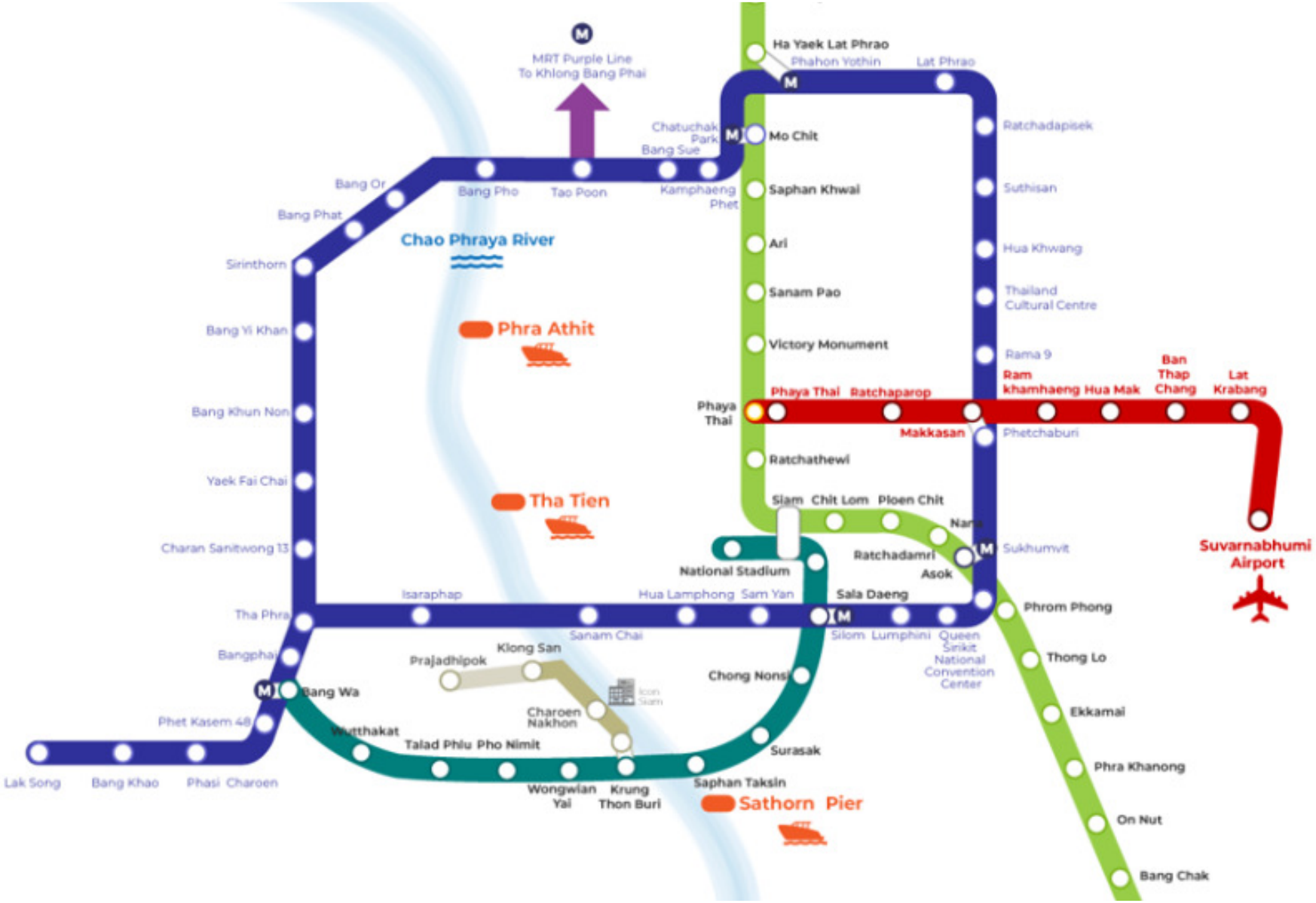
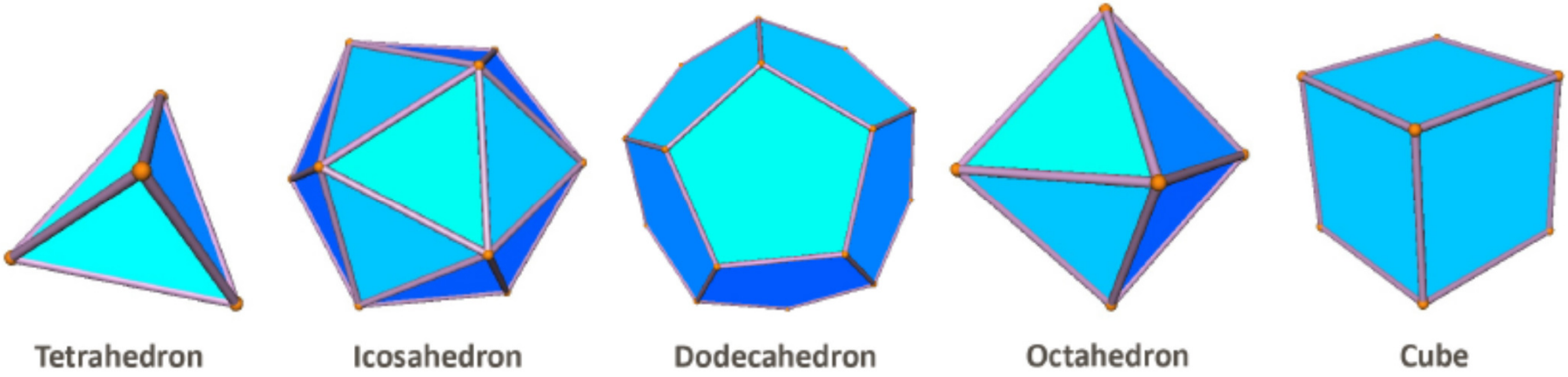
Discrete Mathematics

Graphs and Trees

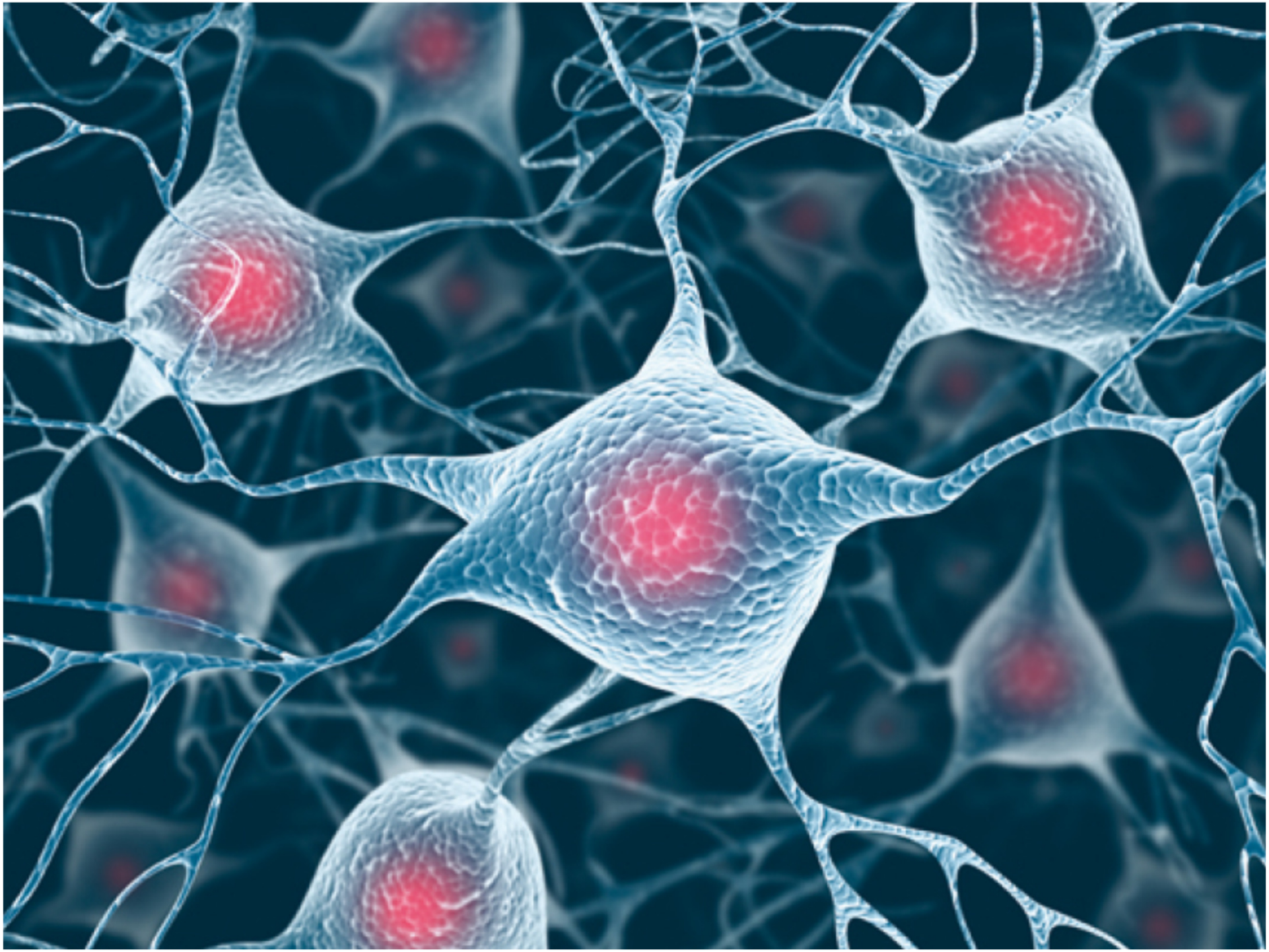
A way of modeling connections between objects.

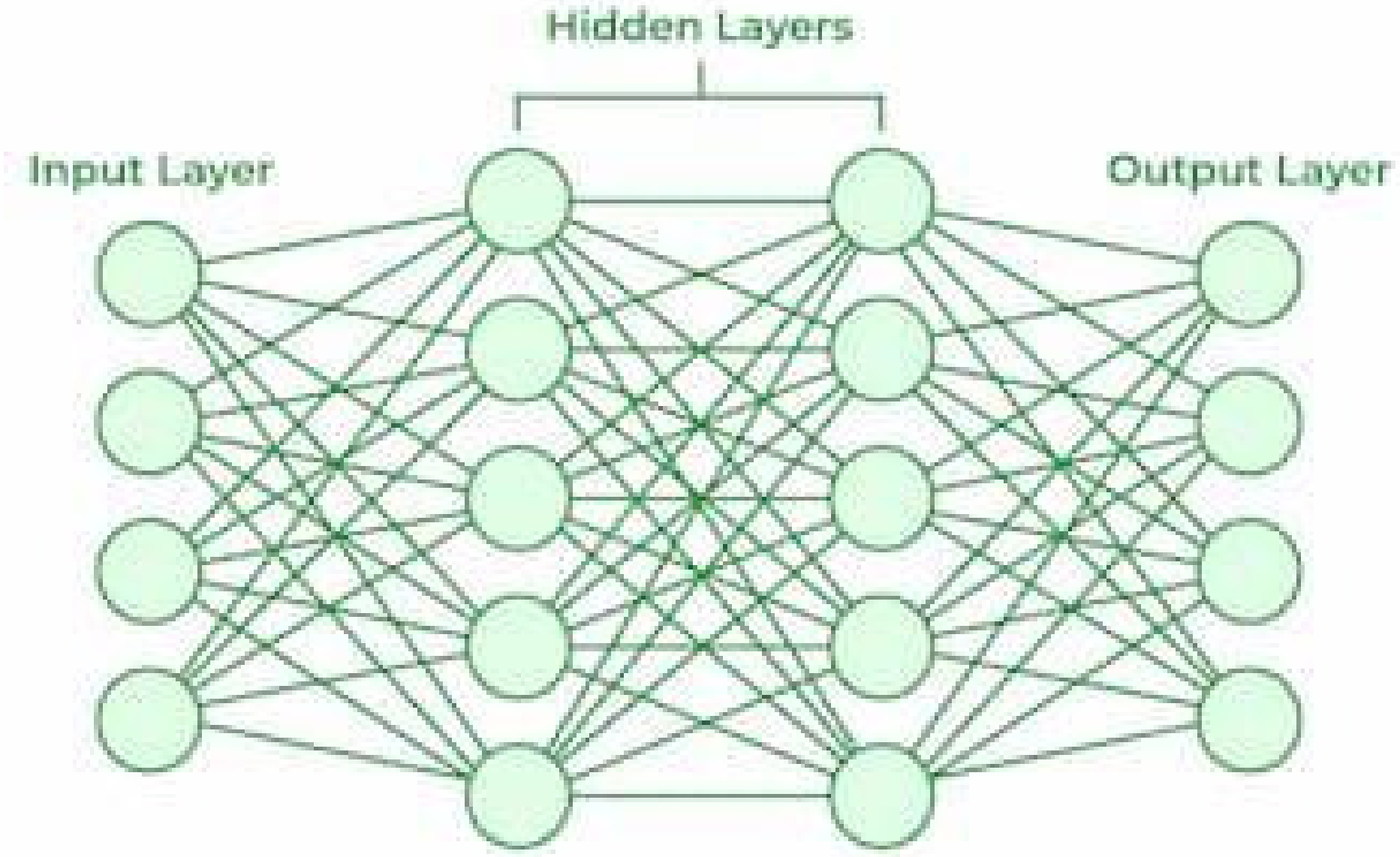
Bangkok BTS And MRT Map 2025





http://www.princeton.edu/pr/pictures/l-r/packingproblem/pu-platonic-solids.jpg



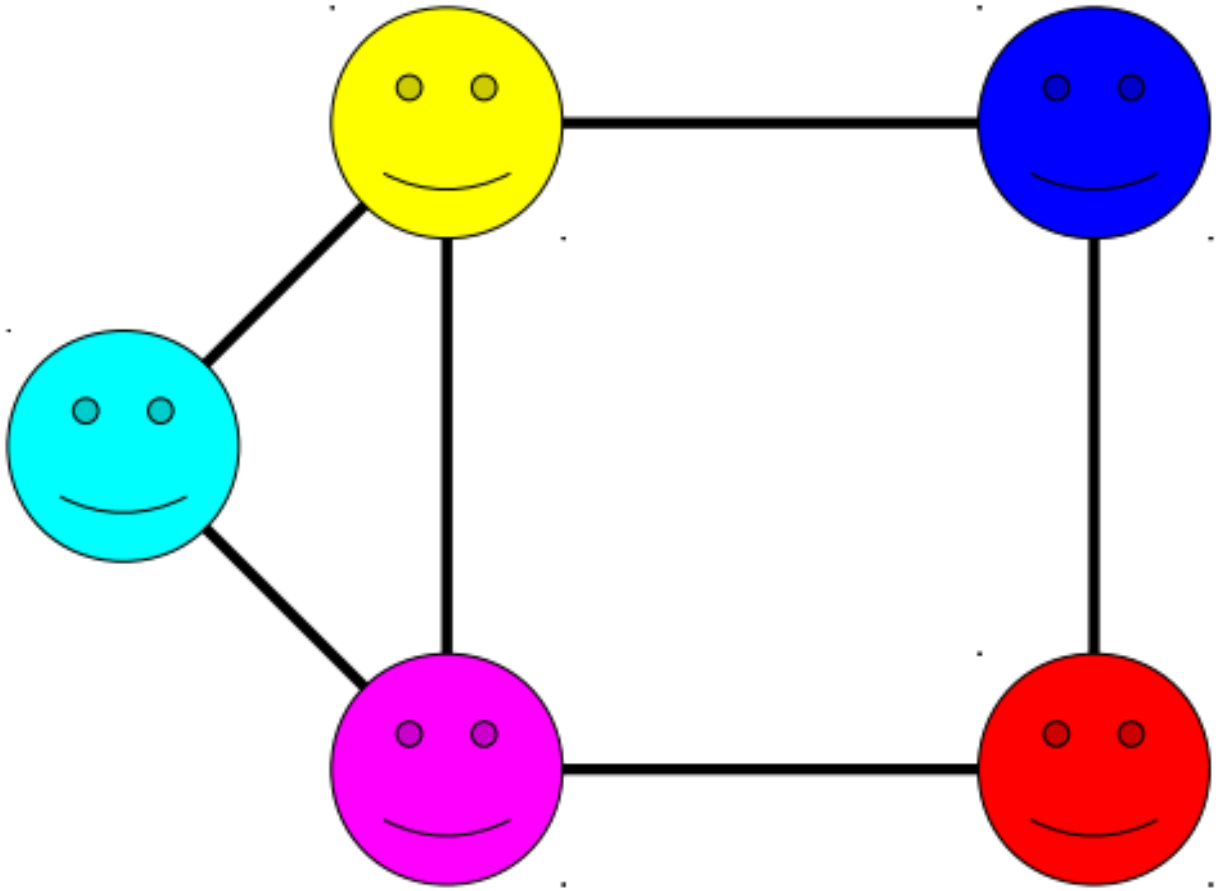

Hidden Layers
Input Layer
Output Layer

facebook®

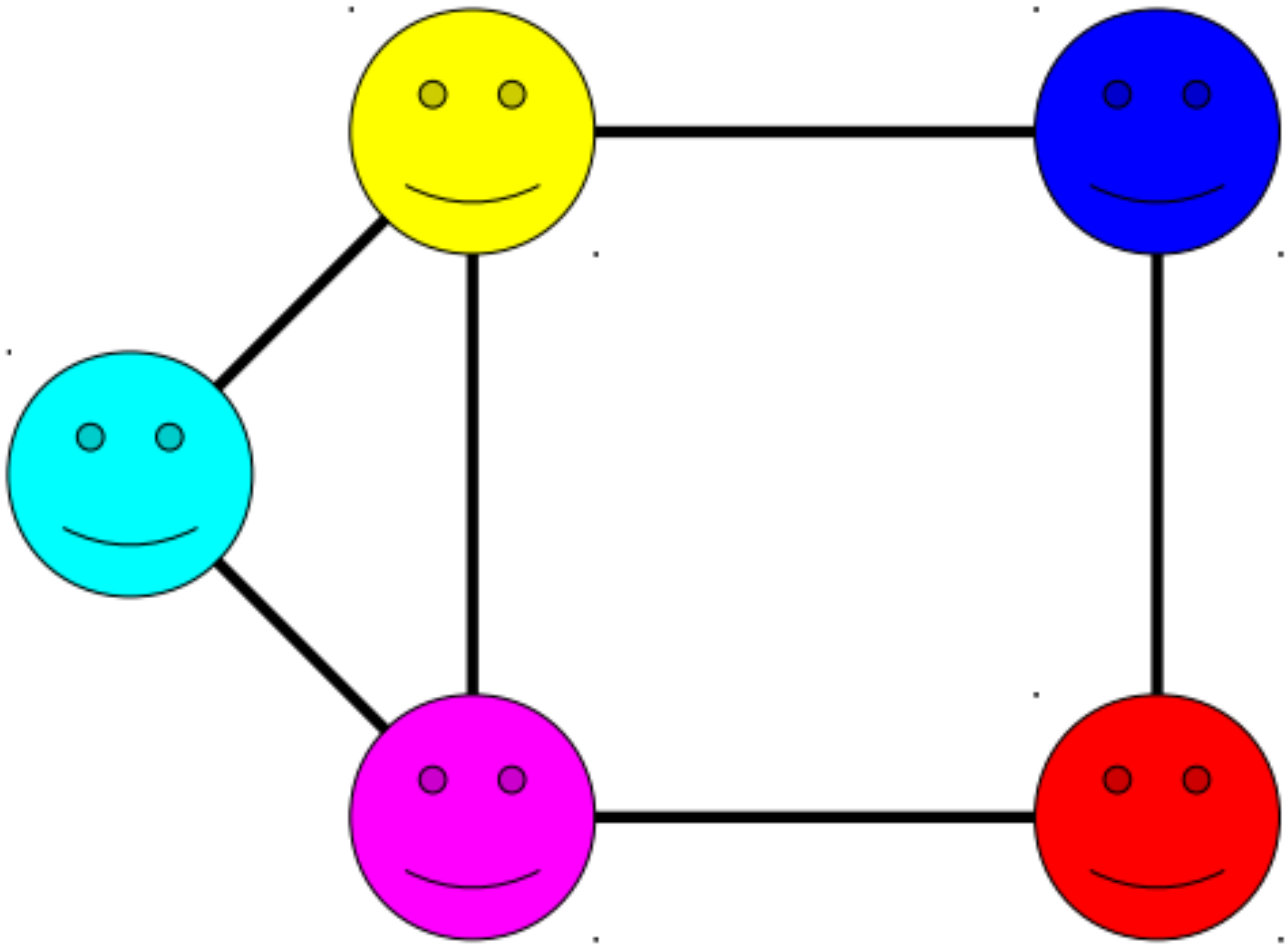
What's in Common

- 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.

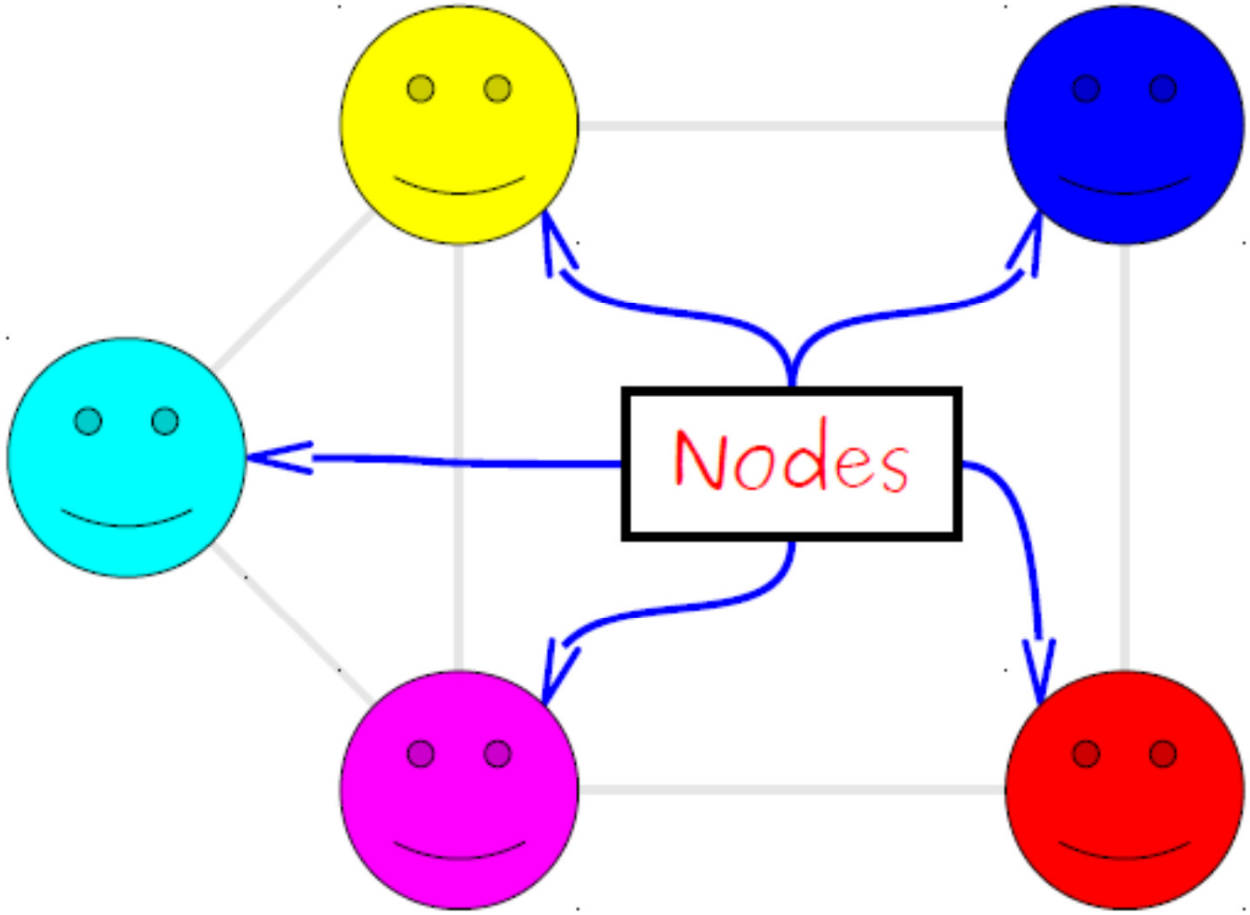


A ***graph*** is a mathematical structure for representing relationships.



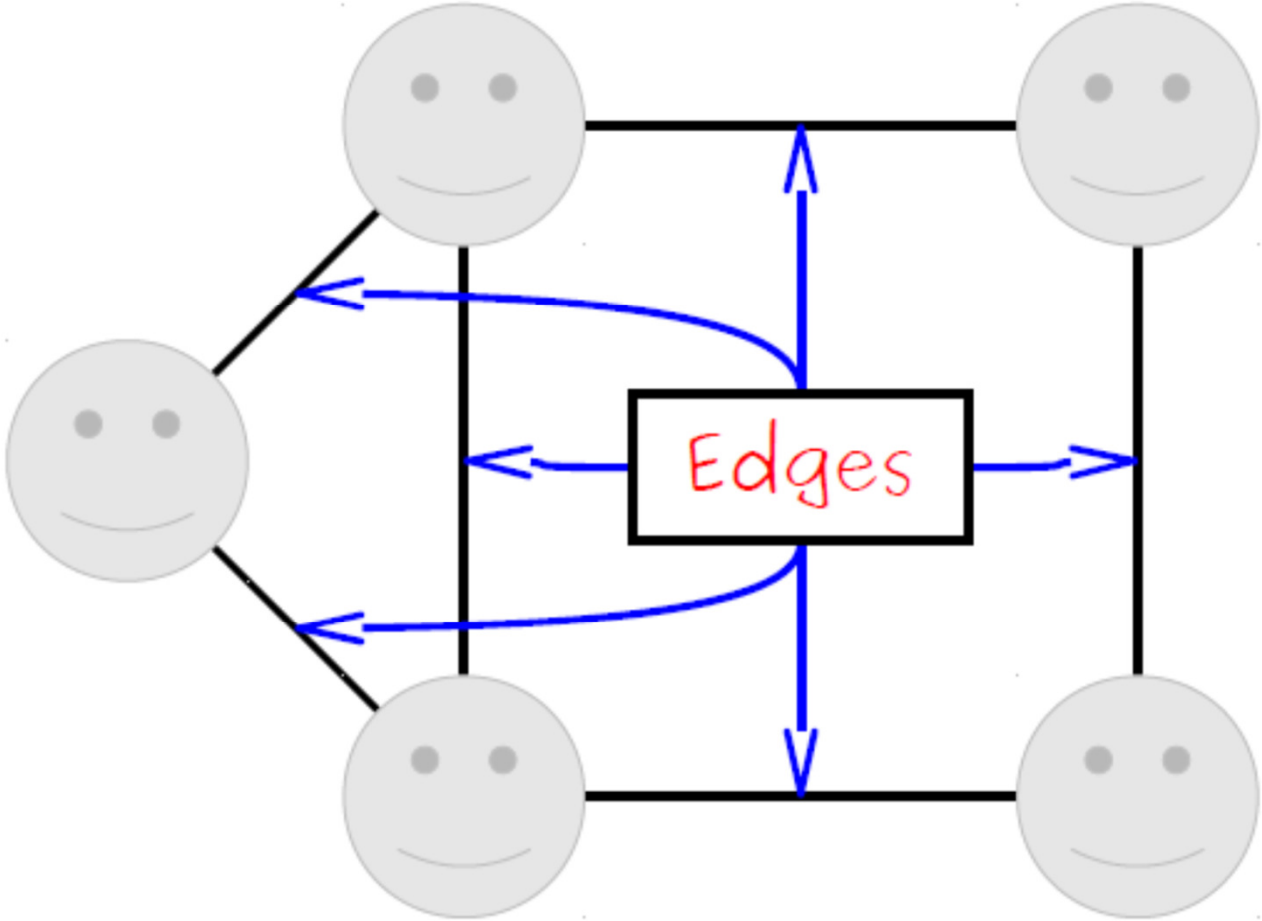
A graph consists of a set of ***nodes*** (or ***vertices***) connected by ***edges*** (or ***arcs***)

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



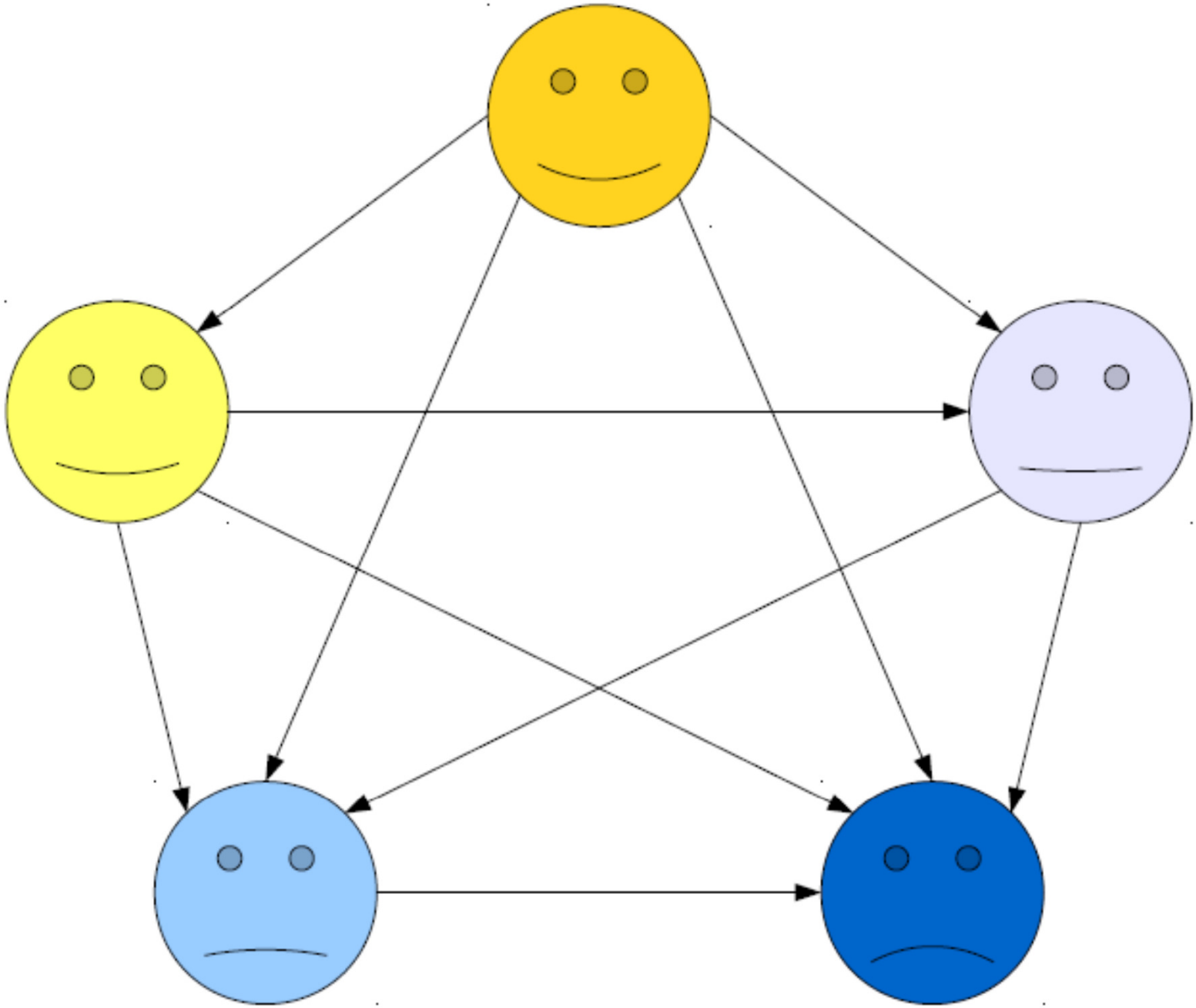
A graph consists of a set of ***nodes*** (or ***vertices***) connected by ***edges*** (or ***arcs***)

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

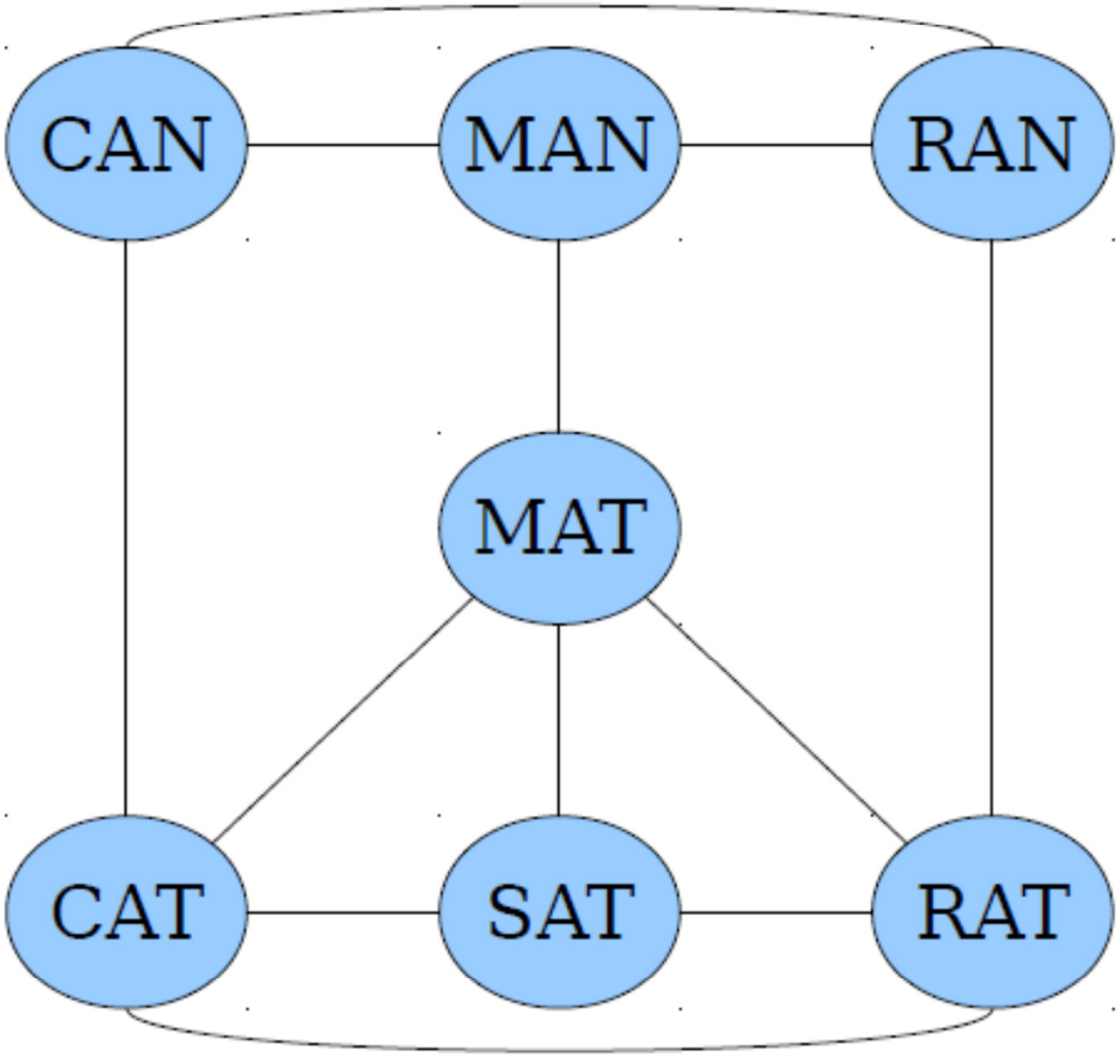


A graph consists of a set of ***nodes*** (or ***vertices***) connected by ***edges*** (or ***arcs***)

Some graphs are **directed**.



Some graphs are **undirected**.

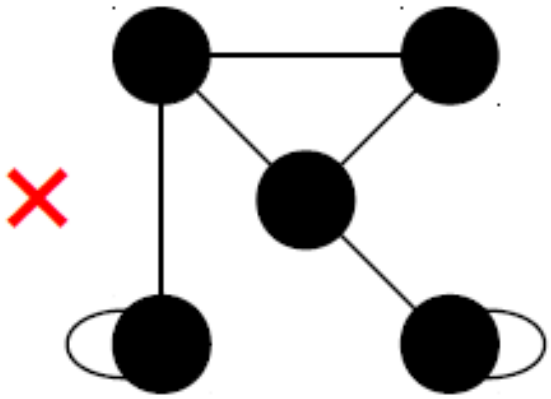


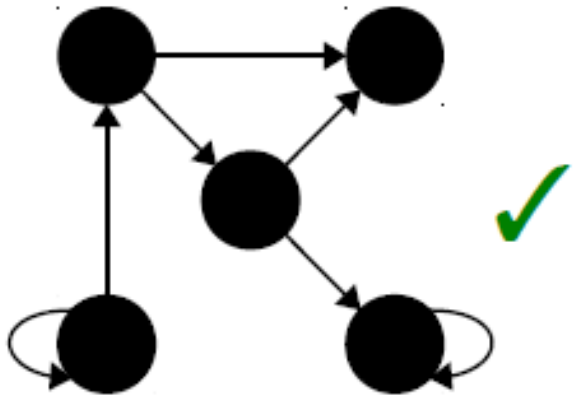
Formalizing Graphs

- 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.

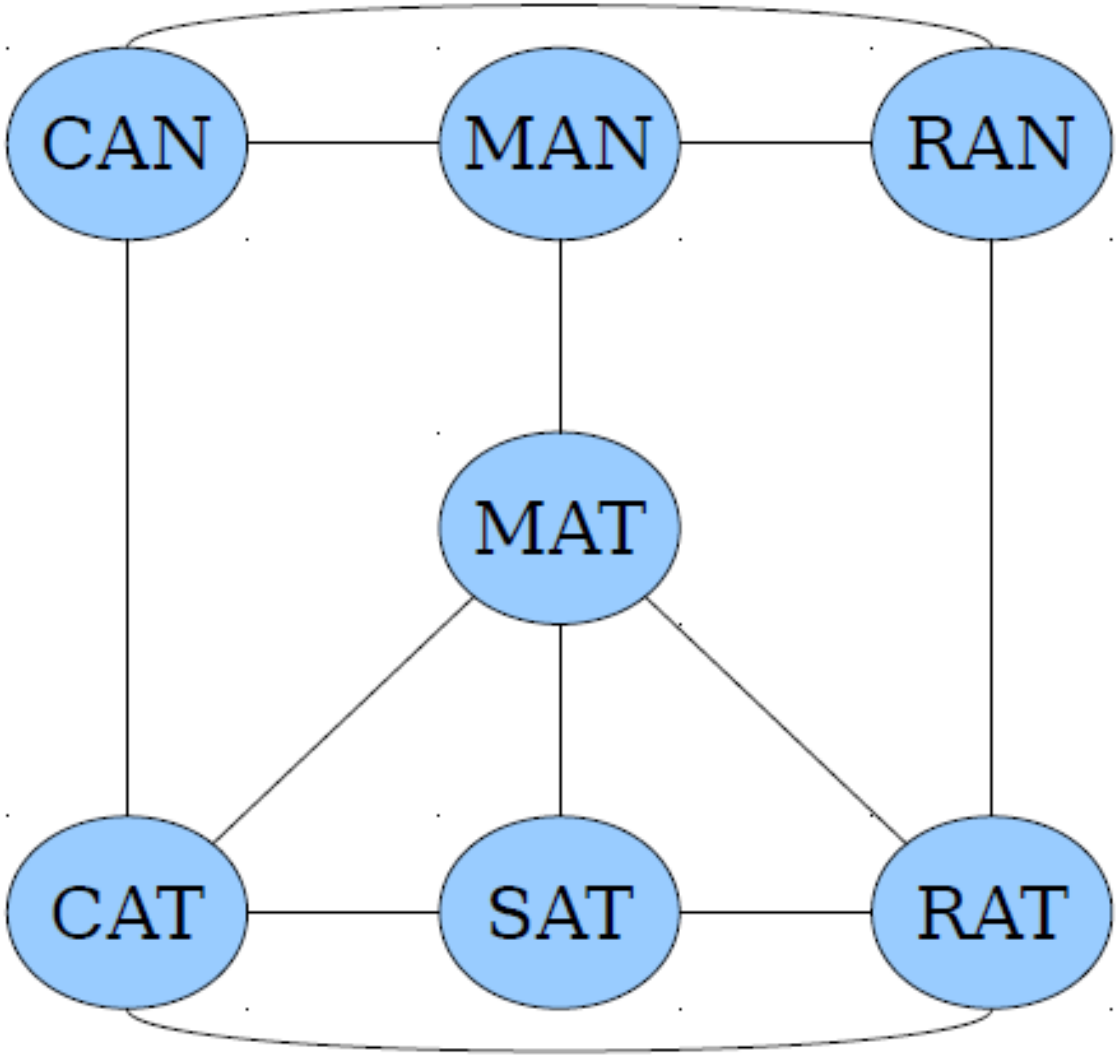
Self-Loops

- 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.

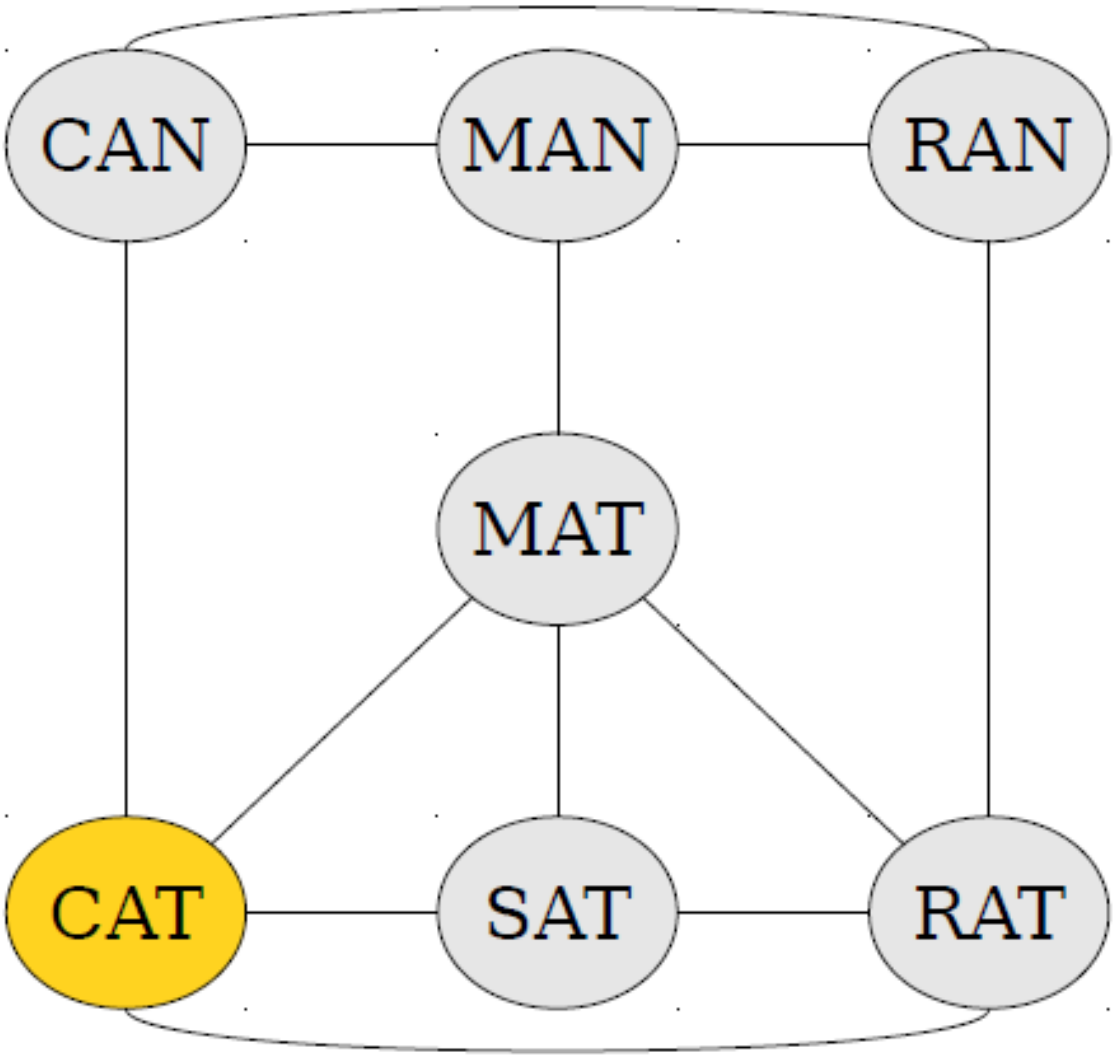




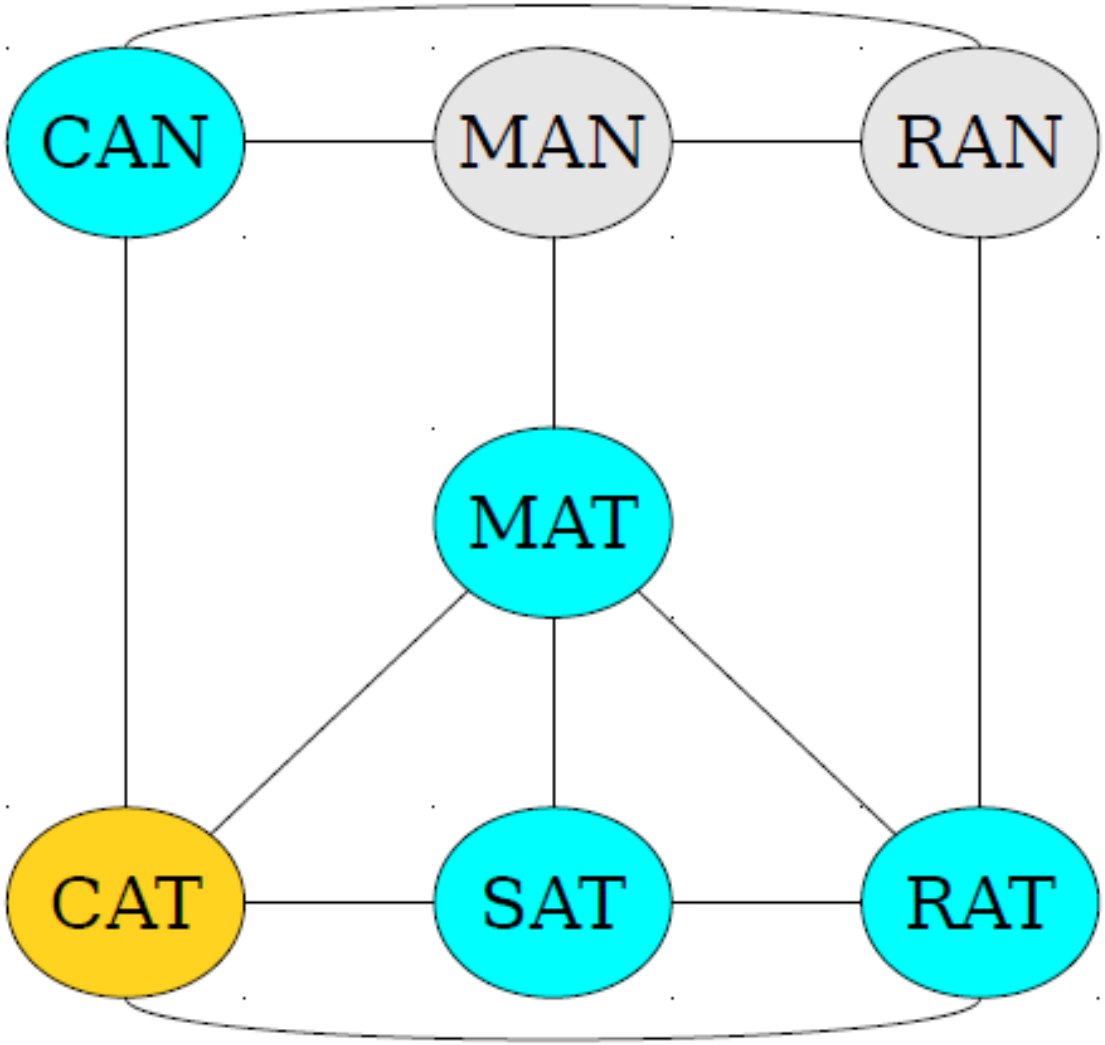
Standard Graph Terminology



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



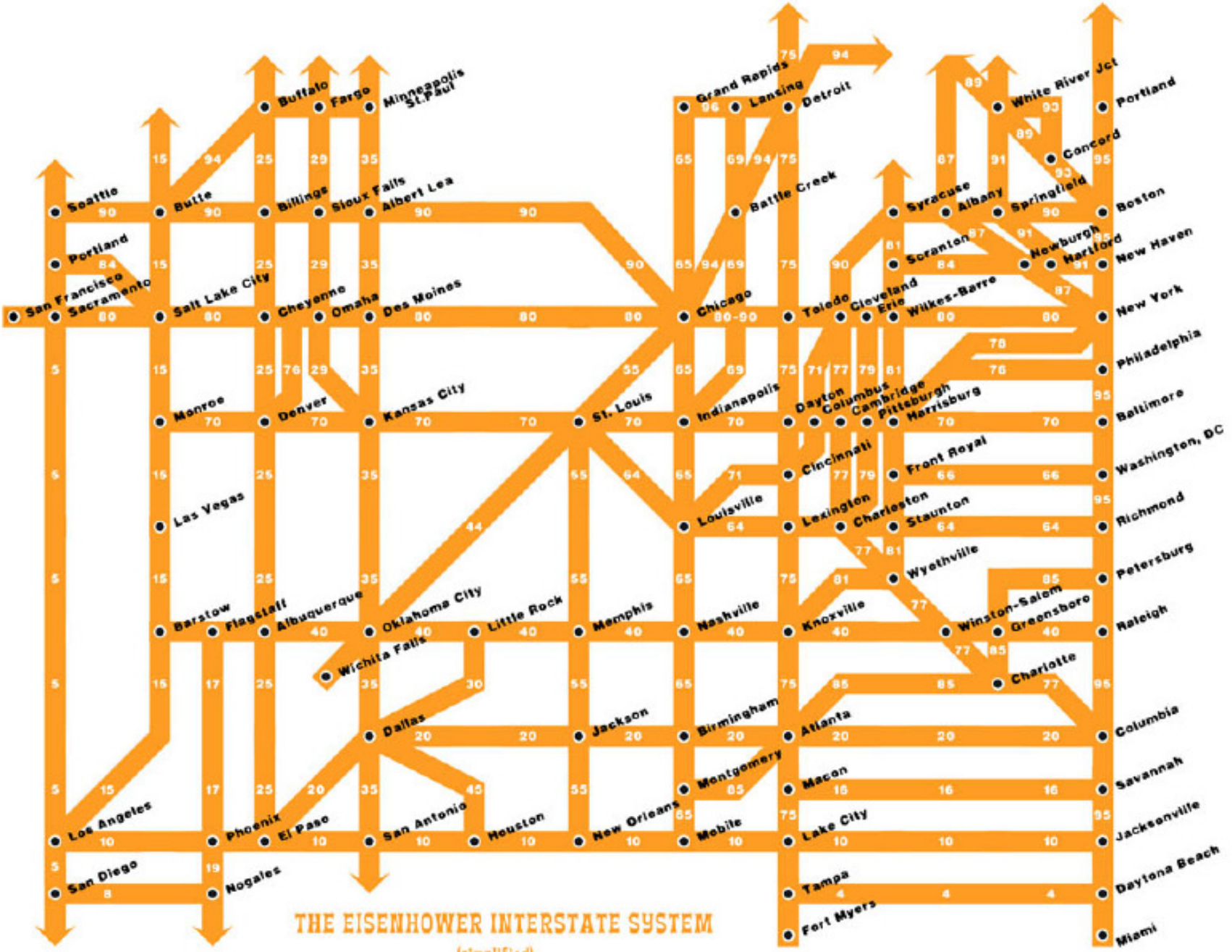
Two nodes are called ***adjacent*** if there is an edge between them.



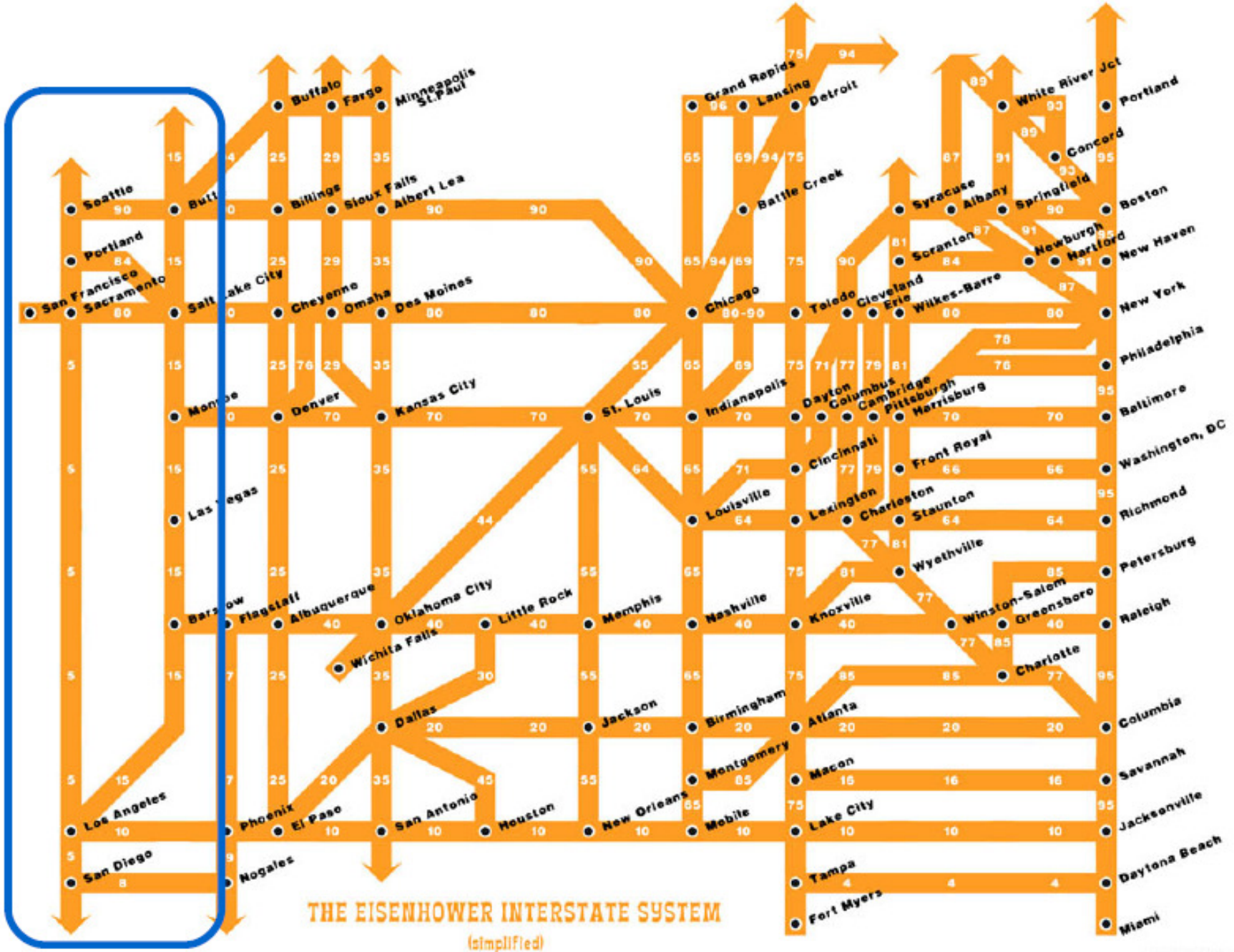
Two nodes are called ***adjacent*** if there is an edge between them.

Using our Formalisms

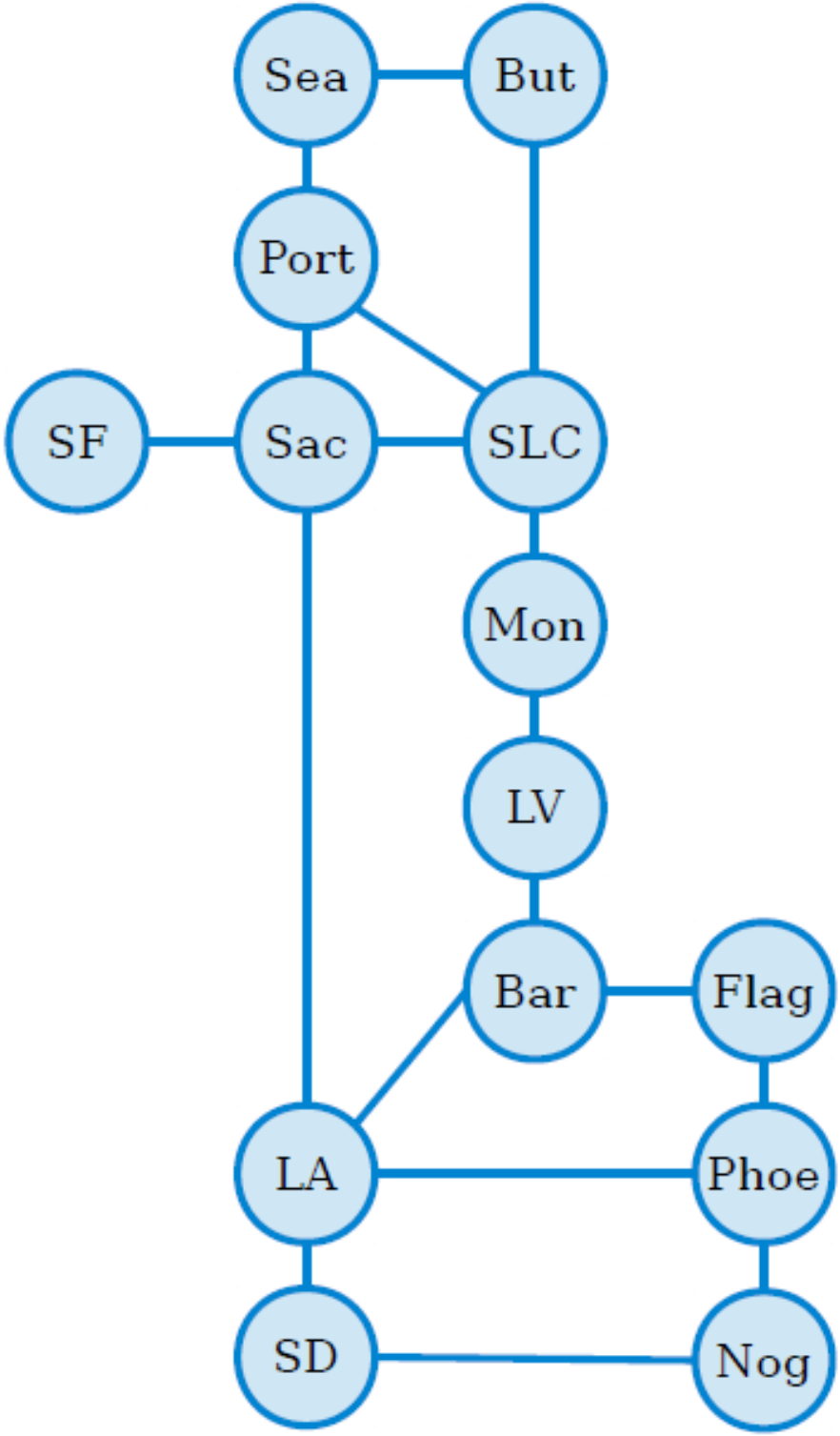
- Let $G = (V, E)$ be a graph.
- Intuitively, two nodes are adjacent if they're linked by an edge.
- Formally speaking, we say that two nodes $u, v \in V$ are ***adjacent*** if $\{u, v\} \in E$.

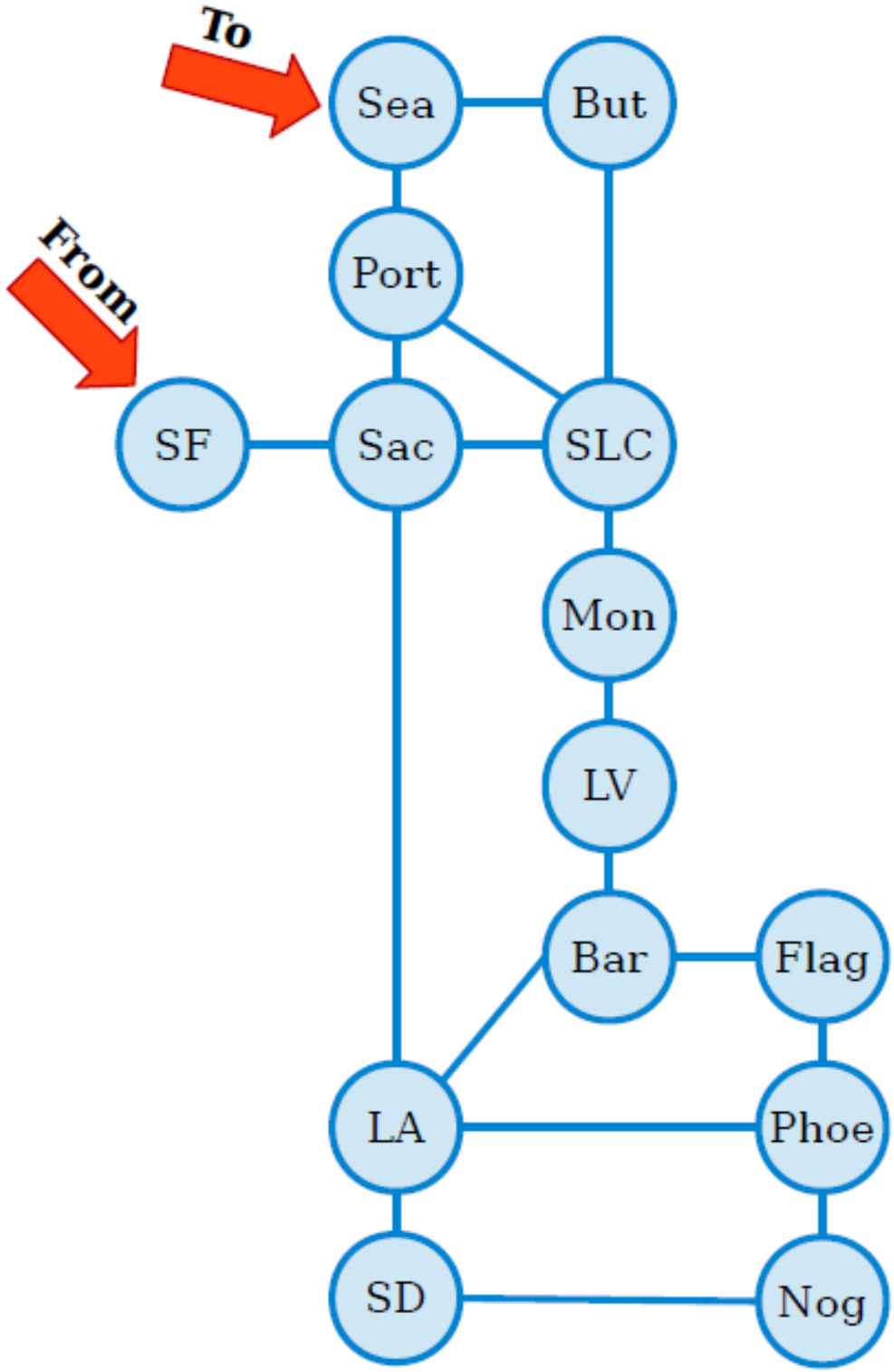


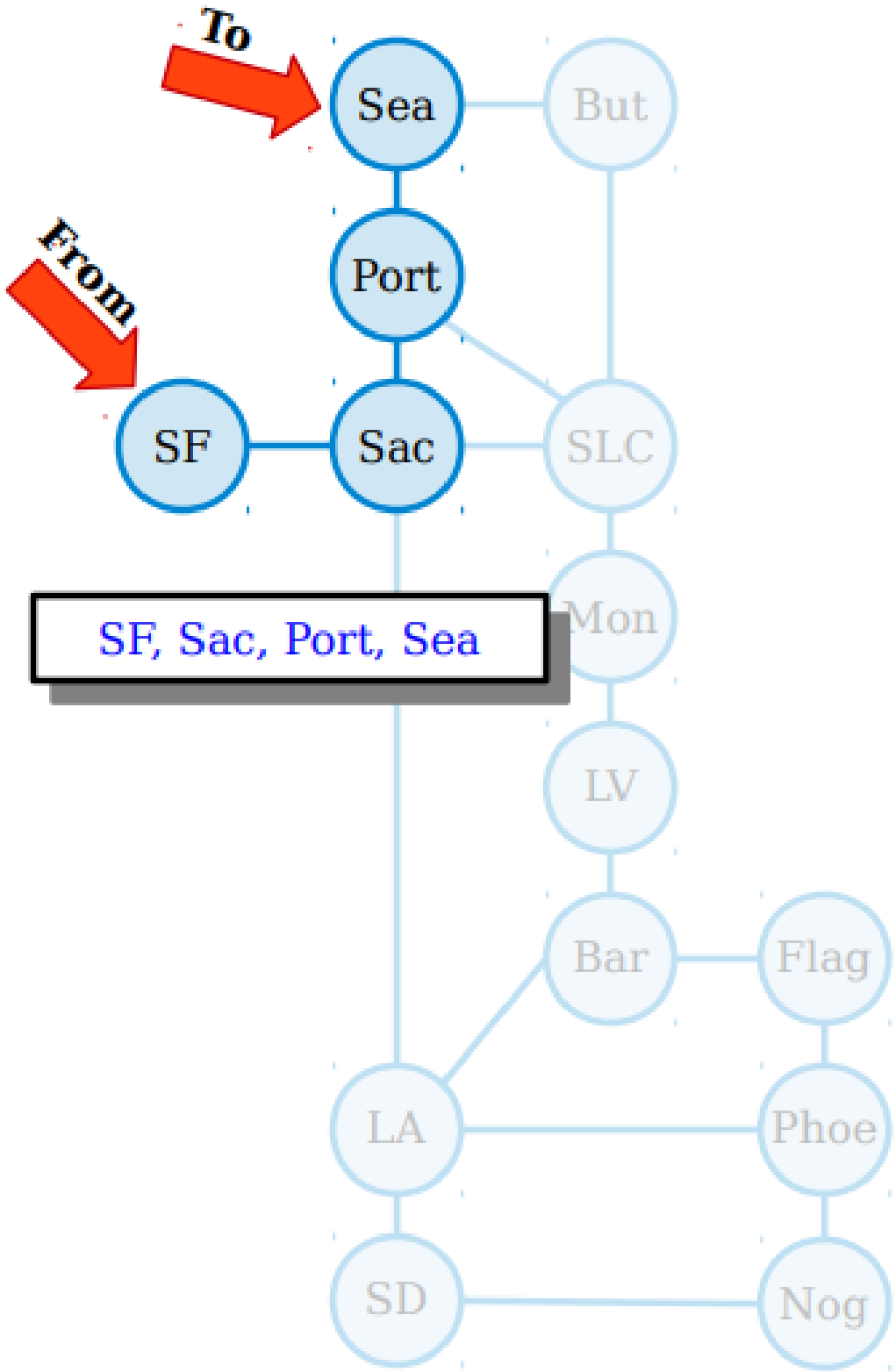
http://strangemaps.files.wordpress.com/2007/02/fullinterstatemap-web.jpg

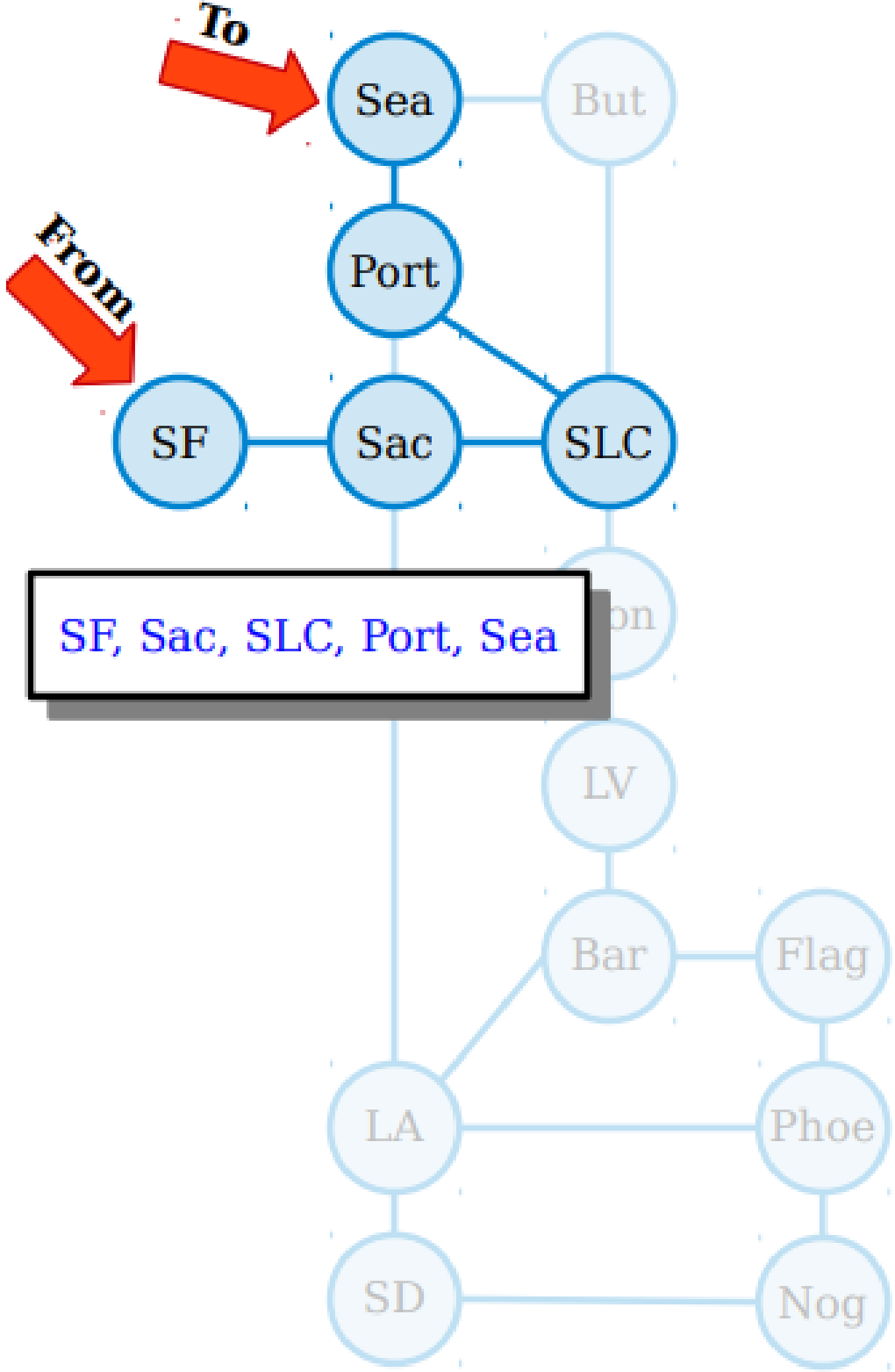


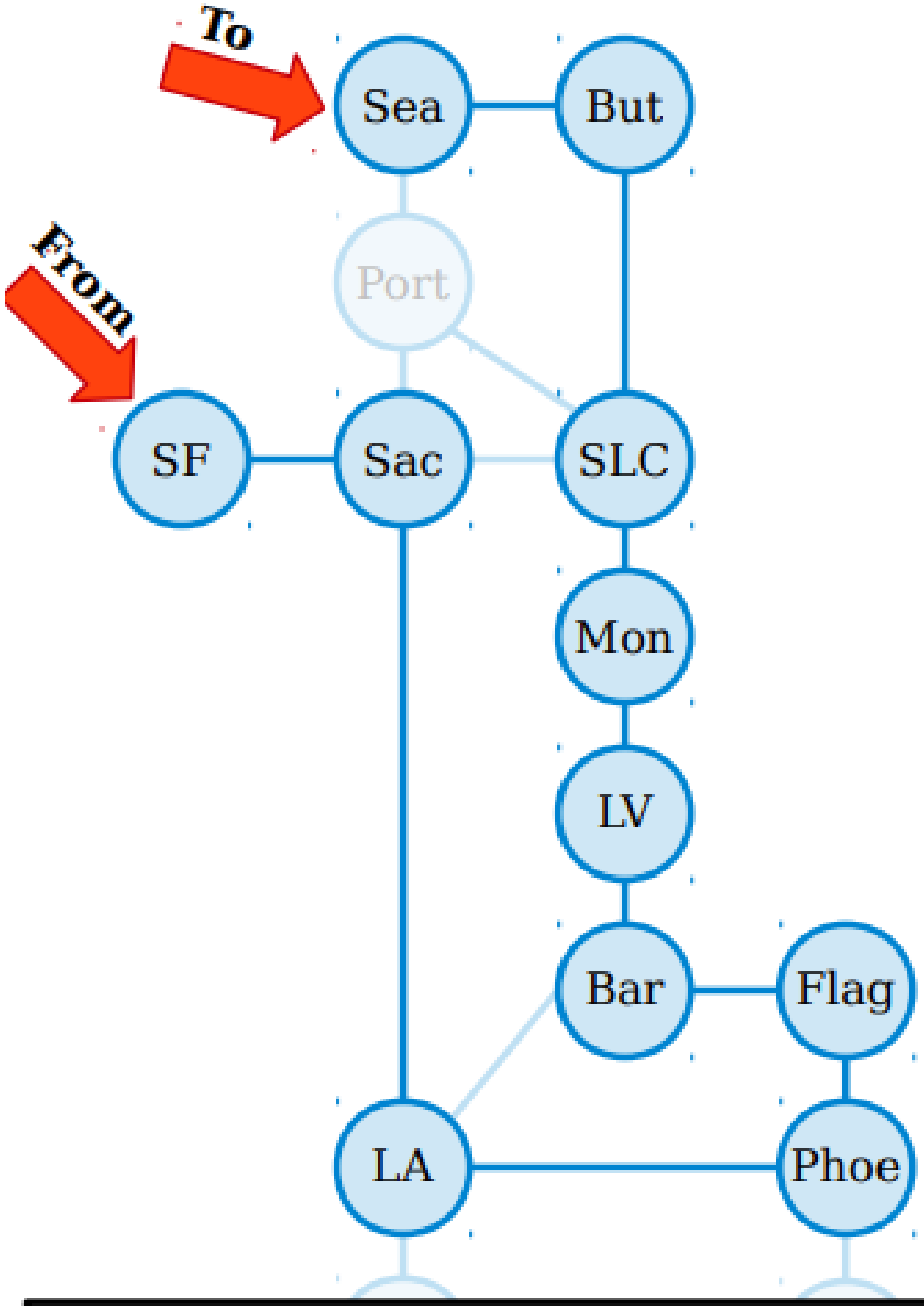
http://strangemaps.files.wordpress.com/2007/02/fullinterstatemap-web.jpg


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

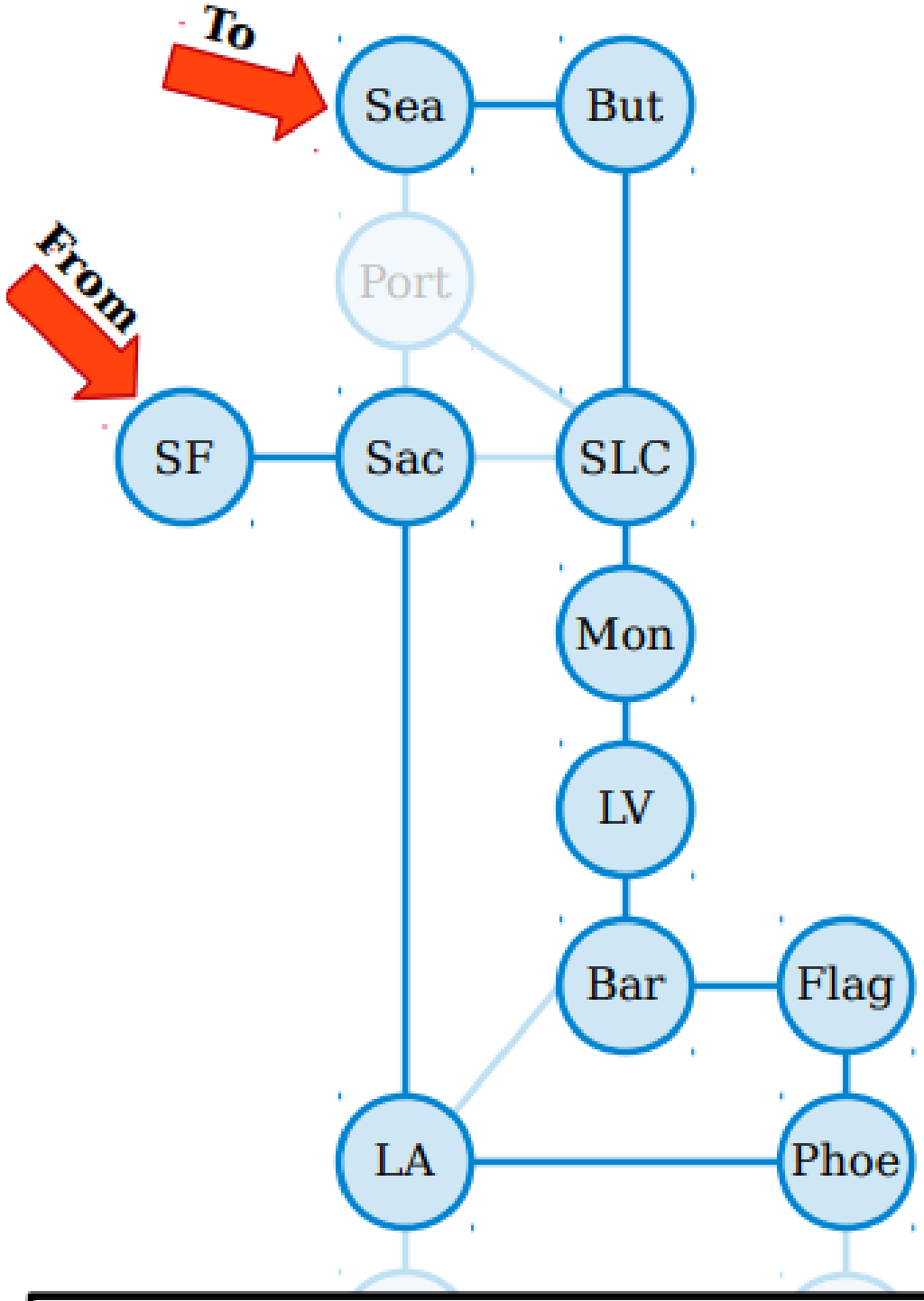

To
From
Sea
But
Port
SF
Sac
SLC
Mon
LV
Bar
Flag
LA
Phoe
SD
Nog


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

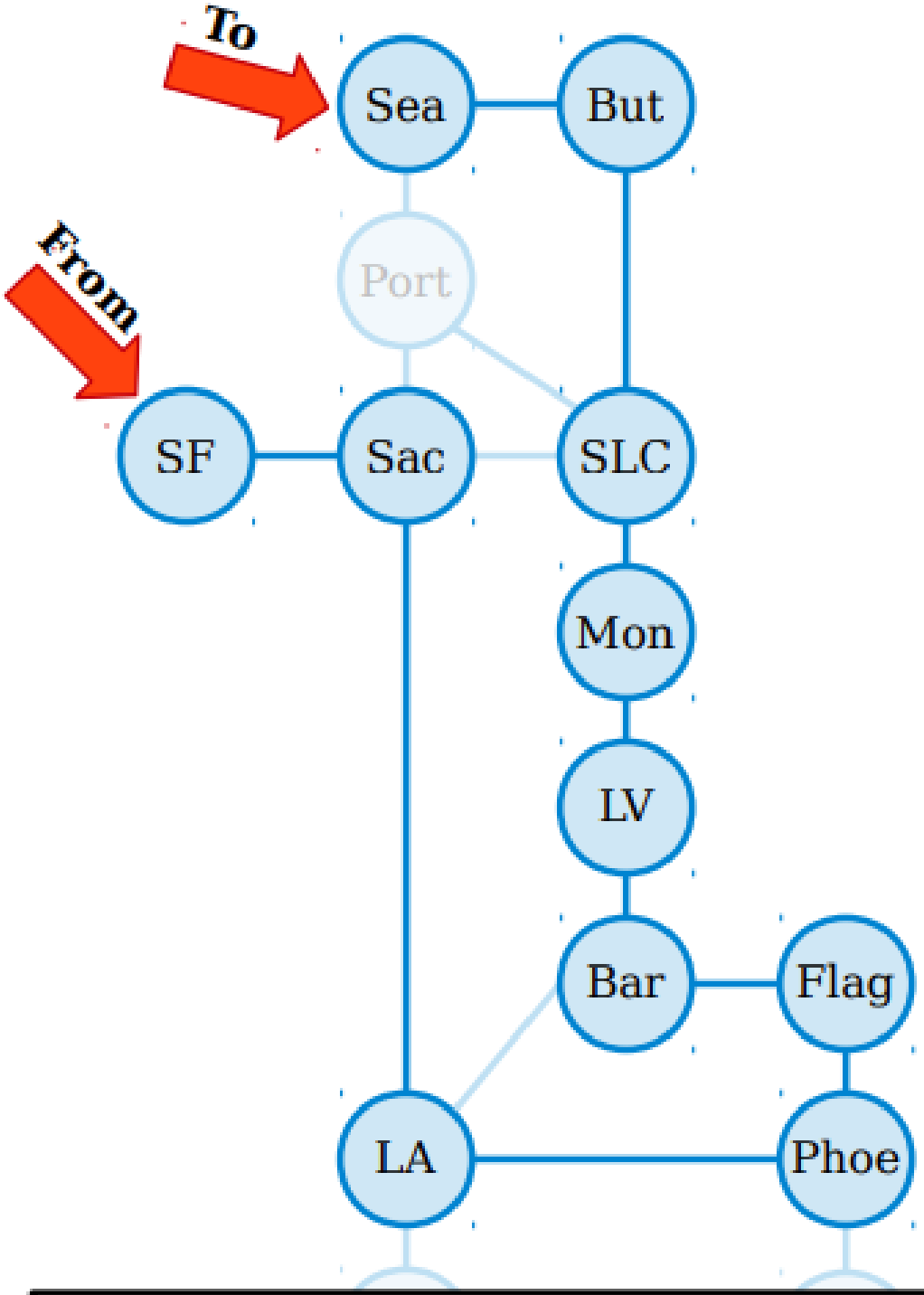


SF, Sac, LA, Phoe, Flag, Bar, LV, Mon, SLC, But, Sea



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.

