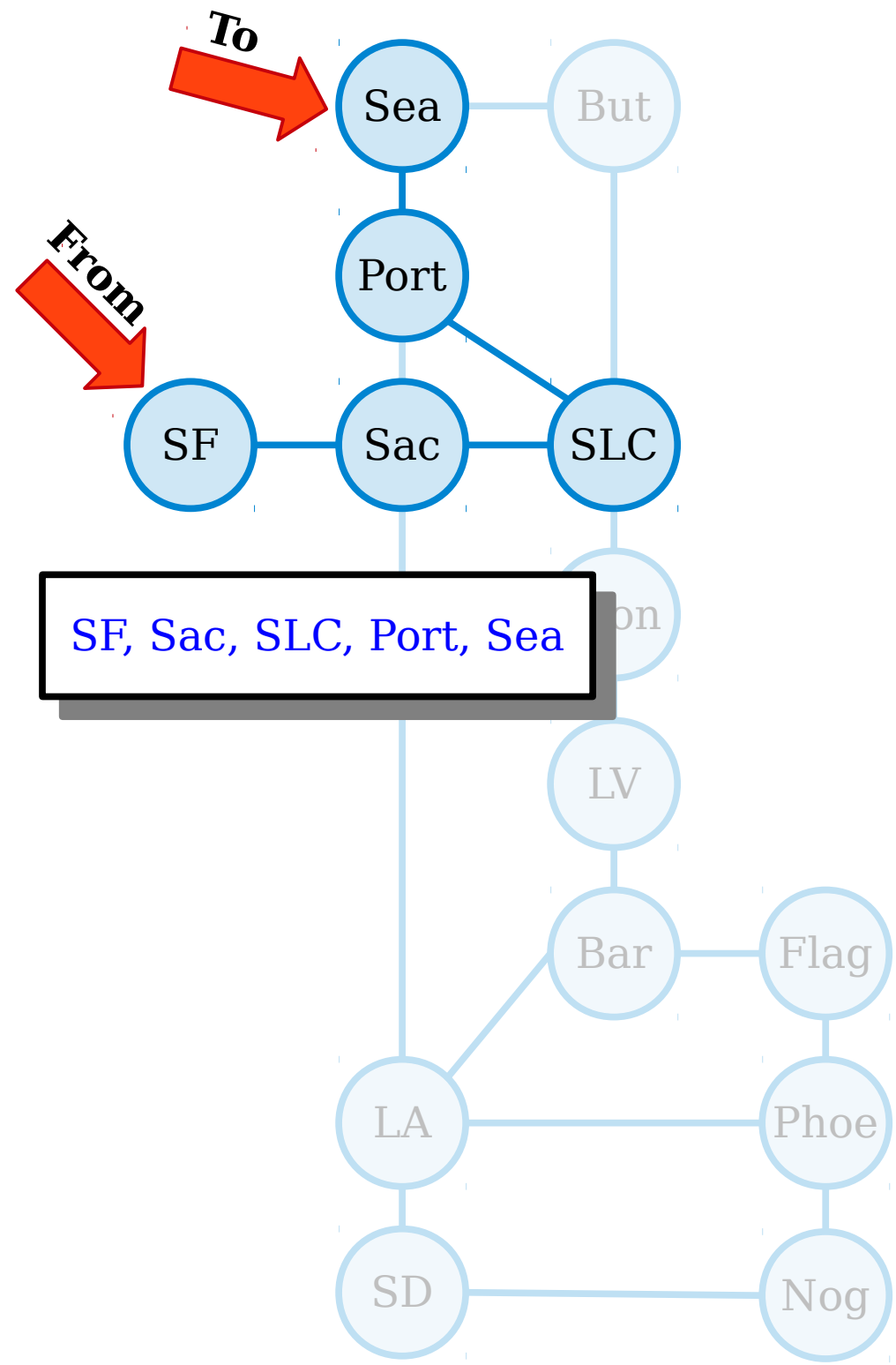
http://strangemaps.files.wordpress.com/2007/02/fullinterstatemap-web.jpg

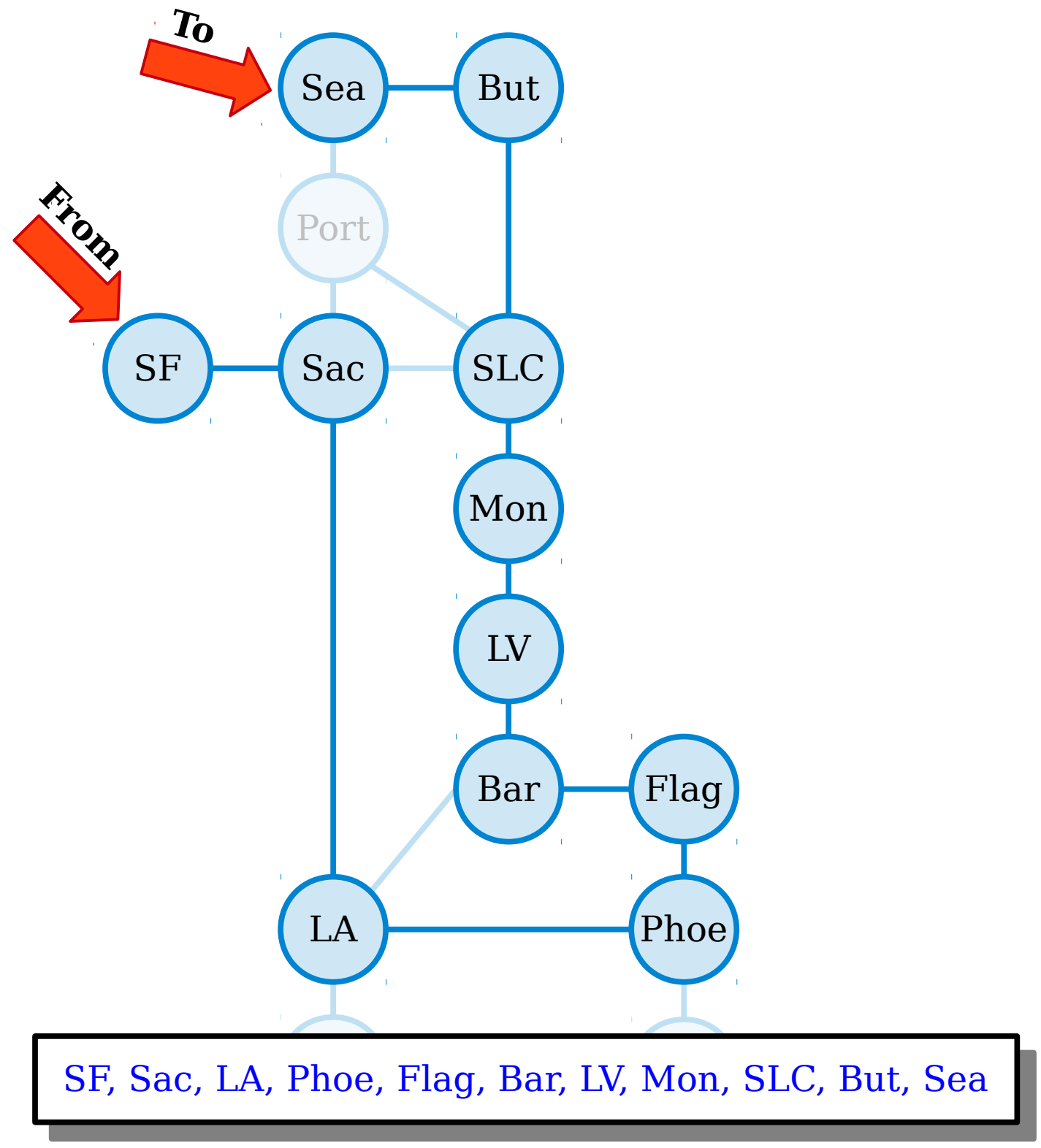


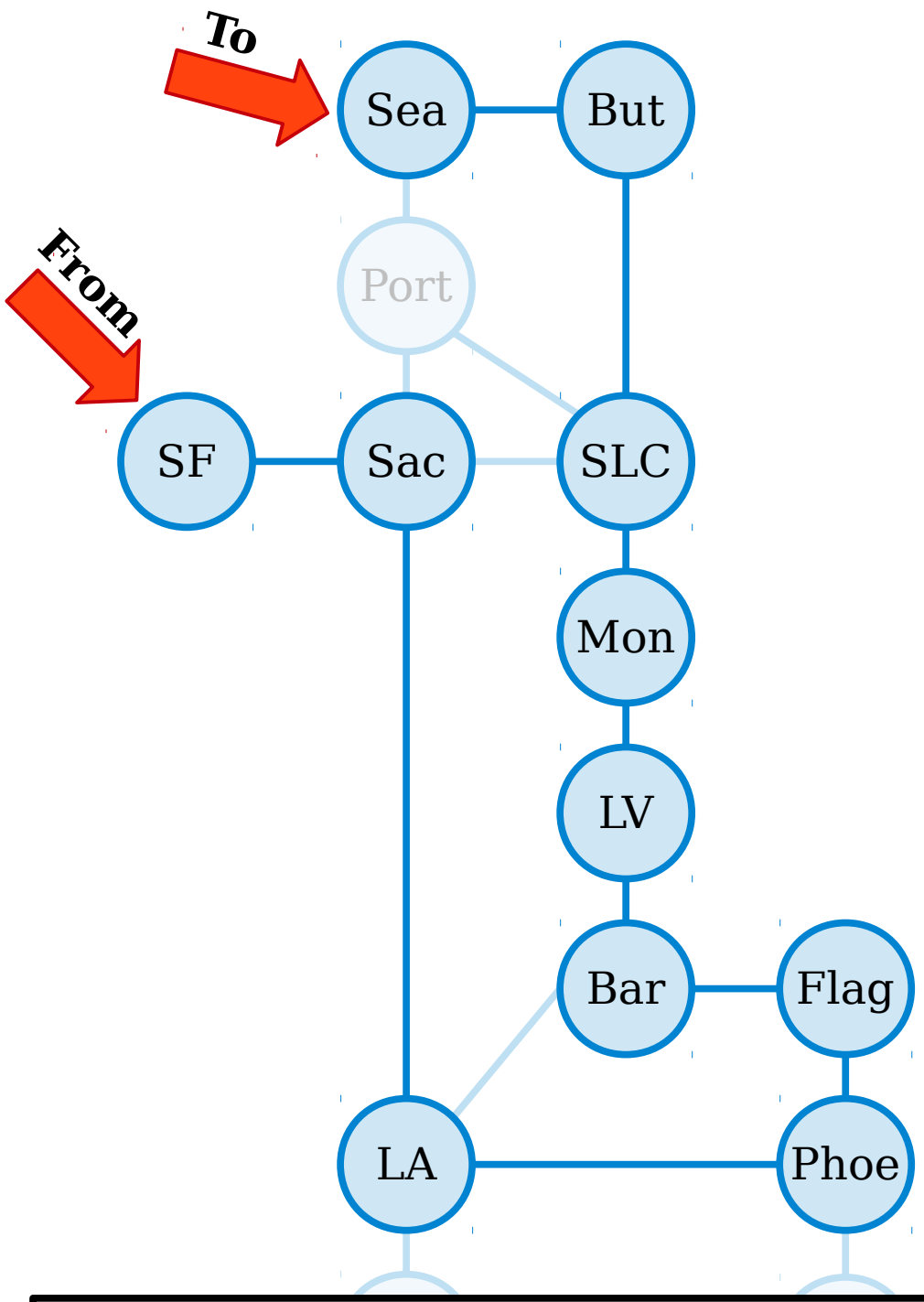

Sea
But
Port
SF
Sac
SLC
Mon
LV
Bar
Flag
LA
Phoe
SD
Nog


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


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

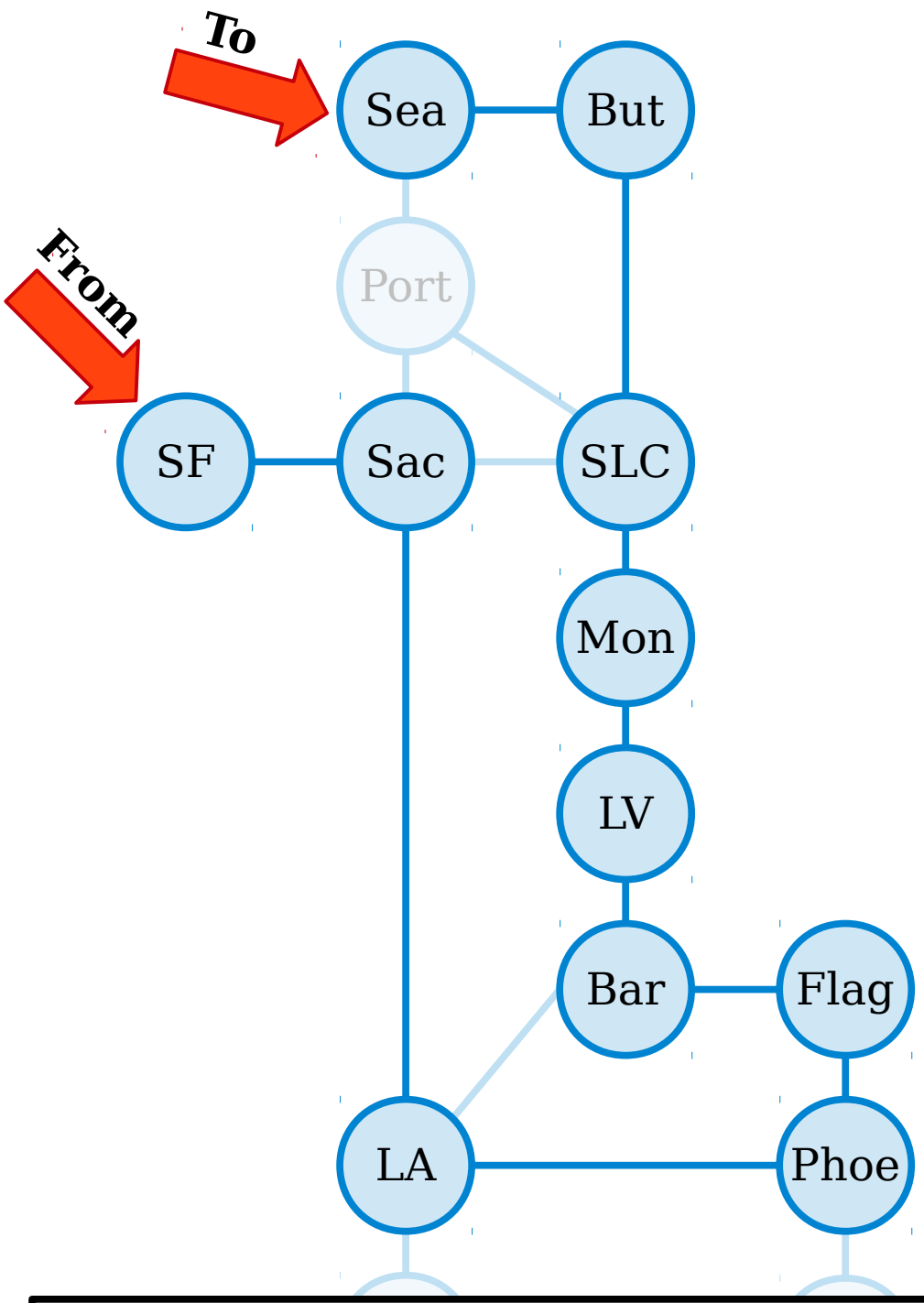

To
From
Sea
But
Port
SF
Sac
SLC
SF, Sac, SLC, Port, Sea
LV
Bar
Flag
LA
Phoe
SD
Nog


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



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.

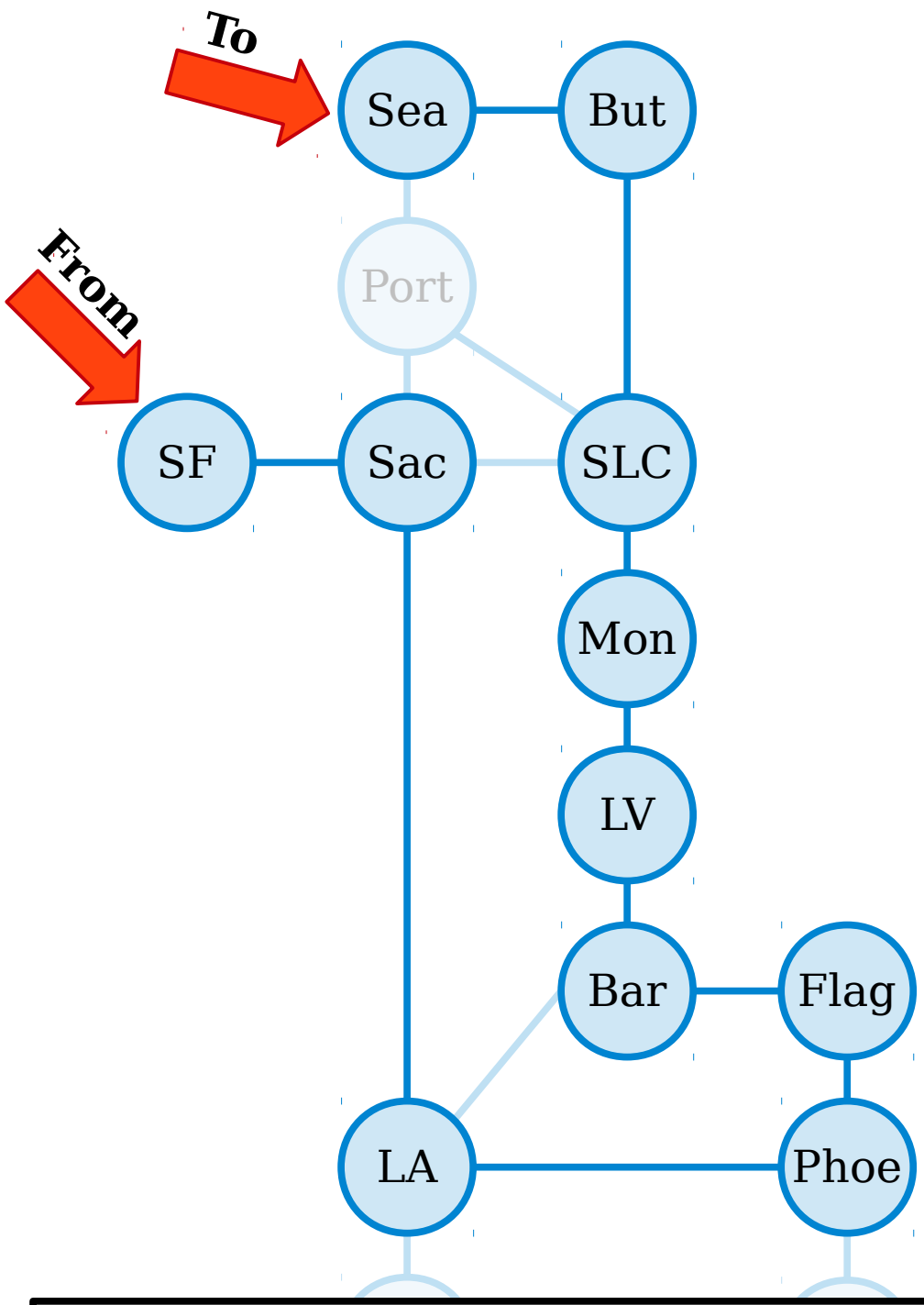
SF, Sac, LA, Phoe, Flag, Bar, LV, Mon, SLC, But, Sea



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 the path $v_1, \ldots, v_n$ is $n - 1$.

SF, Sac, LA, Phoe, Flag, Bar, LV, Mon, SLC, But, Sea

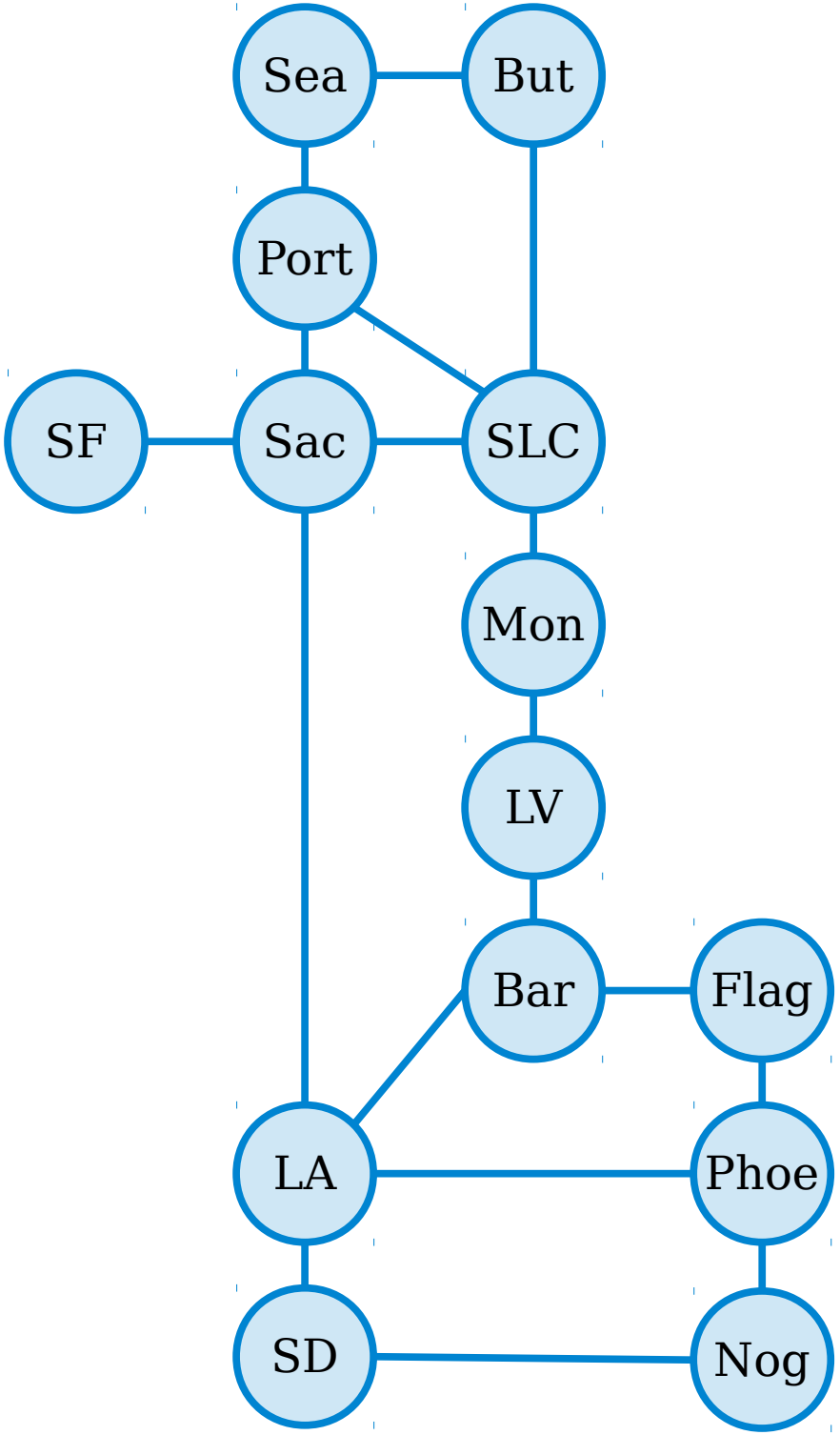


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

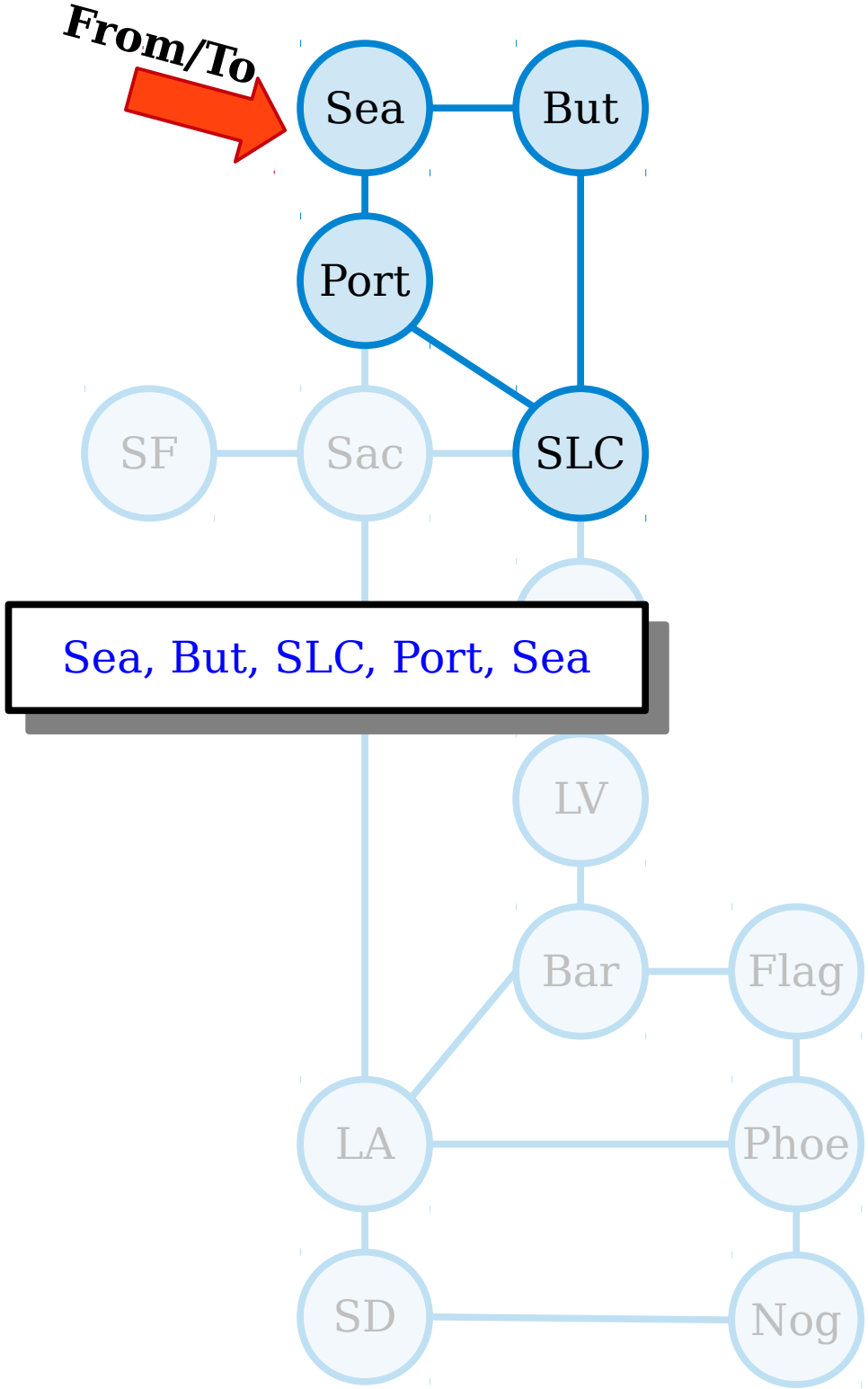
(This path has length 10, but visits 11 cities.)

SF, Sac, LA, Phoe, Flag, Bar, LV, Mon, SLC, But, Sea



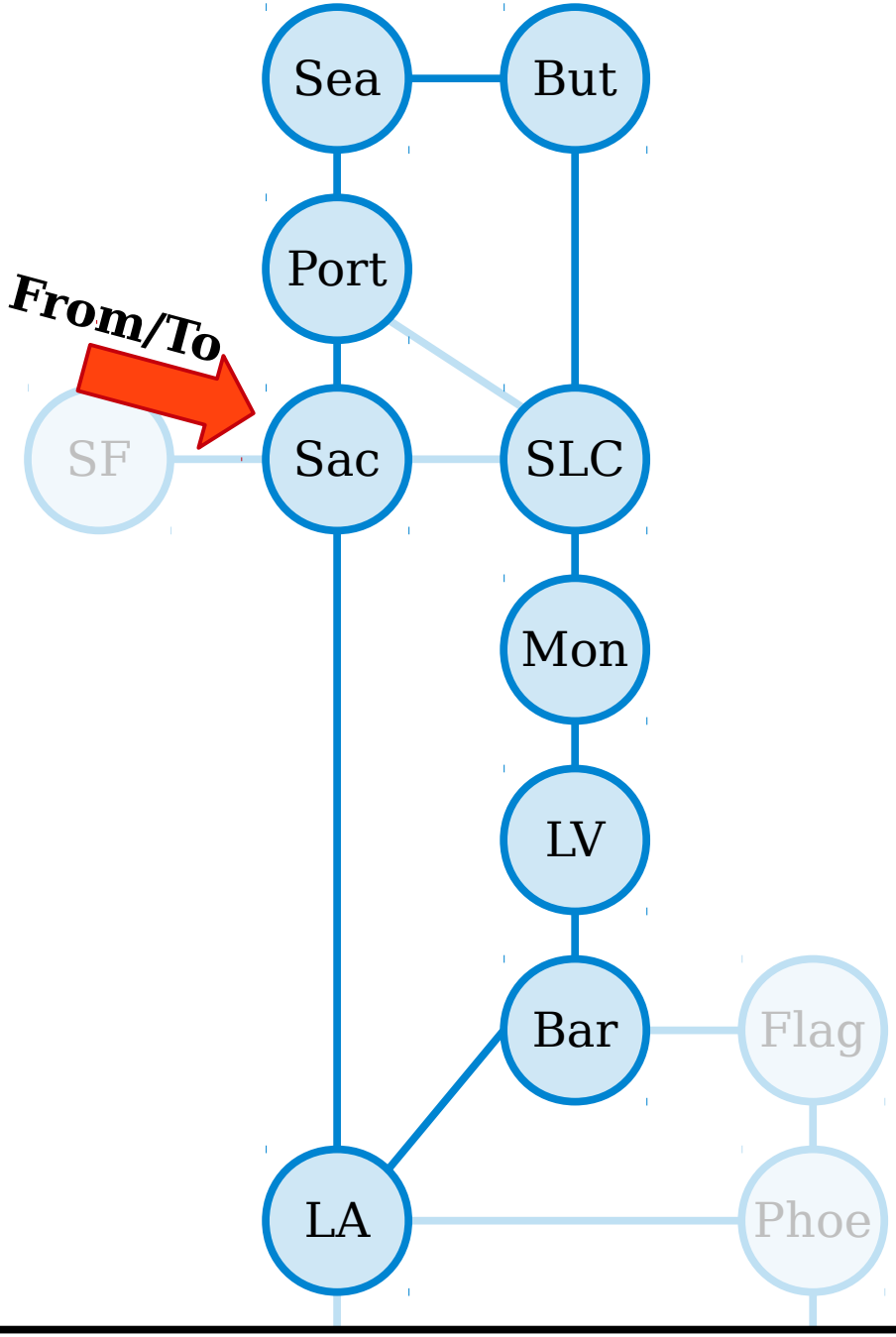
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.



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

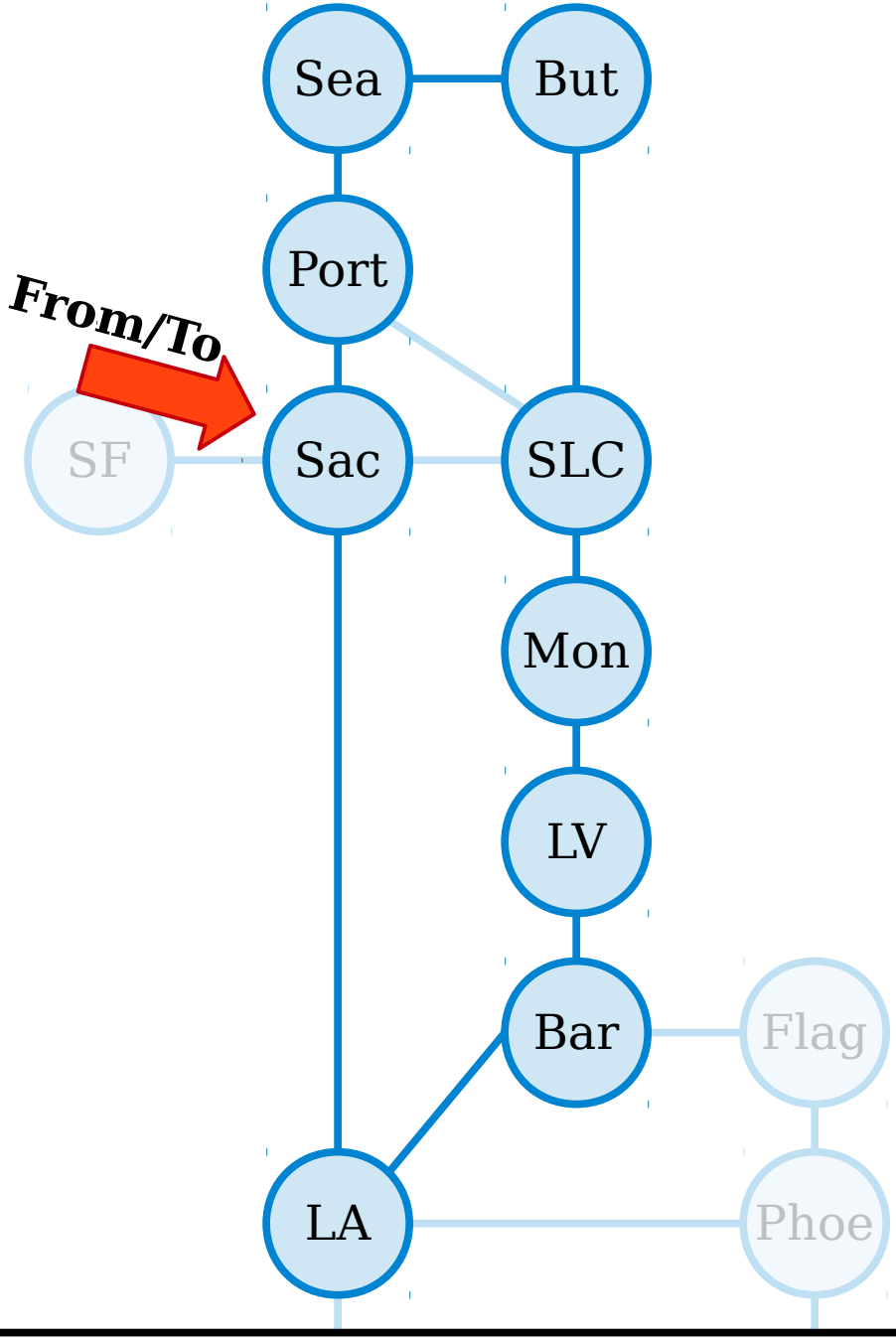
The ***length*** of the path v_1, …, v_n is n – 1.



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

The ***length*** of the path v_1, …, v_n is n – 1.