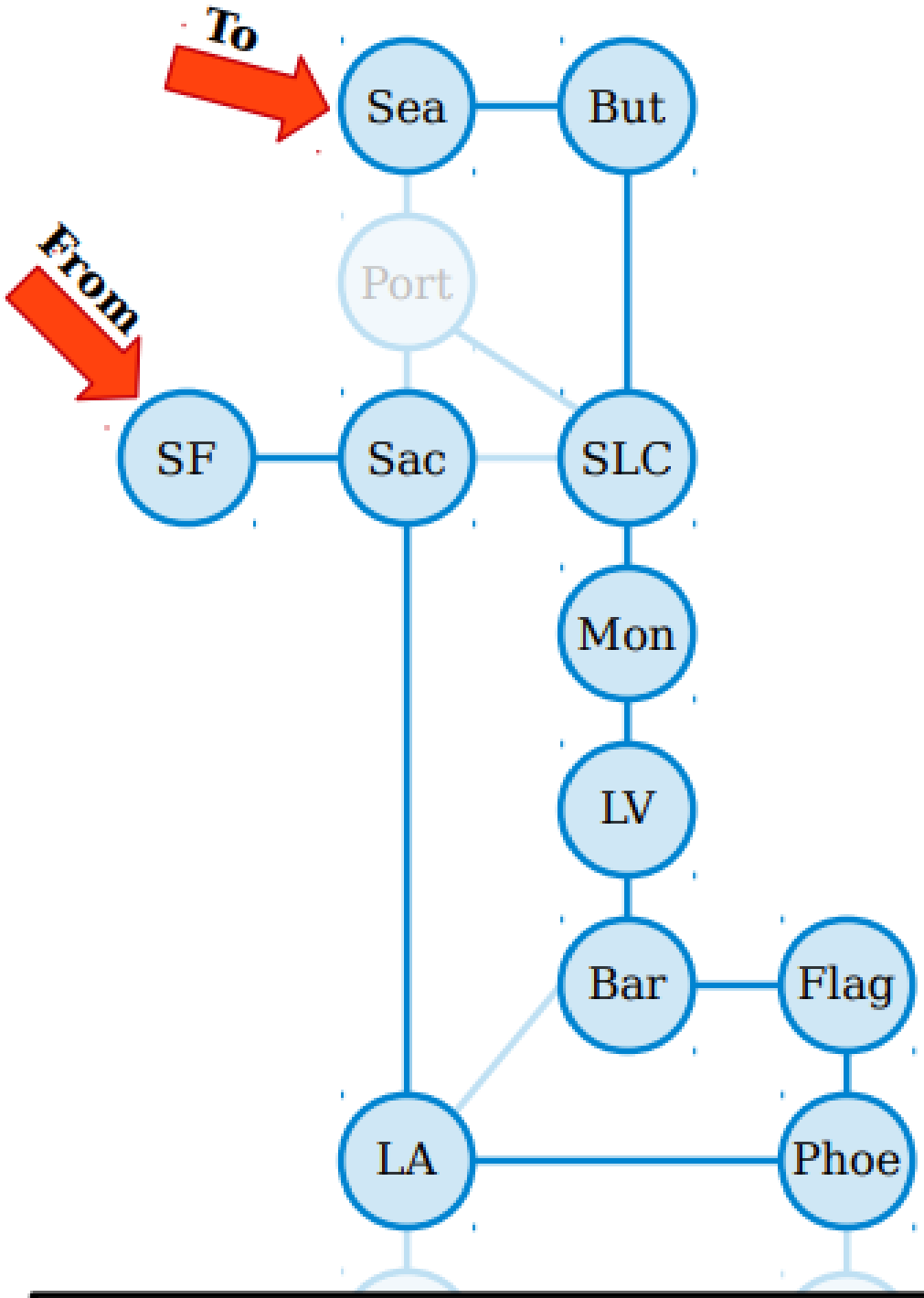
SF, Sac, LA, Phoe, Flag, Bar, LV, Mon, SLC, But, Sea



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

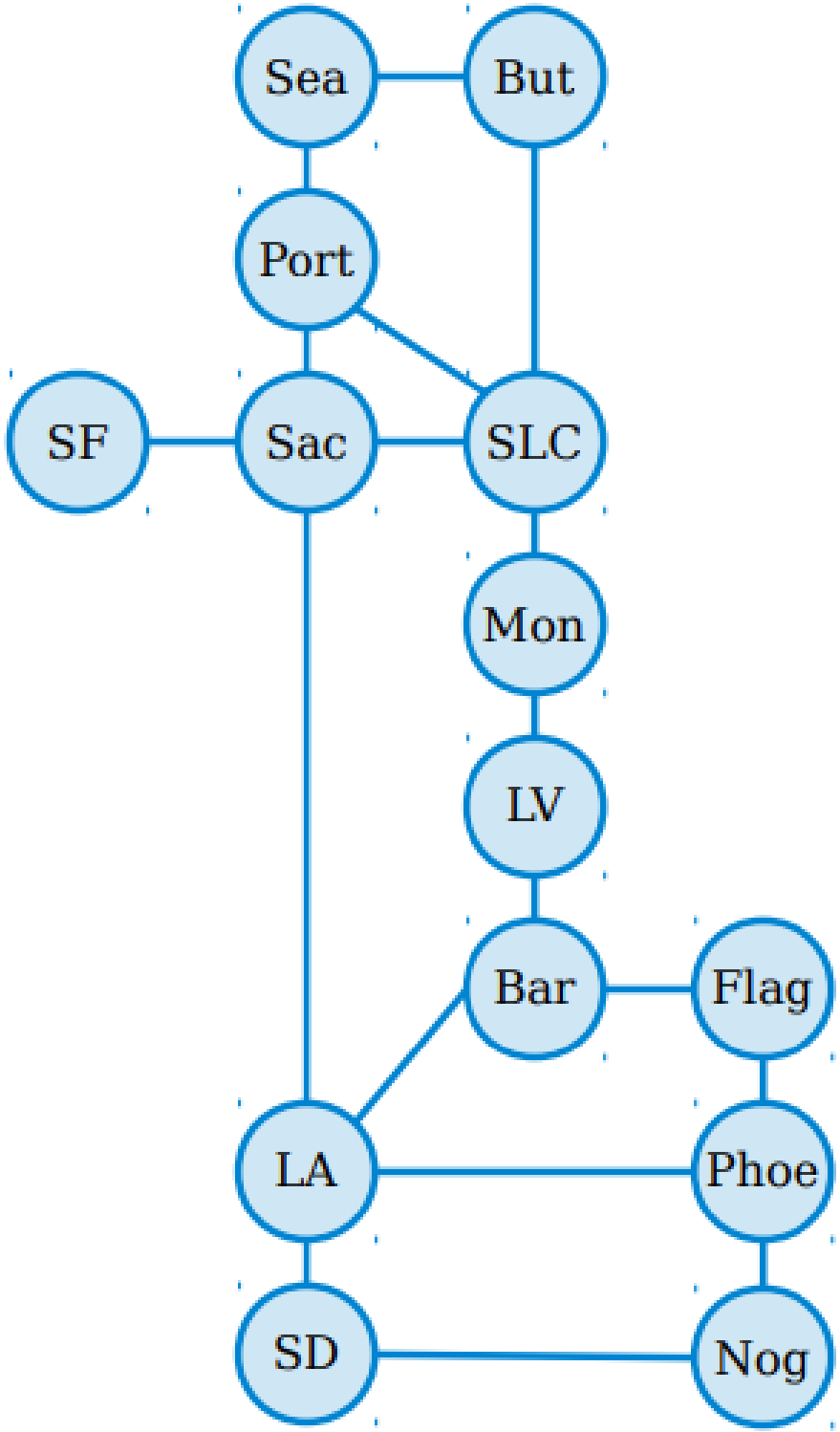


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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

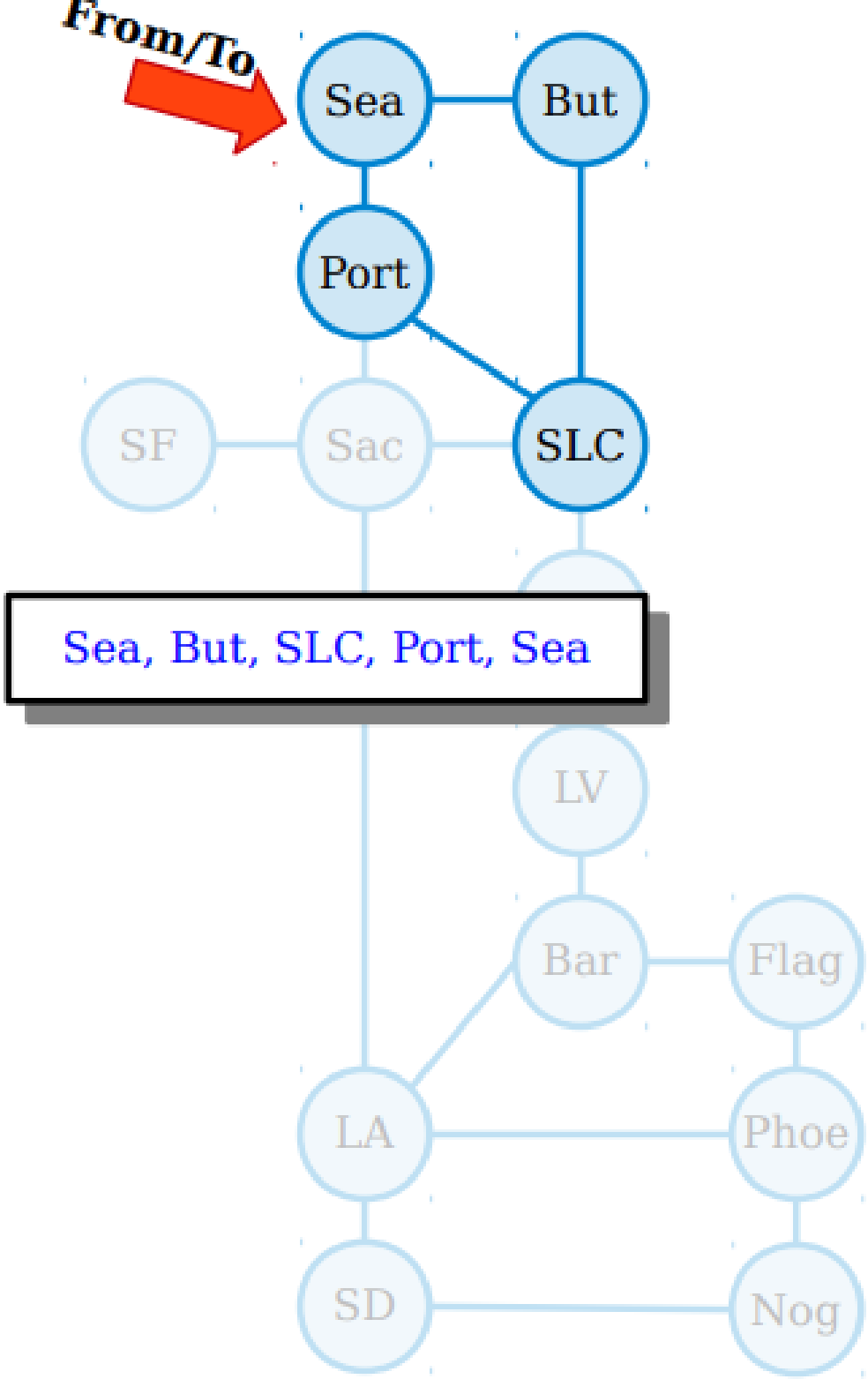
(This path has length 10, but visits 11 cities.)

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.

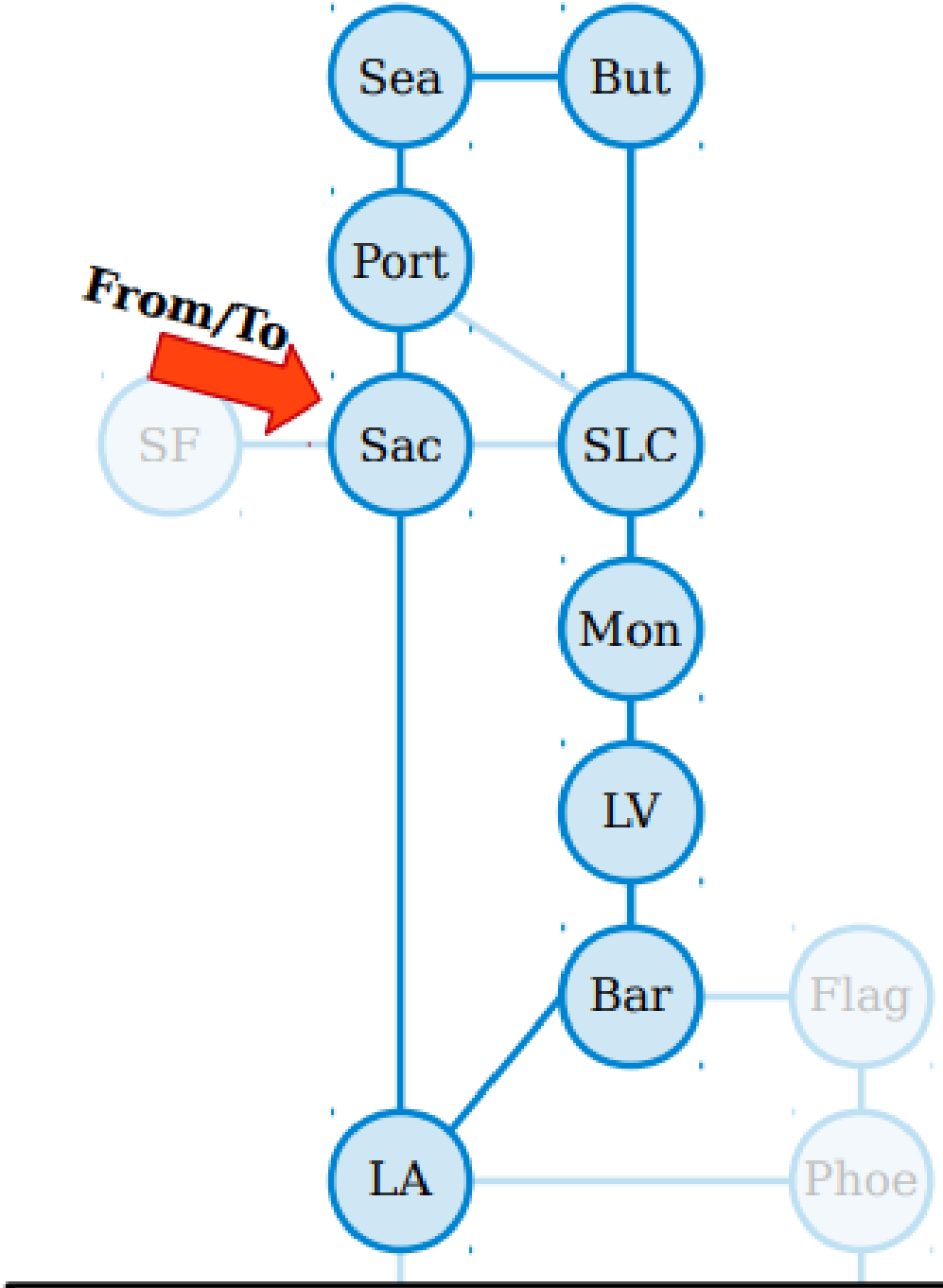
The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.



Sea, But, SLC, Port, 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.

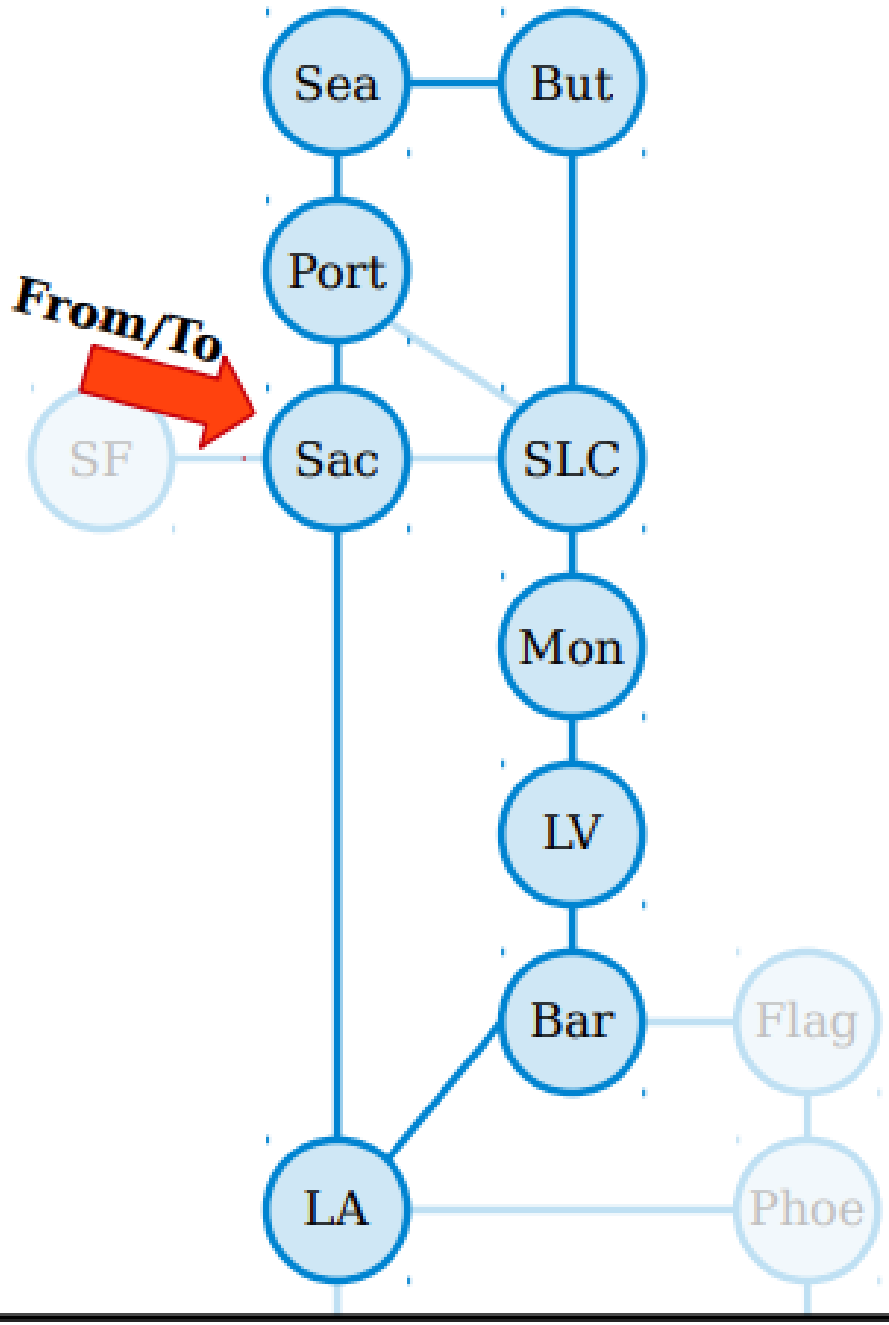
The ***length*** of the path $v_1, \ldots, v_n$ is $n - 1$.



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

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

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

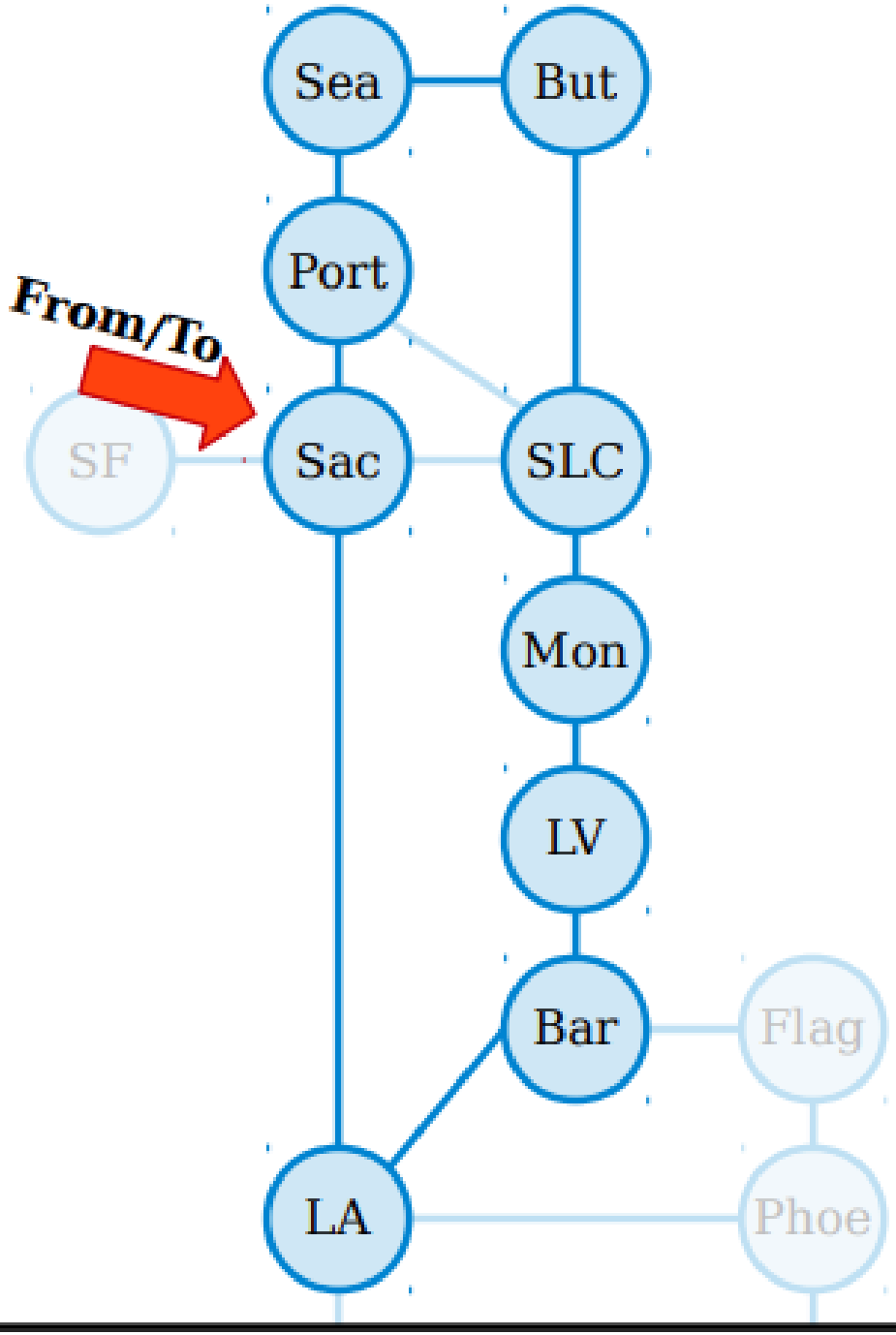


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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



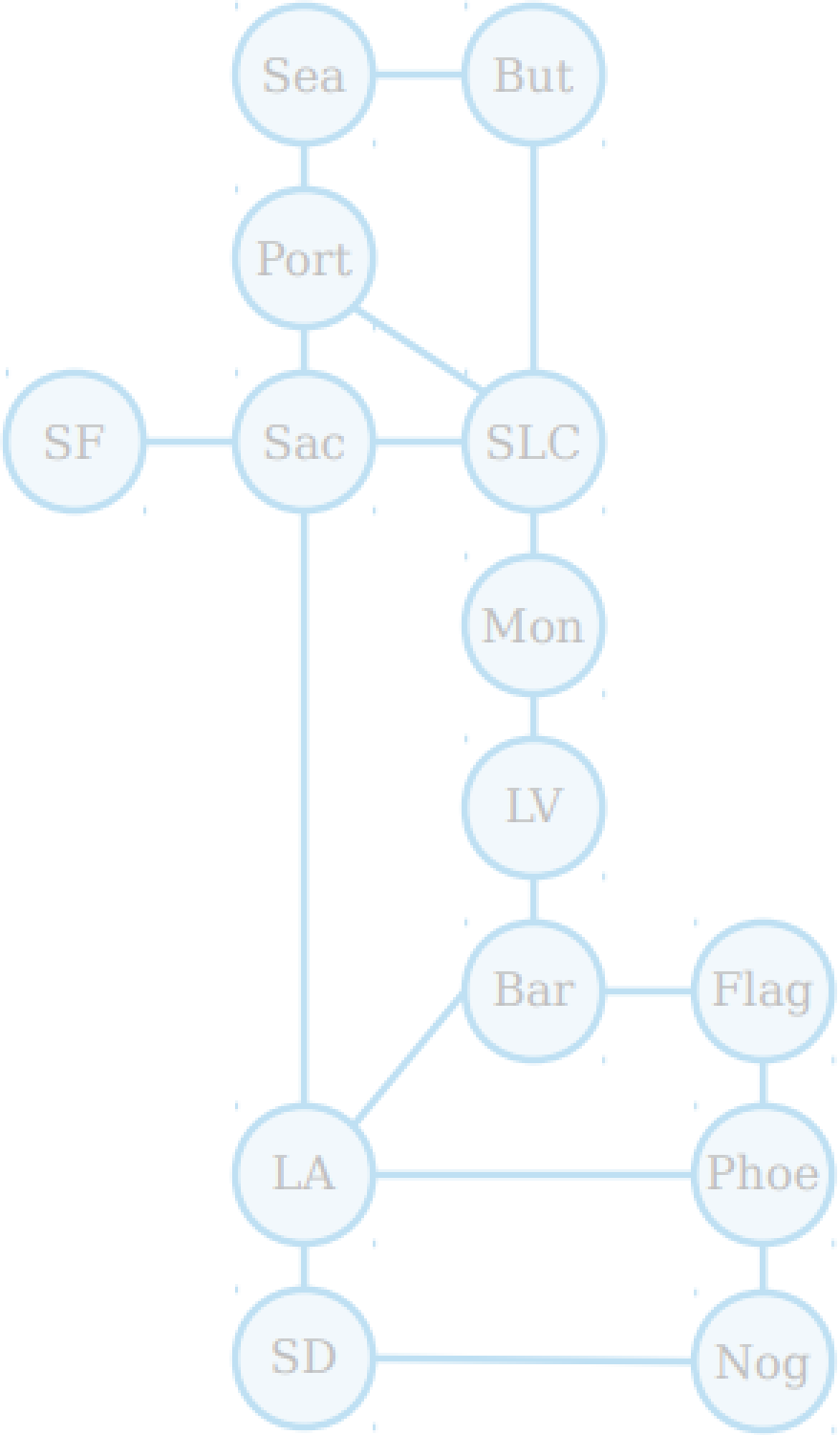
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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.)

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

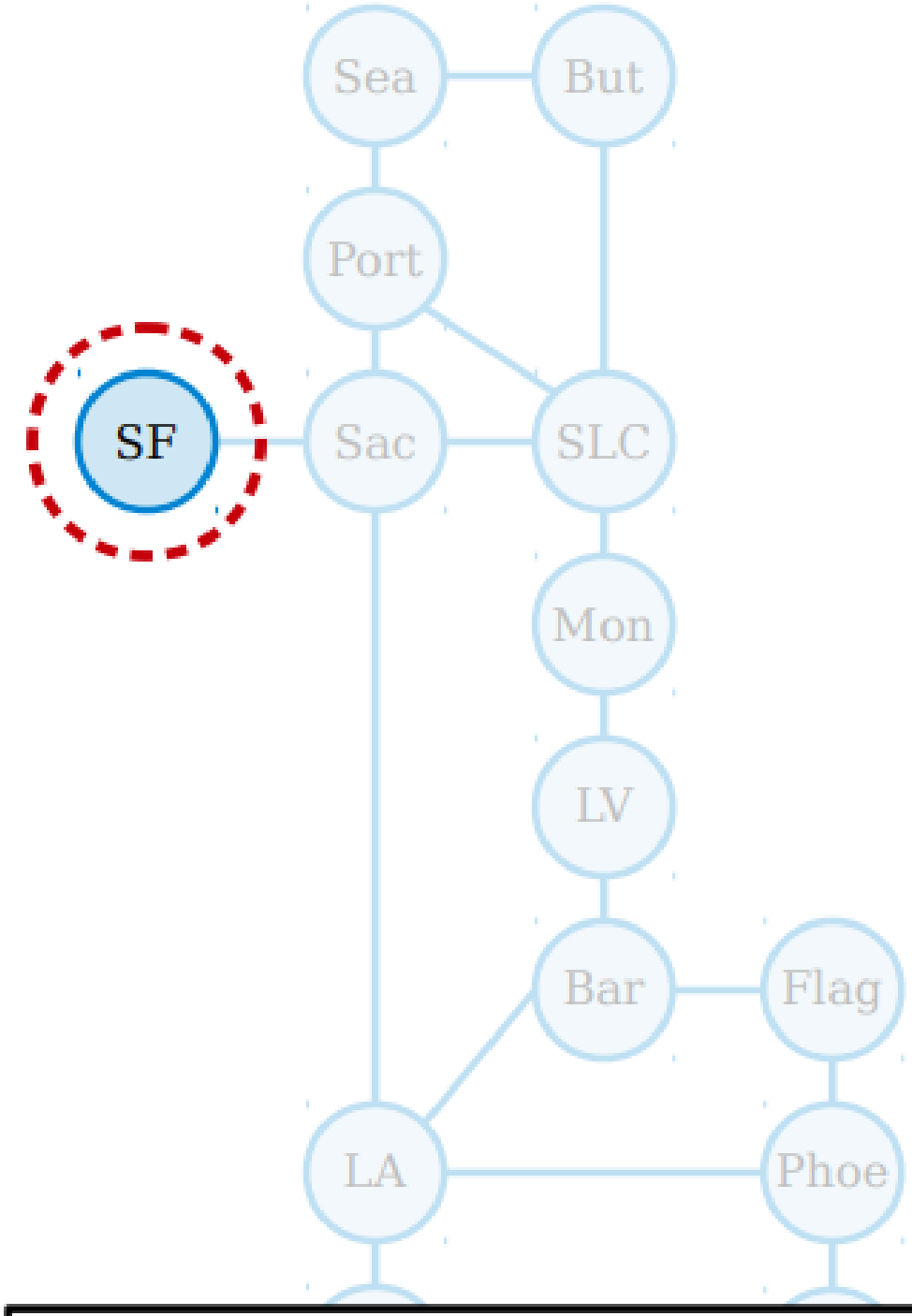
Sac, Port, Sea, But, SLC, Mon, LV, Bar, LA, Sac



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

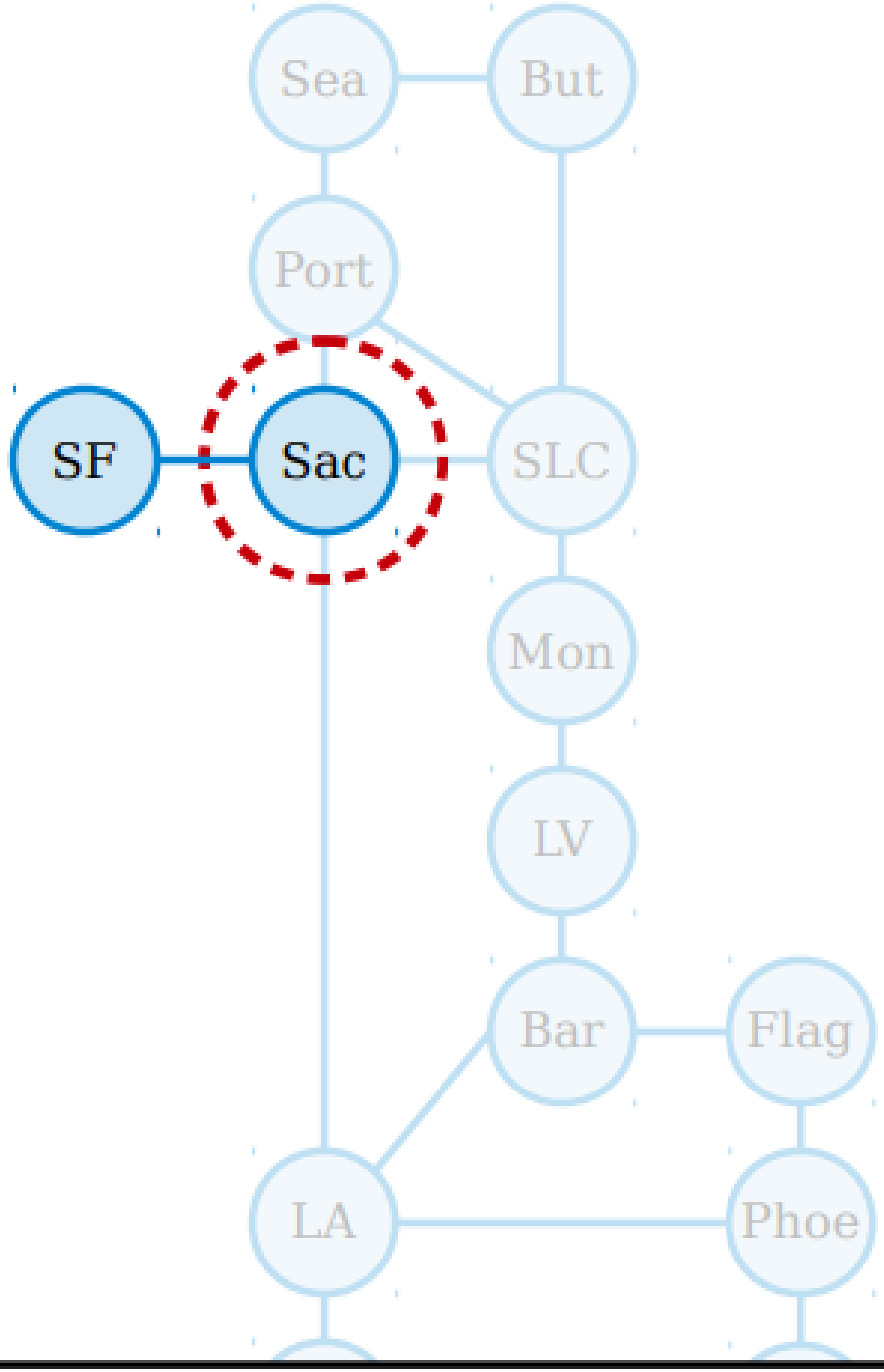


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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

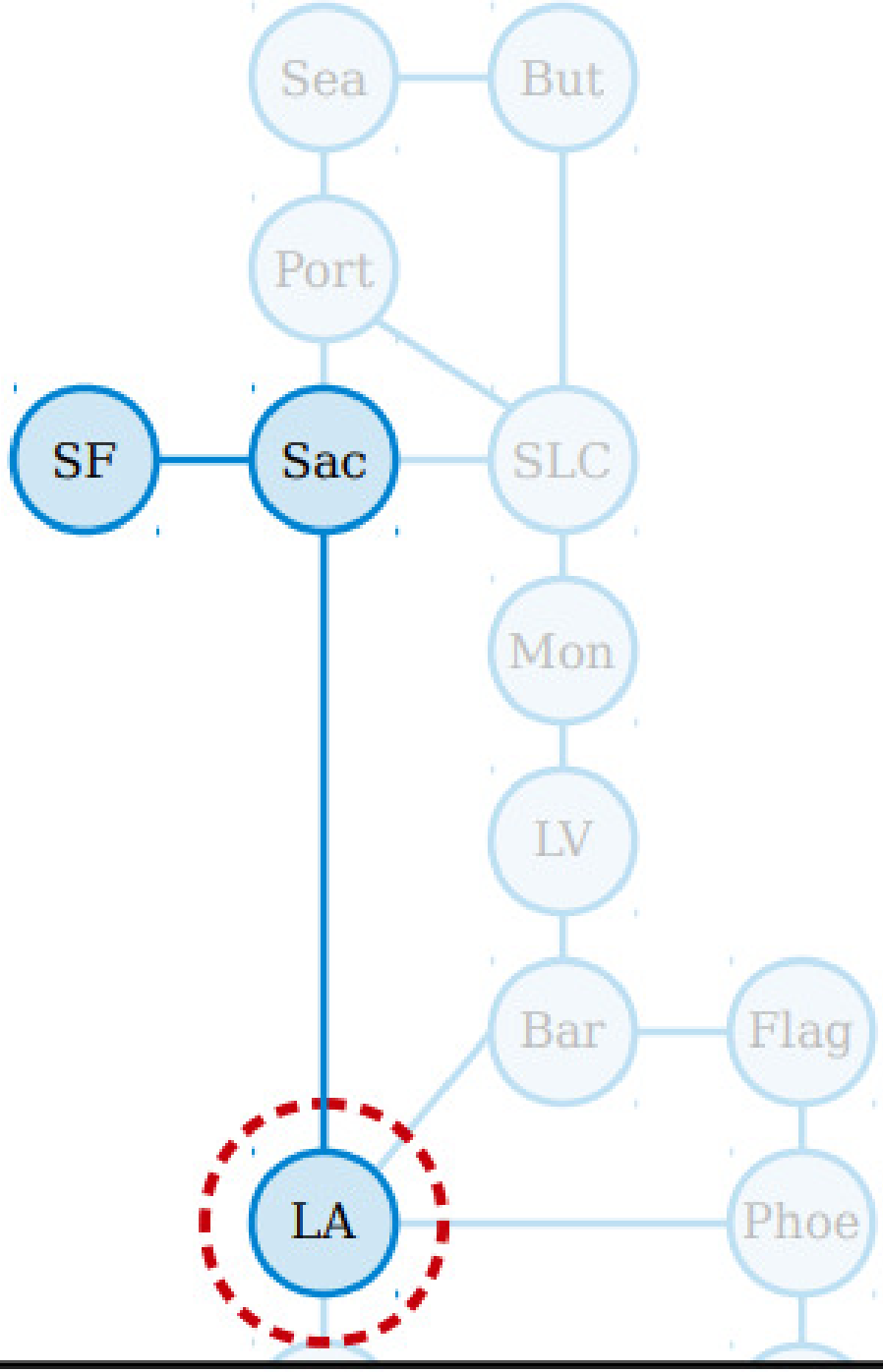


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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

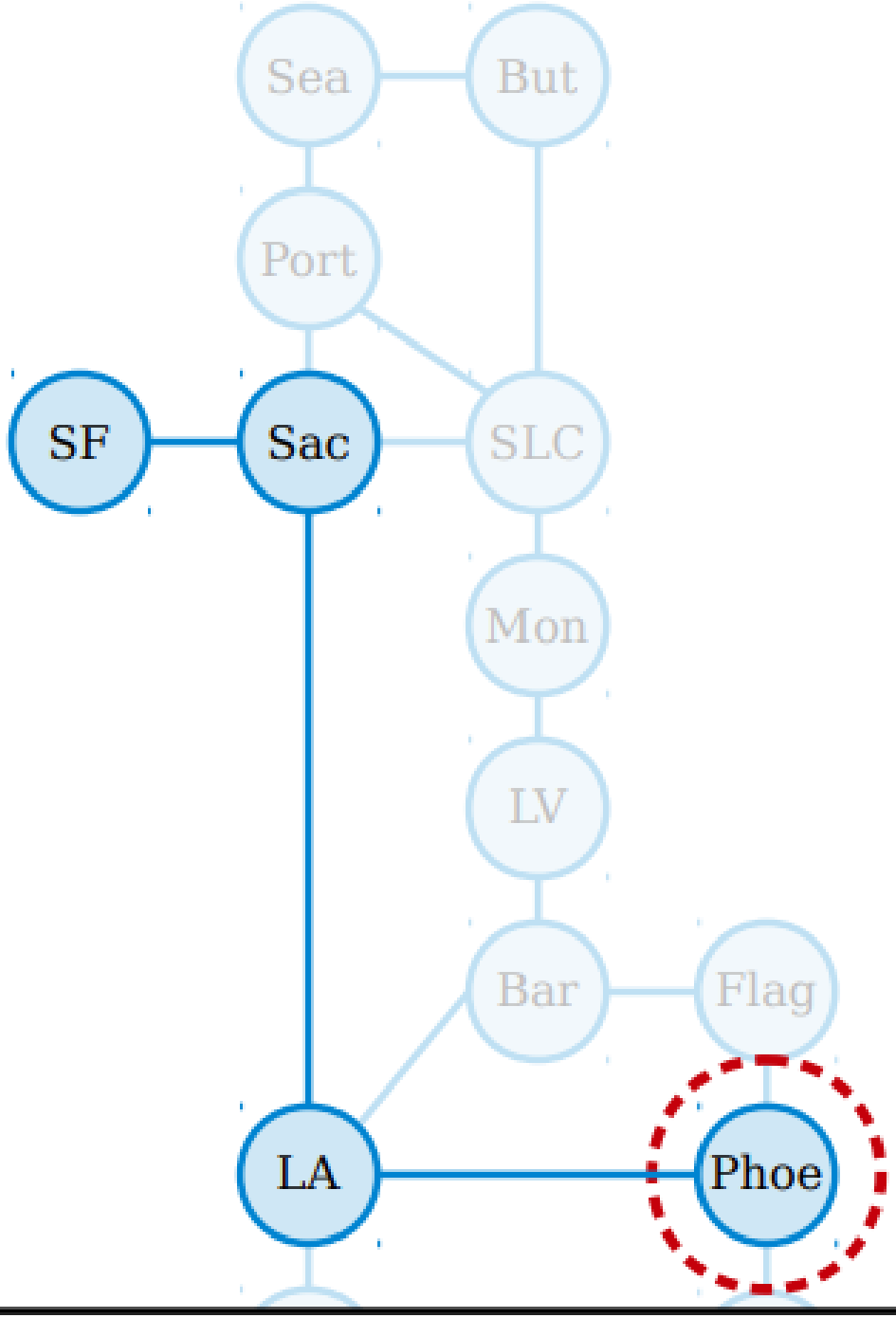


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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

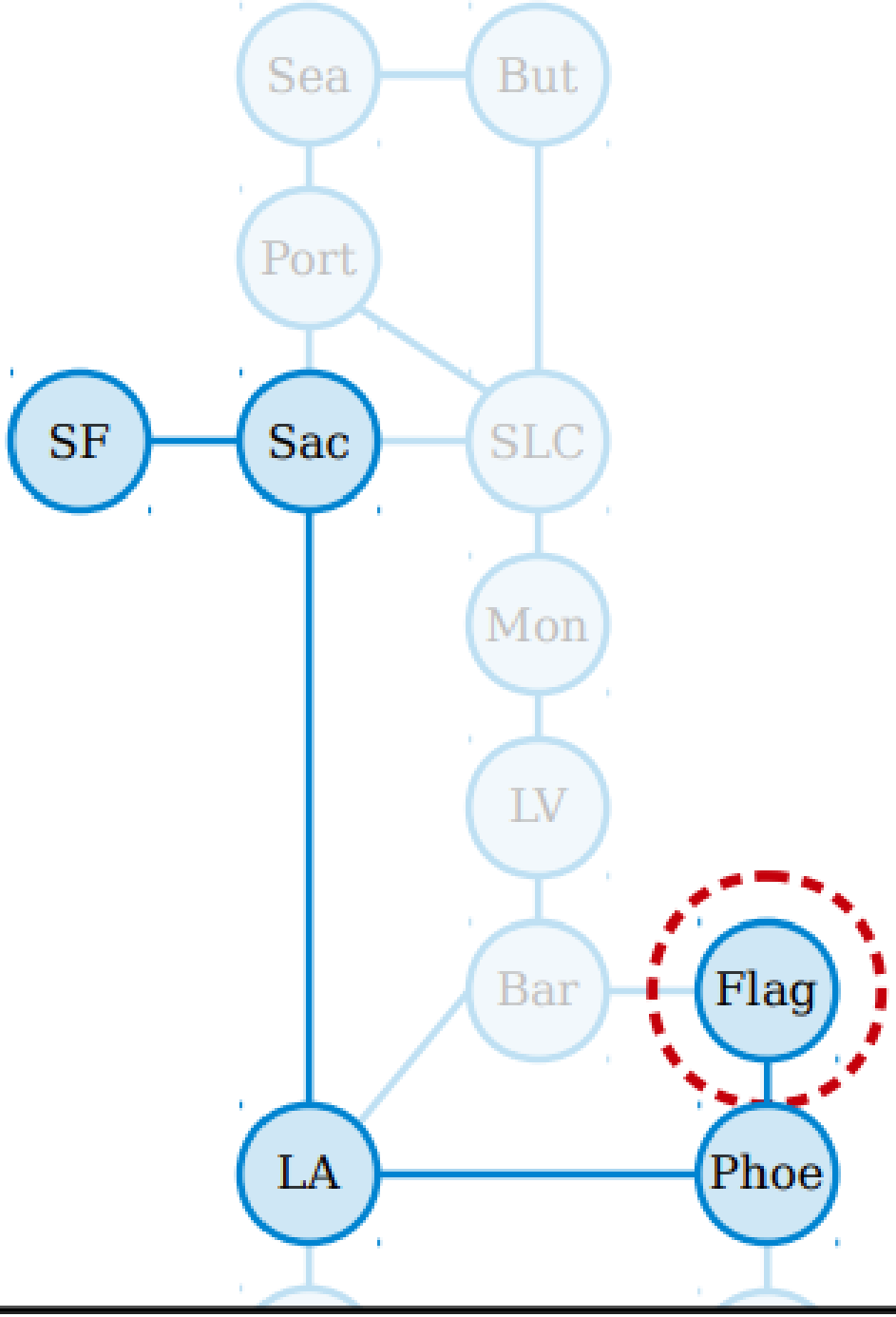


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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

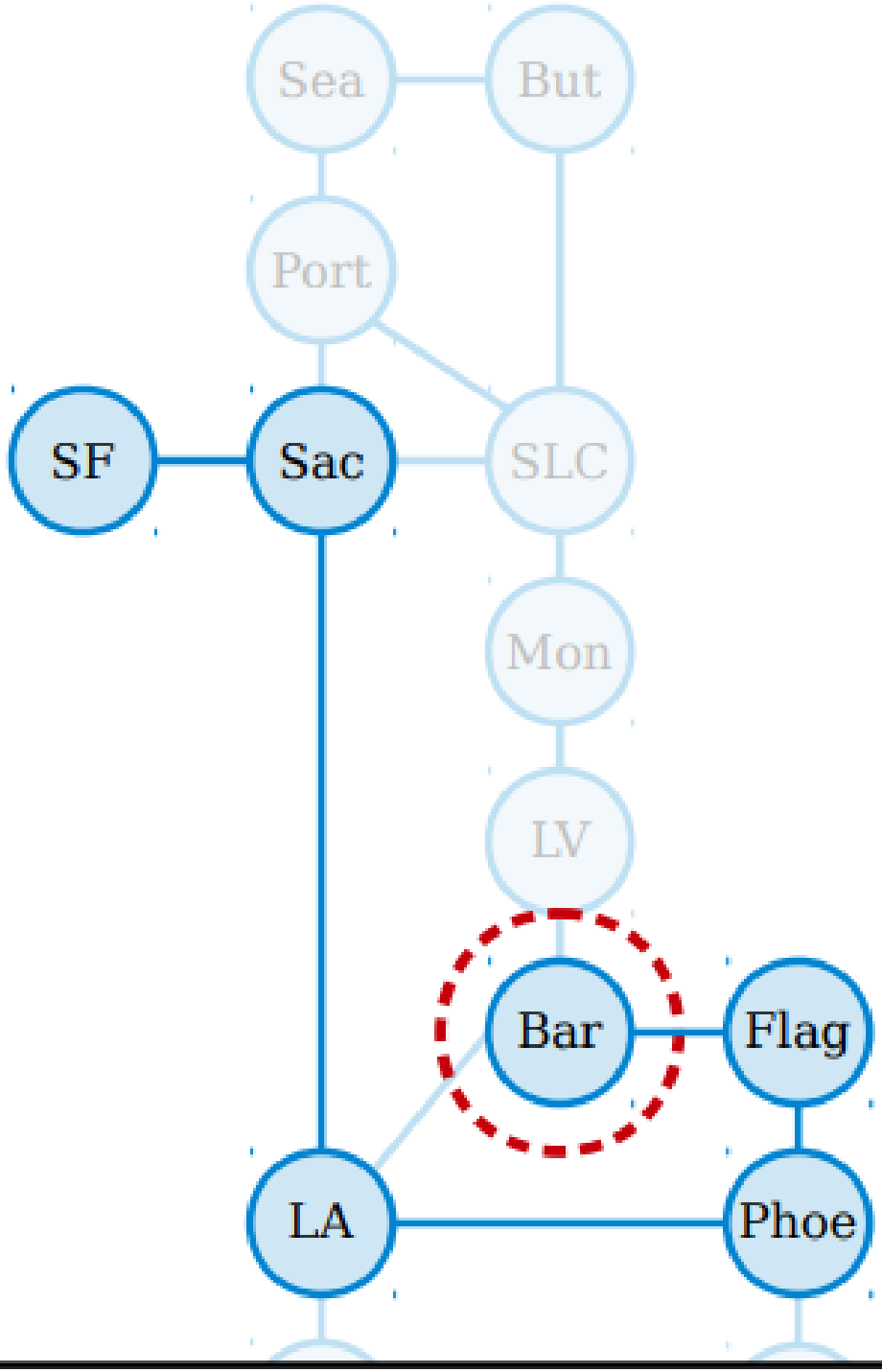


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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

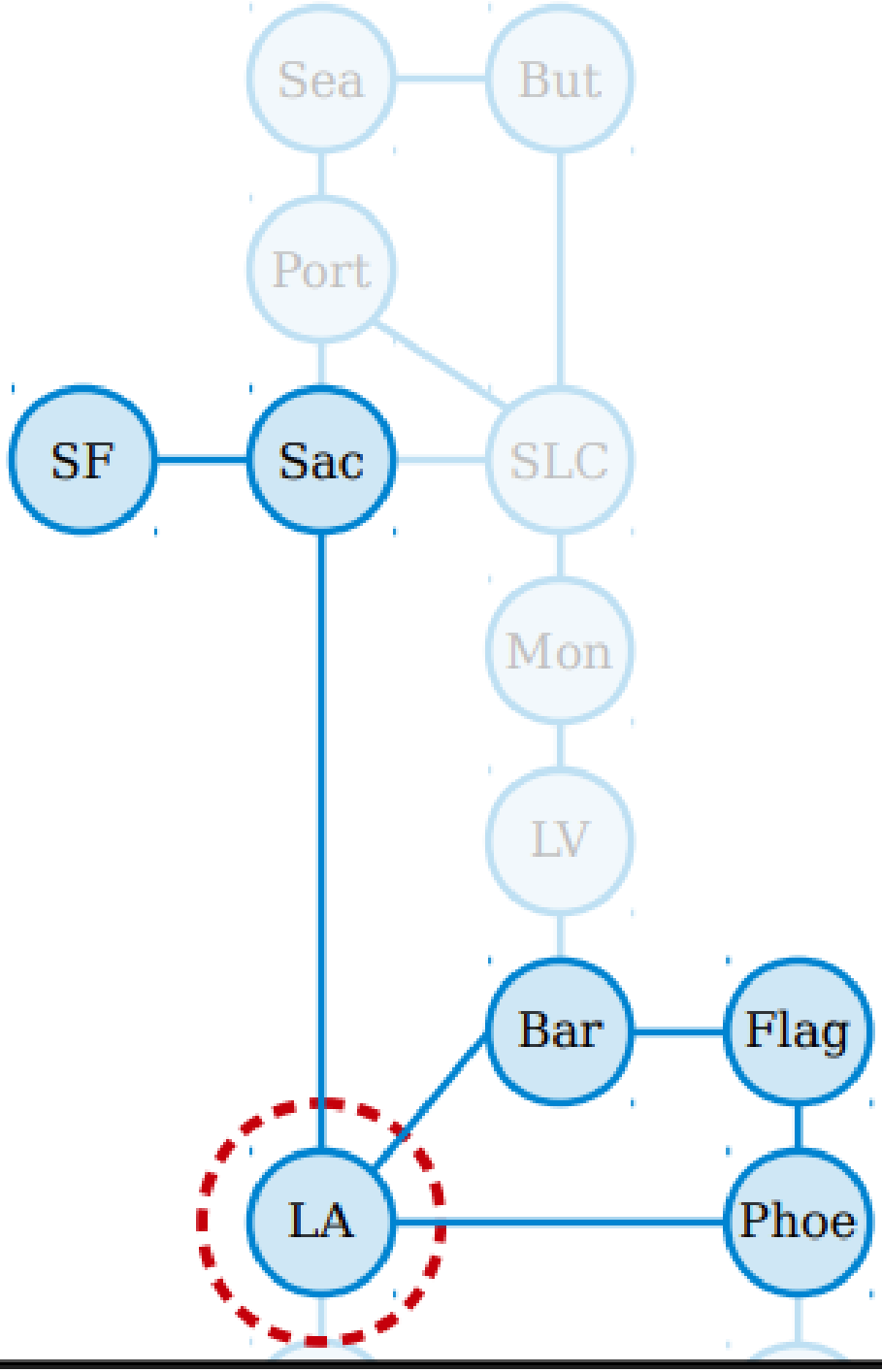


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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

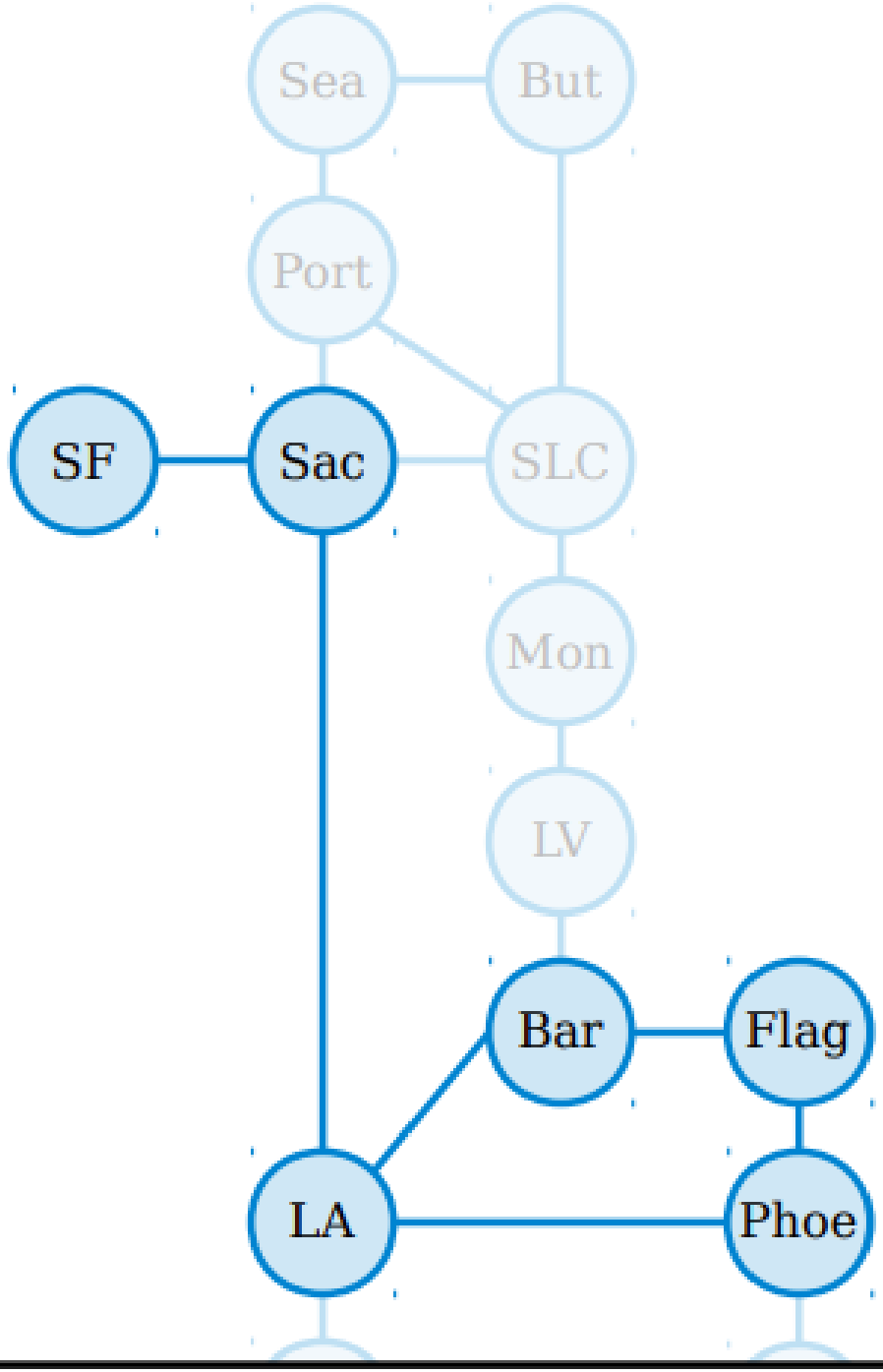


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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

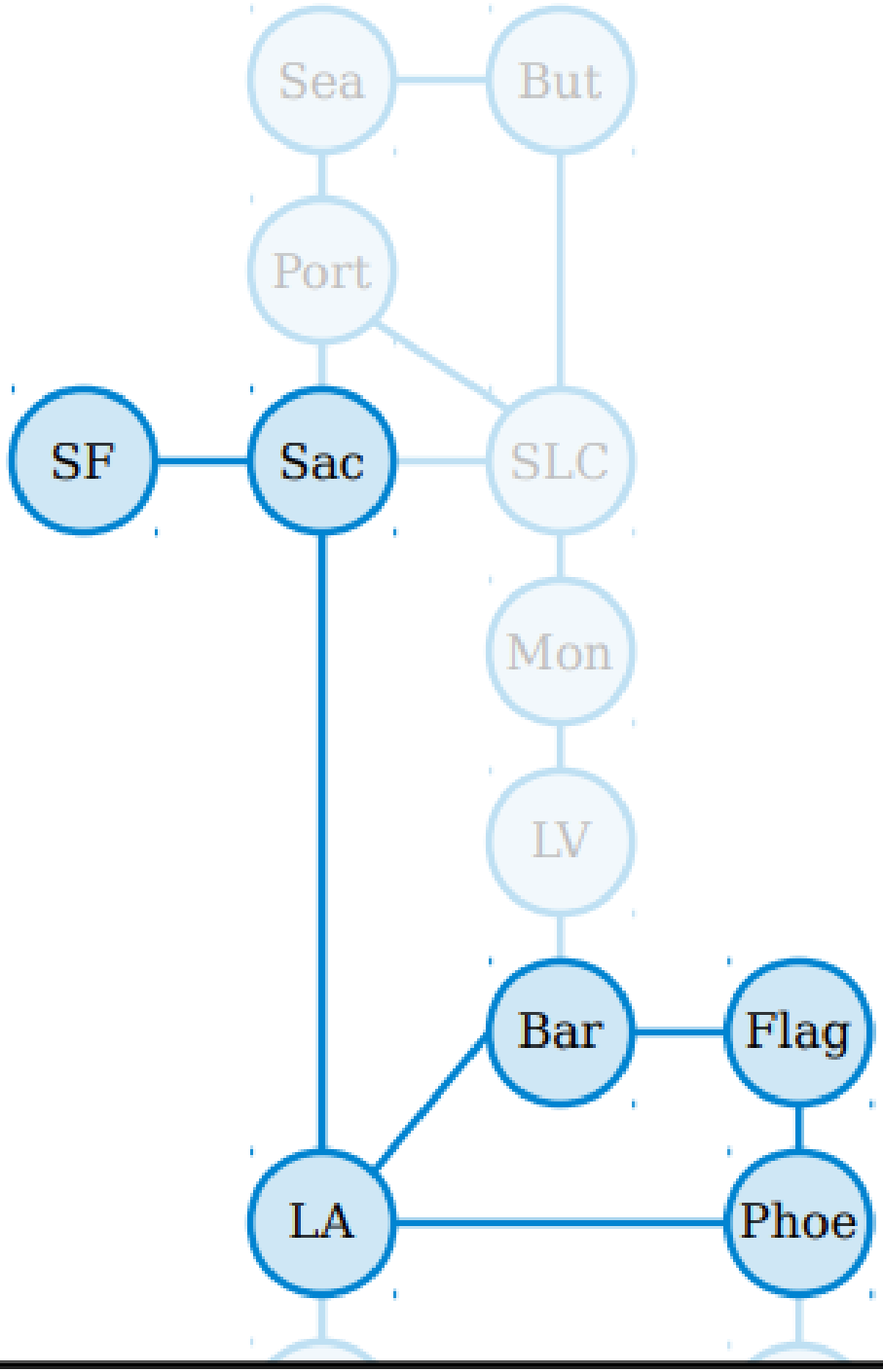


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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



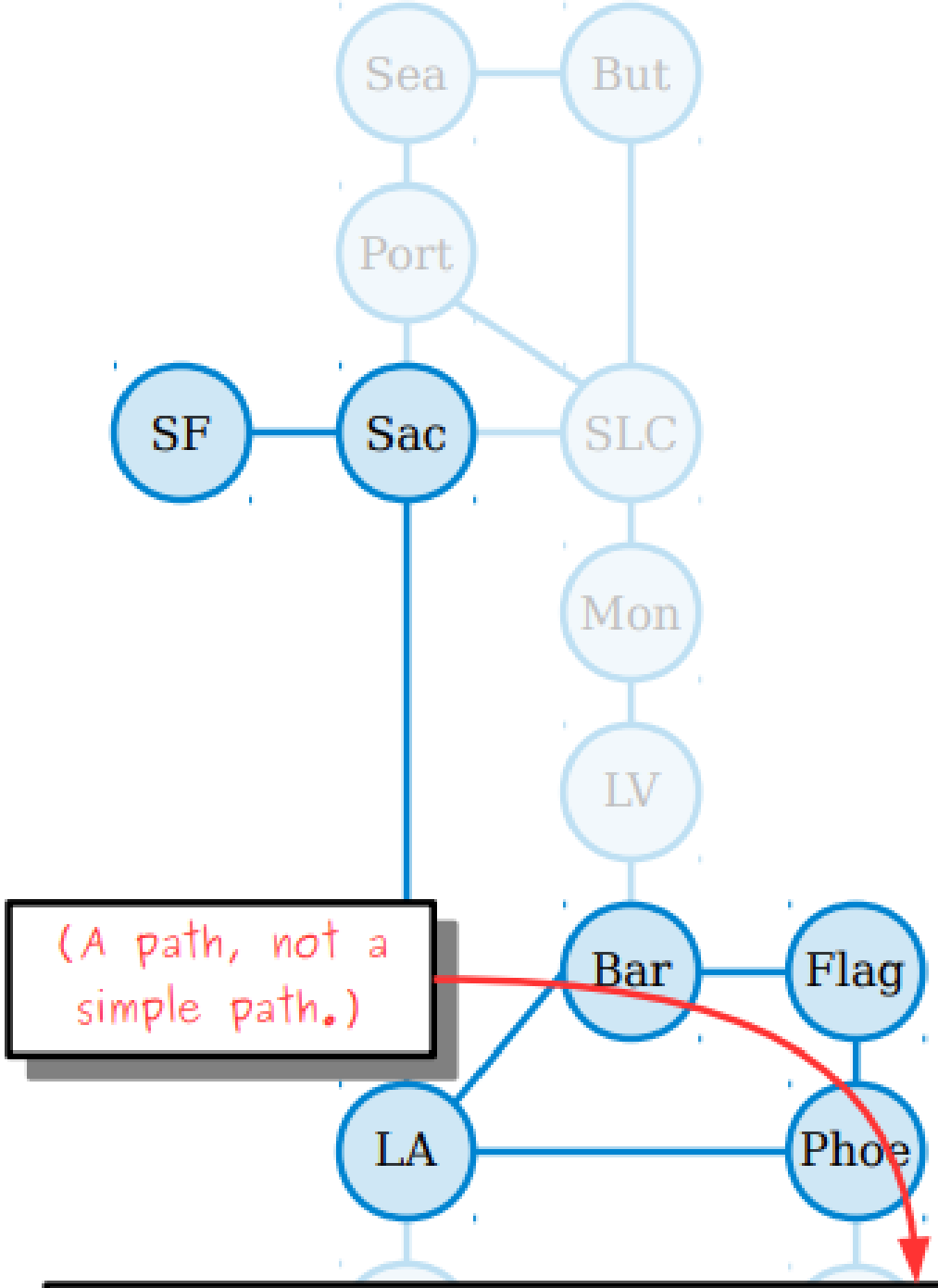
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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.

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



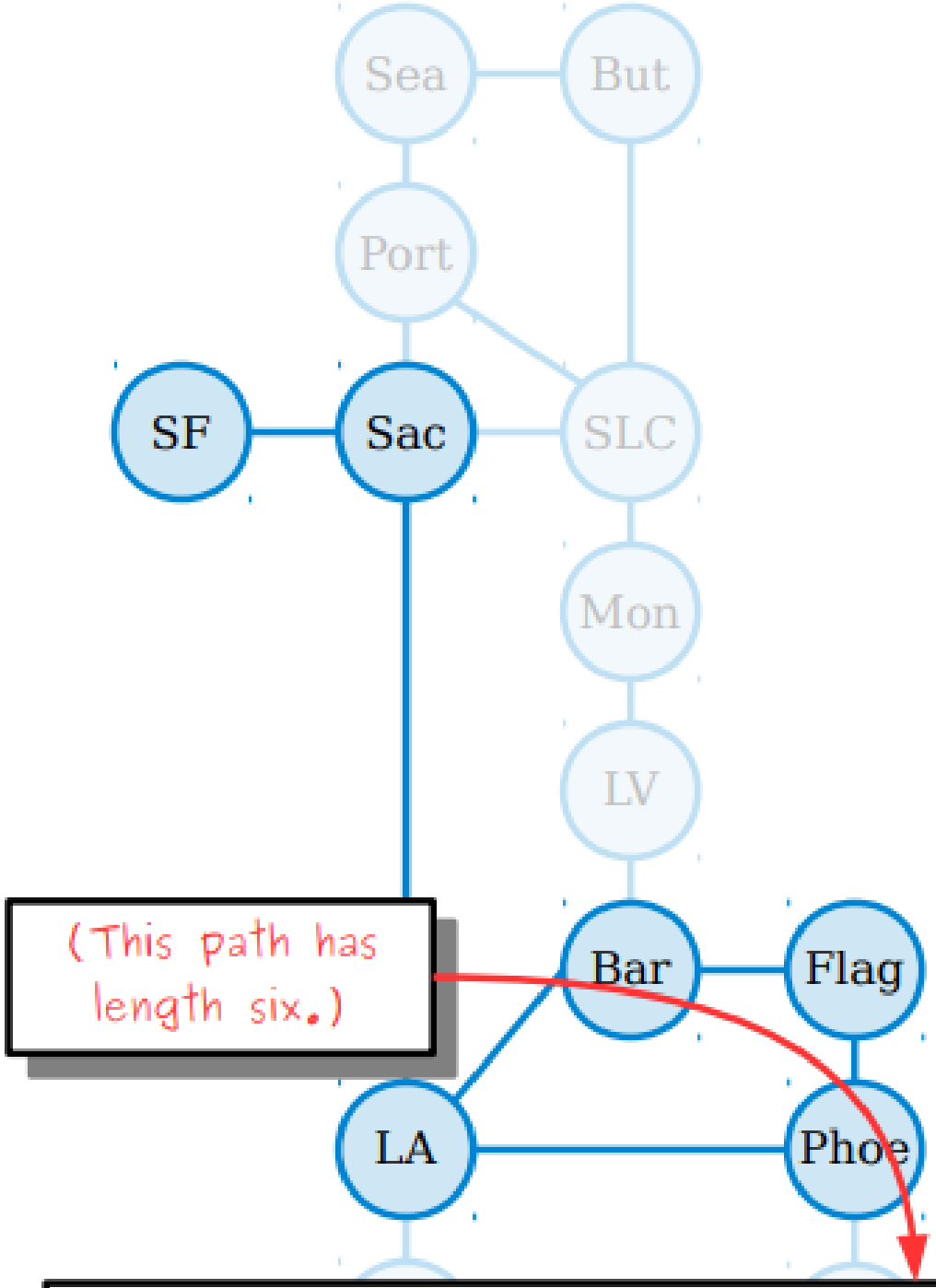
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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.

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



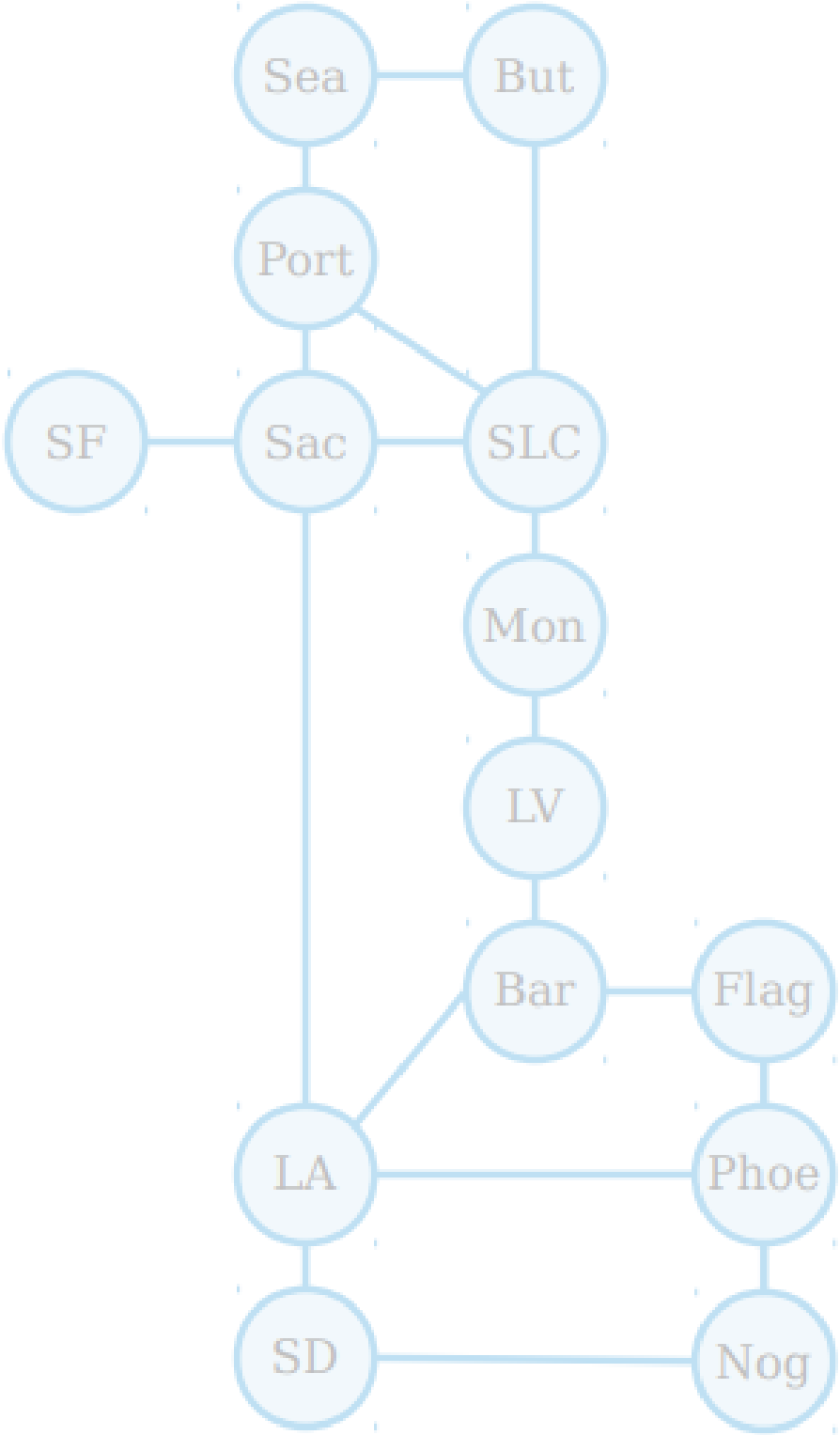
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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.

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

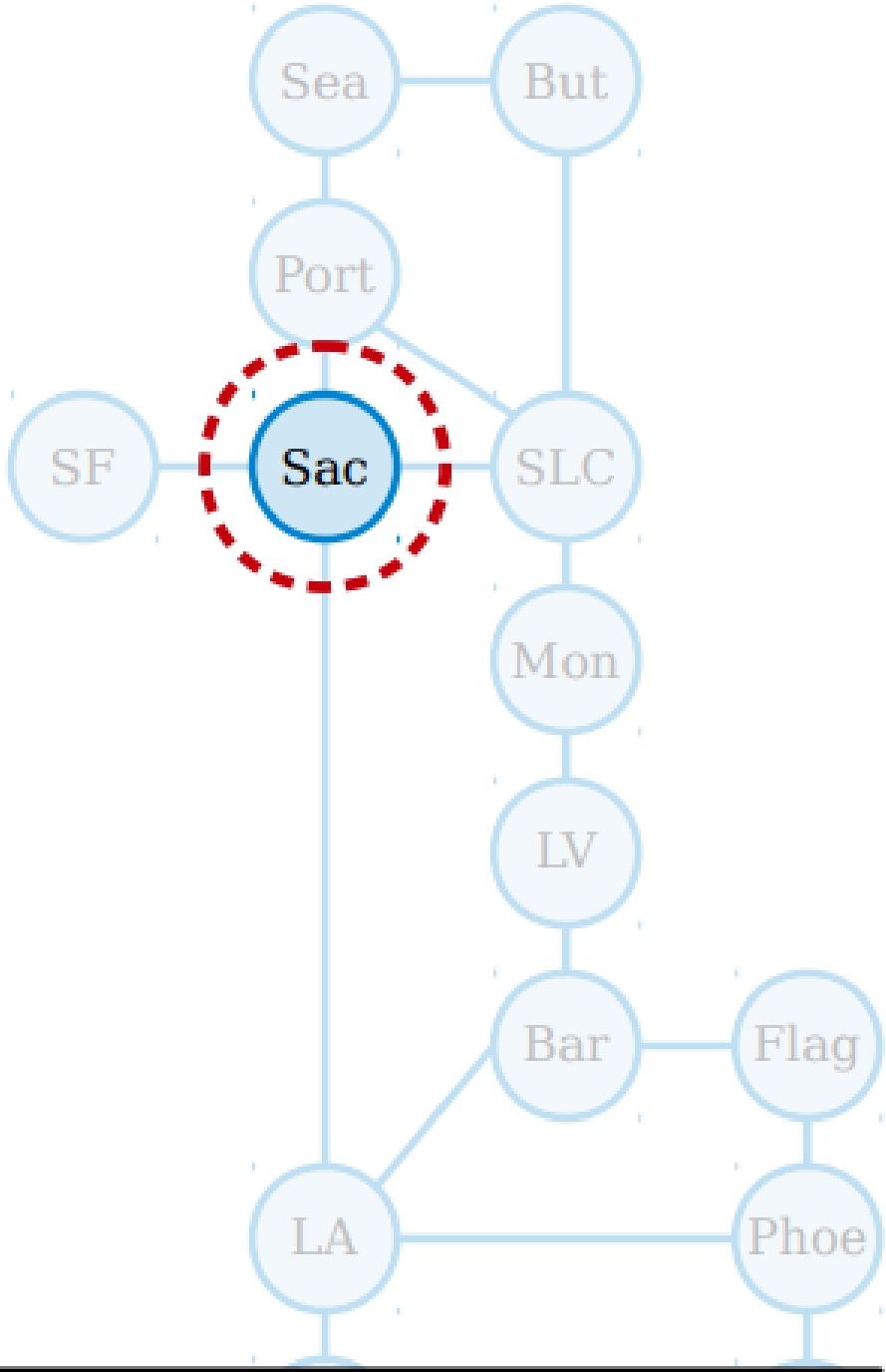


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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.



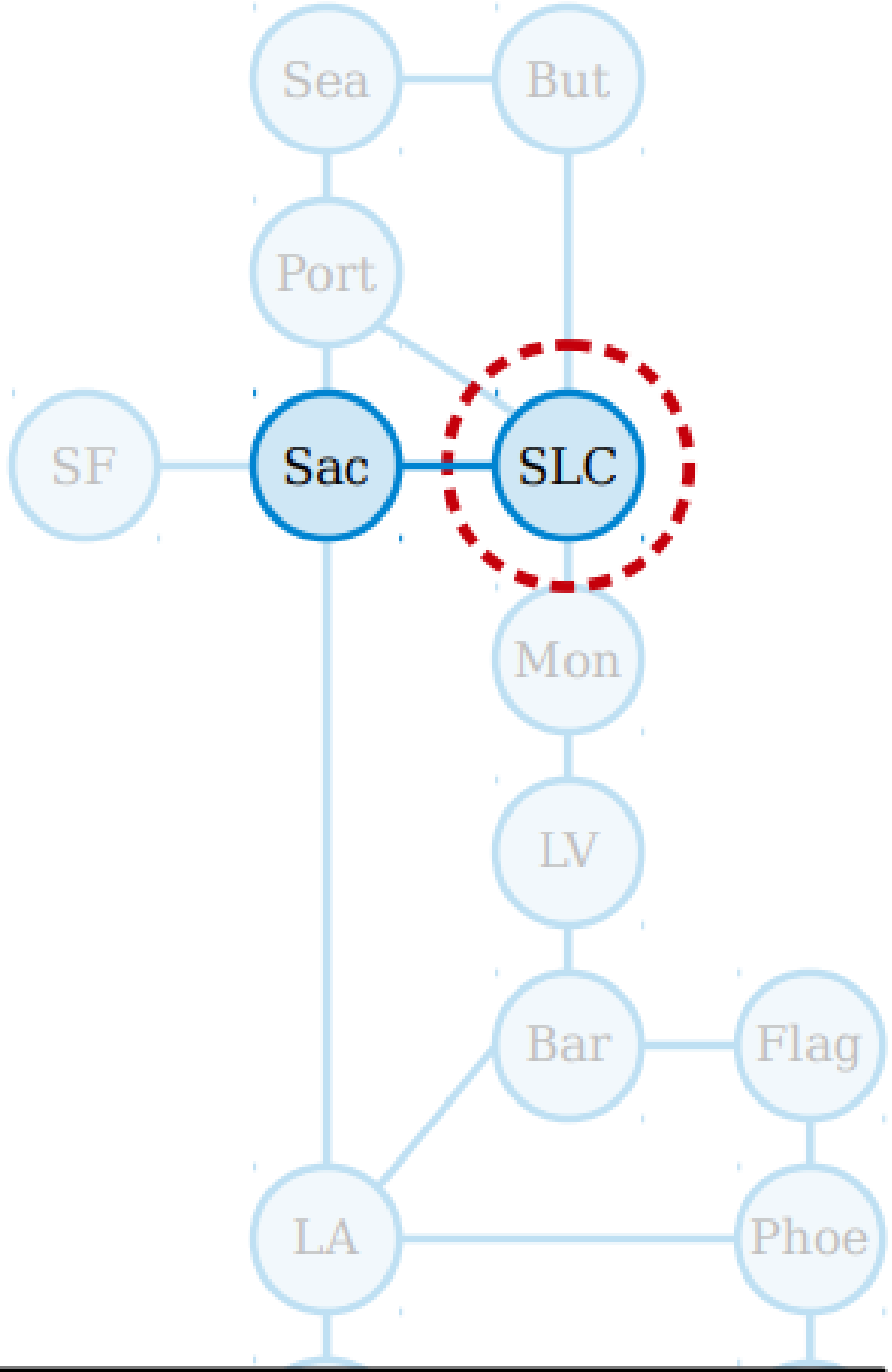
Sac

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.



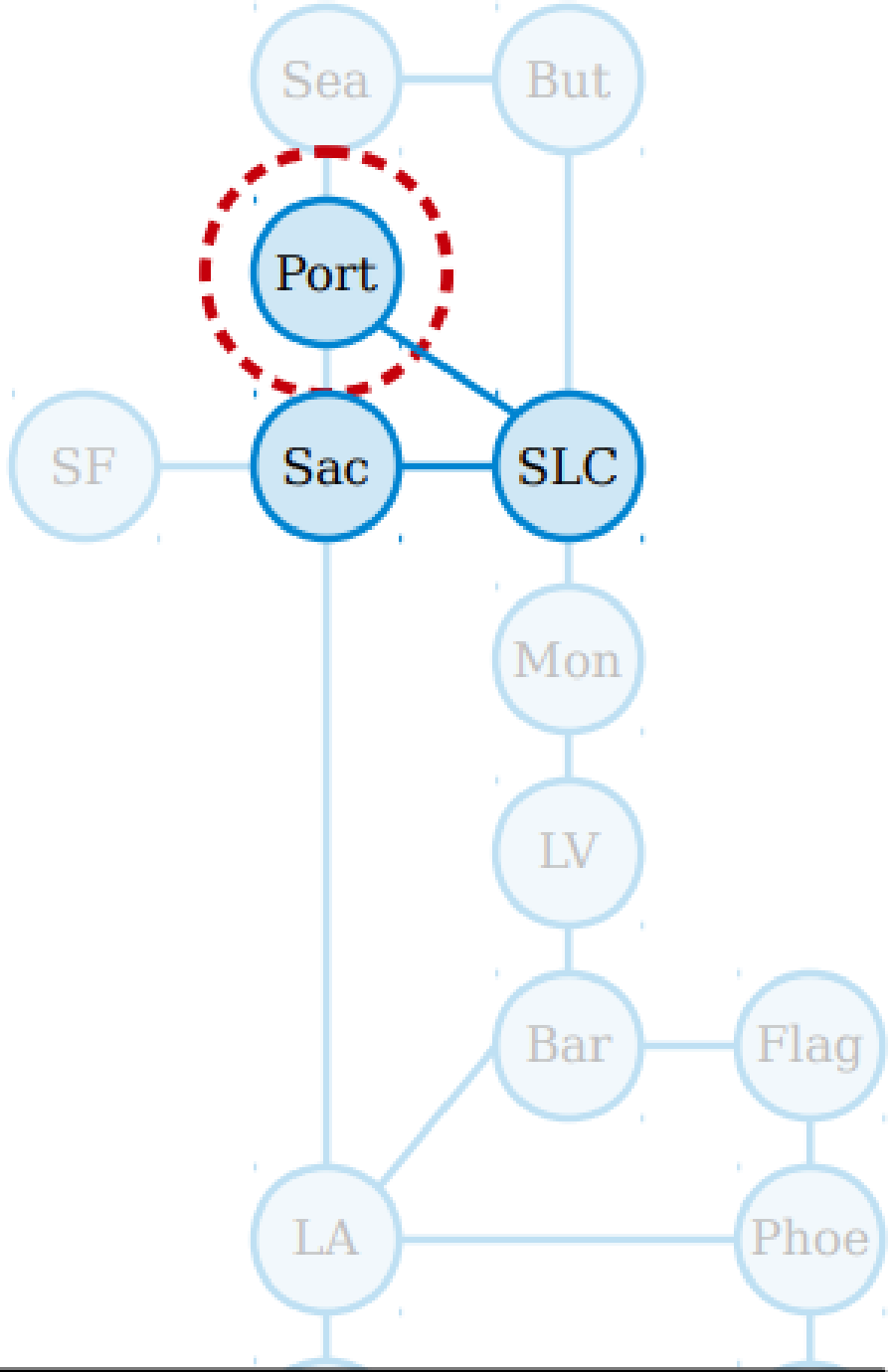
Sac, SLC

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.



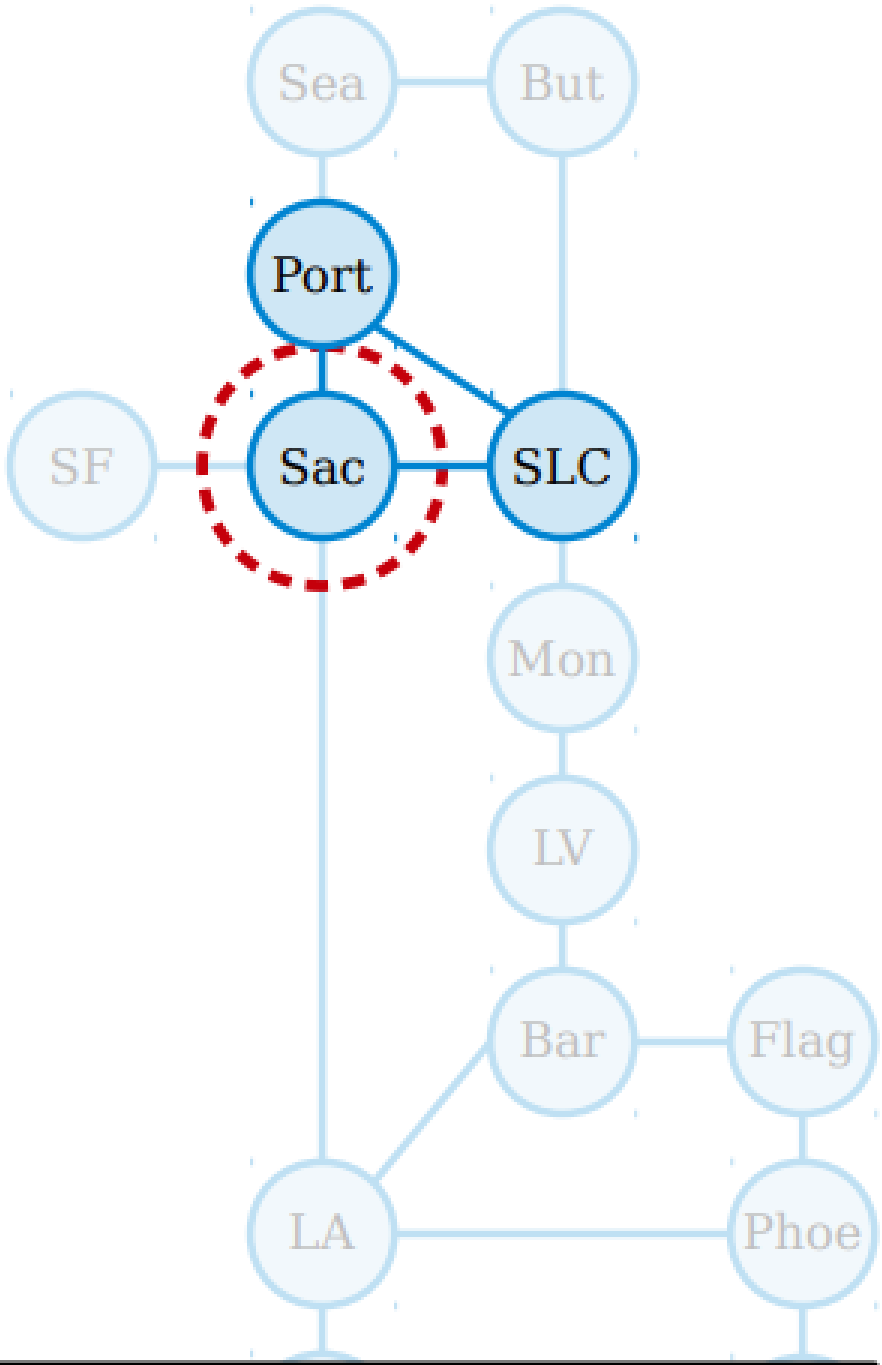
Sac, SLC, Port

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.



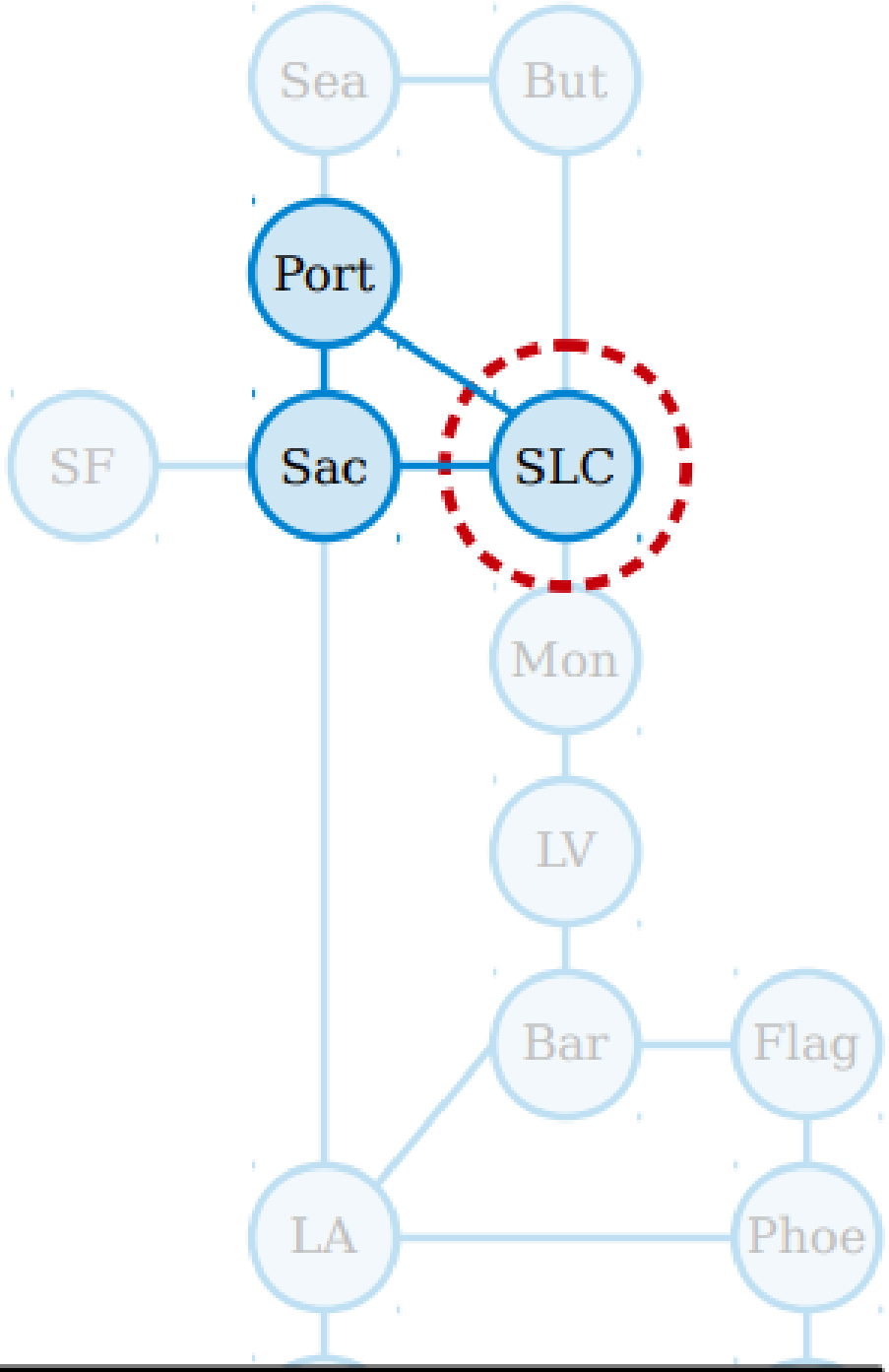
Sac, SLC, Port, Sac

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.



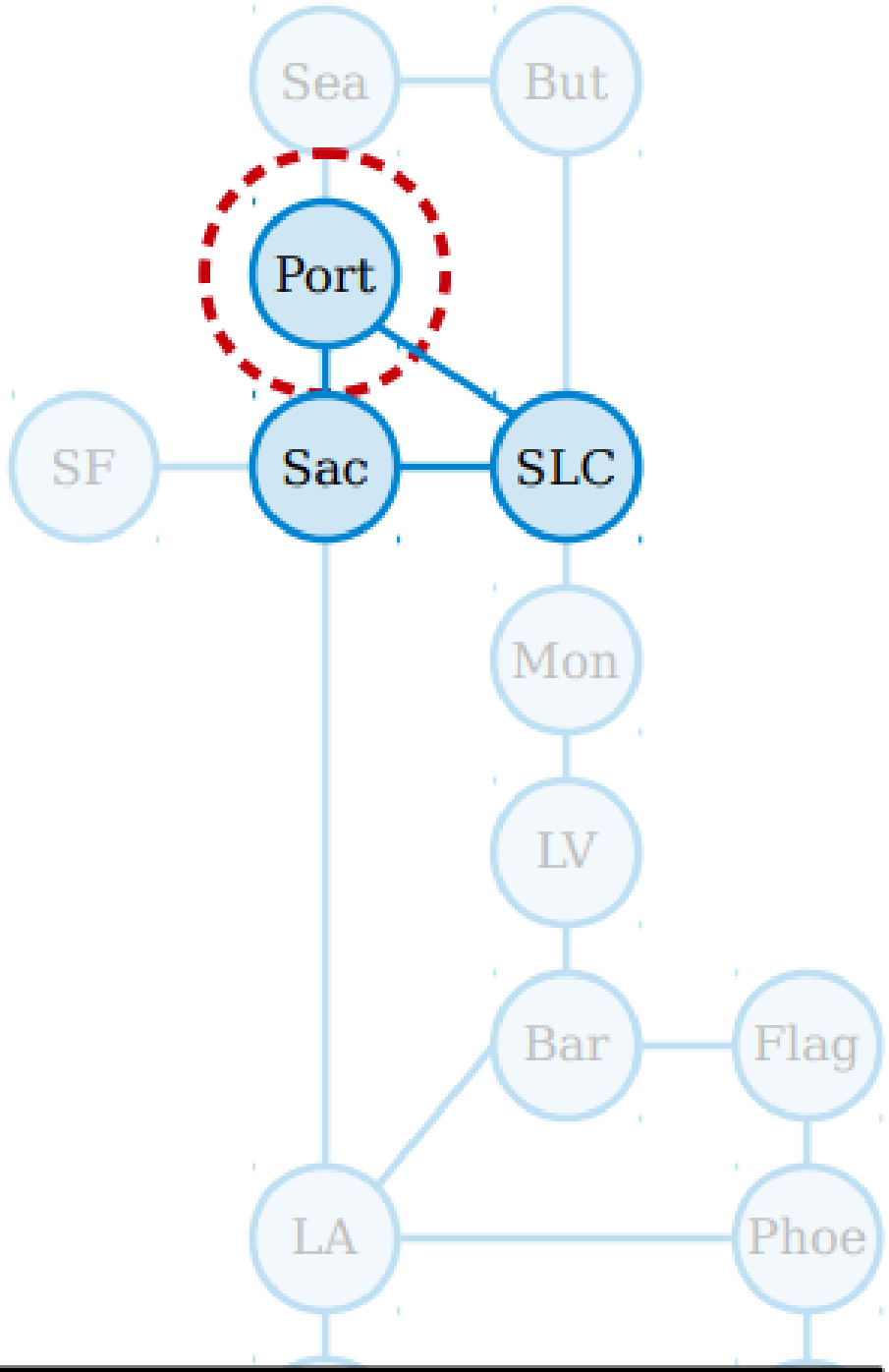
Sac, SLC, Port, Sac, SLC

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



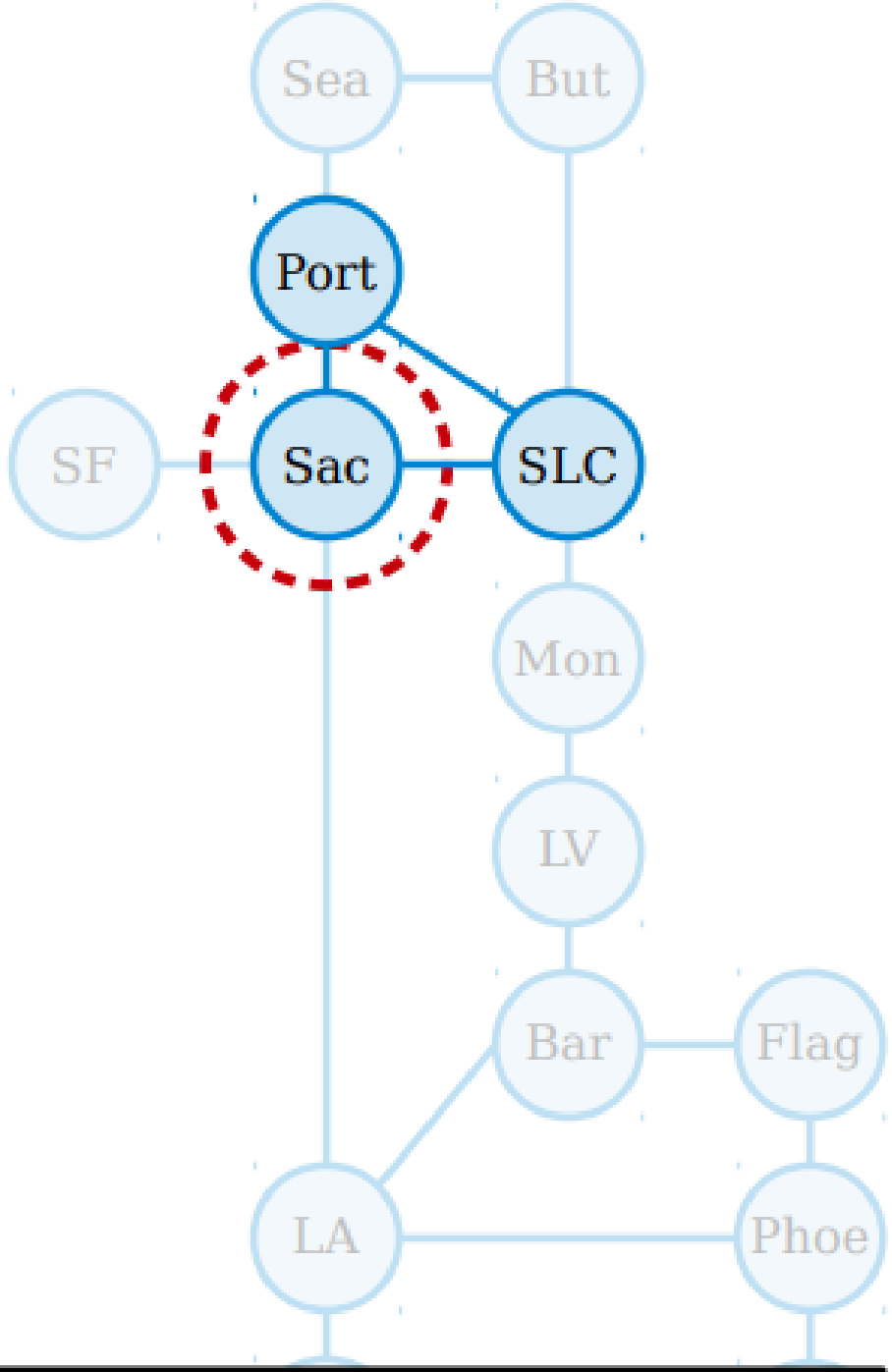
Sac, SLC, Port, Sac, SLC, Port

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



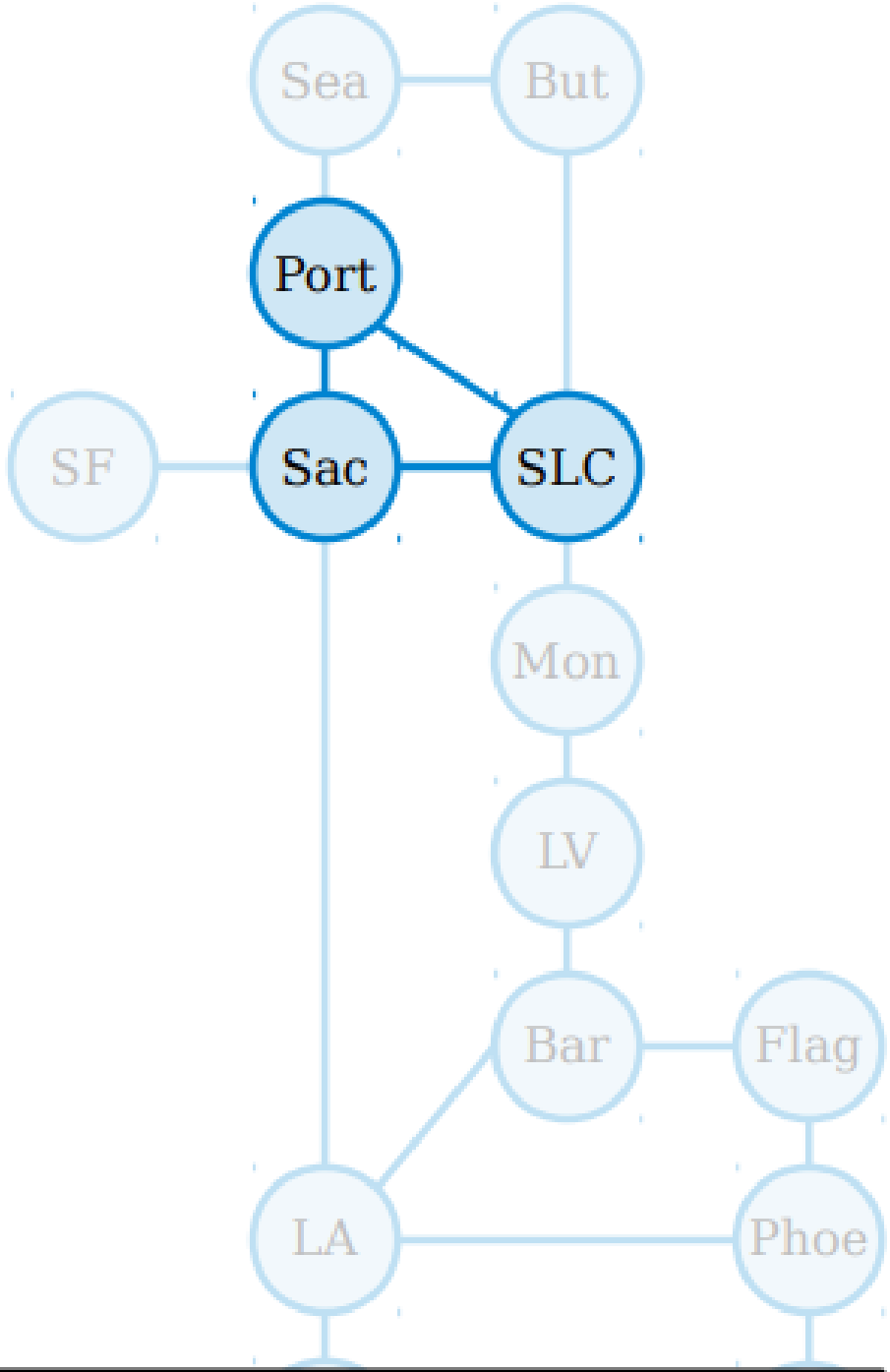
Sac, SLC, Port, Sac, SLC, Port, Sac

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.



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

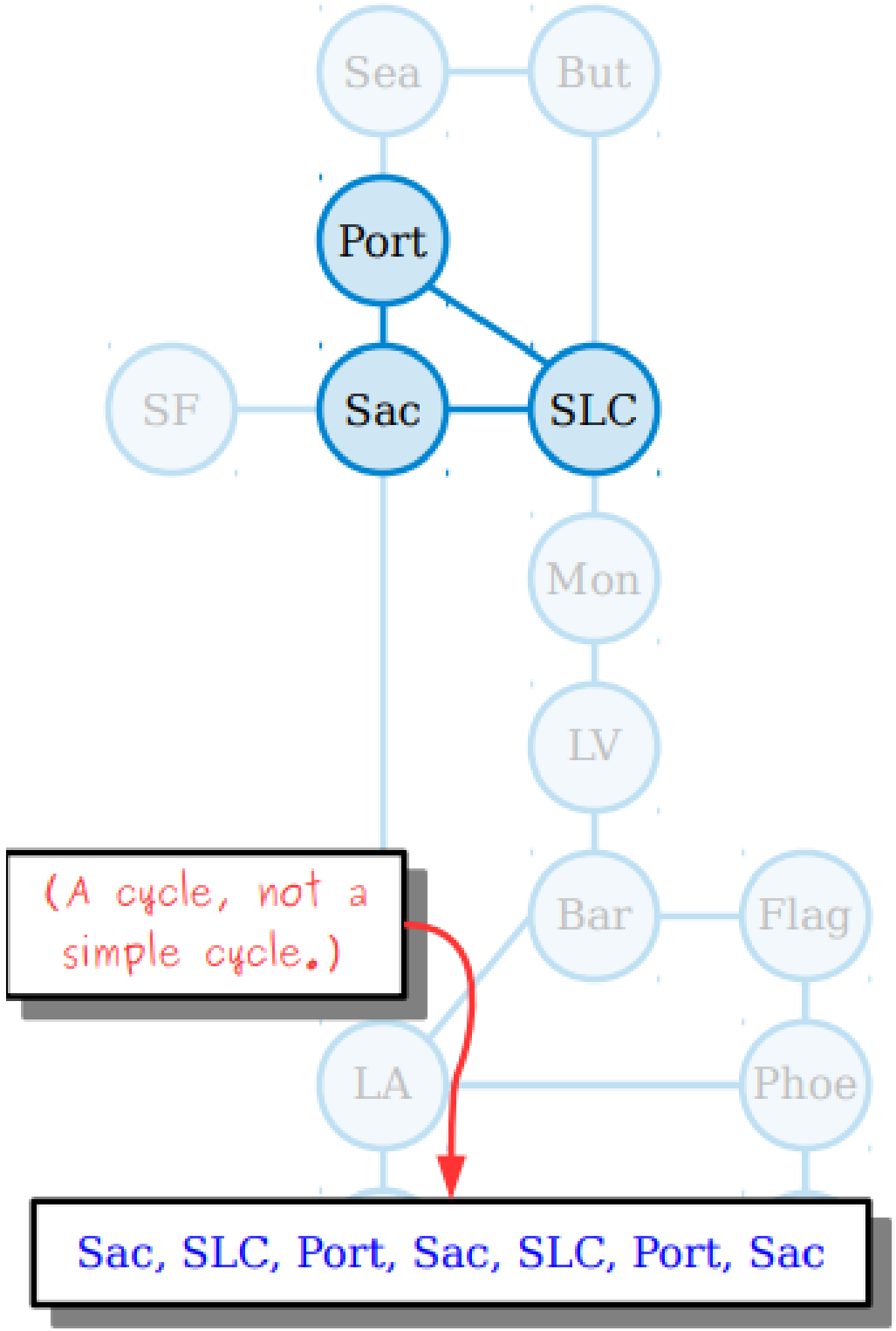
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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.



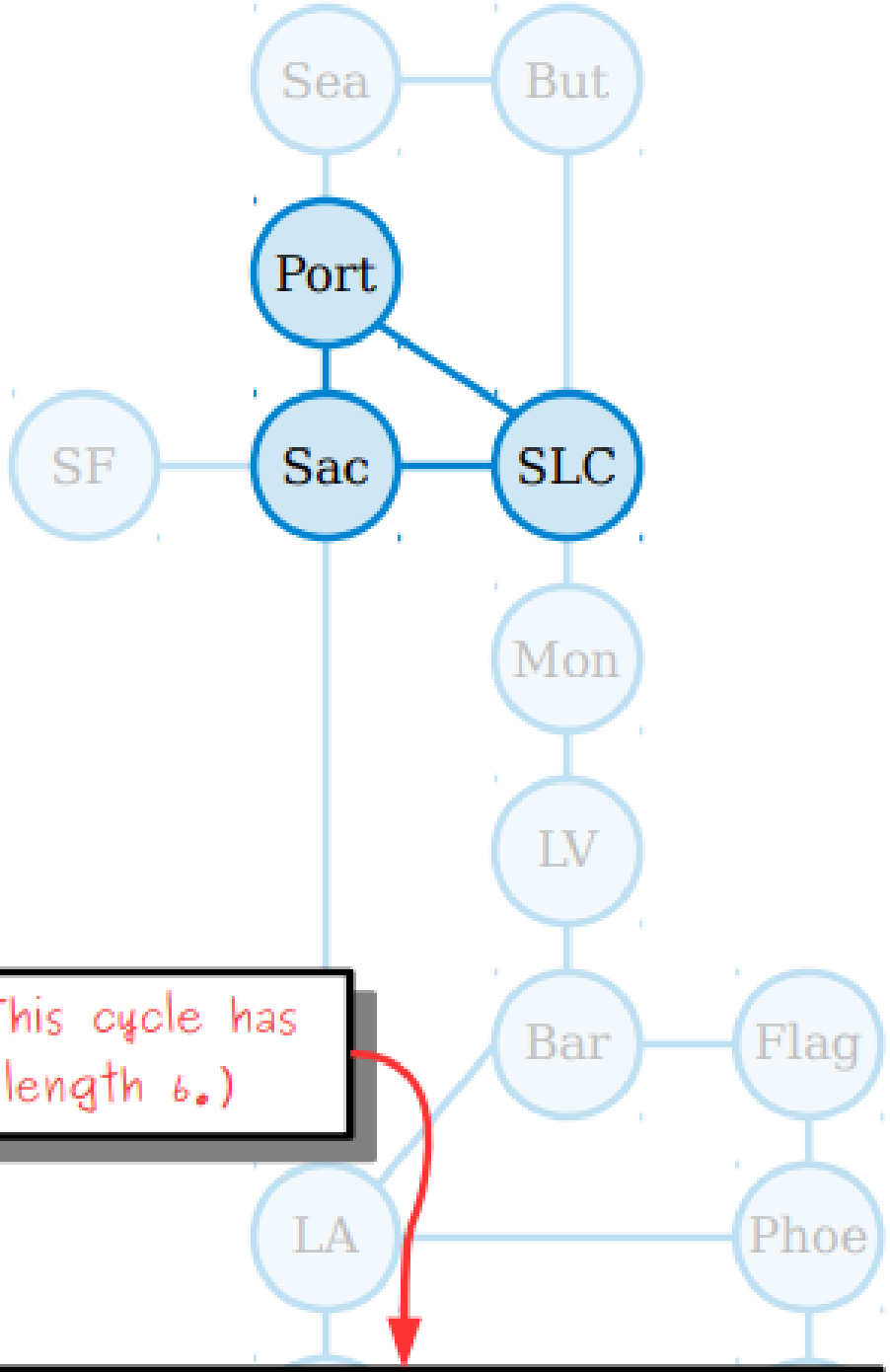
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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.



(This cycle has length 6.)

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

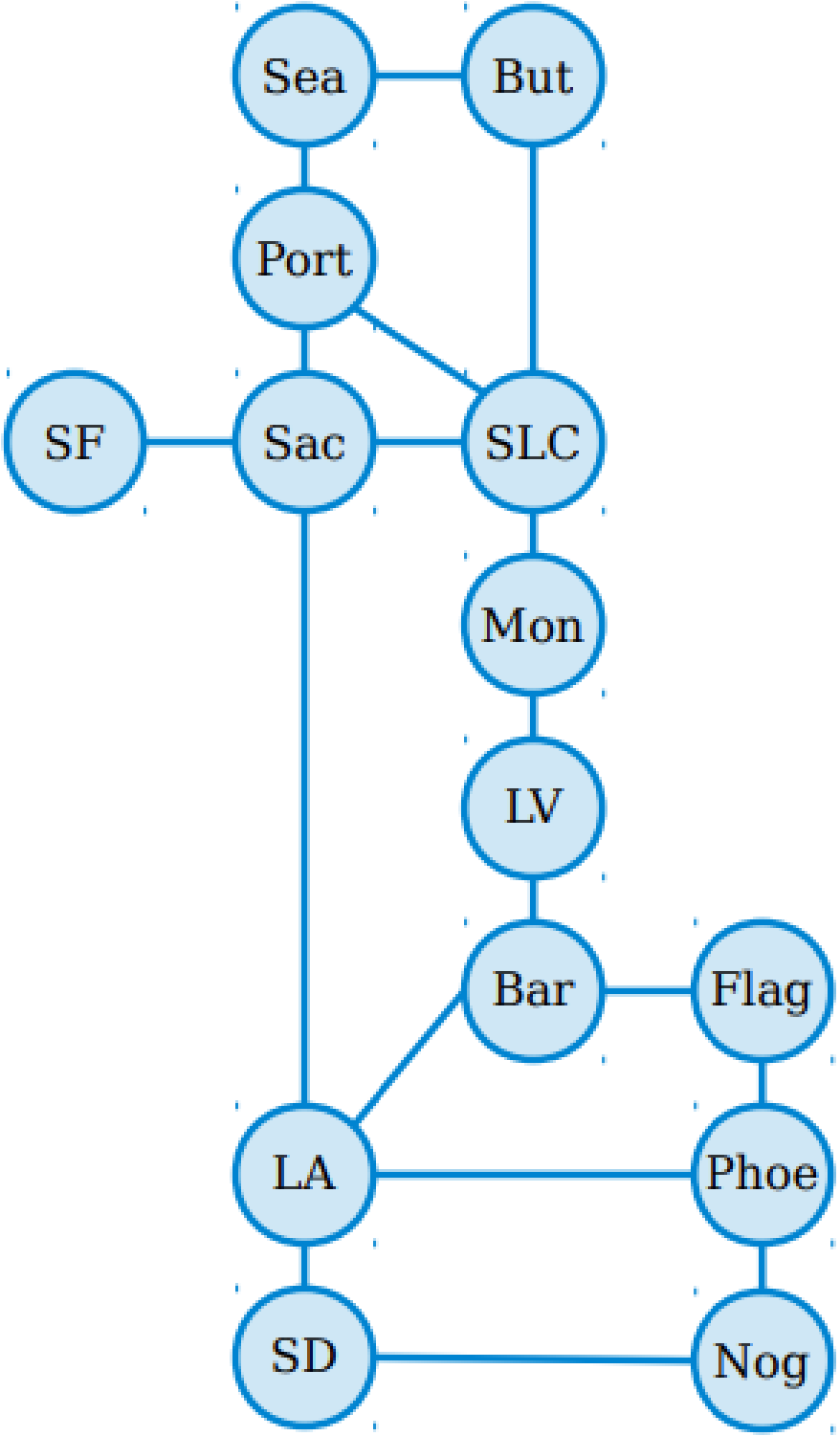
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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

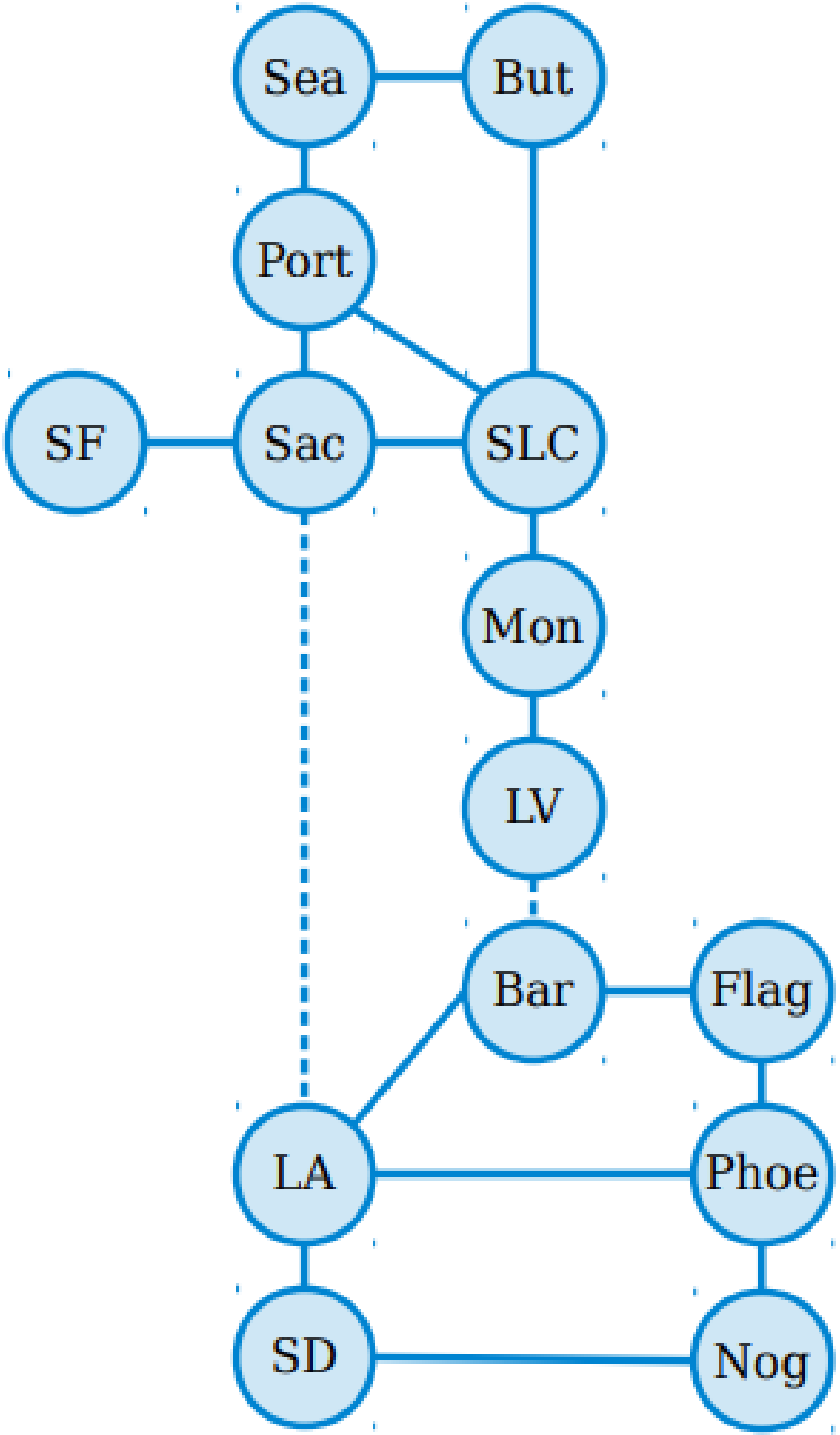
A ***cycle*** in a graph is a path from a node back to itself. (By convention, a cycle cannot have length zero.)

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

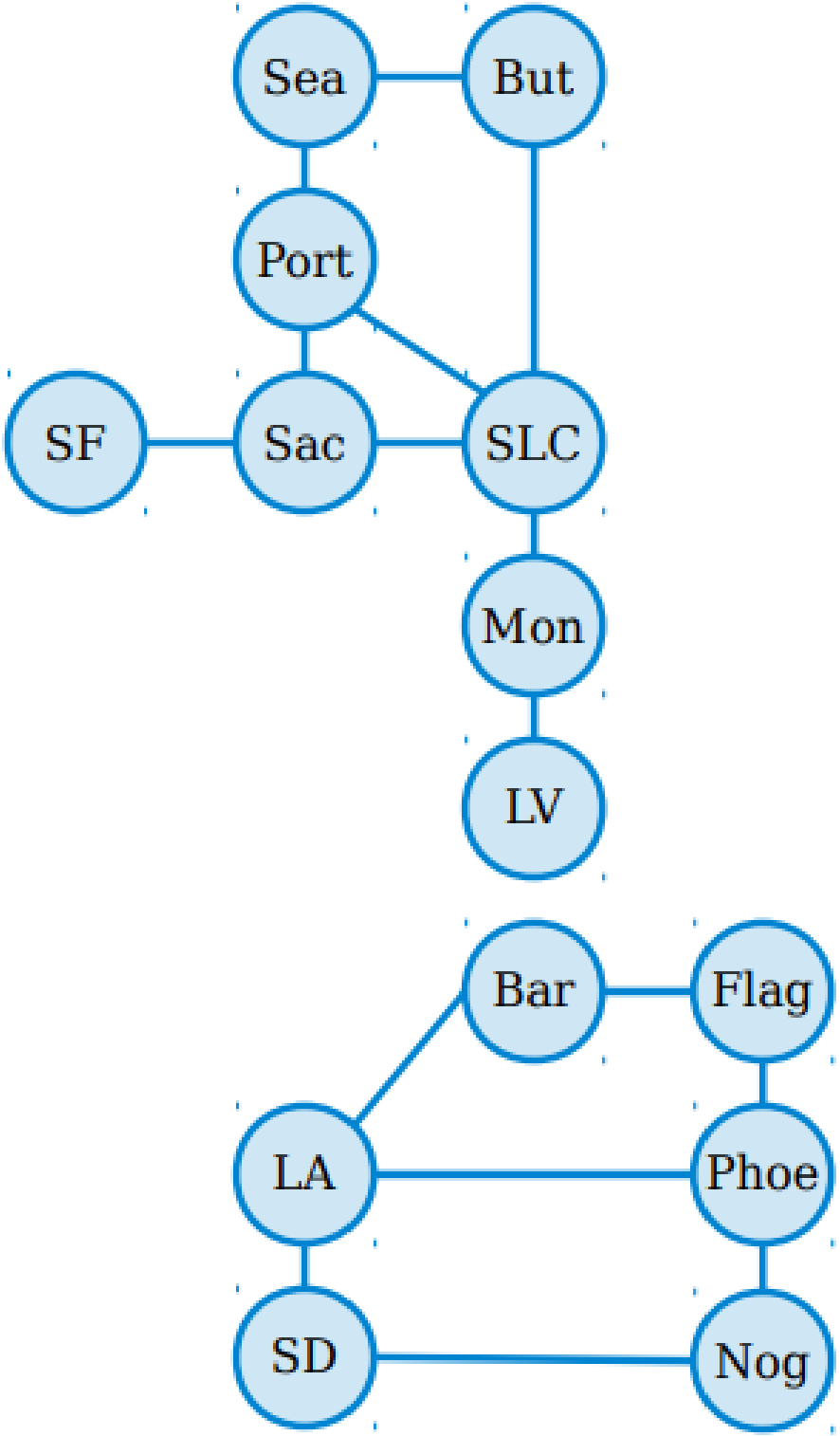
A ***simple cycle*** in a graph is cycle that does not repeat any nodes or edges except the first/last node.



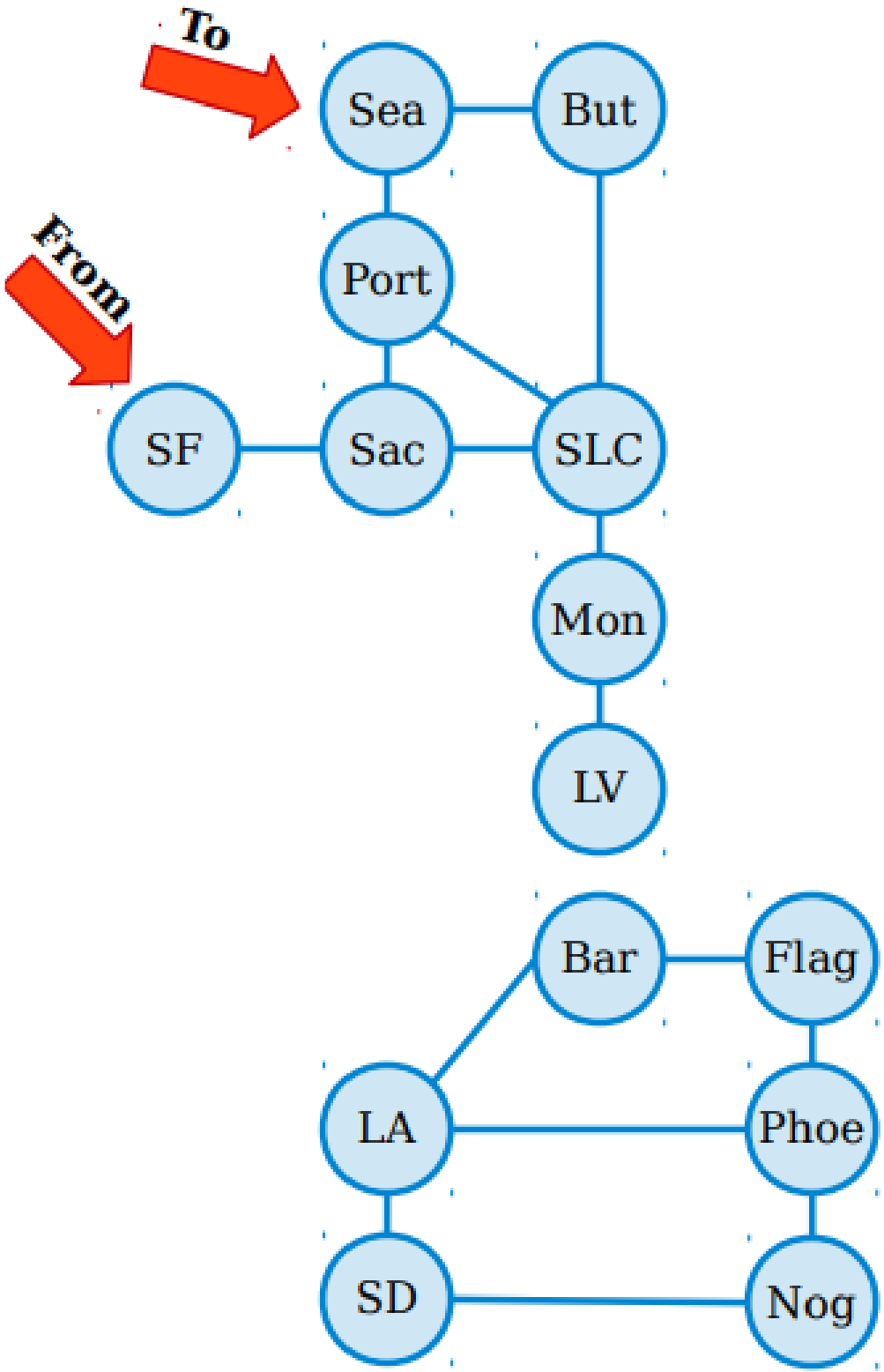
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.



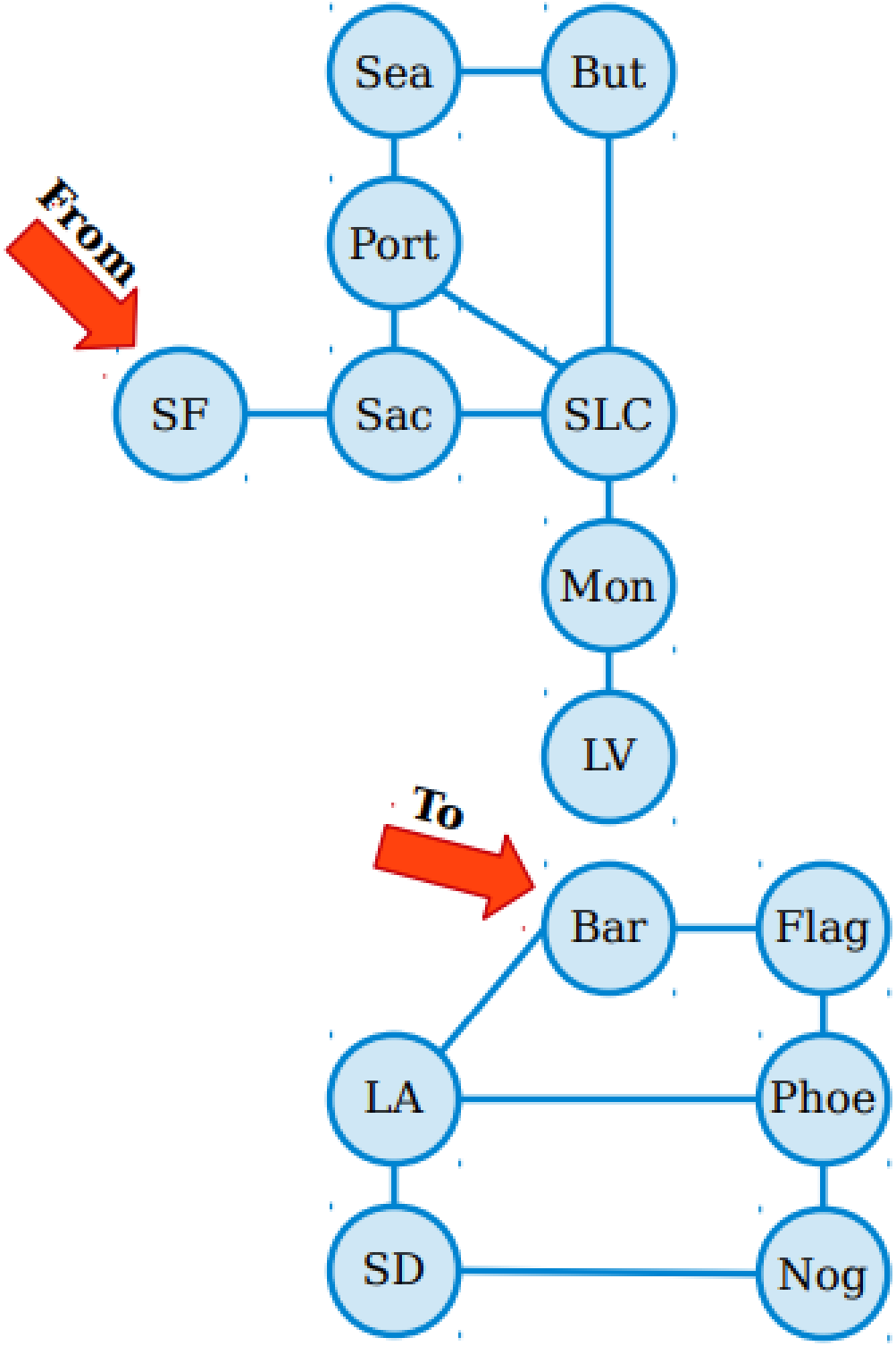
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.



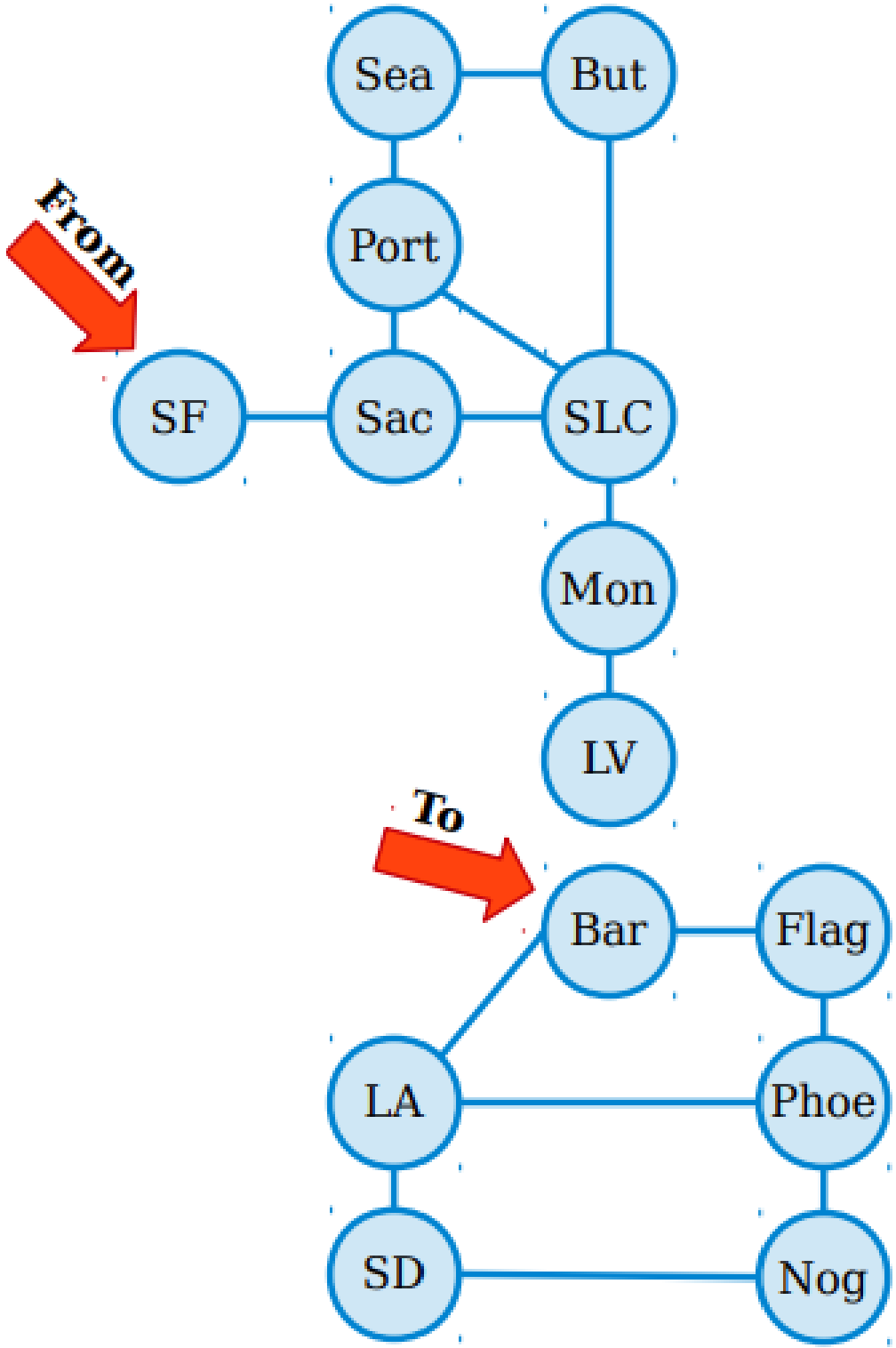
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.



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.

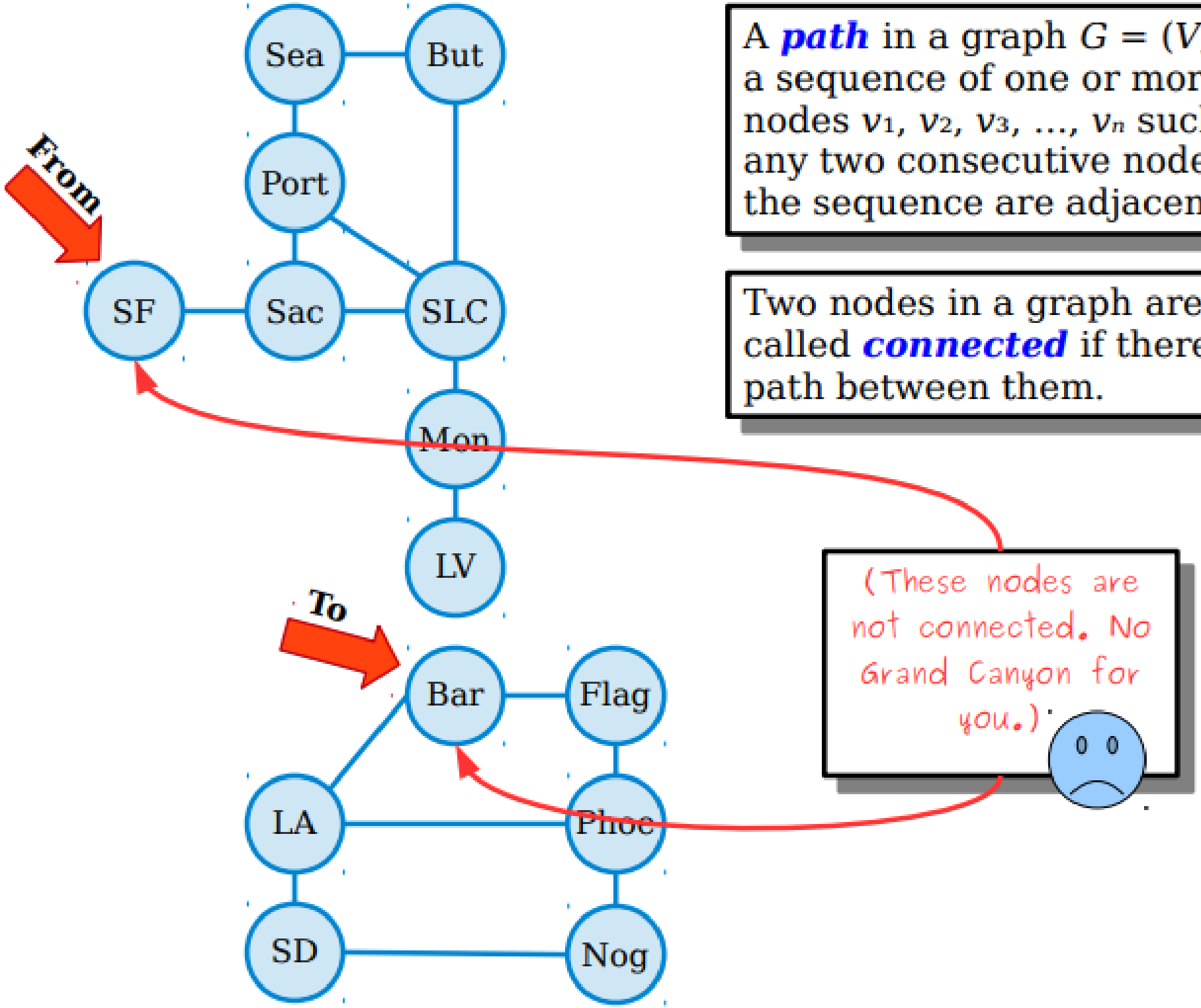


A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.