Sac, Port, Sea, But, SLC, Mon, LV, Bar, LA, Sac

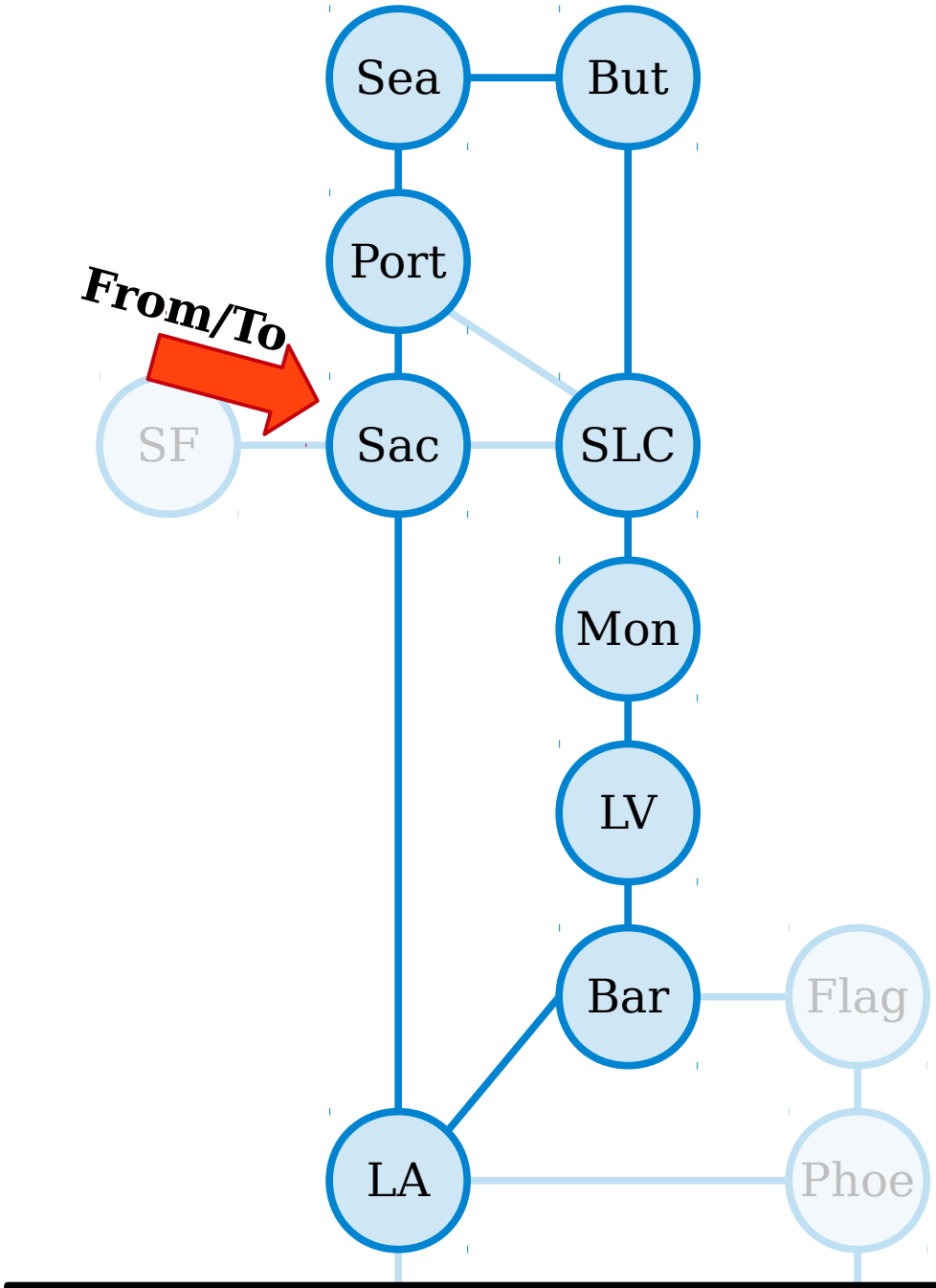


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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

Sac, Port, Sea, But, SLC, Mon, LV, Bar, LA, Sac



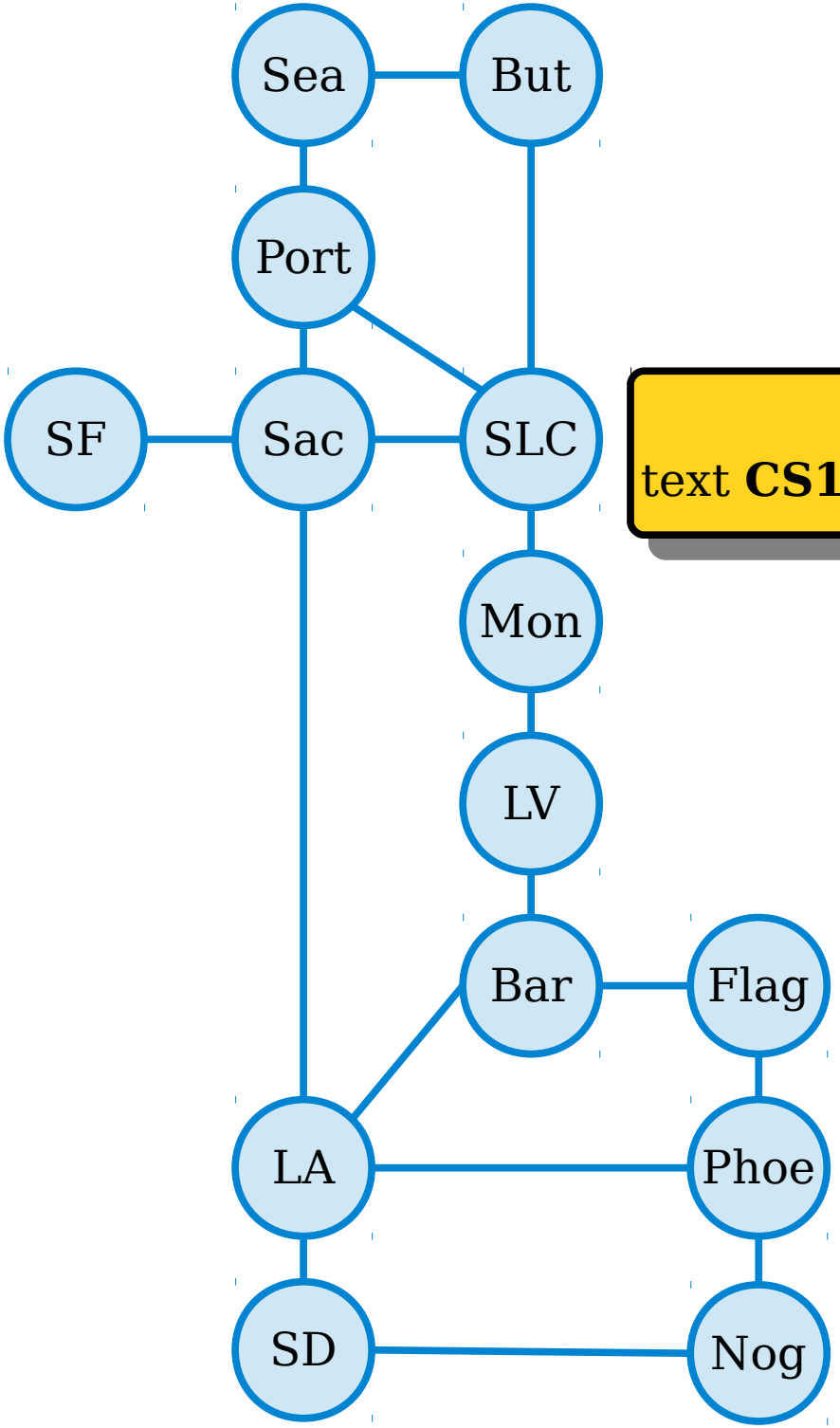
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, v_n such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path v_1, …, v_n is $n - 1$.

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

(This cycle has length nine and visits nine different cities.)

Sac, Port, Sea, But, SLC, Mon, LV, Bar, LA, Sac



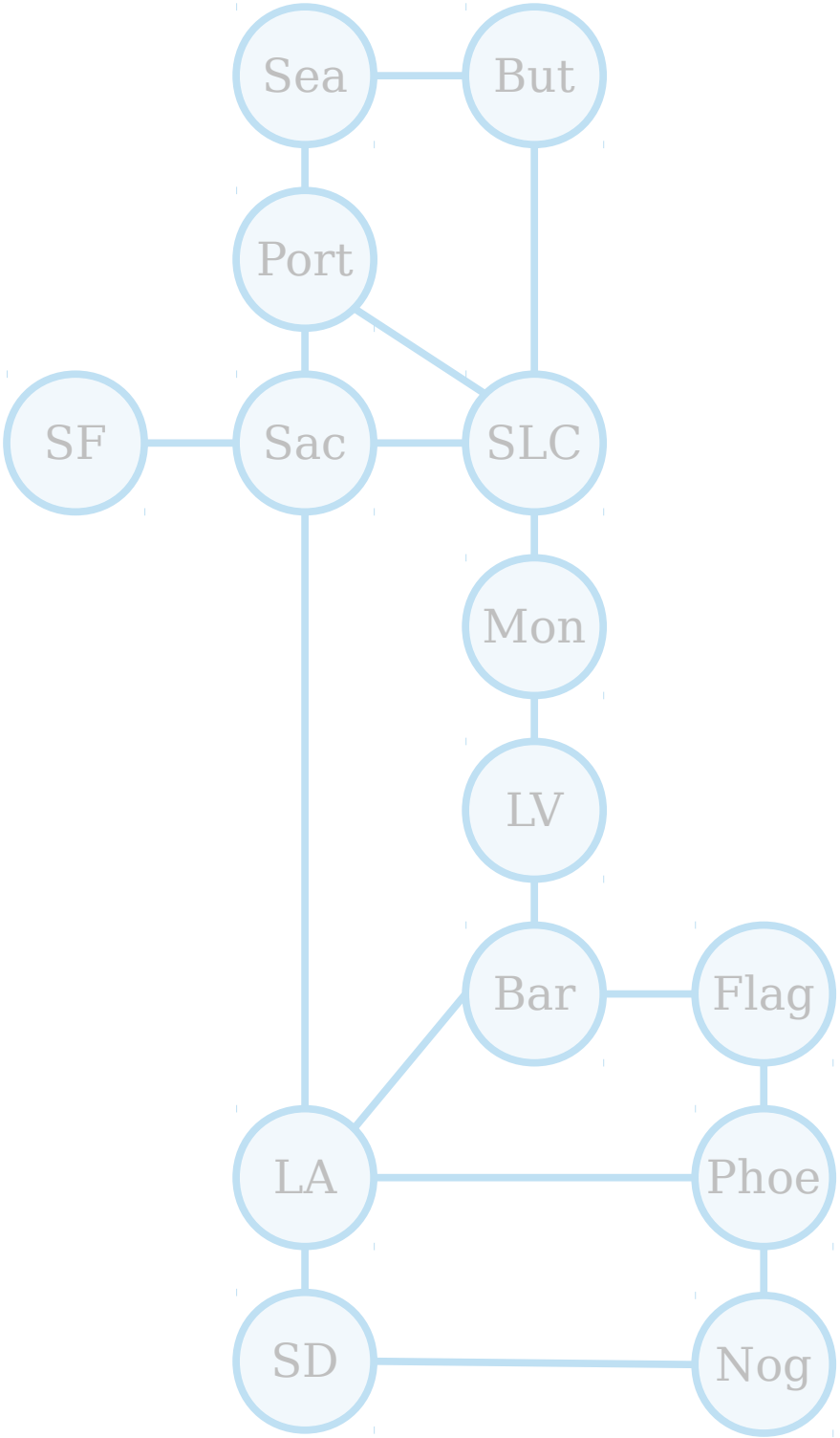
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, v_n such that any two consecutive nodes in the sequence are adjacent.

Answer at **PollEv.com/cs103** or
text **CS103** to **22333** once to join, then **A**, …, or **E**.

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

How many paths in this graph are there from SF to LA?

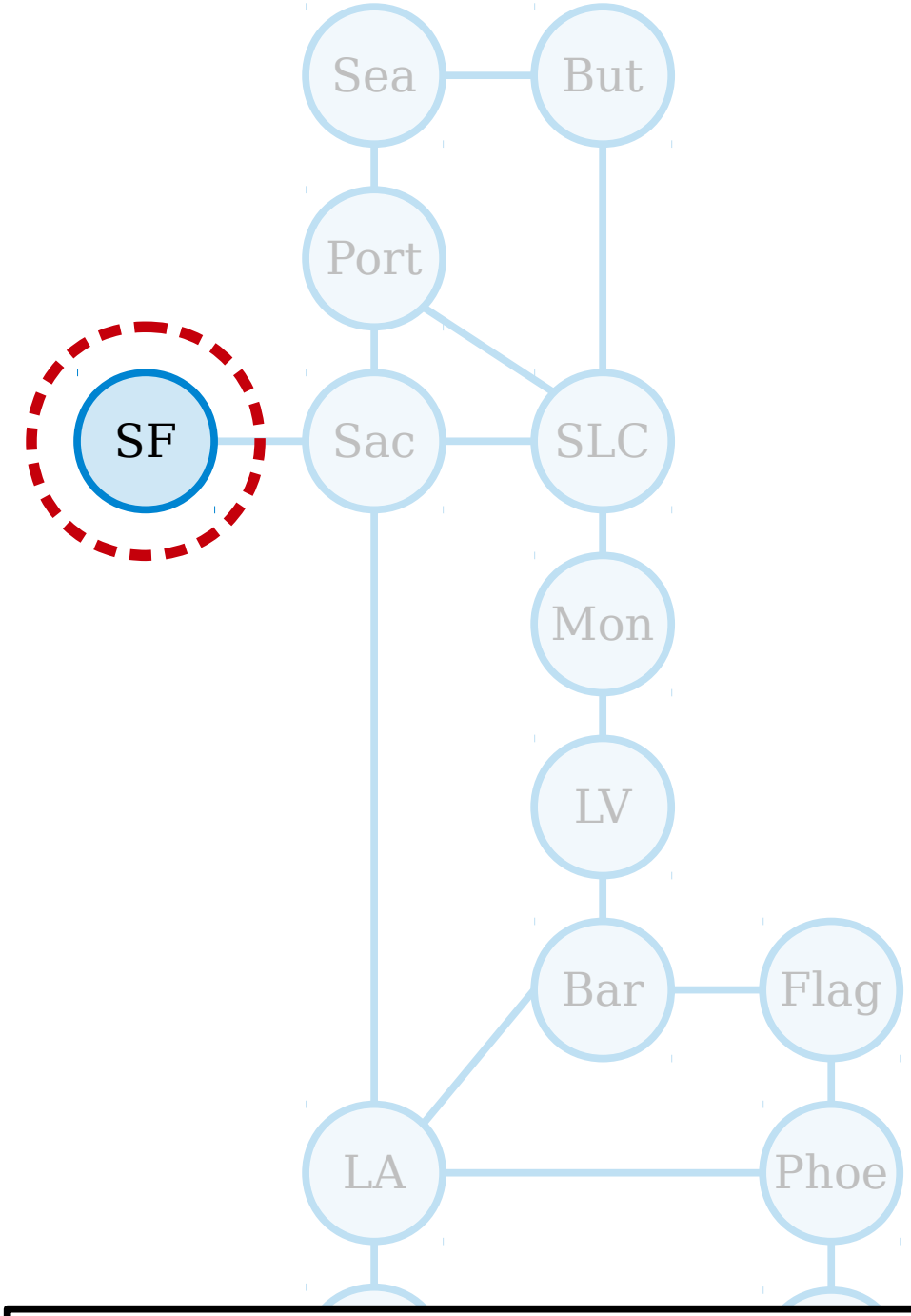
A. 1
B. 4
C. 10
D. 20
E. None of these.



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 the path $v_1, \ldots, v_n$ is $n - 1$.

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

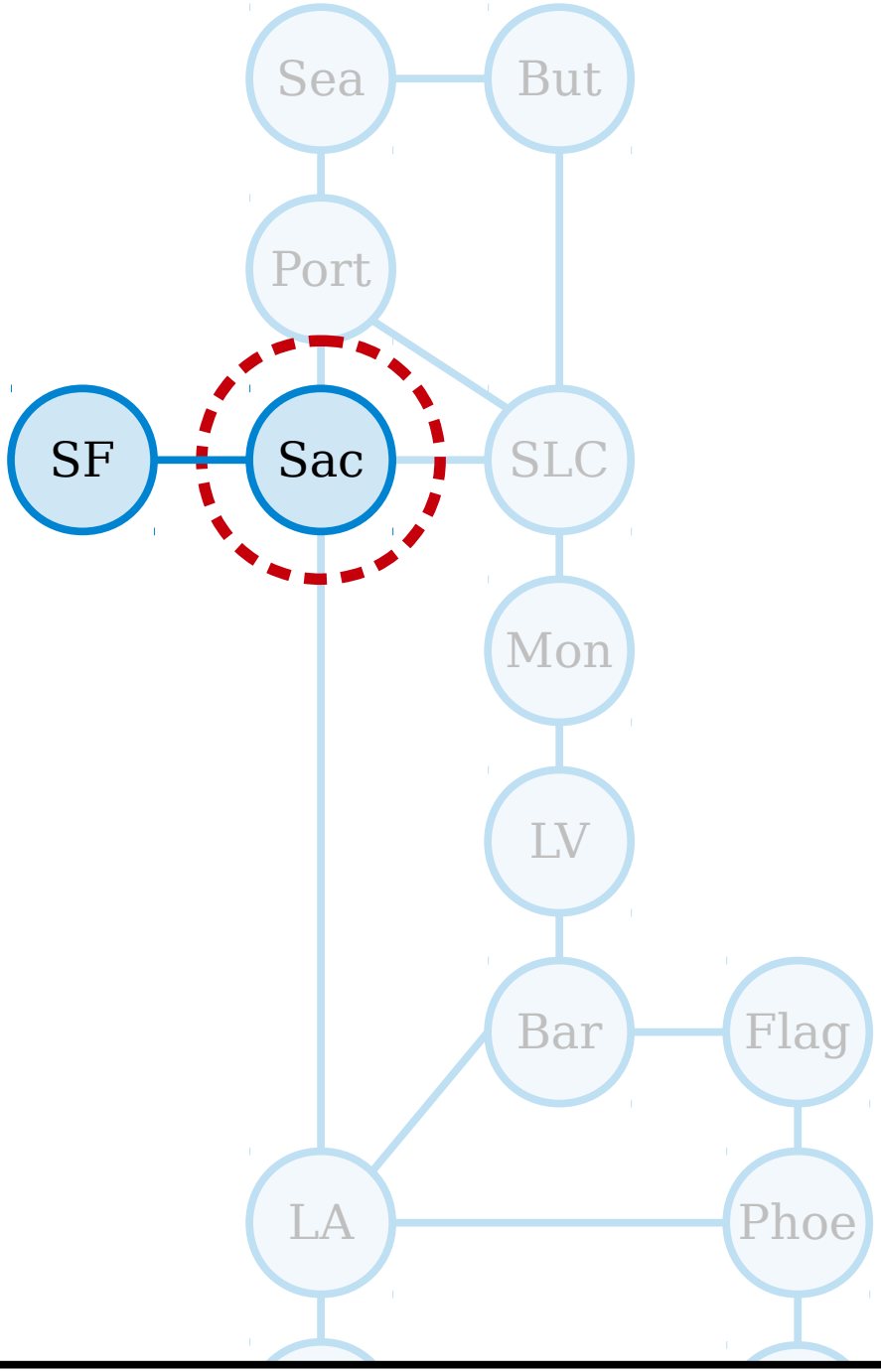


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, v_n such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path v_1, …, v_n is n – 1.

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

SF

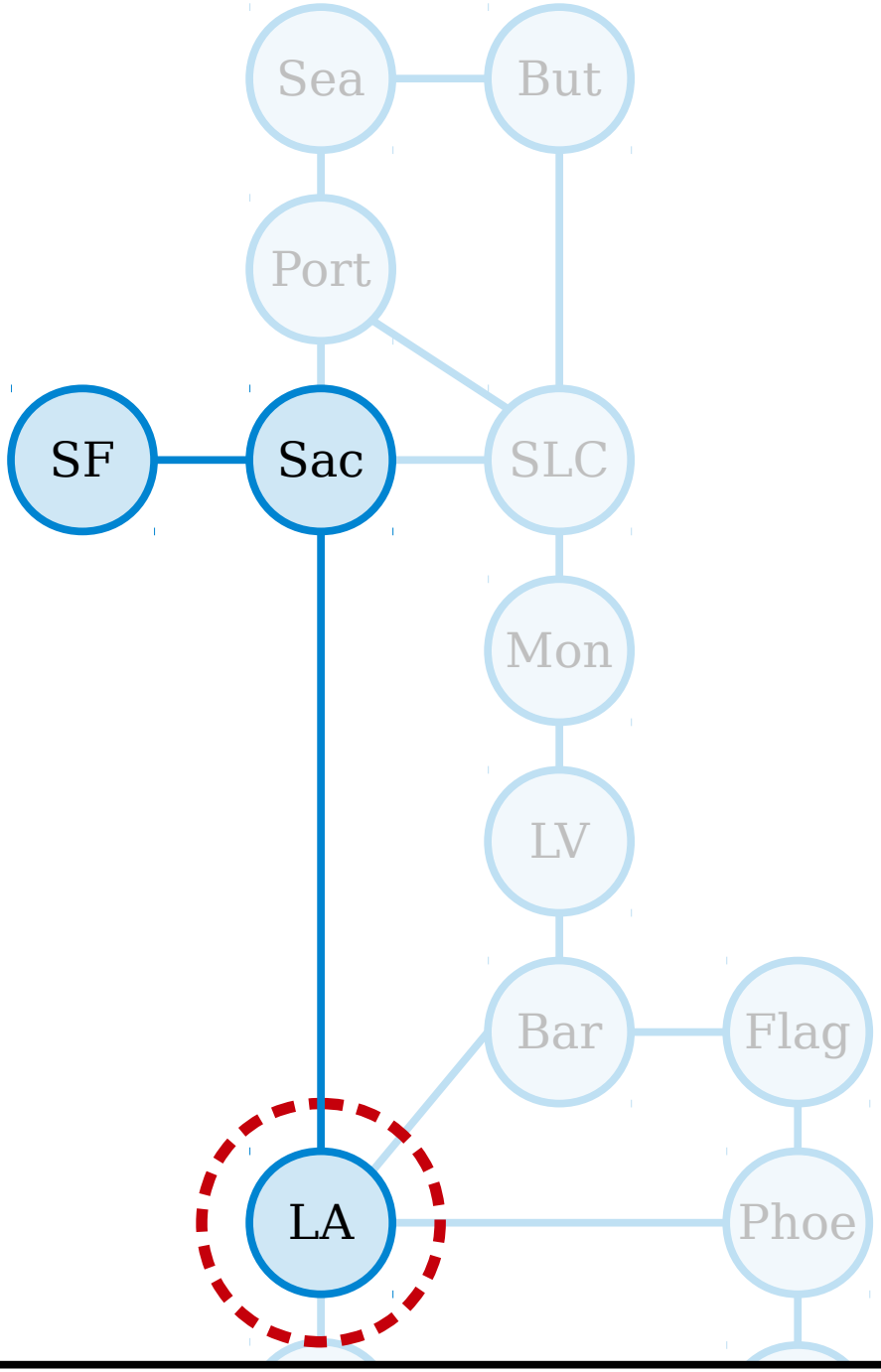


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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

SF, Sac

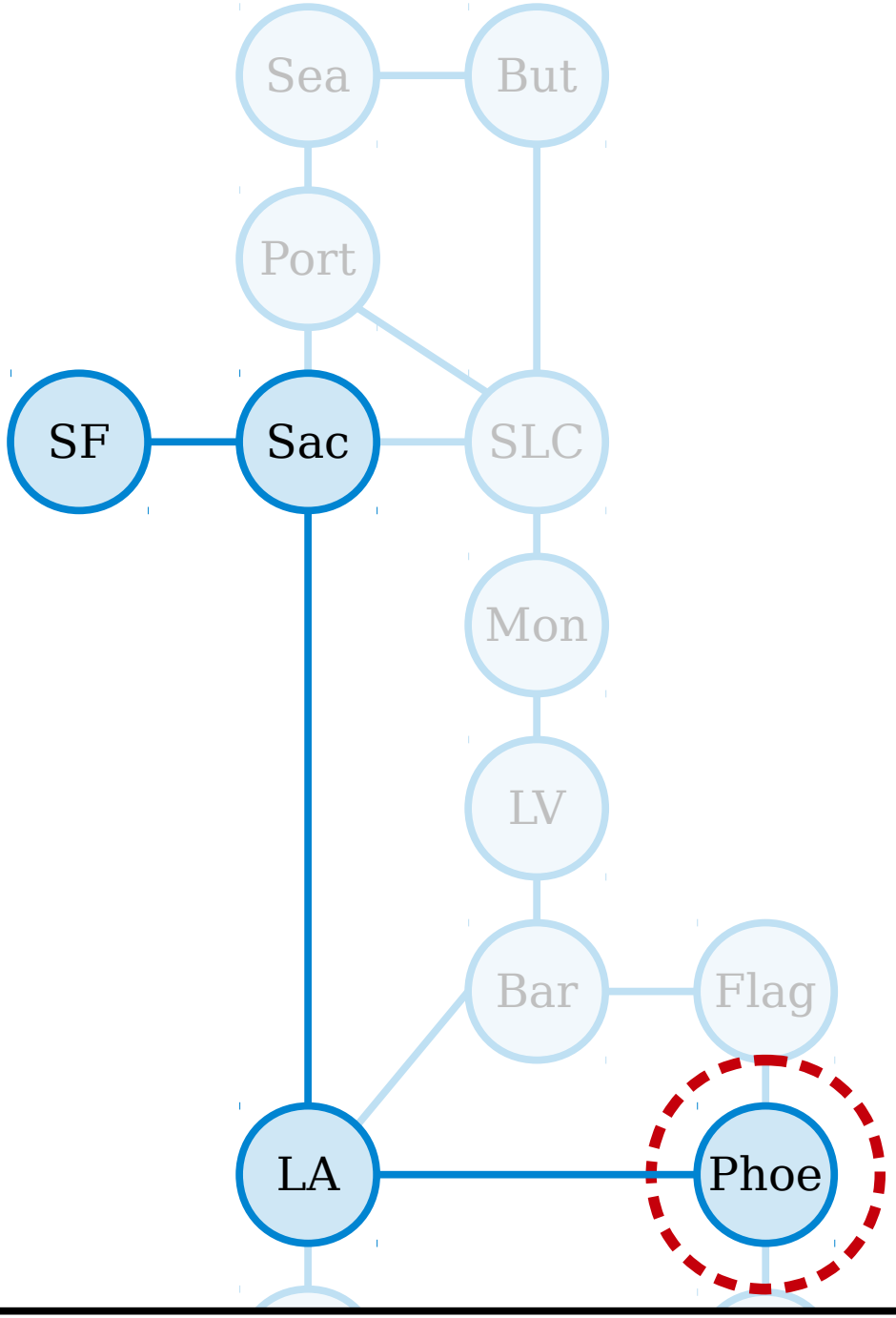


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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

SF, Sac, LA

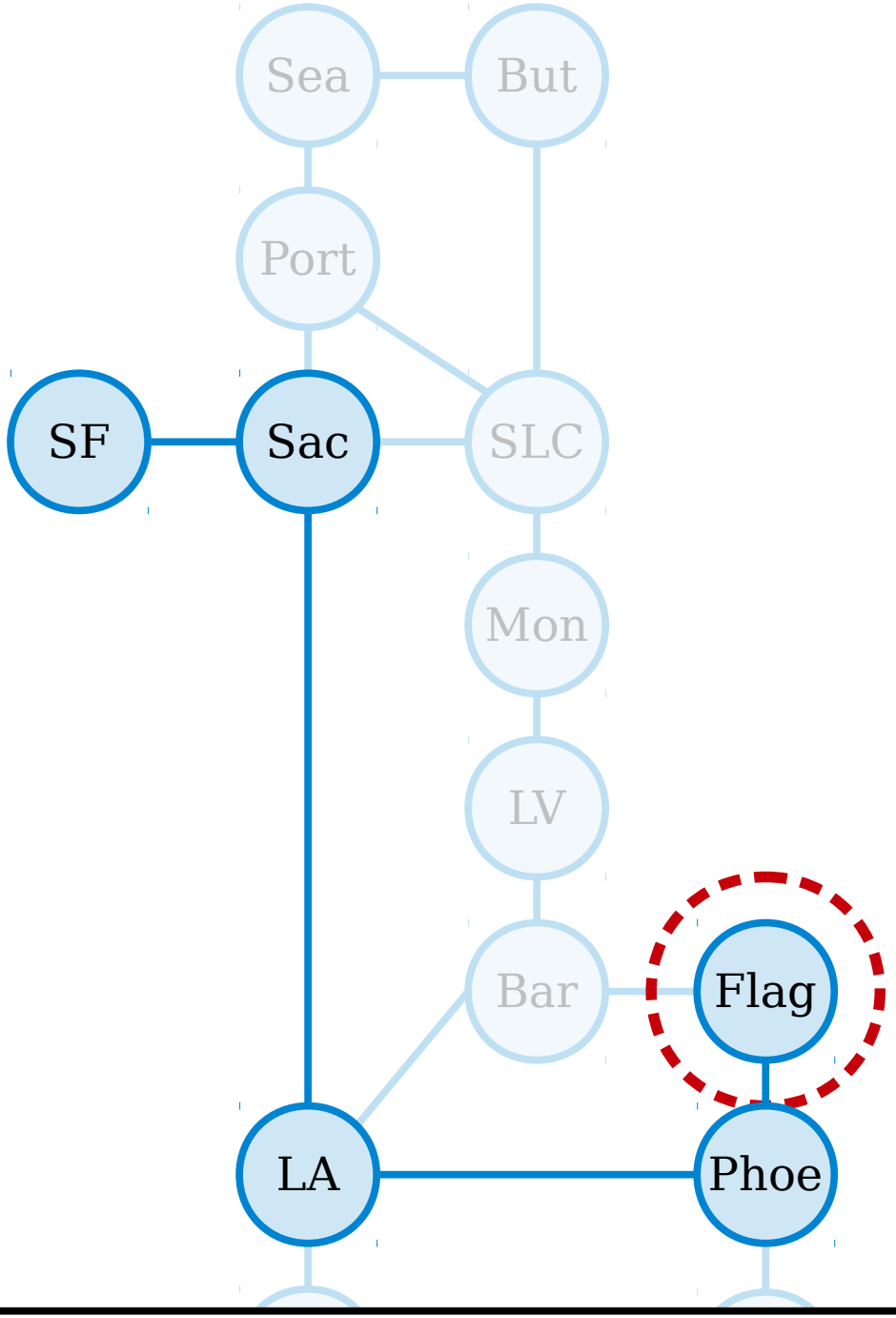


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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

SF, Sac, LA, Phoe

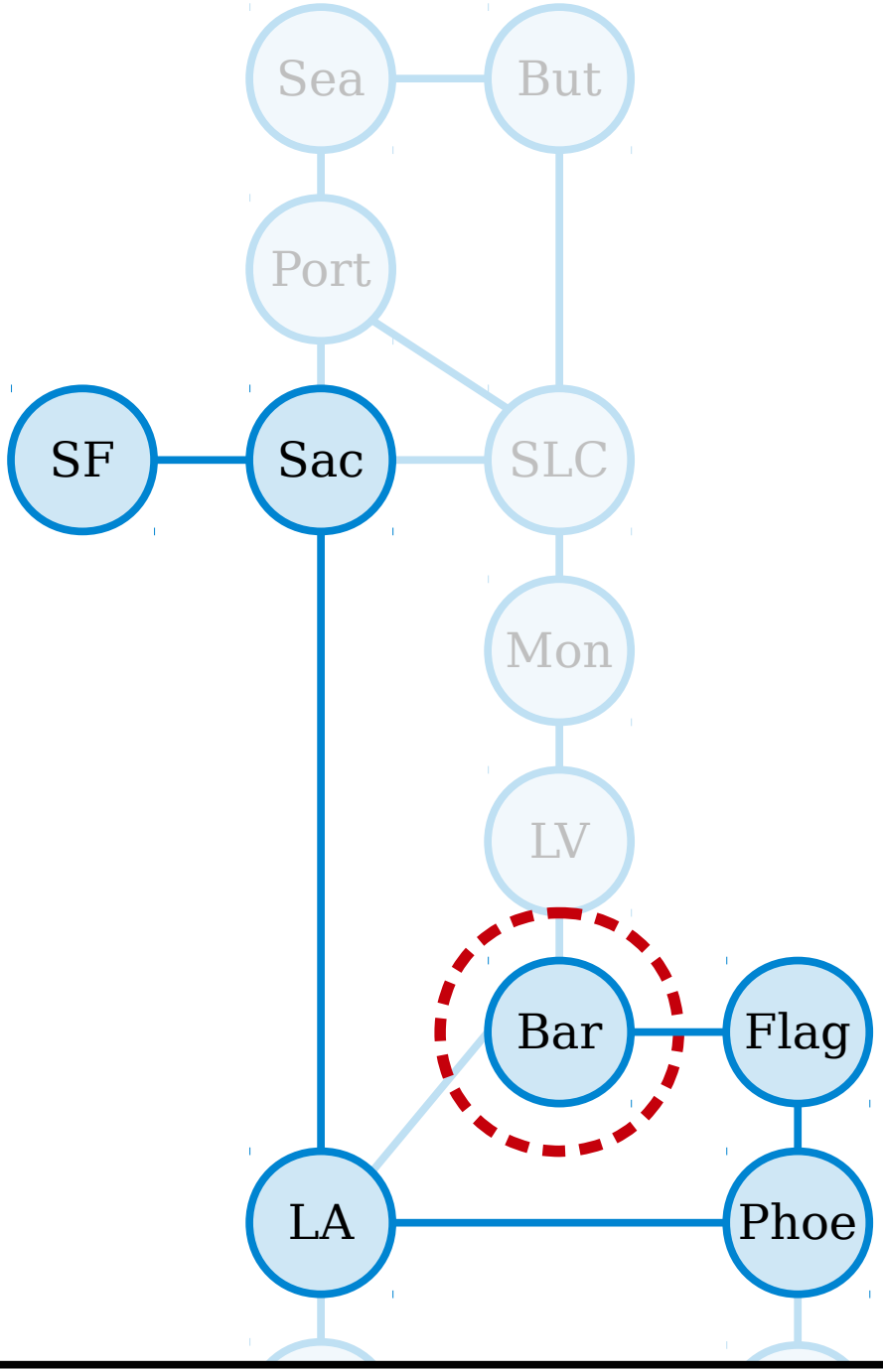


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, v_n such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path v_1, …, v_n is $n - 1$.

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

SF, Sac, LA, Phoe, Flag

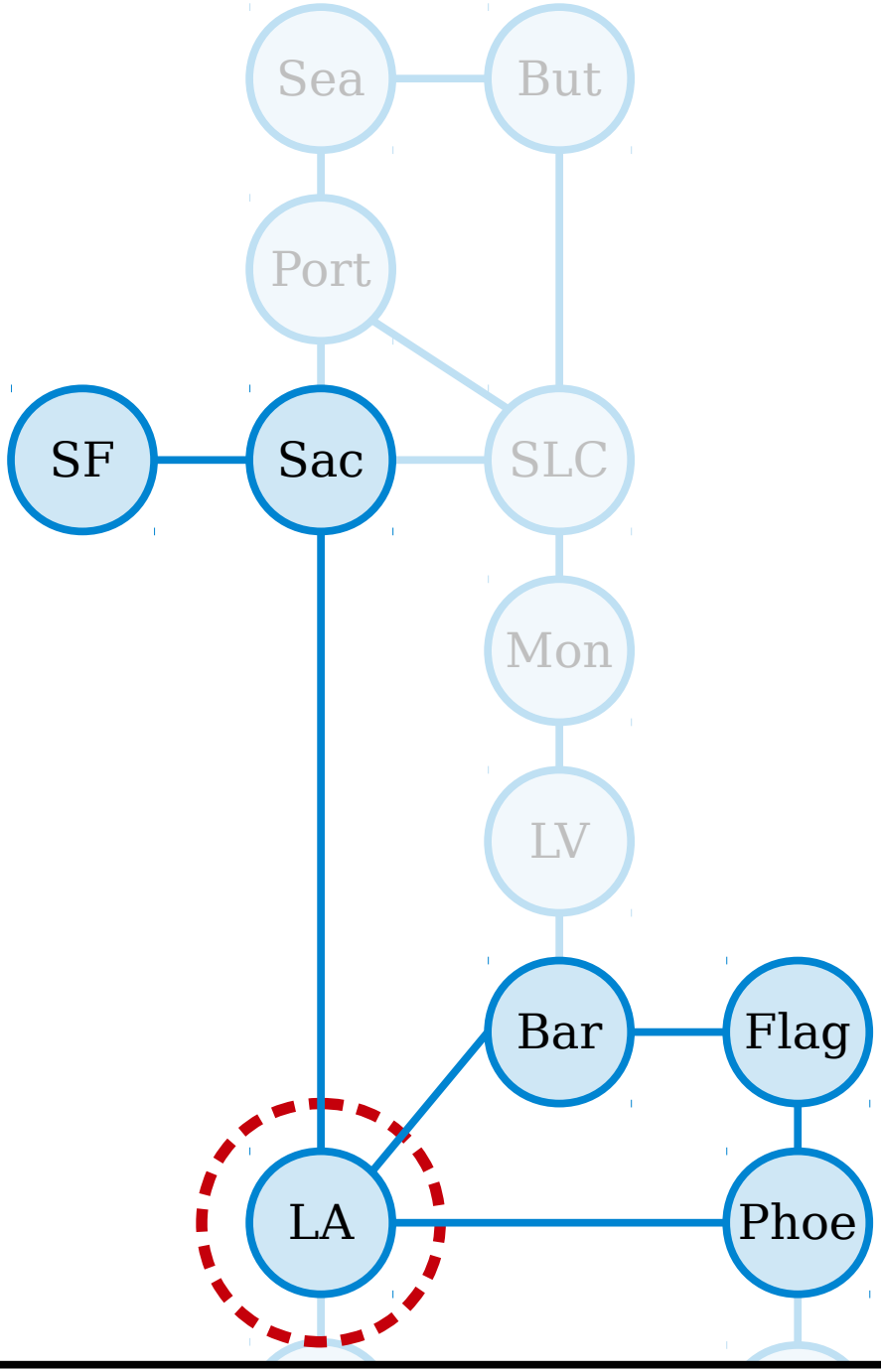


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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

SF, Sac, LA, Phoe, Flag, Bar

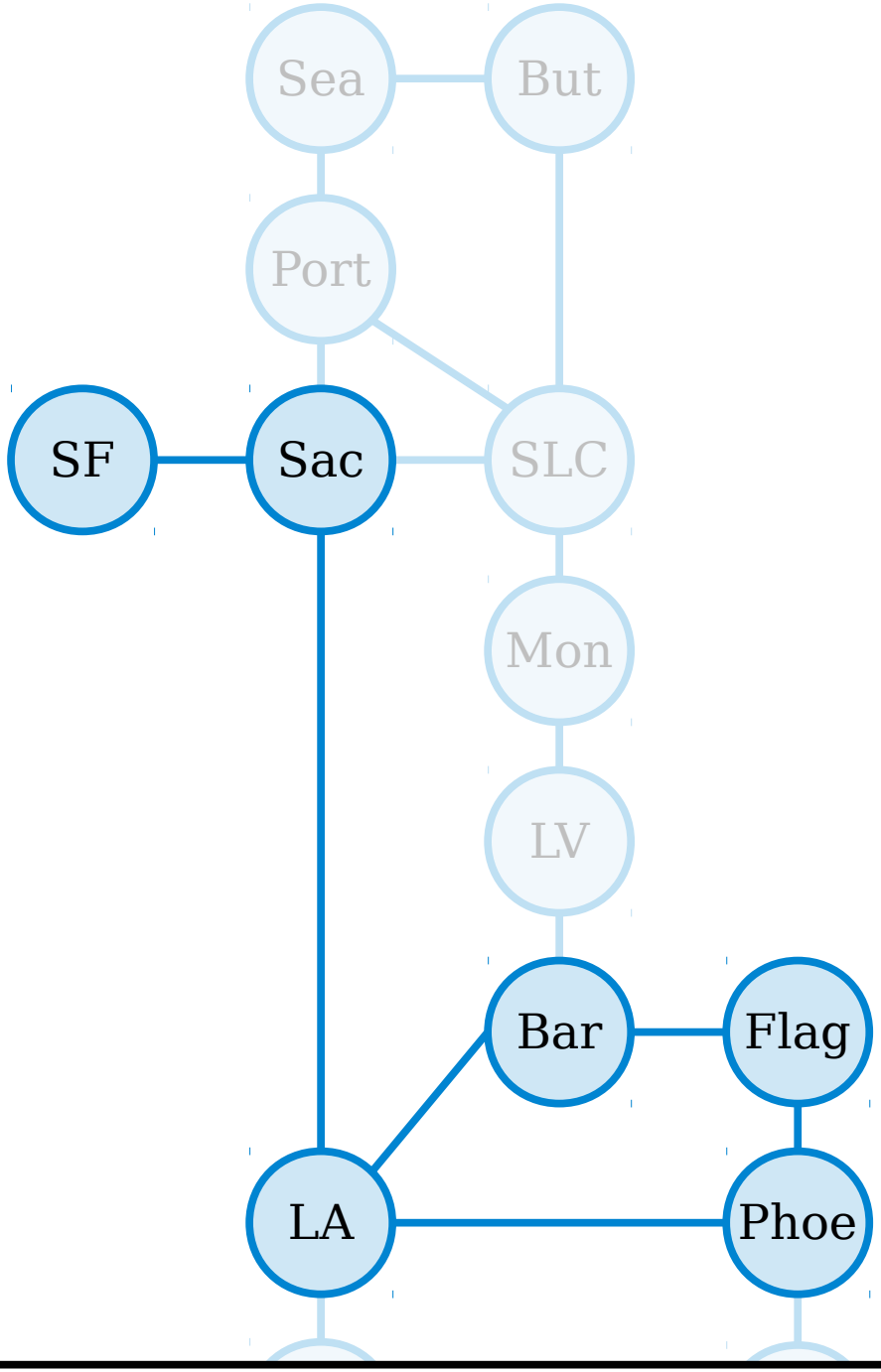


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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

SF, Sac, LA, Phoe, Flag, Bar, LA

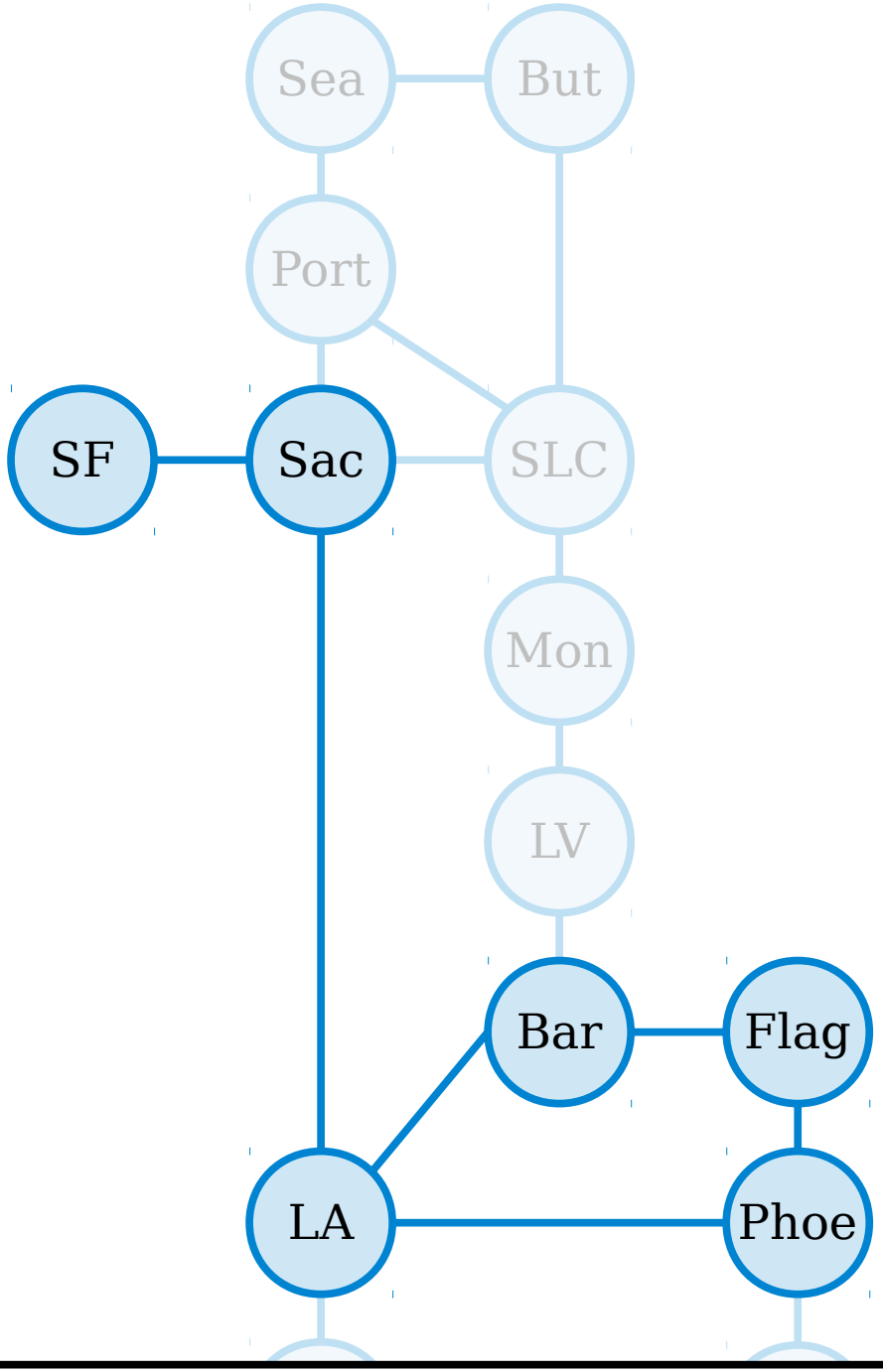


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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

SF, Sac, LA, Phoe, Flag, Bar, LA



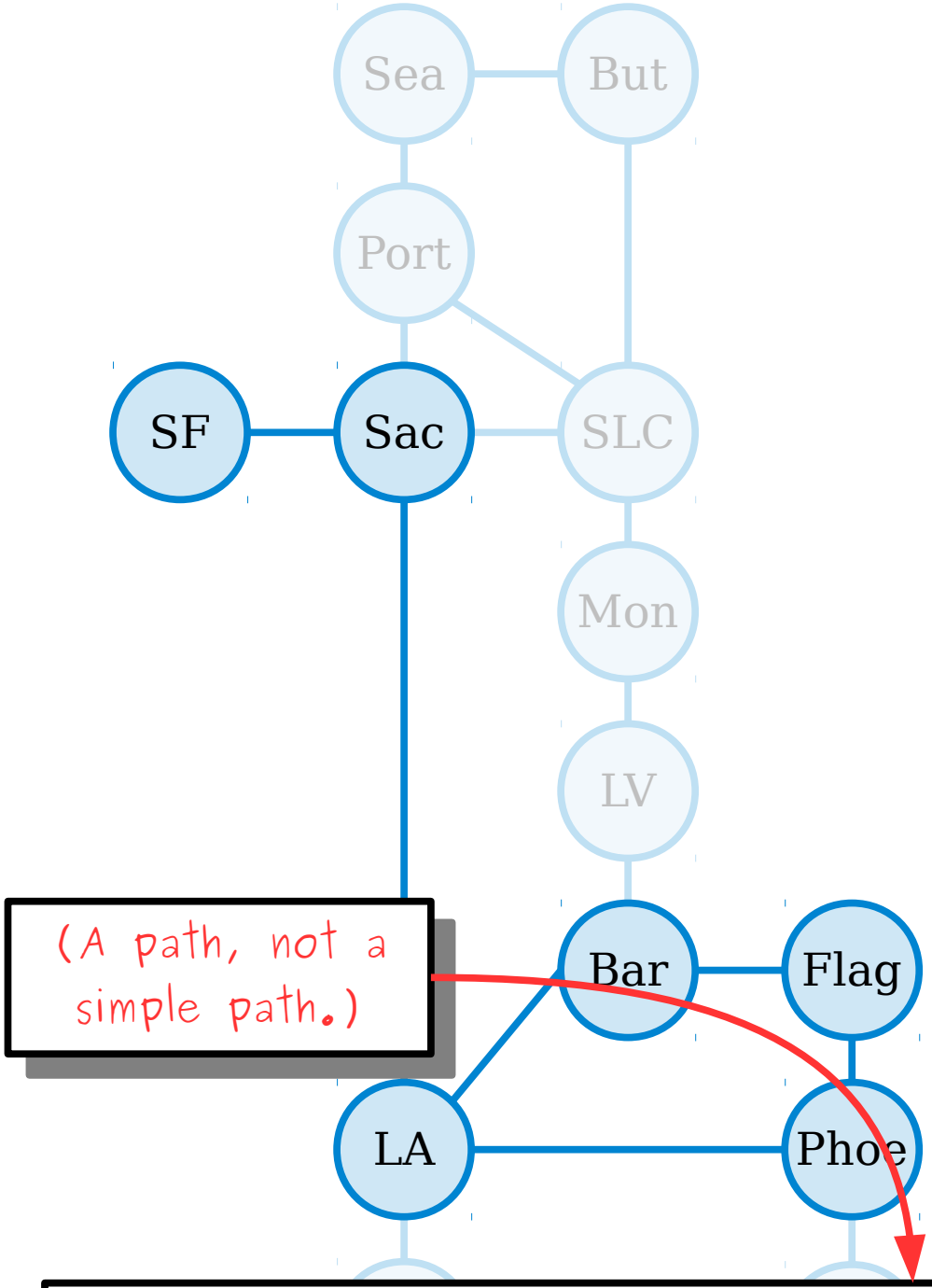
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, v_n such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path v_1, …, v_n is $n - 1$.

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.

SF, Sac, LA, Phoe, Flag, Bar, LA



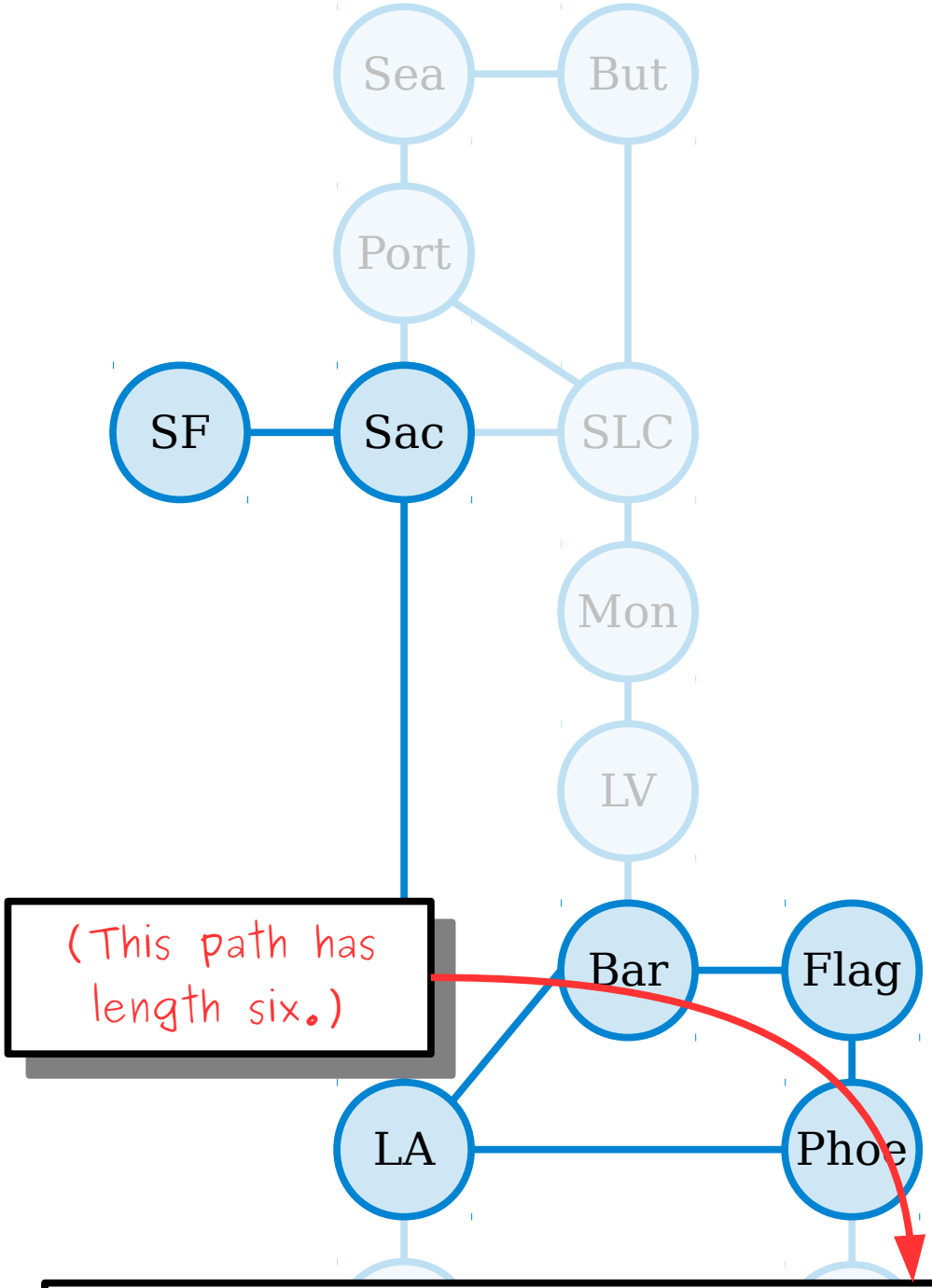
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, v_n such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path v_1, …, v_n is $n - 1$.

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.

SF, Sac, LA, Phoe, Flag, Bar, LA



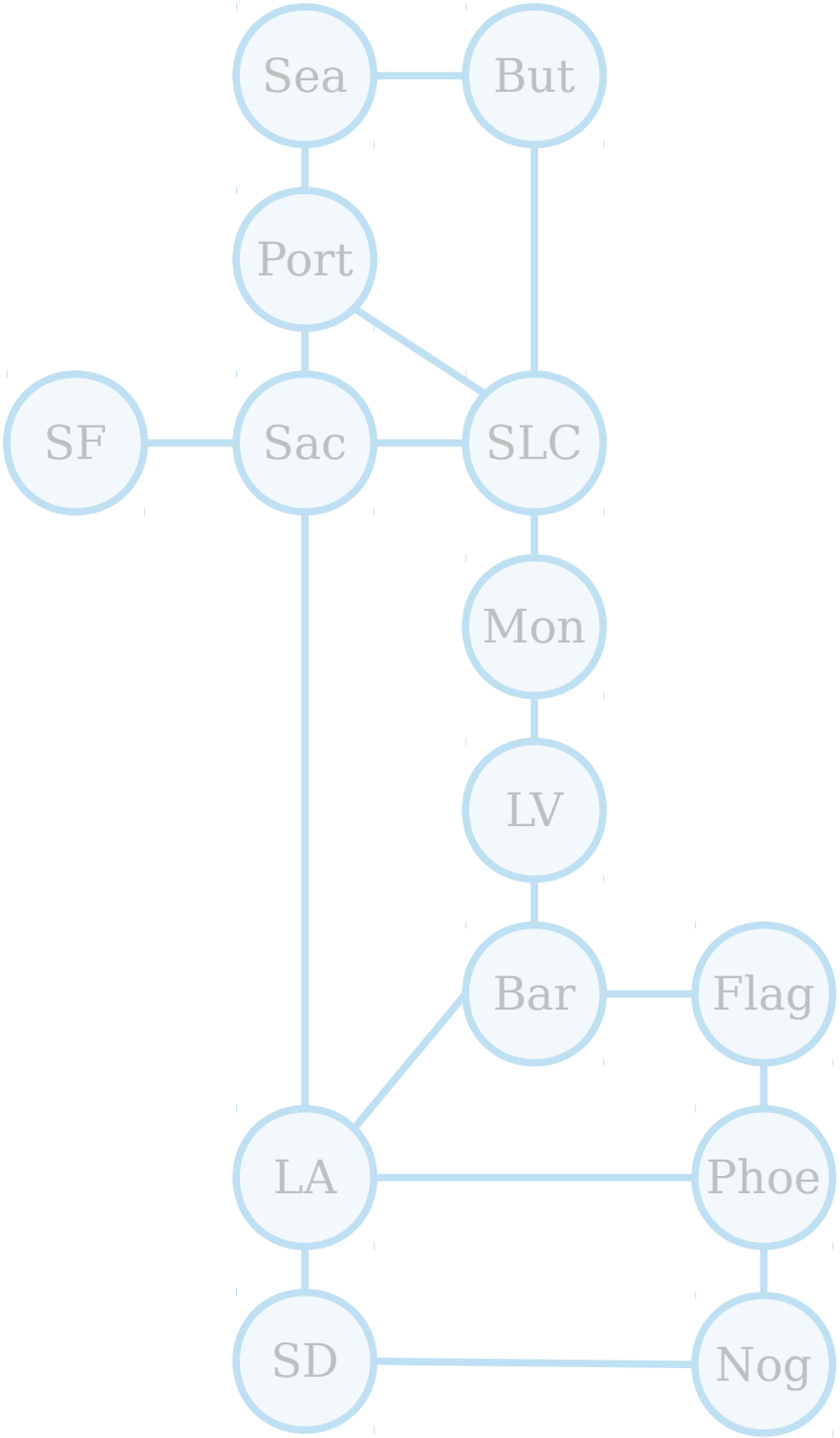
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, v_n such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path v_1, …, v_n is $n - 1$.

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.

SF, Sac, LA, Phoe, Flag, Bar, LA

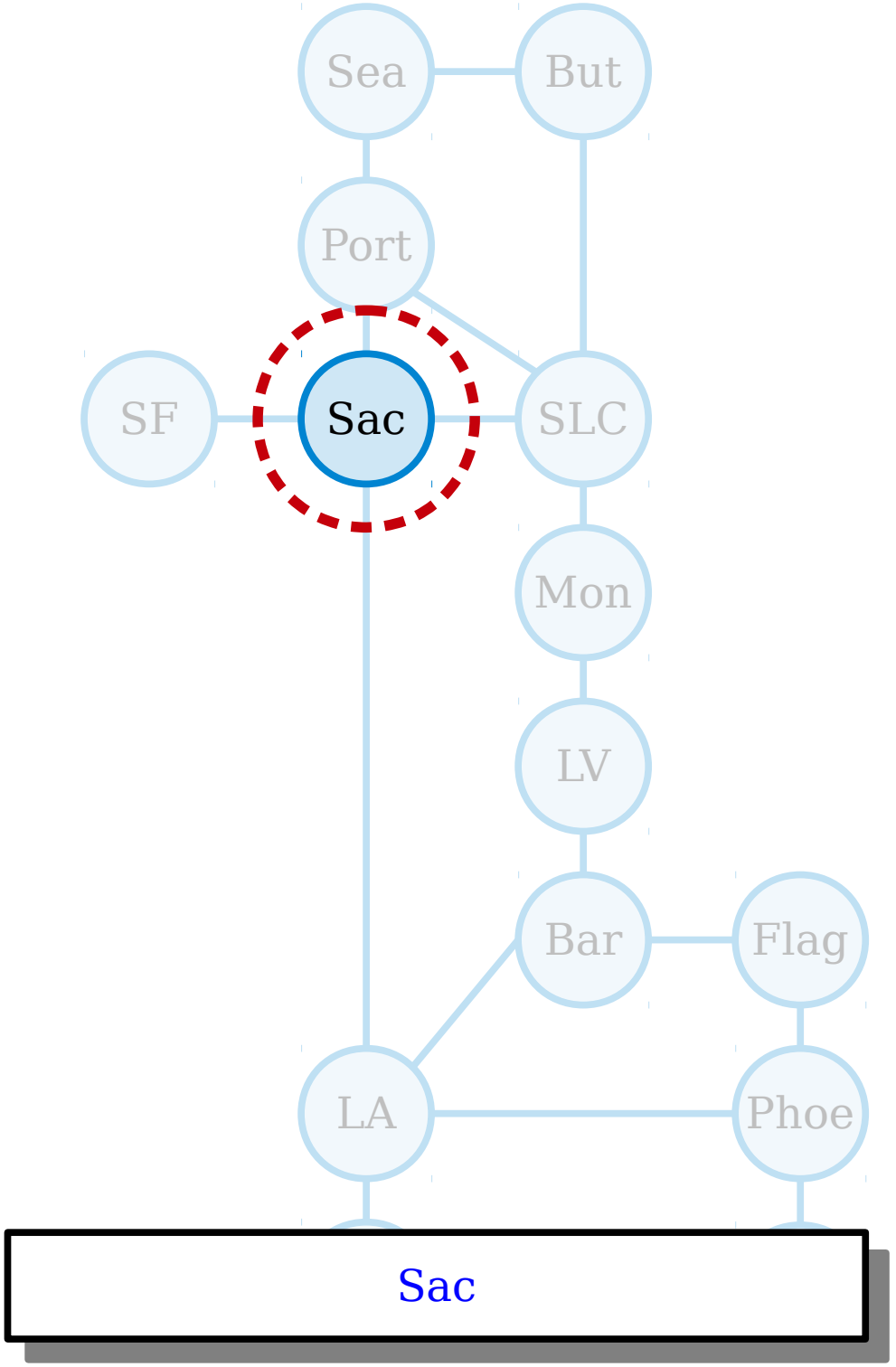


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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.

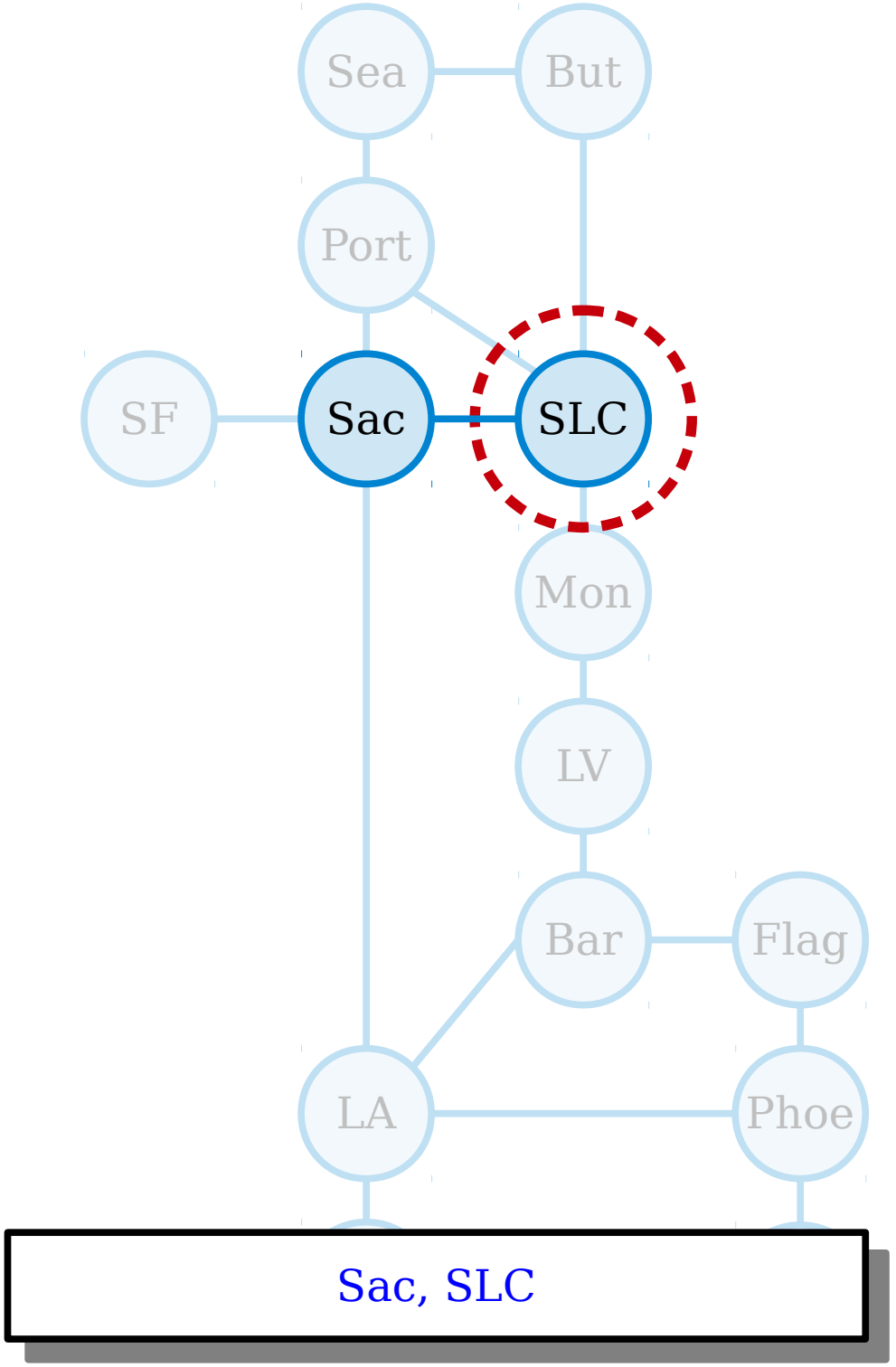


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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.



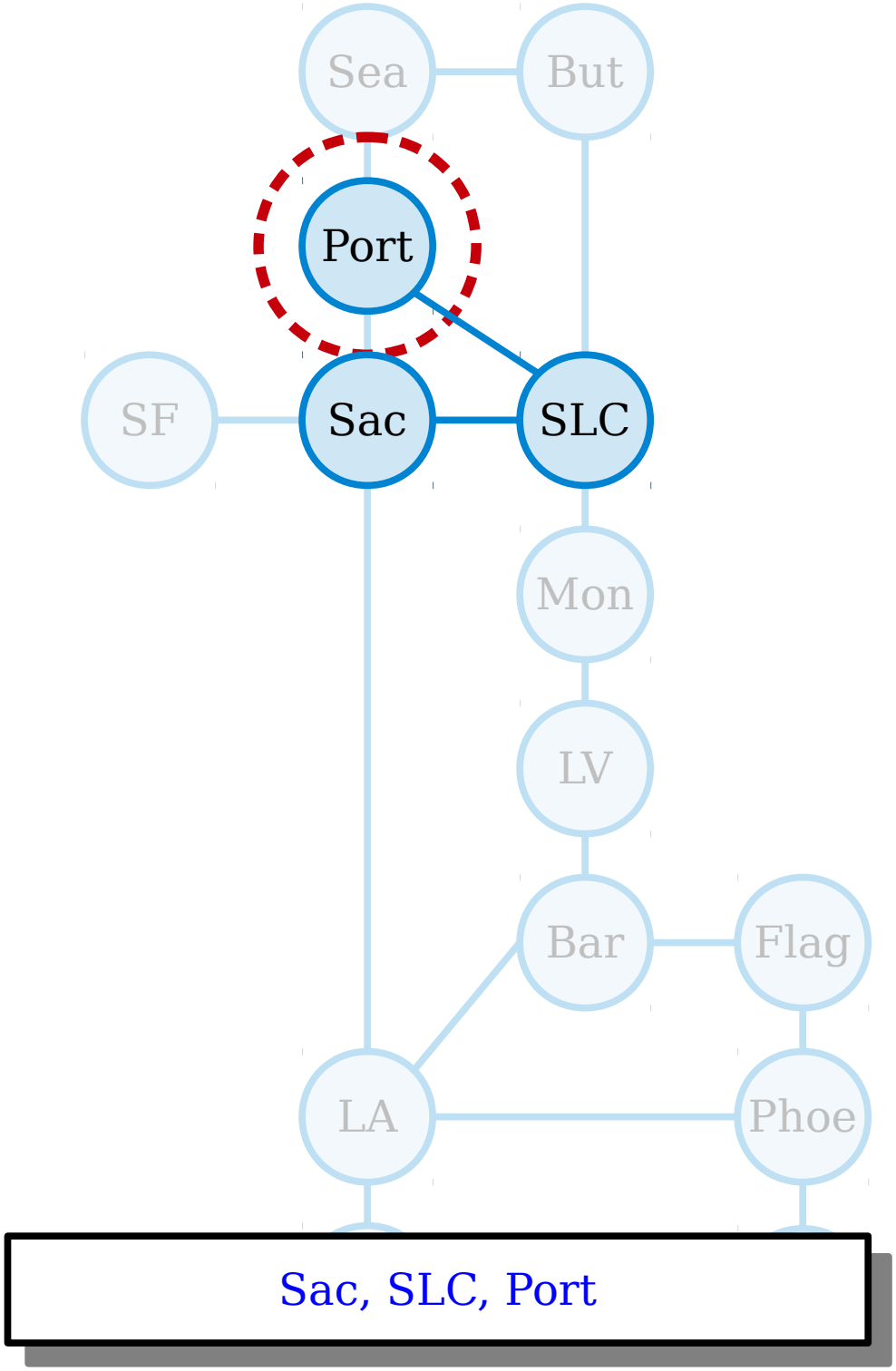
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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.

Sac, SLC



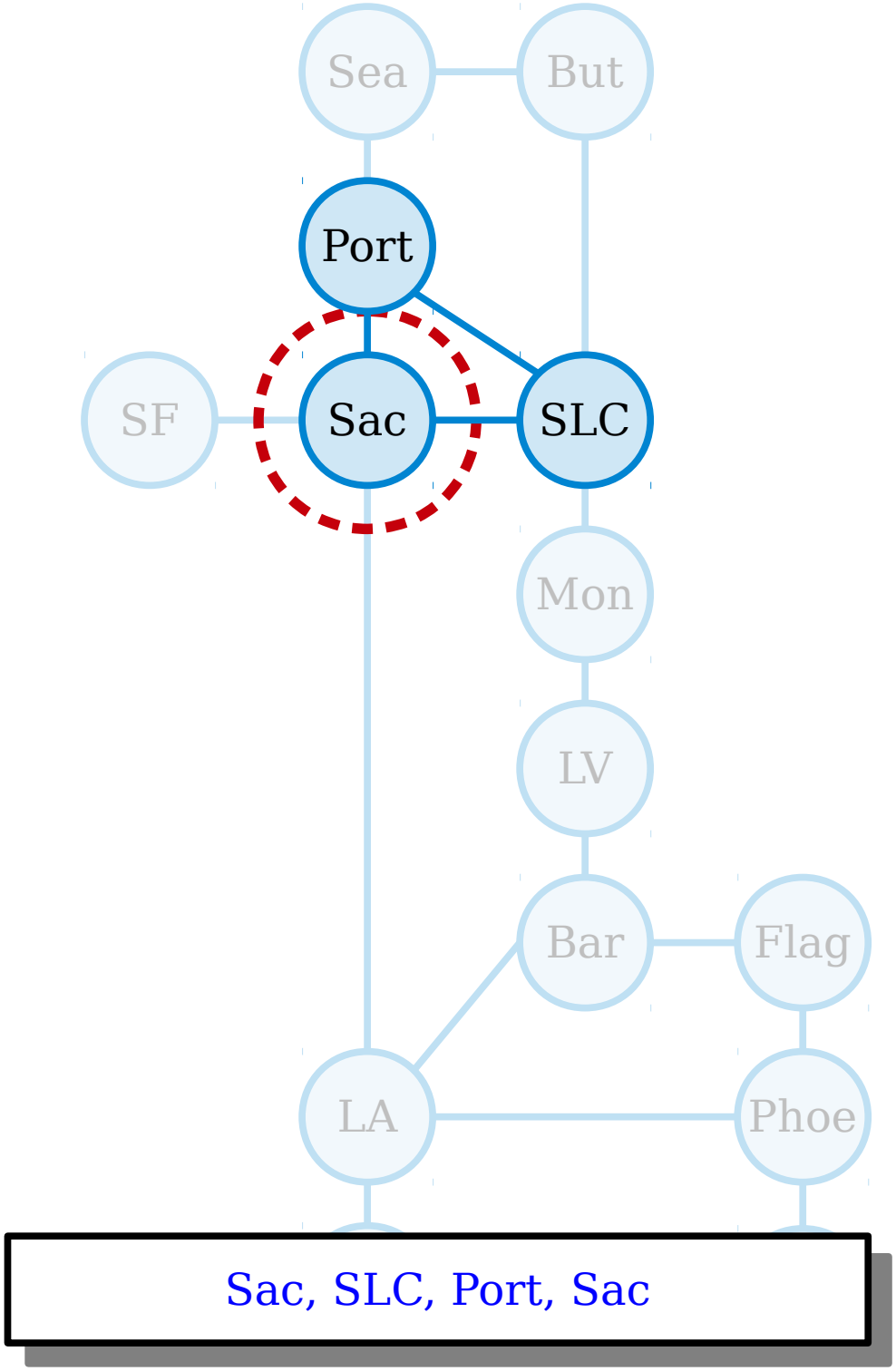
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, v_n such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path v_1, …, v_n is n – 1.