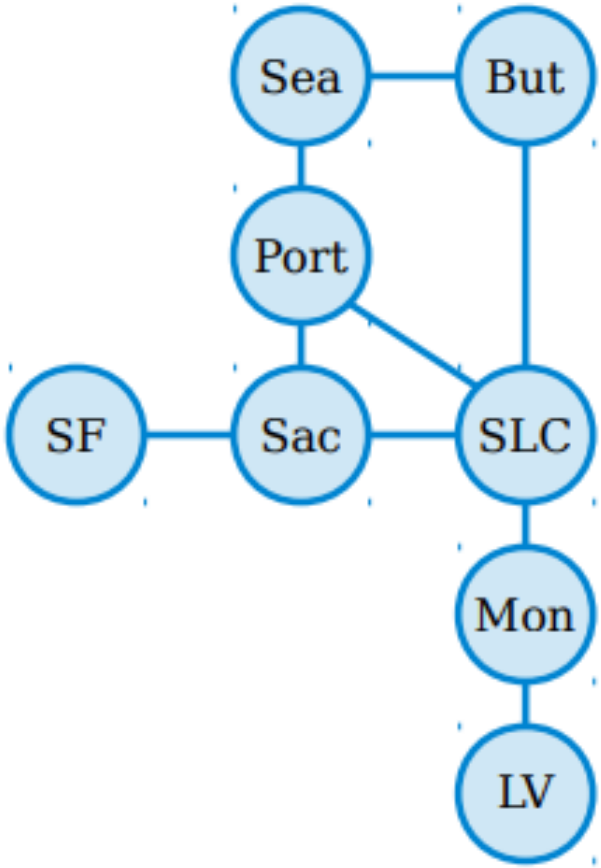
A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

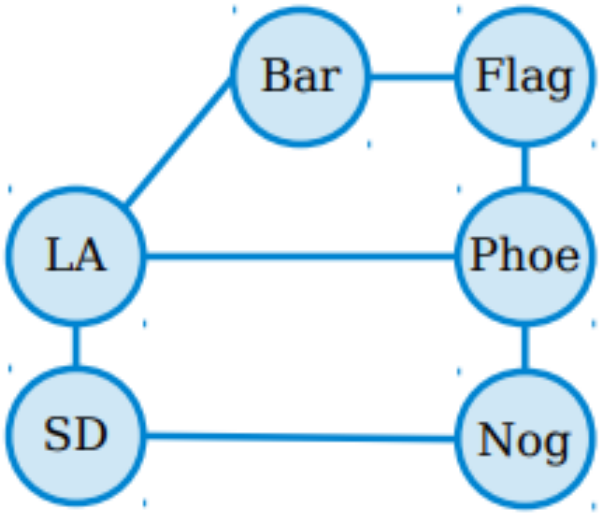
Two nodes in a graph are called ***connected*** if there is a path between them.



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.

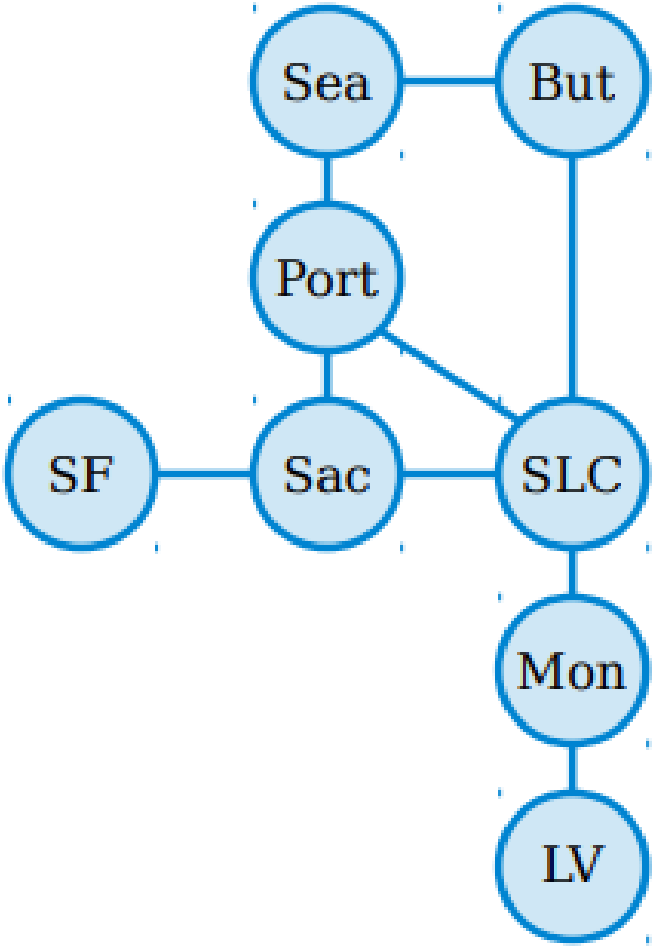




A ***path*** in a graph $G = (V, E)$ is a sequence of one or more nodes $v_1, v_2, v_3, \ldots, v_n$ such that any two consecutive nodes in the sequence are adjacent.

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

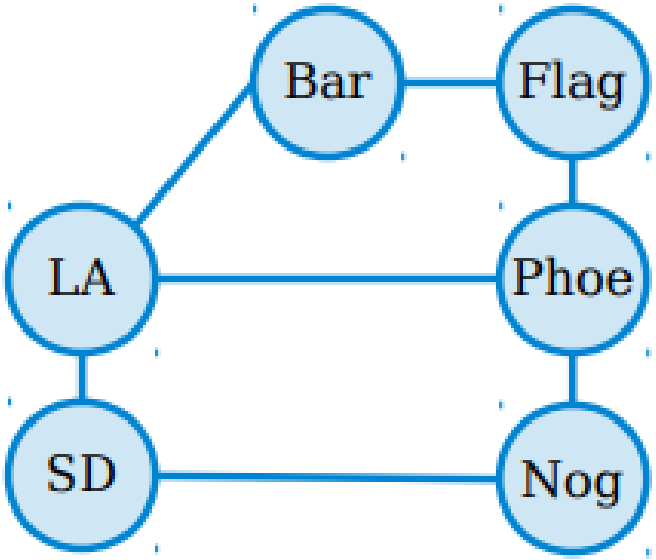
A graph G as a whole is called ***connected*** if all pairs of nodes in G are connected.



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

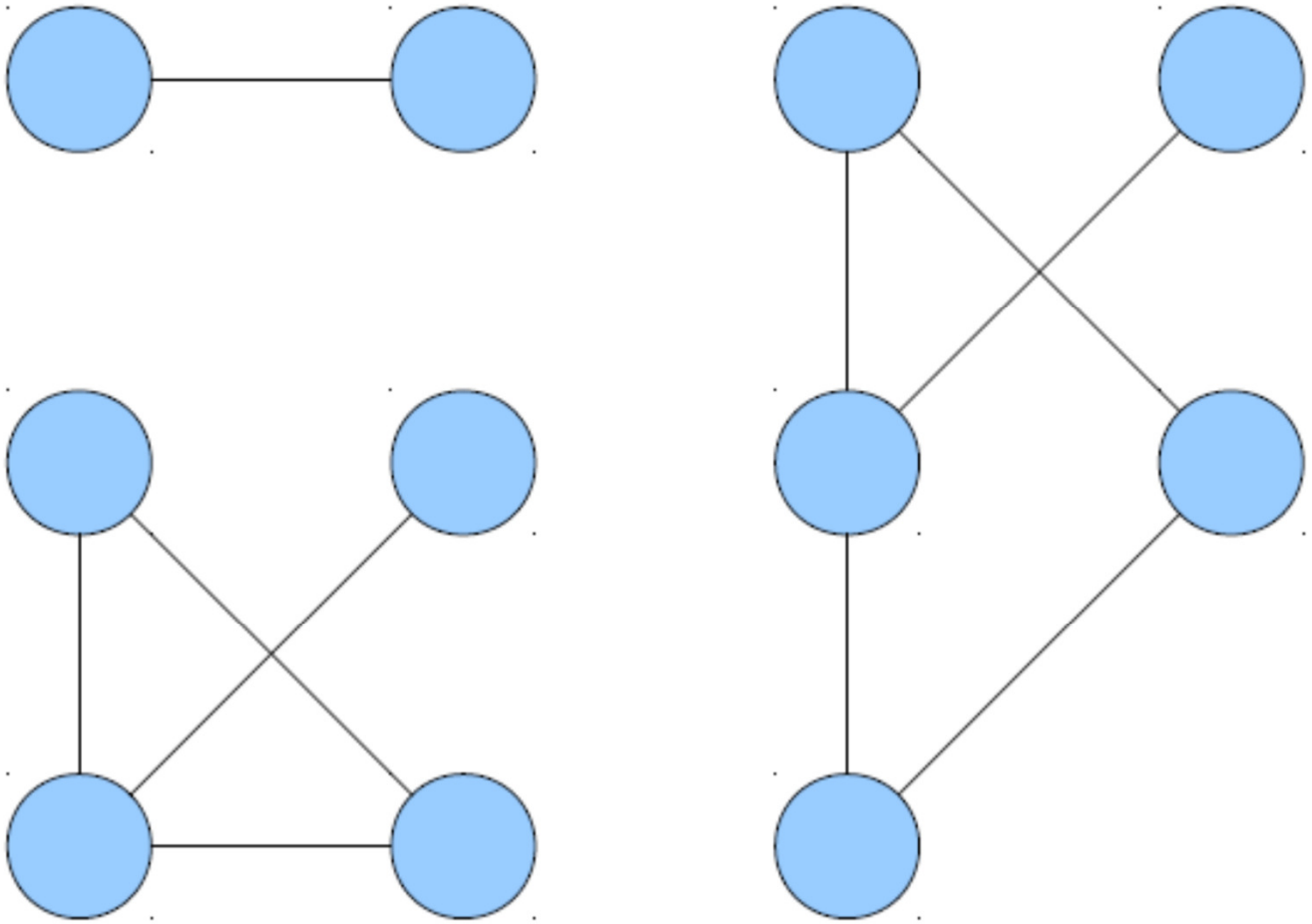
Two nodes in a graph are called ***connected*** if there is a path between them.

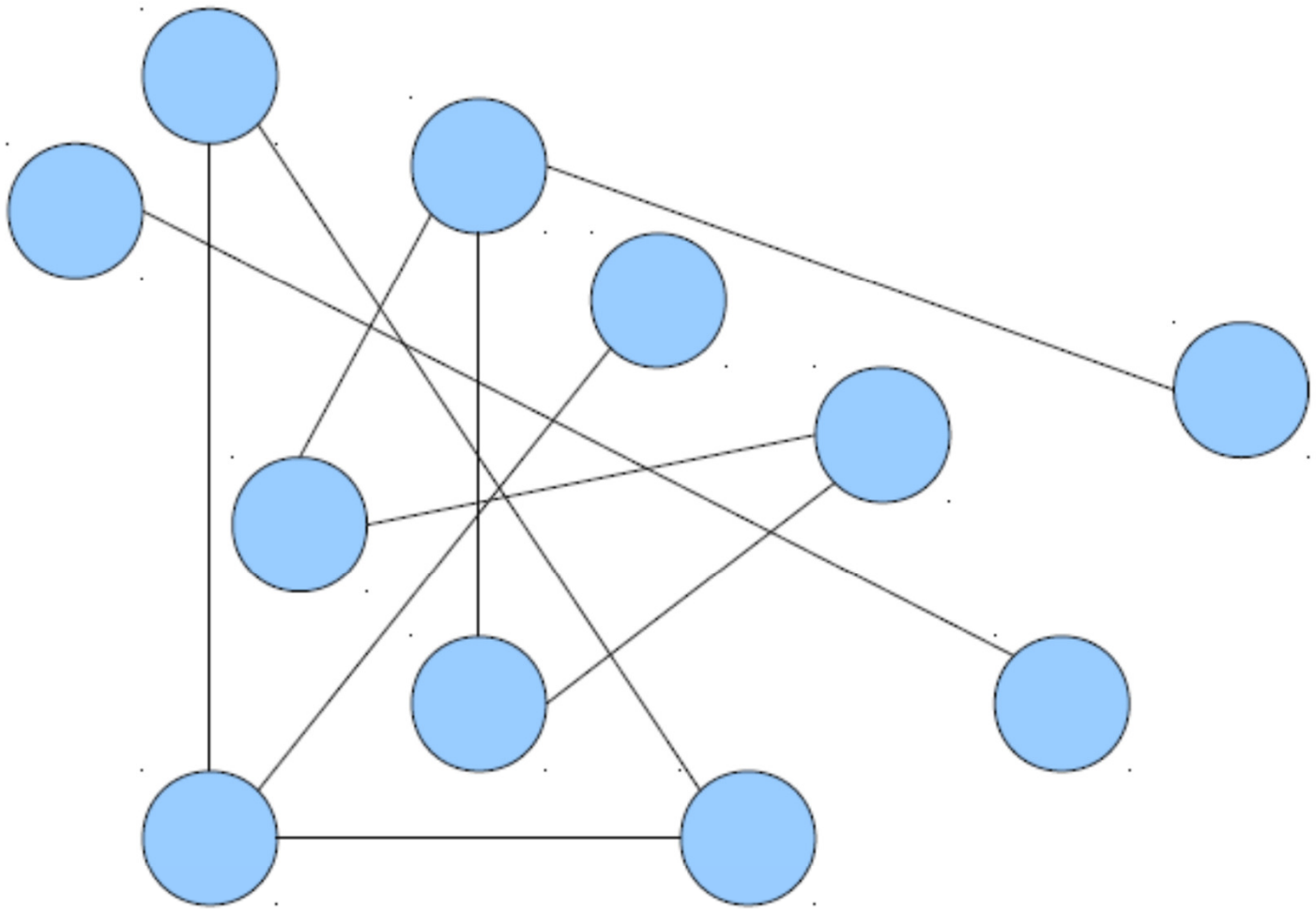
A graph G as a whole is called ***connected*** if all pairs of nodes in G are connected.

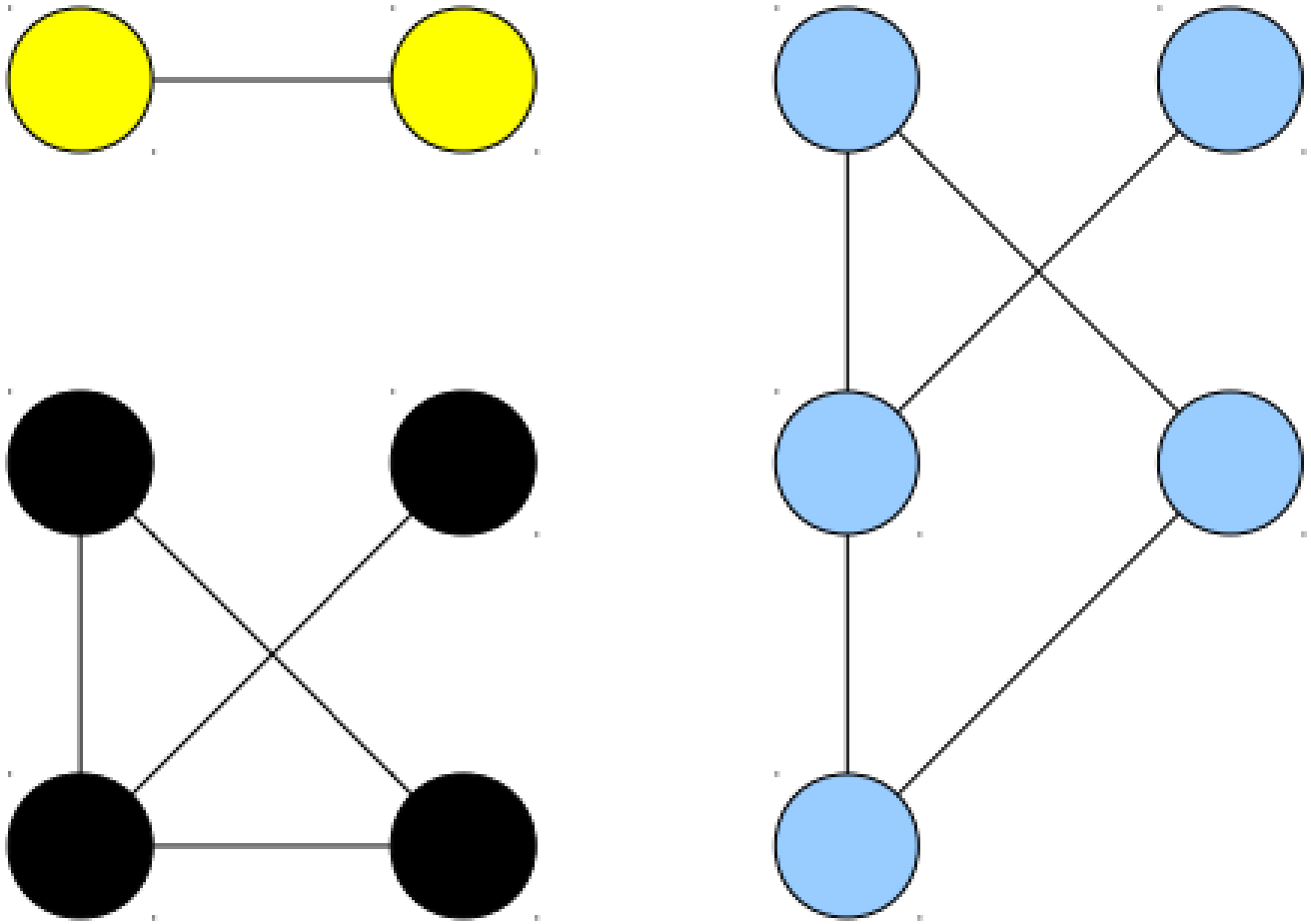


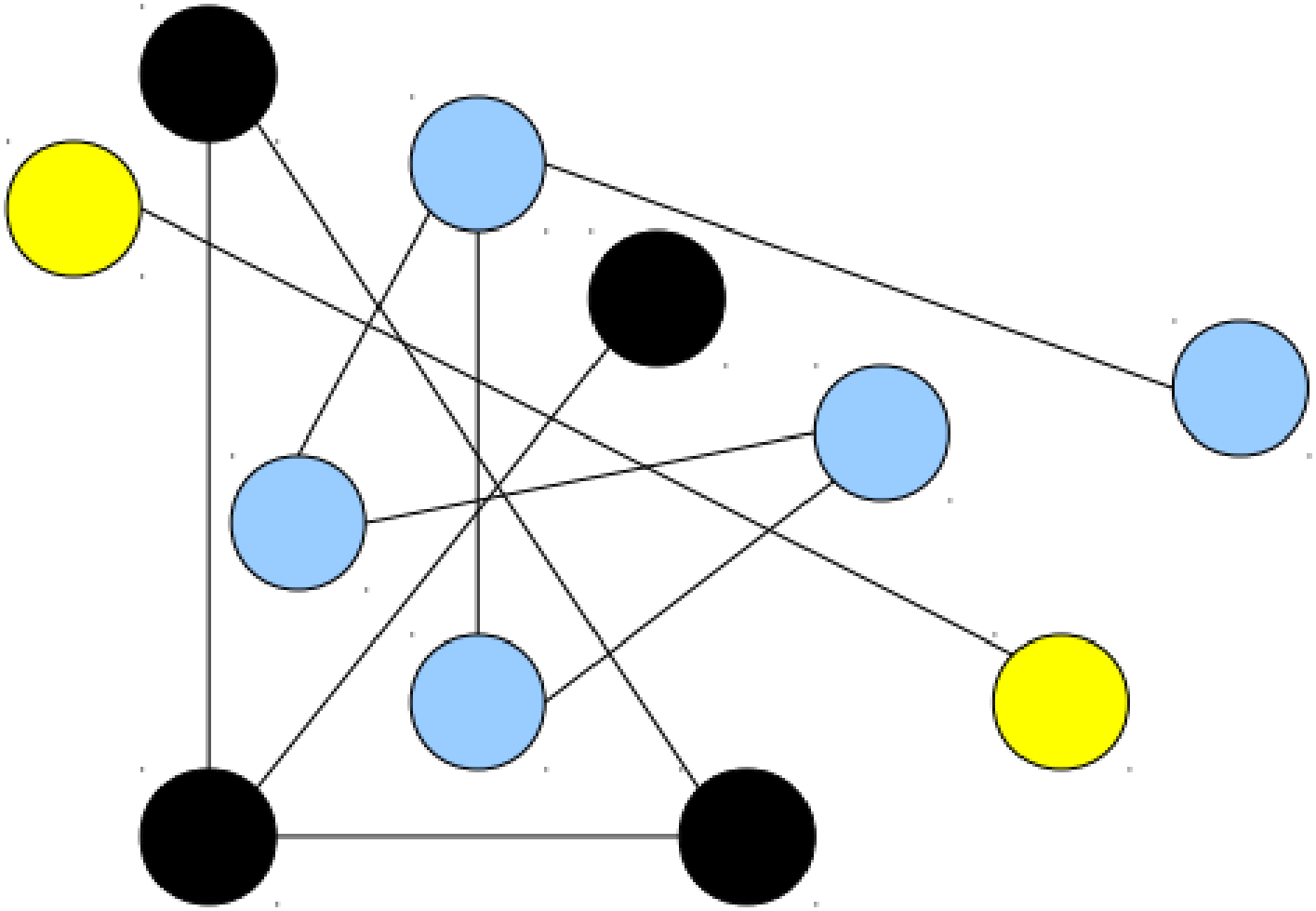
(This graph is not connected.)

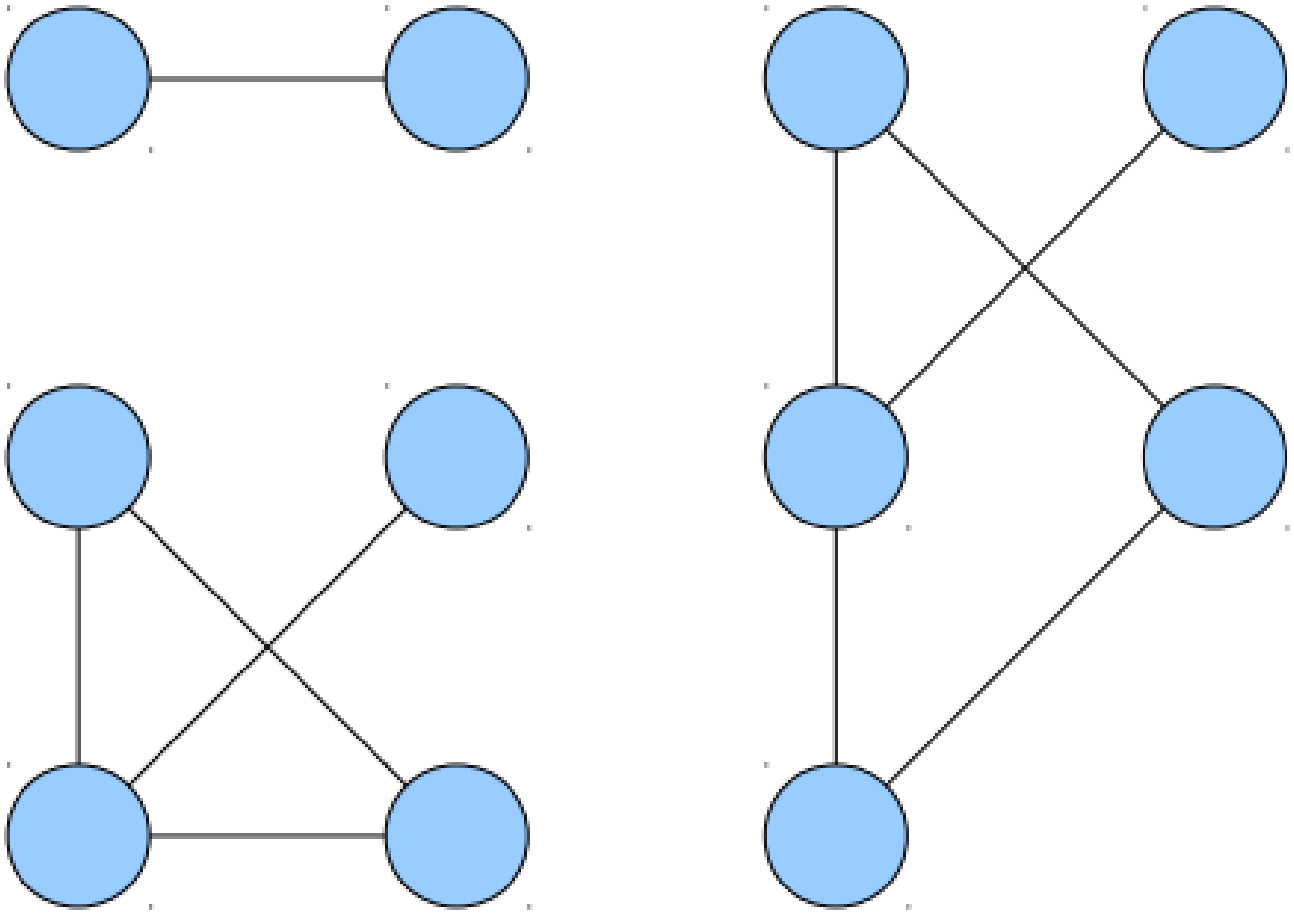
Connected Components

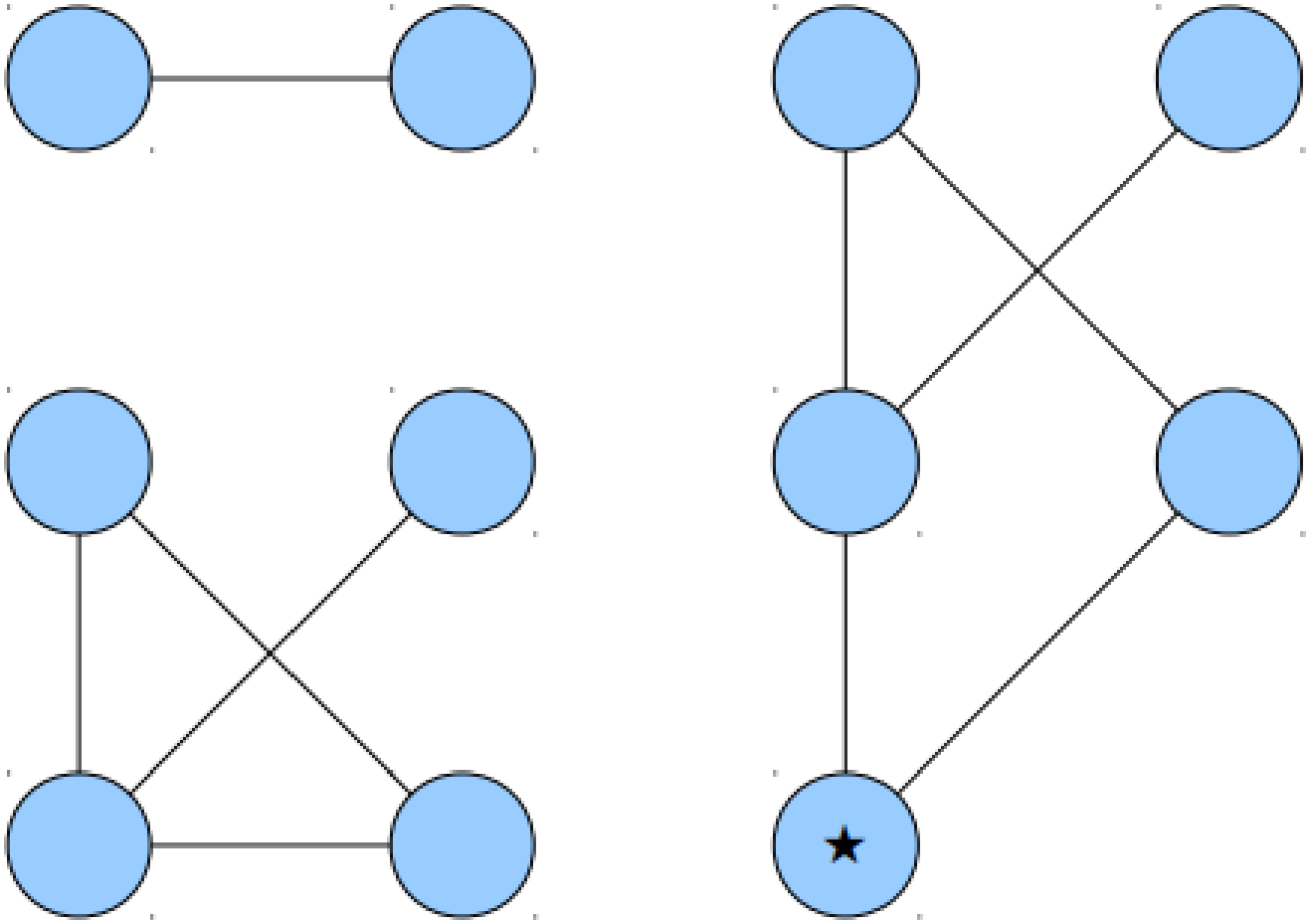


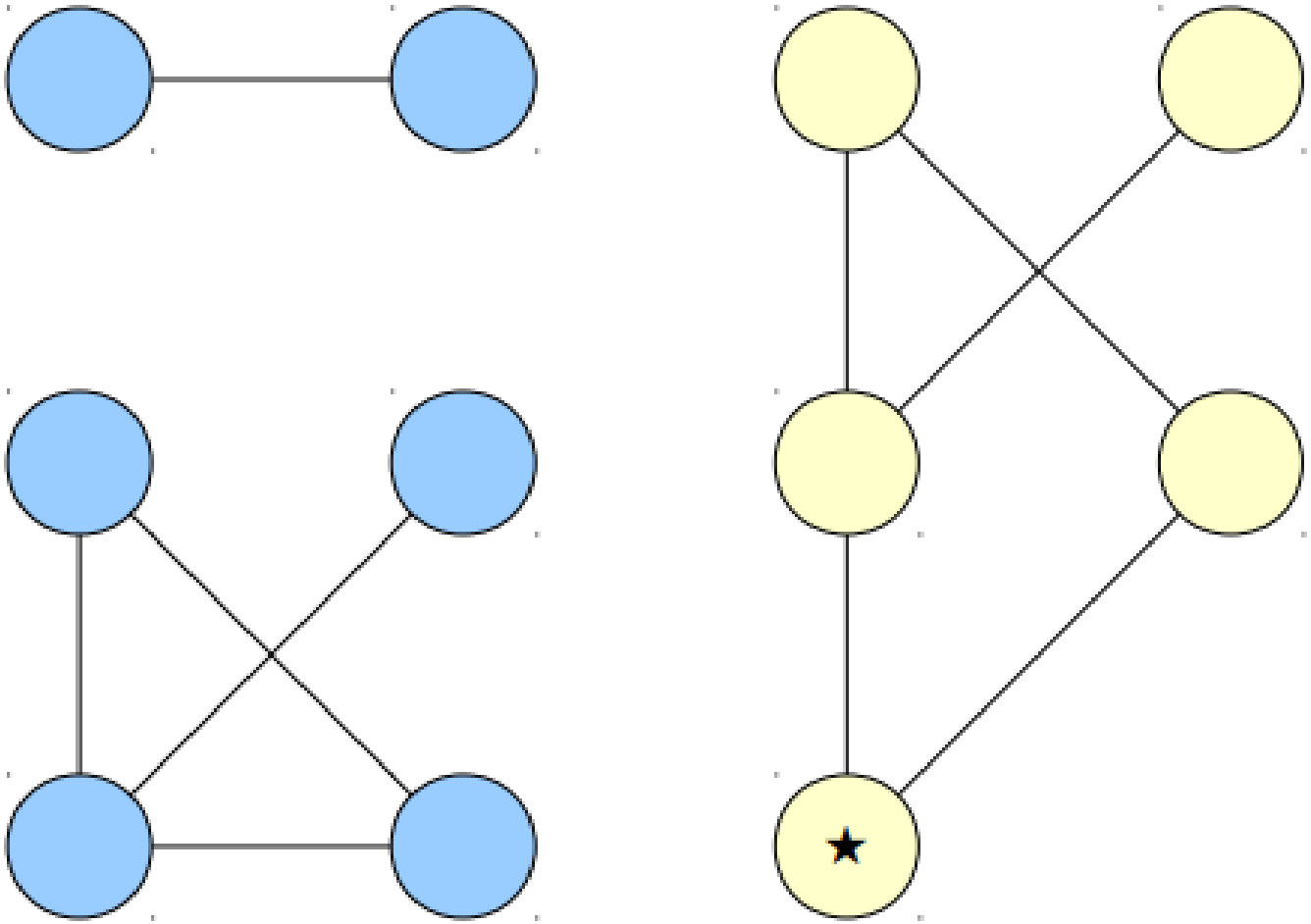












Connected Components

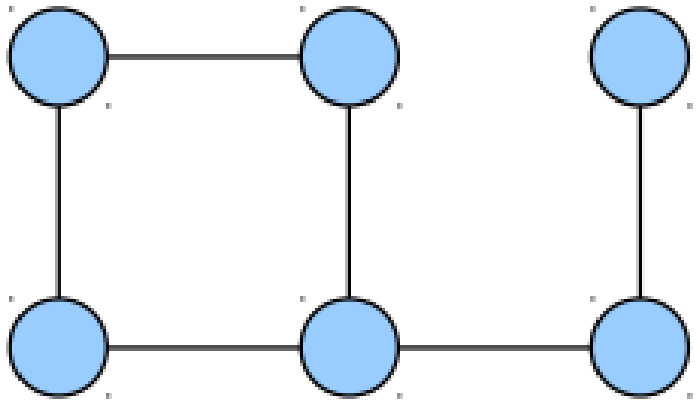
- Let $G = (V, E)$ be a graph. For each $v \in V$, the ***connected component*** containing v is the set

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

- 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.

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

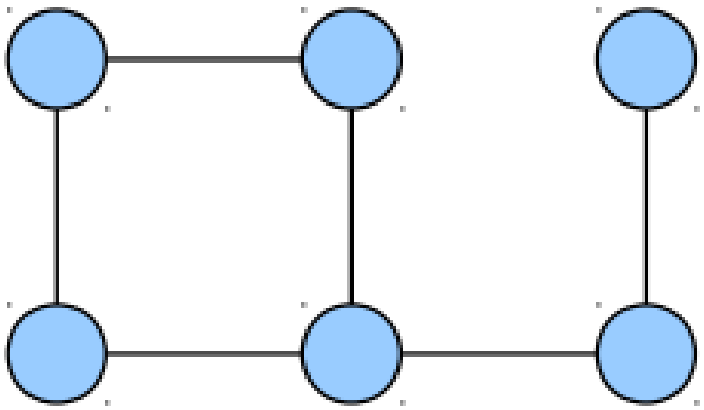
Claim: For any graph G, the "is connected to" relation is an equivalence relation.

- Is it reflexive?

  Is it symmetric?

  Is it transitive?

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



Connectivity

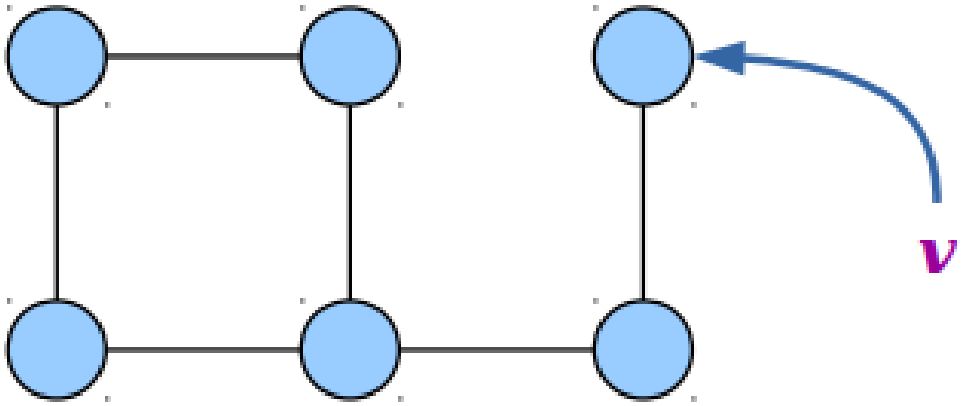
Claim: For any graph G, the "is connected to" relation is an equivalence relation.

- 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

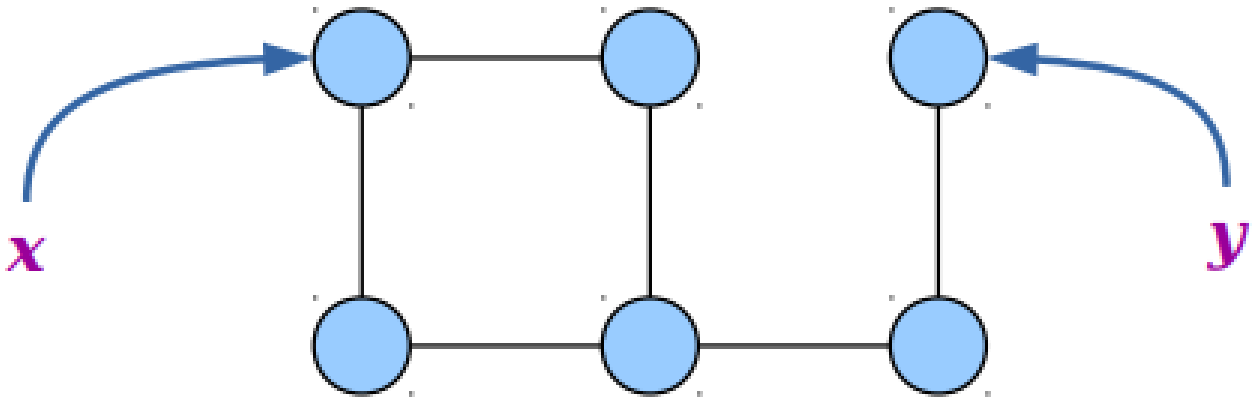
Claim: E
connecte
relation.

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

Is it reflexive?

- Is it symmetric?

Is it transitive?



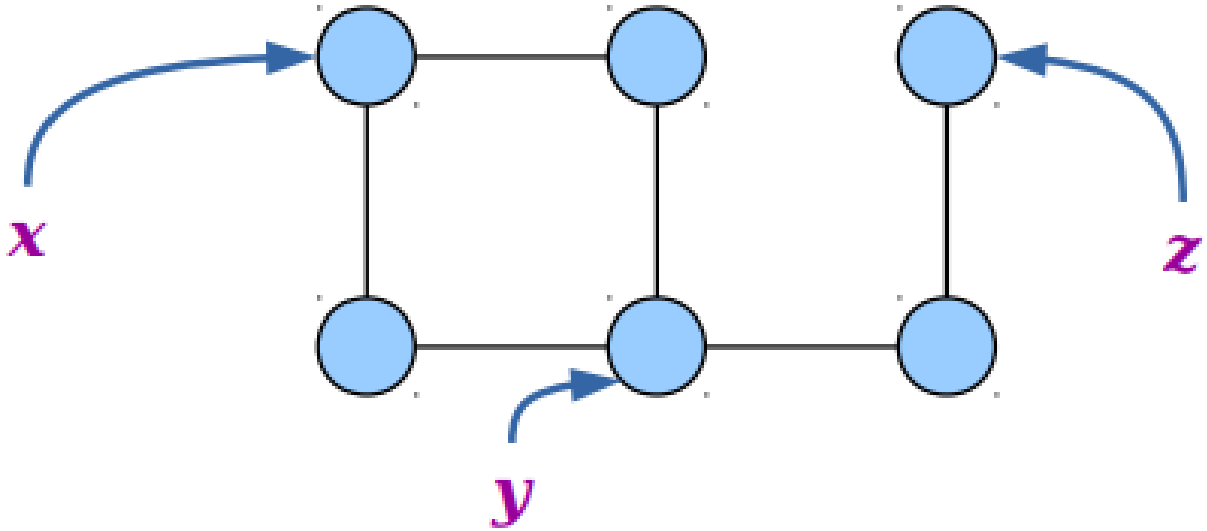
Connectivity

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

Is it reflexive?

Is it symmetric?

- Is it transitive?



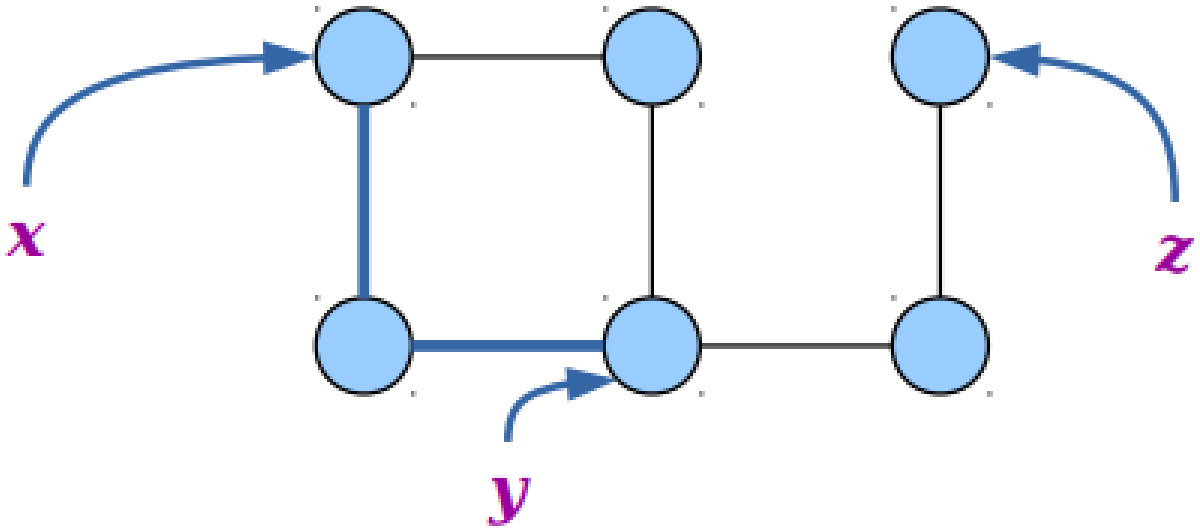
Connectivity

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

Is it reflexive?

Is it symmetric?

- Is it transitive?



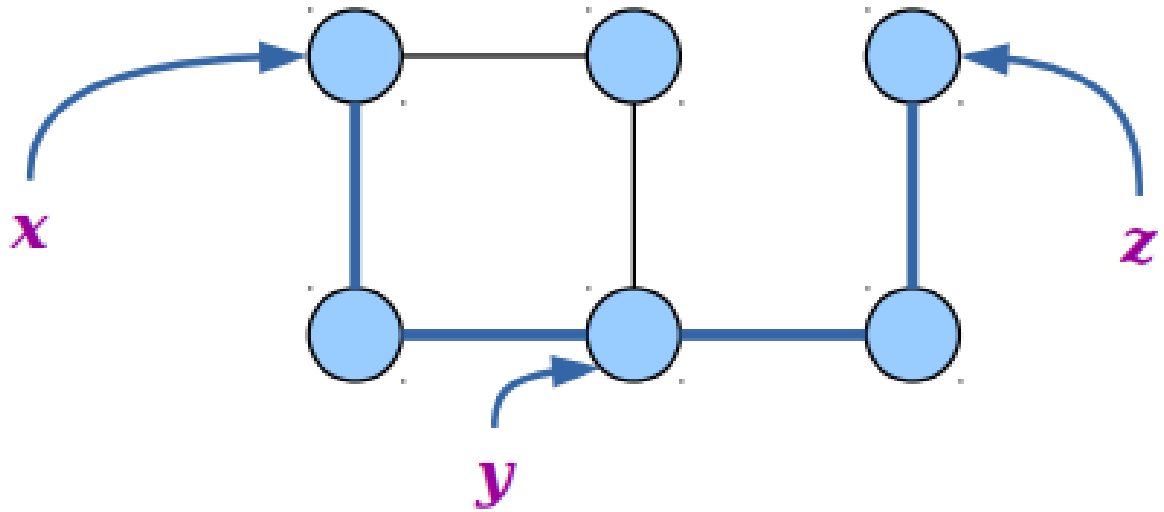
Connectivity

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

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?

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

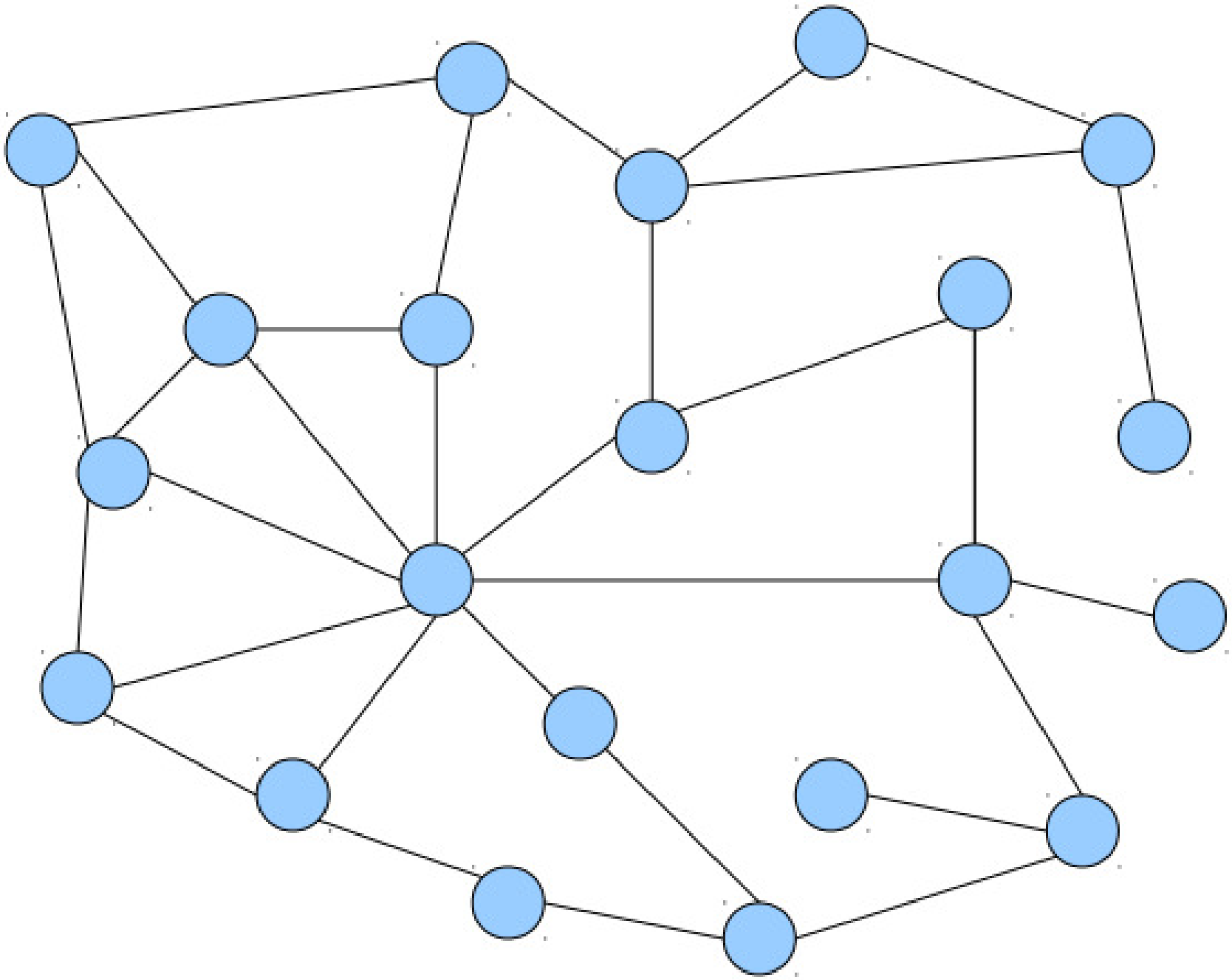
- ***Connected components are equivalence classes of the connectivity relation!***

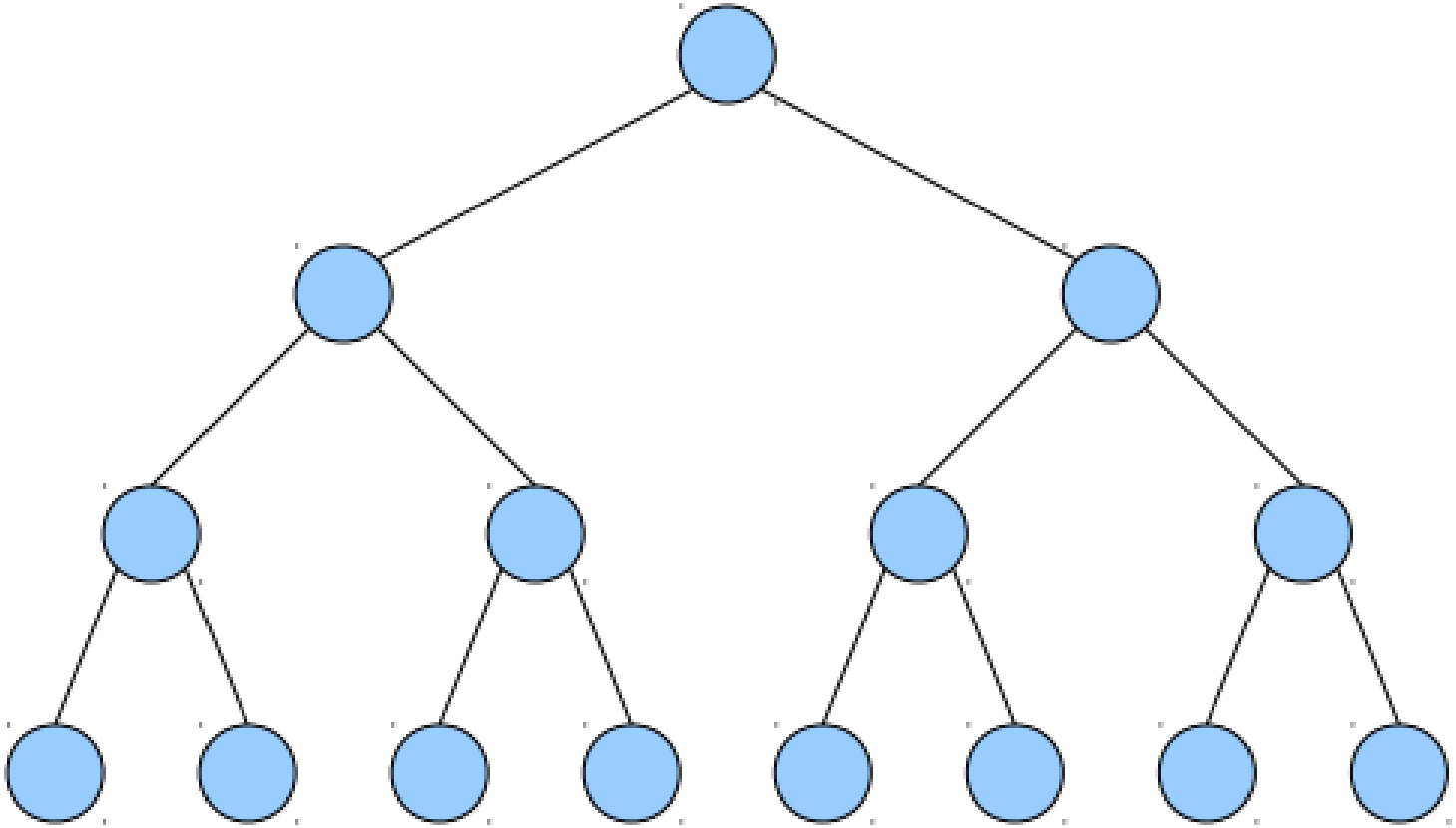
The Fundamental Theorem of Equivalence Relations: Let R be an equivalence relation over a set A. Then every element $a \in A$ belongs to exactly one equivalence class of R.

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. ■

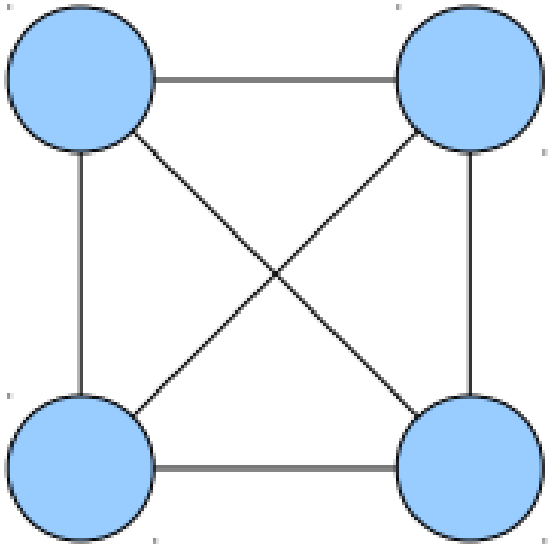
Planar Graphs



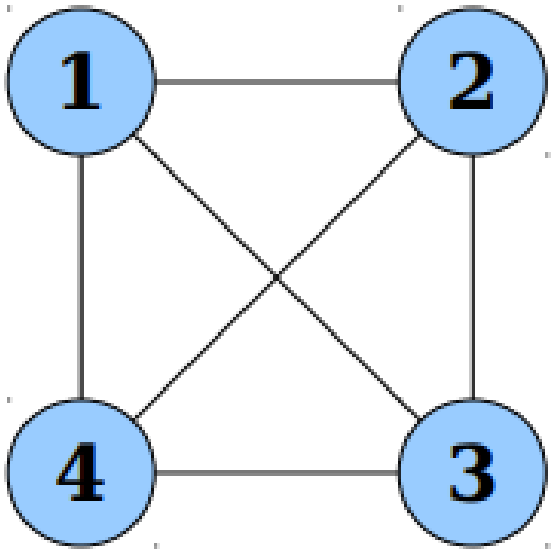


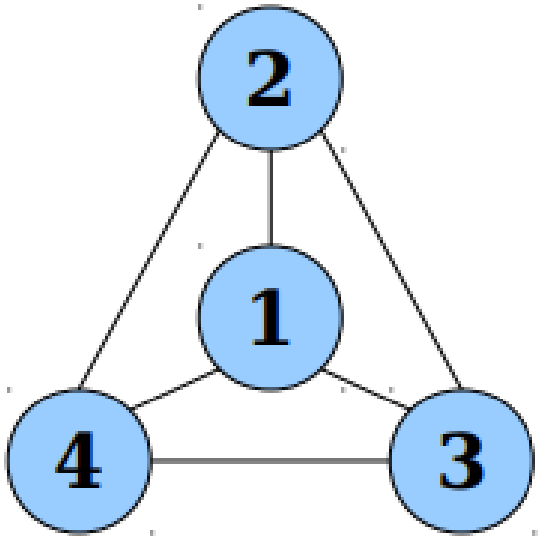
A graph is called a ***planar graph*** if there is some way to draw it in a 2D plane without any of the edges crossing.

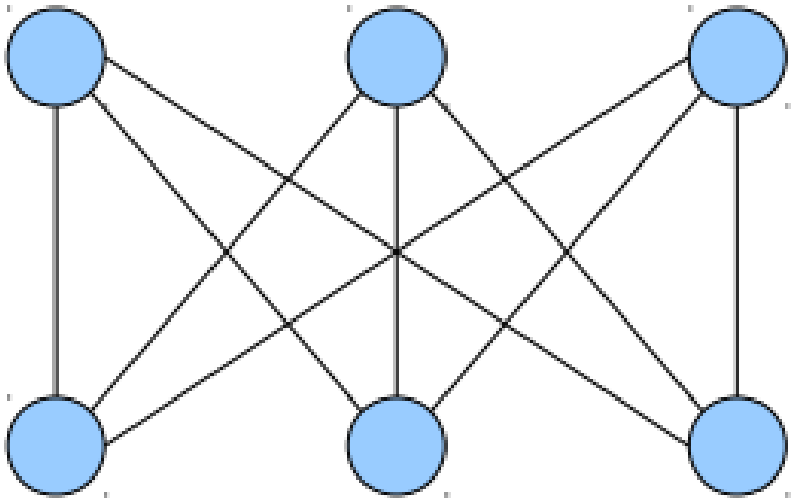
A graph is called a ***planar graph*** if there is some way to draw it in a 2D plane without any of the edges crossing.

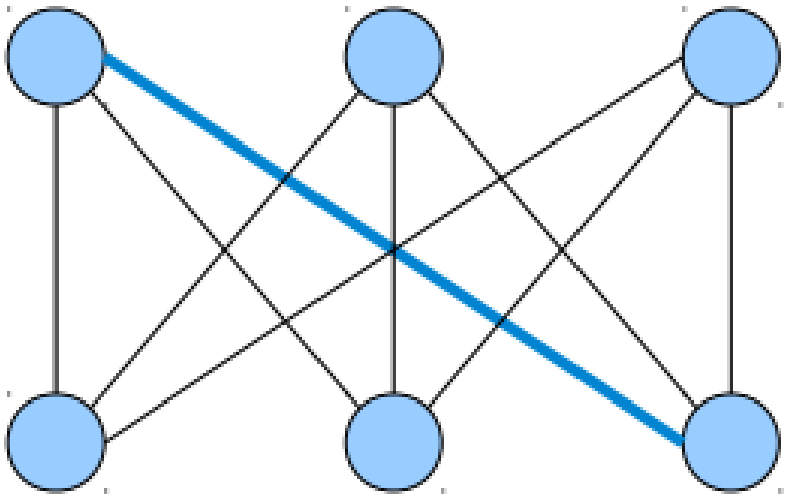


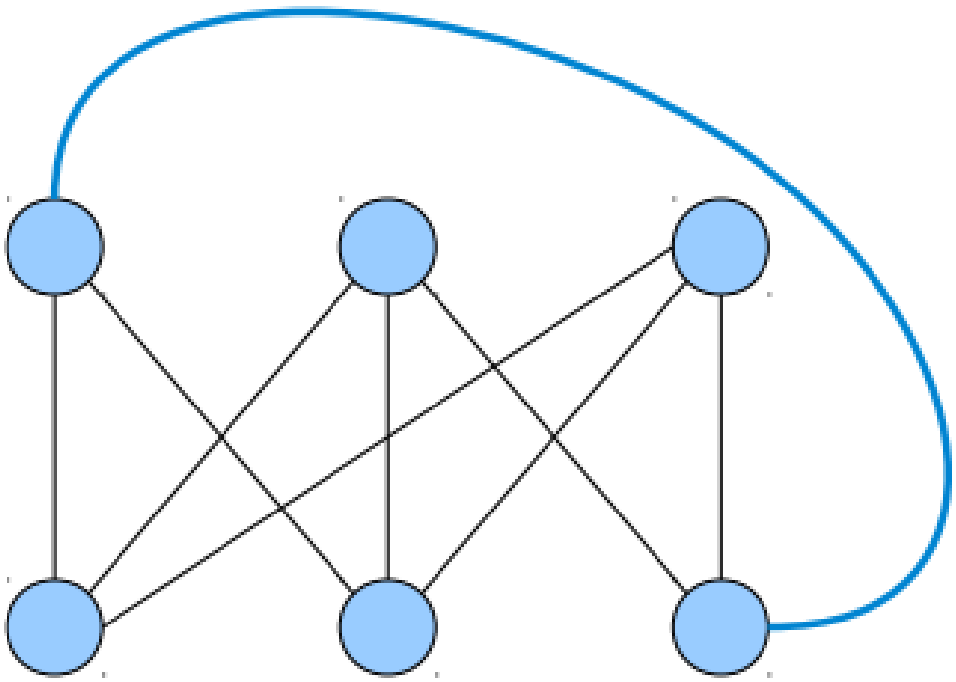
Is this graph planar?

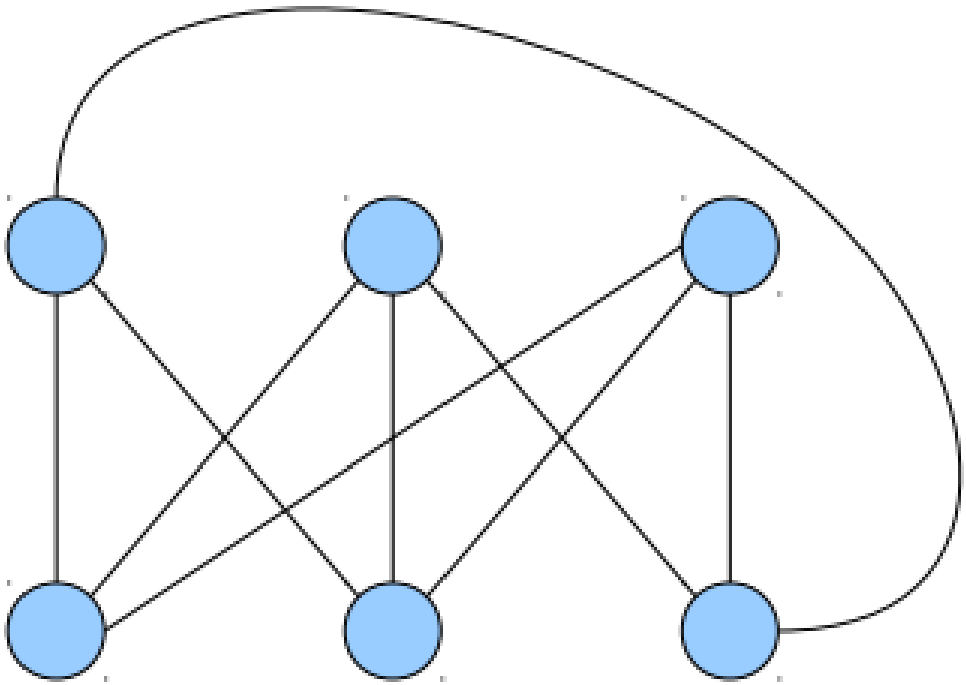

1
2
4
3

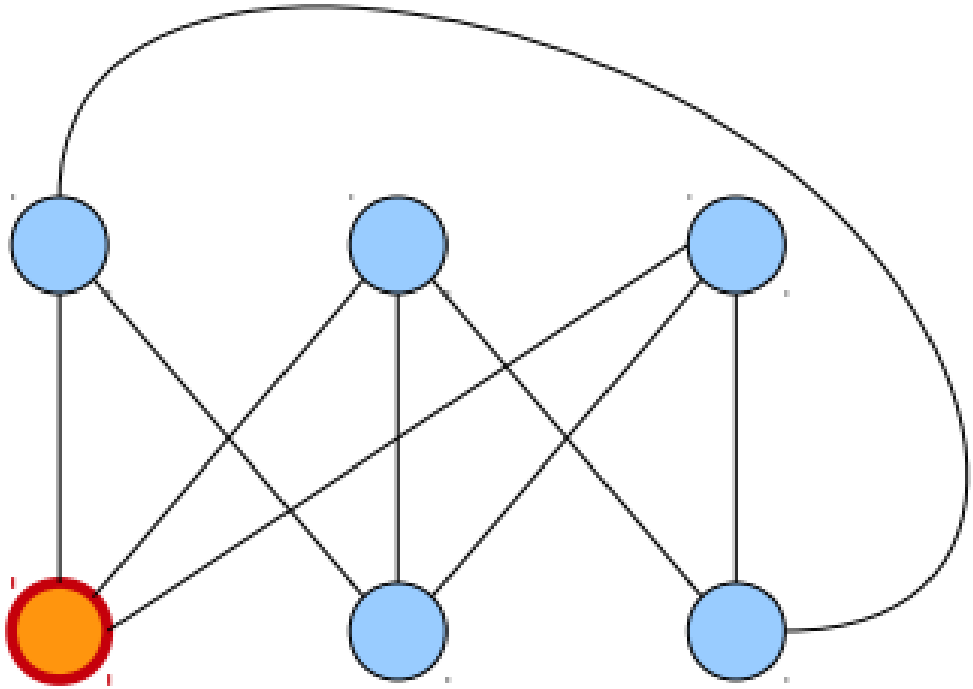

2
1
4
3

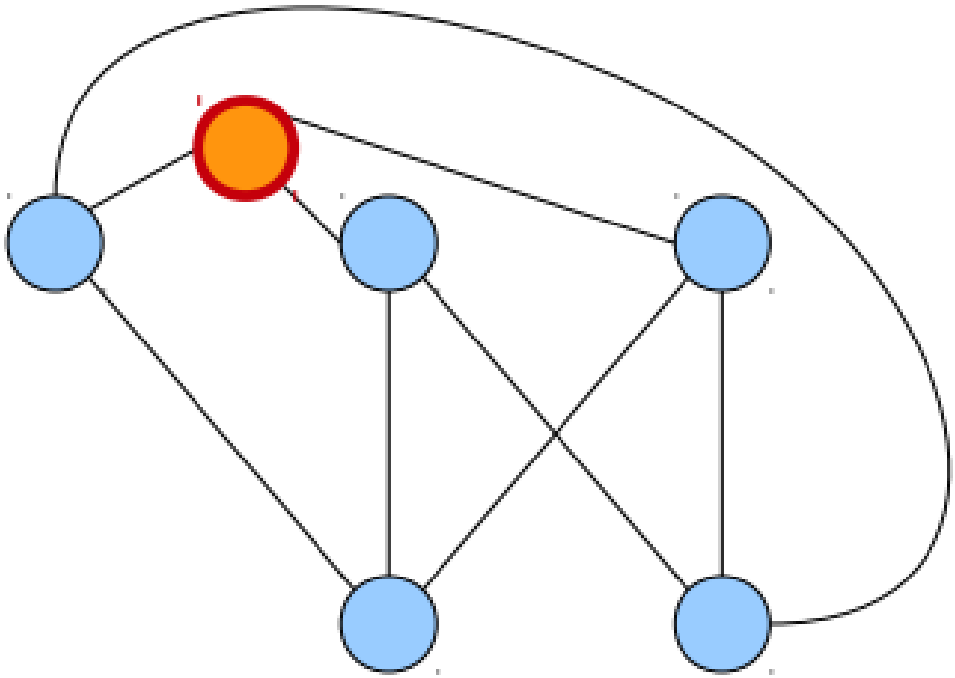


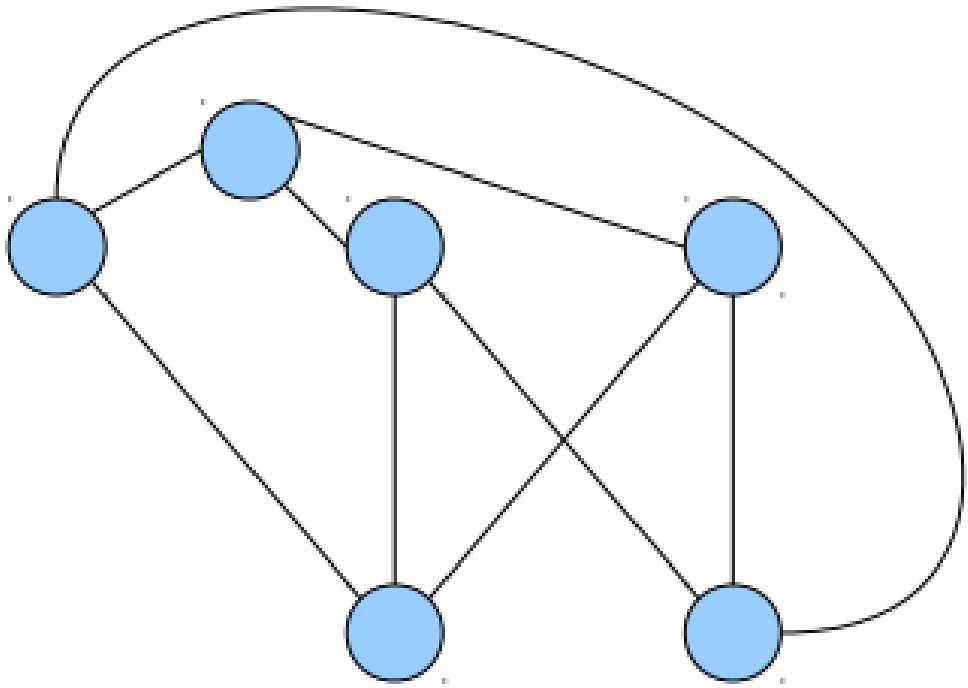


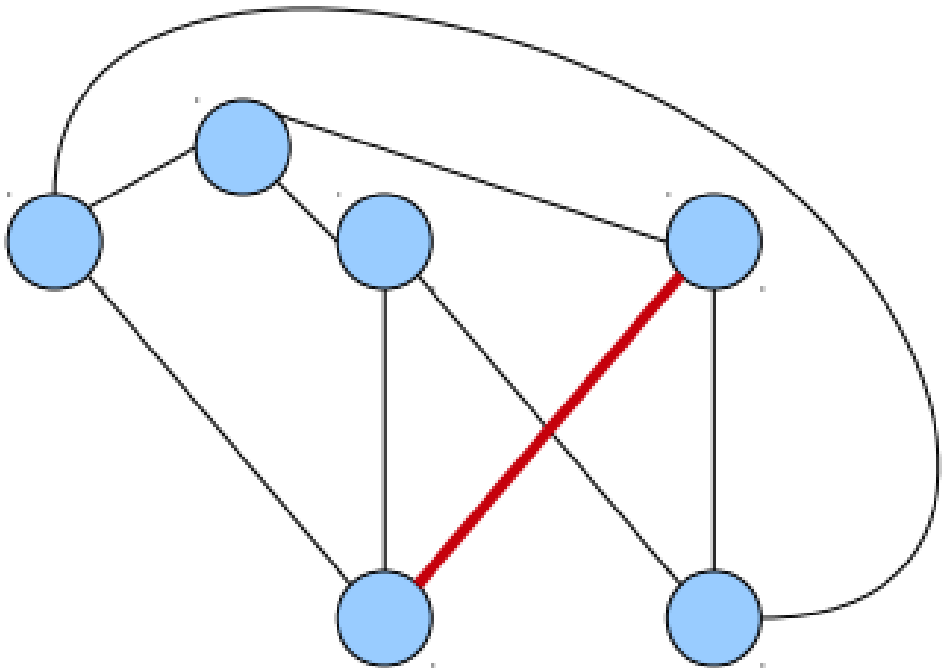


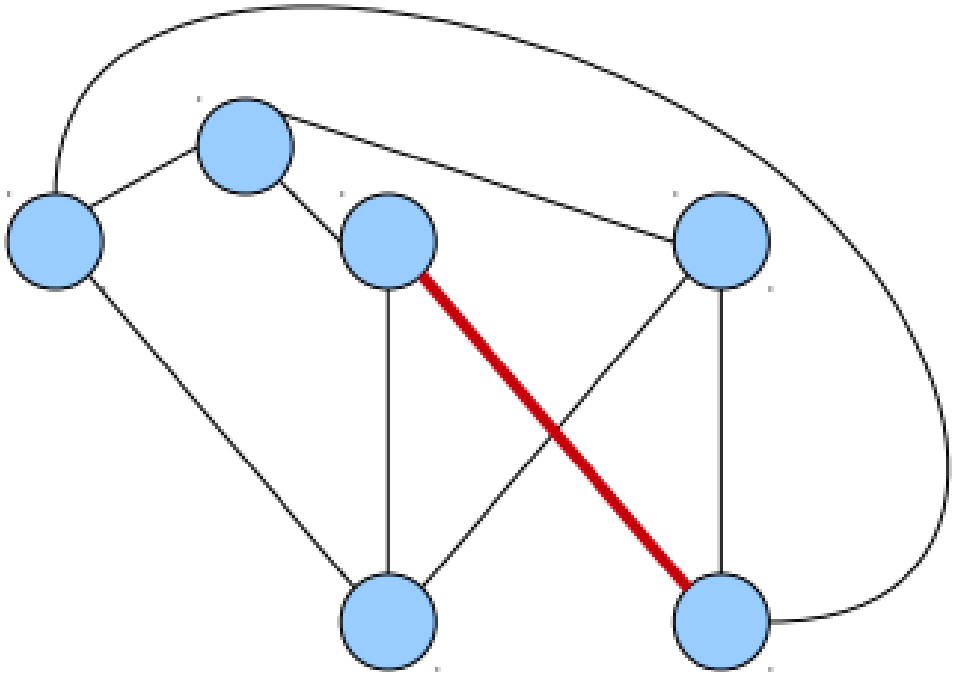


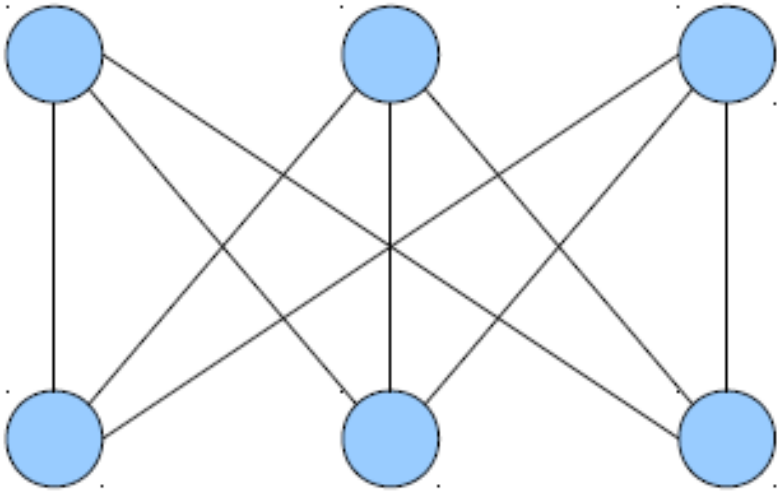












This graph is called the ***utility graph***. There is no way to draw it in the plane without edges crossing.