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.

Sac, SLC, Port



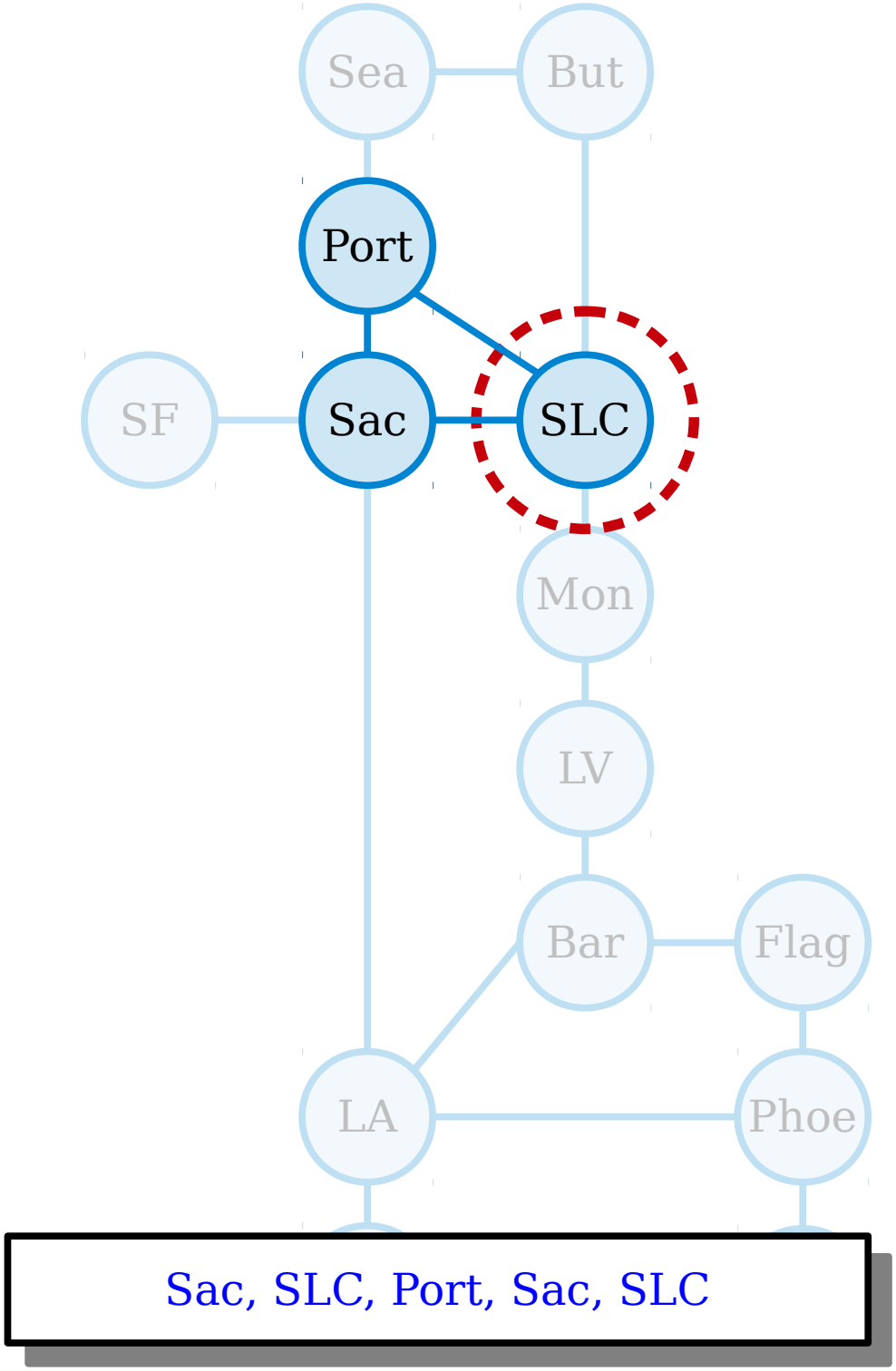
Sac, SLC, Port, Sac

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

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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.



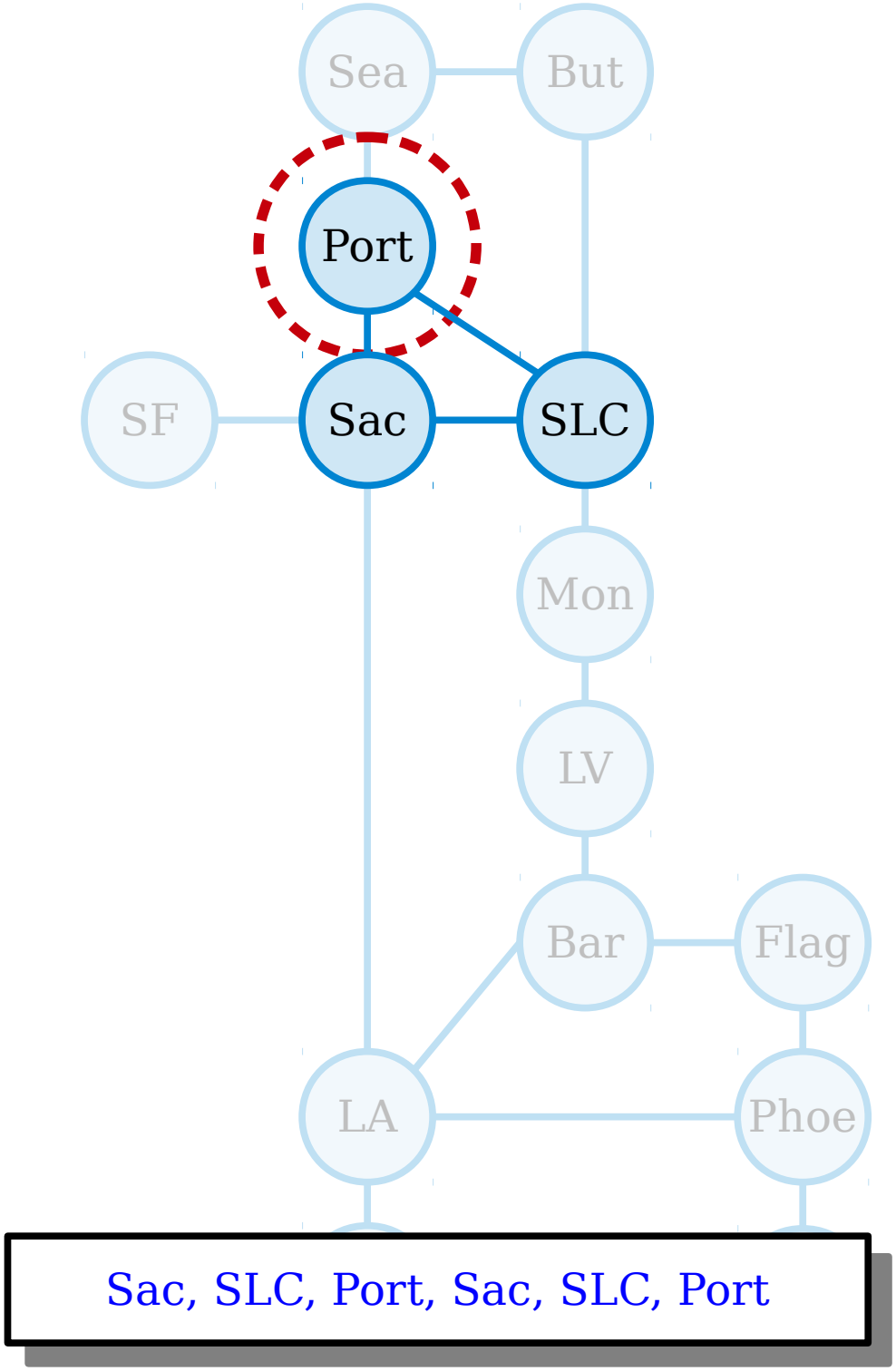
Sac, SLC, Port, Sac, SLC

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

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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.



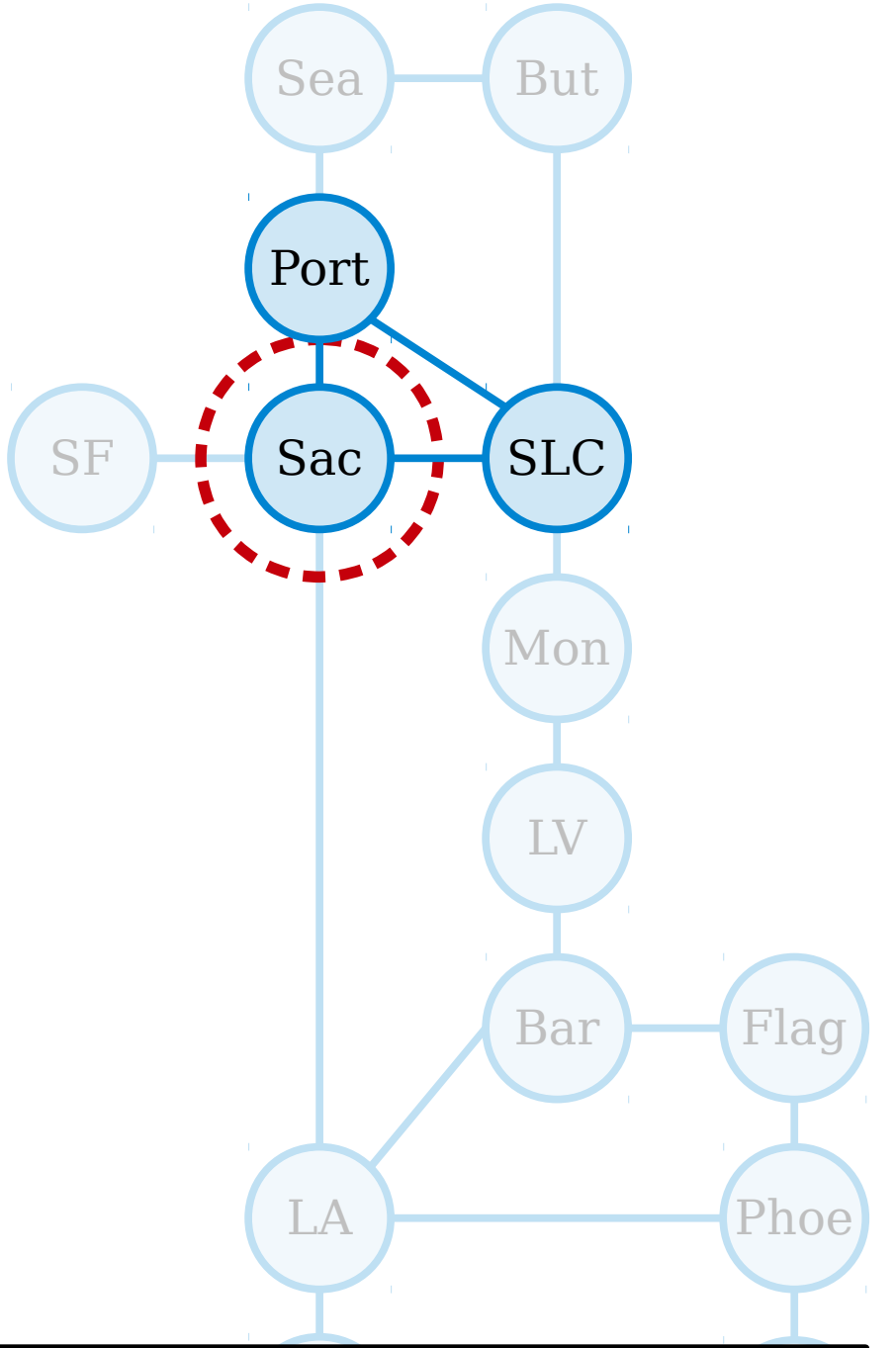
Sac, SLC, Port, Sac, SLC, Port

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

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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.



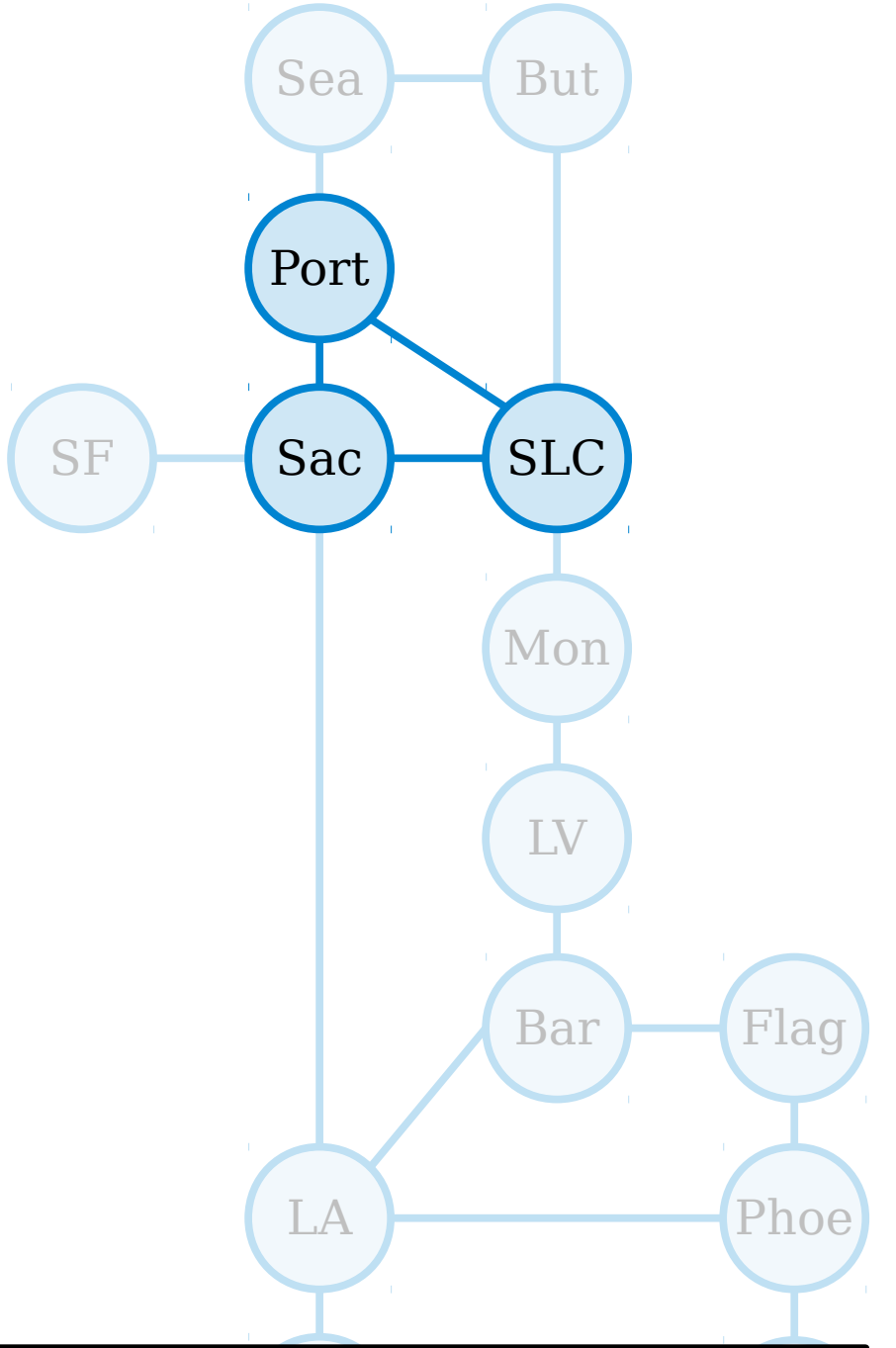
Sac, SLC, Port, Sac, SLC, Port, Sac

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

The ***length*** of the path v_1, …, v_n is n – 1.

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.



Sac, SLC, Port, Sac, SLC, Port, Sac

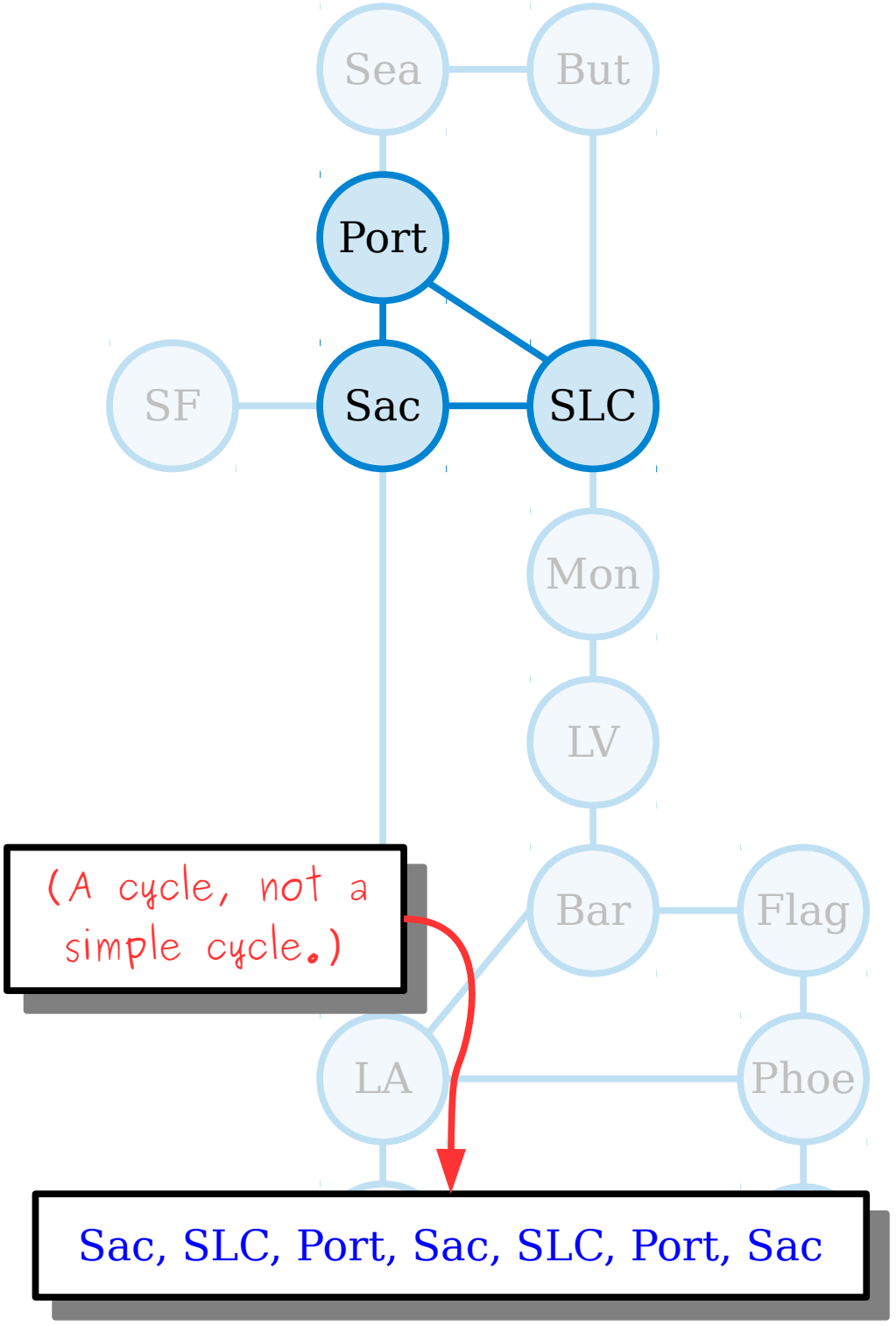
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.

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.



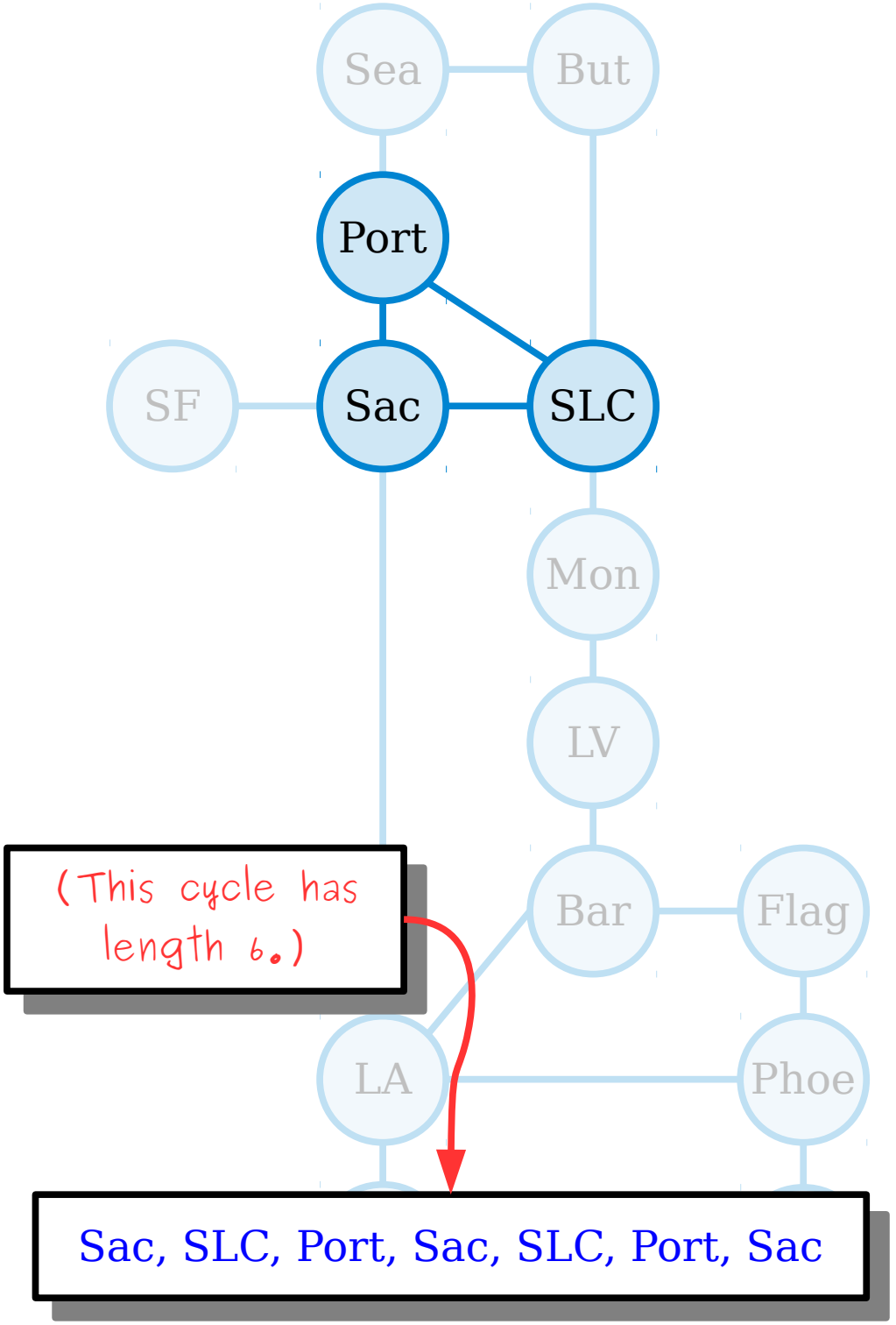
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, v_n such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path v_1, …, v_n is $n - 1$.

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.



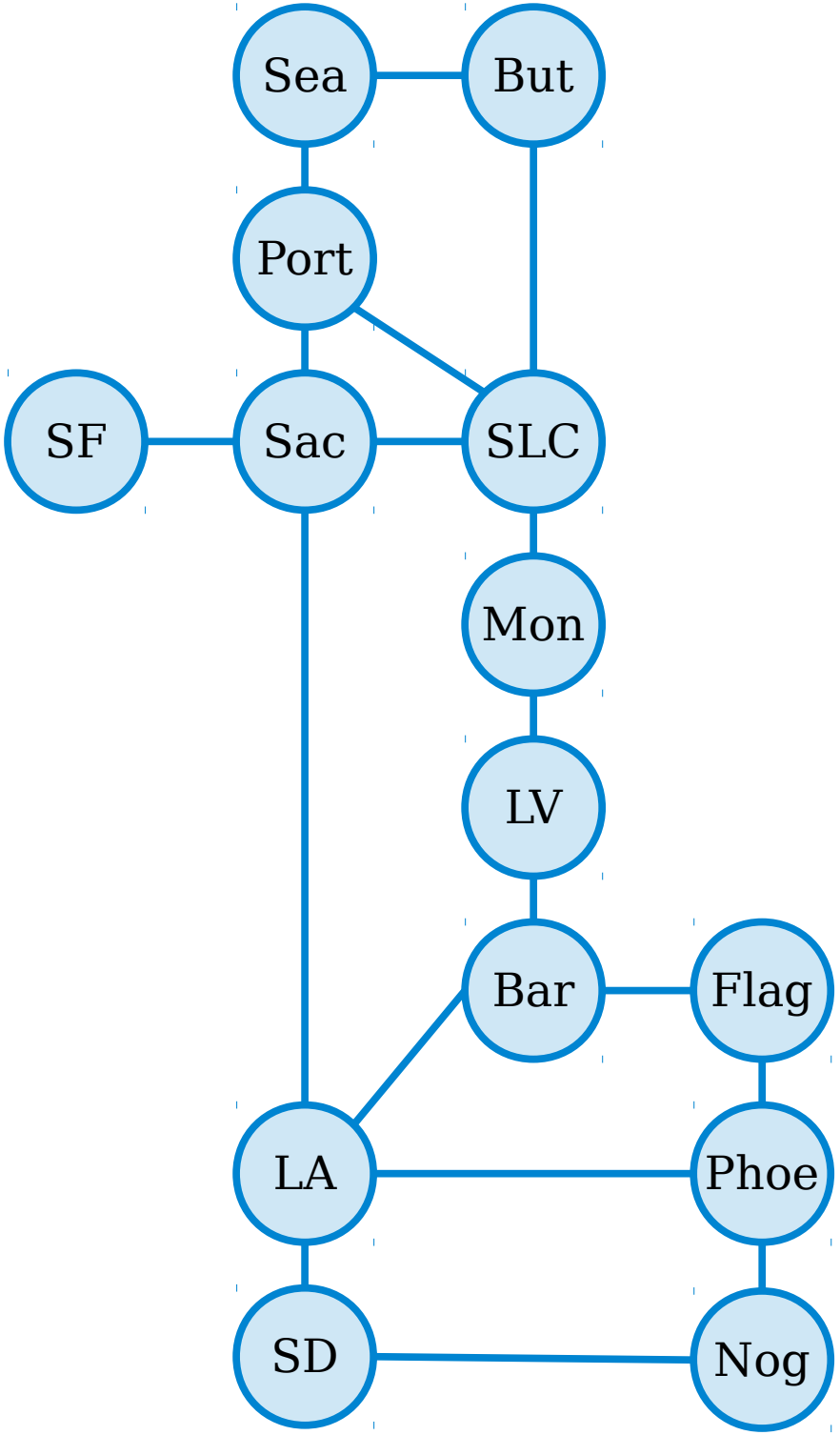
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, v_n such that any two consecutive nodes in the sequence are adjacent.

The ***length*** of the path v_1, …, v_n is $n - 1$.

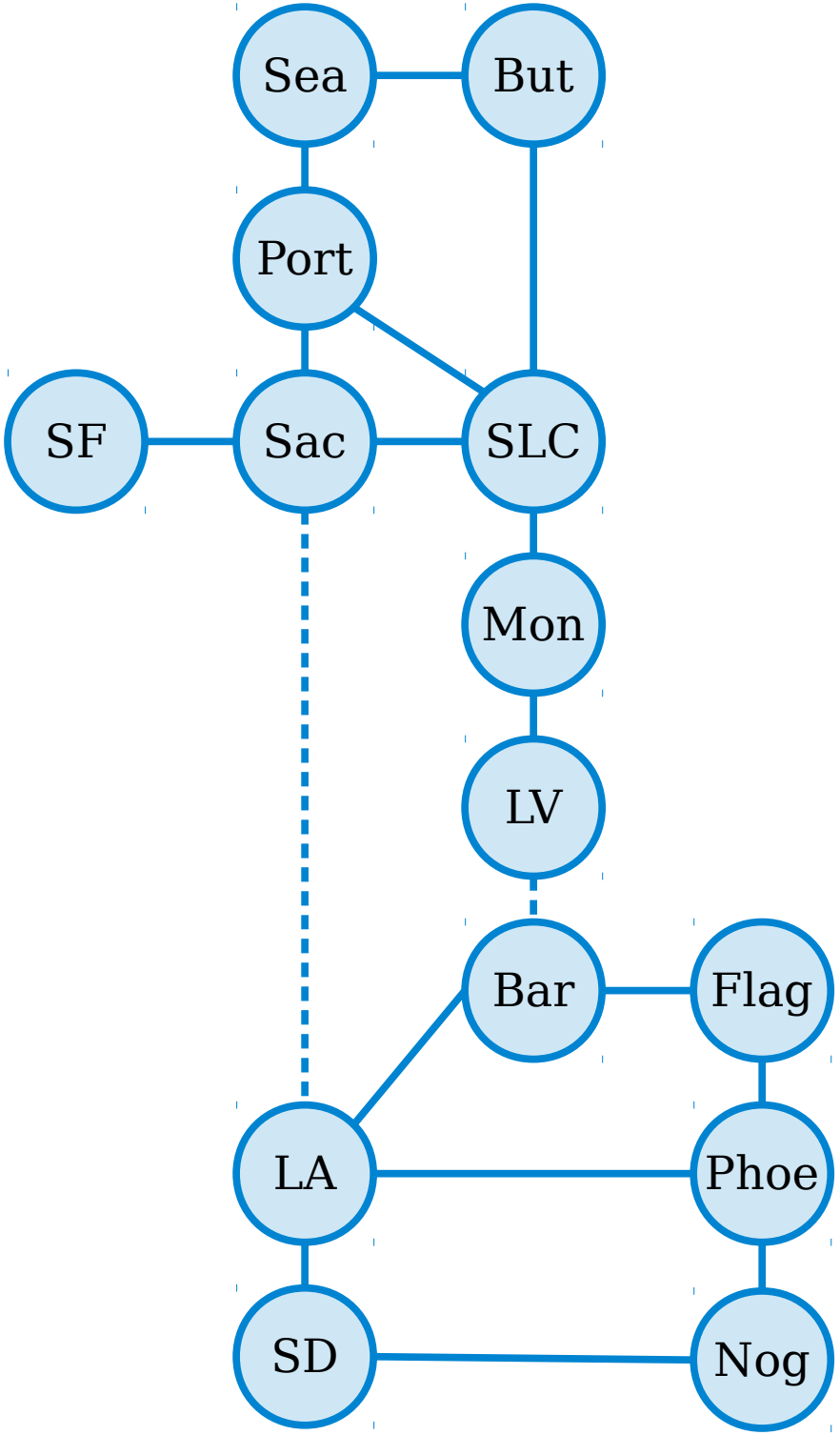
A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)

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

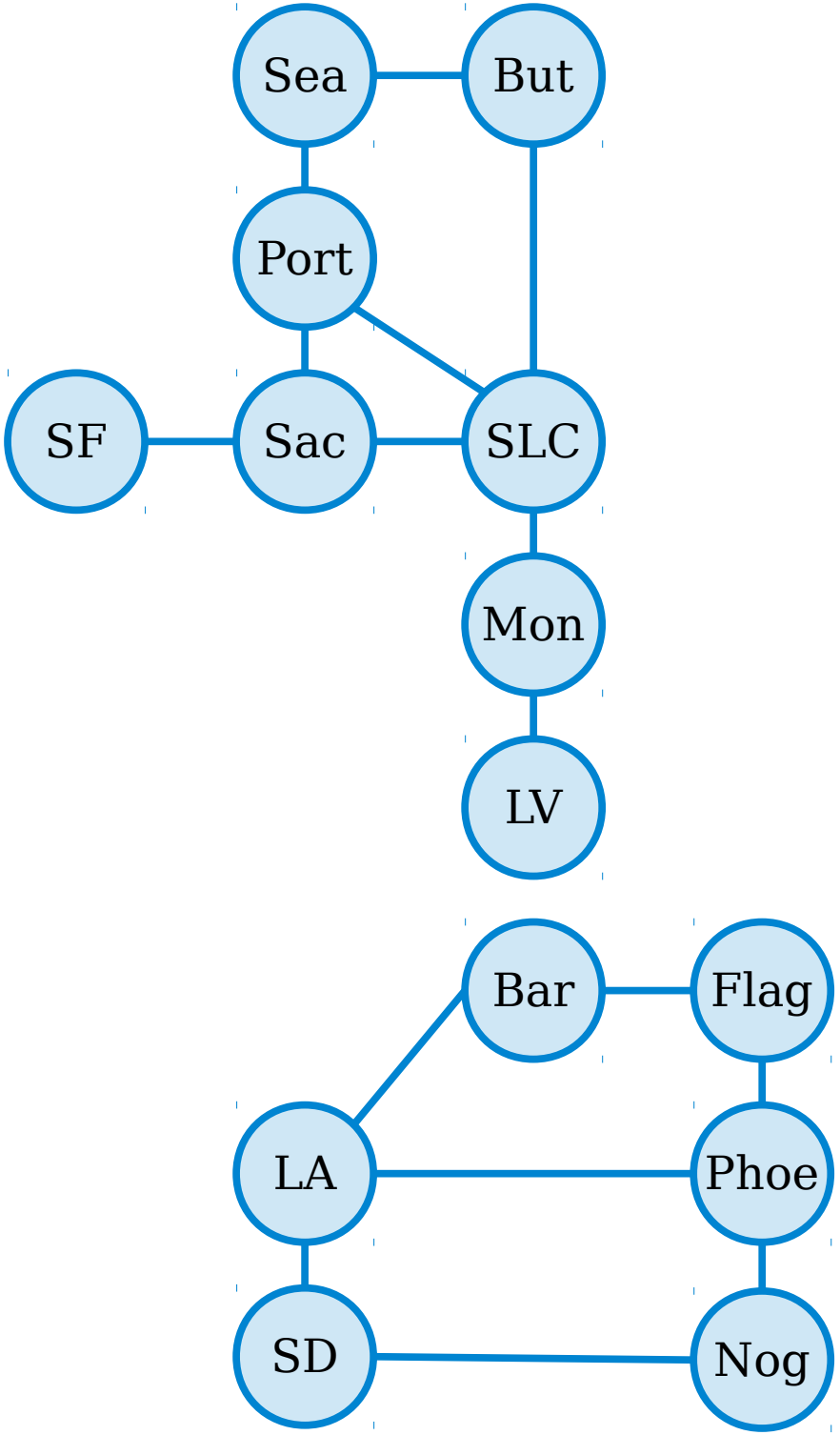
A ***simple cycle*** in a graph is cycle that does not repeat any nodes or edges except the first/last node.



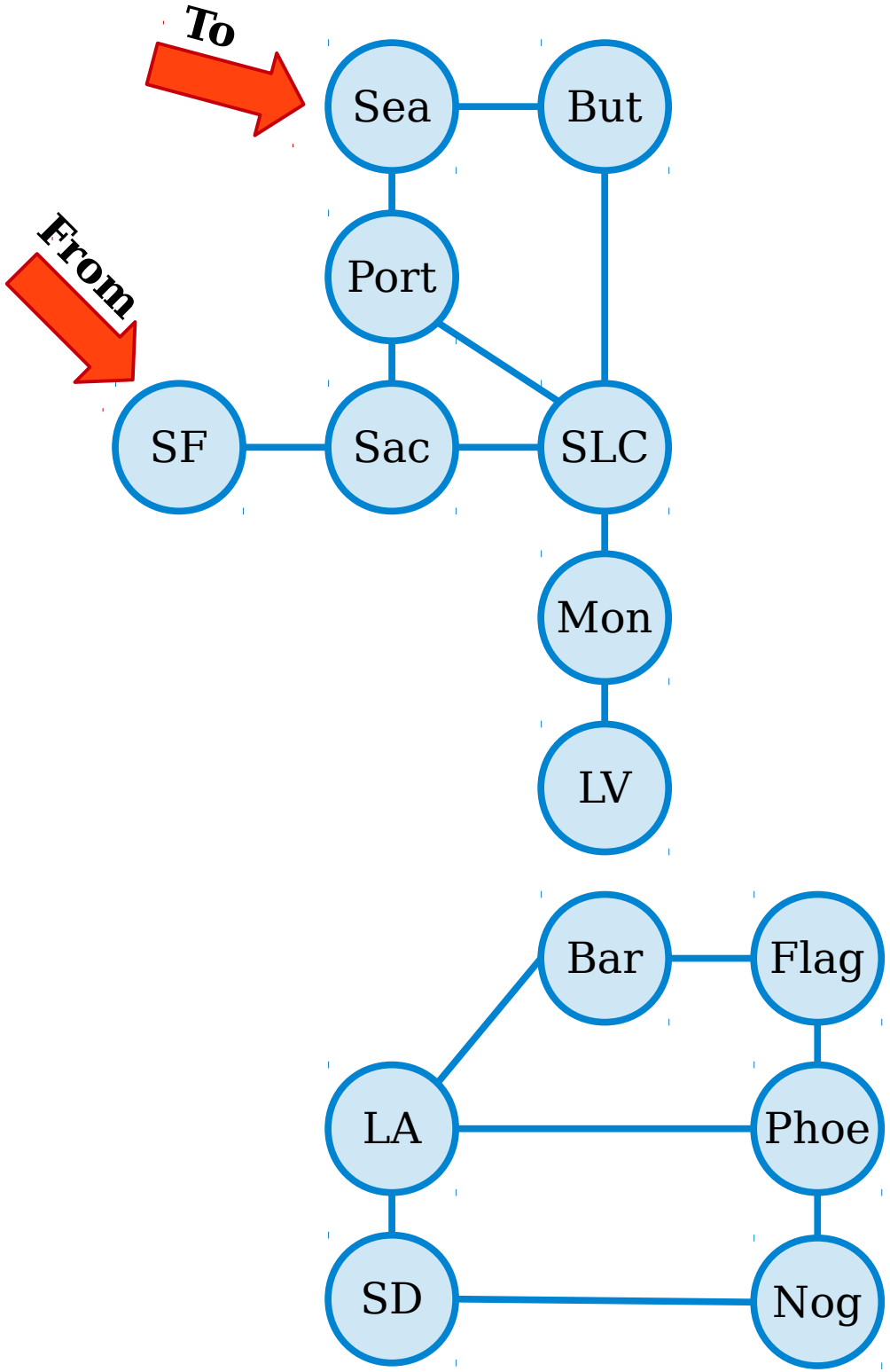
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.



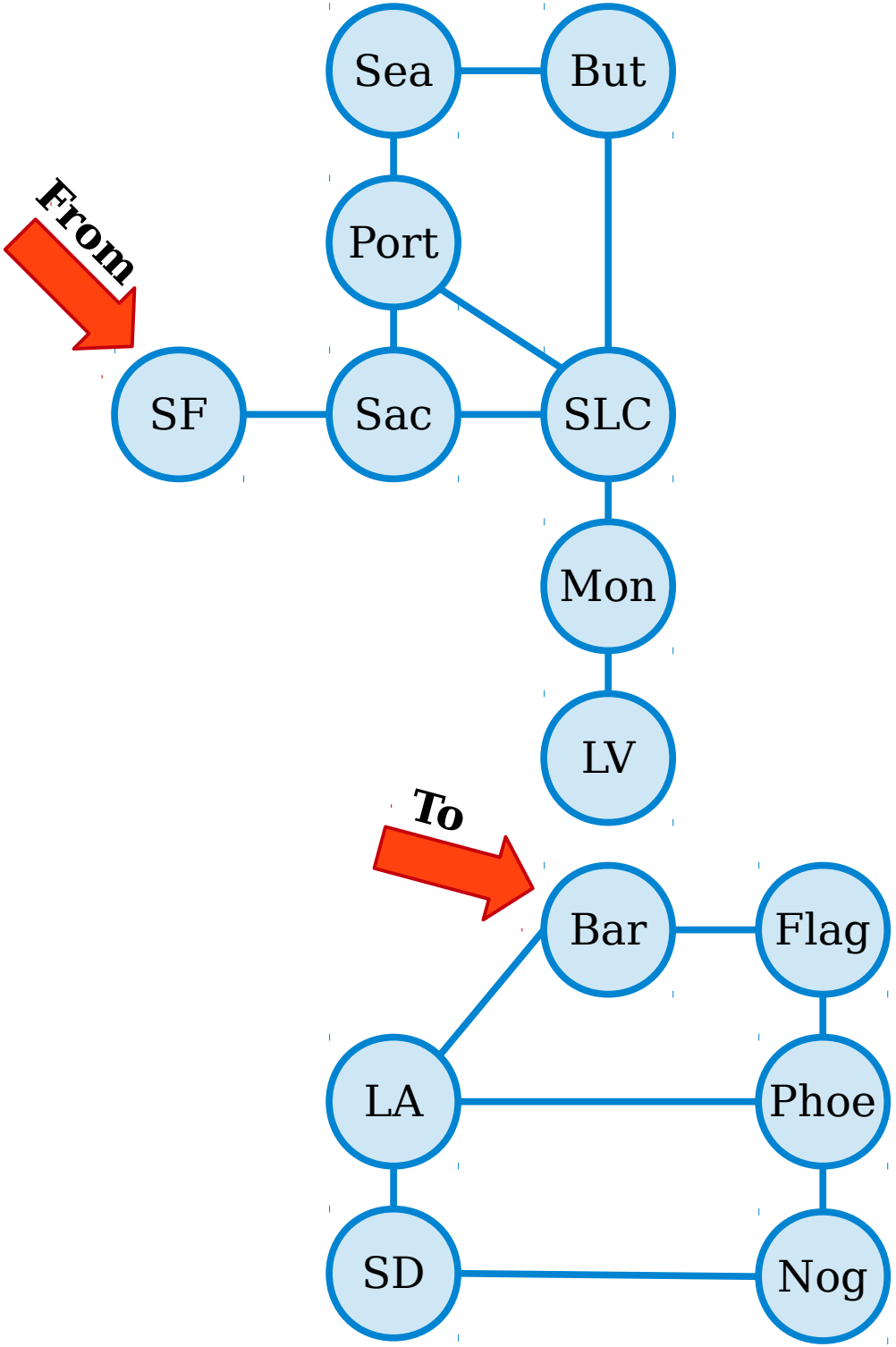
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.



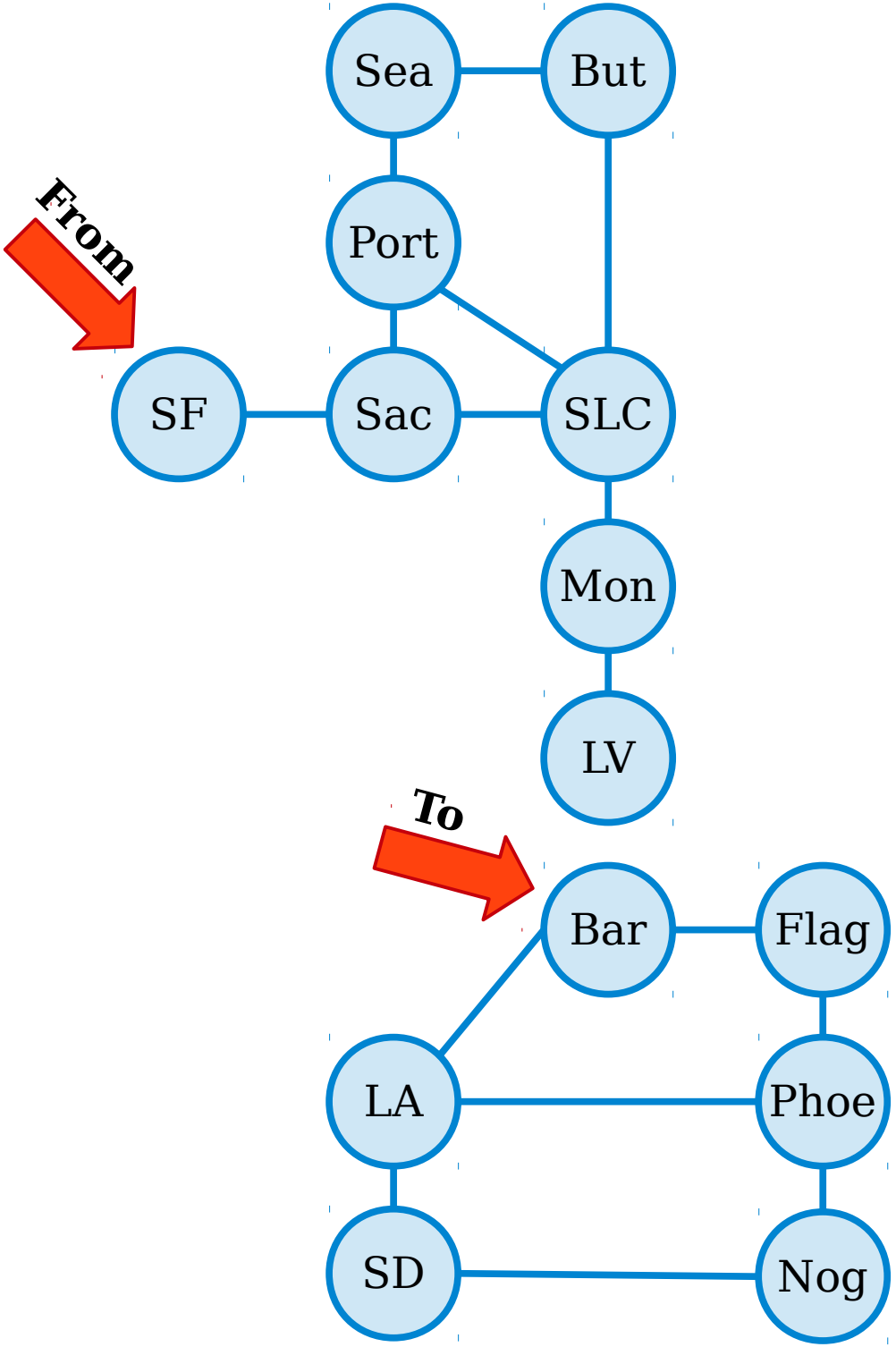
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, v_n such that any two consecutive nodes in the sequence are adjacent.



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

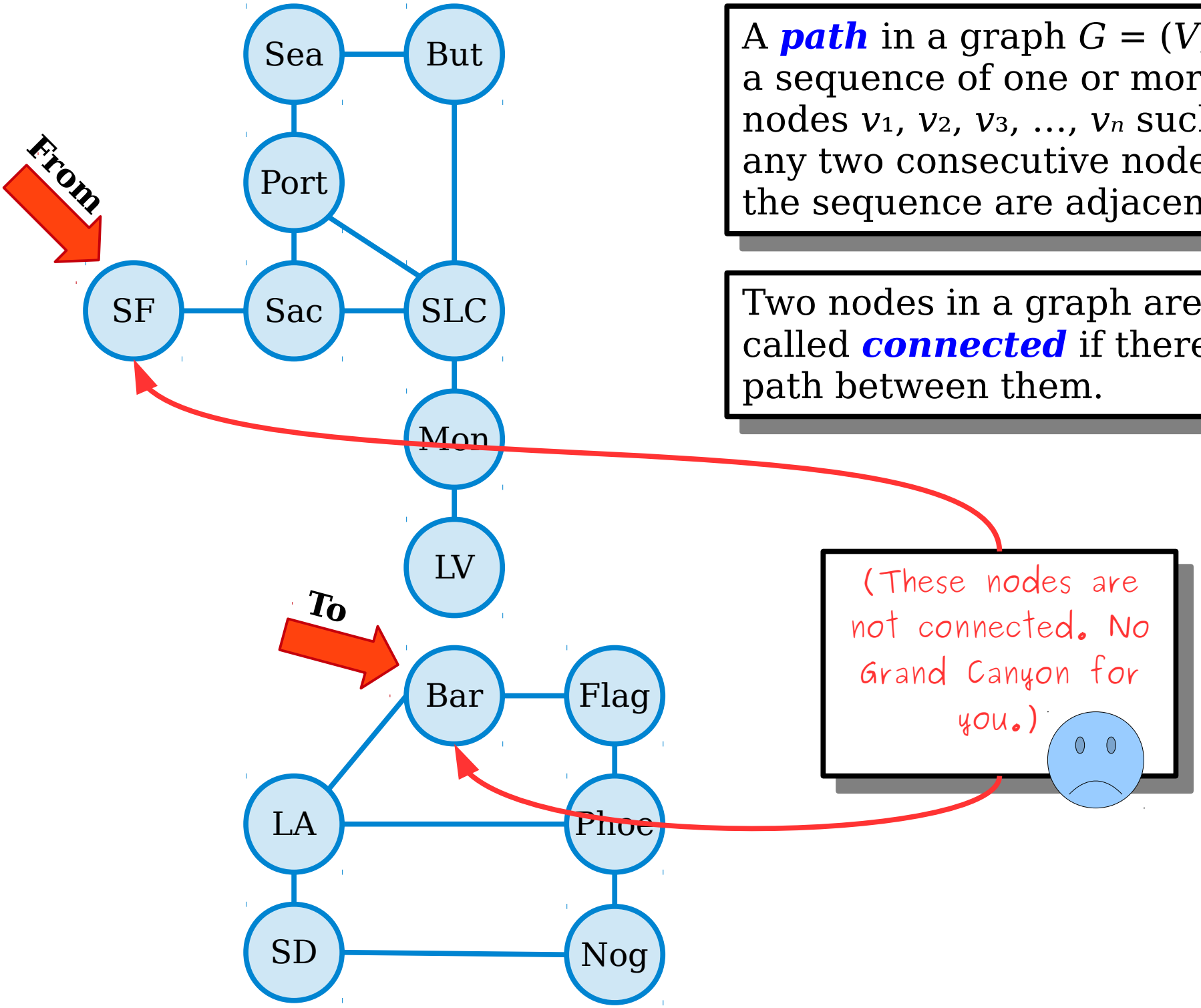


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, v_n such that any two consecutive nodes in the sequence are adjacent.



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

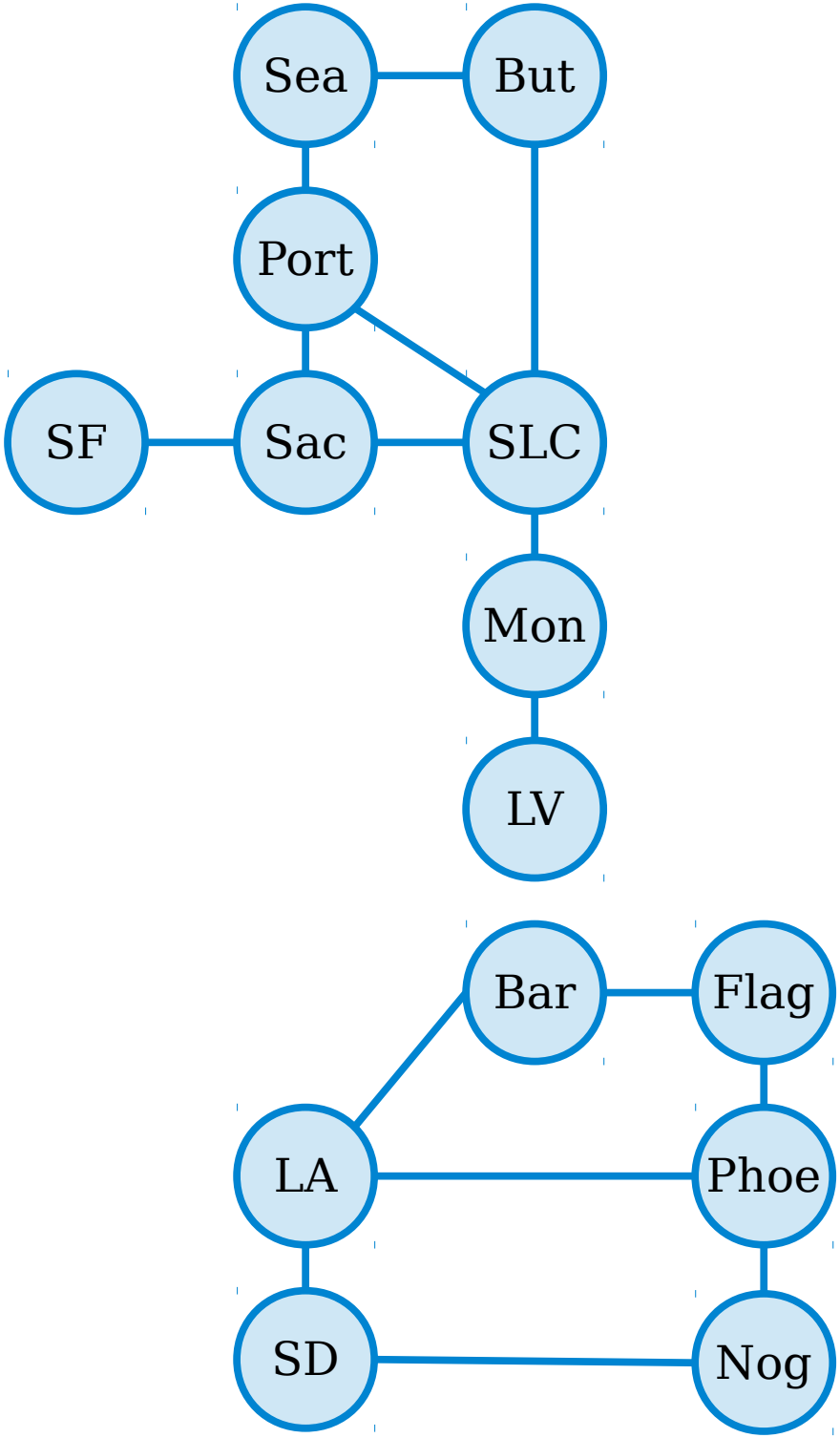
Two nodes in a graph are called ***connected*** if there is a path between them.



A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, 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.

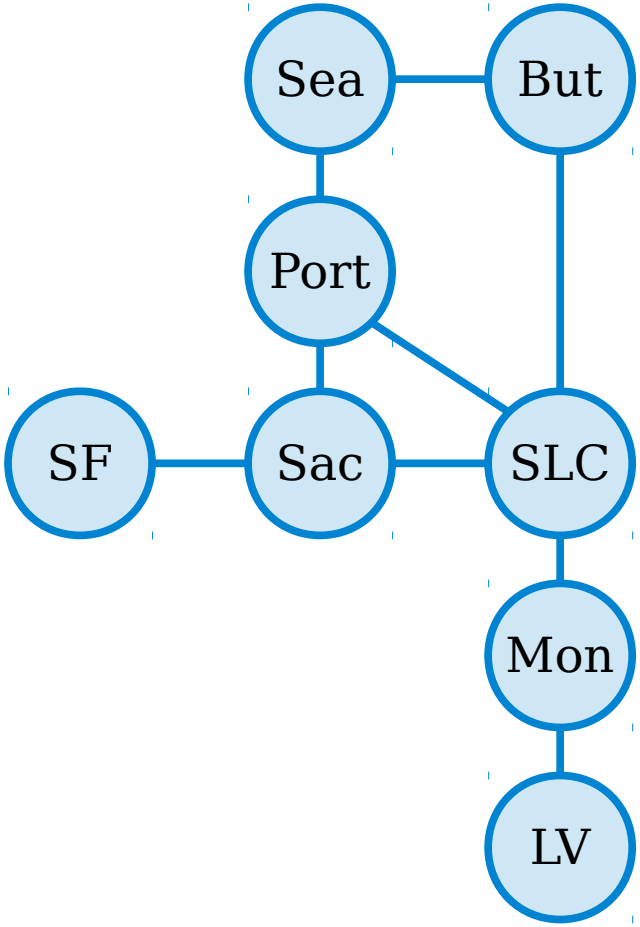
(These nodes are not connected. No Grand Canyon for you.)

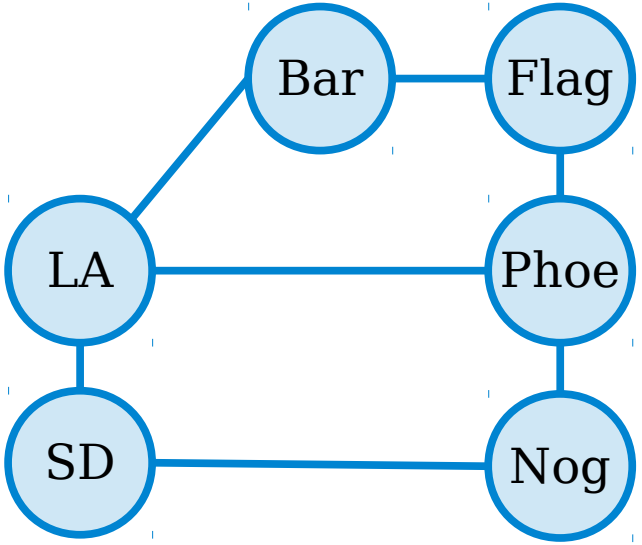


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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.





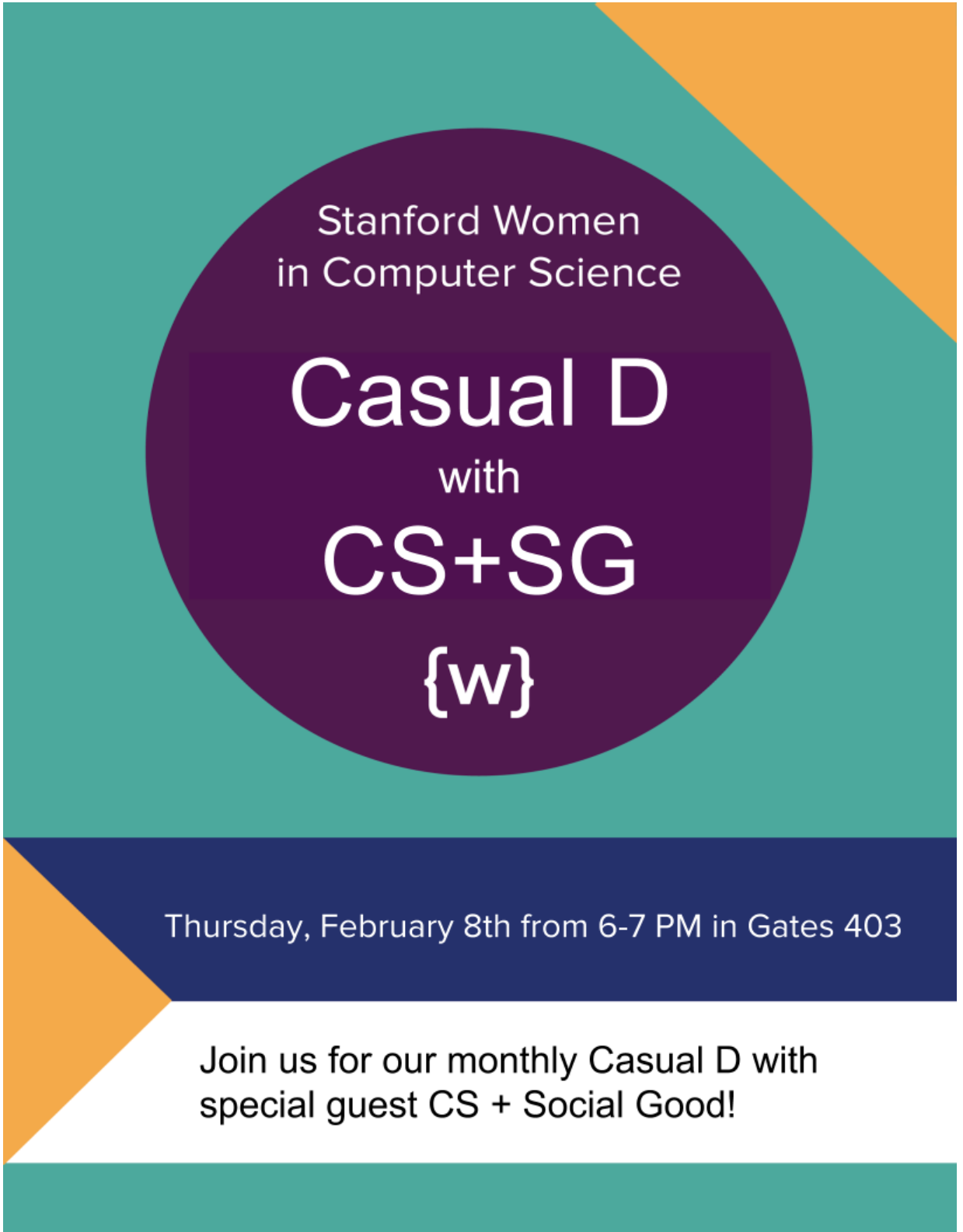
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes v_1, v_2, v_3, …, v_n such that any two consecutive nodes in the sequence are adjacent.

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.

(This graph is not connected.)

Time-Out for Announcements!


Stanford Women
in Computer Science
Casual D
with
CS+SG
{w}
Thursday, February 8th from 6-7 PM in Gates 403
Join us for our monthly Casual D with
special guest CS + Social Good!

Problem Sets

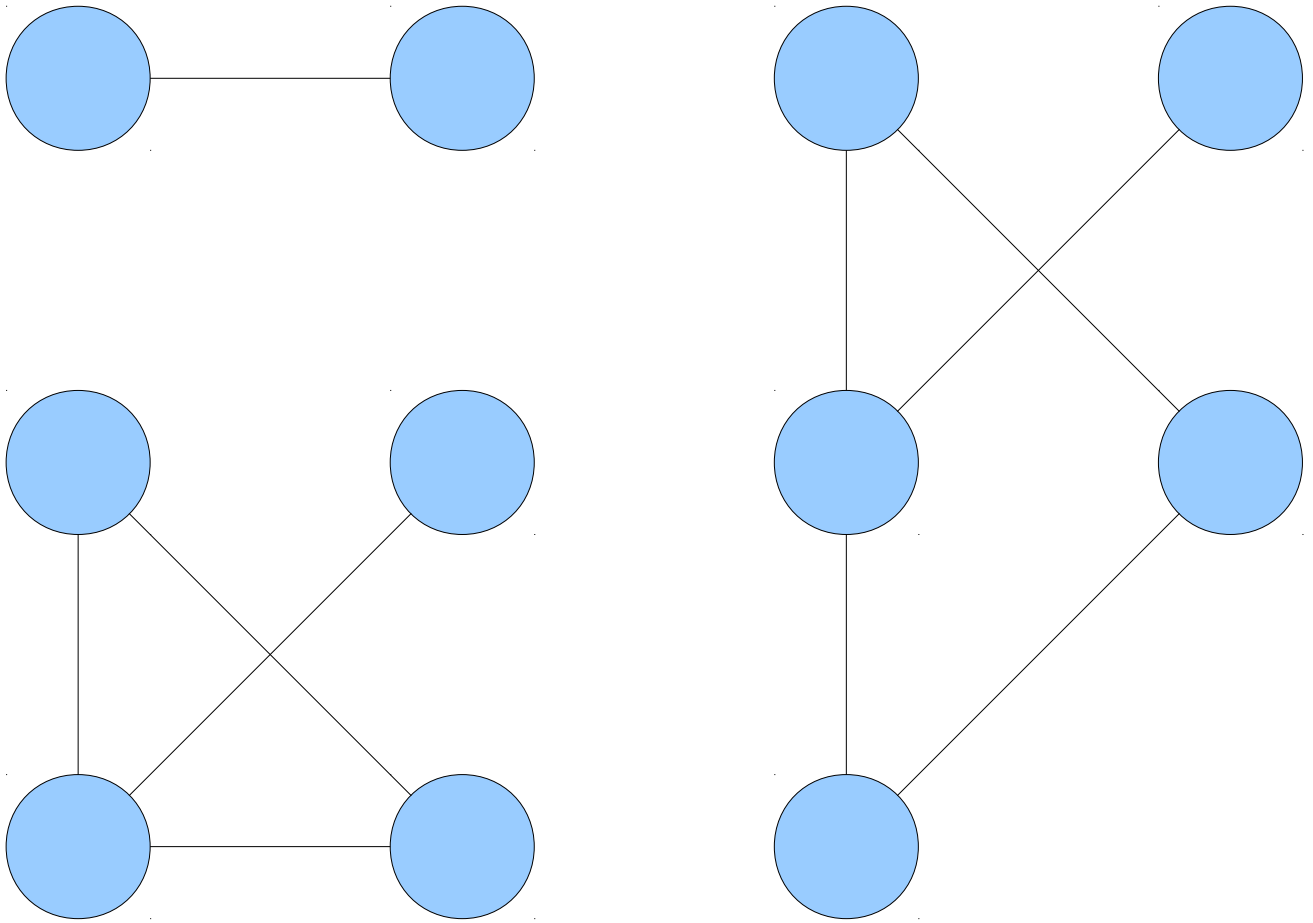
- Problem Set Three was due today at 2:30PM.
  - You can extend the deadline to Sunday at 2:30PM using late days, but ***be careful*** about doing so because the exam is on Monday.
- Problem Set Four goes out today.
  - Checkpoint is due Monday at 2:30PM.
  - Remaining problems due Friday at 2:30PM.
  - Play around with functions, cardinality, graphs, and their applications!

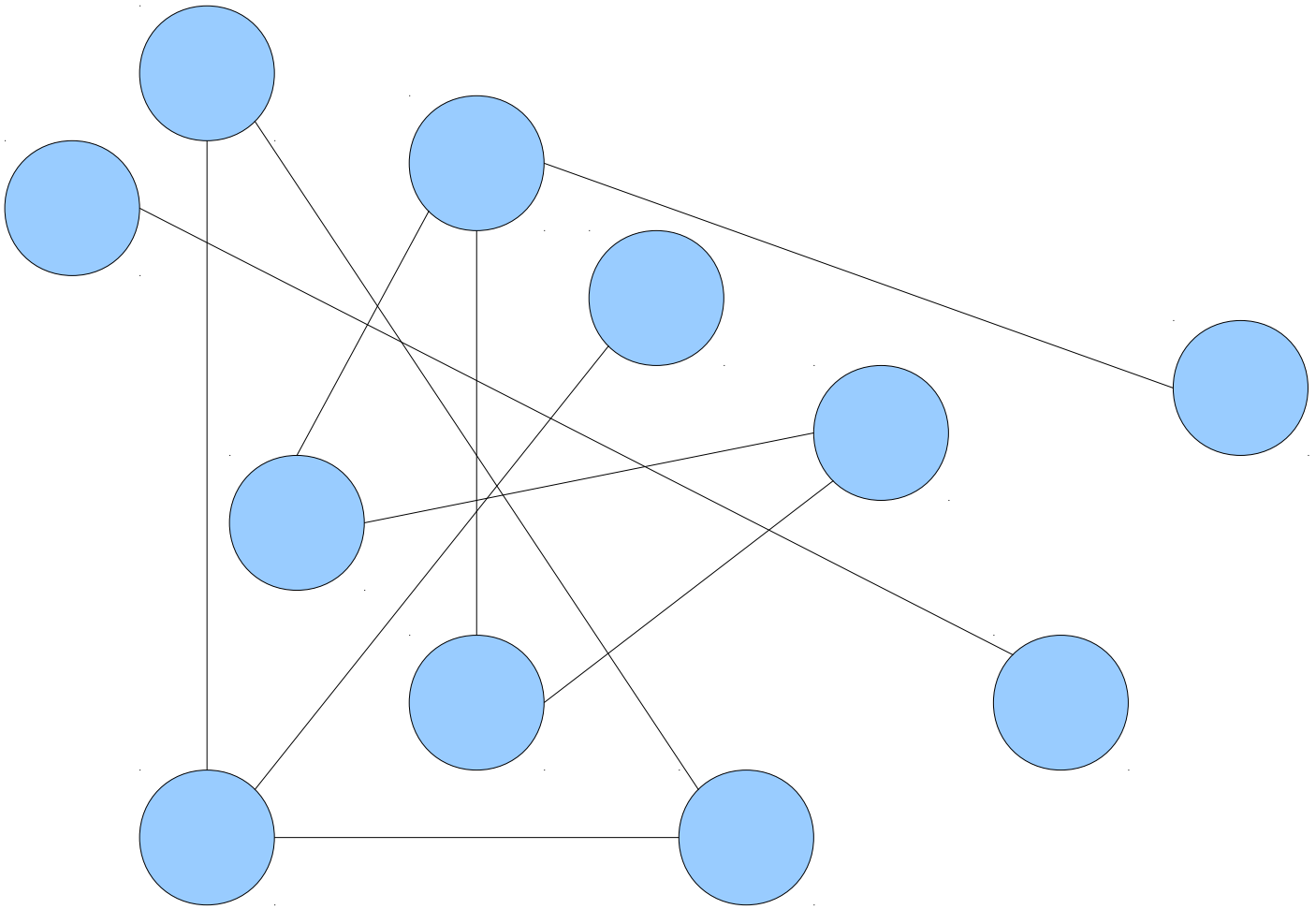
Midterm Exam Logistics

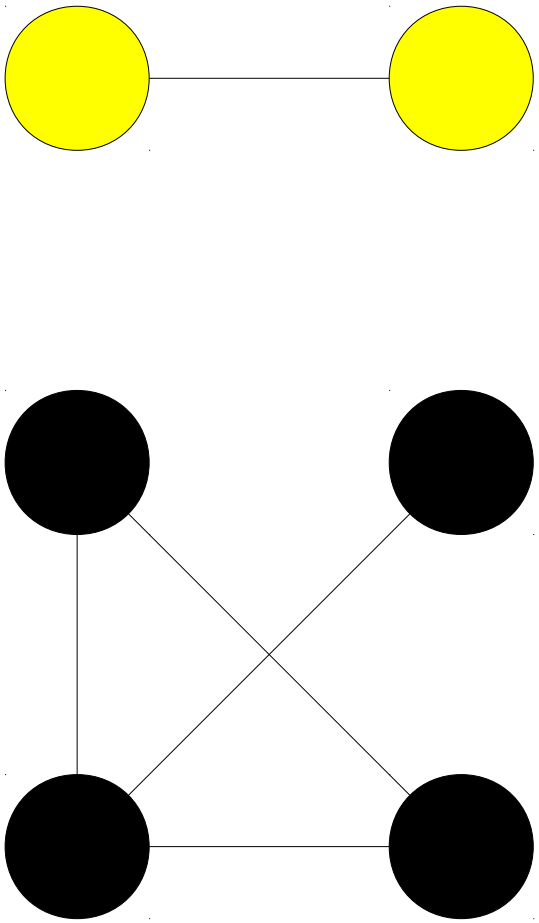
- Our first midterm exam is next ***Monday, February 5th***, from ***7:00PM – 10:00PM***. Locations are divvied up by last (family) name:
  - `A – H`: Go to Cubberley Auditorium.
  - `I – Z`: Go to 320-105.
- You're responsible for Lectures 00 – 05 and topics covered in PS1 – PS2. Later lectures (relations forward) and problem sets (PS3 onward) won't be tested here.
- 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.