A Fun Game: **http://planarity.net/**

Graph Coloring

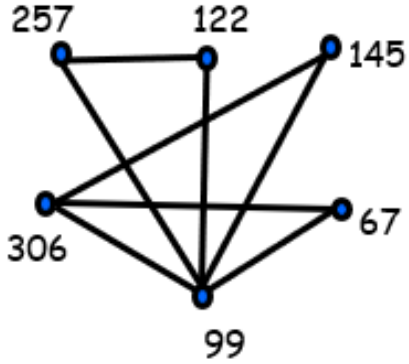
Graph Coloring Problem:

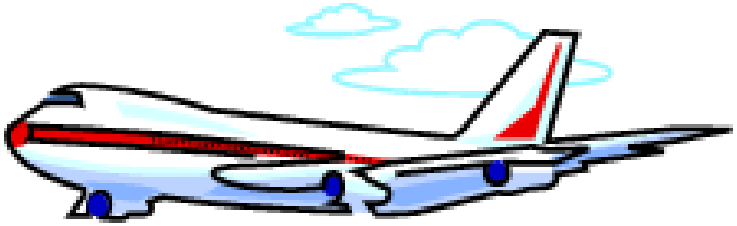
Given a graph, color all the vertices so that two adjacent vertices get different colors.

Objective: use minimum number of colors.

The minimum number is called chromatic number.

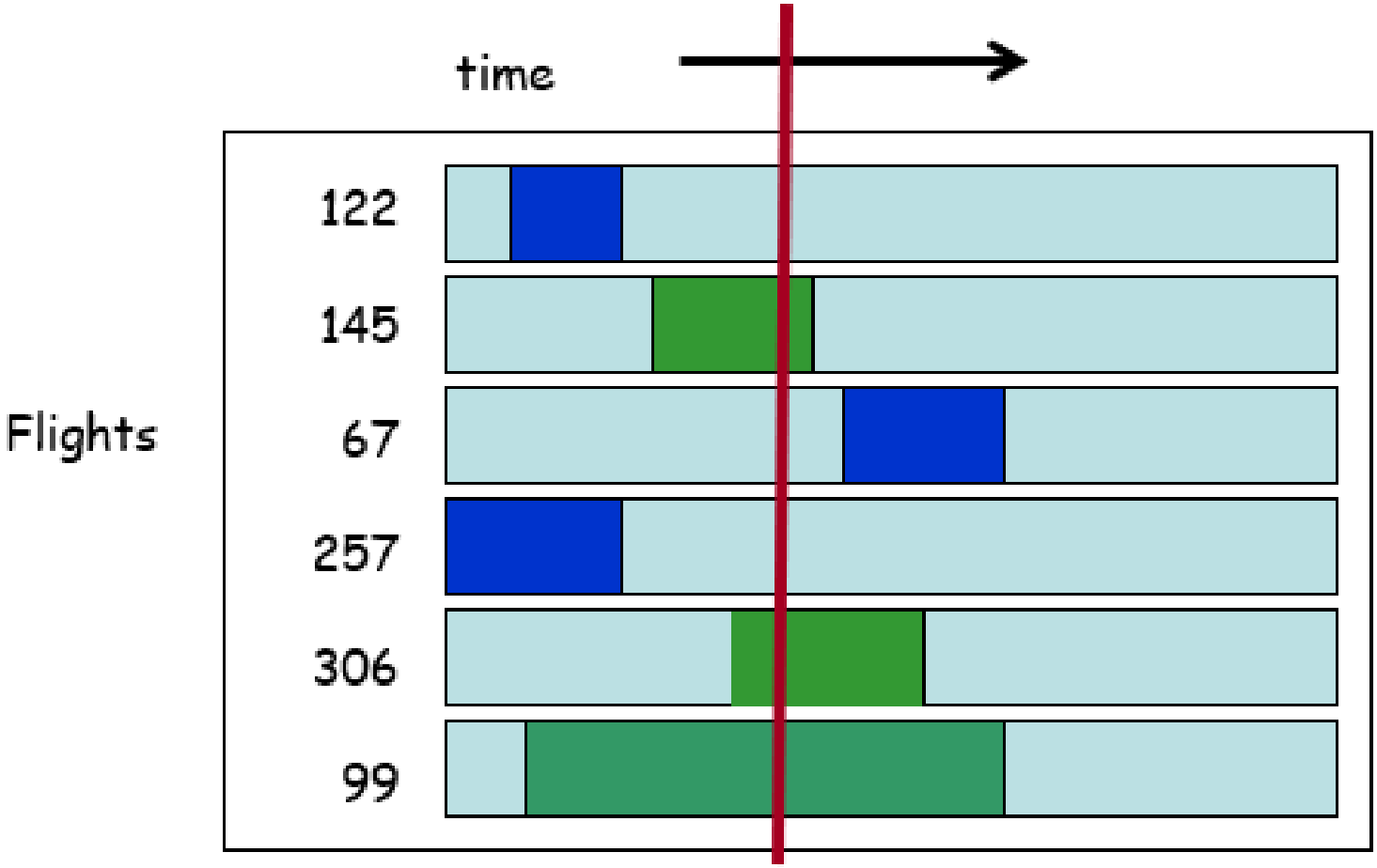
Flight Gates



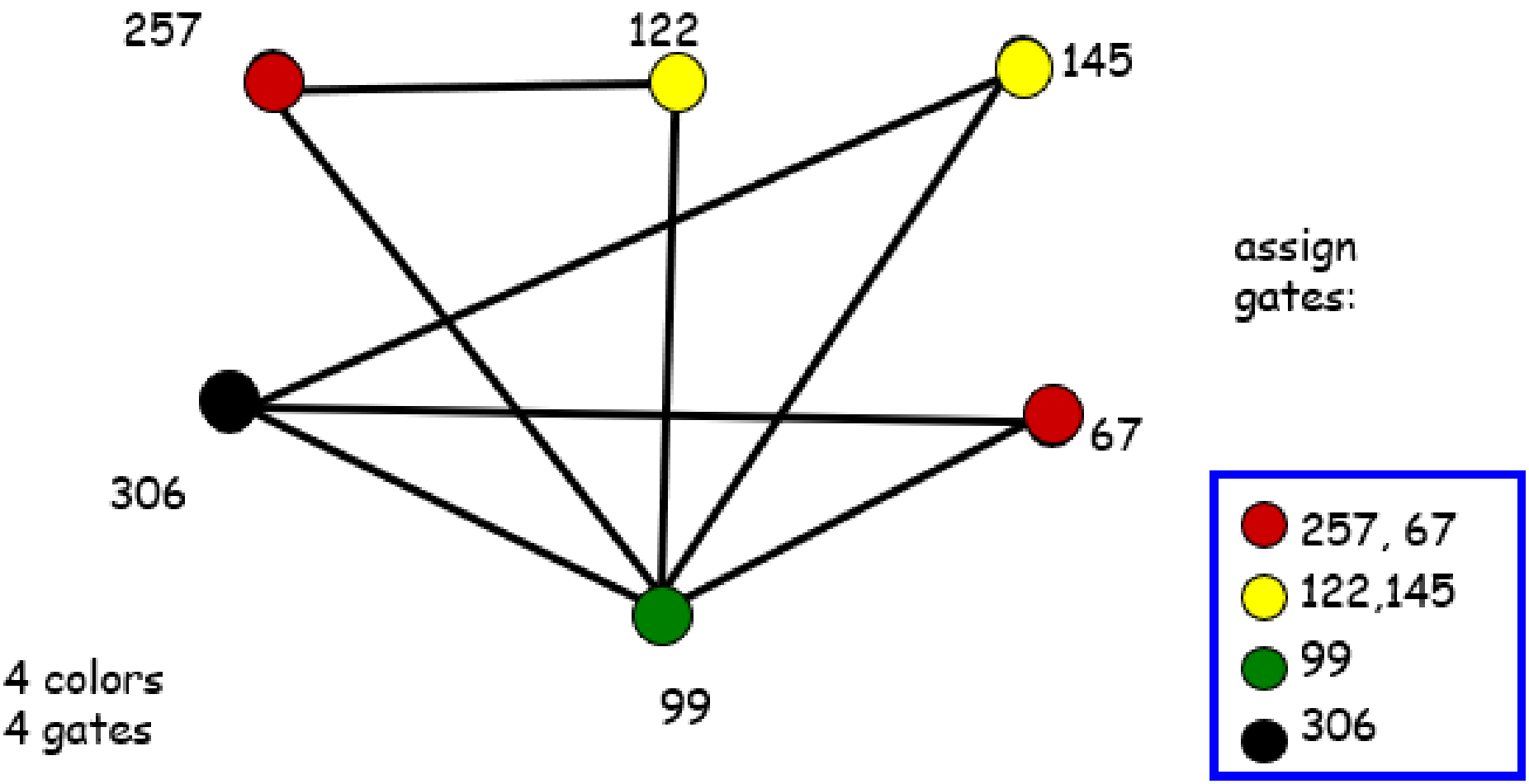


flights need gates, but times overlap.

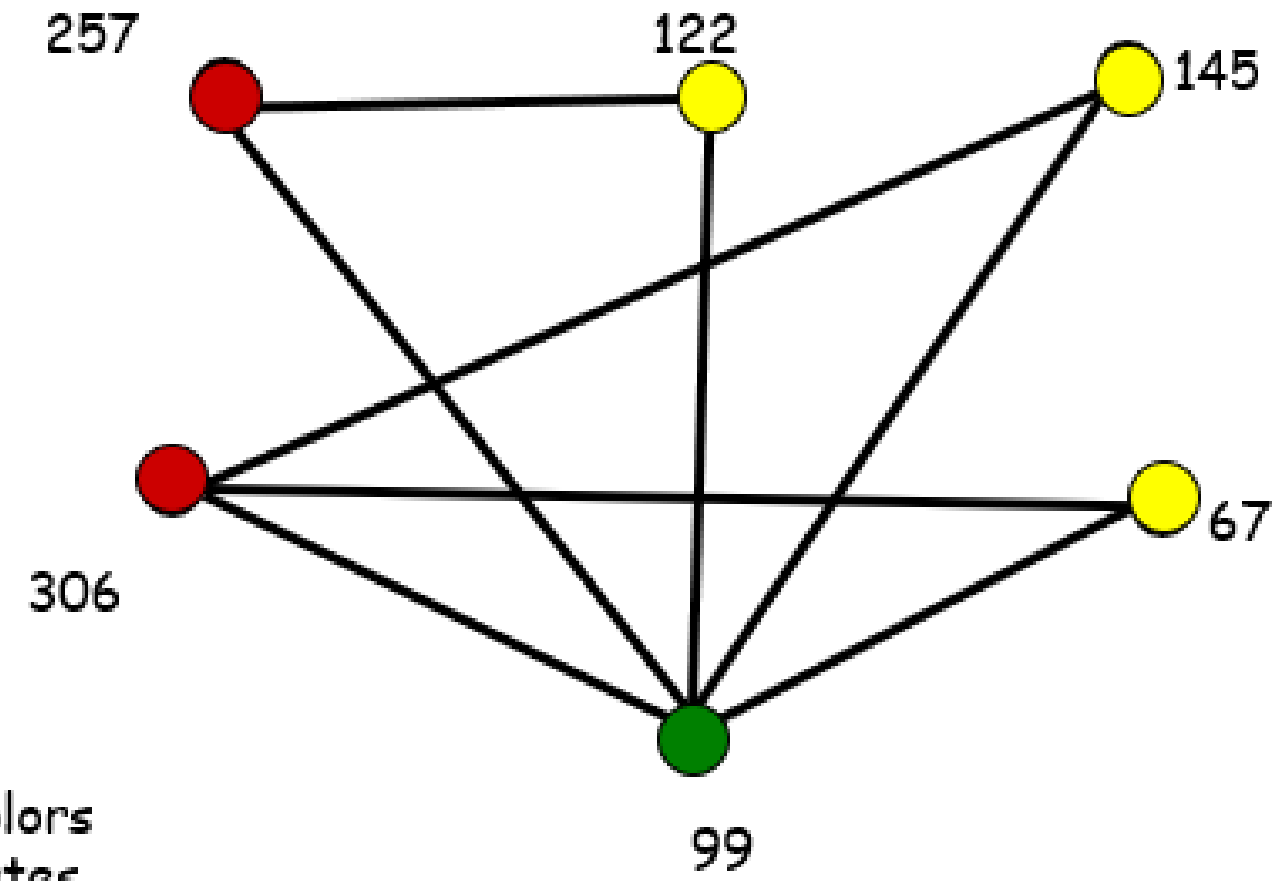
how many gates needed?



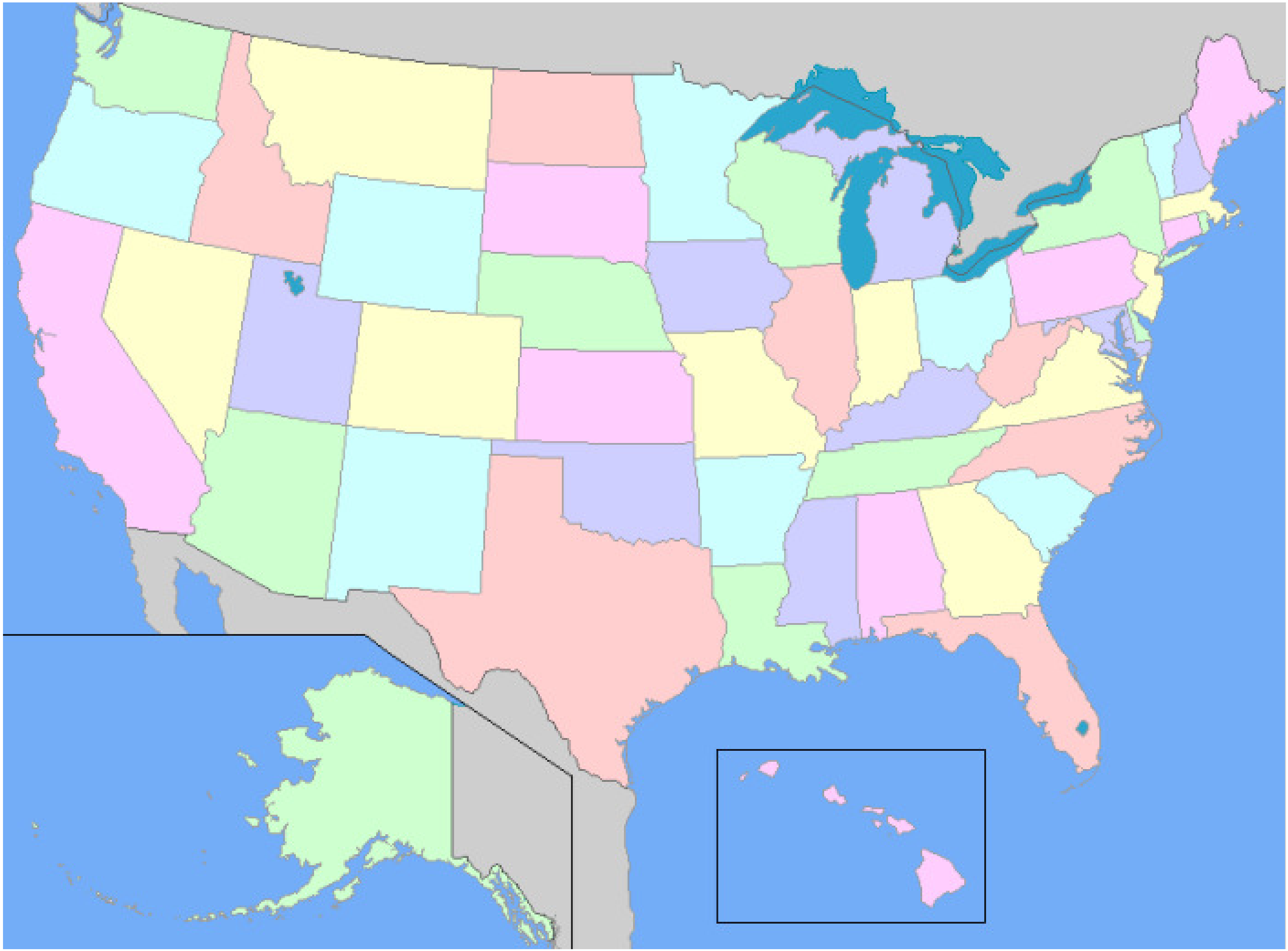
Colouring the Vertices

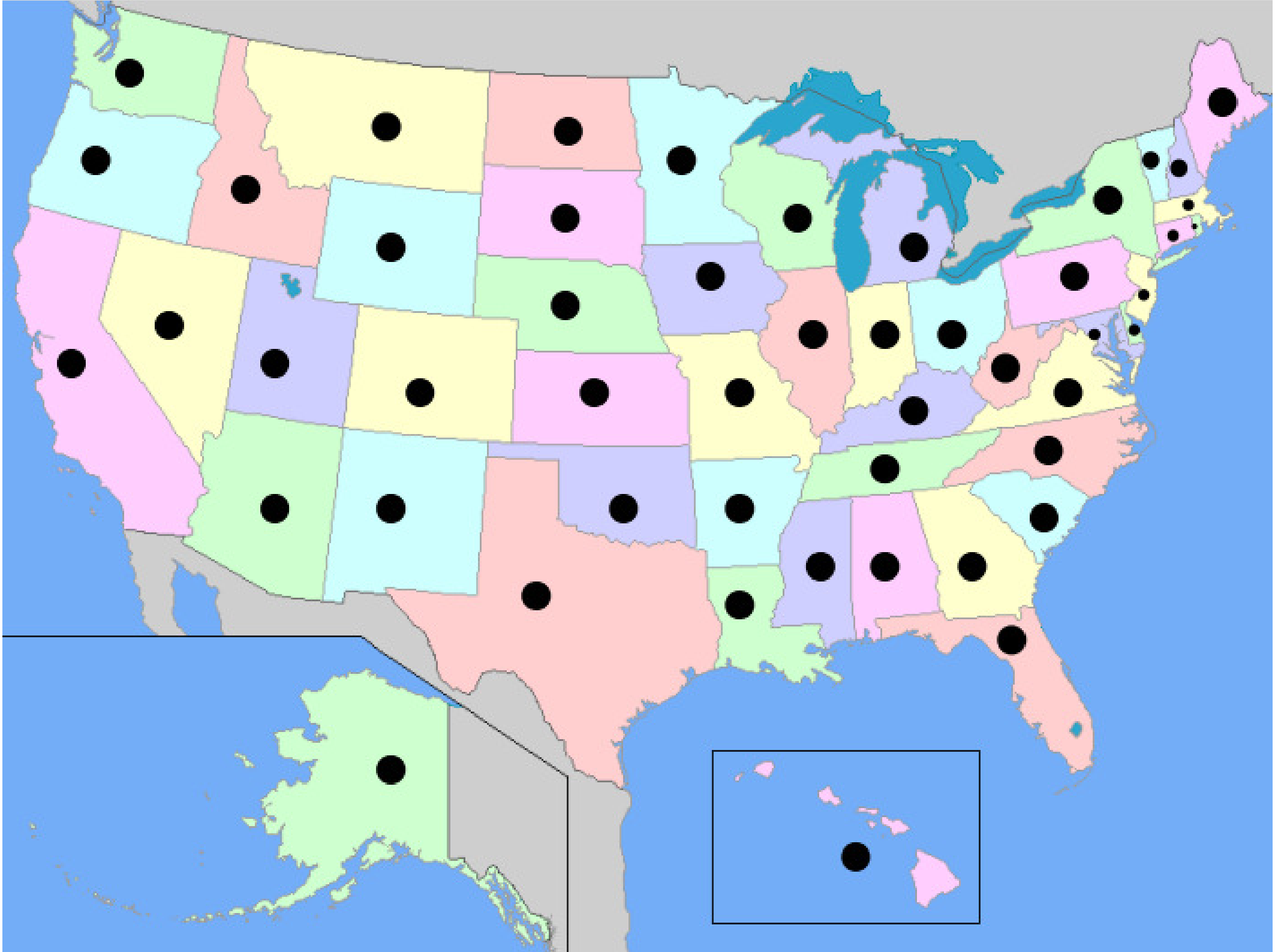


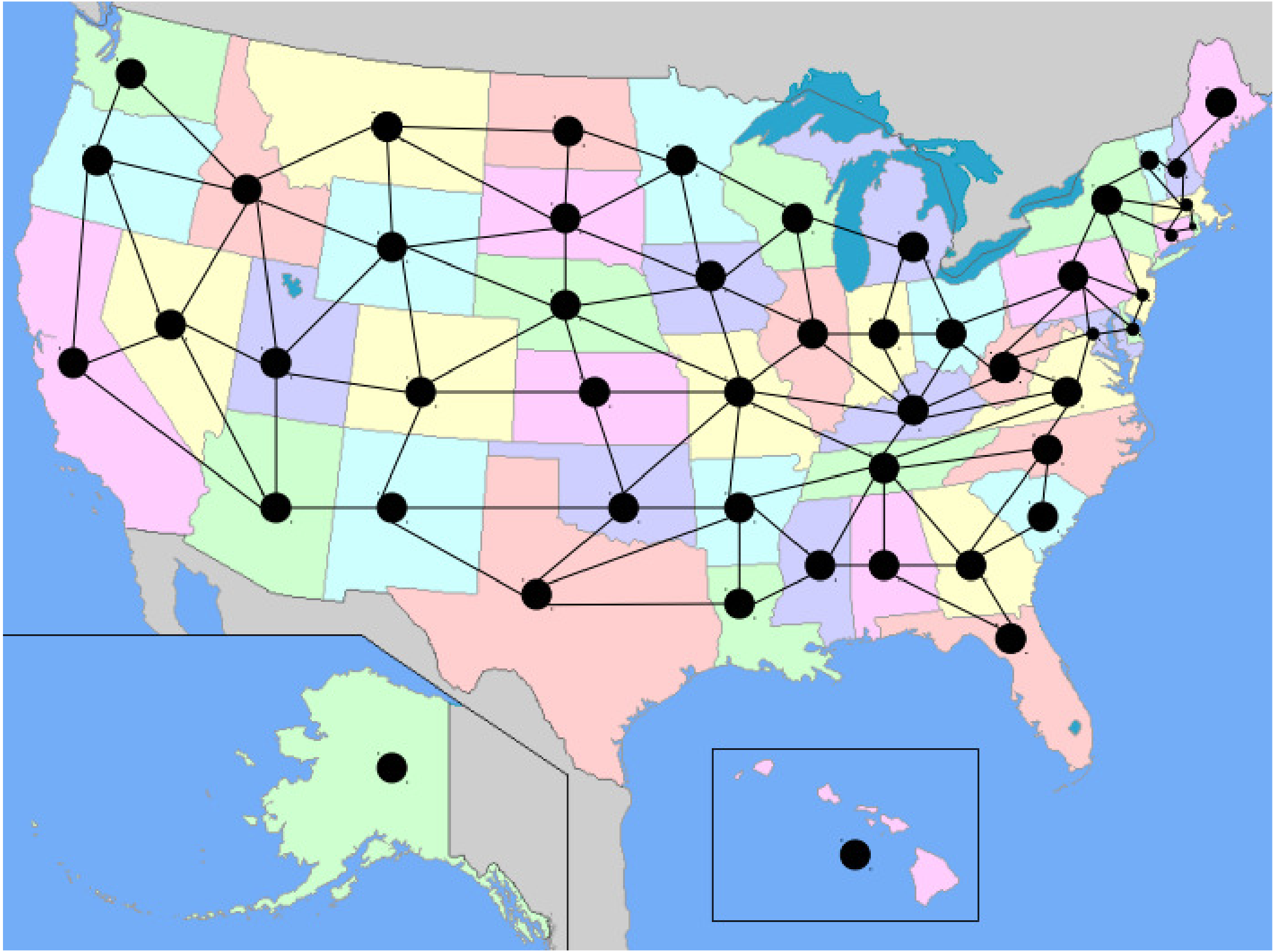
Better Colouring

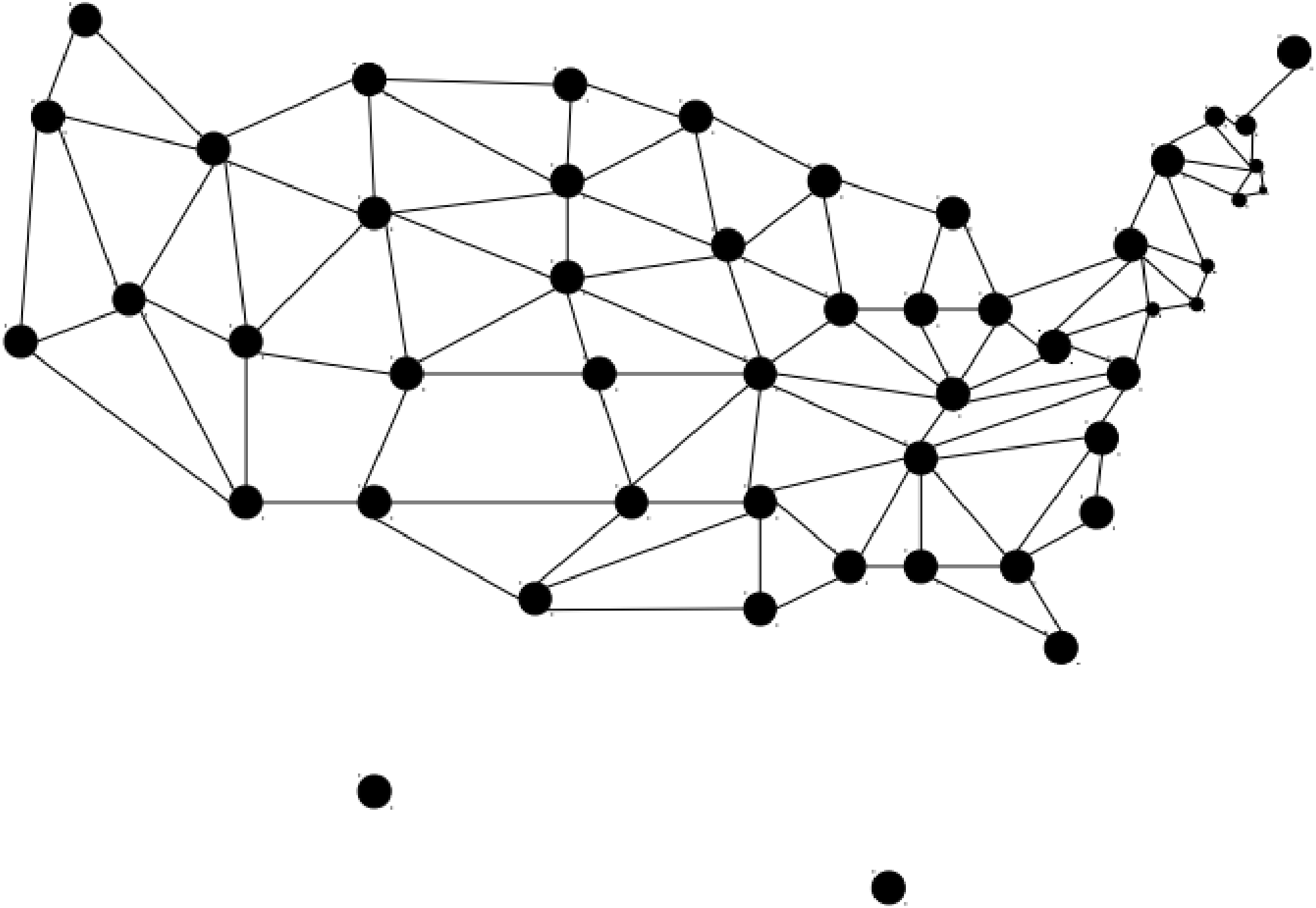


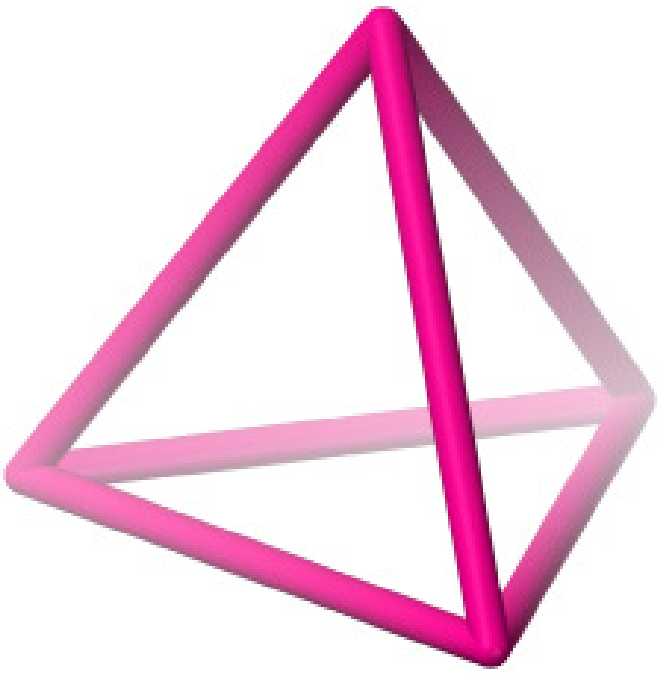
3 colors
3 gates

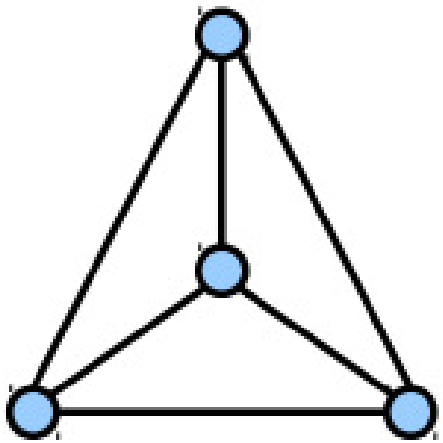


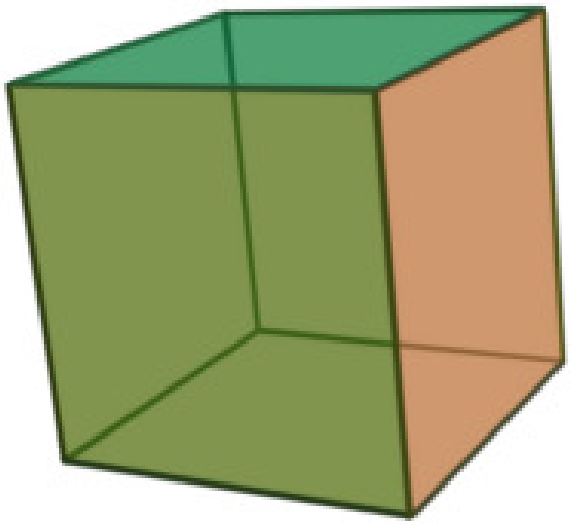


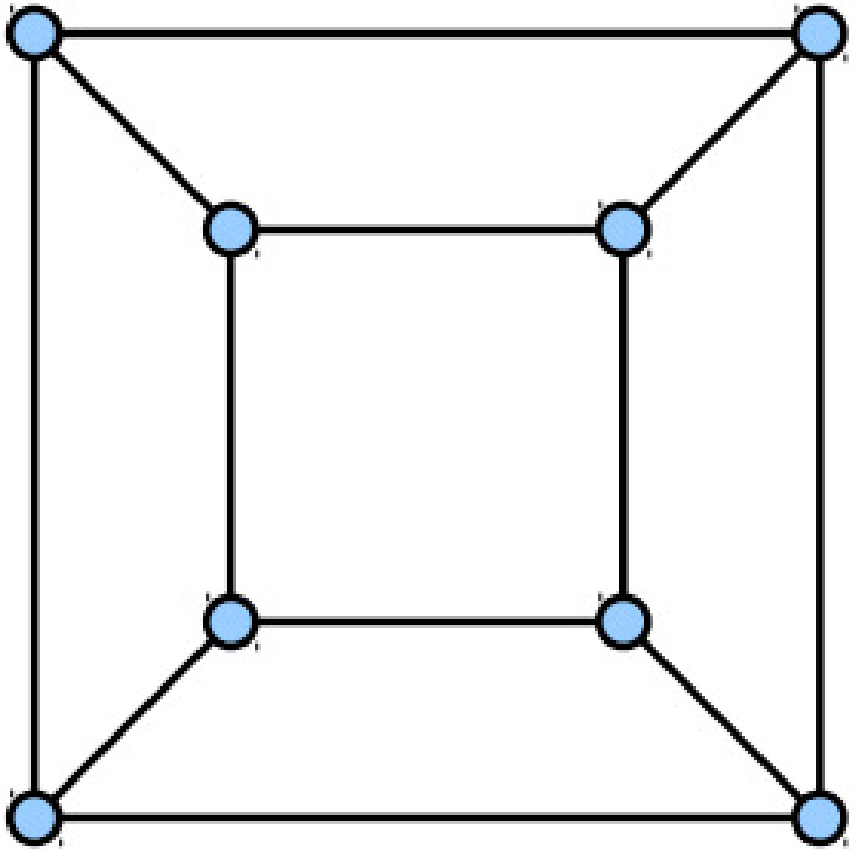


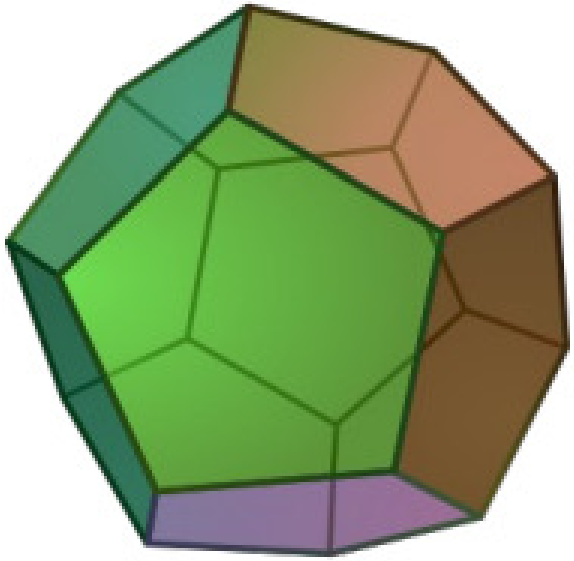


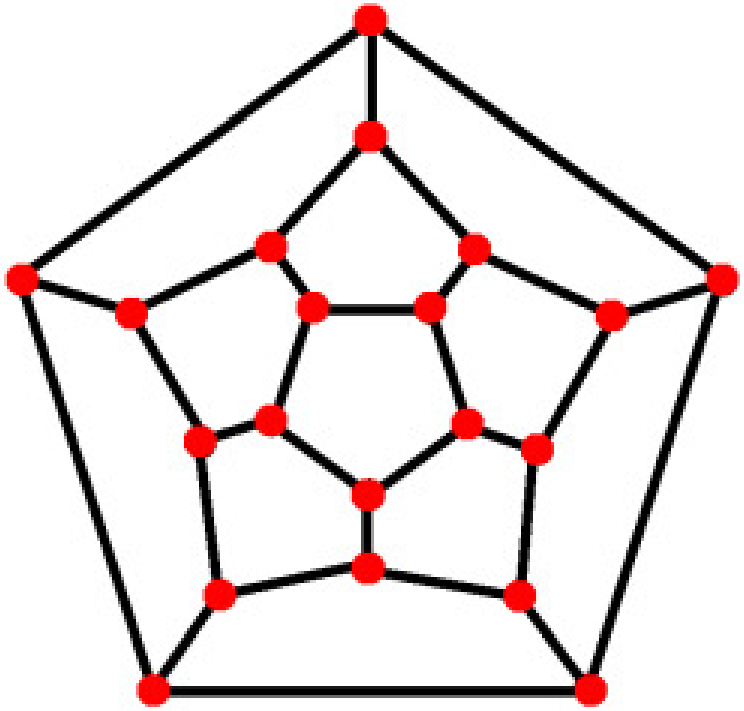




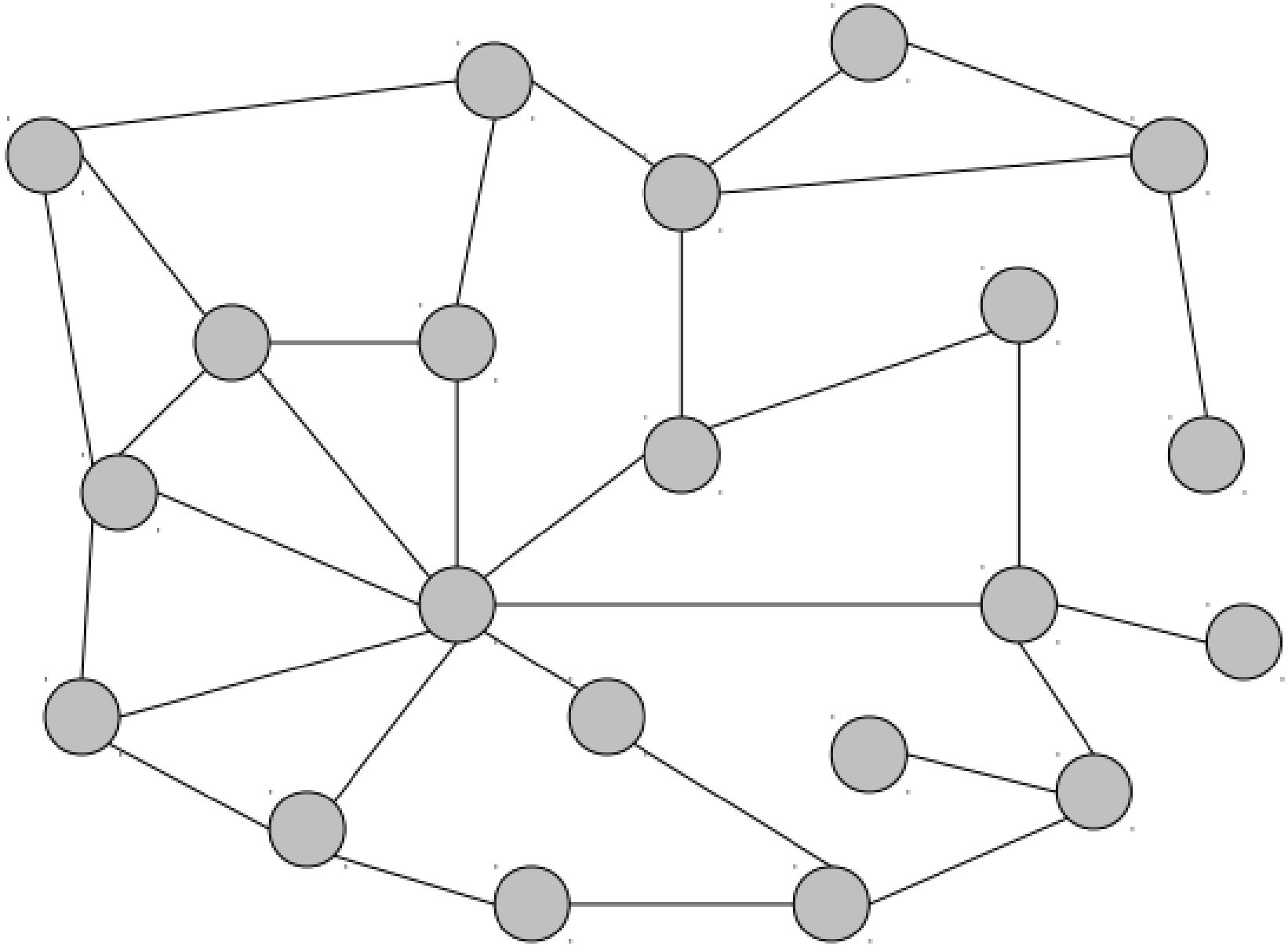




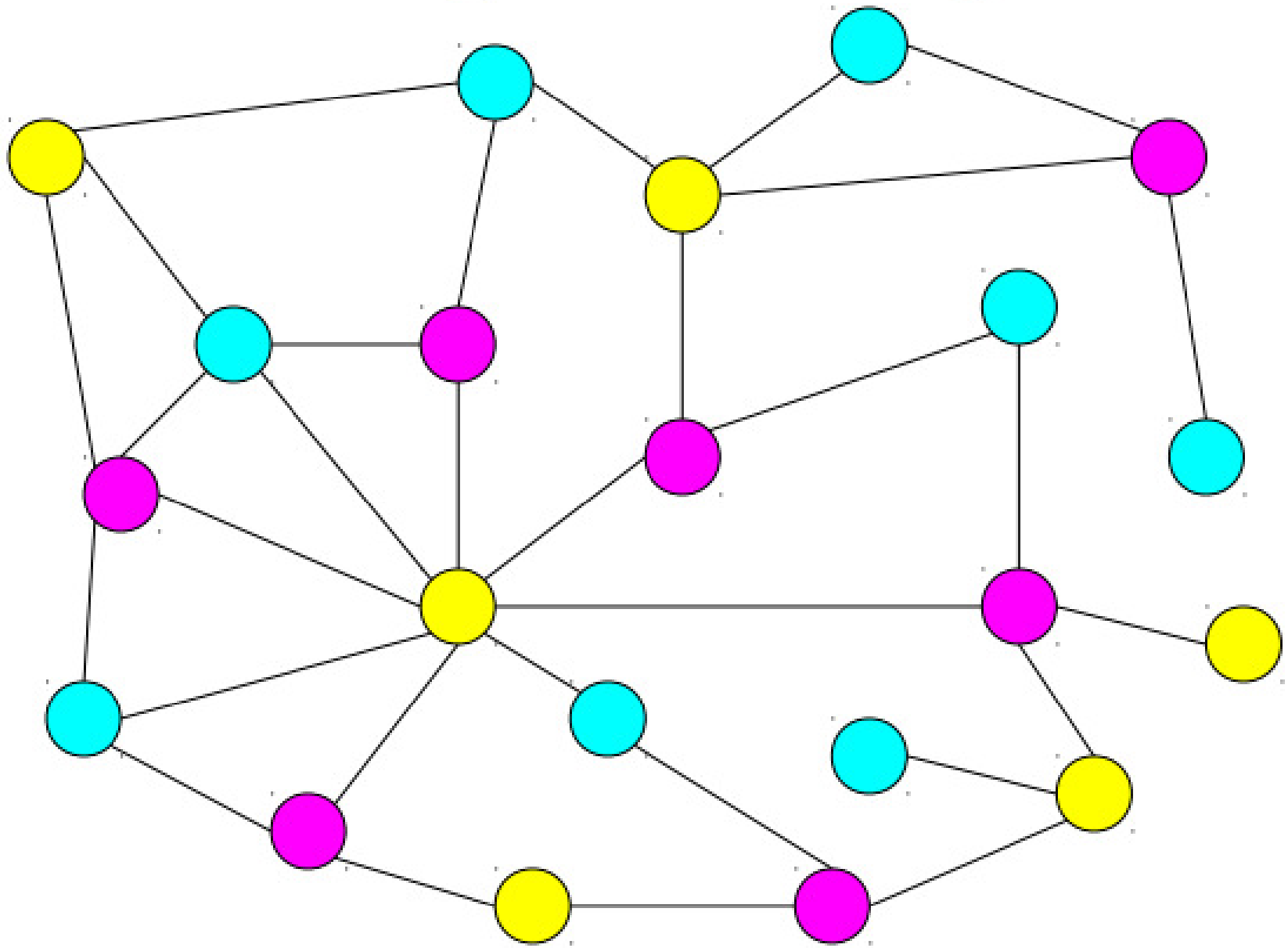


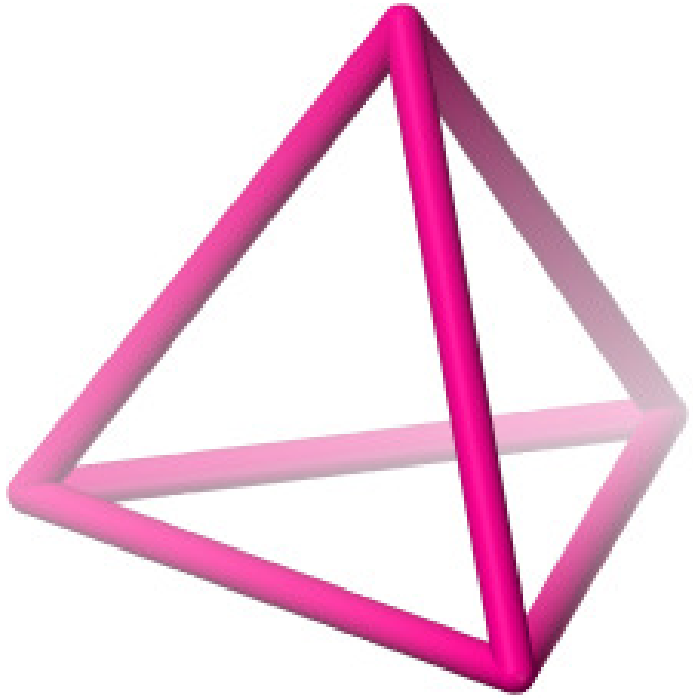


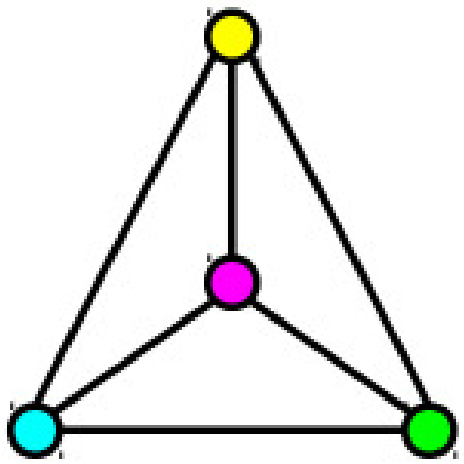
Graph Coloring

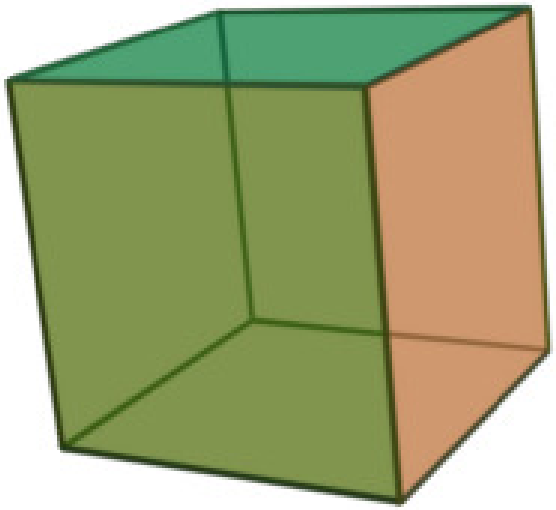


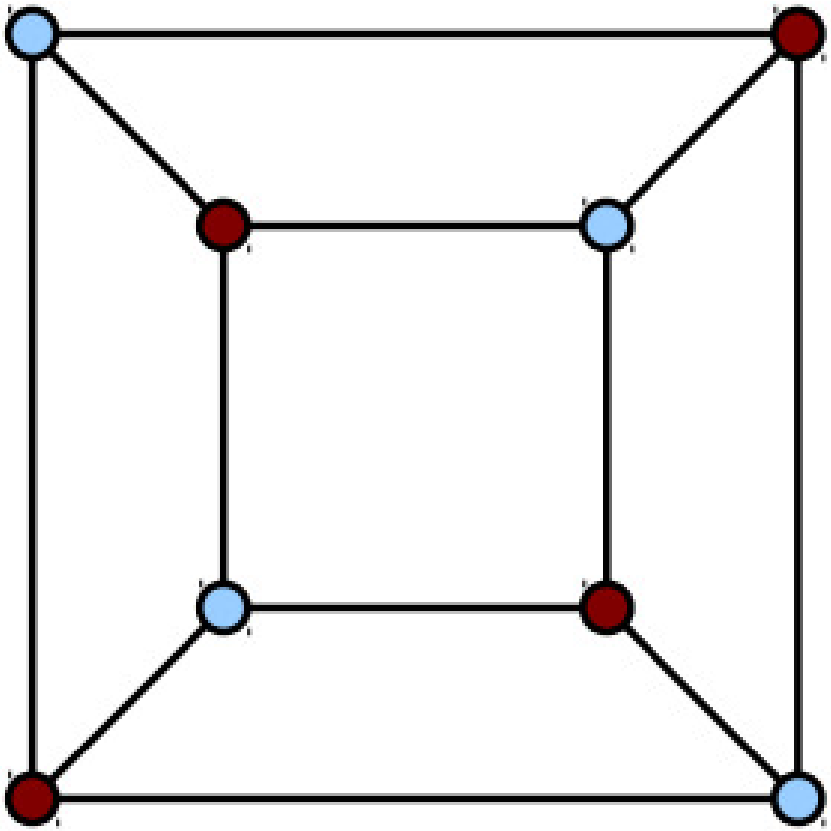
Graph Coloring











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

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

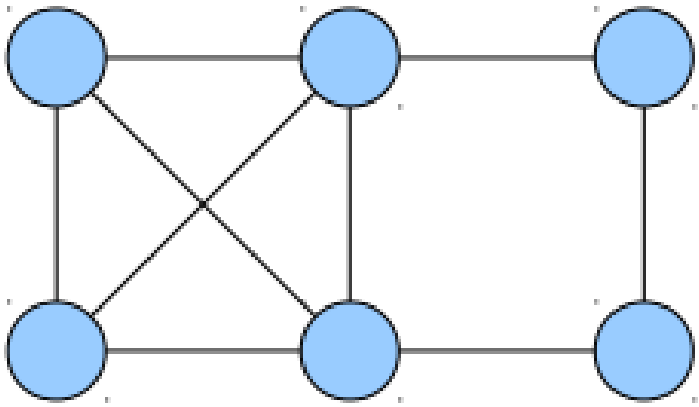
Graph Coloring

- 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

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

such that

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



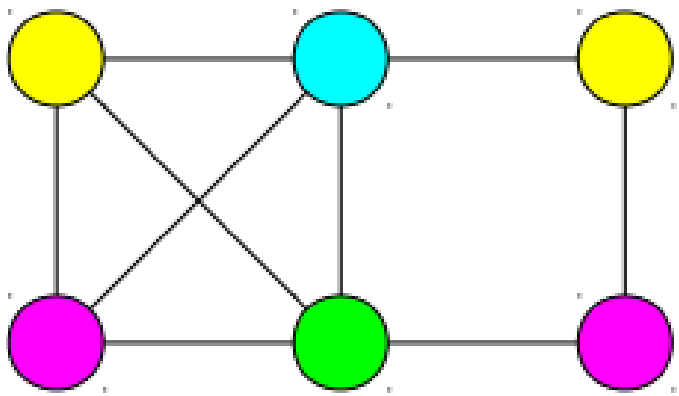
Graph Coloring

- 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

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

such that

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



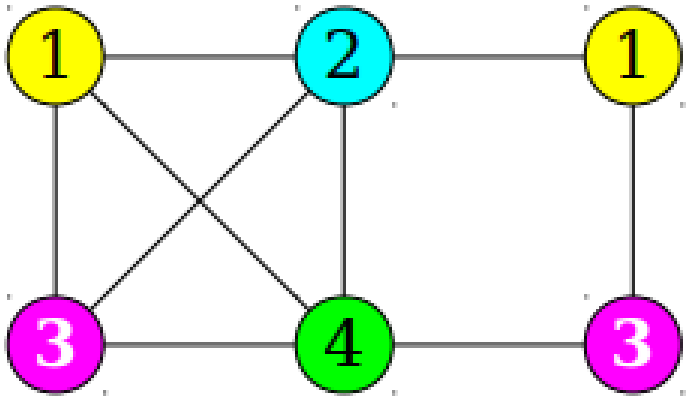
Graph Coloring

- 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

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

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

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

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

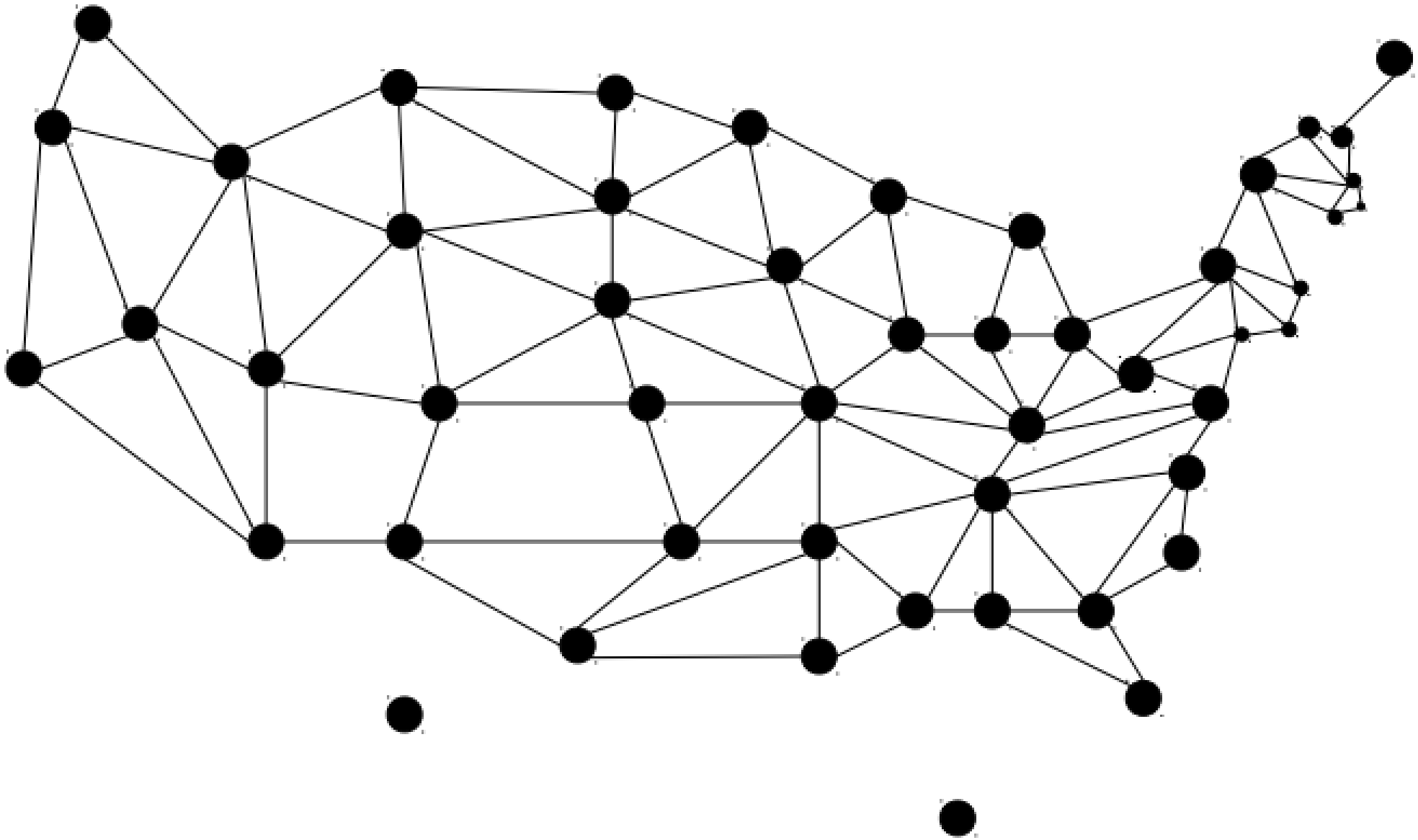
$$f : V \to \{1, 2, \ldots, k\}$$

  such that

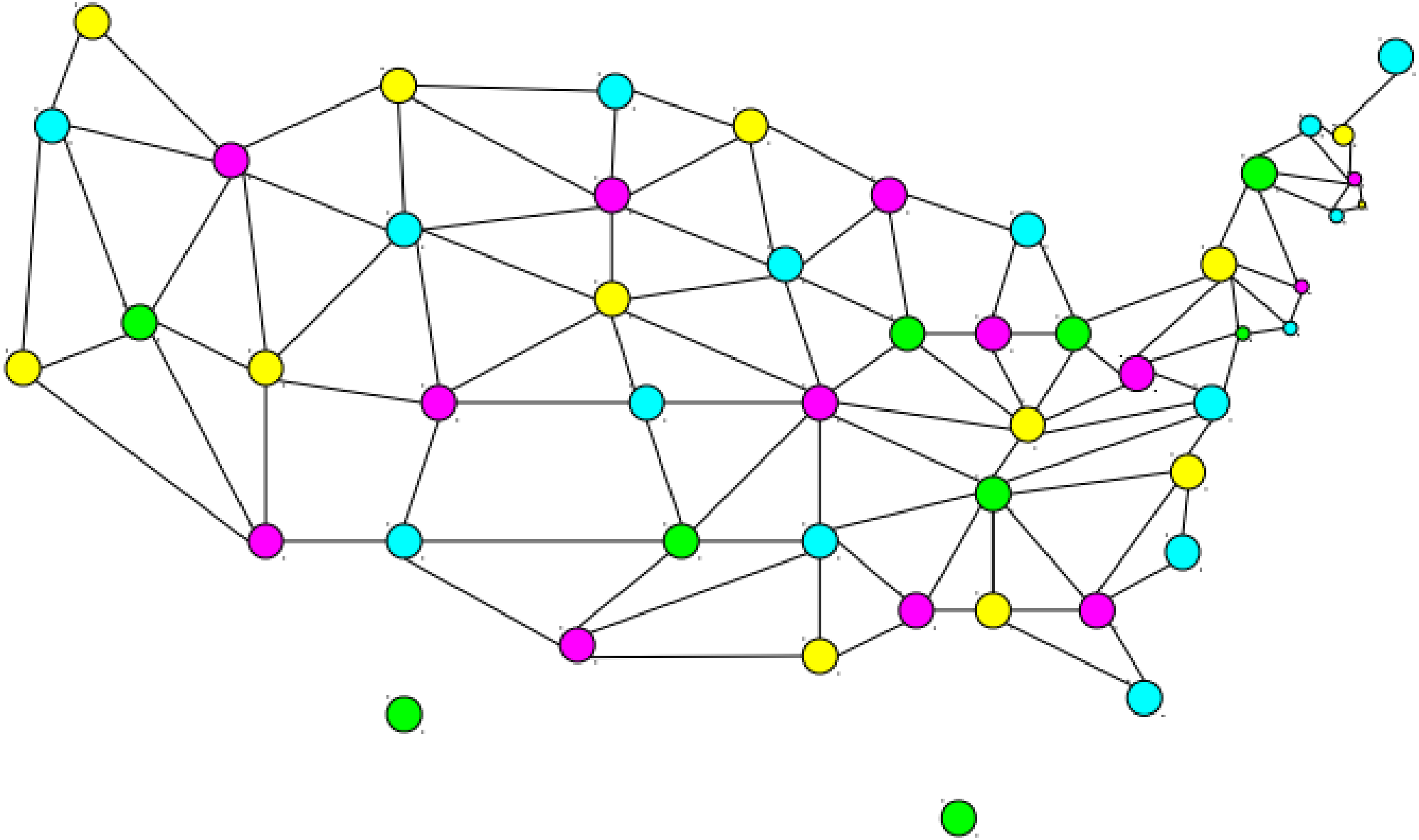
$$\forall u \in V.\ \forall v \in V.\ (\{u, v\} \in E \to f(u) \neq f(v))$$

- 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 *χρώμα*, meaning "color."

Graph Coloring



Graph Coloring



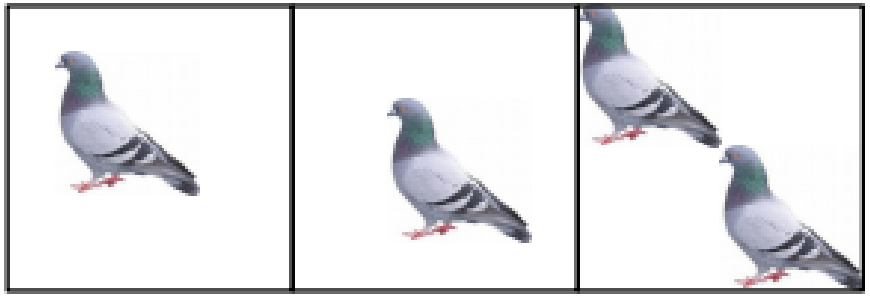
Theorem (Four-Color Theorem): Every planar graph is 4-colorable.

- **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.

The Pigeonhole Principle

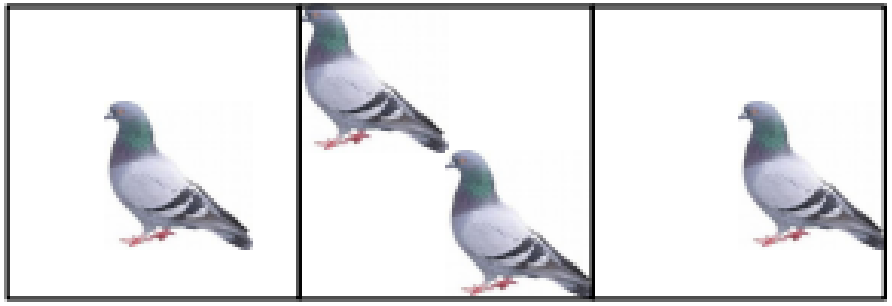
The Pigeonhole Principle

- ***Theorem (The Pigeonhole Principle):*** If m objects are distributed into n bins and $m > n$, then at least one bin will contain at least two objects.



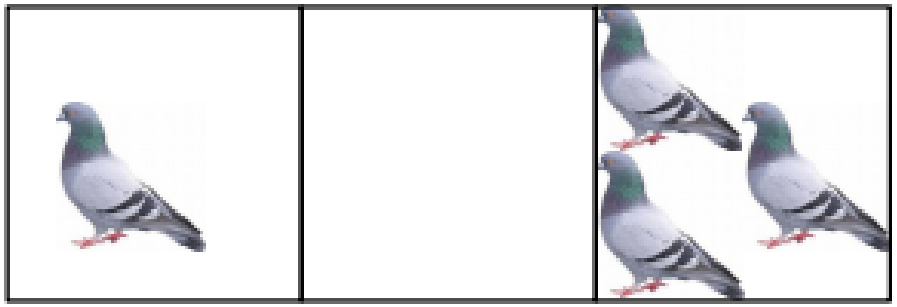
The Pigeonhole Principle

- ***Theorem (The Pigeonhole Principle):*** If m objects are distributed into n bins and $m > n$, then at least one bin will contain at least two objects.



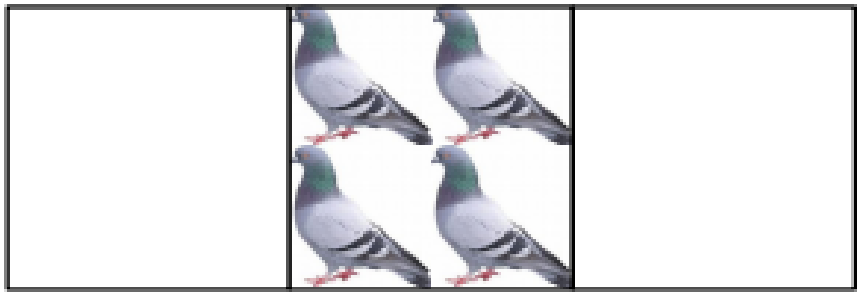
The Pigeonhole Principle

- ***Theorem (The Pigeonhole Principle):*** If m objects are distributed into n bins and $m > n$, then at least one bin will contain at least two objects.

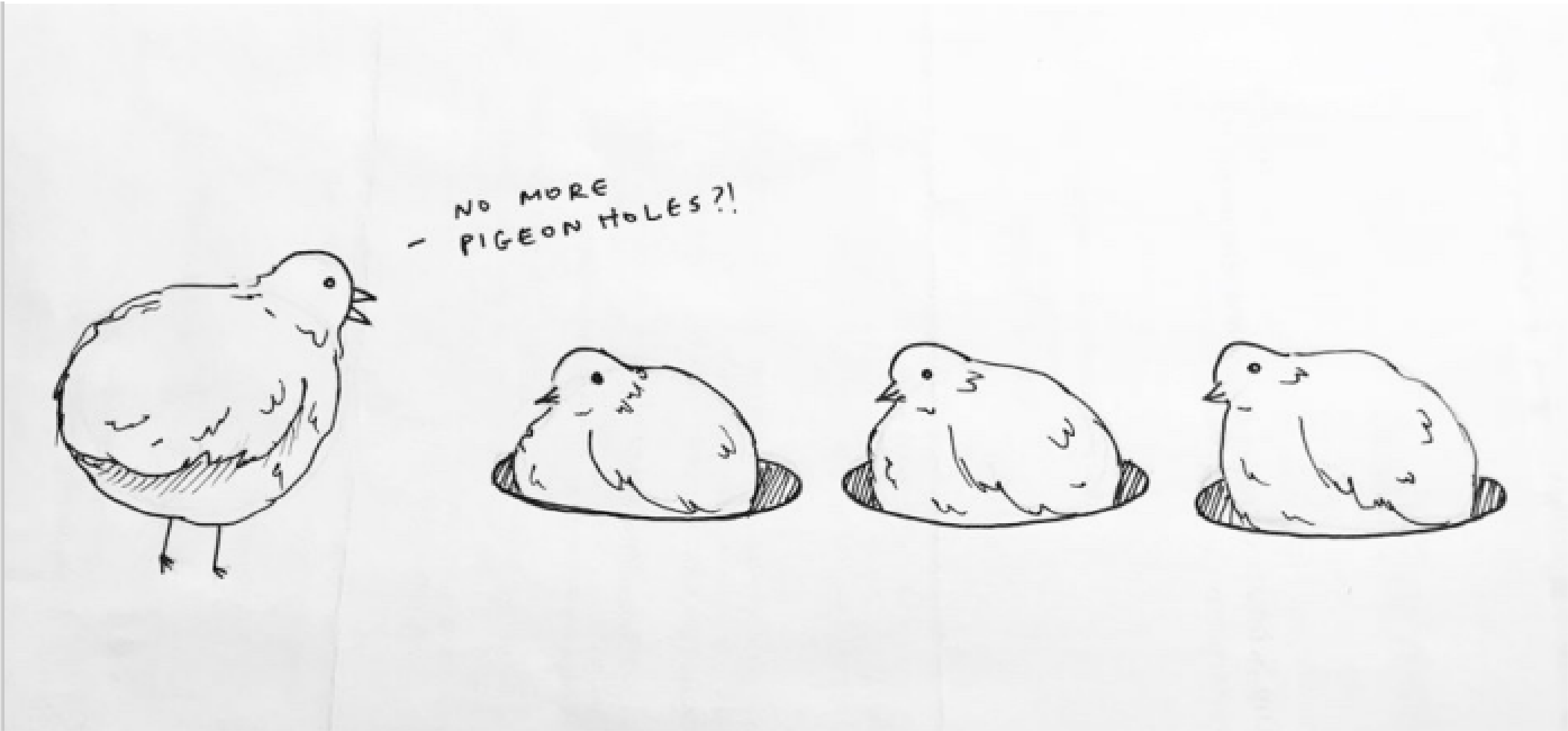


The Pigeonhole Principle

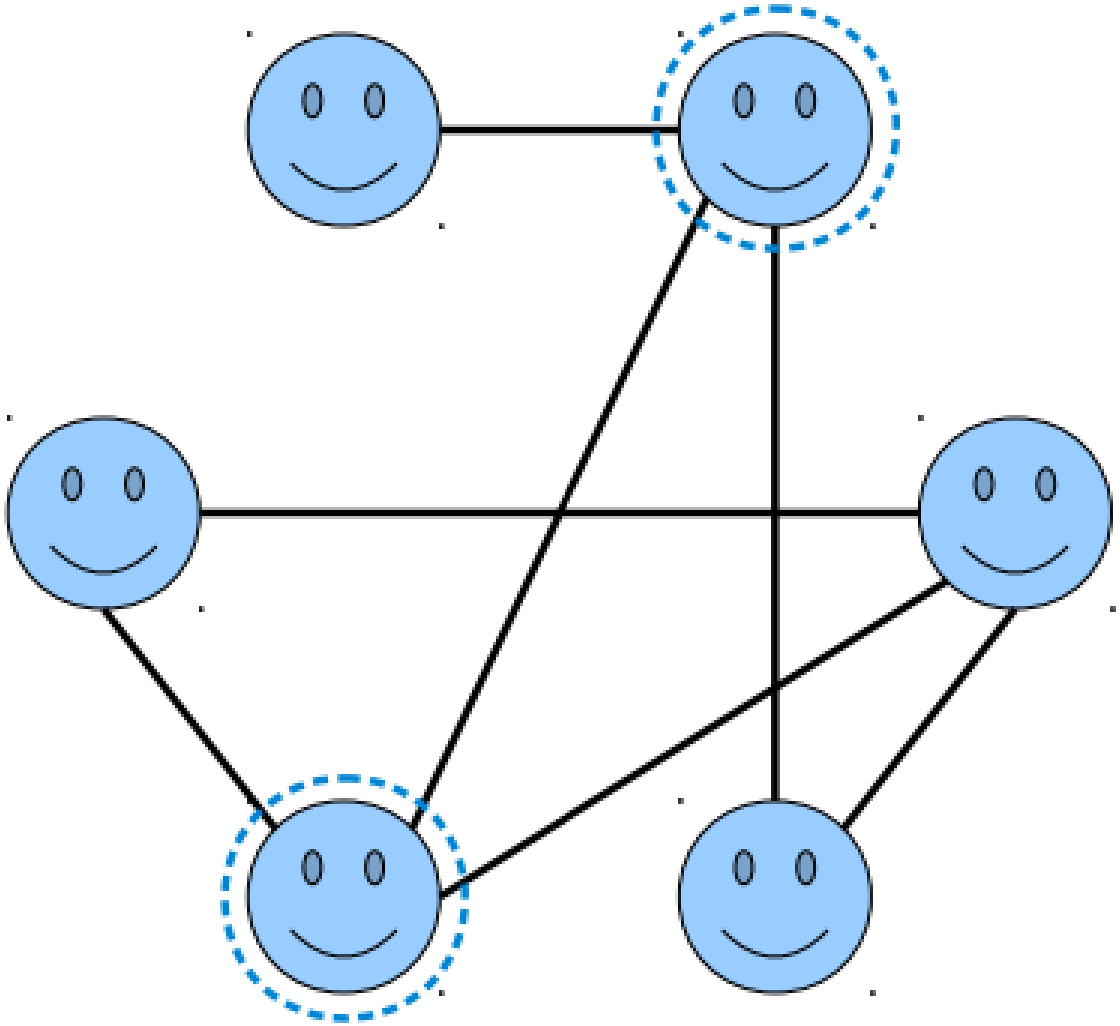
- ***Theorem (The Pigeonhole Principle):*** If m objects are distributed into n bins and $m > n$, then at least one bin will contain at least two objects.