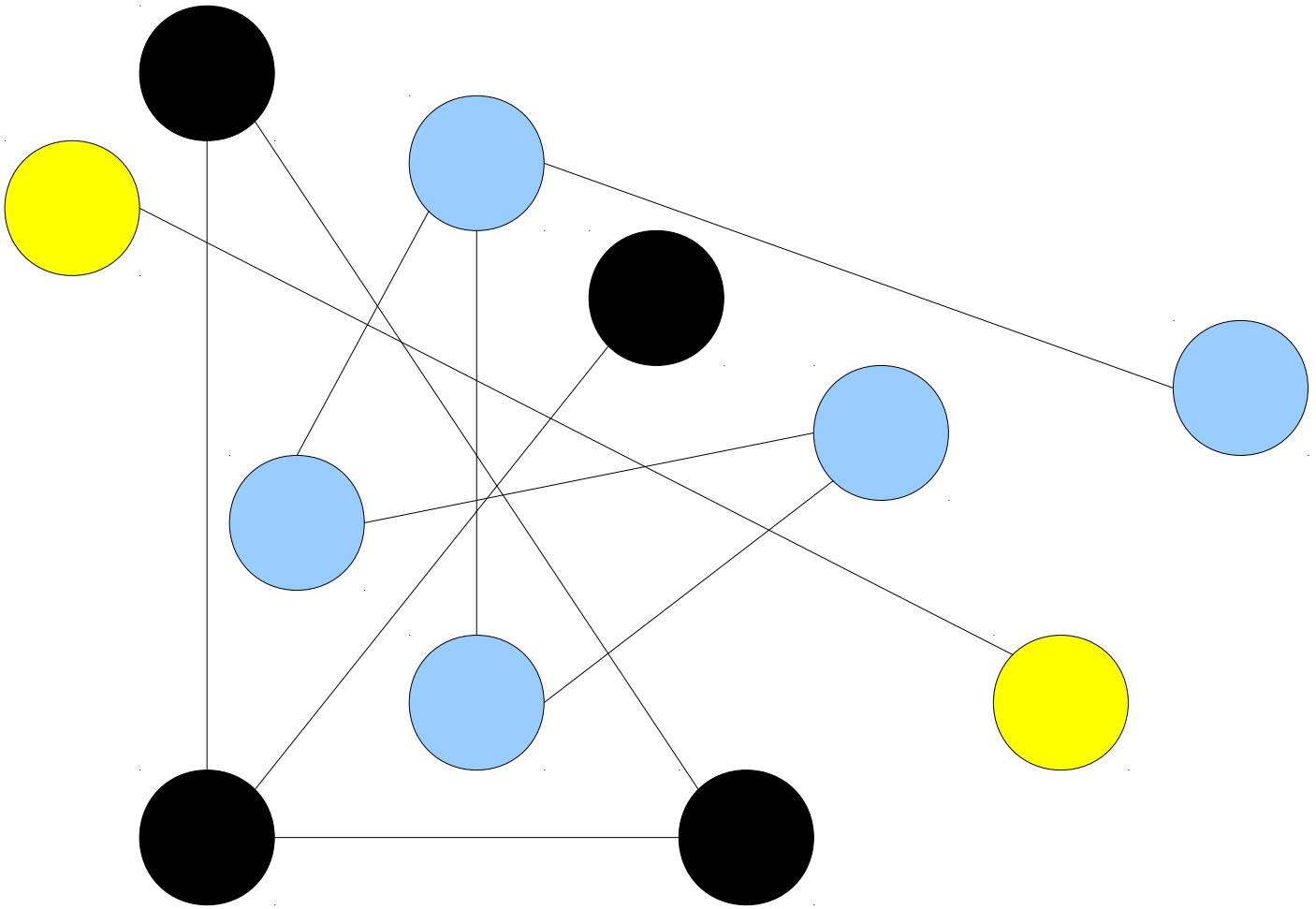
Back to CS103!

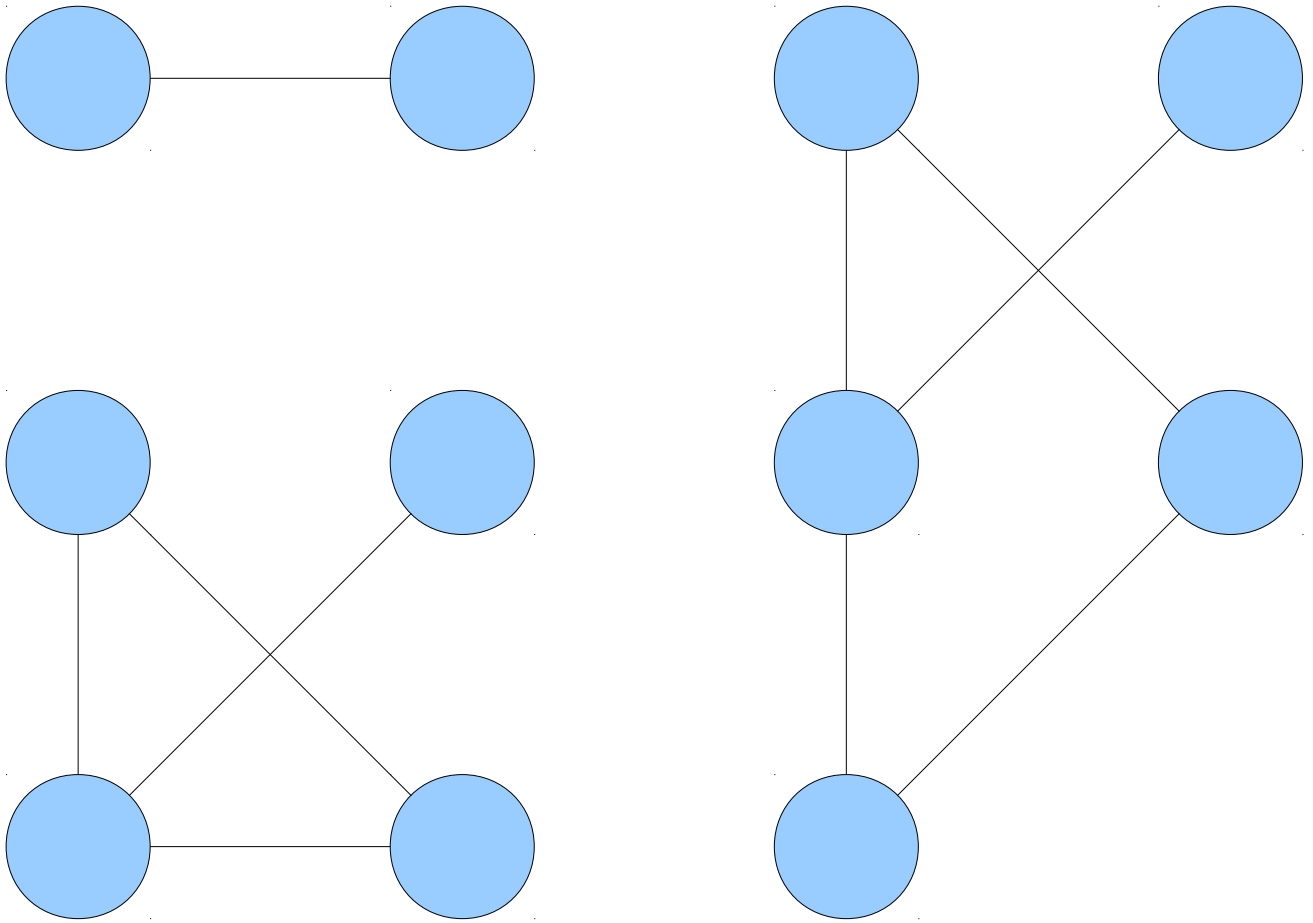
Connected Components

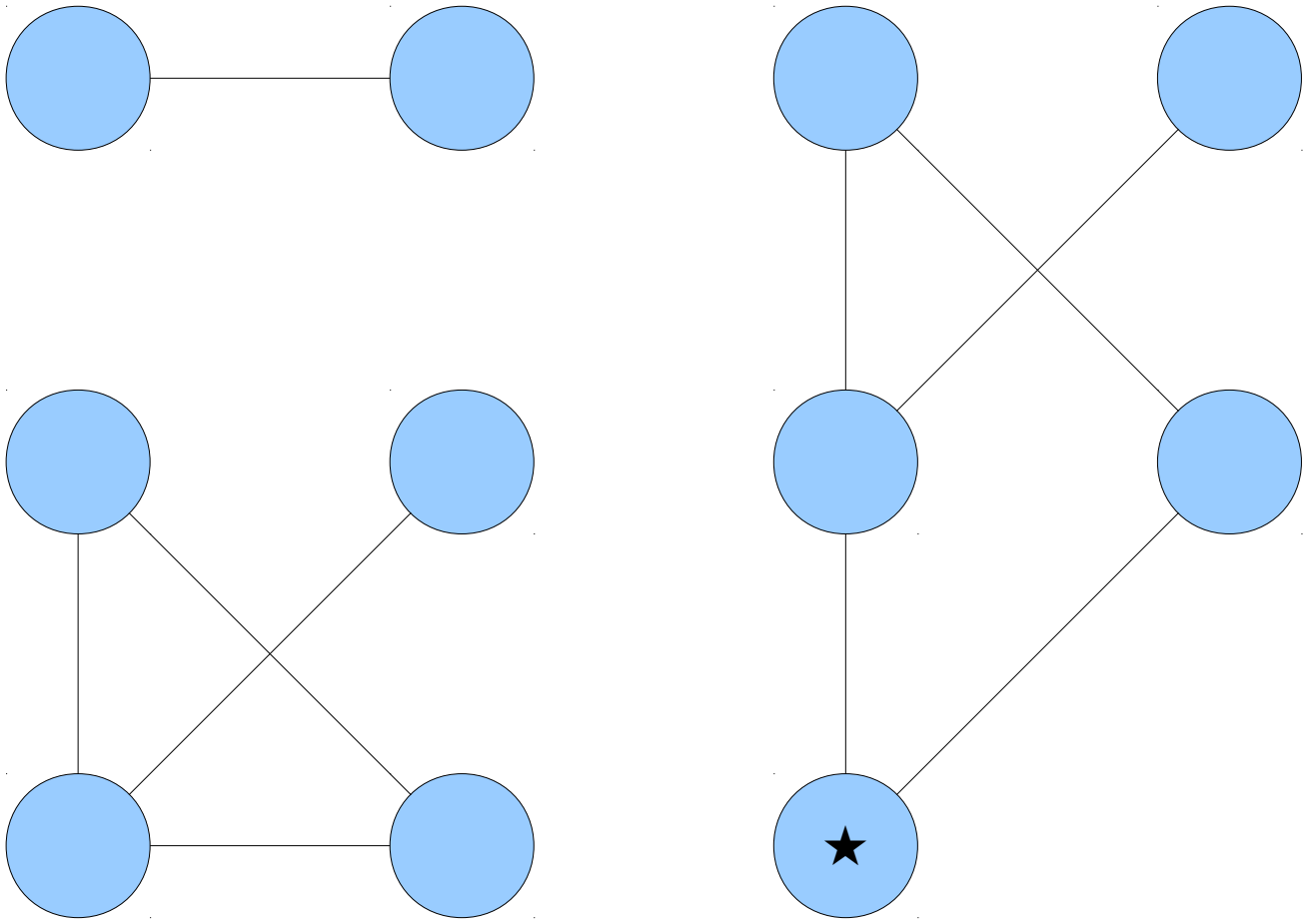


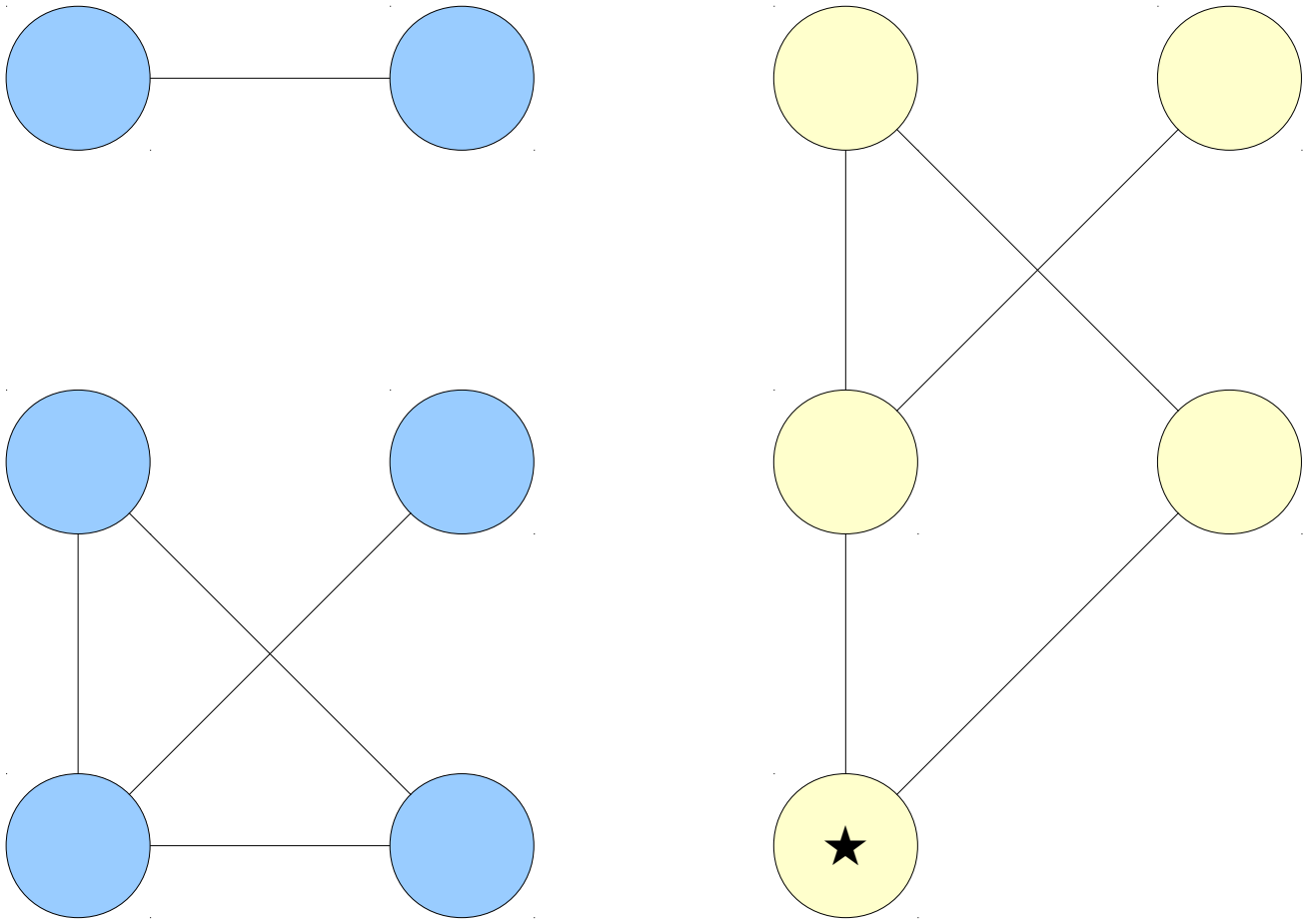












Connected Components

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

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

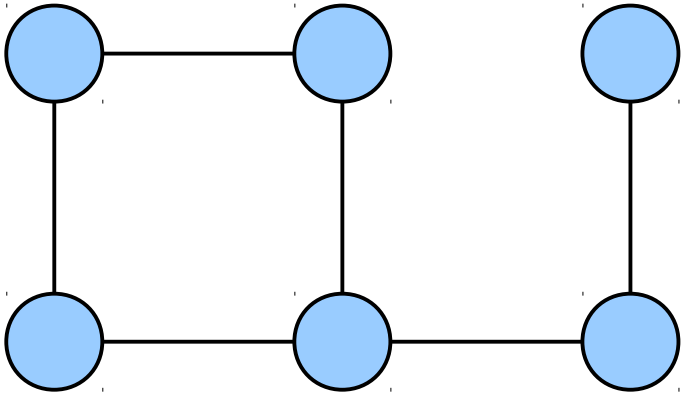
- Intuitively, a connected component is a "piece" of a graph in the sense we just talked about.
- ***Question:*** How do we know that this particular definition of a "piece" of a graph is a good one?
- ***Goal:*** Prove that any graph can be broken apart into different connected components.

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

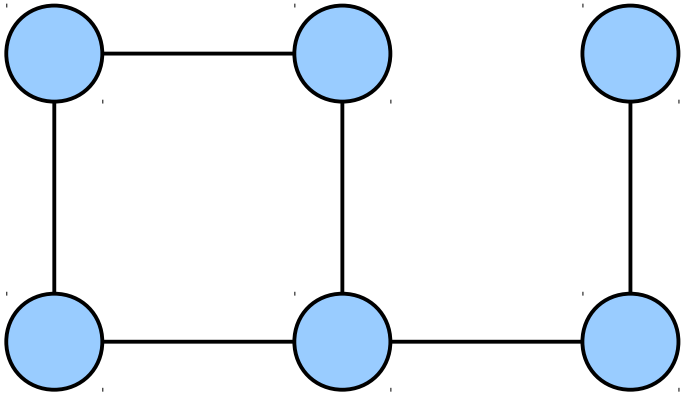
Claim: For any graph G, the "is connected to" relation is an equivalence relation.

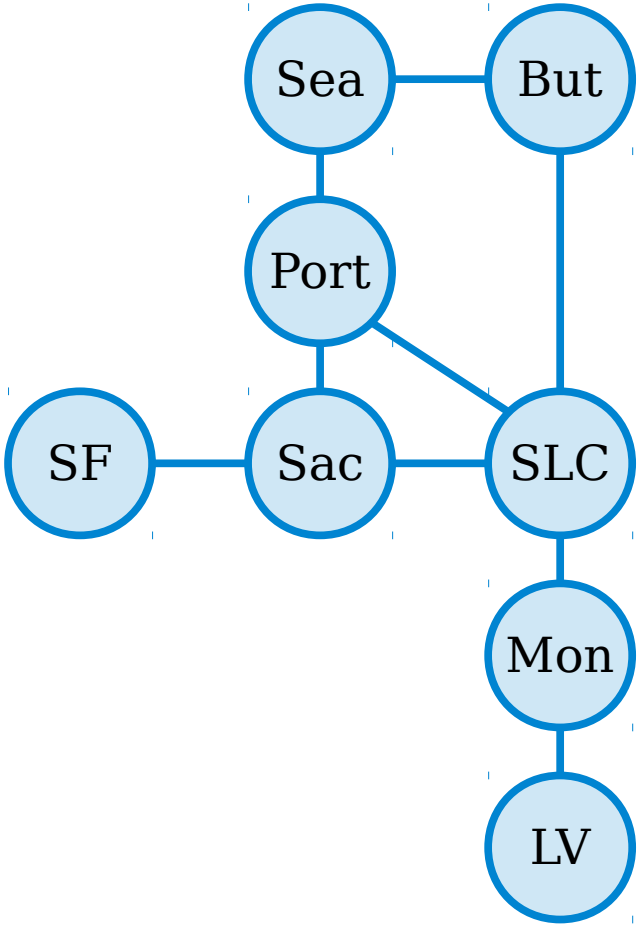
- Is it reflexive?

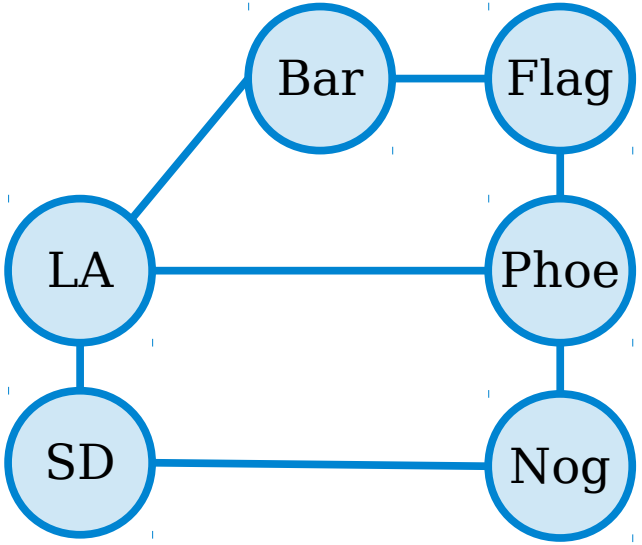
  Is it symmetric?

  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, …, 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.

(This graph is not connected.)

Connectivity

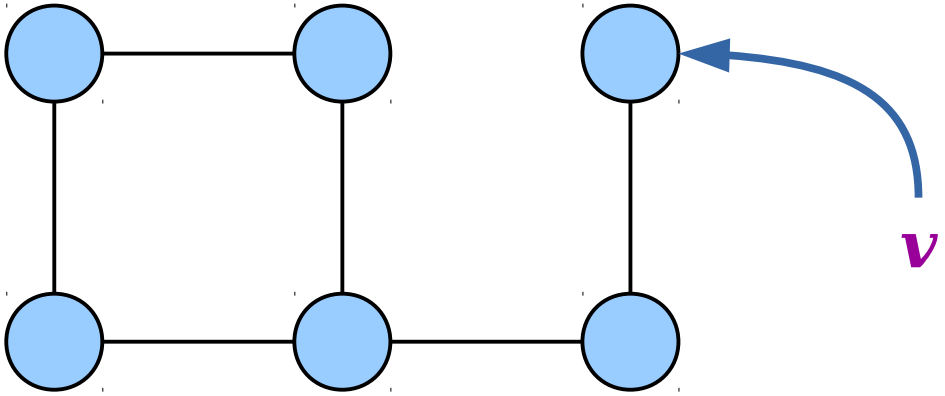
Claim: For any graph G, the "is connected to" relation is an equivalence relation.

- 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

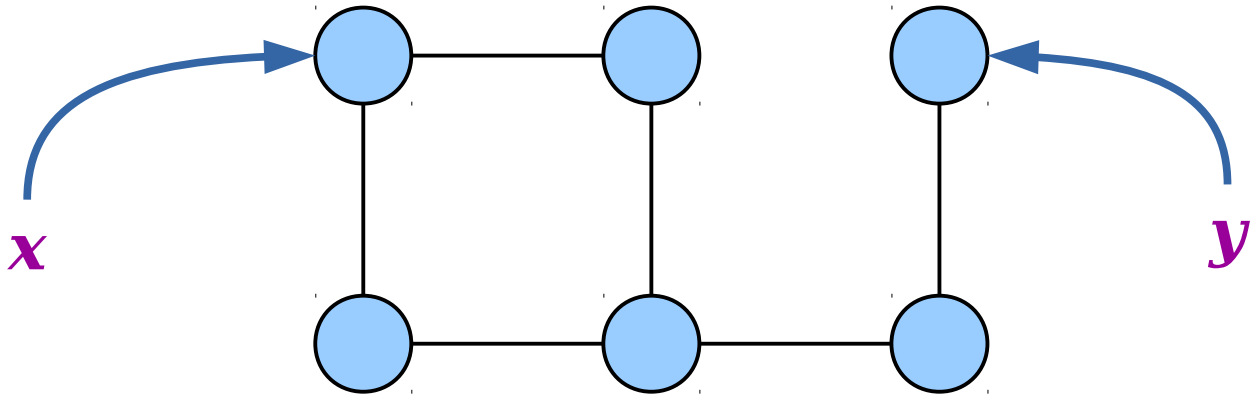
Claim: E
connecte
relation.

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

Is it reflexive?

- Is it symmetric?

Is it transitive?



Connectivity

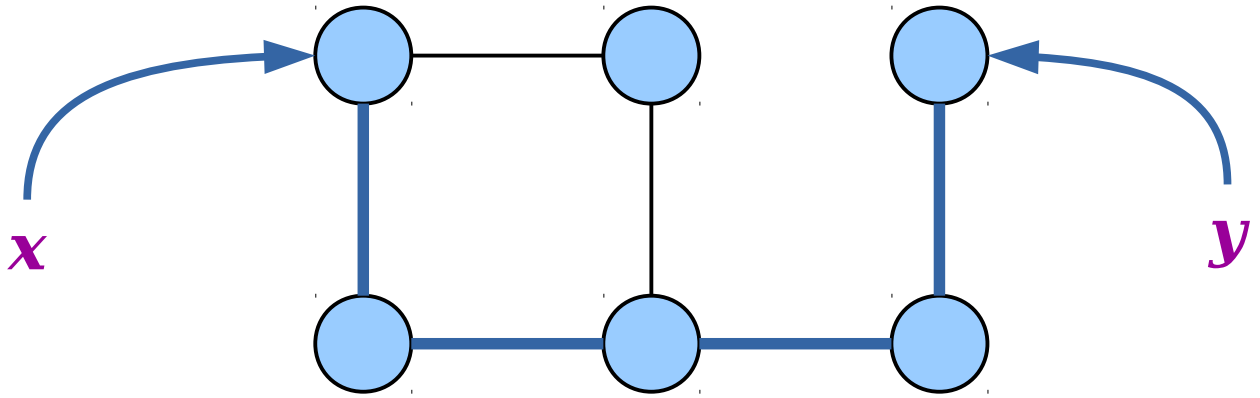
Claim: E
connecte
relation.

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

Is it reflexive?

- Is it symmetric?

Is it transitive?



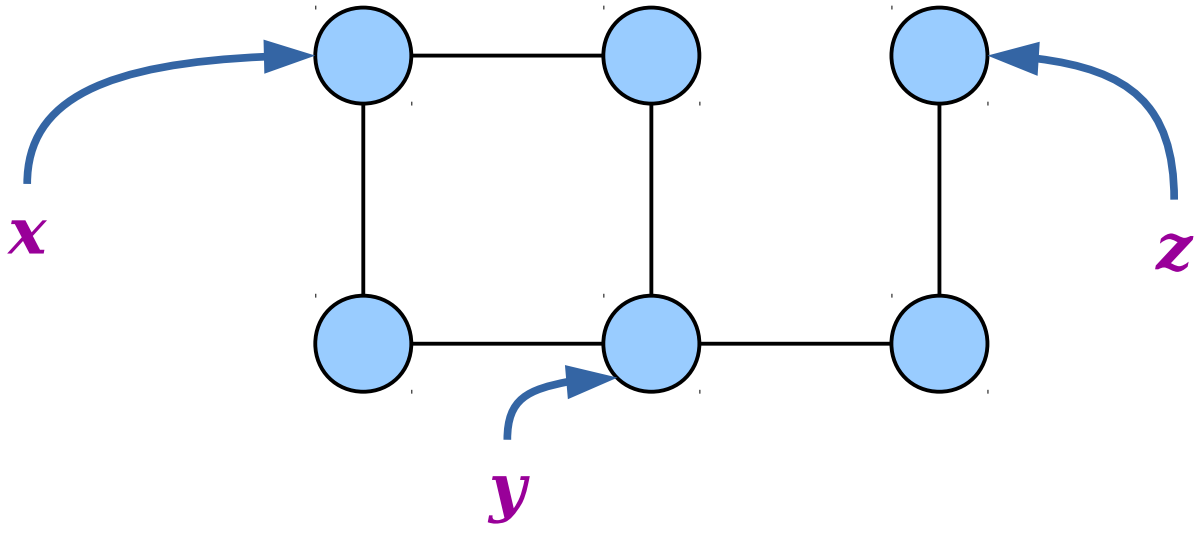
Connectivity

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

Is it reflexive?

Is it symmetric?

- Is it transitive?



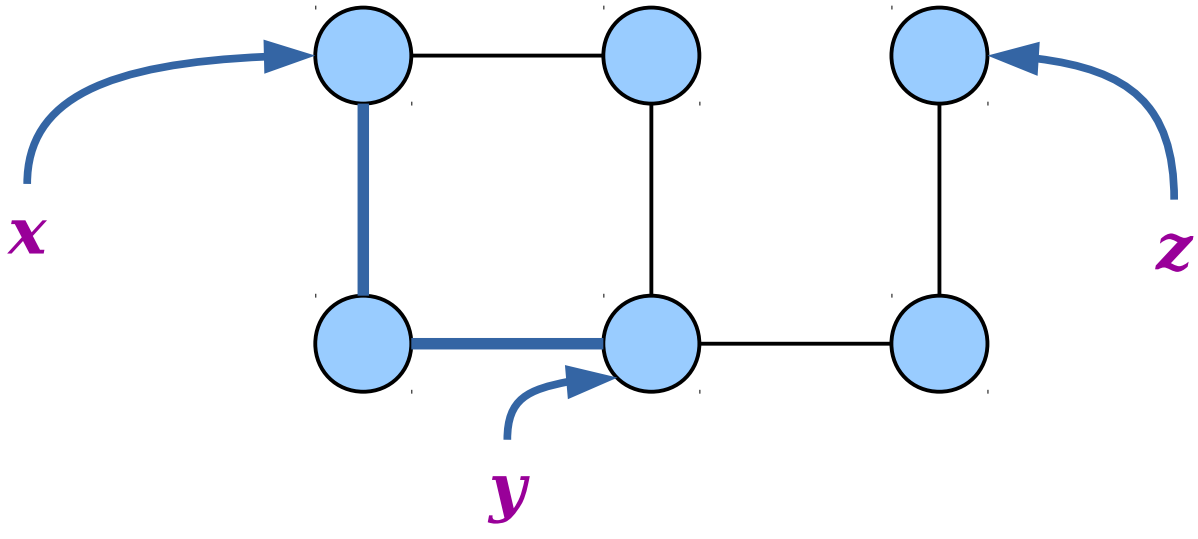
Connectivity

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

Is it reflexive?

Is it symmetric?

- Is it transitive?



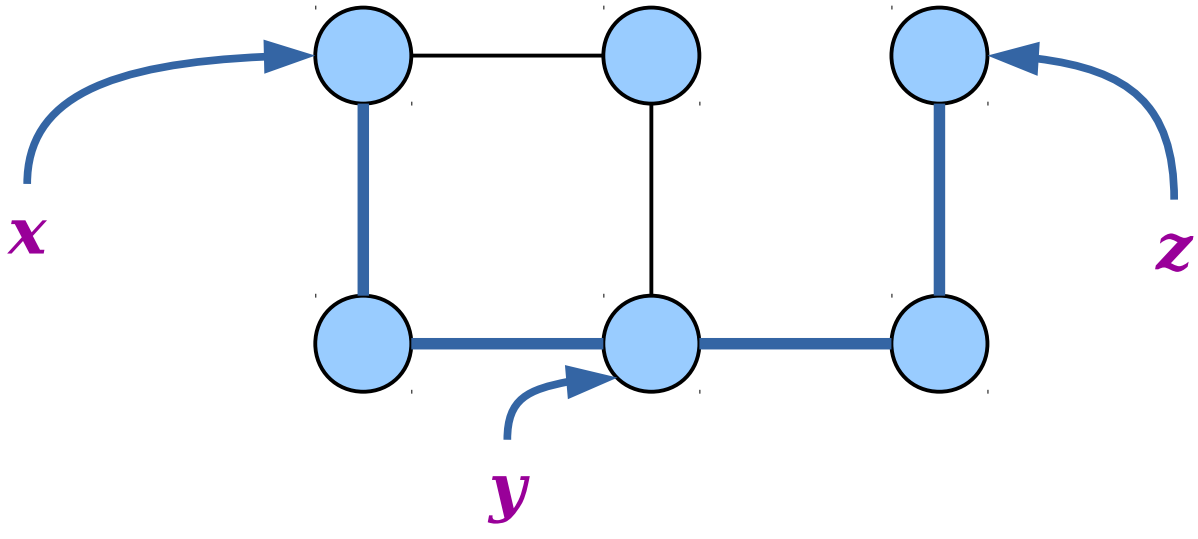
Connectivity

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

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.

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?

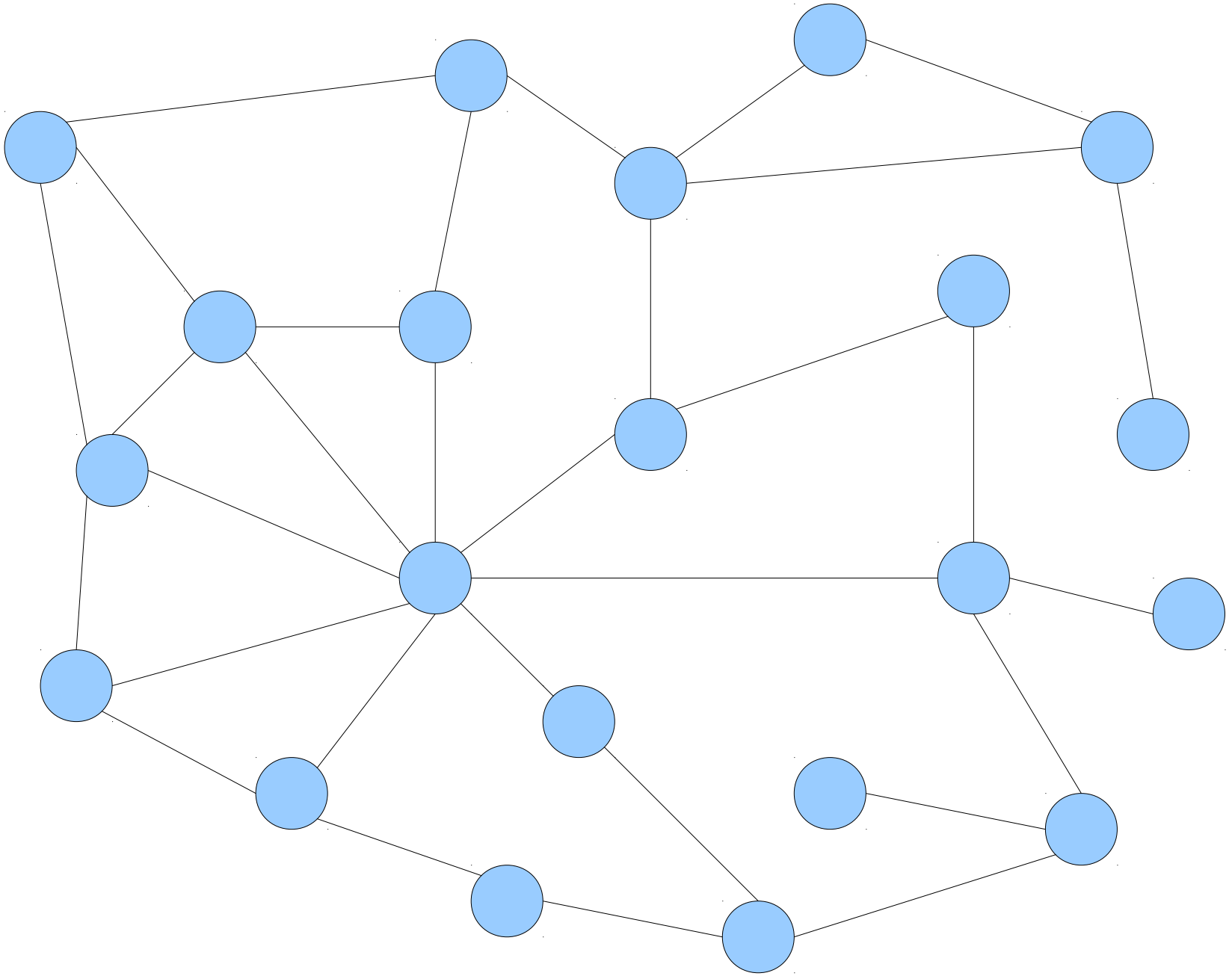
$$[v] = \{ x \in V \mid v \text{ is connected to } x \}$$

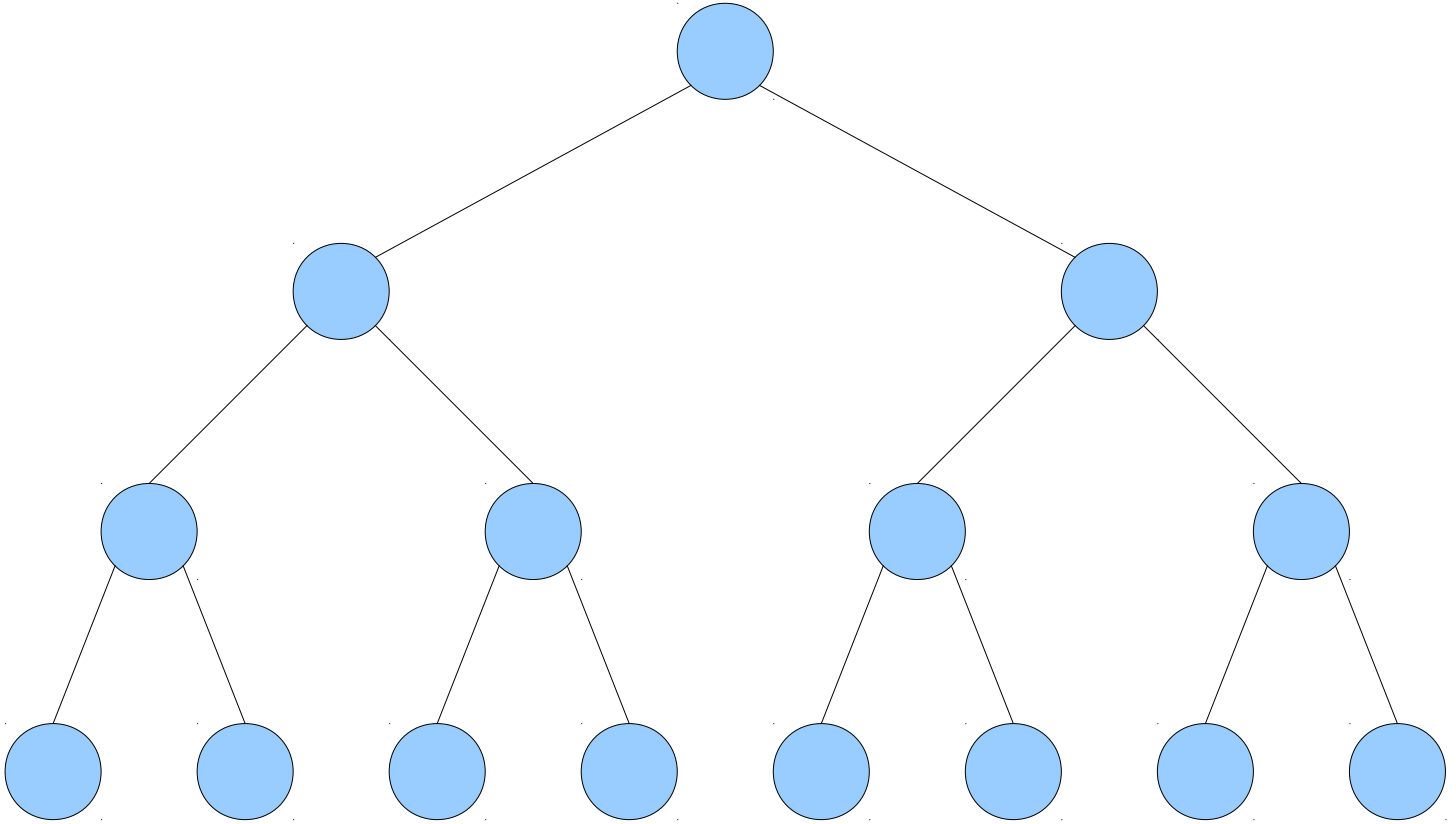
- ***Connected components are equivalence classes of the connectivity relation!***

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. ■

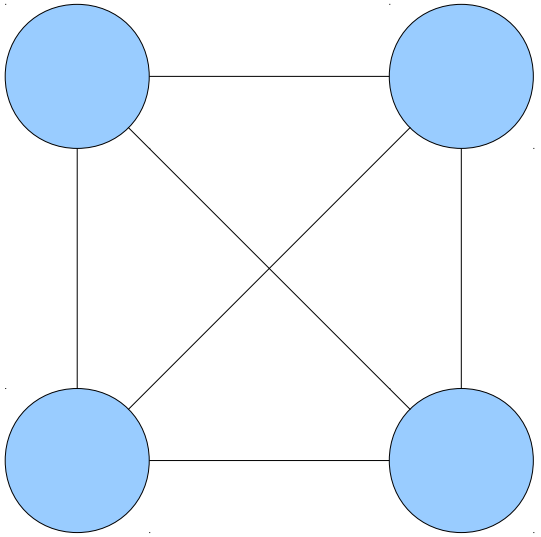
Planar Graphs





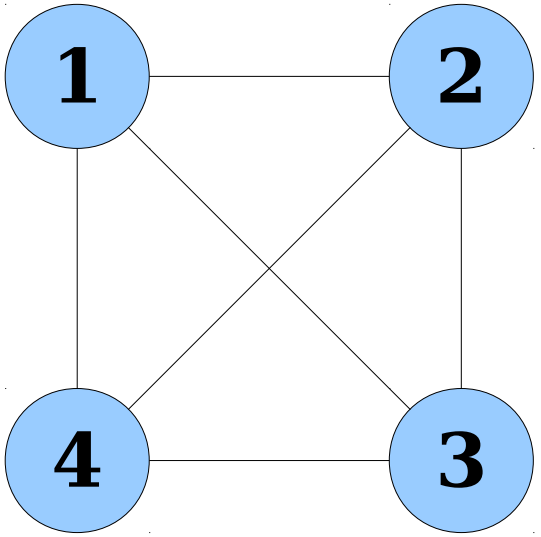
A graph is called a ***planar graph*** if there is some way to draw it in a 2D plane without any of the edges crossing.