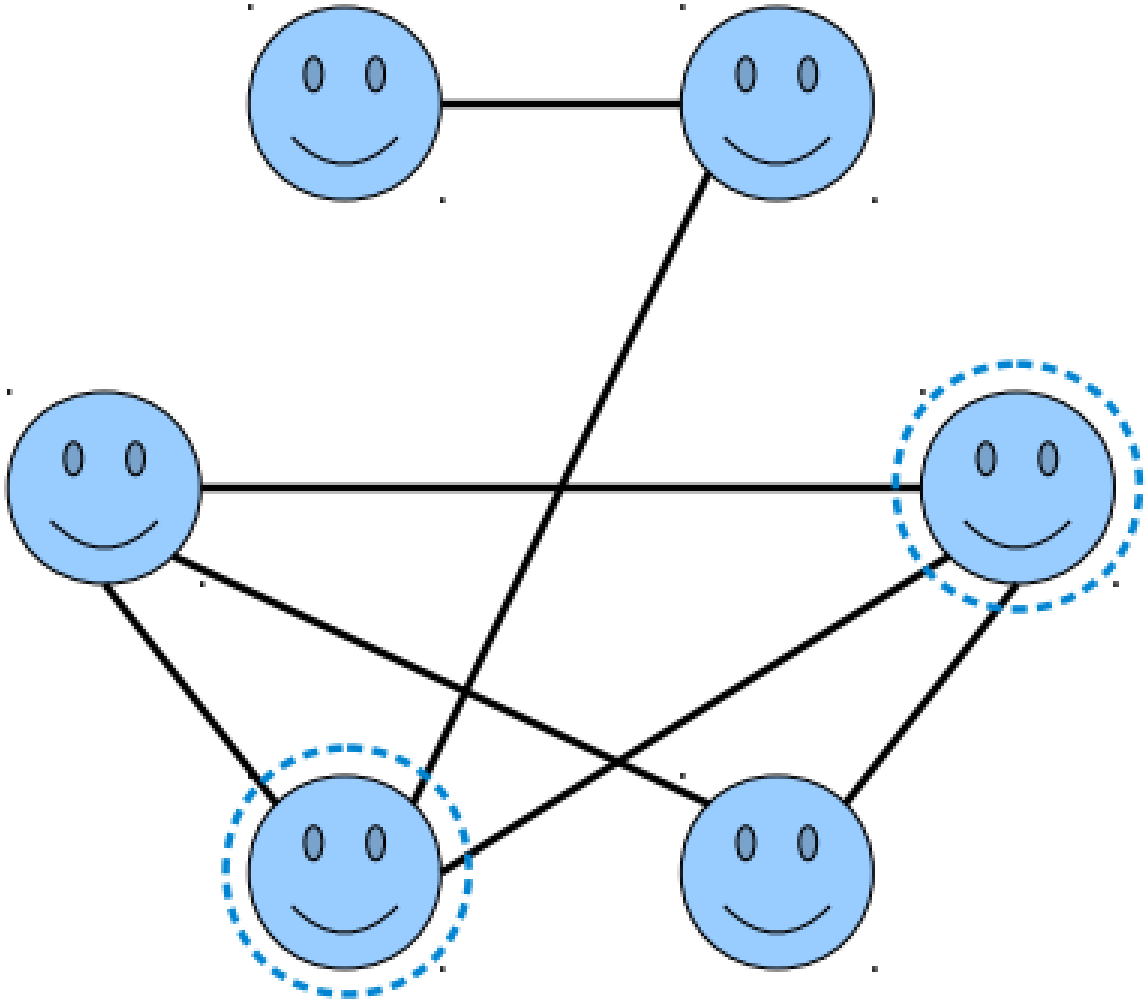


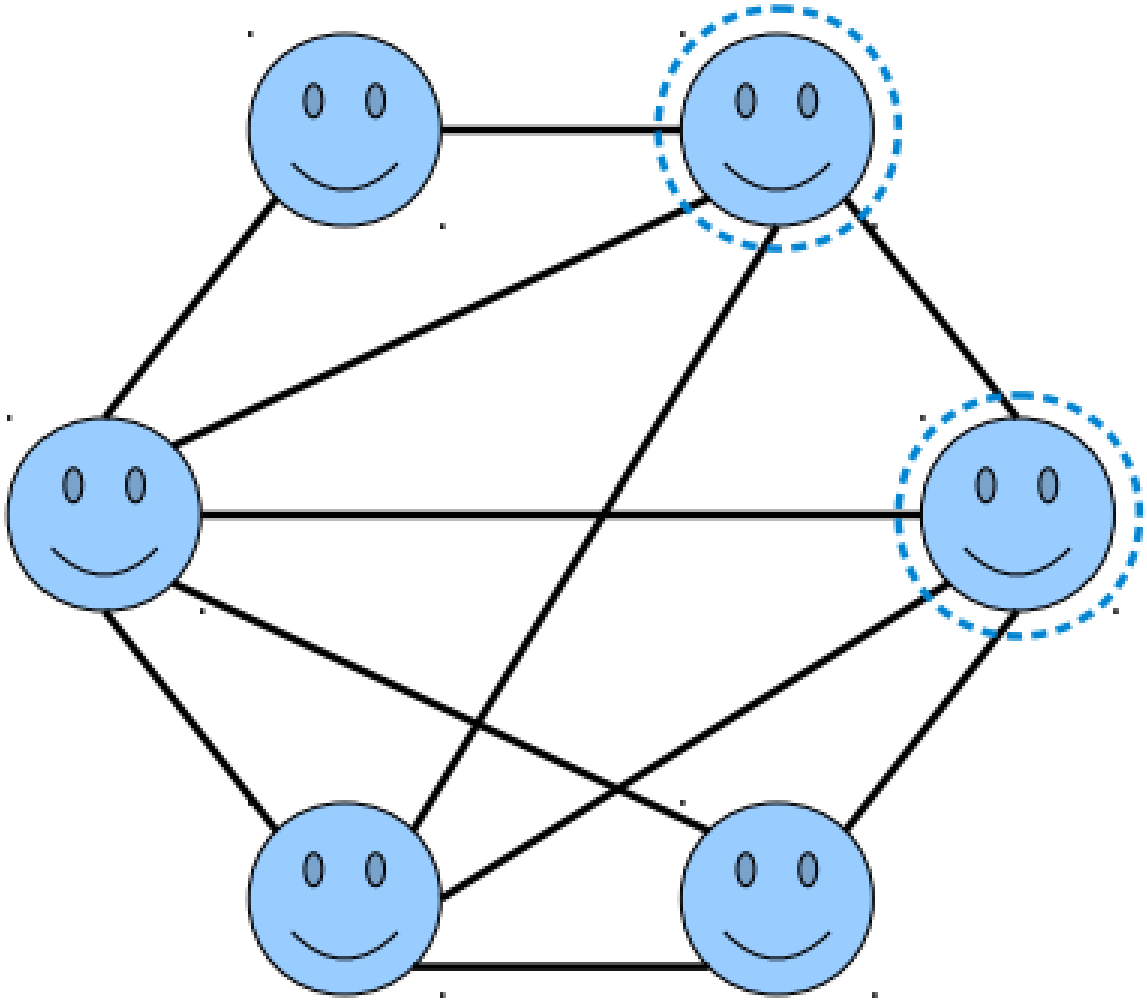
$$m = 4,\ n = 3$$



Pigeonhole Principle Party Tricks

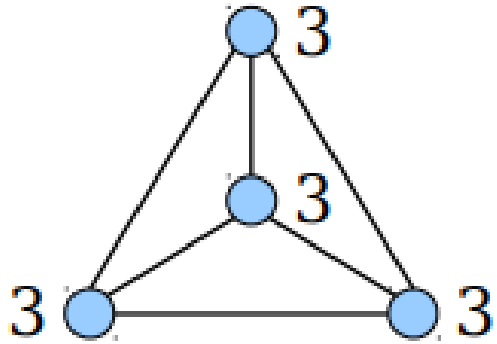




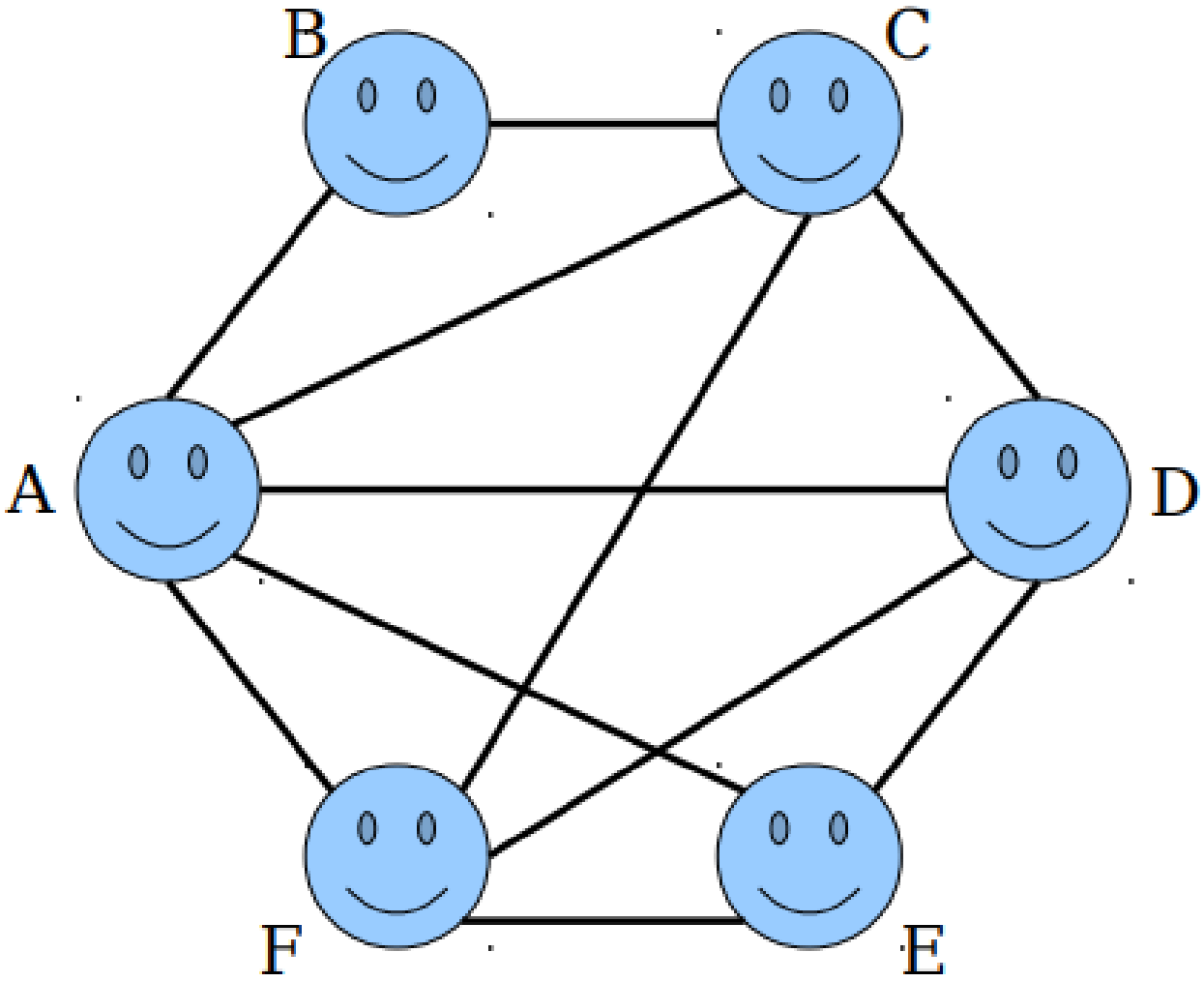


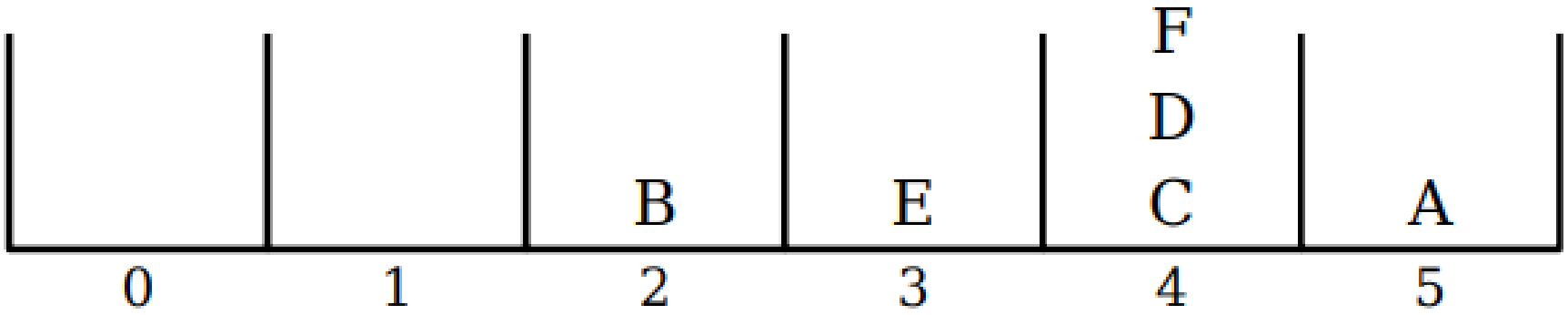
Degrees

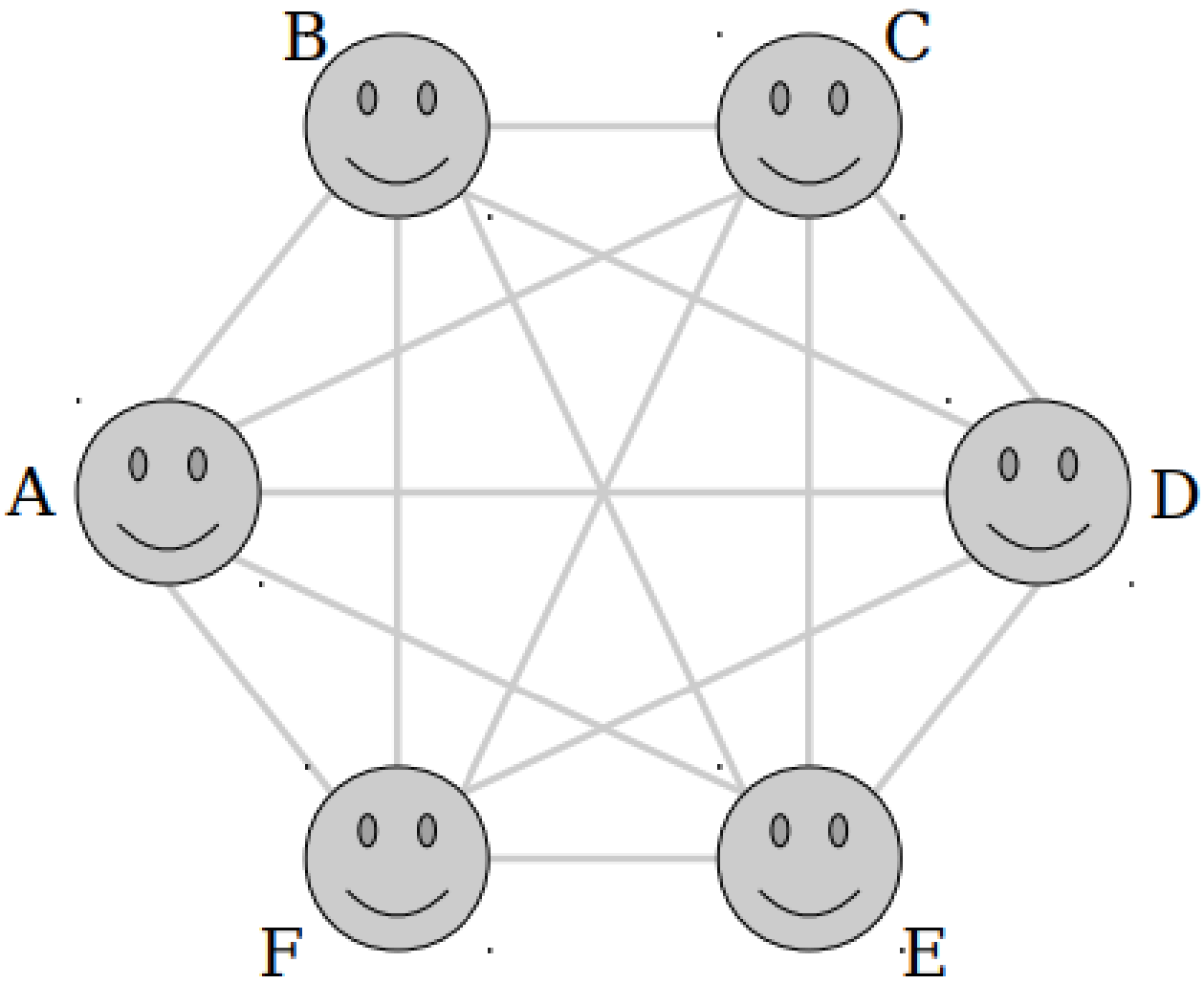
- The ***degree*** of a node v in a graph is the number of nodes that v is adjacent to.

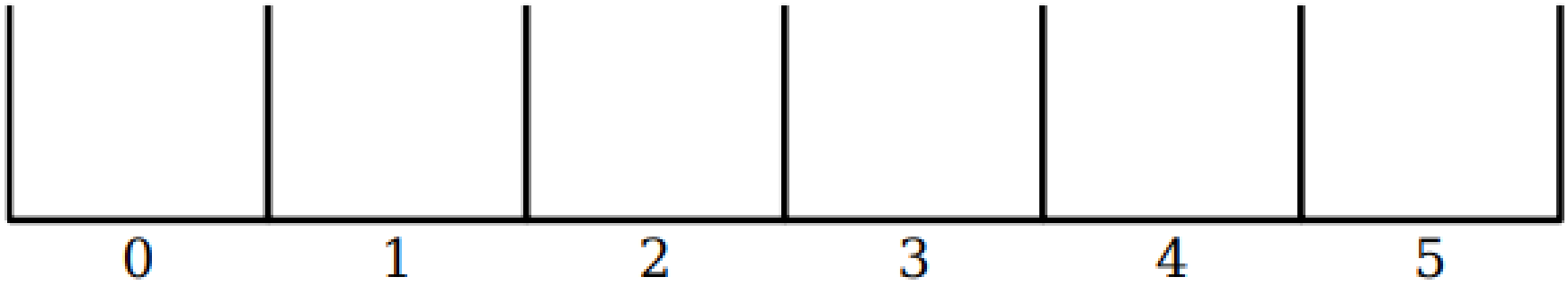


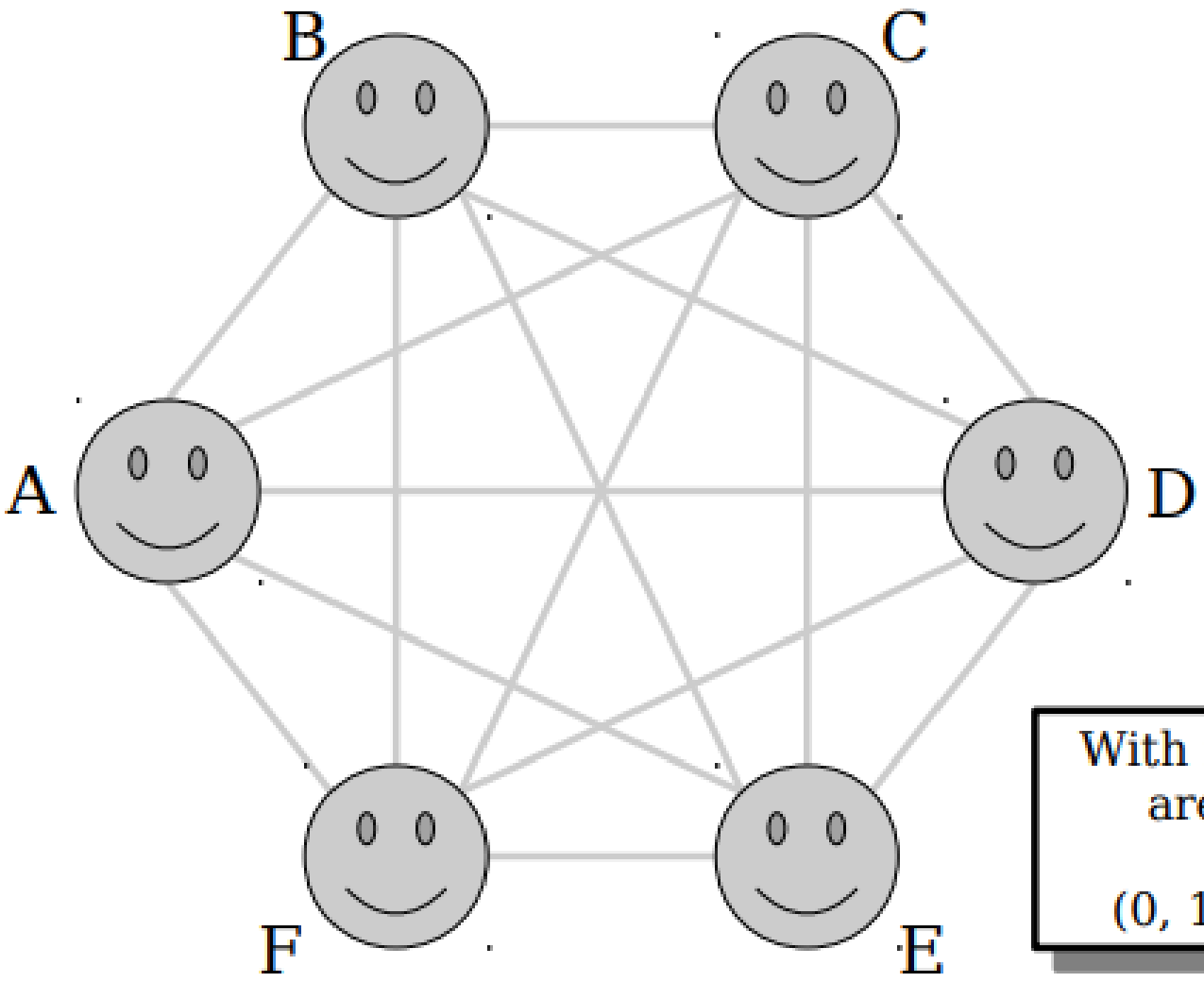
- ***Theorem:*** Every graph with at least two nodes has at least two nodes with the same degree.
  - Equivalently: at any party with at least two people, there are at least two people with the same number of Facebook friends at the party.


B
C
A
D
F
E

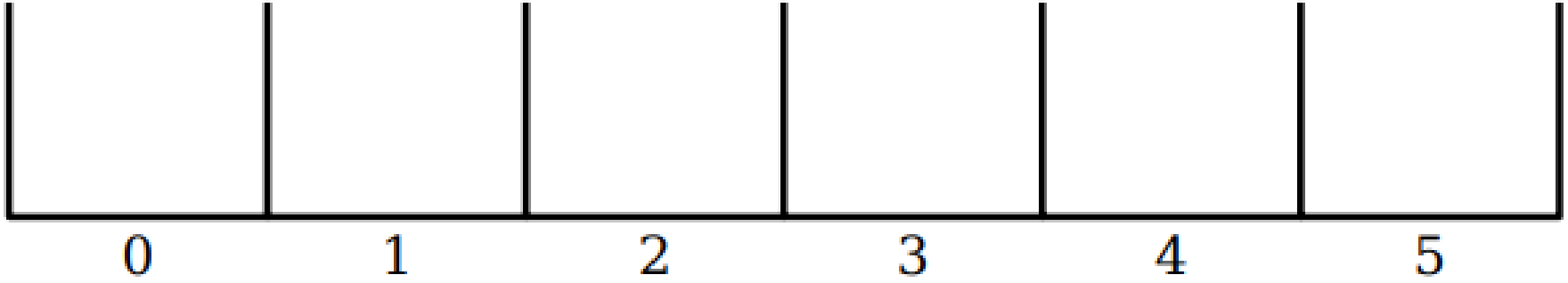

F
D
B
E
C
A
0
1
2
3
4
5

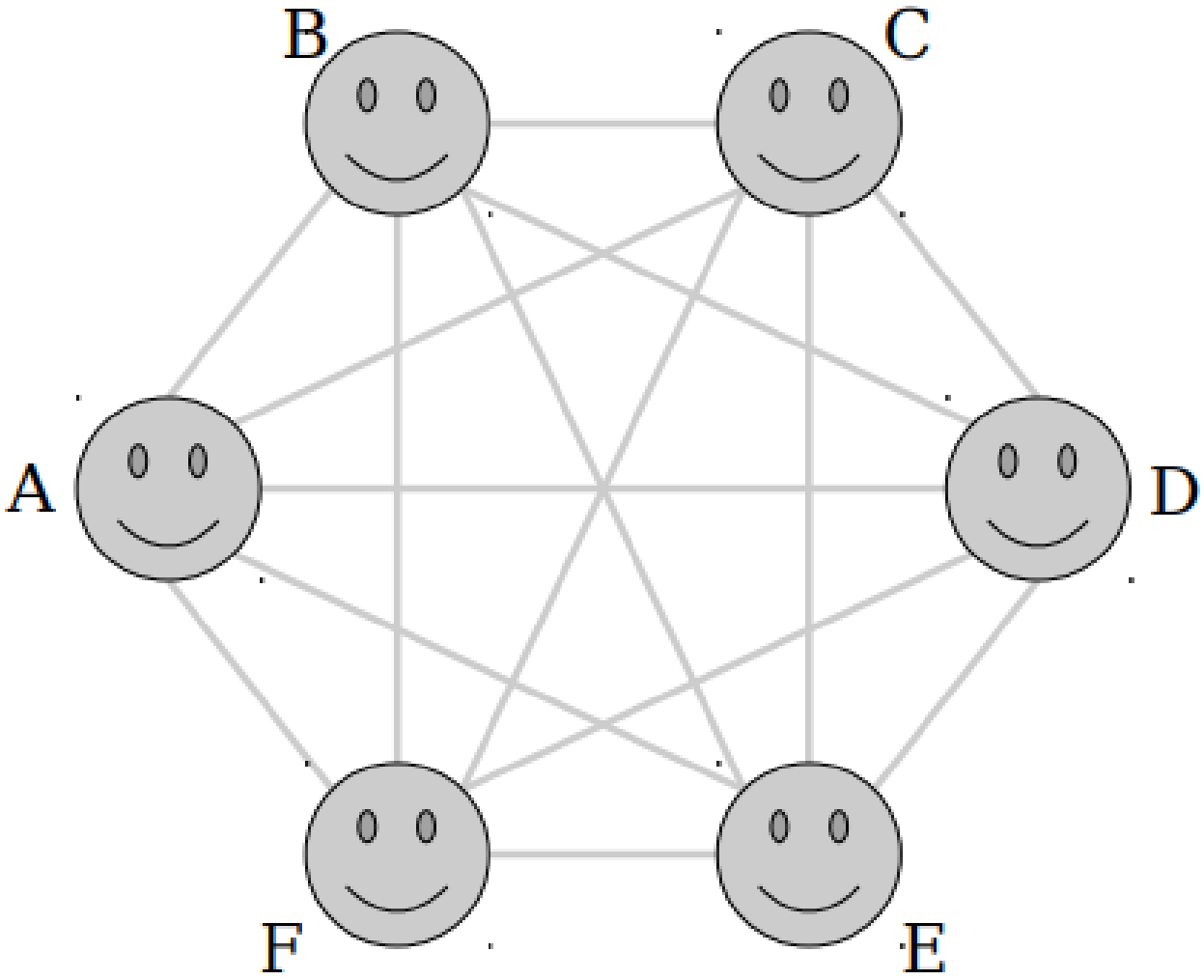

B
C
A
D
F
E

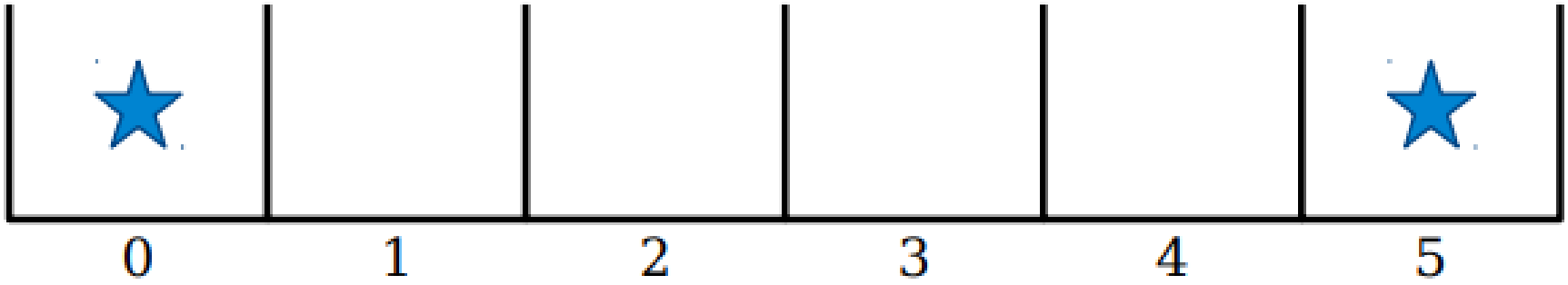

0
1
2
3
4
5

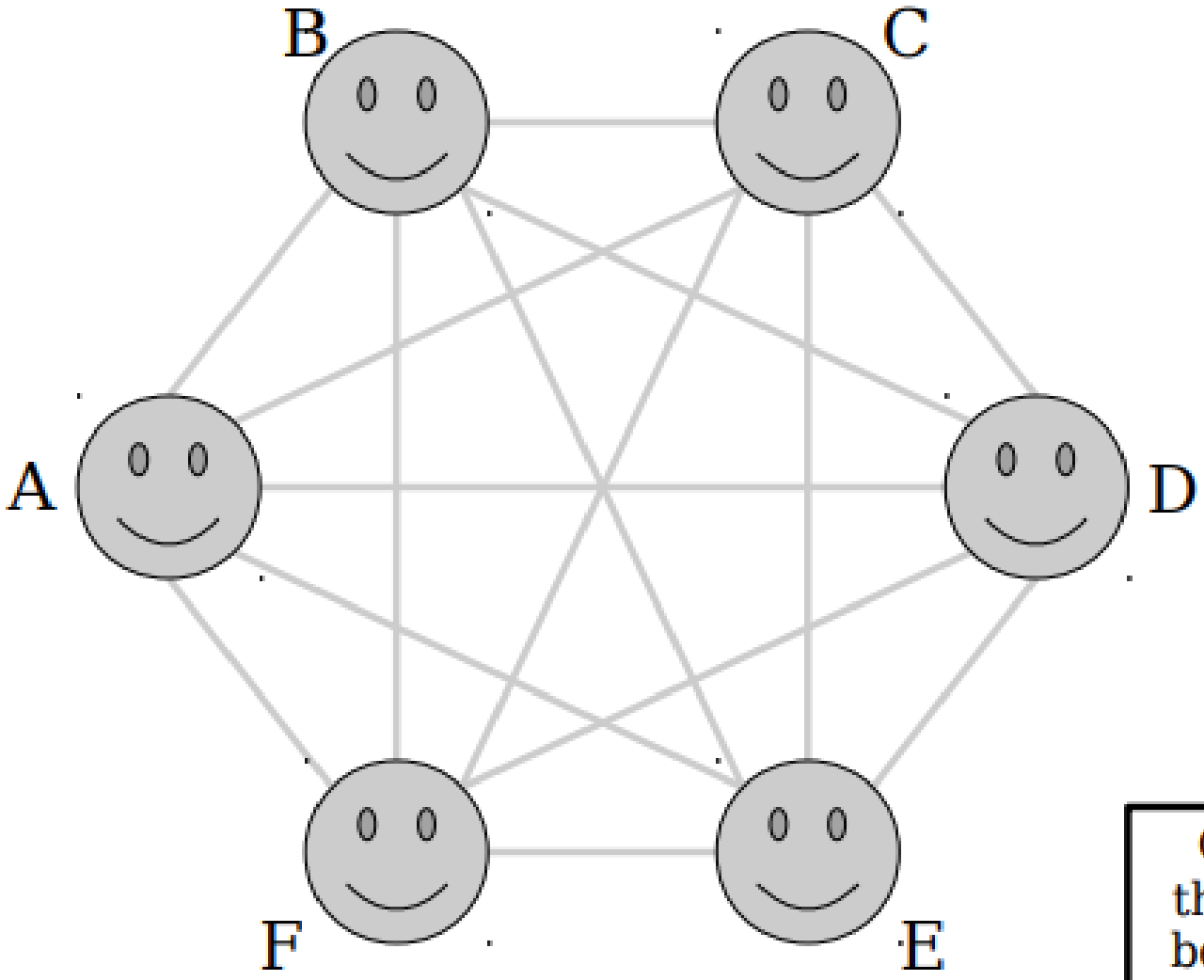


With n nodes, there are n possible degrees $(0, 1, 2, \ldots, n - 1)$

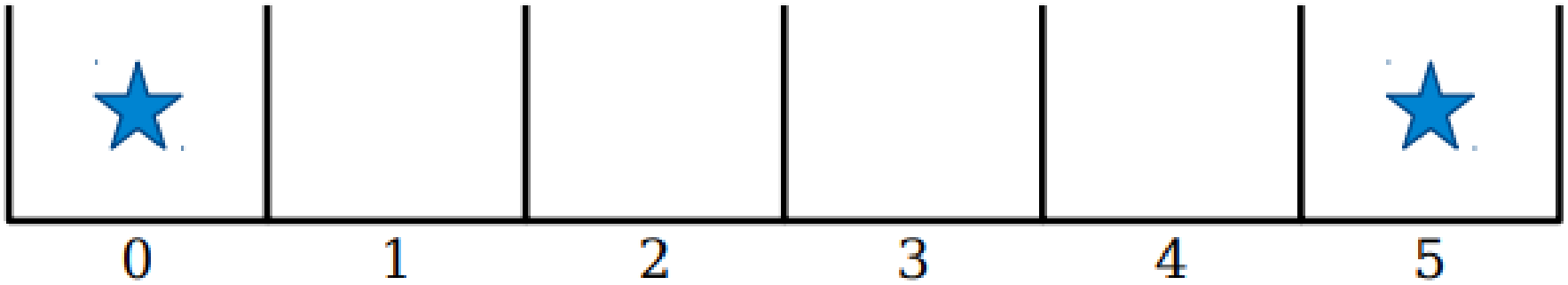


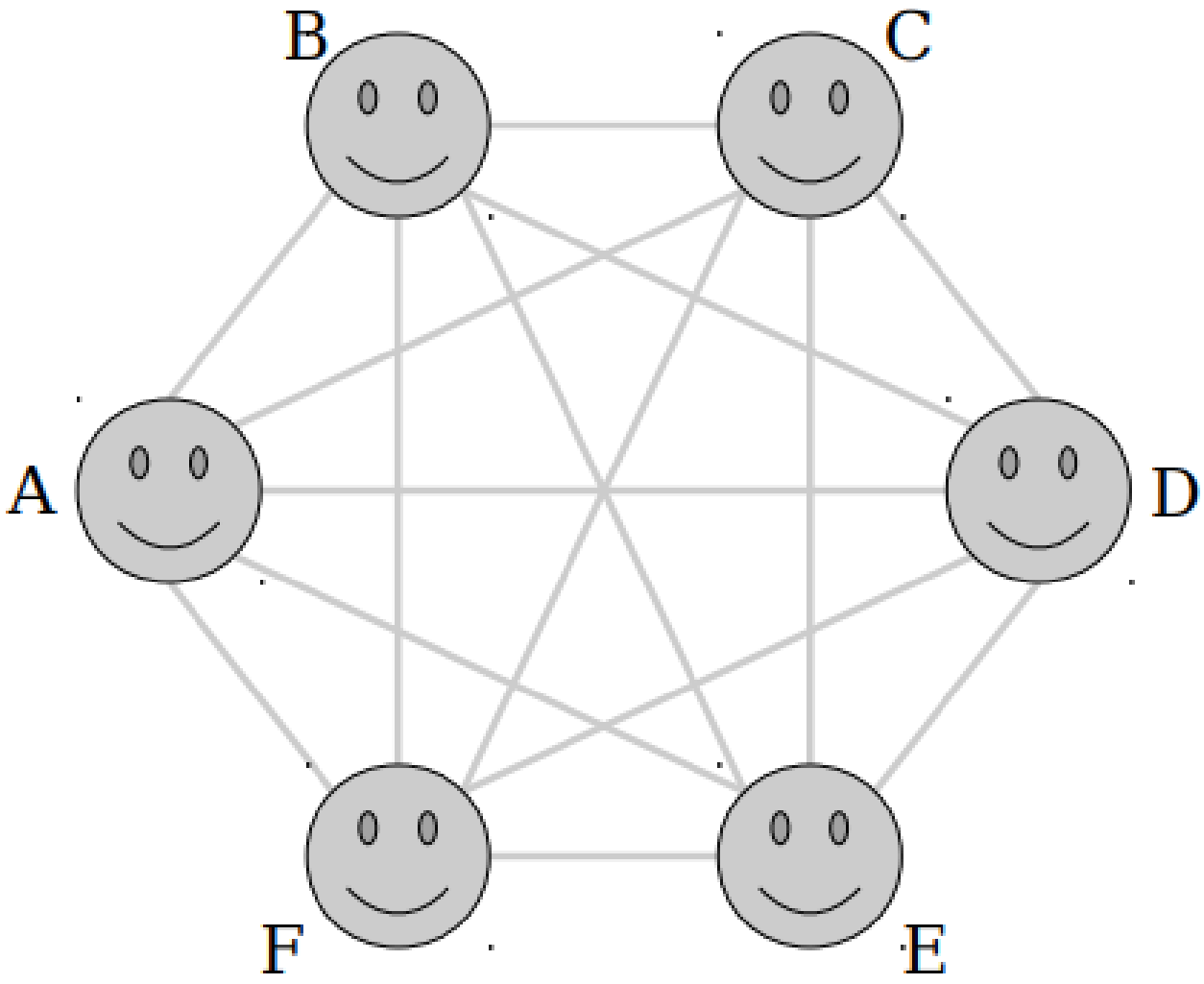

B
C
A
D
F
E

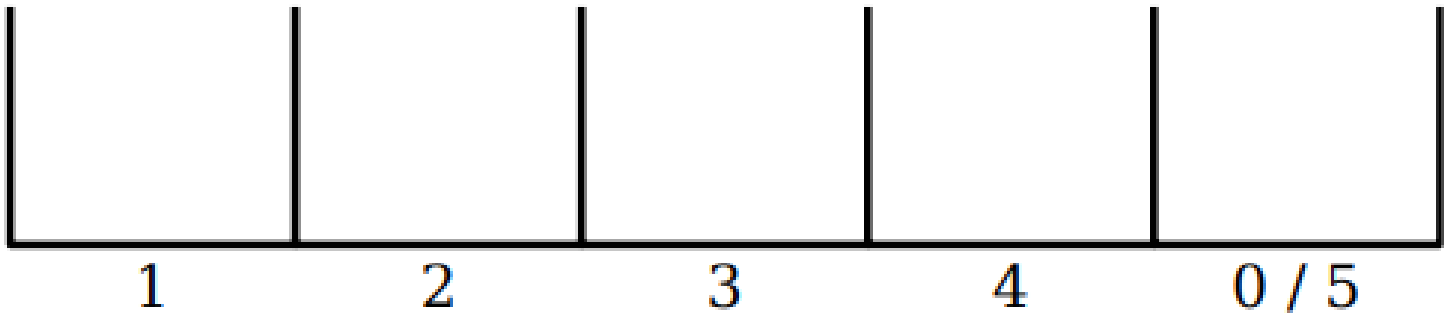

0
1
2
3
4
5



Can both of these buckets be nonempty?




B
C
A
D
F
E


1
2
3
4
0 / 5

Theorem: In any graph with at least two nodes, there are at least two nodes of the same degree.

Theorem: In any graph with at least two nodes, there are at least two nodes of the same degree.

Proof 1: Let G be a graph with $n \geq 2$ nodes.

Theorem: In any graph with at least two nodes, there are at least two nodes of the same degree.

Proof 1: Let G be a graph with $n \geq 2$ nodes. There are n possible choices for the degrees of nodes in G, namely, 0, 1, 2, ..., and $n - 1$.

Theorem: In any graph with at least two nodes, there are at least two nodes of the same degree.

Proof 1: Let G be a graph with $n \geq 2$ nodes. There are n possible choices for the degrees of nodes in G, namely, 0, 1, 2, ..., and $n - 1$.

We claim that G cannot simultaneously have a node u of degree 0 and a node v of degree $n - 1$:

Theorem: In any graph with at least two nodes, there are at least two nodes of the same degree.

Proof 1: Let G be a graph with $n \geq 2$ nodes. There are n possible choices for the degrees of nodes in G, namely, 0, 1, 2, ..., and $n - 1$.

We claim that G cannot simultaneously have a node u of degree 0 and a node v of degree $n - 1$: if there were such nodes, then node u would be adjacent to no other nodes and node v would be adjacent to all other nodes, including u.

Theorem: In any graph with at least two nodes, there are at least two nodes of the same degree.

Proof 1: Let G be a graph with $n \geq 2$ nodes. There are n possible choices for the degrees of nodes in G, namely, 0, 1, 2, ..., and $n - 1$.

We claim that G cannot simultaneously have a node u of degree 0 and a node v of degree $n - 1$: if there were such nodes, then node u would be adjacent to no other nodes and node v would be adjacent to all other nodes, including u. (Note that u and v must be different nodes, since v has degree at least 1 and u has degree 0.)

Theorem: In any graph with at least two nodes, there are at least two nodes of the same degree.

Proof 1: Let G be a graph with $n \geq 2$ nodes. There are n possible choices for the degrees of nodes in G, namely, 0, 1, 2, ..., and $n - 1$.

We claim that G cannot simultaneously have a node u of degree 0 and a node v of degree $n - 1$: if there were such nodes, then node u would be adjacent to no other nodes and node v would be adjacent to all other nodes, including u. (Note that u and v must be different nodes, since v has degree at least 1 and u has degree 0.)

We therefore see that the possible options for degrees of nodes in G are either drawn from 0, 1, ..., $n - 2$ or from 1, 2, ..., $n - 1$.

Theorem: In any graph with at least two nodes, there are at least two nodes of the same degree.

Proof 1: Let G be a graph with $n \geq 2$ nodes. There are n possible choices for the degrees of nodes in G, namely, 0, 1, 2, ..., and $n - 1$.

We claim that G cannot simultaneously have a node u of degree 0 and a node v of degree $n - 1$: if there were such nodes, then node u would be adjacent to no other nodes and node v would be adjacent to all other nodes, including u. (Note that u and v must be different nodes, since v has degree at least 1 and u has degree 0.)

We therefore see that the possible options for degrees of nodes in G are either drawn from 0, 1, ..., $n - 2$ or from 1, 2, ..., $n - 1$. In either case, there are n nodes and $n - 1$ possible degrees, so by the pigeonhole principle two nodes in G must have the same degree.

Theorem: In any graph with at least two nodes, there are at least two nodes of the same degree.

Proof 1: Let G be a graph with $n \geq 2$ nodes. There are n possible choices for the degrees of nodes in G, namely, 0, 1, 2, ..., and $n - 1$.

We claim that G cannot simultaneously have a node u of degree 0 and a node v of degree $n - 1$: if there were such nodes, then node u would be adjacent to no other nodes and node v would be adjacent to all other nodes, including u. (Note that u and v must be different nodes, since v has degree at least 1 and u has degree 0.)

We therefore see that the possible options for degrees of nodes in G are either drawn from 0, 1, ..., $n - 2$ or from 1, 2, ..., $n - 1$. In either case, there are n nodes and $n - 1$ possible degrees, so by the pigeonhole principle two nodes in G must have the same degree. ■

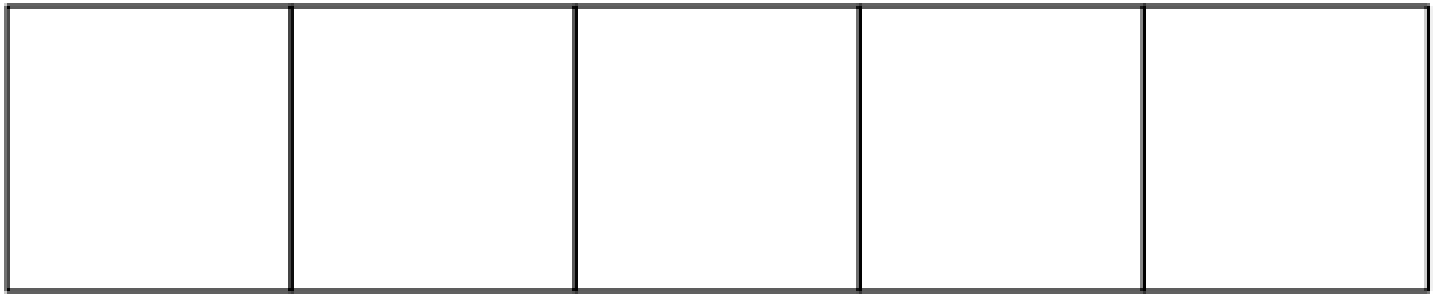
Theorem: In any graph with at least two nodes, there are at least two nodes of the same degree.

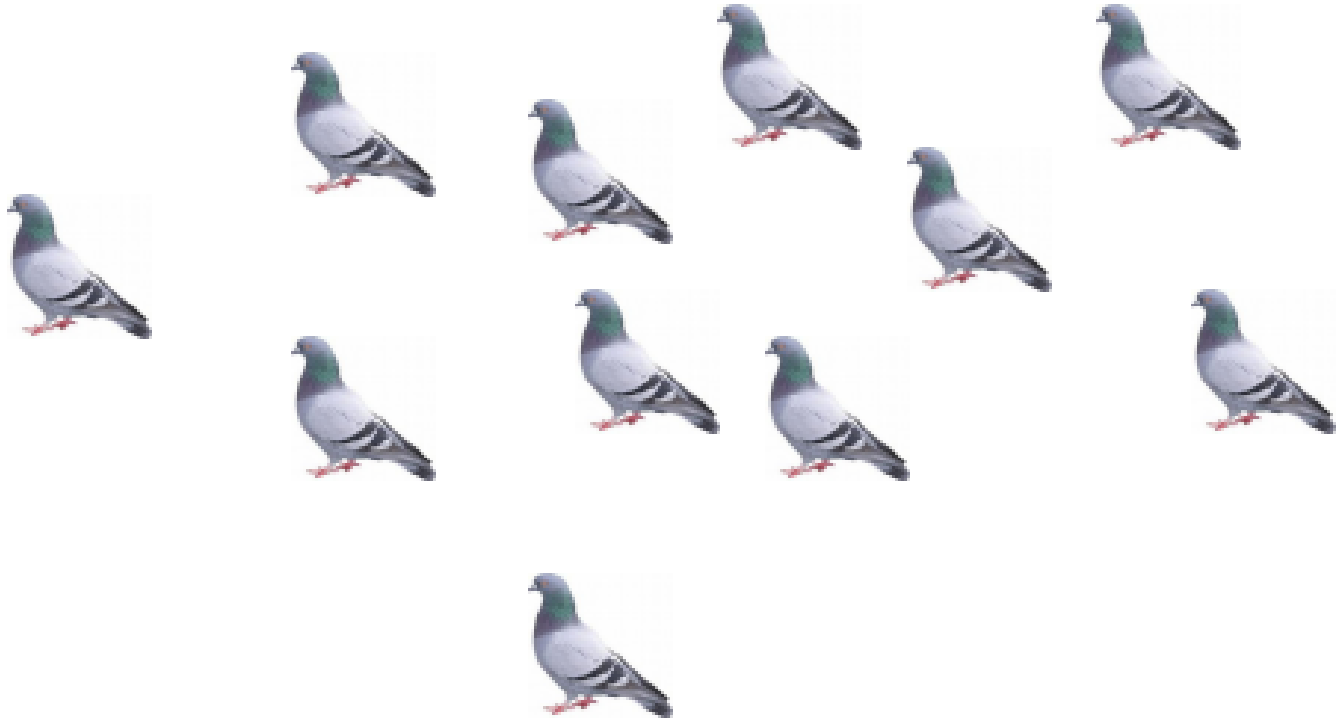
Proof 2: Assume for the sake of contradiction that there is a graph G with $n \geq 2$ nodes where no two nodes have the same degree. There are n possible choices for the degrees of nodes in G, namely $0, 1, 2, \ldots, n - 1$, so this means that G must have exactly one node of each degree. However, this means that G has a node of degree 0 and a node of degree $n - 1$. (These can't be the same node, since $n \geq 2$.) This first node is adjacent to no other nodes, but this second node is adjacent to every other node, which is impossible.

We have reached a contradiction, so our assumption must have been wrong. Thus if G is a graph with at least two nodes, G must have at least two nodes of the same degree. ■

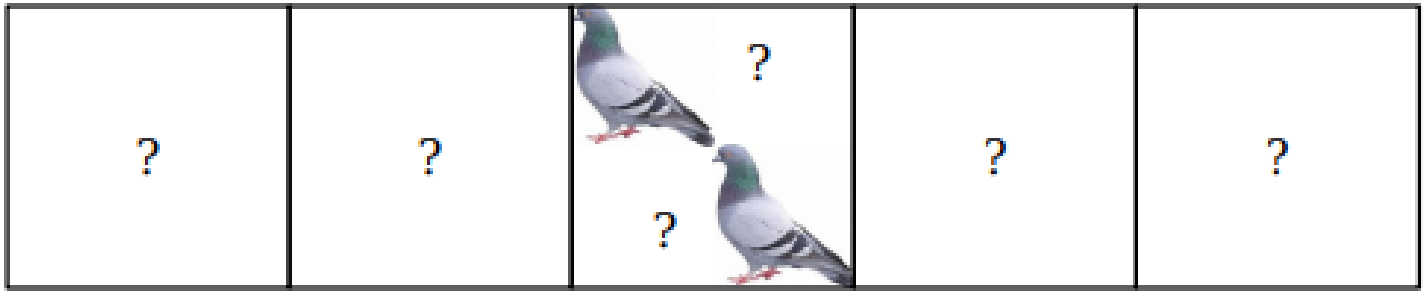
The Generalized Pigeonhole Principle

The Pigeonhole Principle

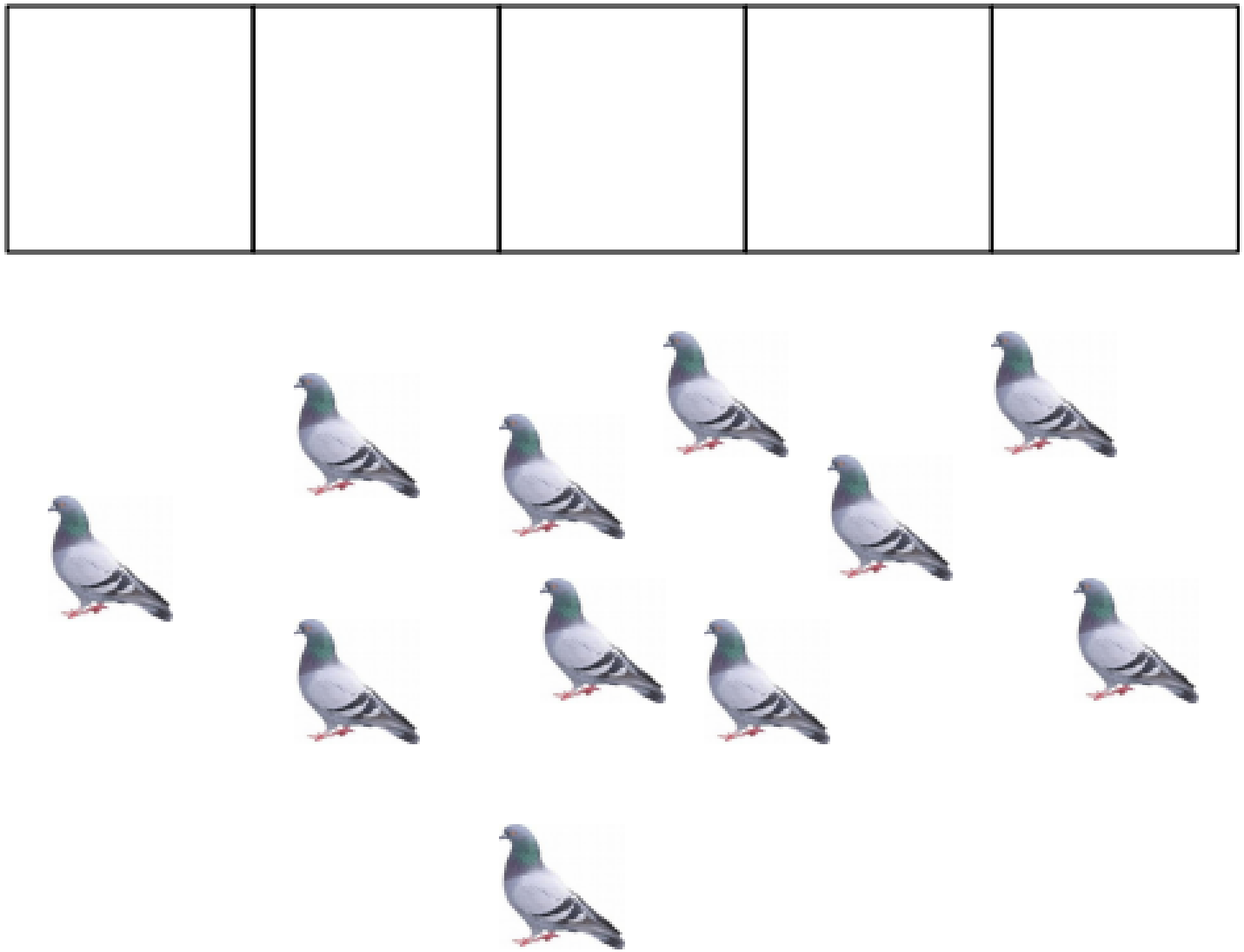




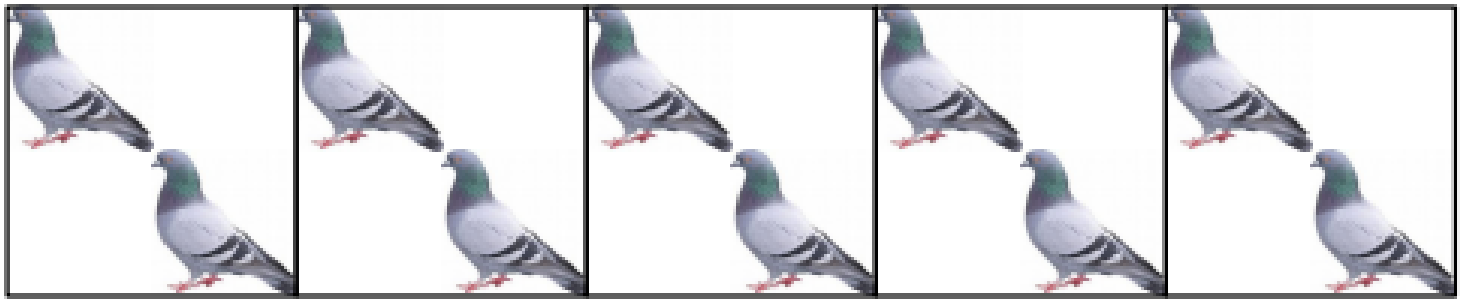
The Pigeonhole Principle



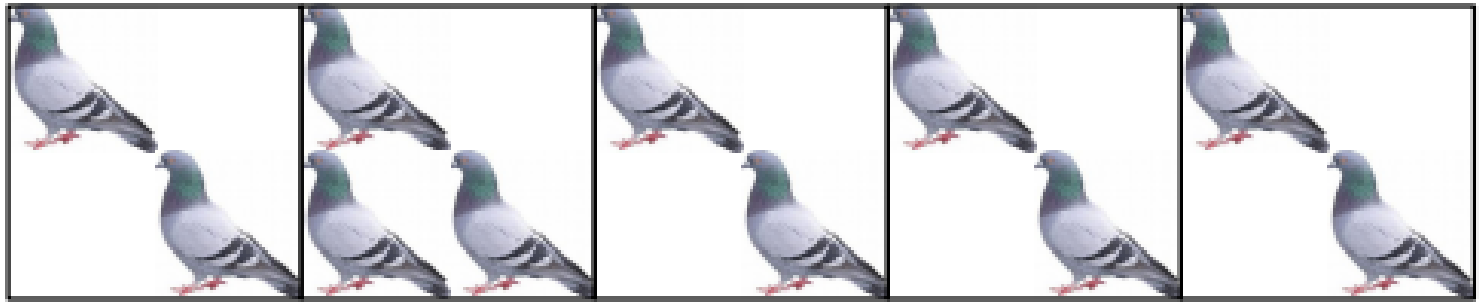
The Pigeonhole Principle



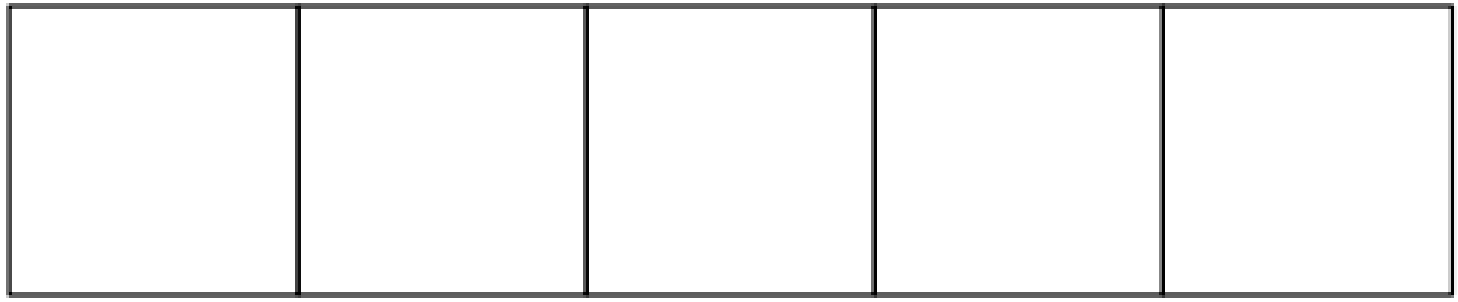
The Pigeonhole Principle

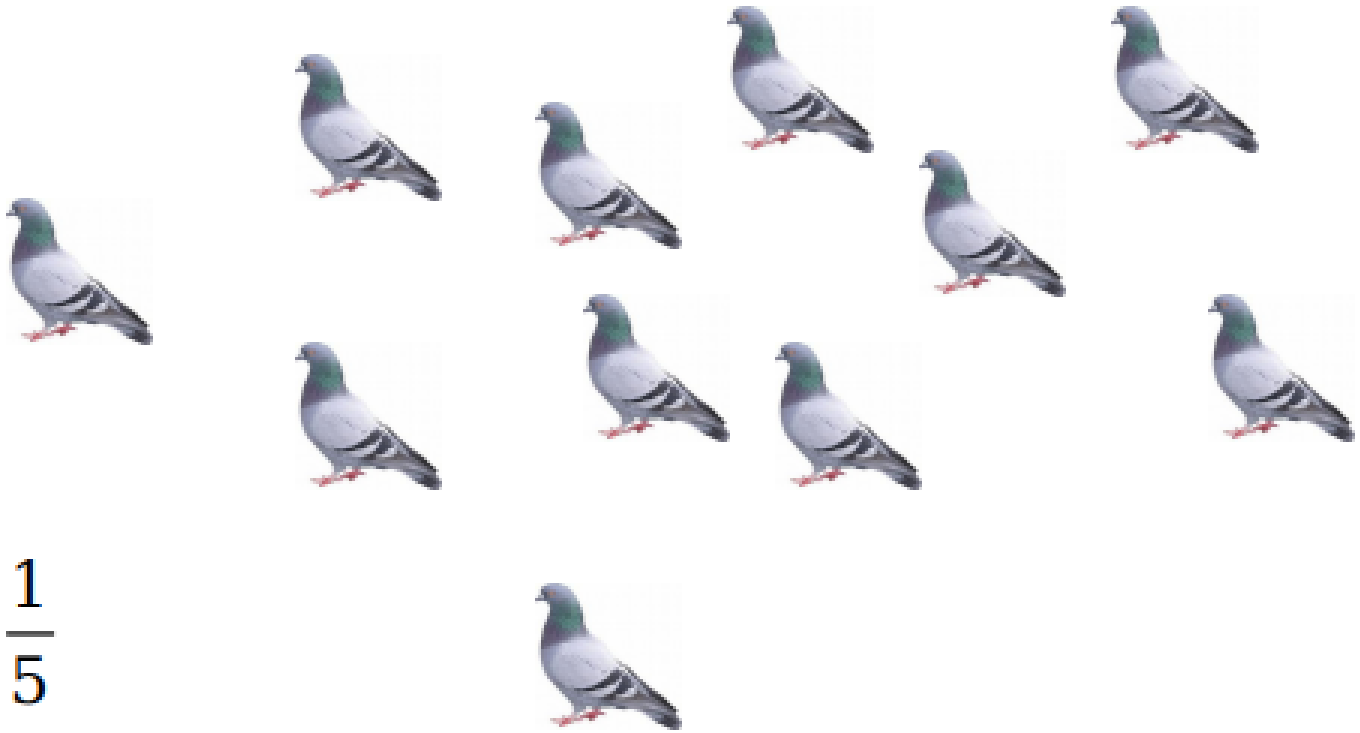


The Pigeonhole Principle



The Pigeonhole Principle



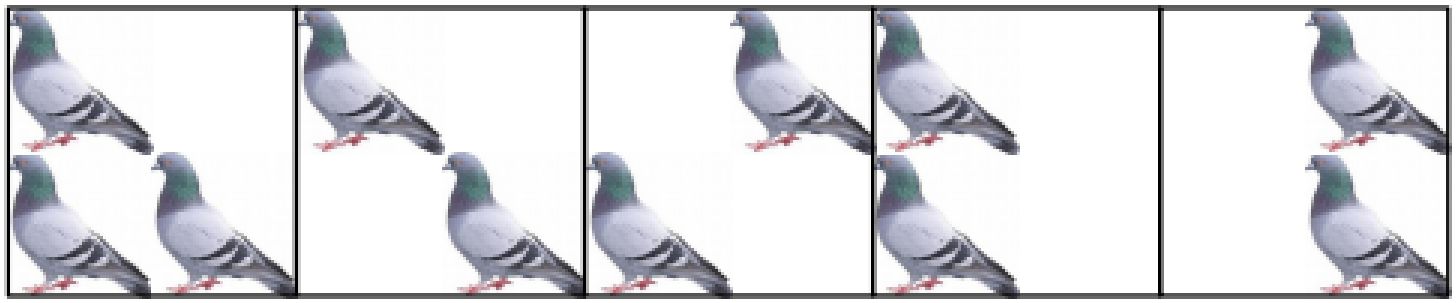


$$\frac{11}{5} = 2\frac{1}{5}$$

A More General Version

- The ***generalized pigeonhole principle*** says that if you distribute m objects into n bins, then
  - some bin will have at least $\lceil m/n \rceil$ objects in it, and
  - some bin will have at most $\lfloor m/n \rfloor$ objects in it.

$\lceil m/n \rceil$ means "m/n, rounded up."
$\lfloor m/n \rfloor$ means "m/n, rounded down."



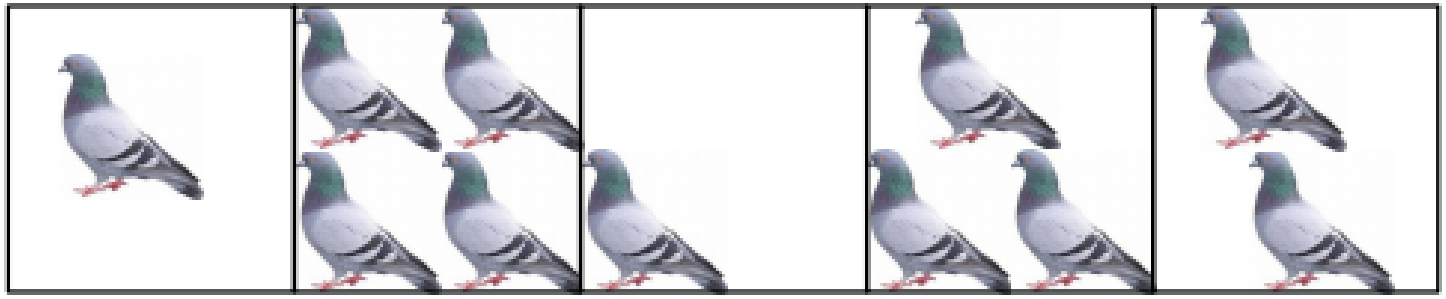
$m = 11$
$n = 5$

$\lceil m / n \rceil = 3$
$\lfloor m / n \rfloor = 2$

A More General Version

- The ***generalized pigeonhole principle*** says that if you distribute m objects into n bins, then
  - some bin will have at least $\lceil m/n \rceil$ objects in it, and
  - some bin will have at most $\lfloor m/n \rfloor$ objects in it.

$\lceil m/n \rceil$ means "m/n, rounded up."
$\lfloor m/n \rfloor$ means "m/n, rounded down."



$m = 11$
$n = 5$

$\lceil m / n \rceil = 3$
$\lfloor m / n \rfloor = 2$

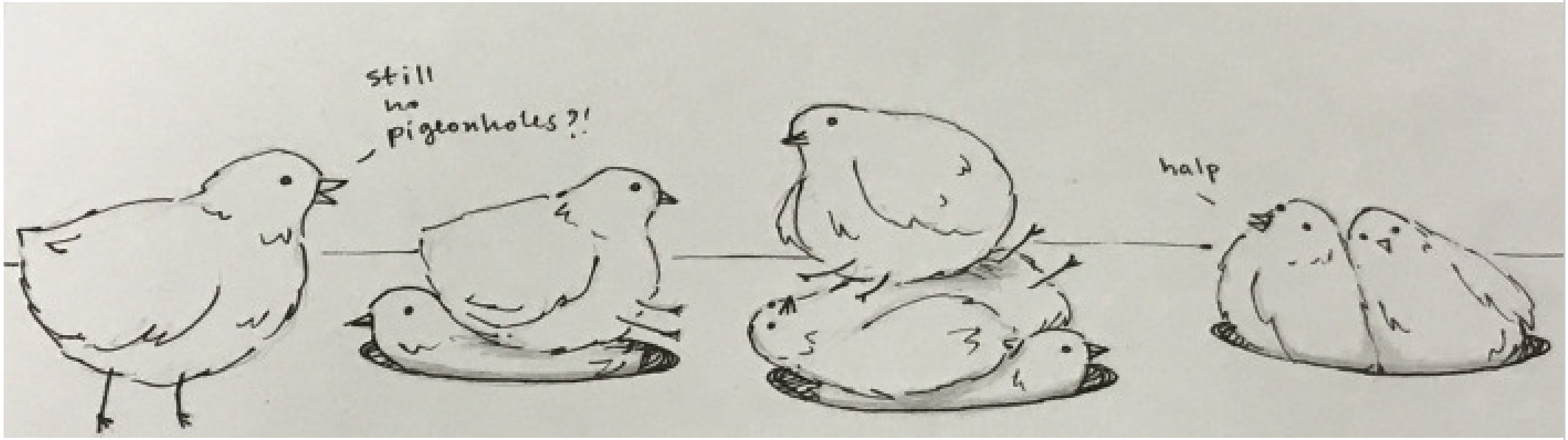
A More General Version

- The ***generalized pigeonhole principle*** says that if you distribute m objects into n bins, then
  - some bin will have at least $\lceil m/n \rceil$ objects in it, and
  - some bin will have at most $\lfloor m/n \rfloor$ objects in it.

$\lceil m/n \rceil$ means "m/n, rounded up."
$\lfloor m/n \rfloor$ means "m/n, rounded down."

$m = 11$
$n = 5$

$\lceil m / n \rceil = 3$
$\lfloor m / n \rfloor = 2$



$$m = 8,\ n = 3$$

Theorem: If m objects are distributed into $n > 0$ bins, then some bin will contain at least $\lceil m/n \rceil$ objects.

Proof: We will prove that if m objects are distributed into n bins, then some bin contains at least m/n objects. Since the number of objects in each bin is an integer, this will prove that some bin must contain at least $\lceil m/n \rceil$ objects.

To do this, we proceed by contradiction. Suppose that, for some m and n, there is a way to distribute m objects into n bins such that each bin contains fewer than m/n objects.