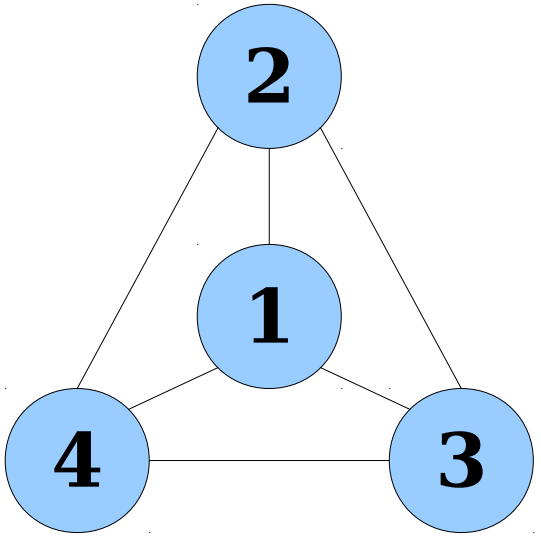
A graph is called a ***planar graph*** if there is some way to draw it in a 2D plane without any of the edges crossing.

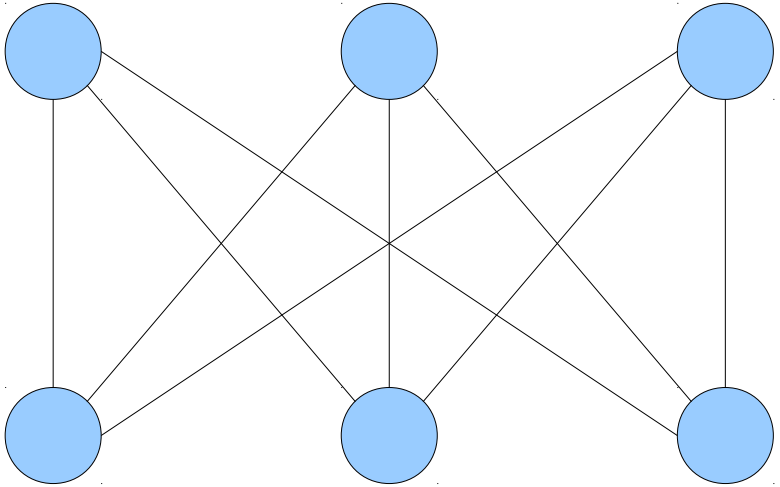


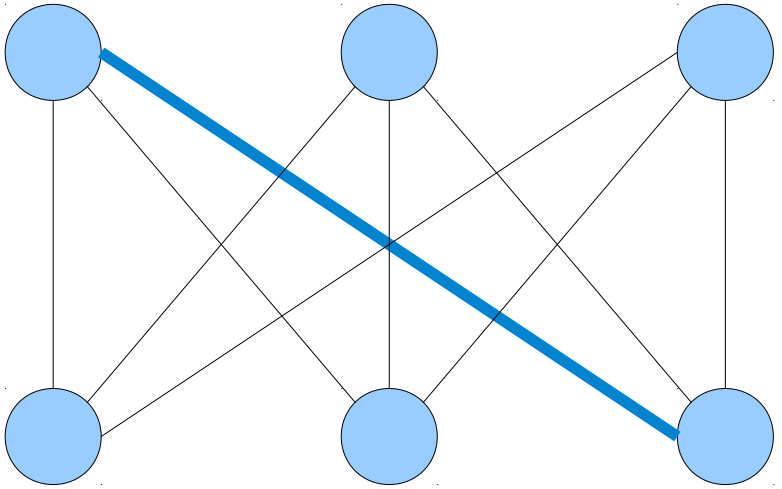
Is this graph planar?

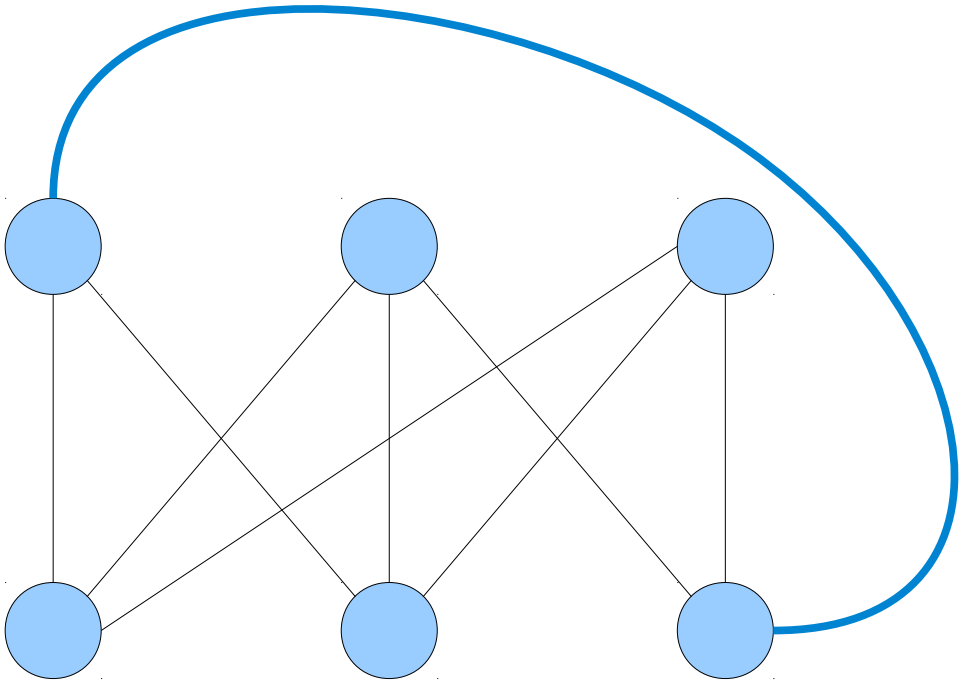
Answer at **PollEv.com/cs103** or
text **CS103** to **22333** once to join, then **Y** or **N**.

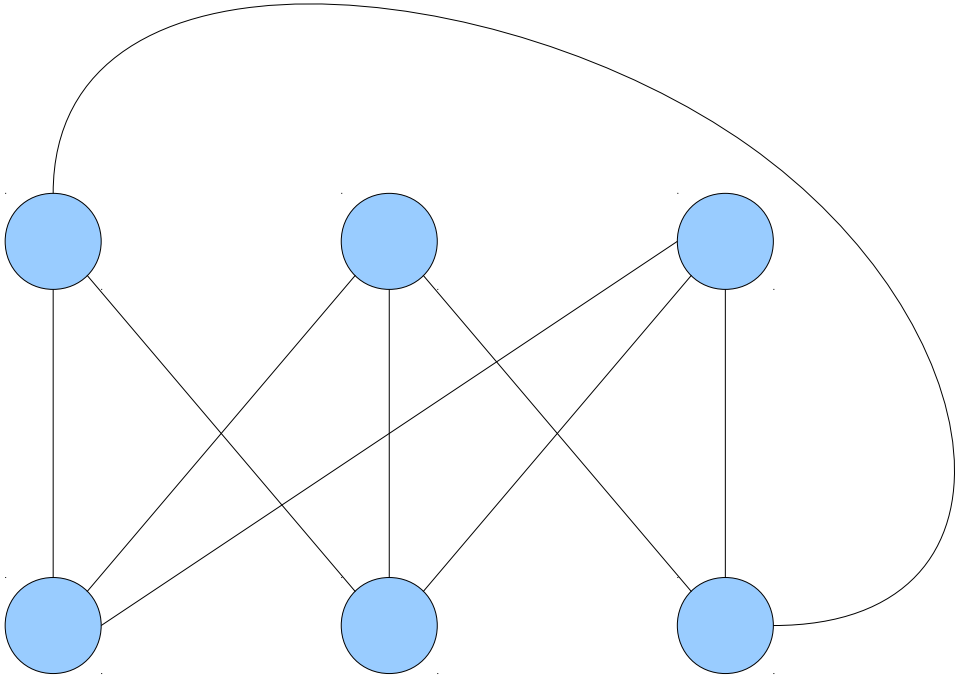

1
2
4
3

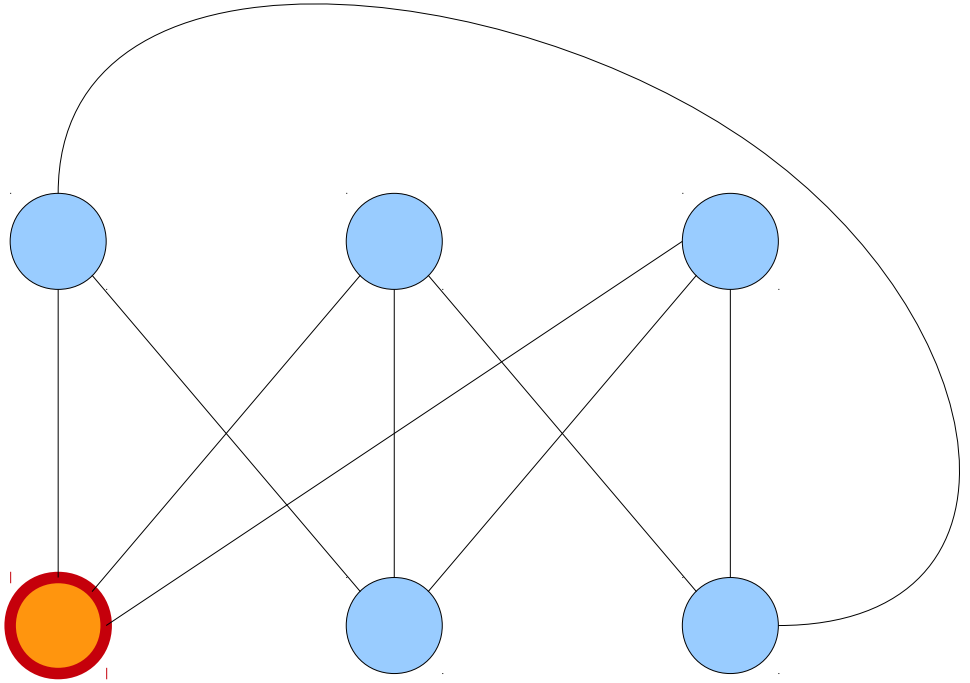

2
1
4
3

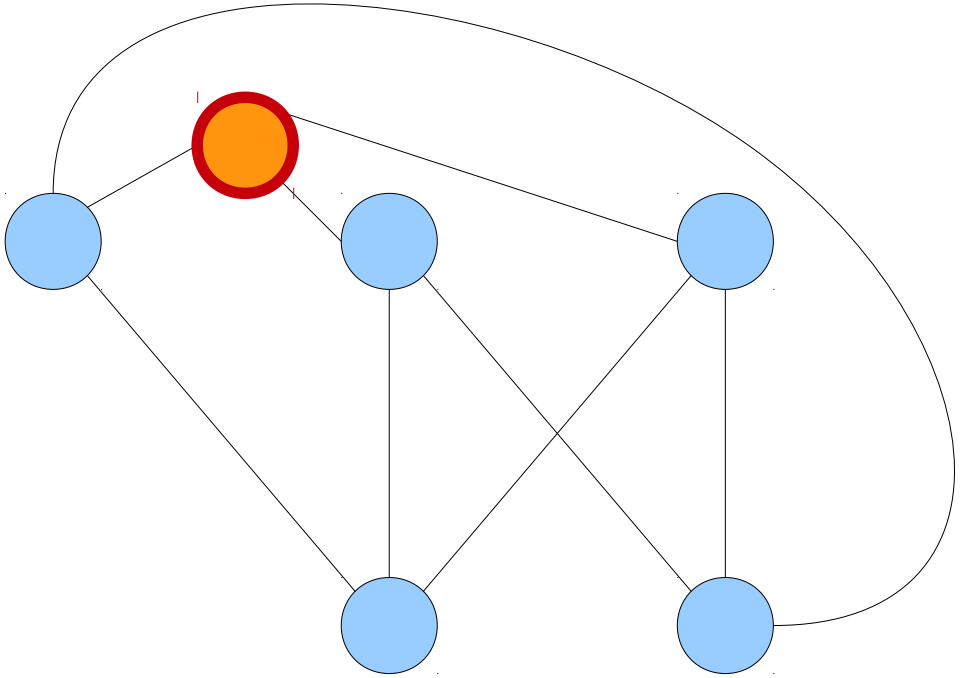


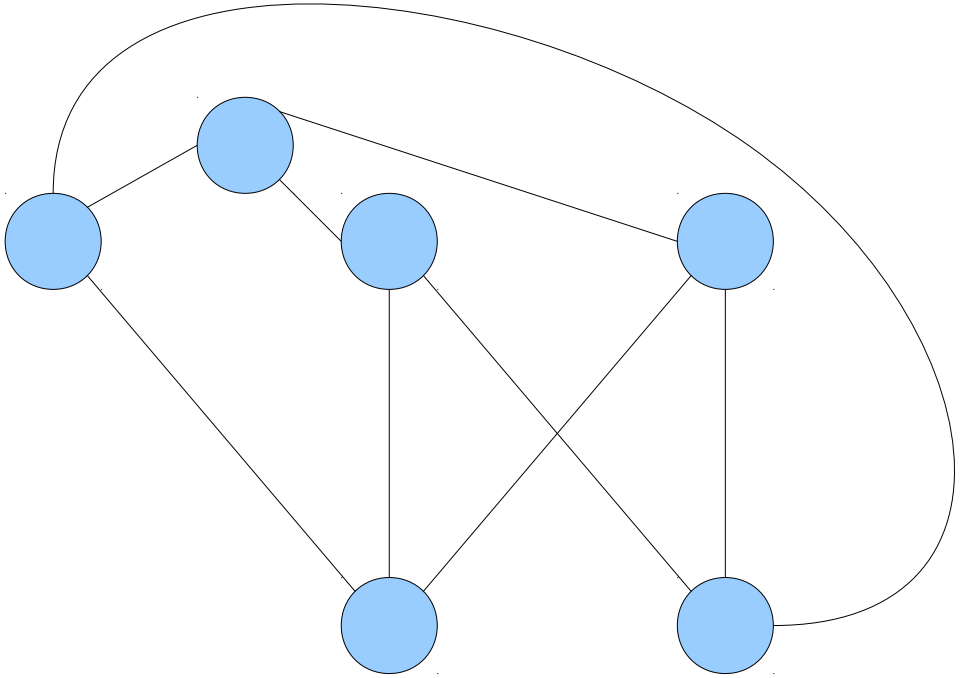


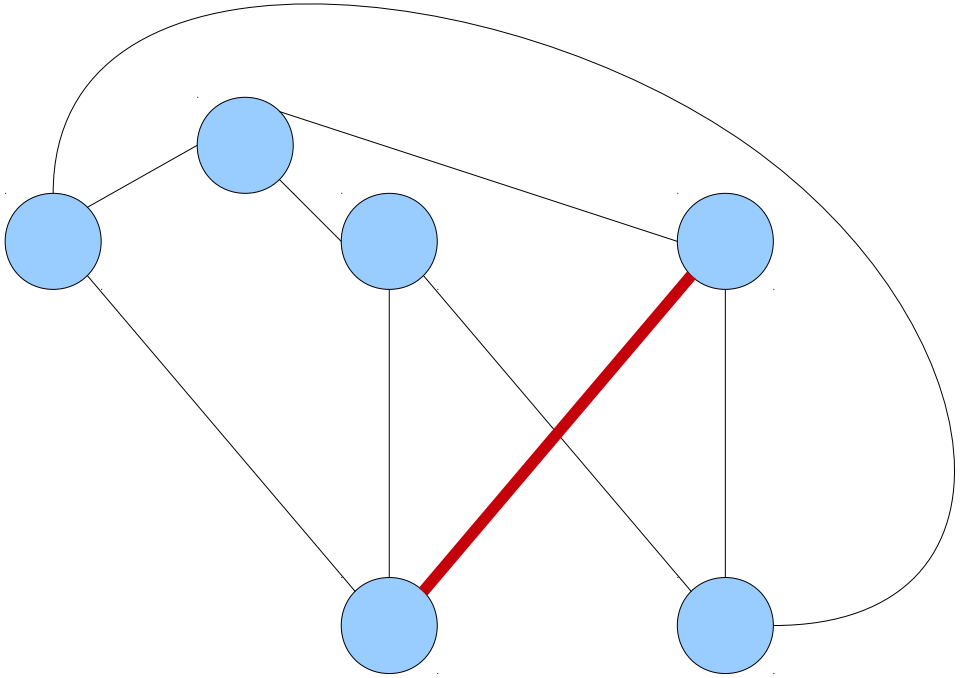


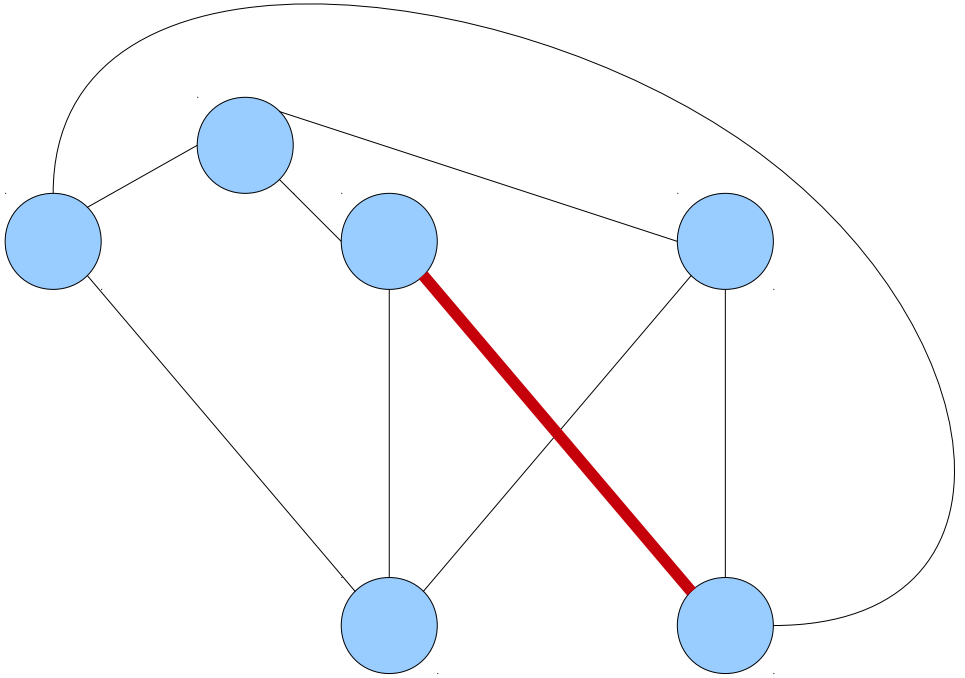


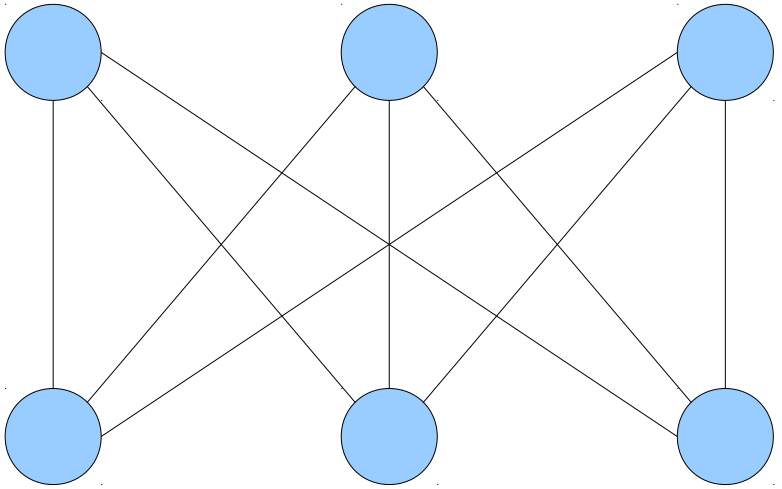






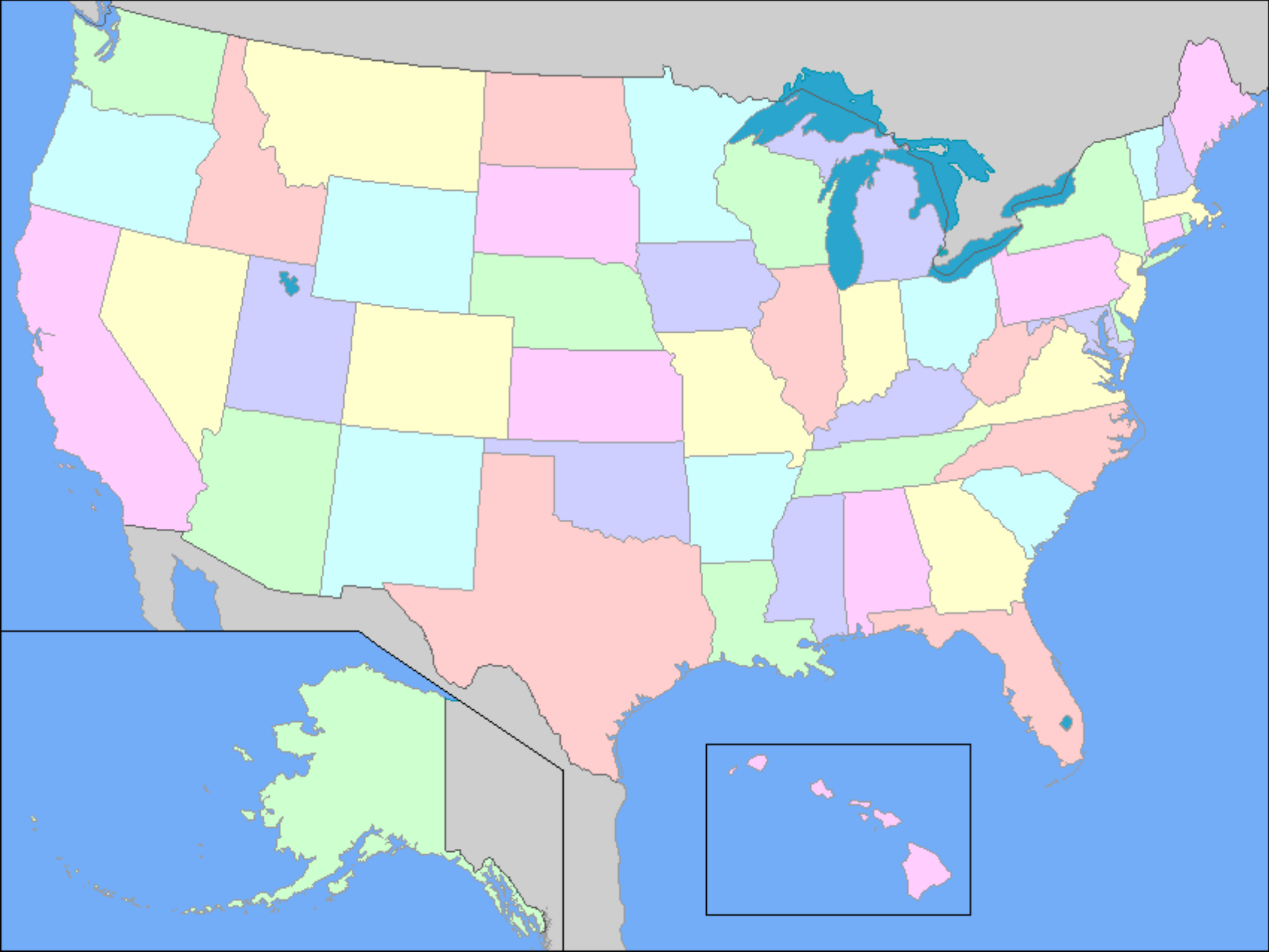


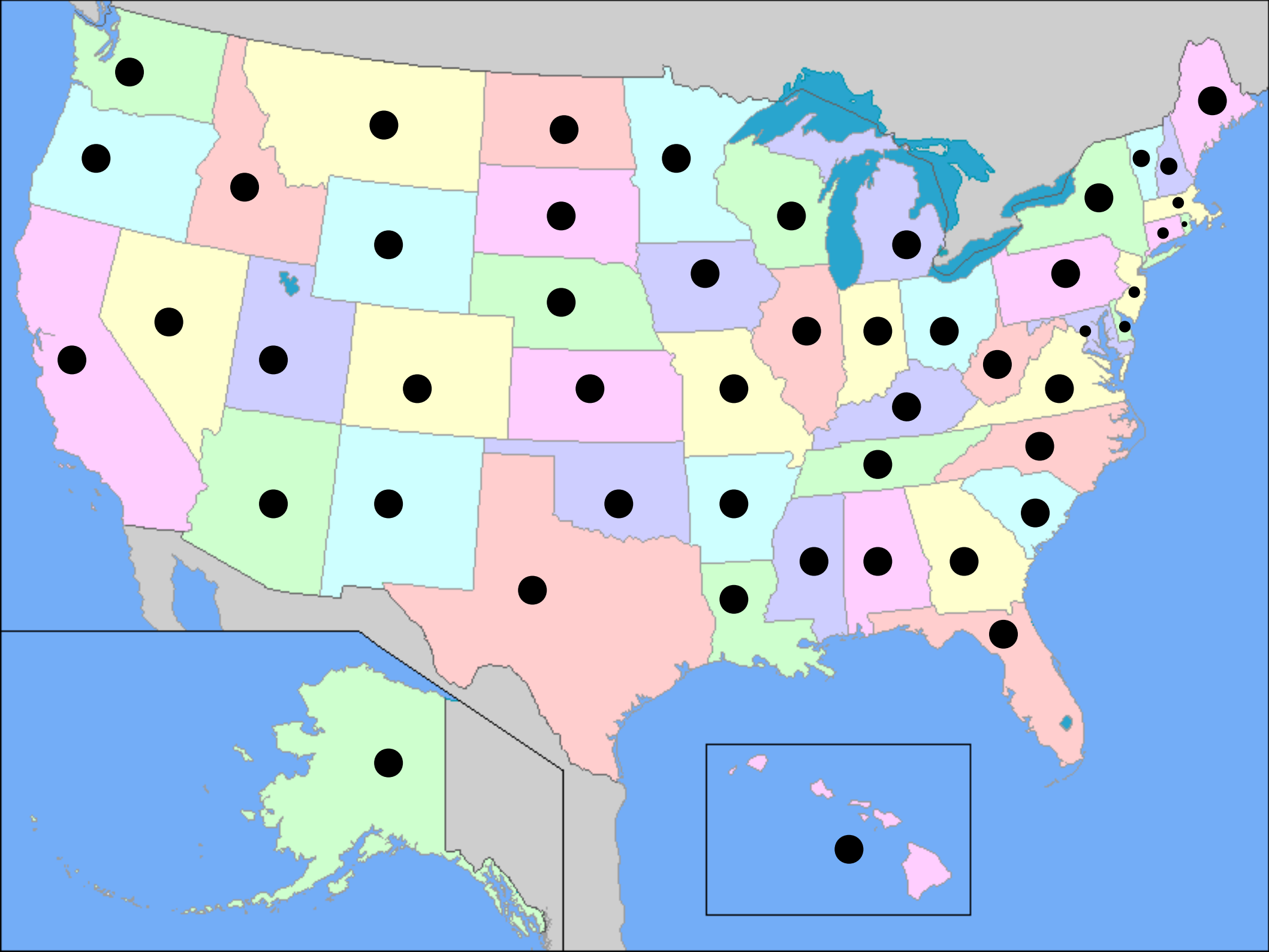


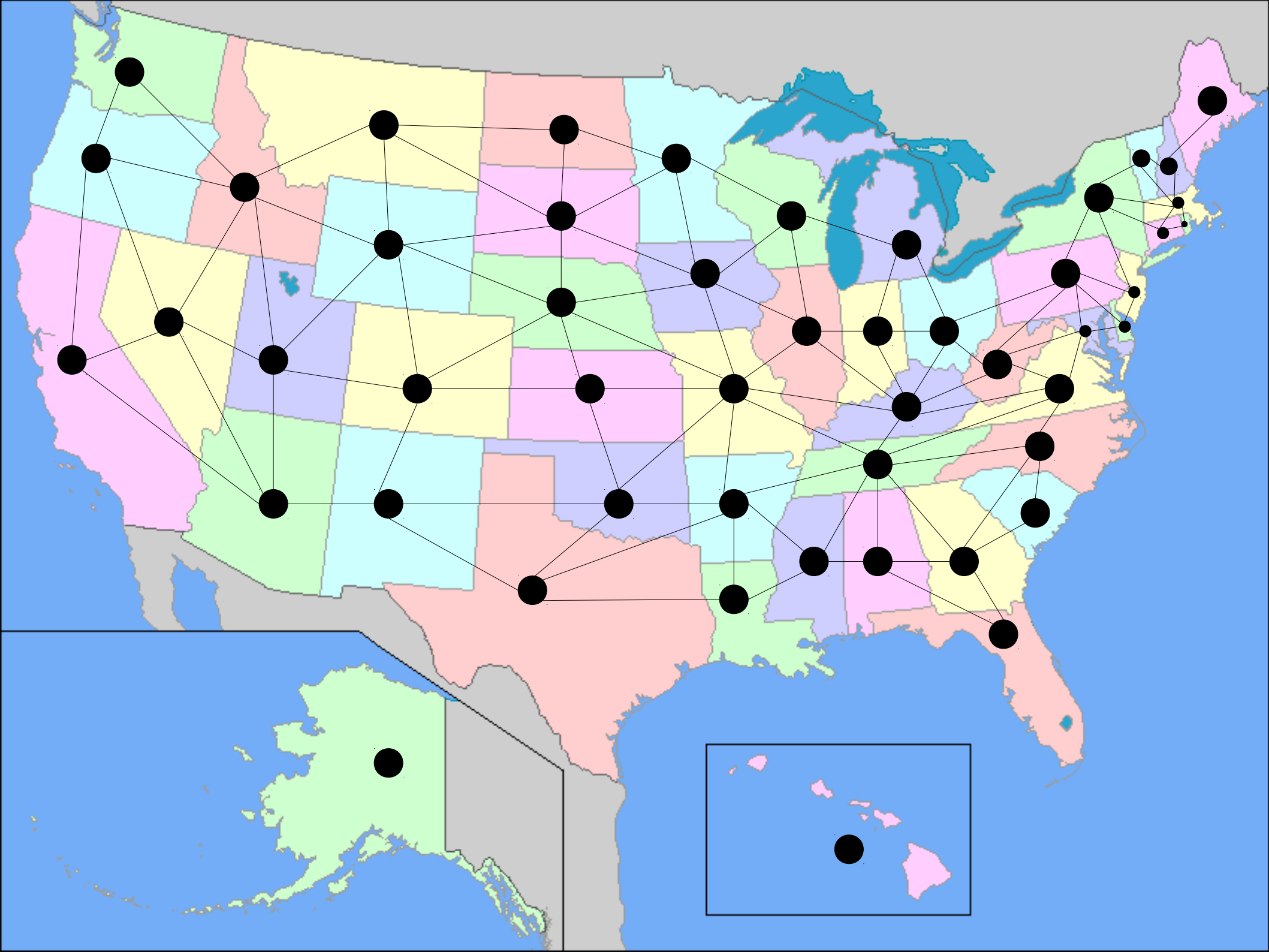


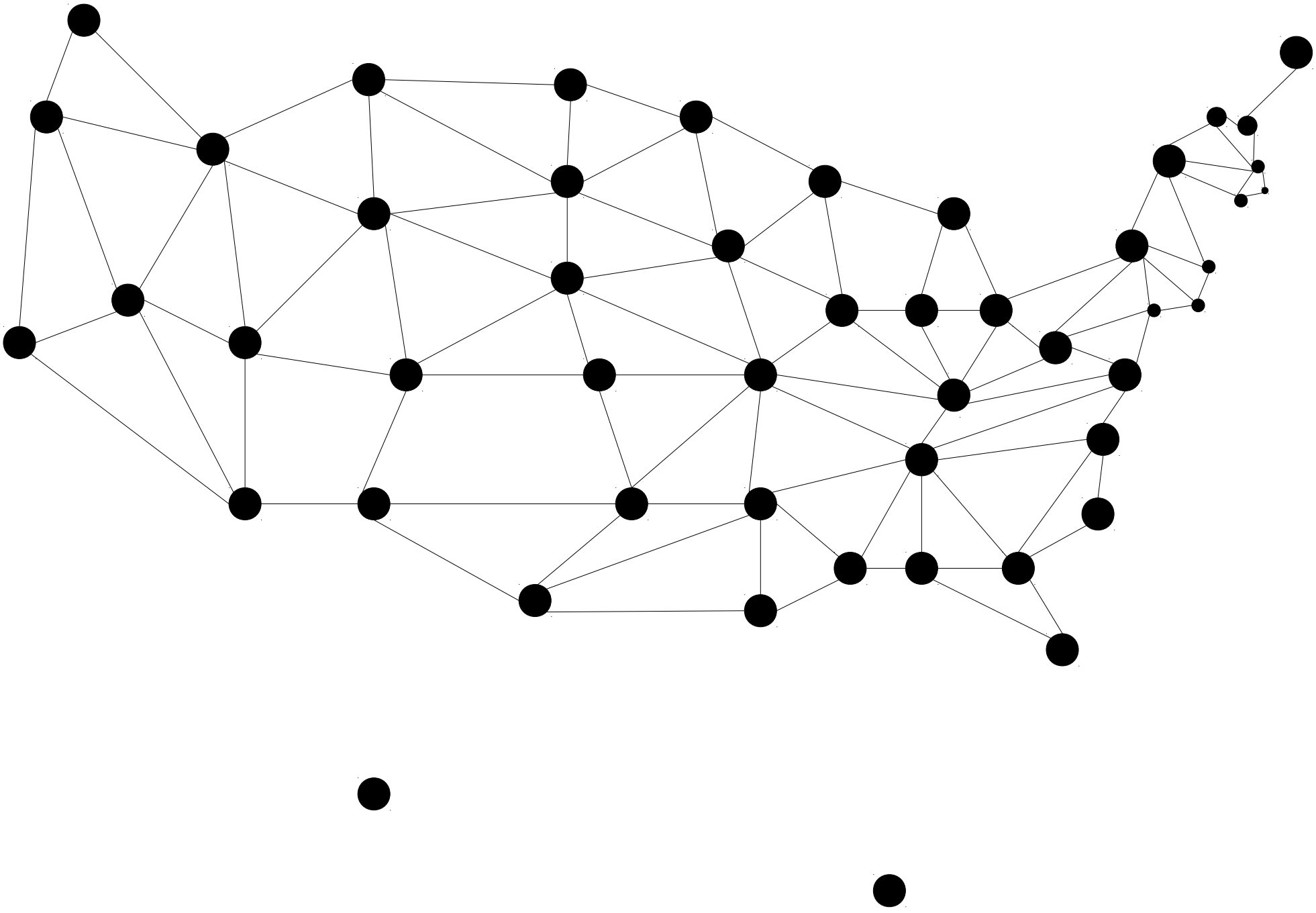
This graph is called the ***utility graph***. There is no way to draw it in the plane without edges crossing. Check out ***this video*** for an explanation!

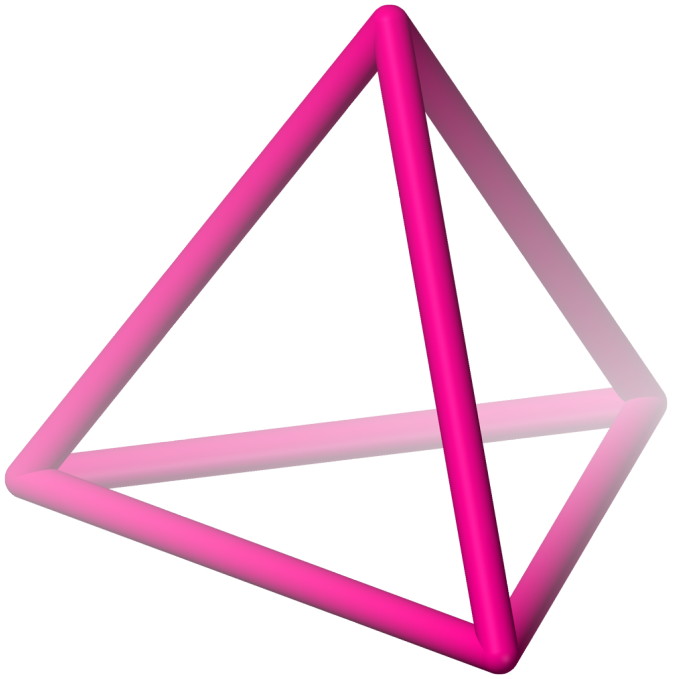
A fun game by a former CS103er:
http://www.nkhem.com/planarity-knot/

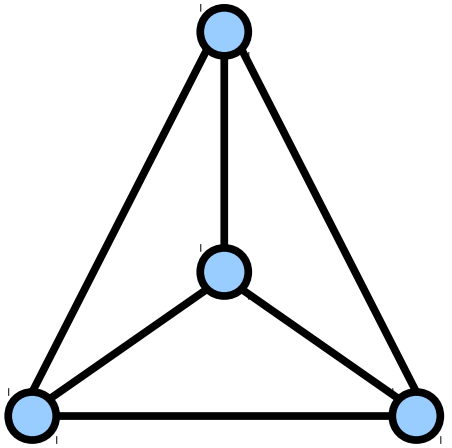


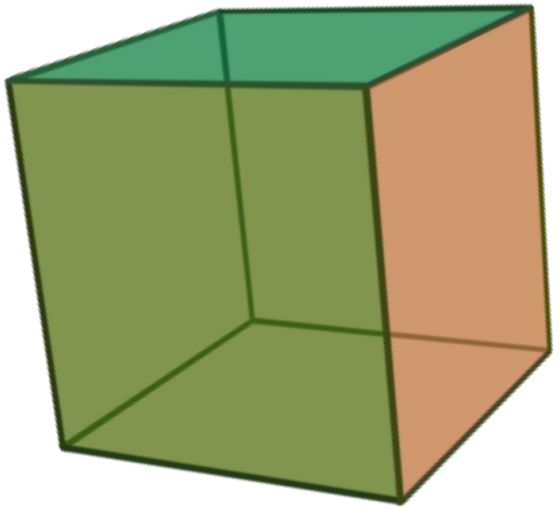


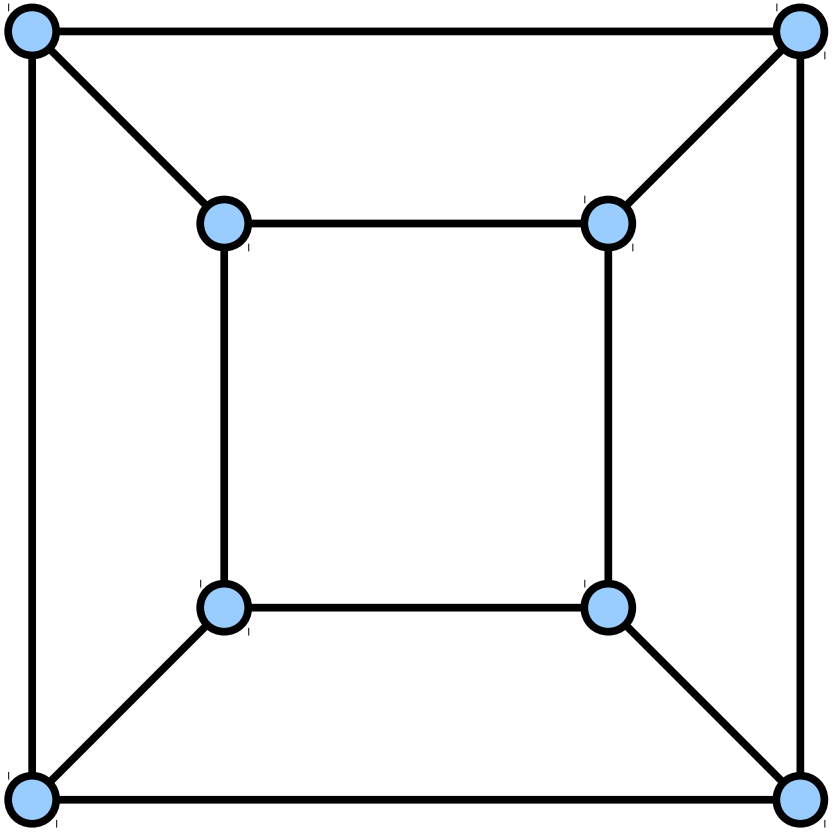


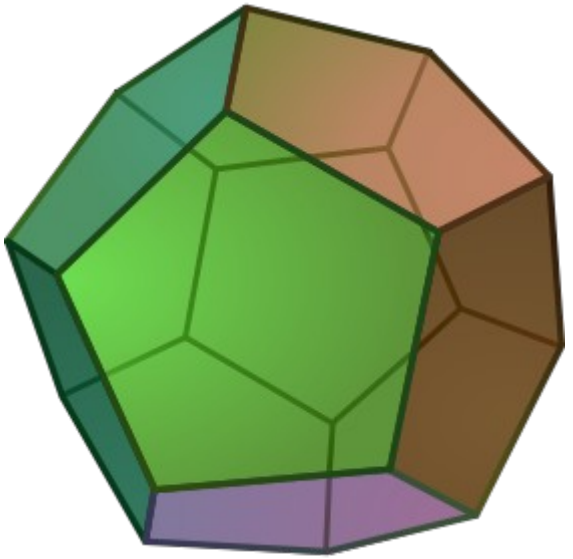


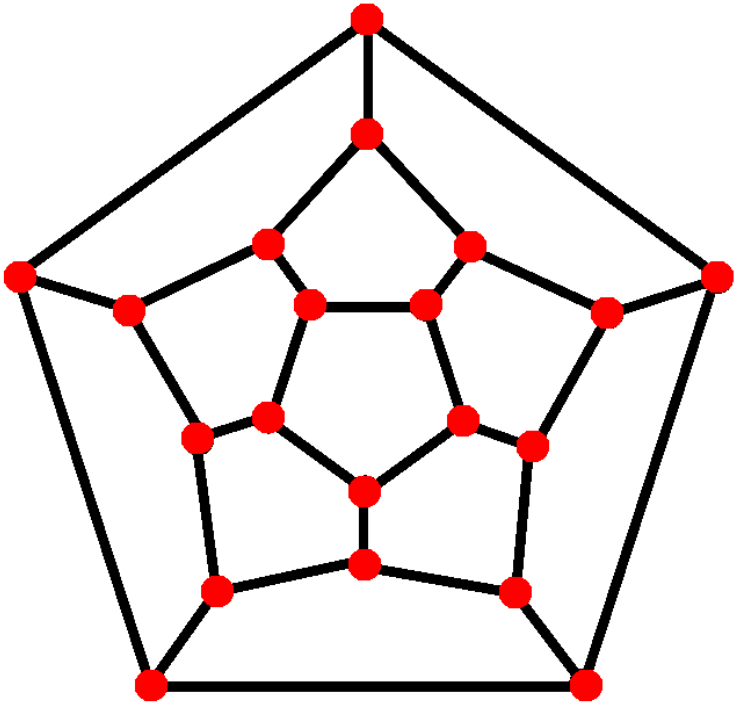




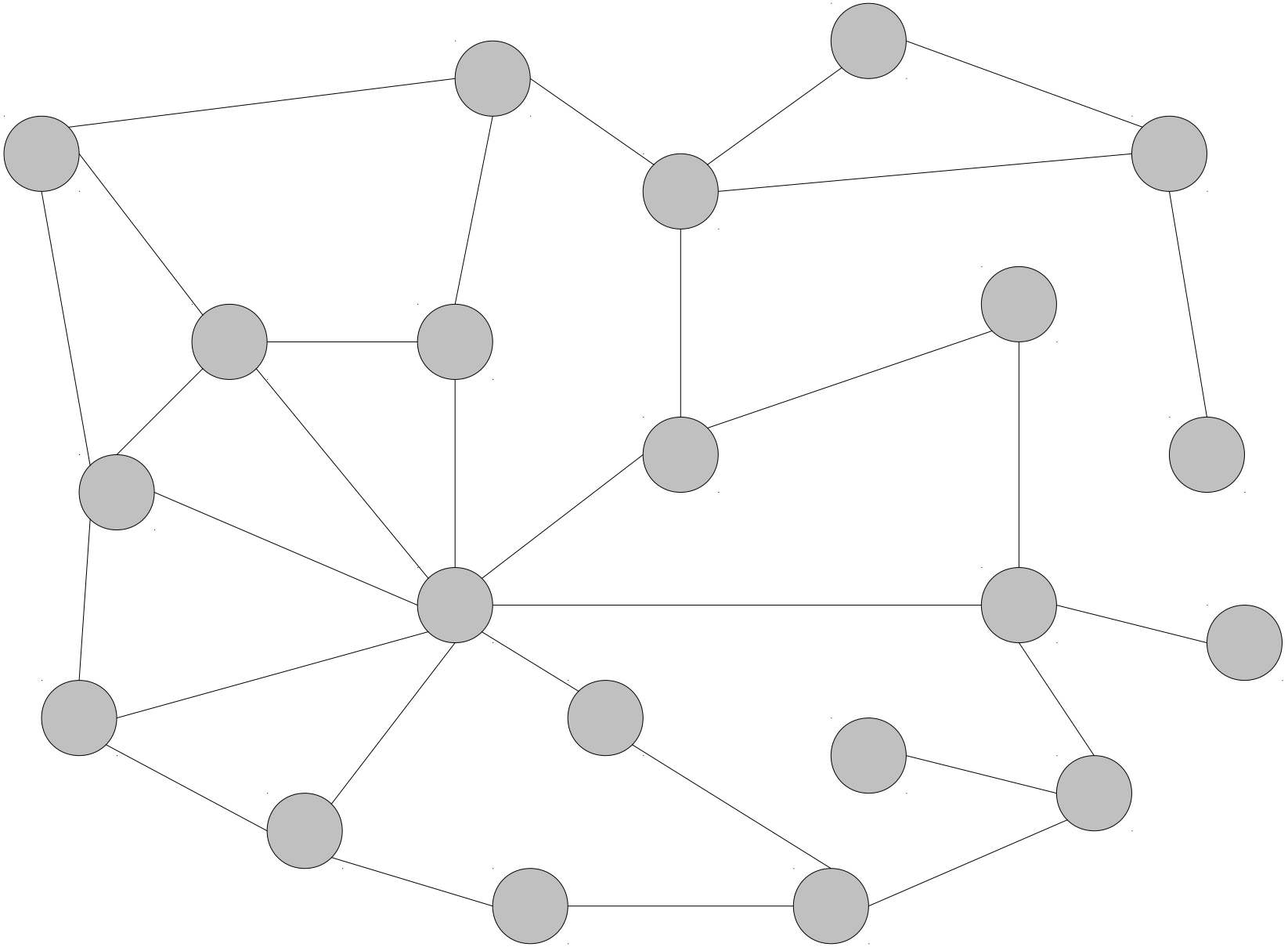




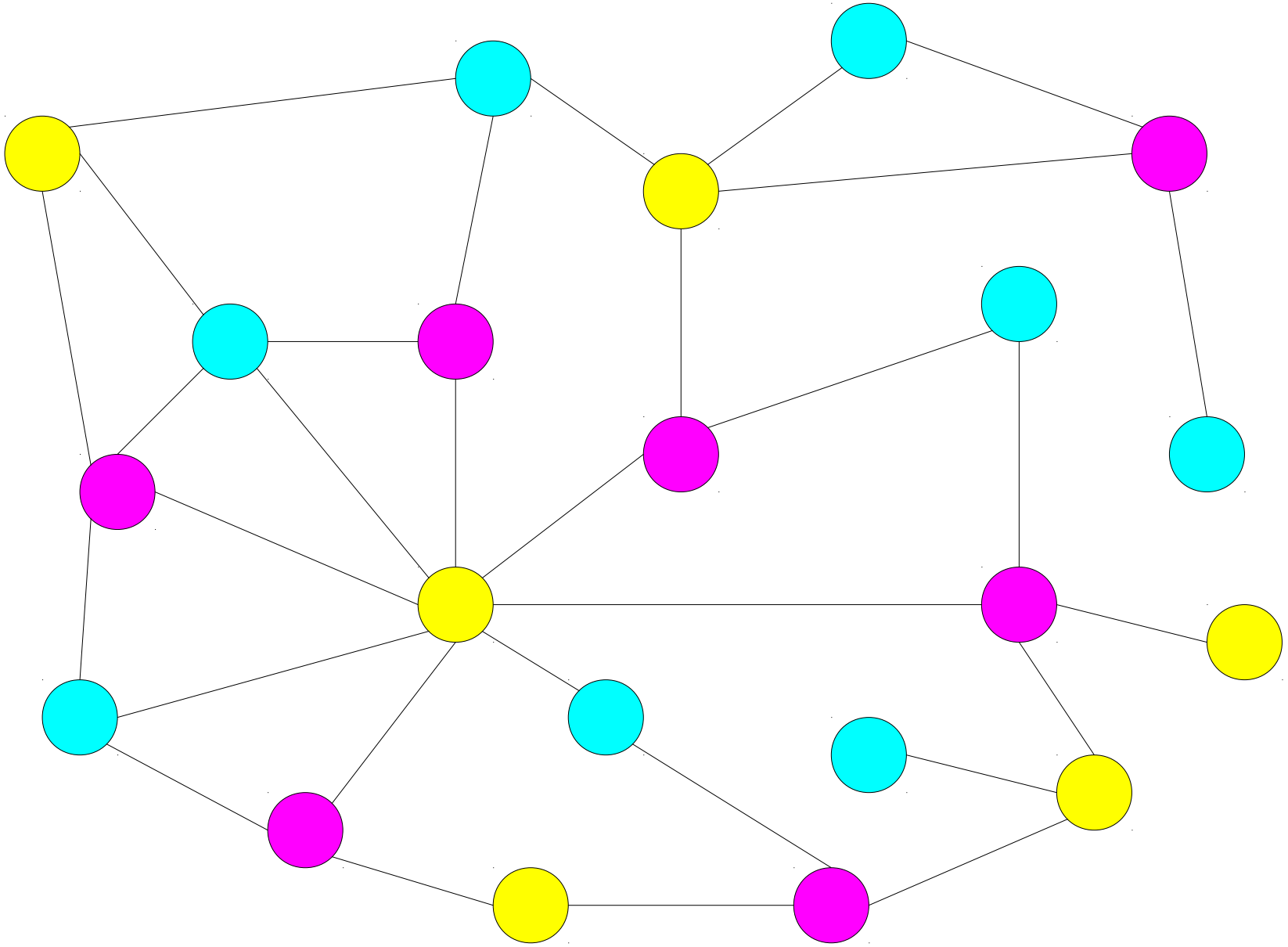


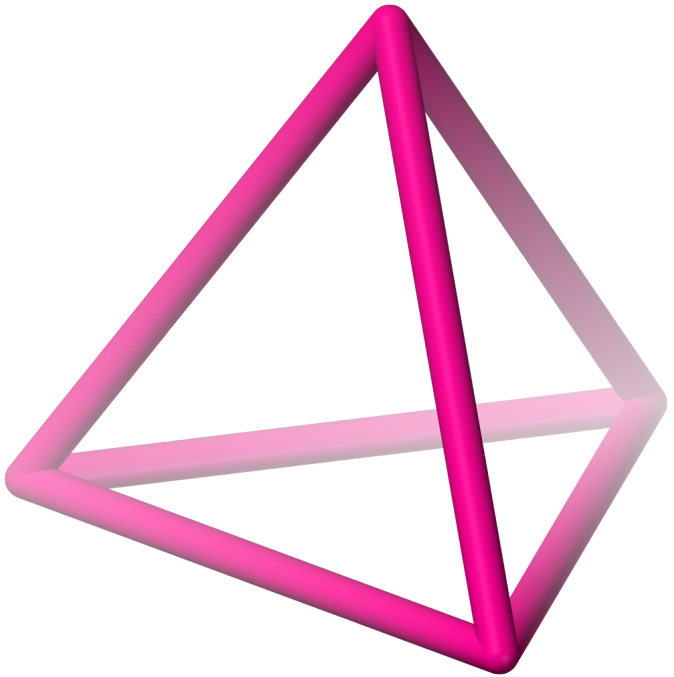


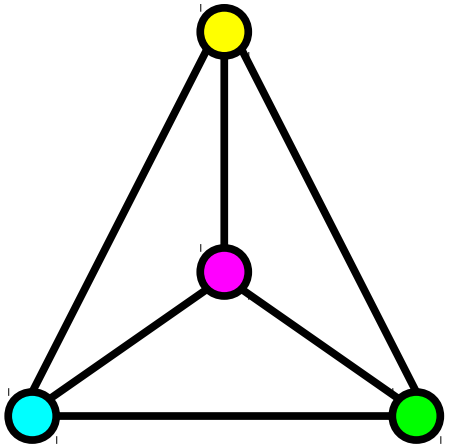
Graph Coloring

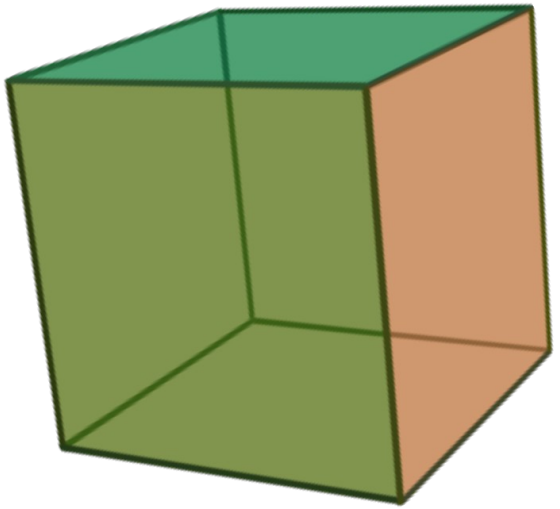


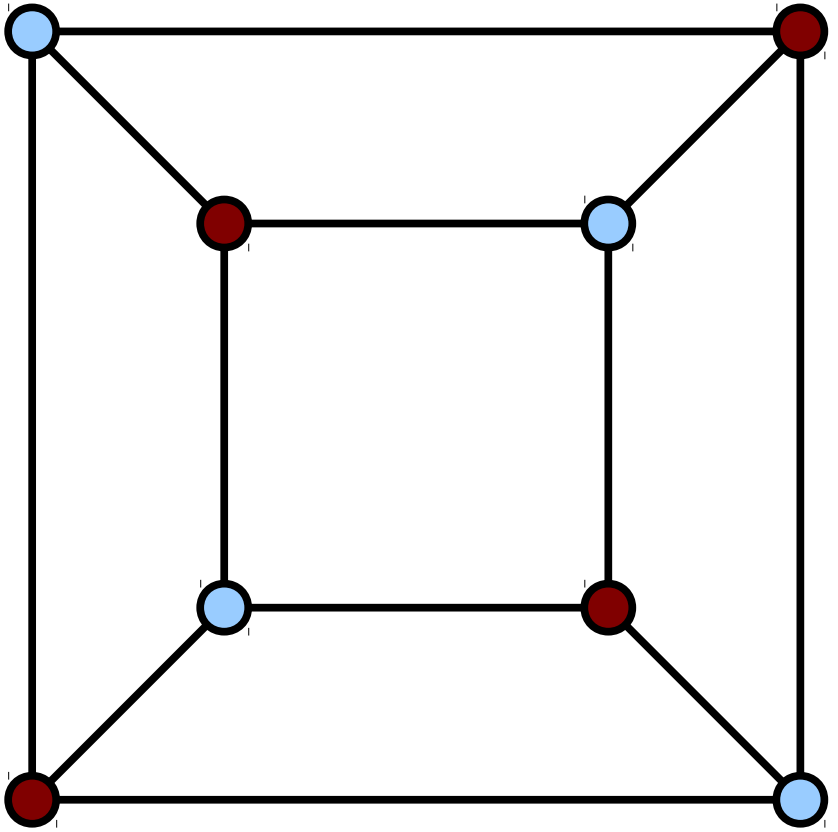
Graph Coloring











Graph Coloring

- Intuitively, a ***k-vertex-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-vertex-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.
- A ***k-vertex-coloring*** of a graph $G = (V, E)$ is a function

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

  such that

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

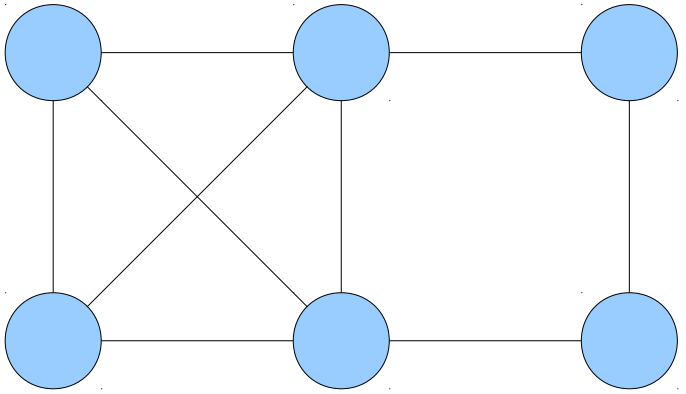
Graph Coloring

- Intuitively, a ***k-vertex-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.
- A ***k-vertex-coloring*** of a graph $G = (V, E)$ is a function

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

such that

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



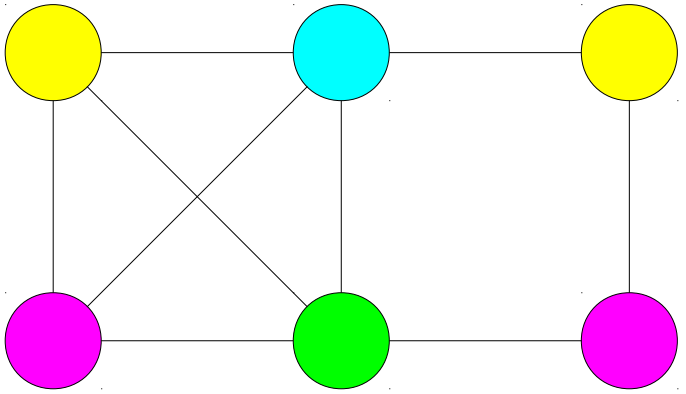
Graph Coloring

- Intuitively, a ***k-vertex-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.
- A ***k-vertex-coloring*** of a graph $G = (V, E)$ is a function

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

such that

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



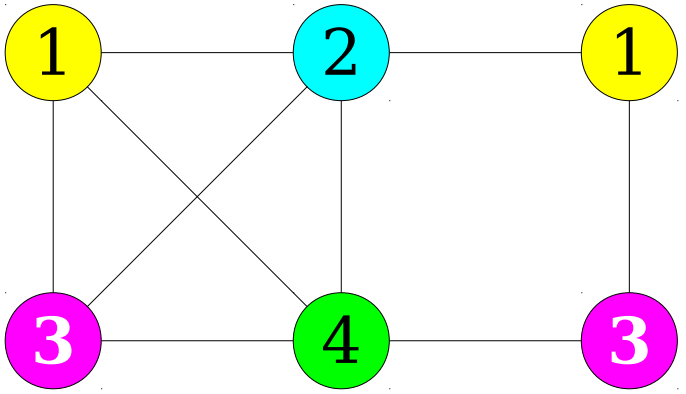
Graph Coloring

- Intuitively, a ***k-vertex-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.
- A ***k-vertex-coloring*** of a graph $G = (V, E)$ is a function

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

such that

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



Graph Coloring

- Intuitively, a ***k-vertex-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.
- A ***k-vertex-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-vertex-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-vertex-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.
- A ***k-vertex-coloring*** of a graph $G = (V, E)$ is a function

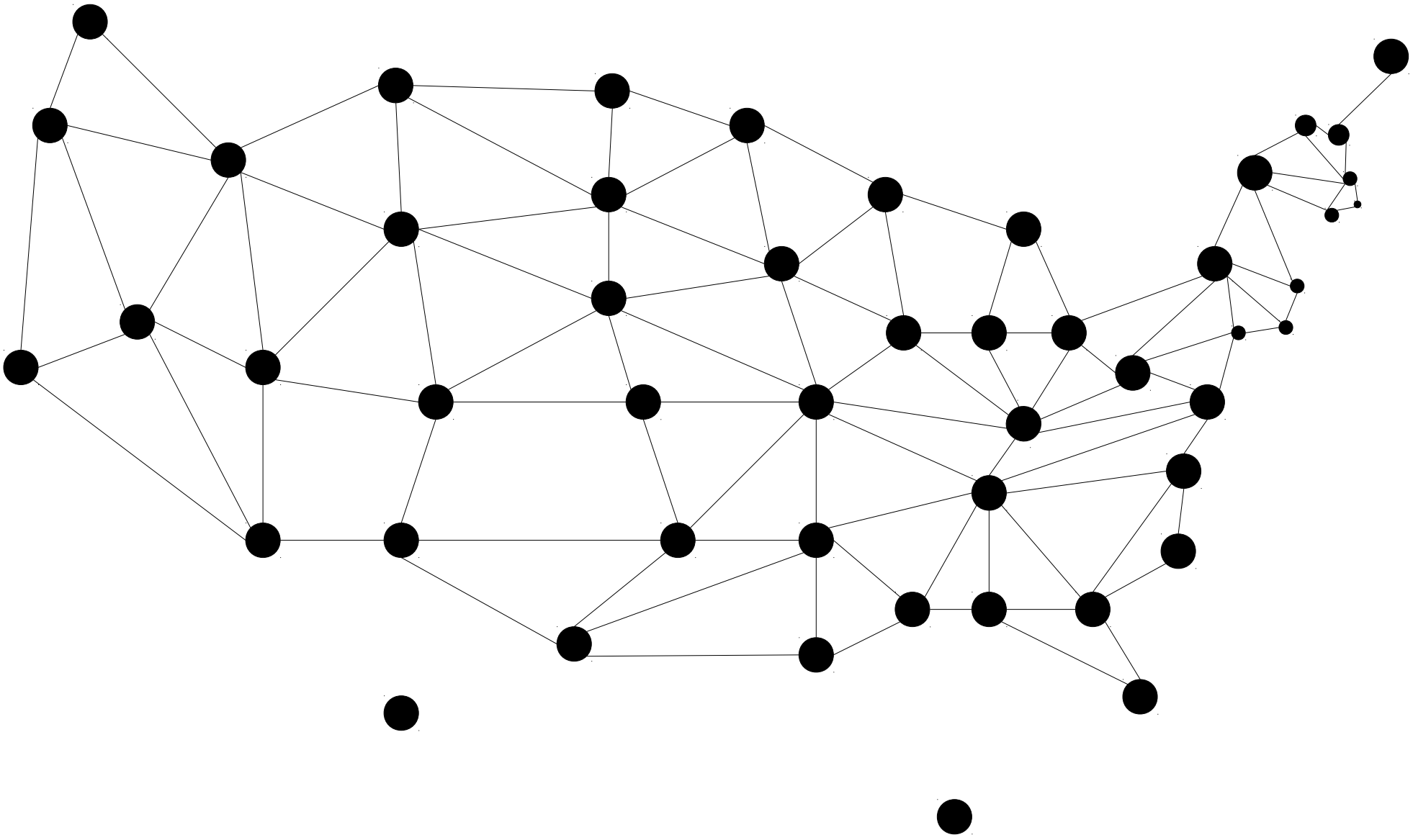
$$f : V \to \{1, 2, \ldots, k\}$$

  such that

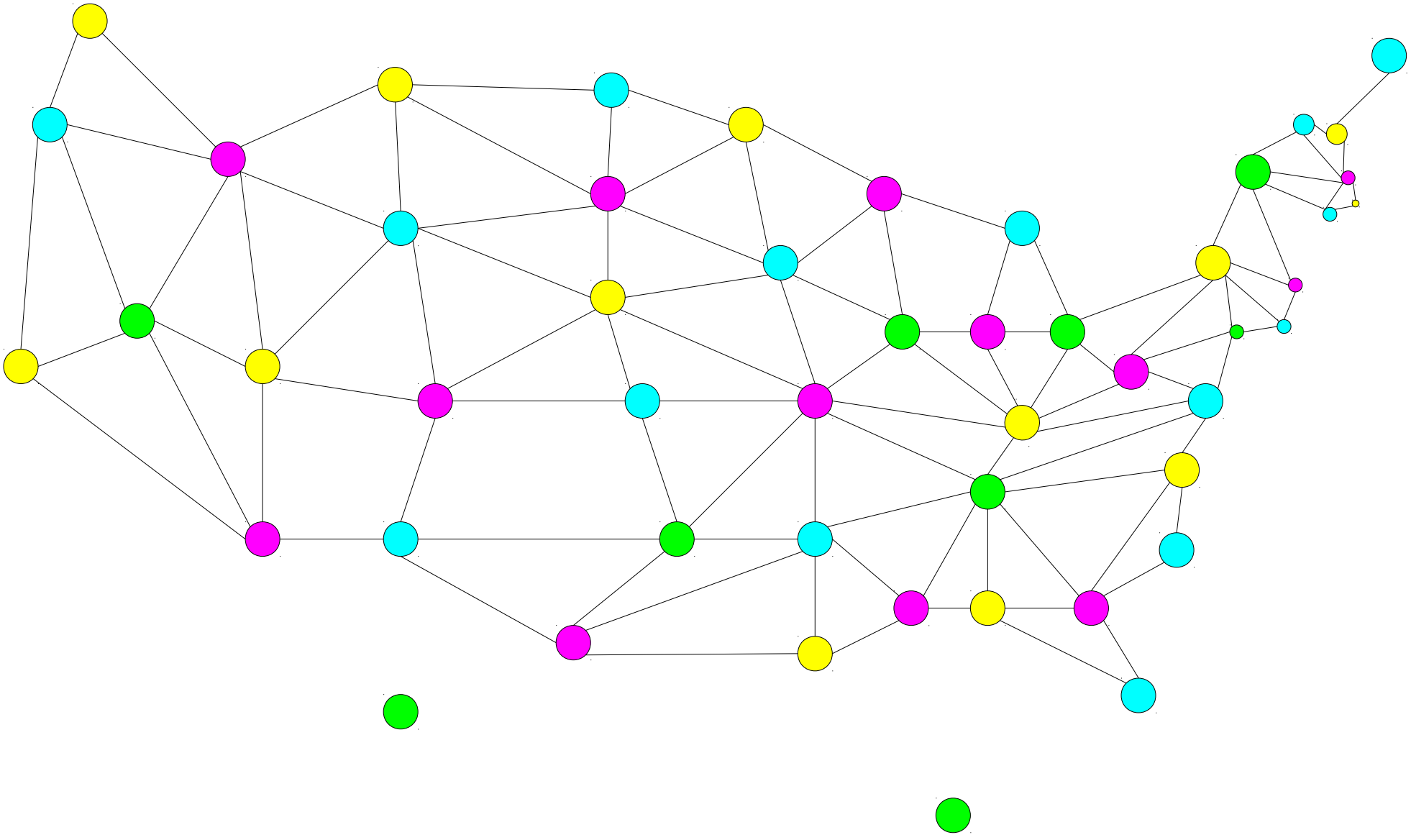
$$\forall u \in V. \forall v \in V. (\{u, v\} \in E \to f(u) \neq f(v))$$

- A graph G is ***k-colorable*** if a k-vertex 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 $\chi\rho\acute{\omega}\mu\alpha$, 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?

A Fantastic Video on a Cool Theorem:
https://youtu.be/-9OUyo8NFZg

Next Time

- ***The Pigeonhole Principle***
  - A simple, powerful, versatile theorem.
- ***Graph Theory Party Tricks***
  - Applying math to graphs of people!