Number the bins 1, 2, 3, ..., n and let x_i denote the number of objects in bin i. Since there are m objects in total, we know that

$$m = x_1 + x_2 + \ldots + x_n.$$

Since each bin contains fewer than m/n objects, we see that $x_i < m/n$ for each i. Therefore, we have that

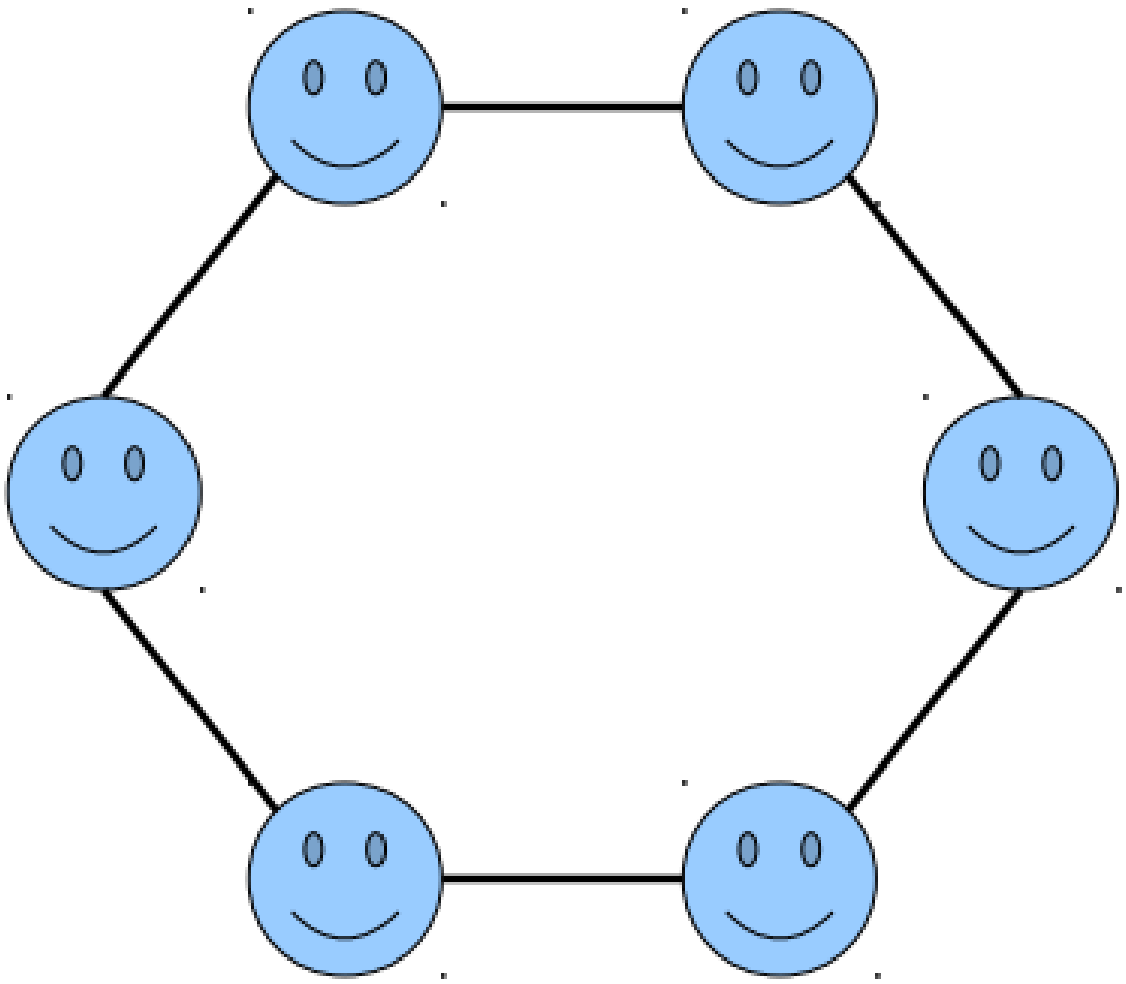
$$\begin{aligned} m &= x_1 + x_2 + \ldots + x_n \\ &< m/n + m/n + \ldots + m/n \quad (n \text{ times}) \\ &= m. \end{aligned}$$

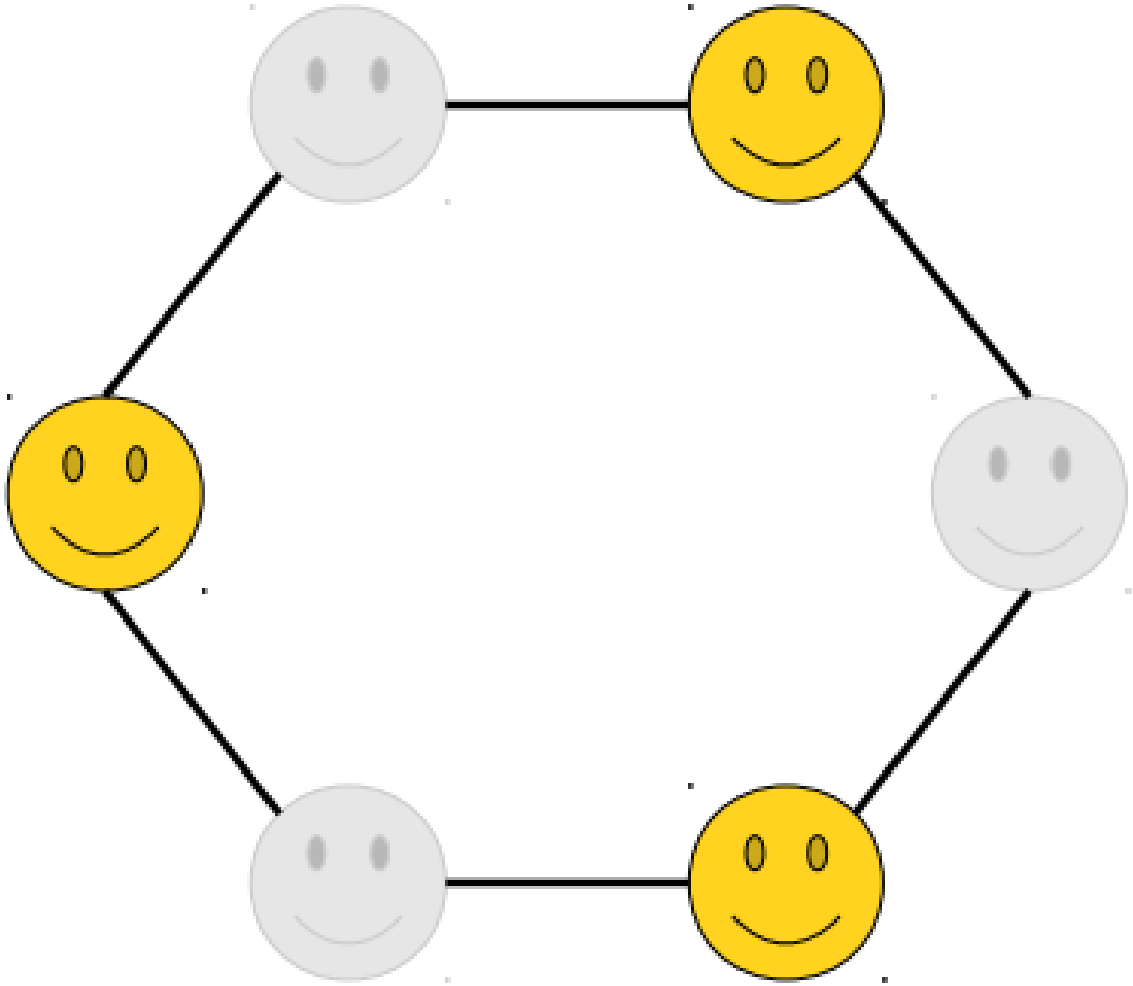
But this means that $m < m$, which is impossible. We have reached a contradiction, so our initial assumption must have been wrong. Therefore, if m objects are distributed into n bins, some bin must contain at least $\lceil m/n \rceil$ objects. ■

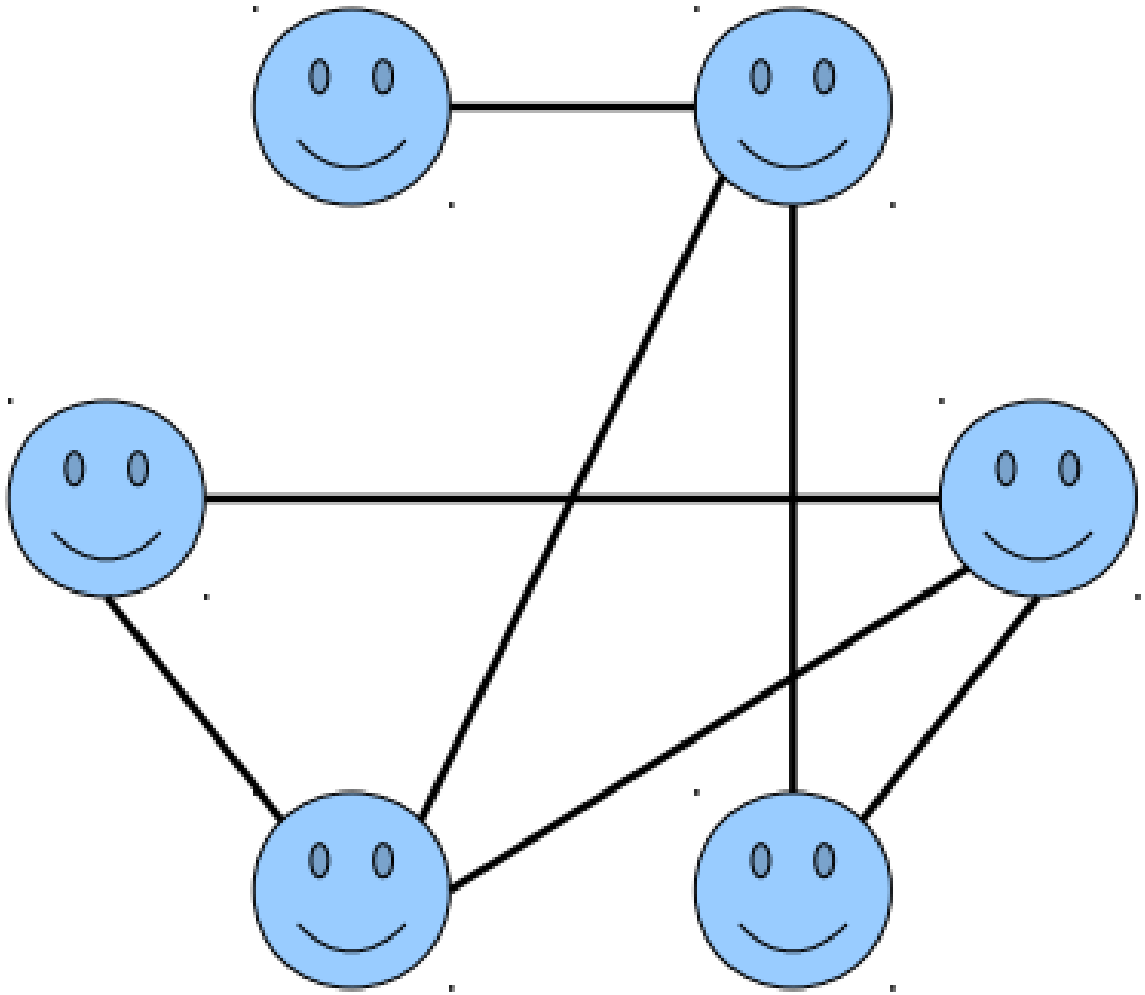
An Application: Friends and Strangers

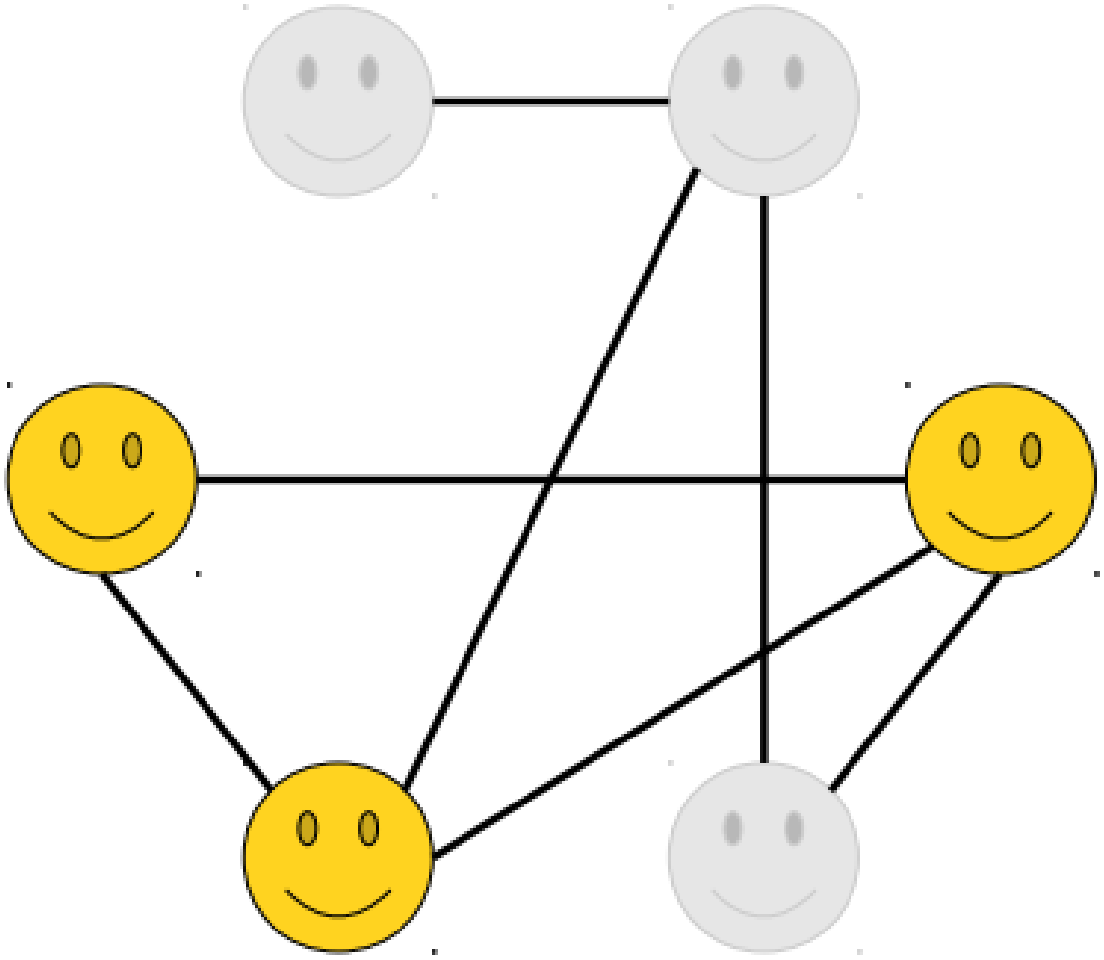
Friends and Strangers

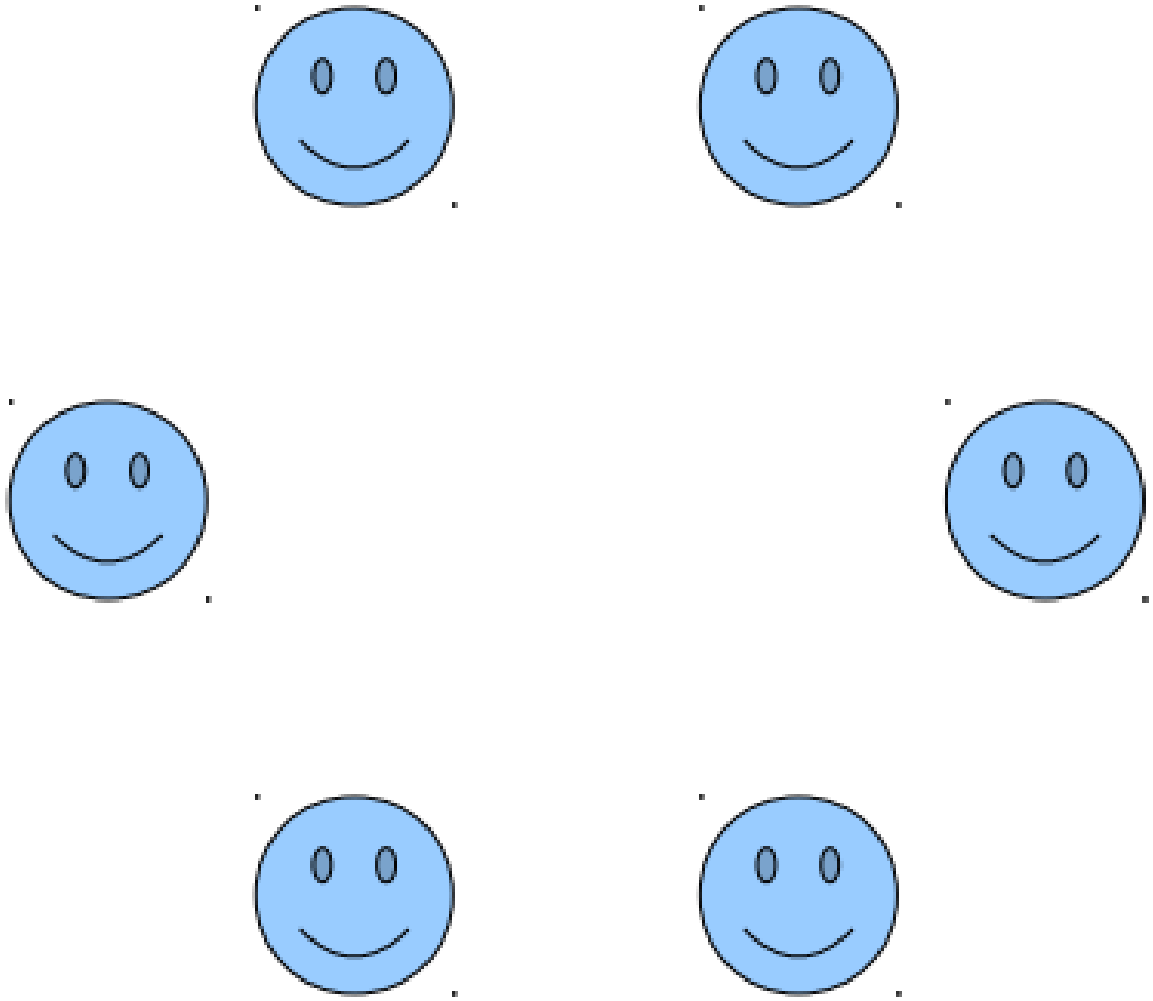
- Suppose you have a party of six people. Each pair of people are either friends (they know each other) or strangers (they do not).
- ***Theorem:*** Any such party must have a group of three mutual friends (three people who all know one another) or three mutual strangers (three people, none of whom know any of the others).

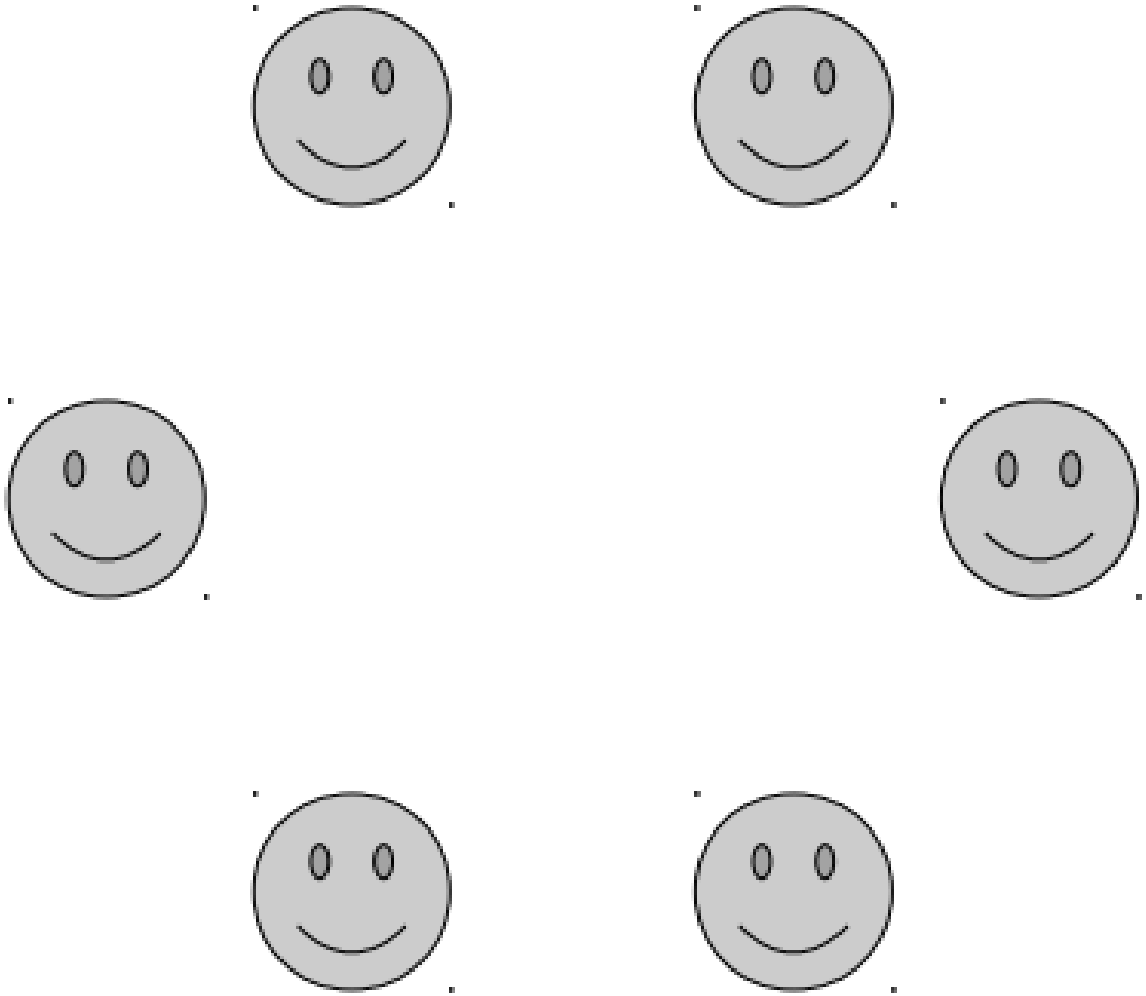


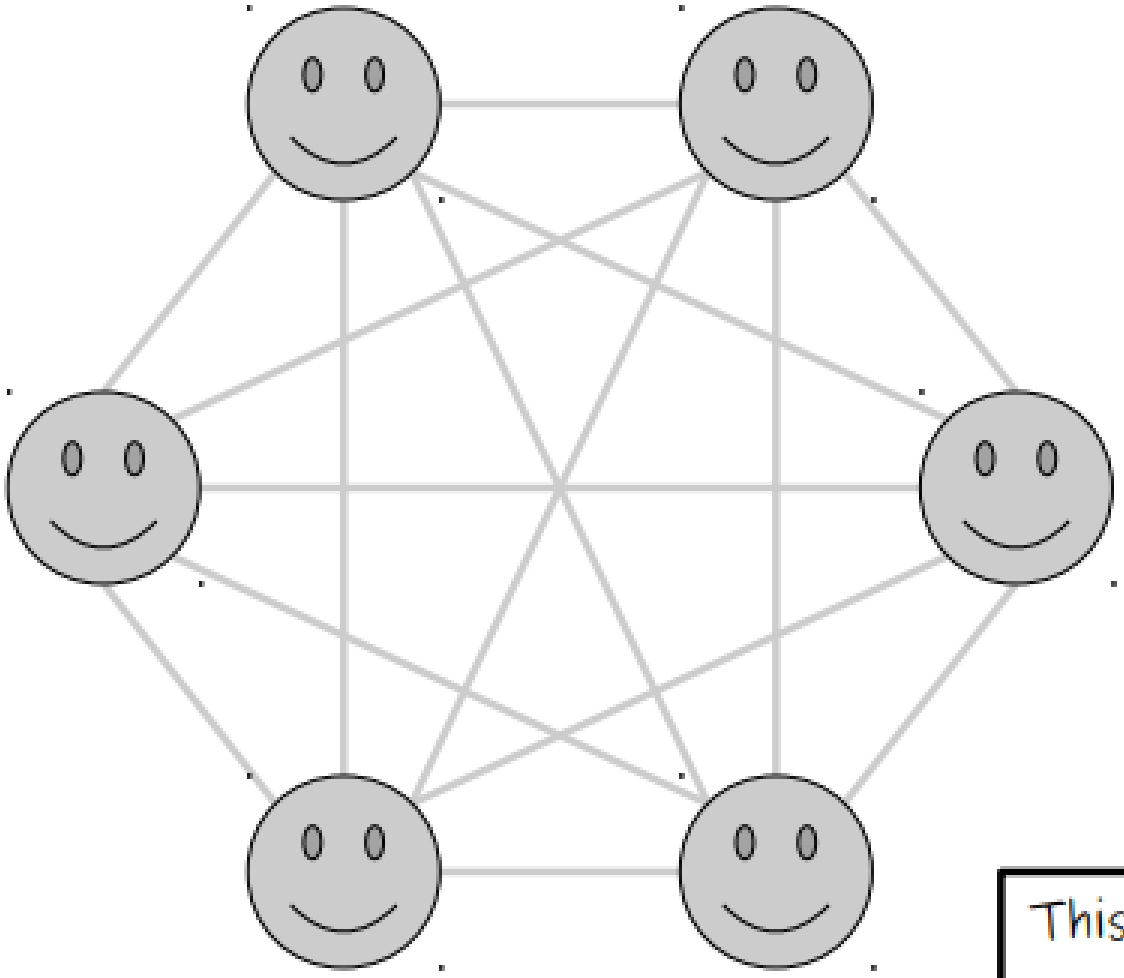




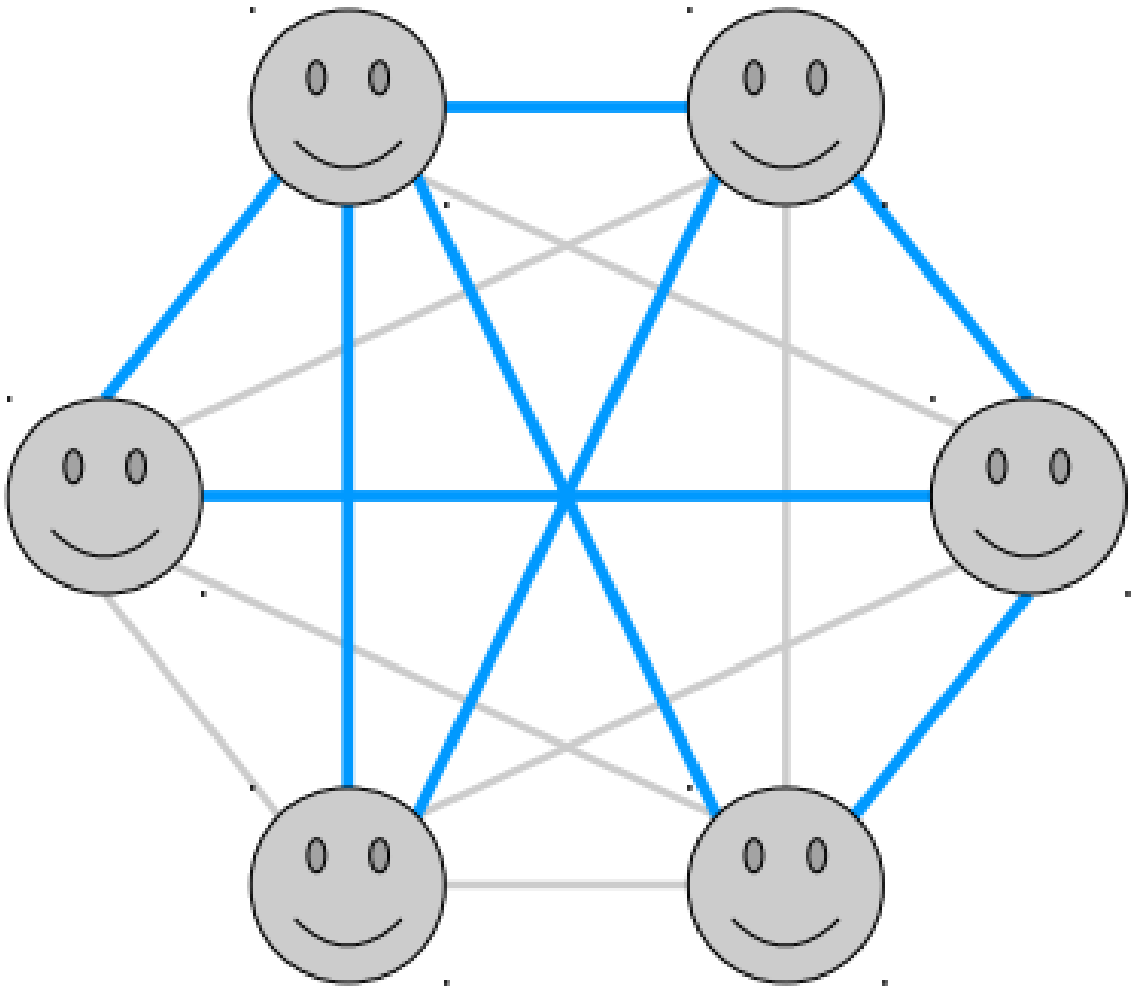


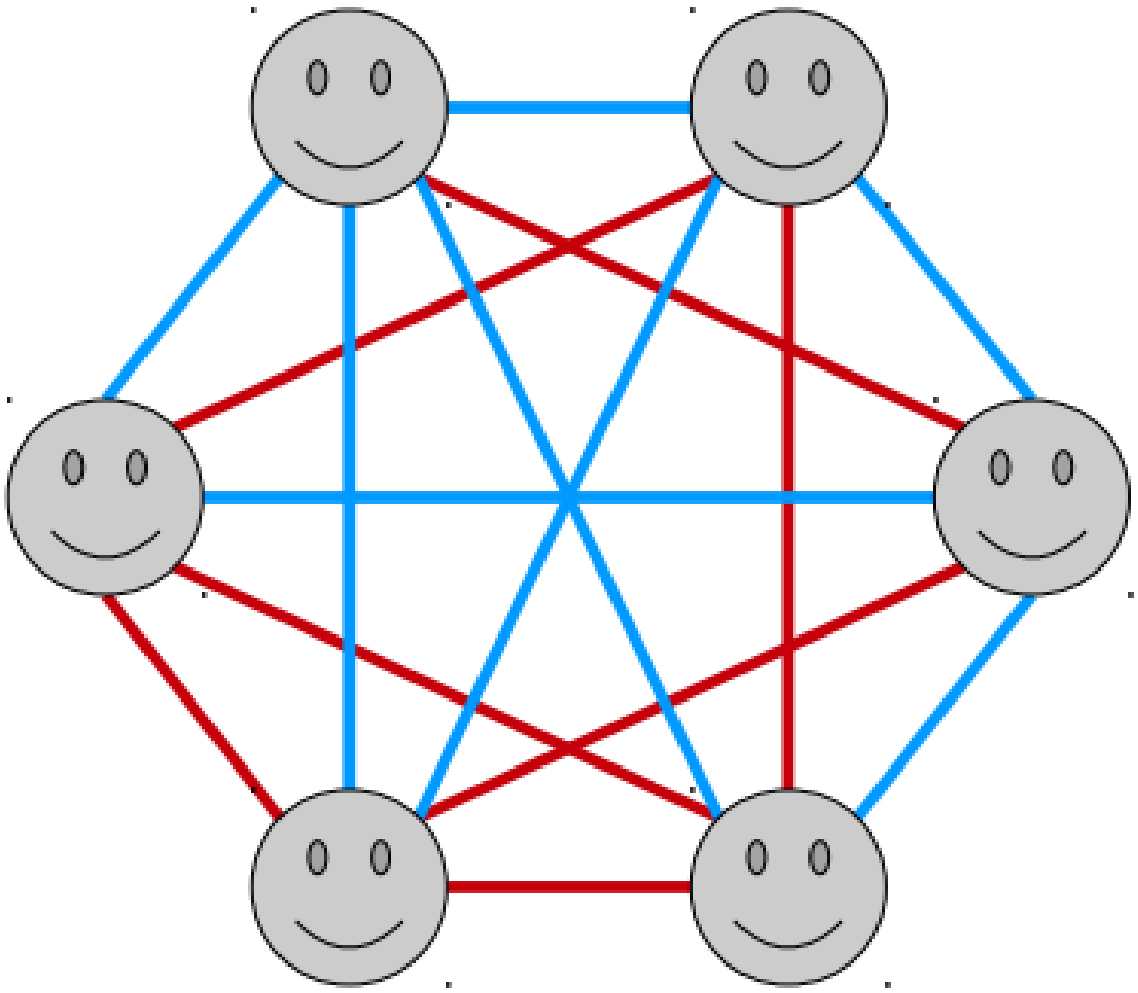


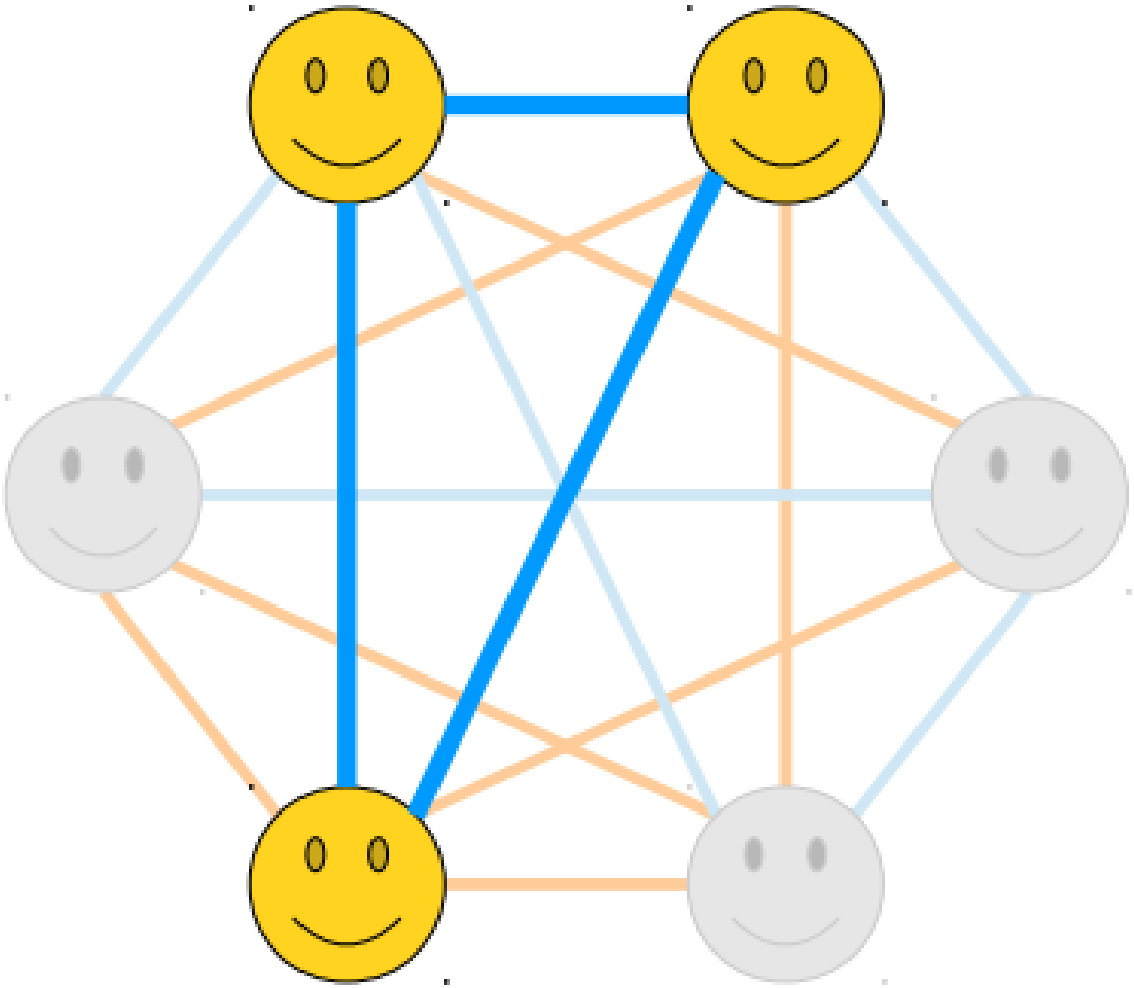


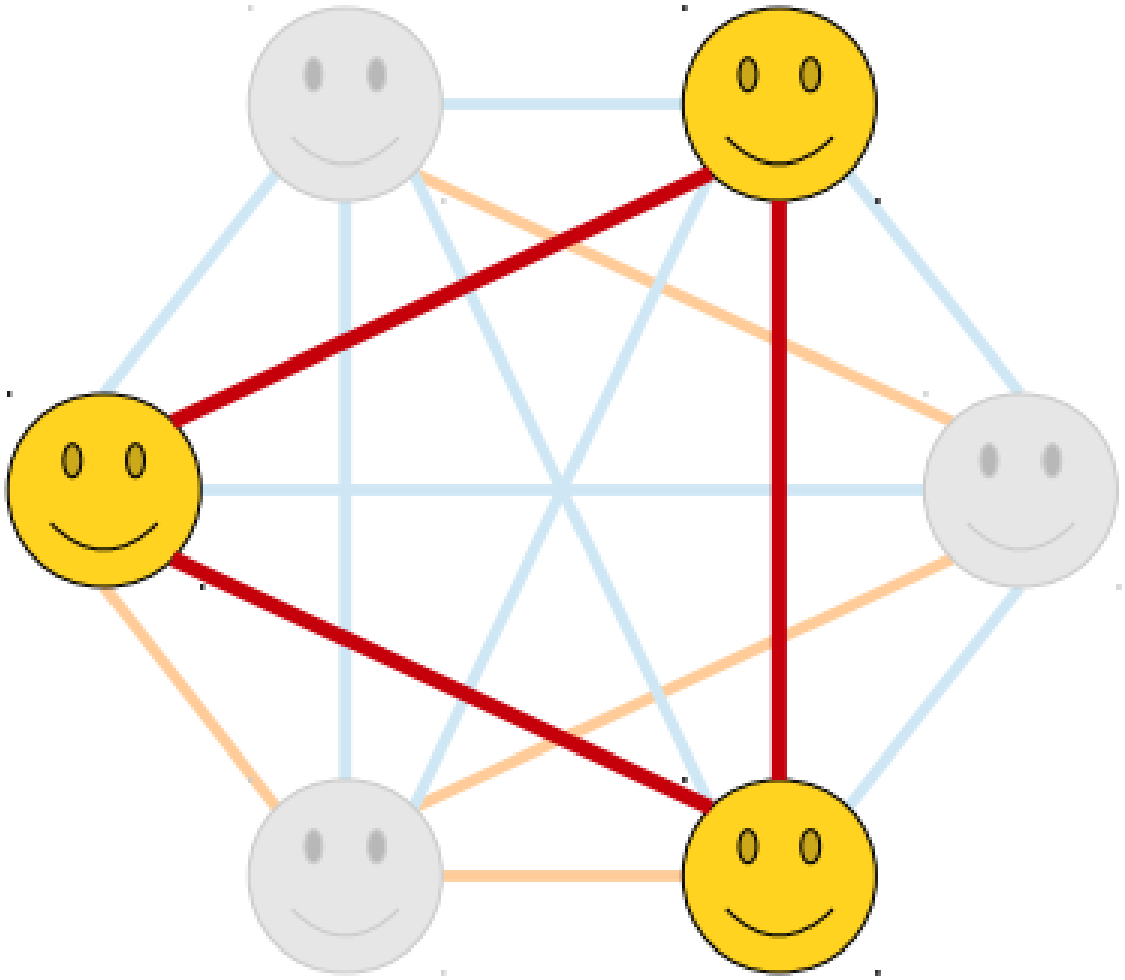


This graph is called a *6-clique*, by the way.



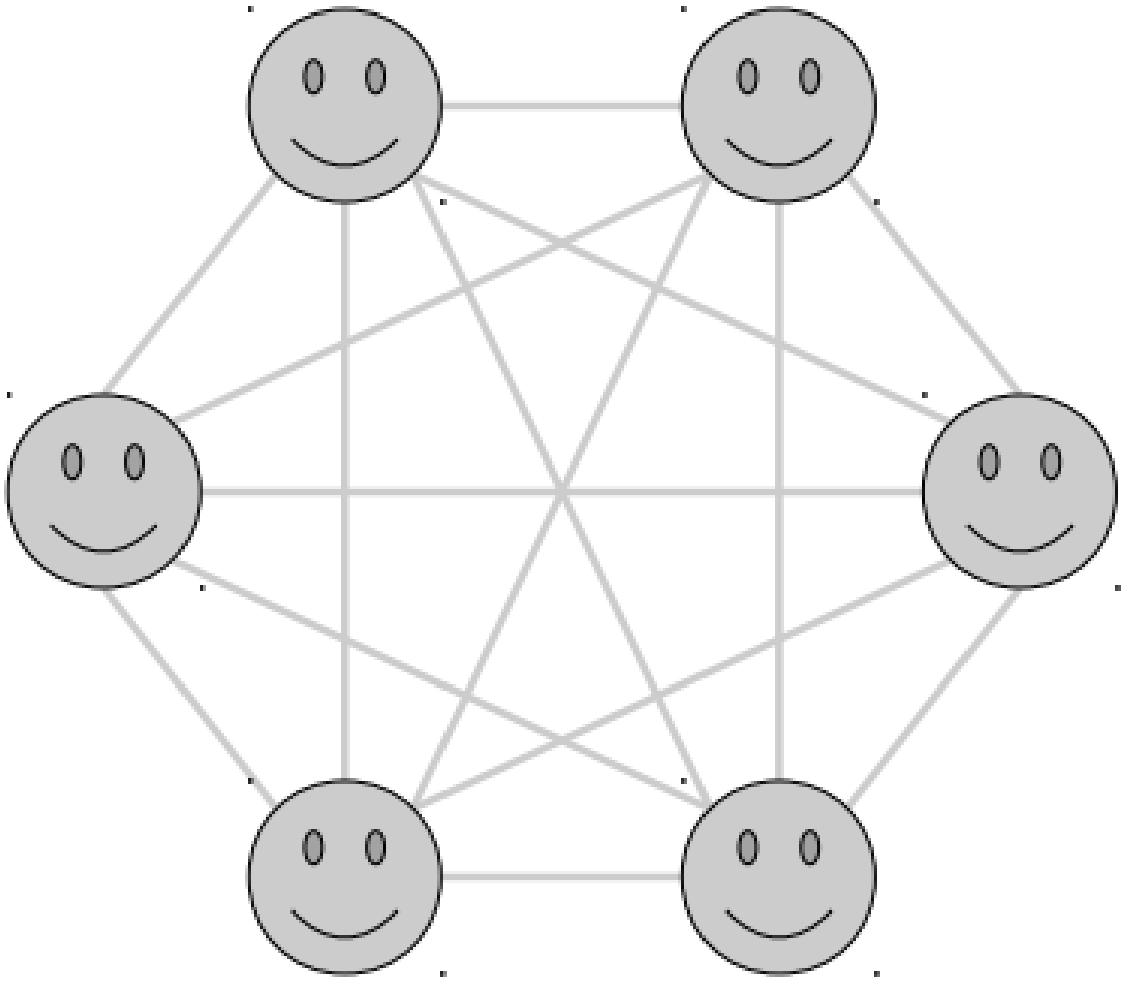


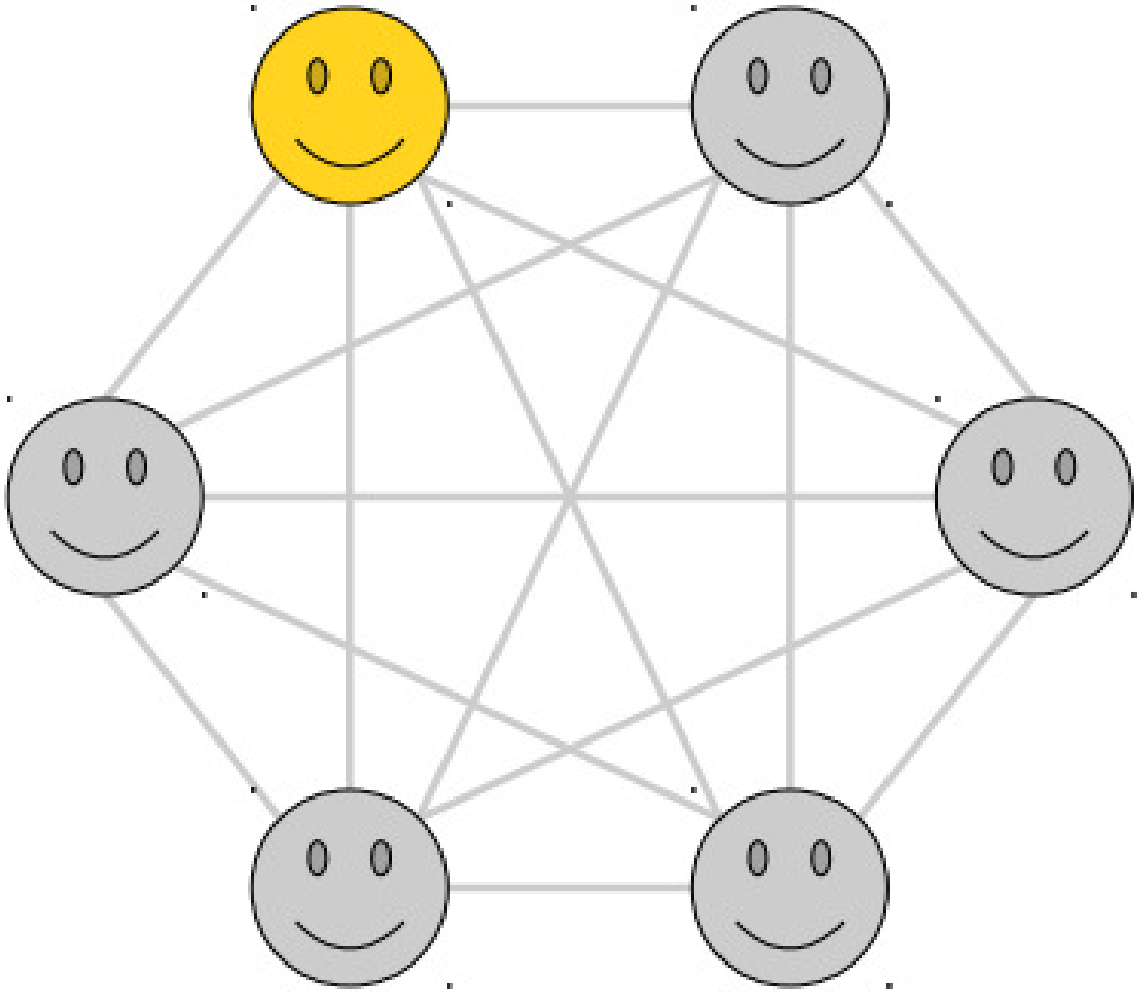


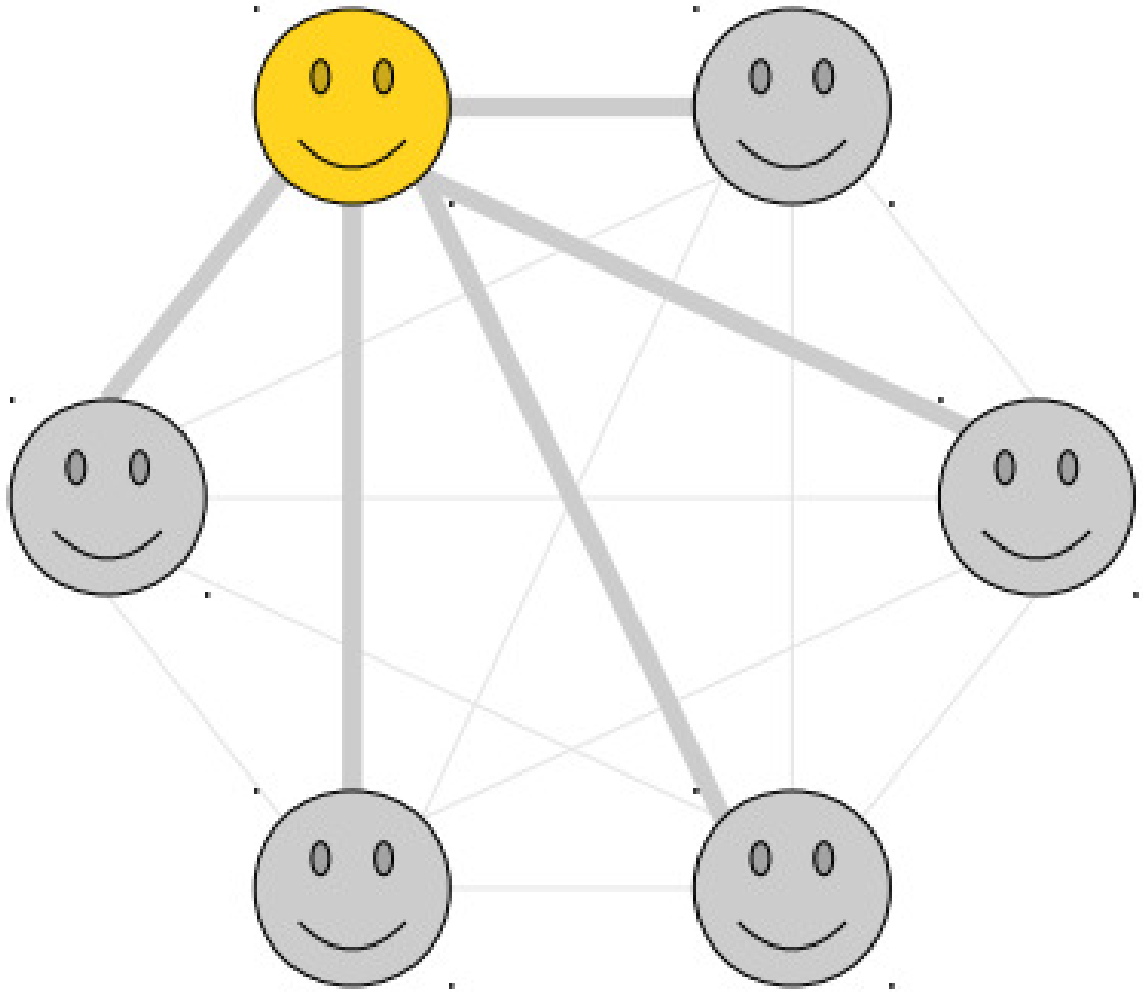


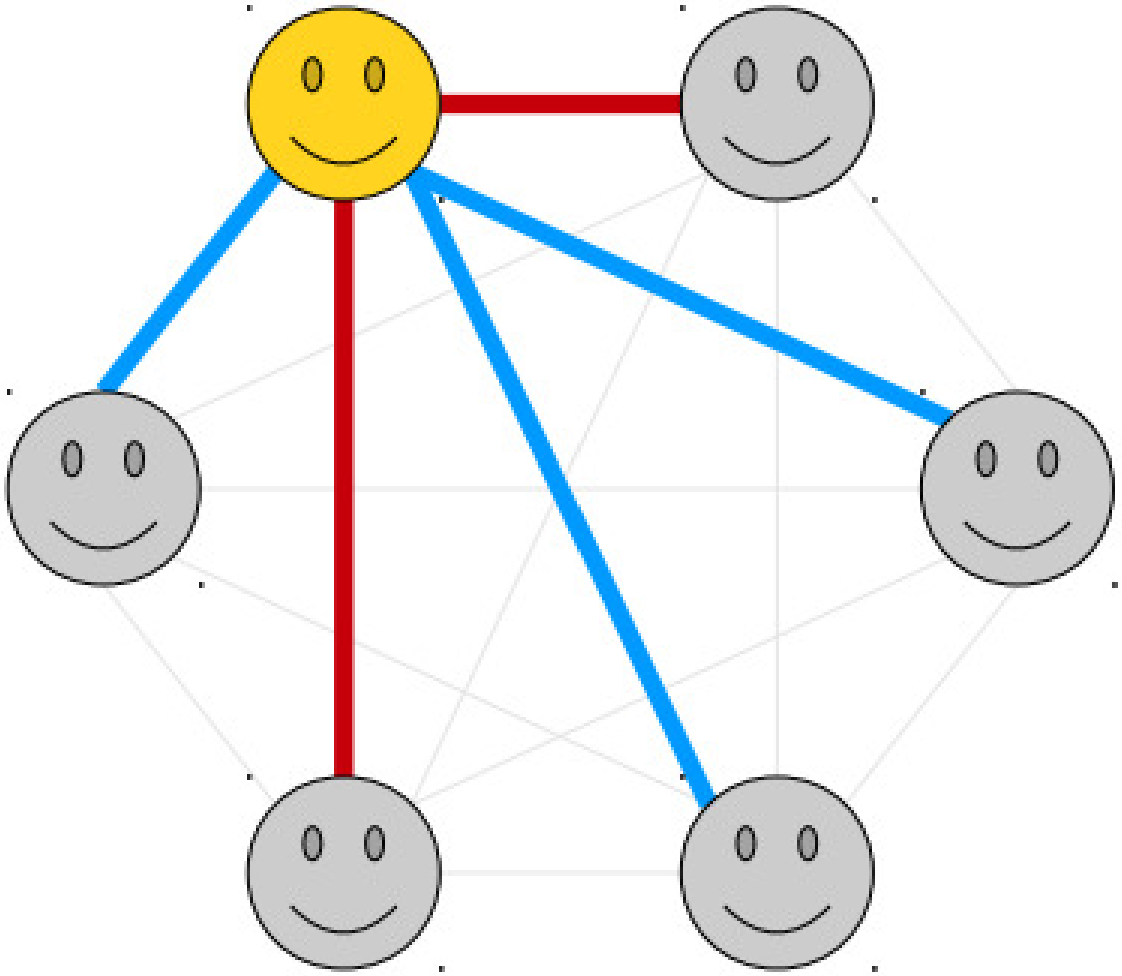
Friends and Strangers Restated

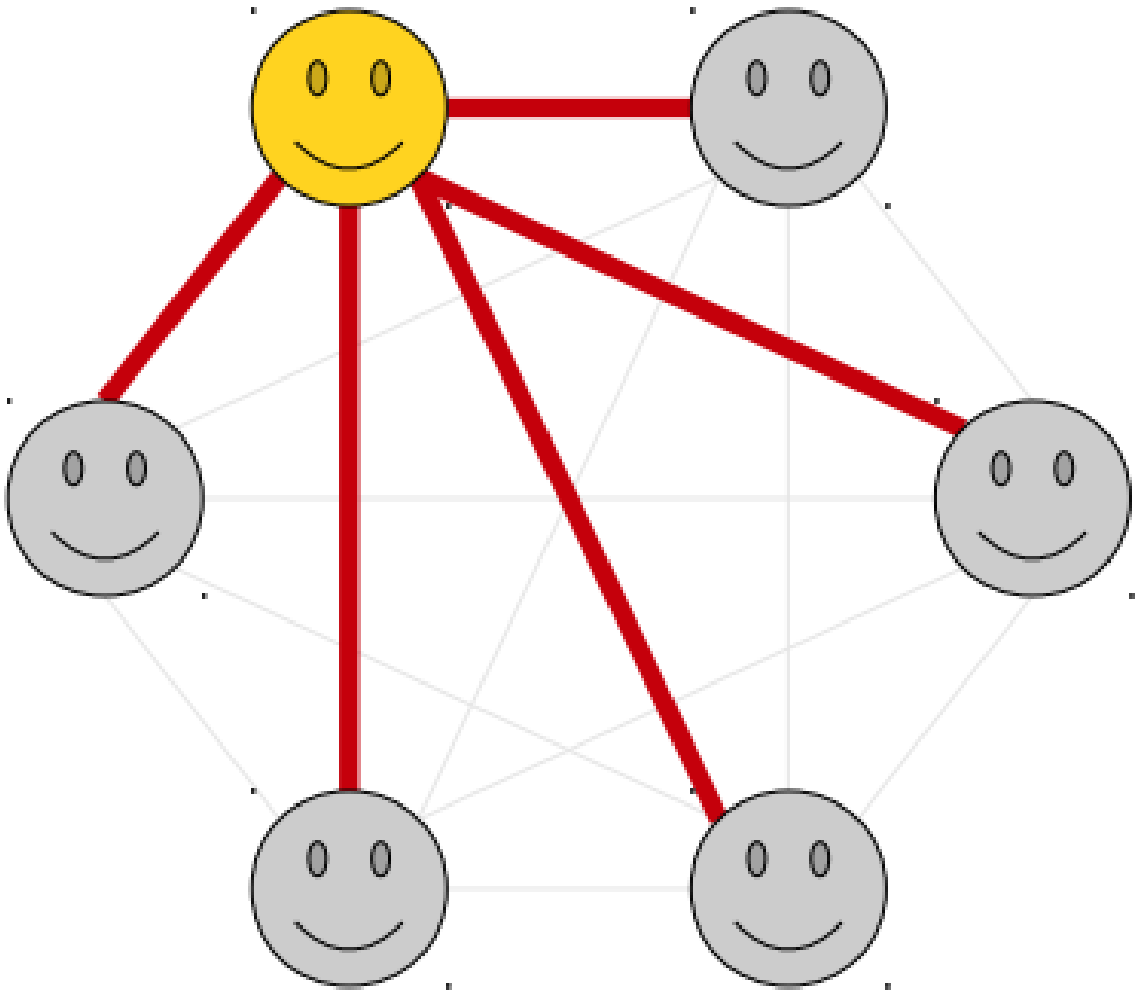
- From a graph-theoretic perspective, the Theorem on Friends and Strangers can be restated as follows:
- ***Theorem:*** Consider a 6-clique where every edge is colored red or blue. The the graph contains a red triangle, a blue triangle, or both.
- How can we prove this?

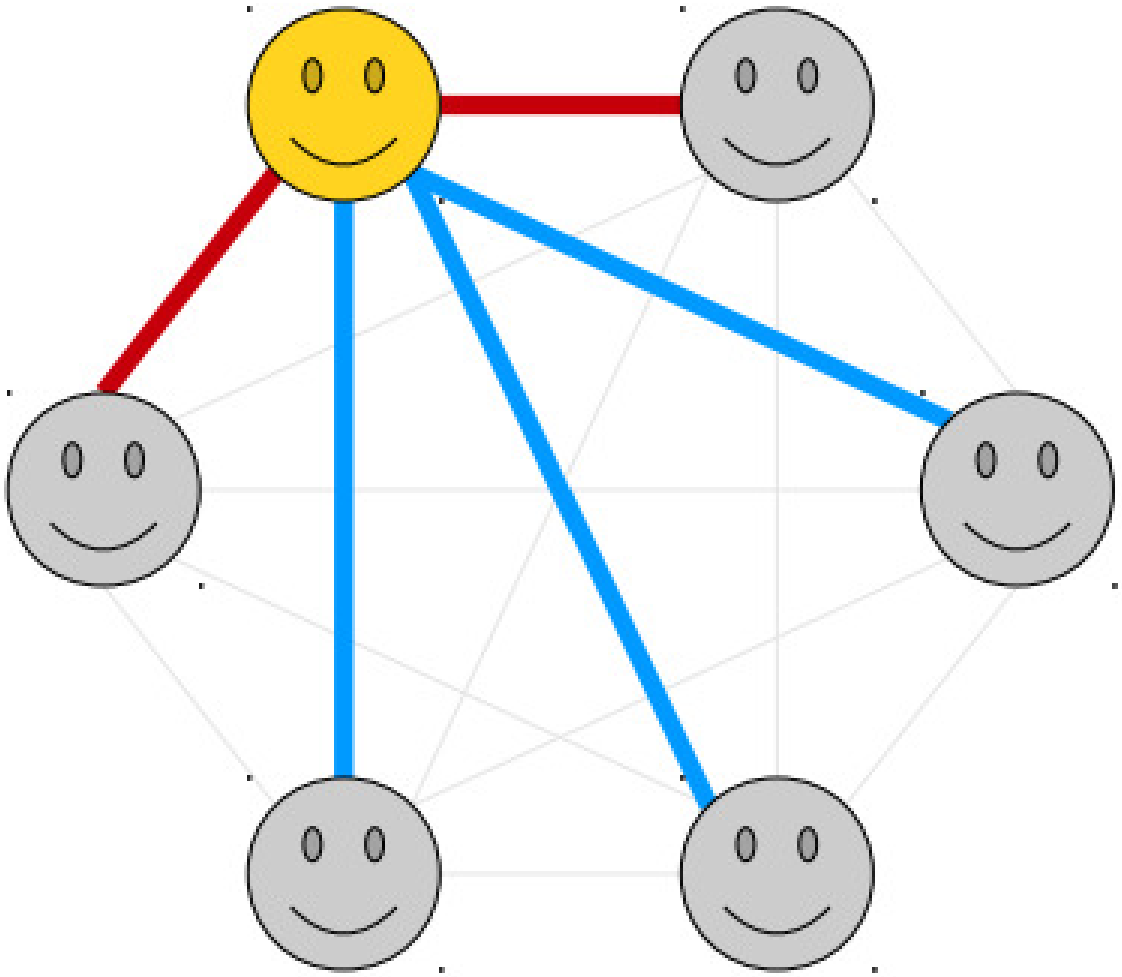


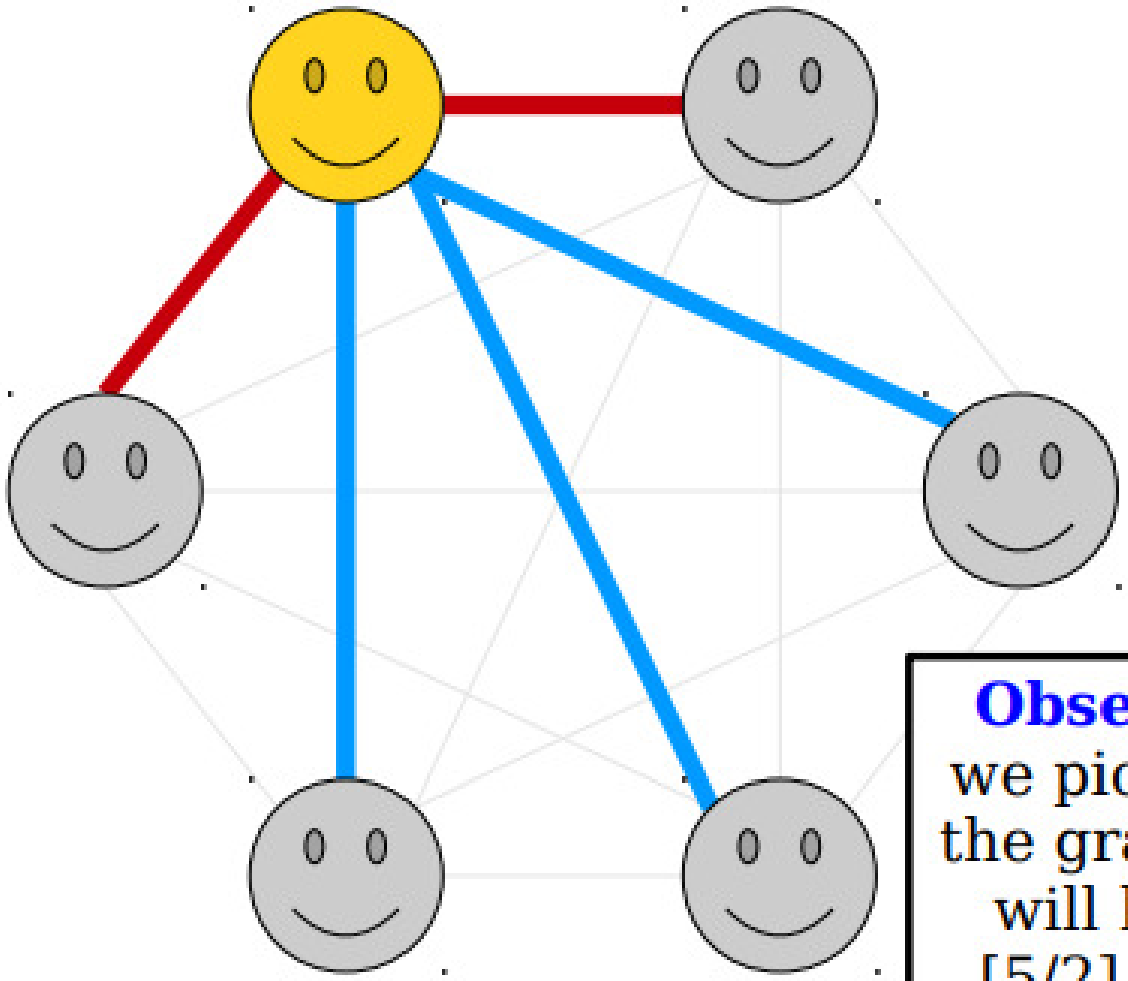




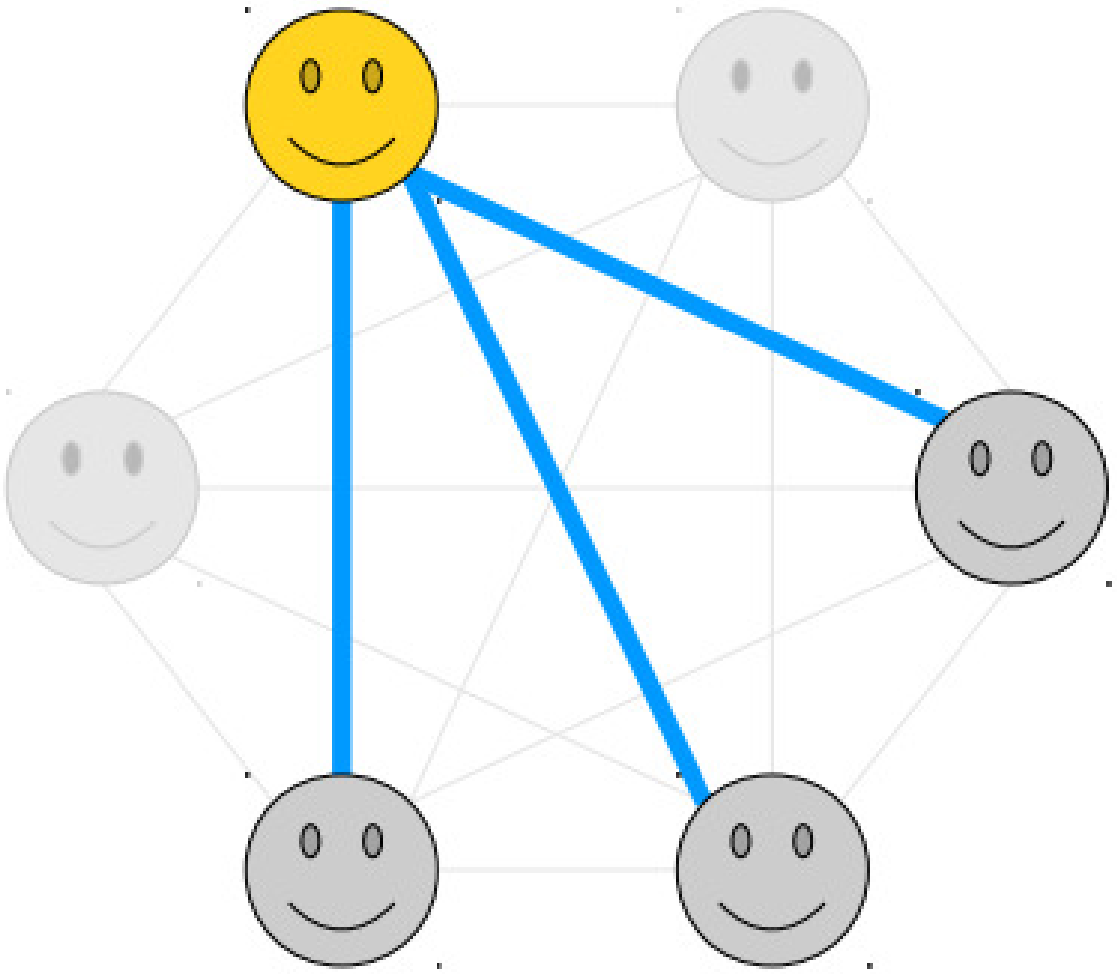


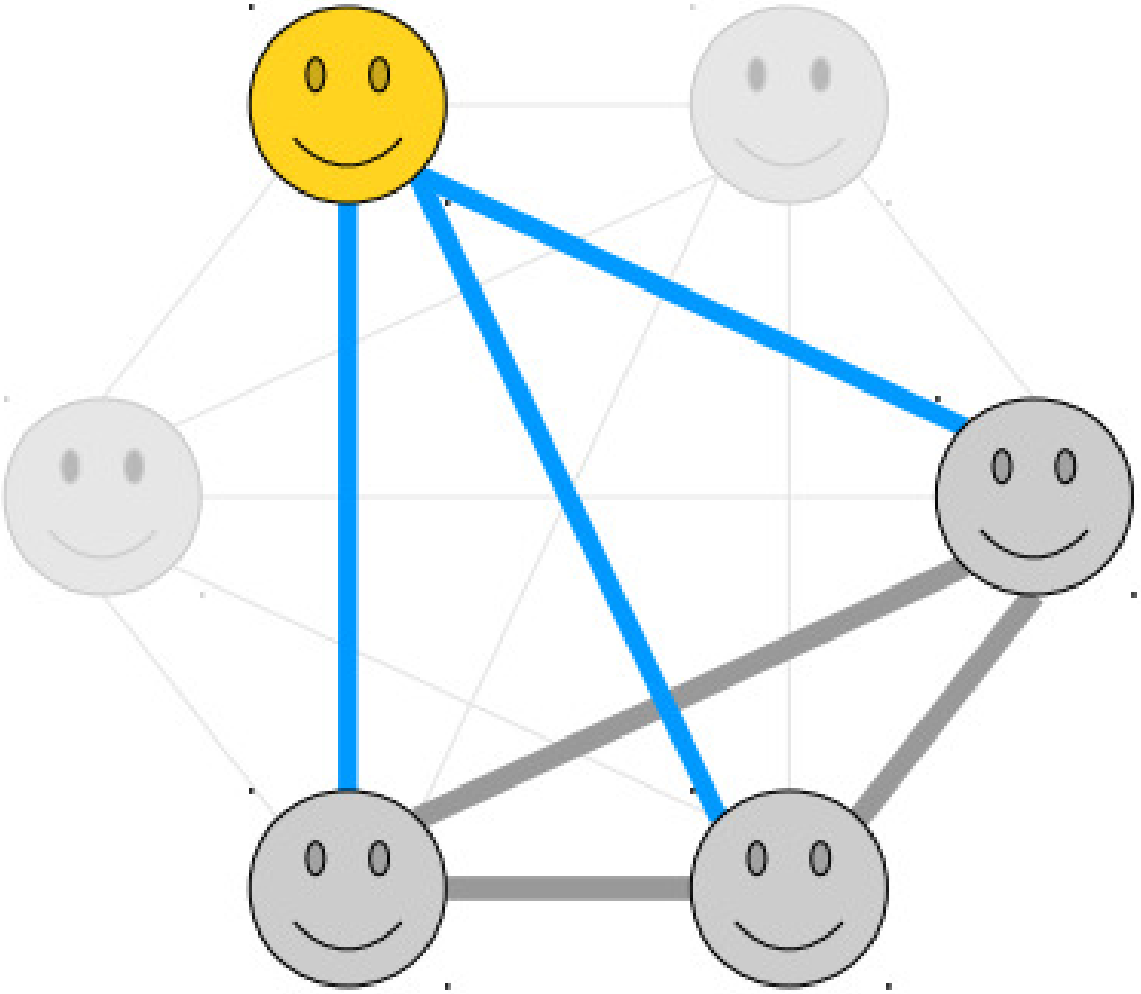


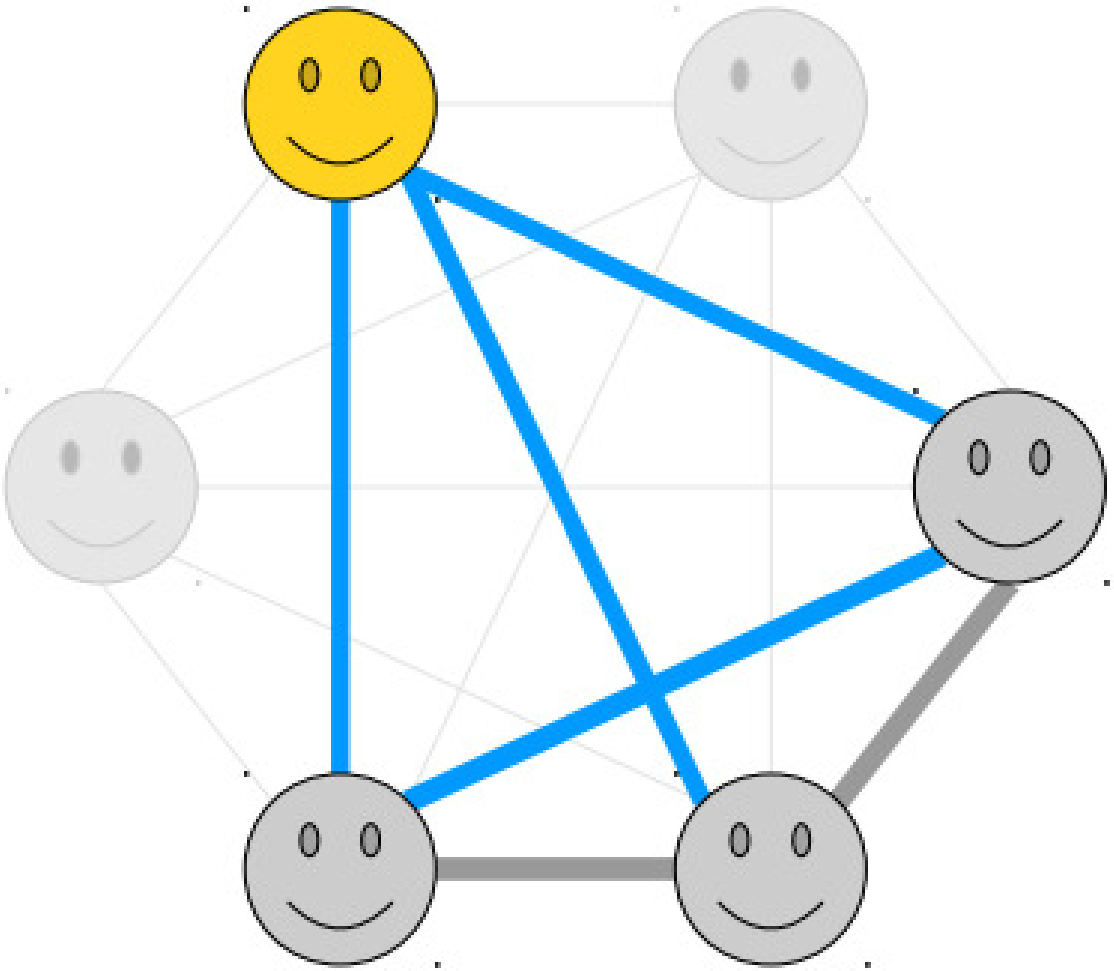


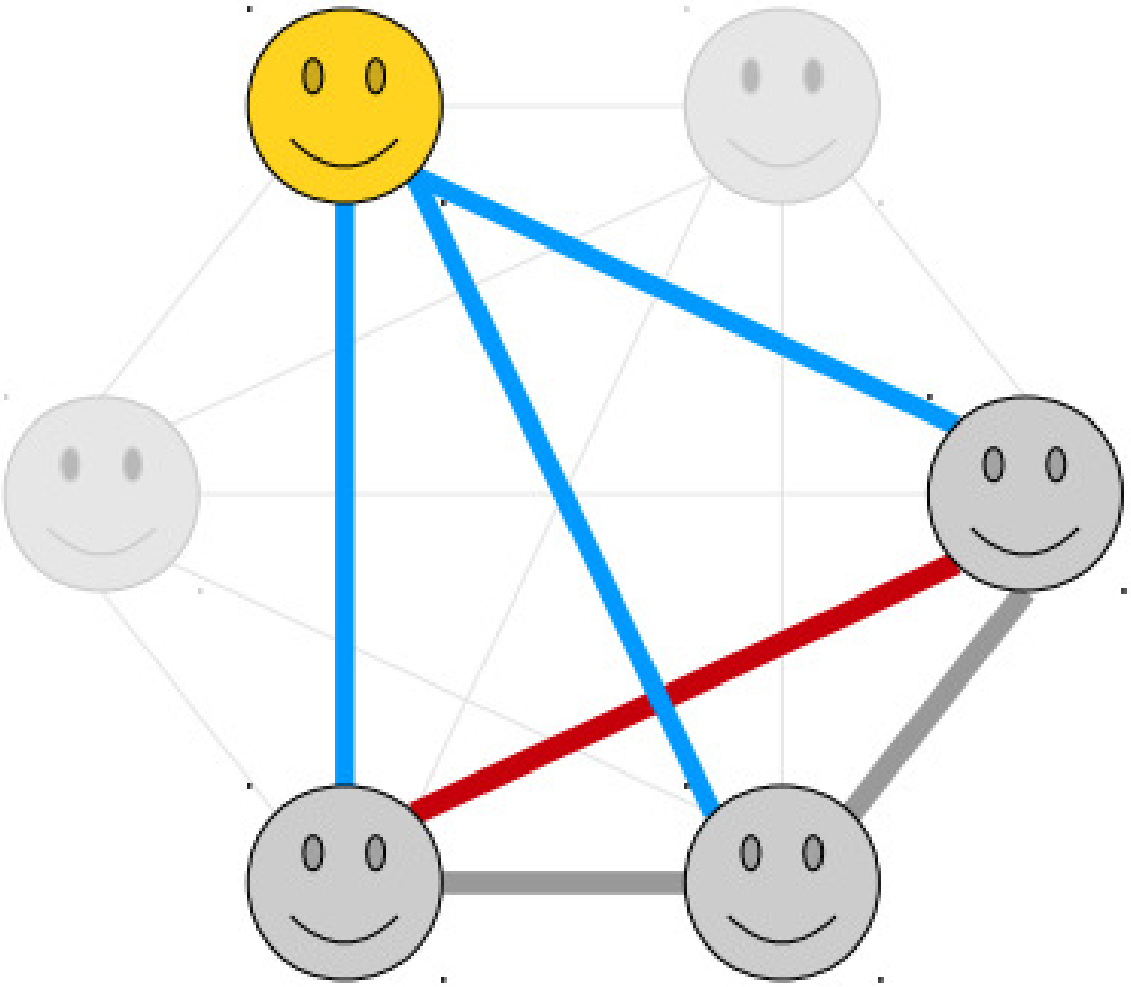


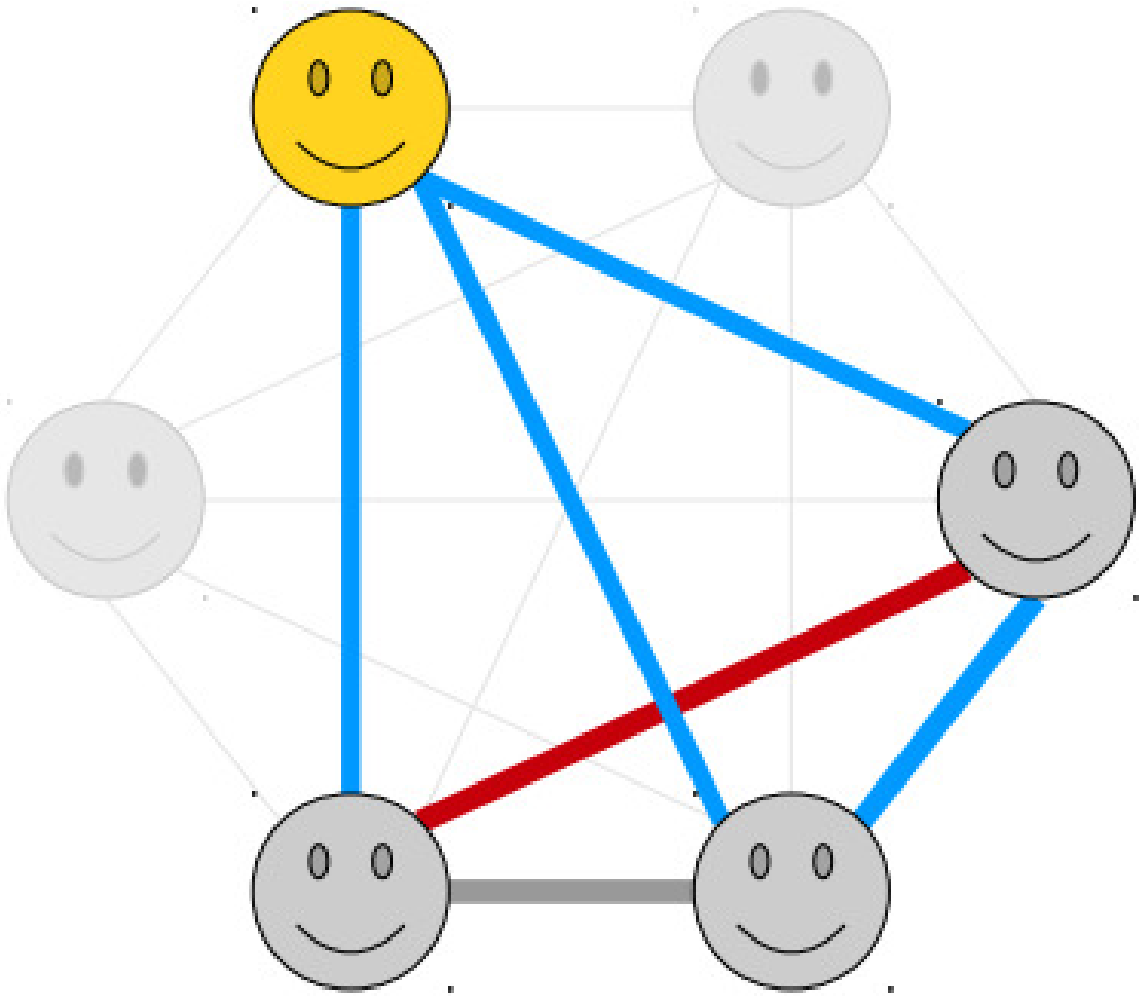
Observation 1: If we pick any node in the graph, that node will have at least $\lceil 5/2 \rceil = 3$ edges of the same color incident to it.

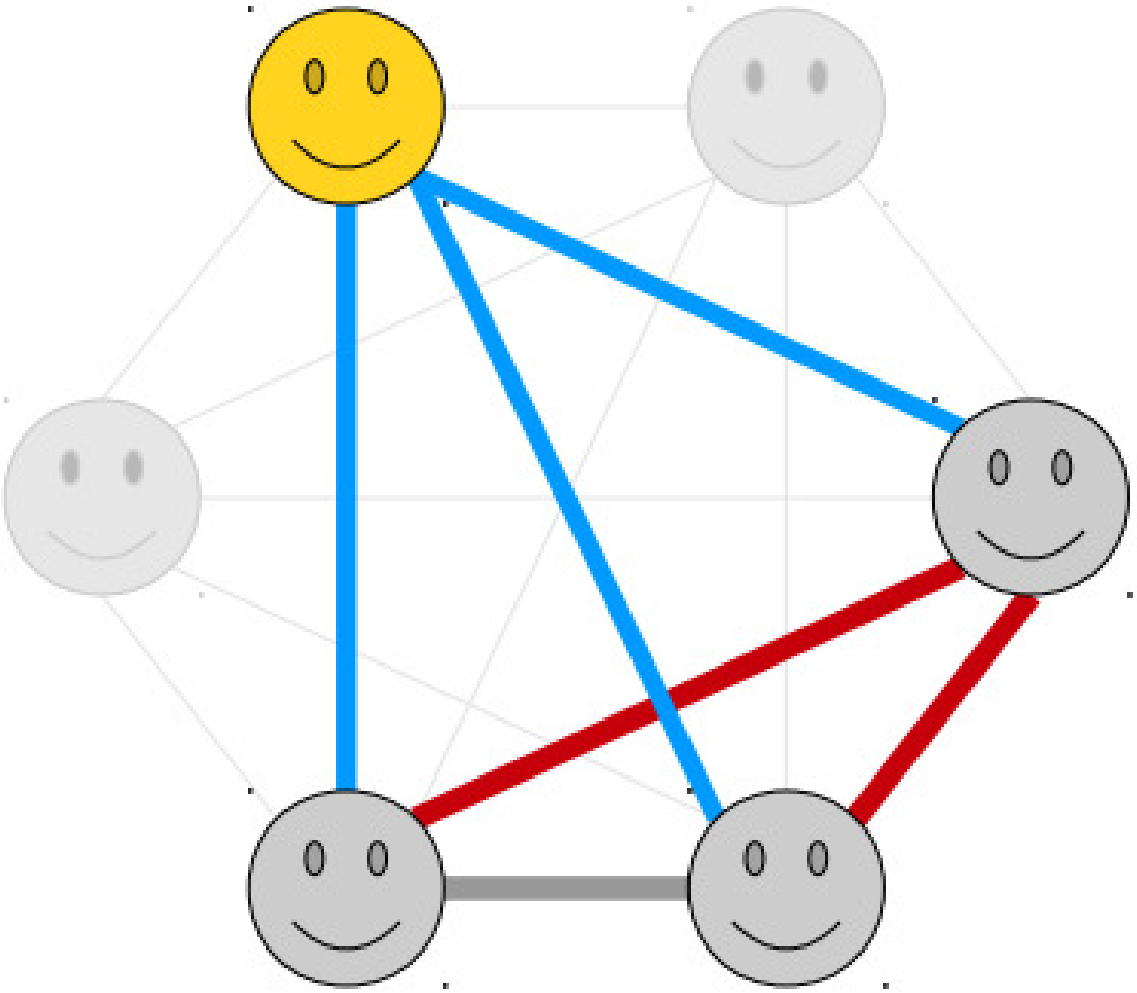


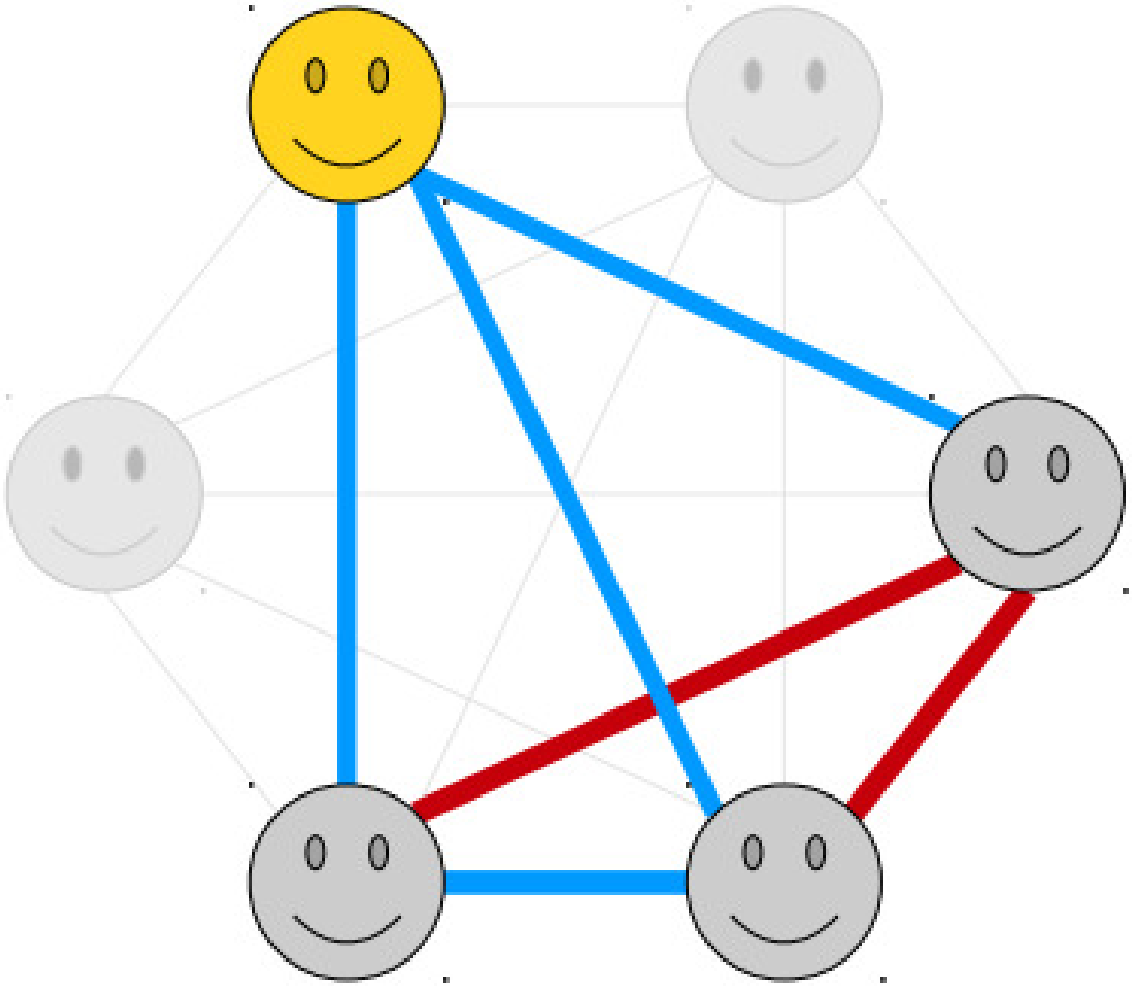


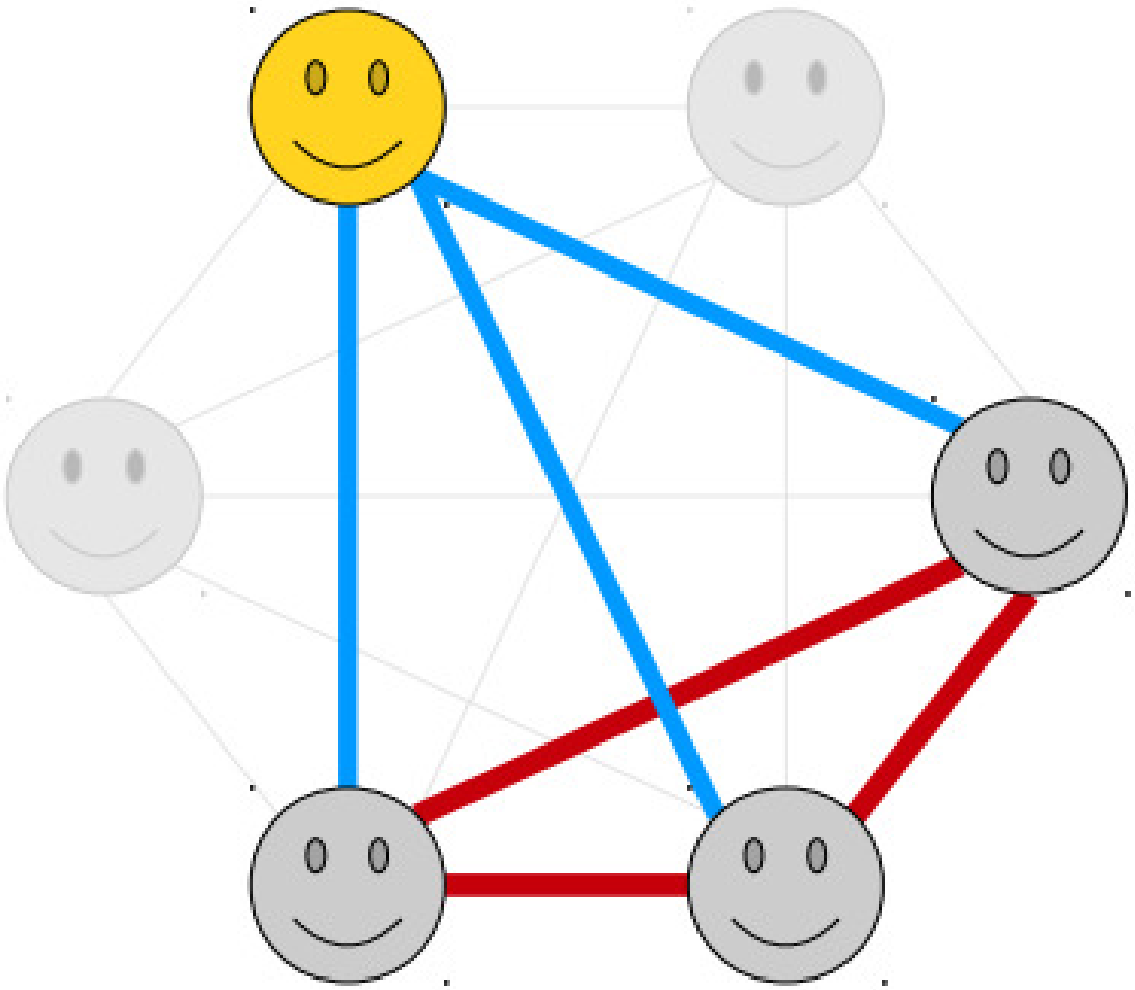












Theorem: Consider a 6-clique in which every edge is colored either red or blue. Then there must be a triangle of red edges, a triangle of blue edges, or both.

Proof: Color the edges of the 6-clique either red or blue arbitrarily. Let x be any node in the 6-clique. It is incident to five edges and there are two possible colors for those edges. Therefore, by the generalized pigeonhole principle, at least $\lceil 5/2 \rceil = 3$ of those edges must be the same color. Call that color c_1 and let the other color be c_2.

Let r, s, and t be three of the nodes adjacent to node x along an edge of color c_1. If any of the edges $\{r, s\}$, $\{r, t\}$, or $\{s, t\}$ are of color c_1, then one of those edges plus the two edges connecting back to node x form a triangle of color c_1. Otherwise, all three of those edges are of color c_2, and they form a triangle of color c_2. Overall, this gives a red triangle or a blue triangle, as required. ■

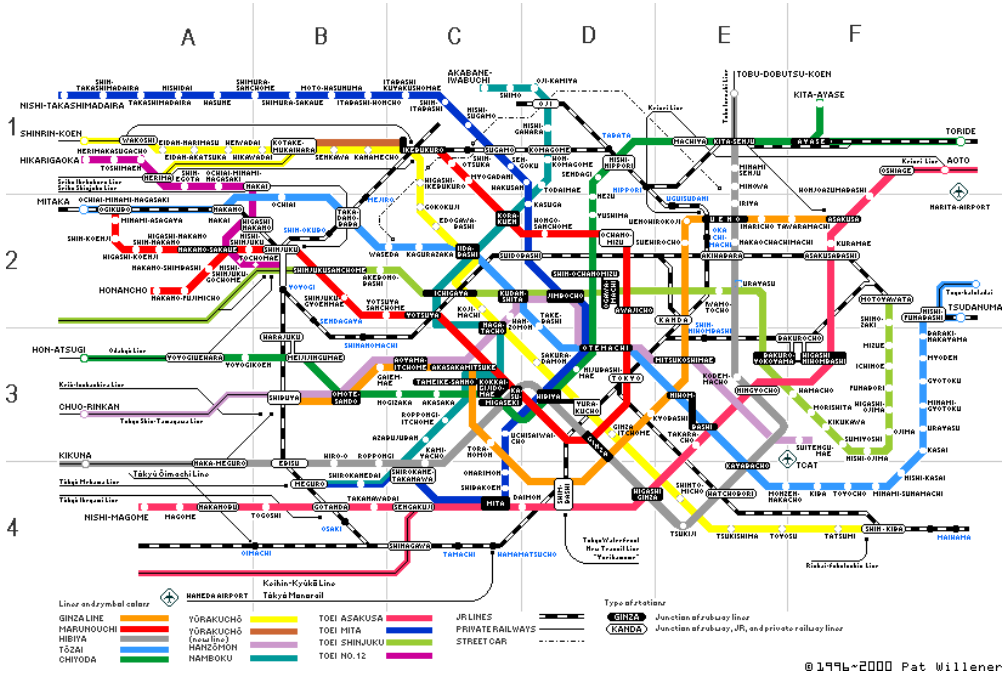
Ramsey Theory

- The proof we did is a special case of a broader result.
- ***Theorem (Ramsey's Theorem):*** For any natural number n, there is a smallest natural number $R(n)$ such that if the edges of an $R(n)$-clique are colored red or blue, the resulting graph will contain either a red n-clique or a blue n-clique.
  - Our proof was that $R(3) \leq 6$.
- A more philosophical take on this theorem: true disorder is impossible at a large scale, since no matter how you organize things, you're guaranteed to find some interesting substructure.

Some Graph Problems

Shortest Path Problem

Given a weighted graph and two vertices u and v, we want to find a path of minimum total weight between u and v.



http://www.hyperdia.com/en/ Tokyo Subway Map

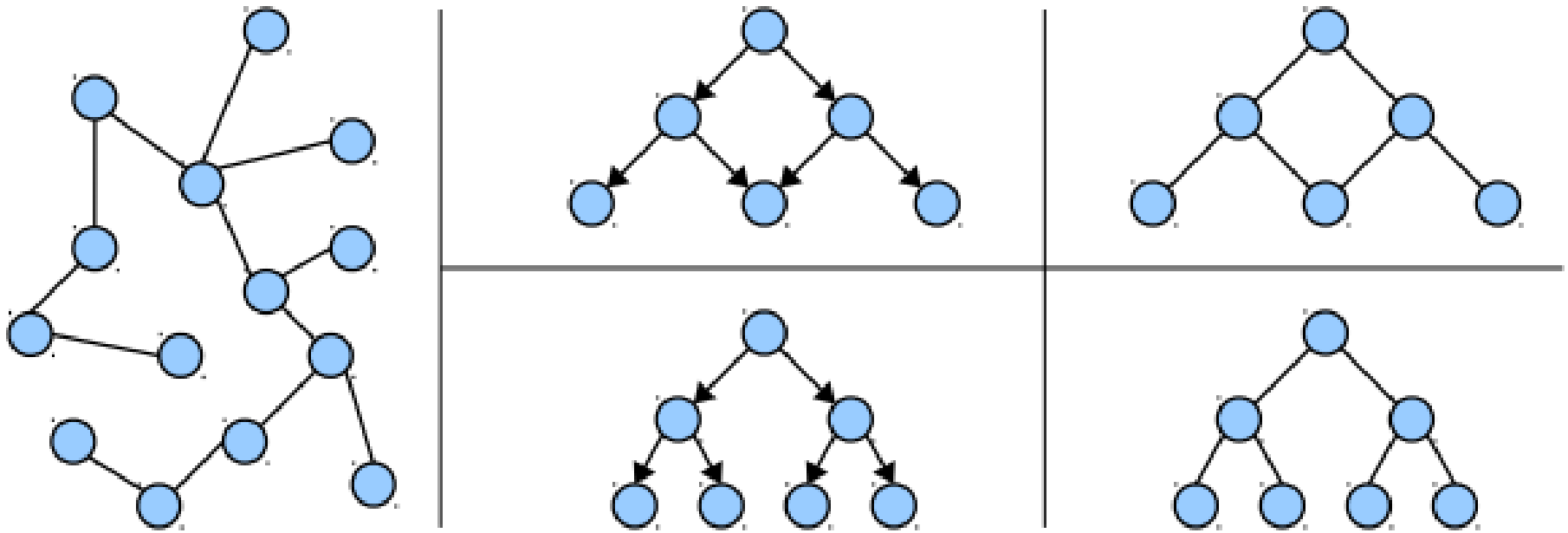
TSP: Traveling Salesman Problem

Given a number of cities and the costs of traveling from any city to any other city, what is the cheapest round-trip route that visits each city exactly once and then returns to the starting city?

A Special Type of Graph: ***Trees***

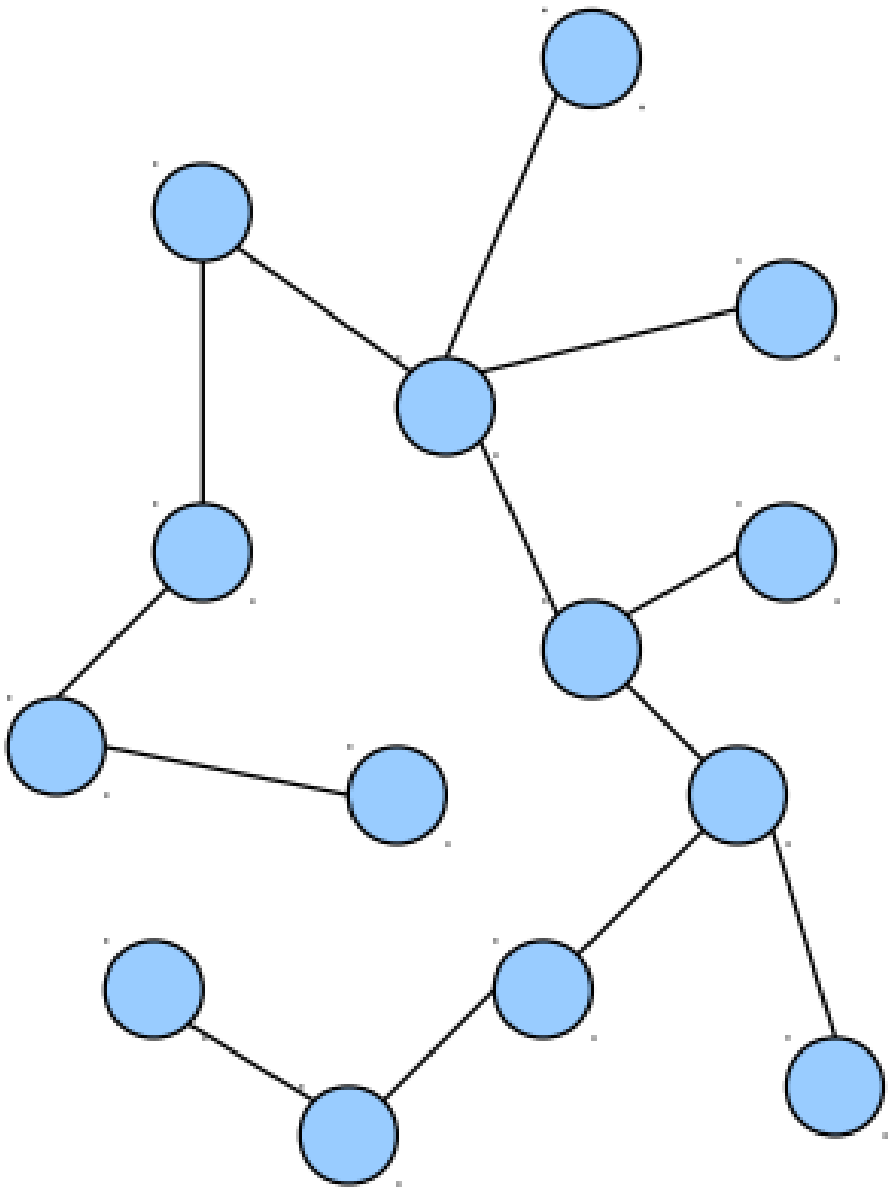
- A ***tree*** is a connected, nonempty graph with no simple cycles.

According to the above definition of trees, how many of these graphs are trees?



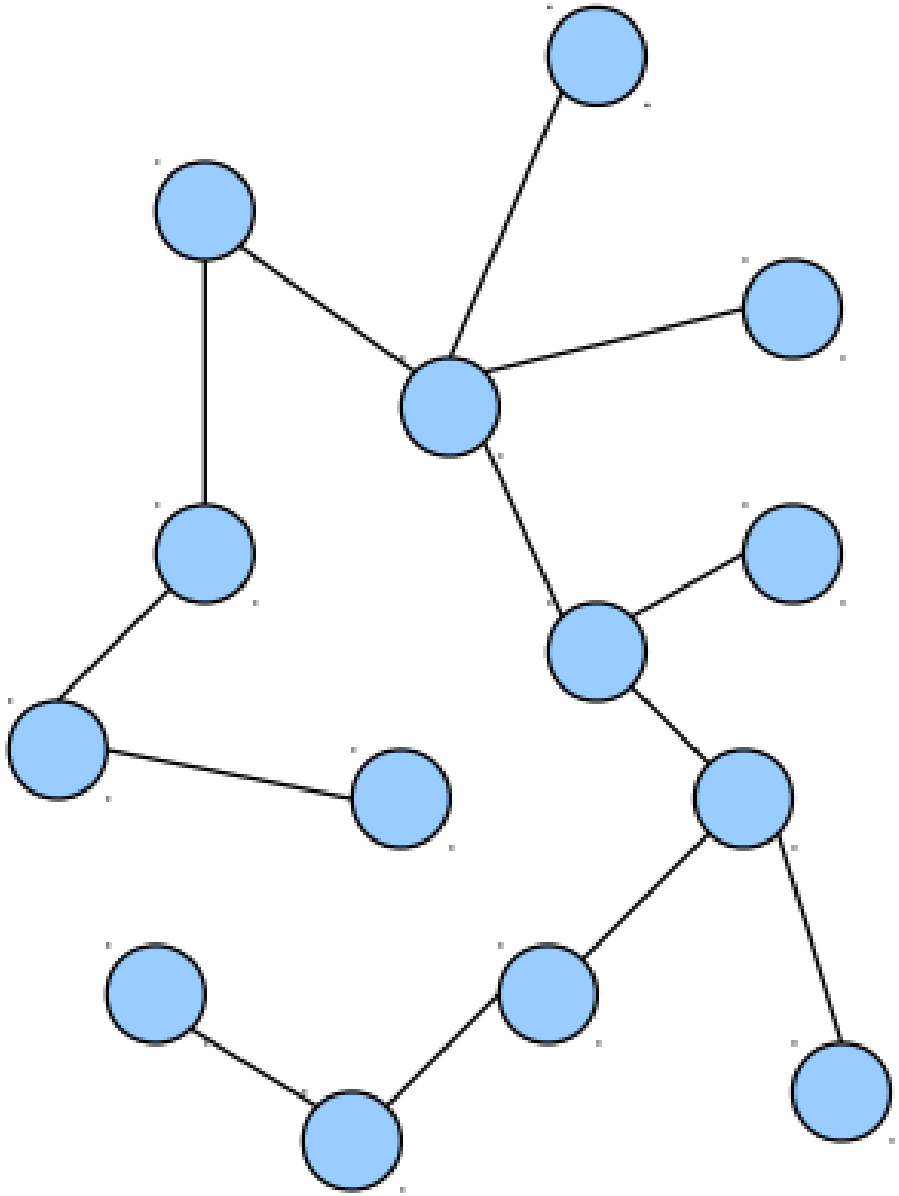
Trees

- A ***tree*** is a connected, nonempty graph with no simple cycles.



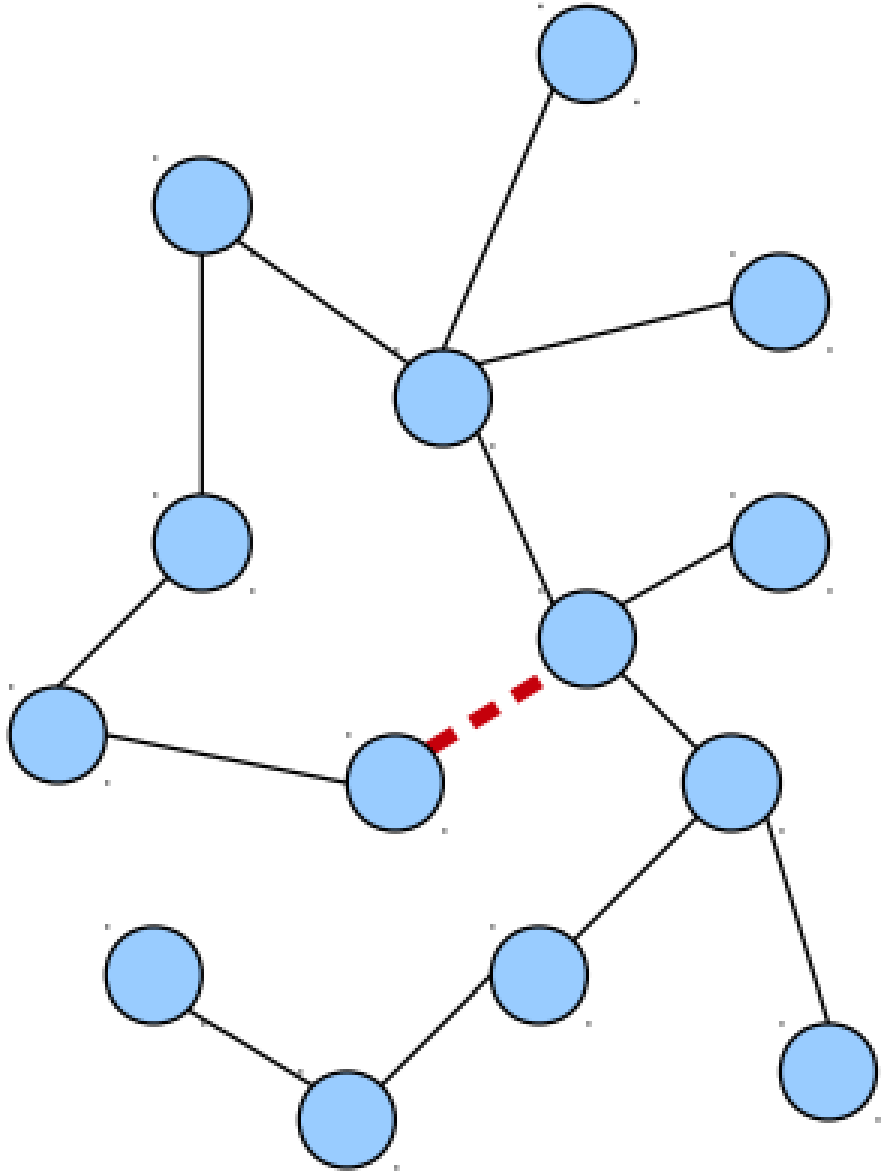
Trees

- A ***tree*** is a connected, nonempty graph with no simple cycles.
- Trees have tons of nice properties:
  - They're ***maximally acyclic*** (adding any missing edge creates a simple cycle)



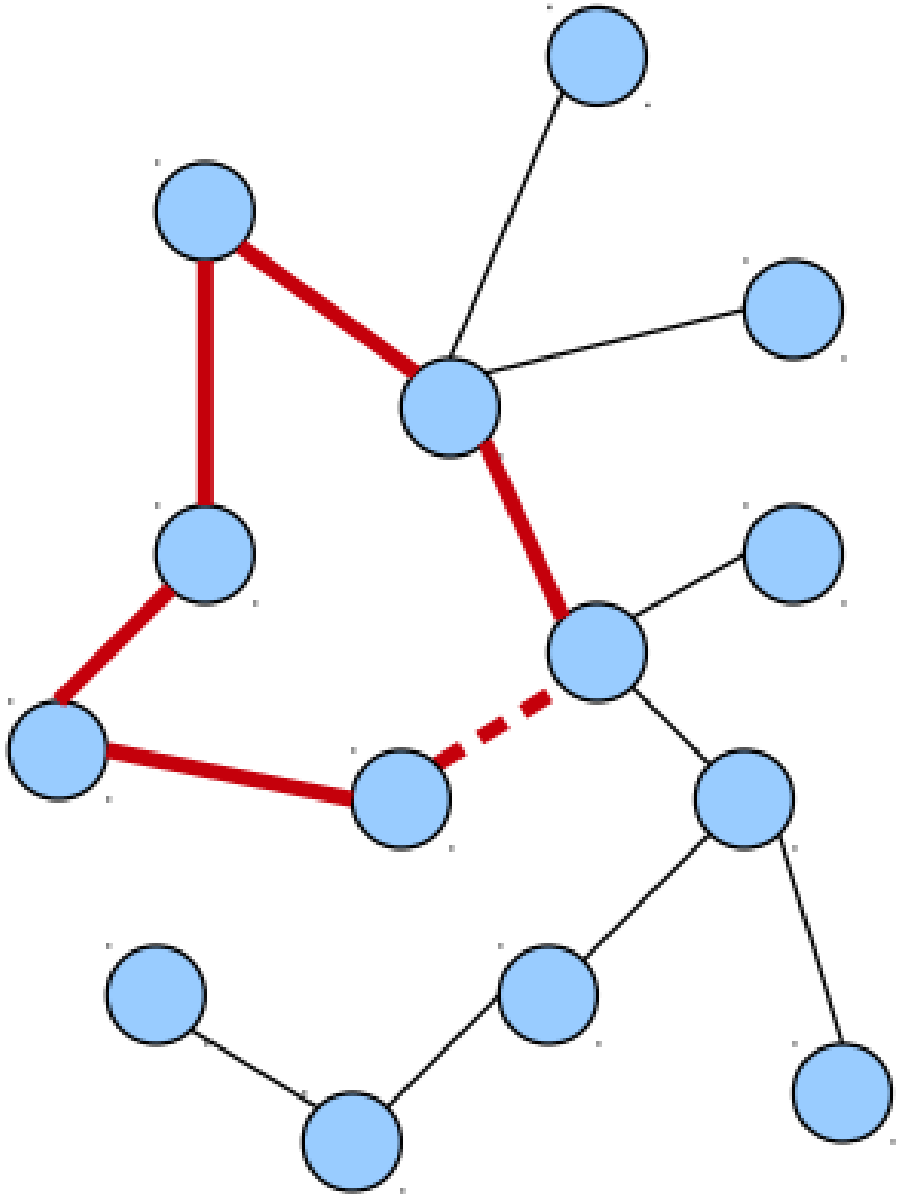
Trees

- A ***tree*** is a connected, nonempty graph with no simple cycles.
- Trees have tons of nice properties:
  - They're ***maximally acyclic*** (adding any missing edge creates a simple cycle)



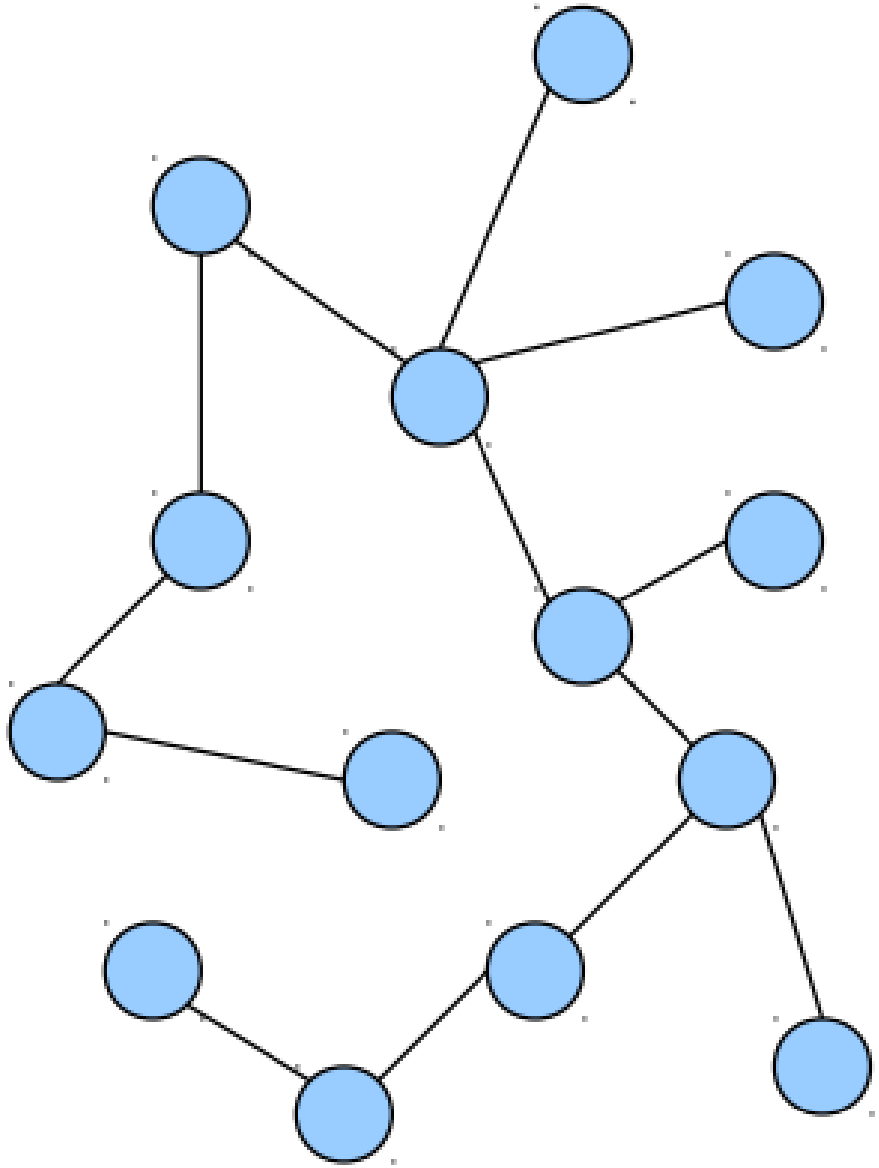
Trees

- A ***tree*** is a connected, nonempty graph with no simple cycles.
- Trees have tons of nice properties:
  - They're ***maximally acyclic*** (adding any missing edge creates a simple cycle)



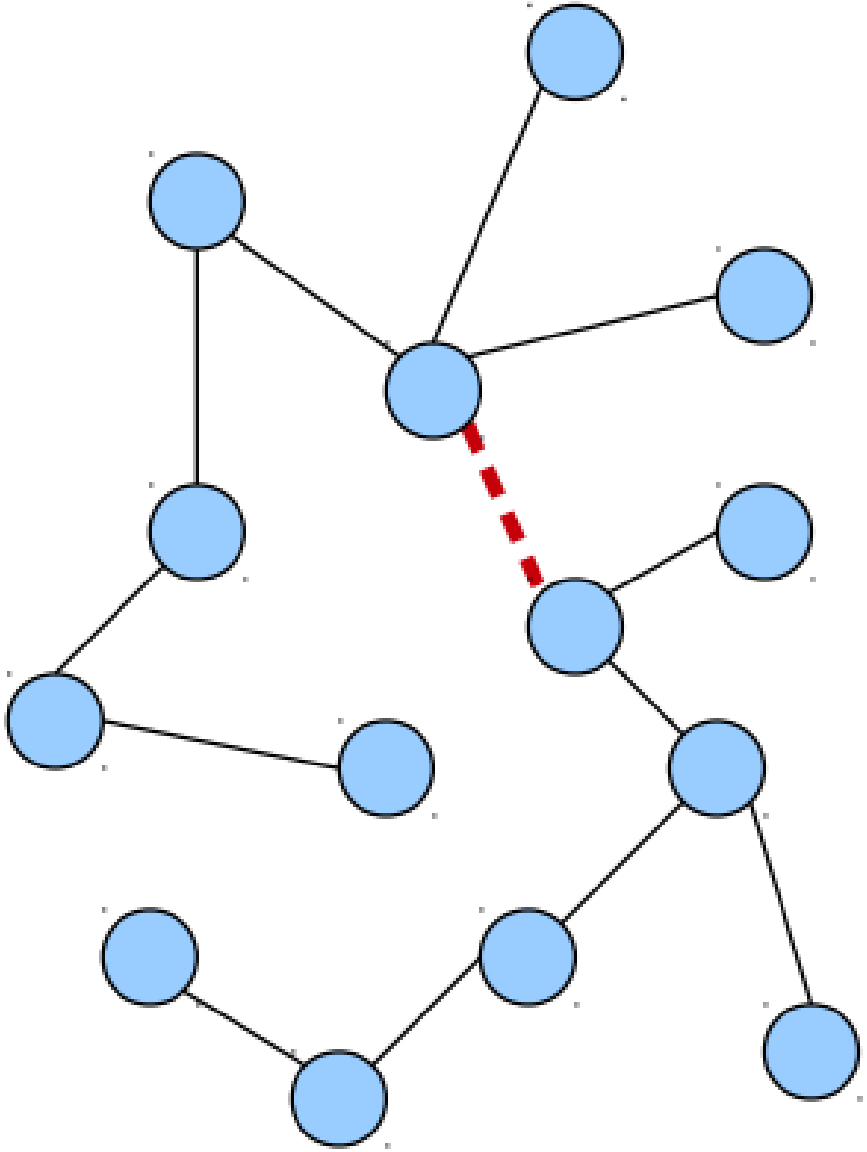
Trees

- A ***tree*** is a connected, nonempty graph with no simple cycles.
- Trees have tons of nice properties:
  - They're ***maximally acyclic*** (adding any missing edge creates a simple cycle)
  - They're ***minimally connected*** (deleting any edge disconnects the graph)



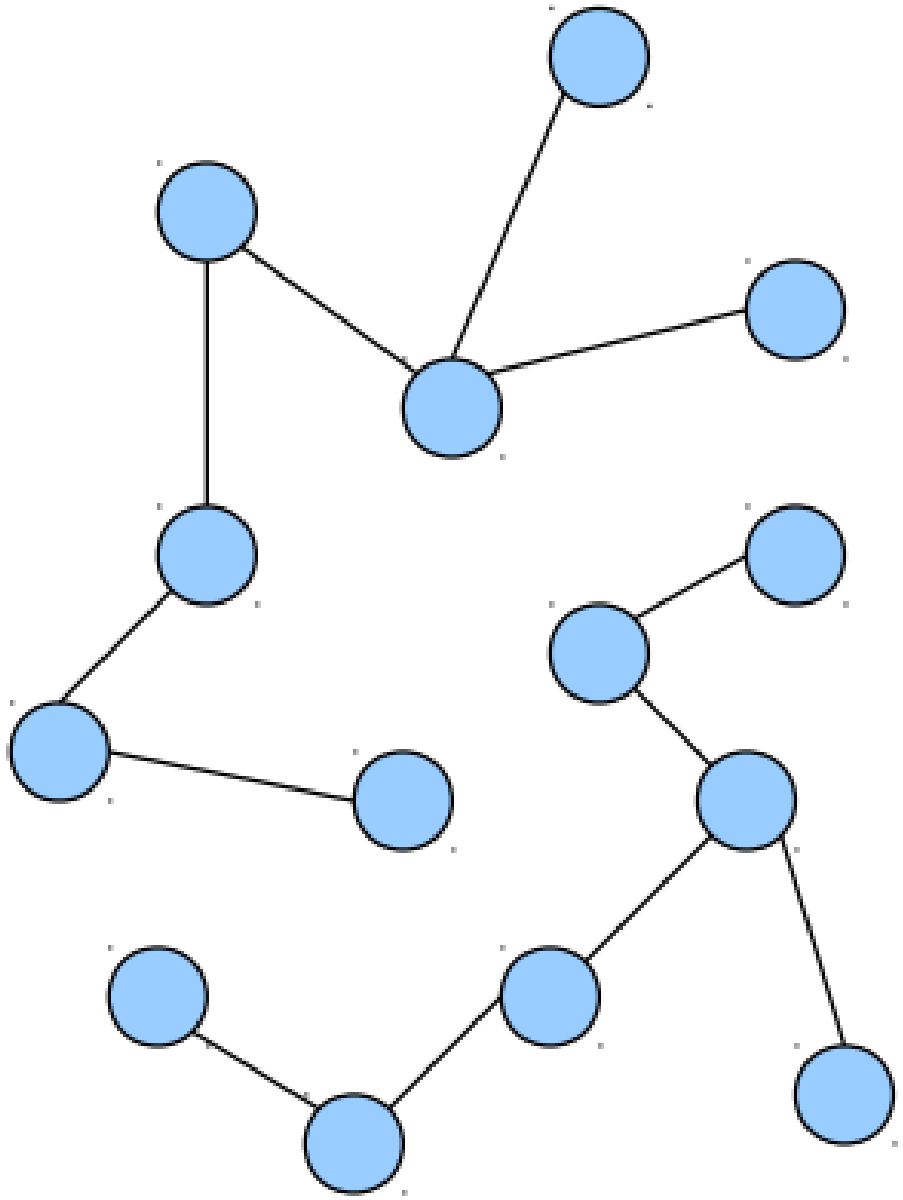
Trees

- A ***tree*** is a connected, nonempty graph with no simple cycles.
- Trees have tons of nice properties:
  - They're ***maximally acyclic*** (adding any missing edge creates a simple cycle)
  - They're ***minimally connected*** (deleting any edge disconnects the graph)



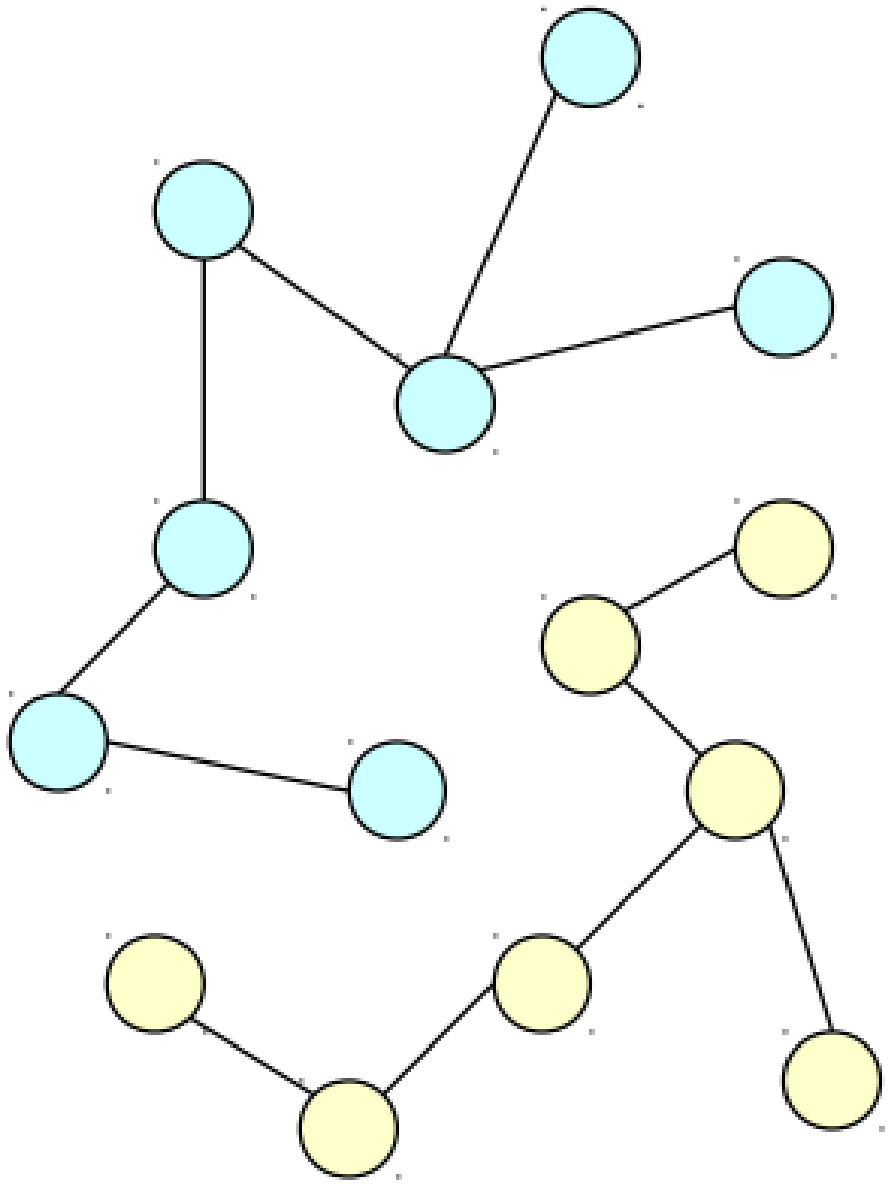
Trees

- A ***tree*** is a connected, nonempty graph with no simple cycles.
- Trees have tons of nice properties:
  - They're ***maximally acyclic*** (adding any missing edge creates a simple cycle)
  - They're ***minimally connected*** (deleting any edge disconnects the graph)



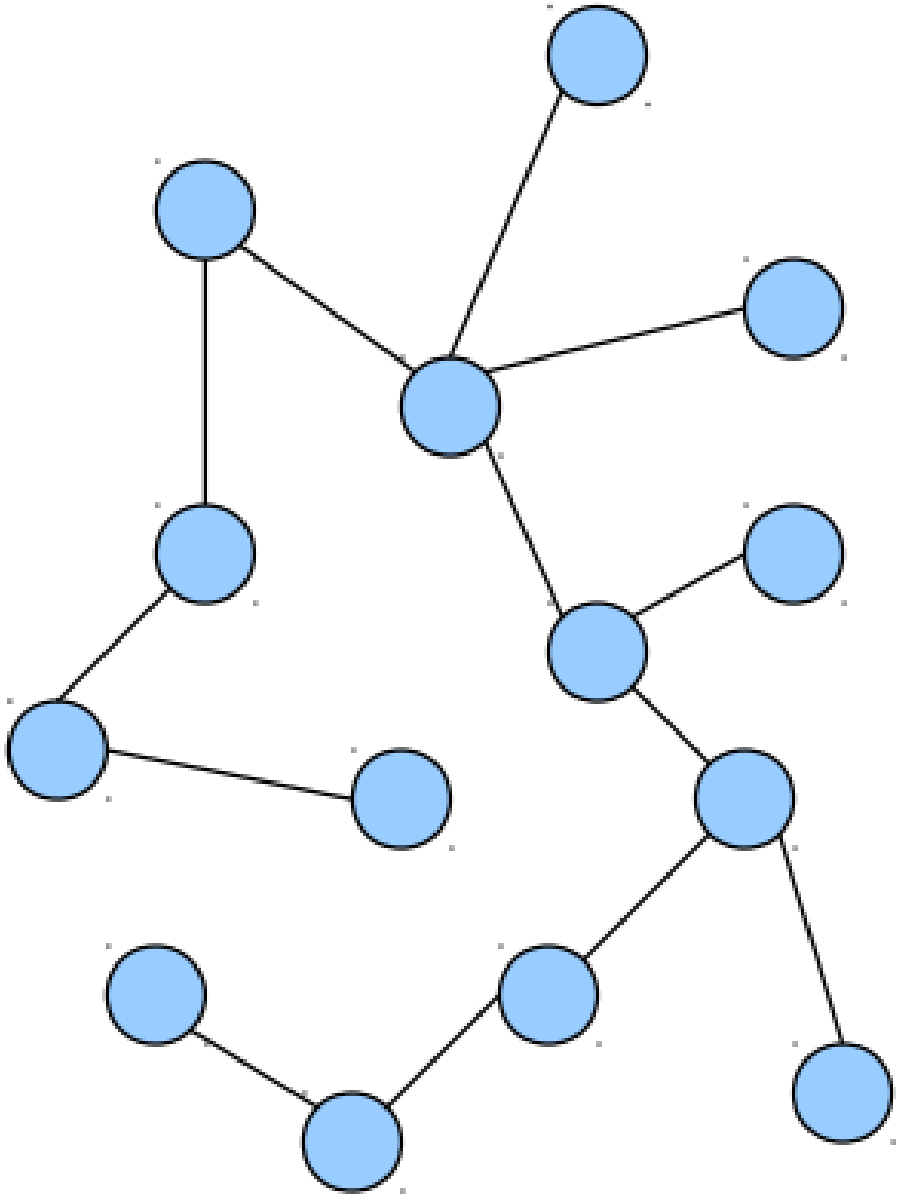
Trees

- A ***tree*** is a connected, nonempty graph with no simple cycles.
- Trees have tons of nice properties:
  - They're ***maximally acyclic*** (adding any missing edge creates a simple cycle)
  - They're ***minimally connected*** (deleting any edge disconnects the graph)



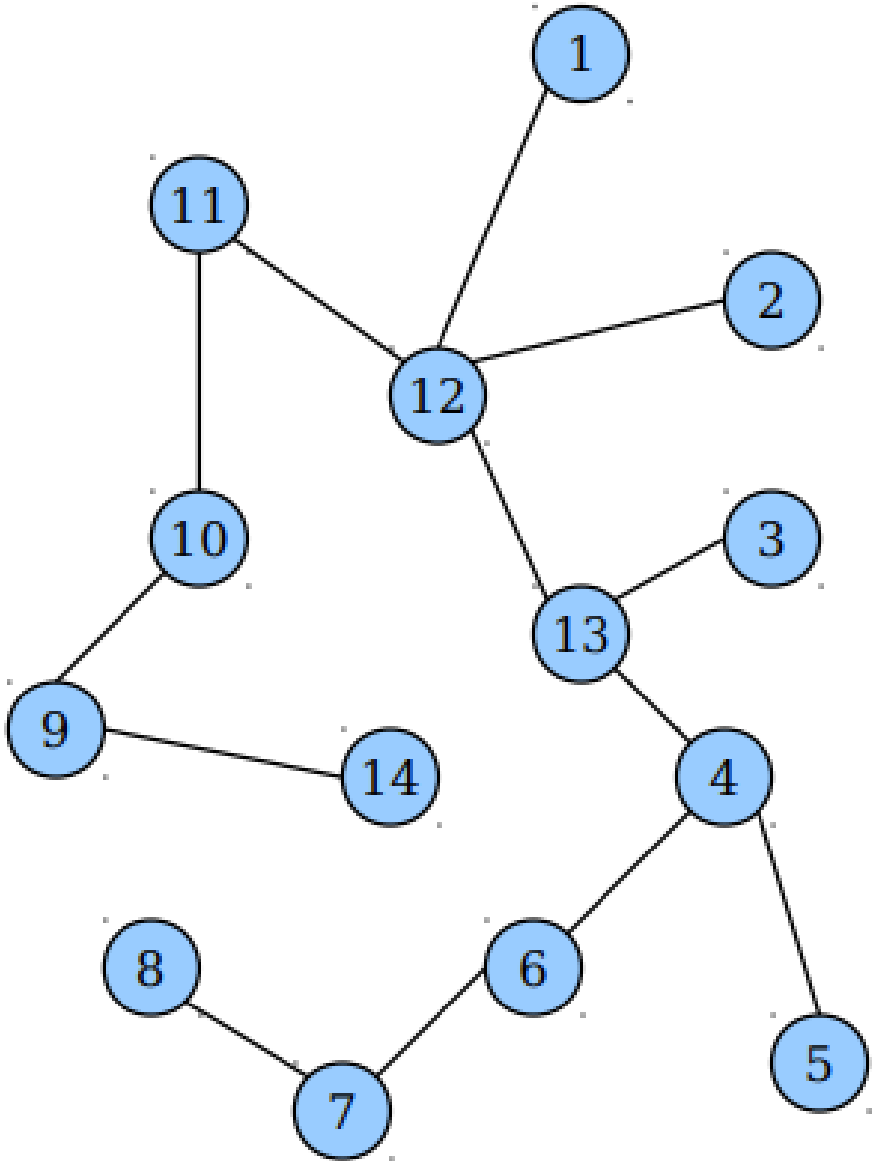
Trees

- ***Theorem:*** If T is a tree with $n \geq 1$ nodes, then T has exactly n-1 edges.
- ***Proof:*** Up next!



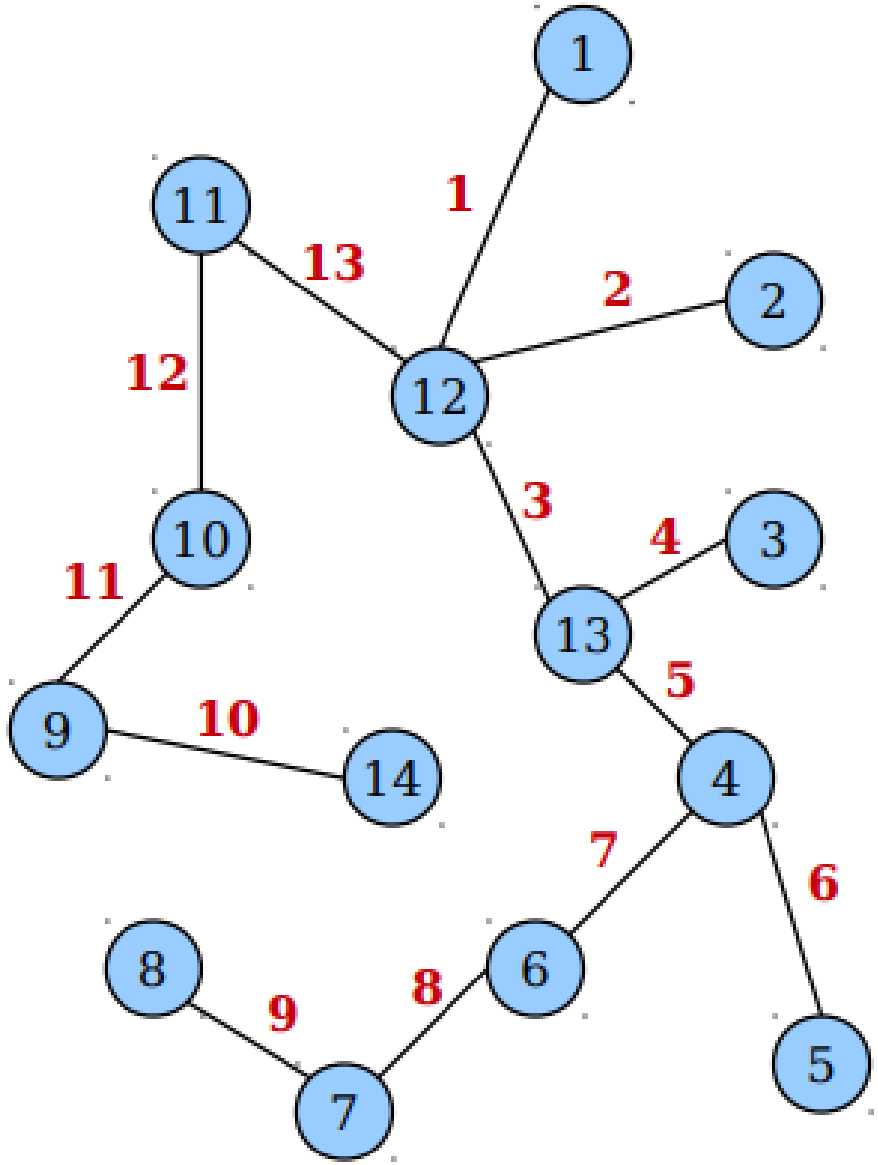
Trees

- ***Theorem:*** If T is a tree with $n \geq 1$ nodes, then T has exactly n-1 edges.
- ***Proof:*** Up next!



Trees

- ***Theorem:*** If T is a tree with $n \geq 1$ nodes, then T has exactly n-1 edges.
- ***Proof:*** Up next!



Our Base Case

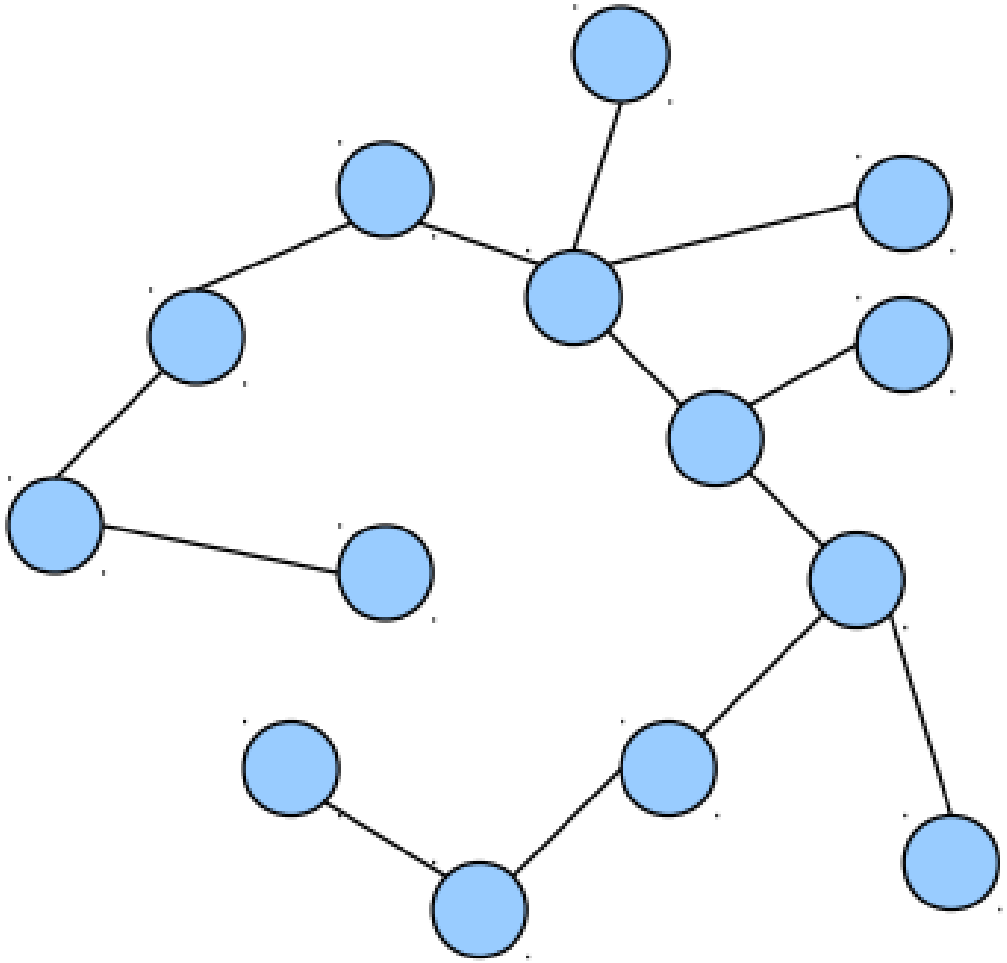
Assume any tree with at most k nodes has one more node than edge.

Assume any tree with at most k nodes has one more node than edge.

Consider an arbitrary tree with $k+1$ nodes.

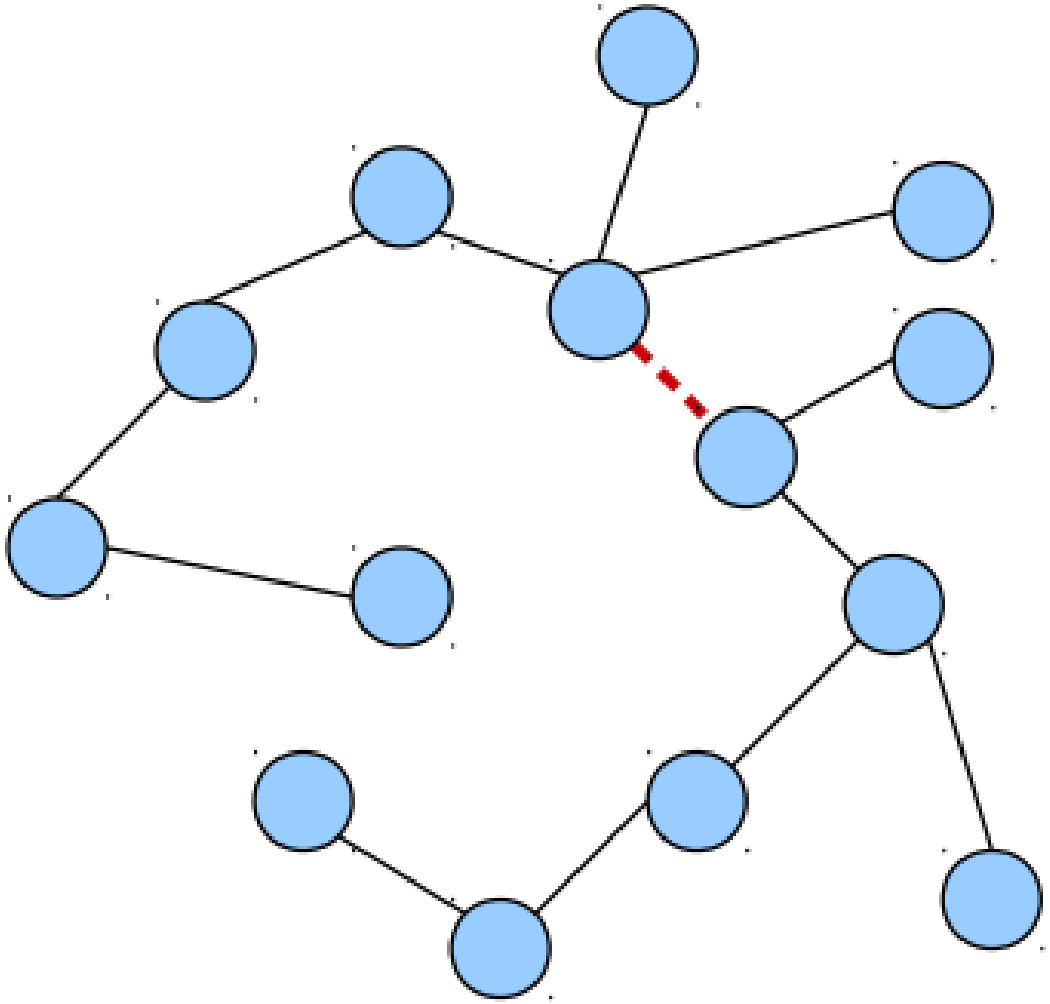
Assume any tree with at most k nodes has one more node than edge.

Consider an arbitrary tree with $k+1$ nodes.



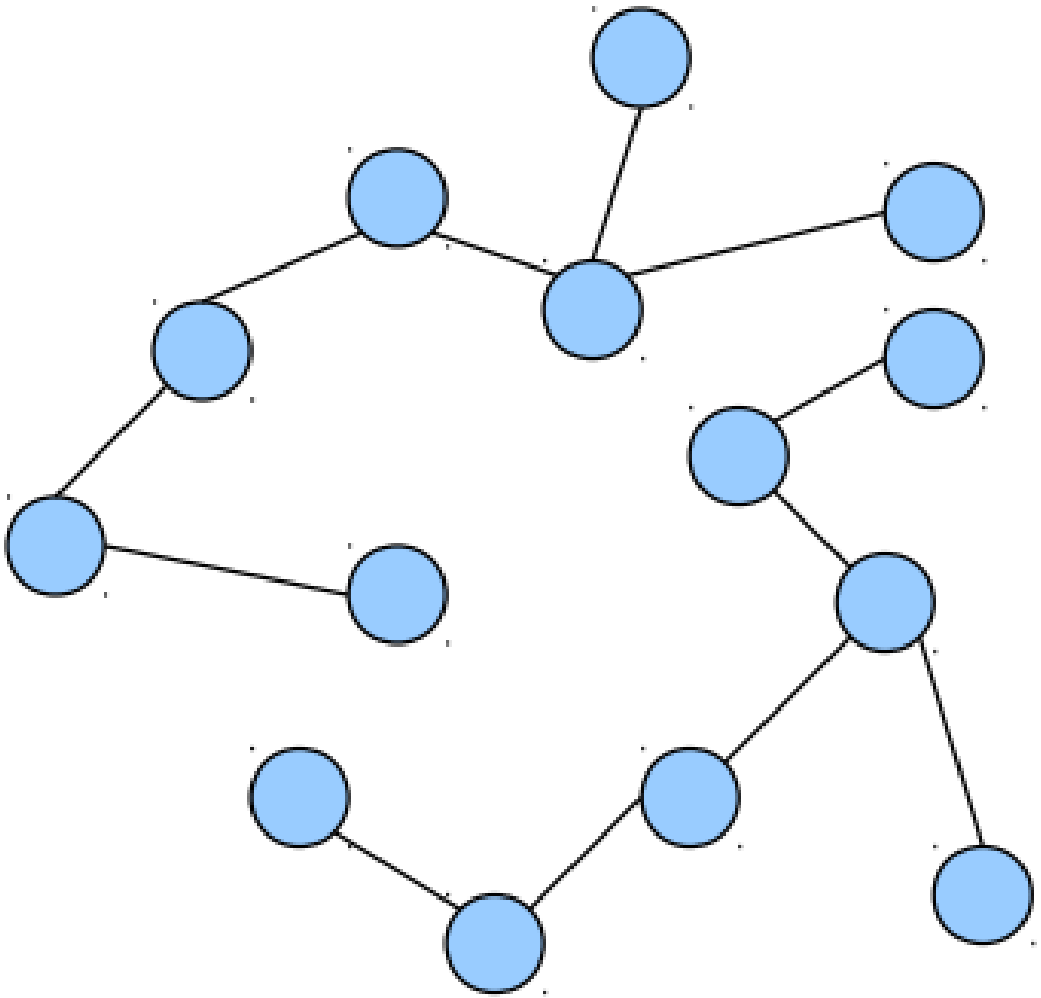
Assume any tree with at most k nodes has one more node than edge.

Consider an arbitrary tree with $k+1$ nodes.



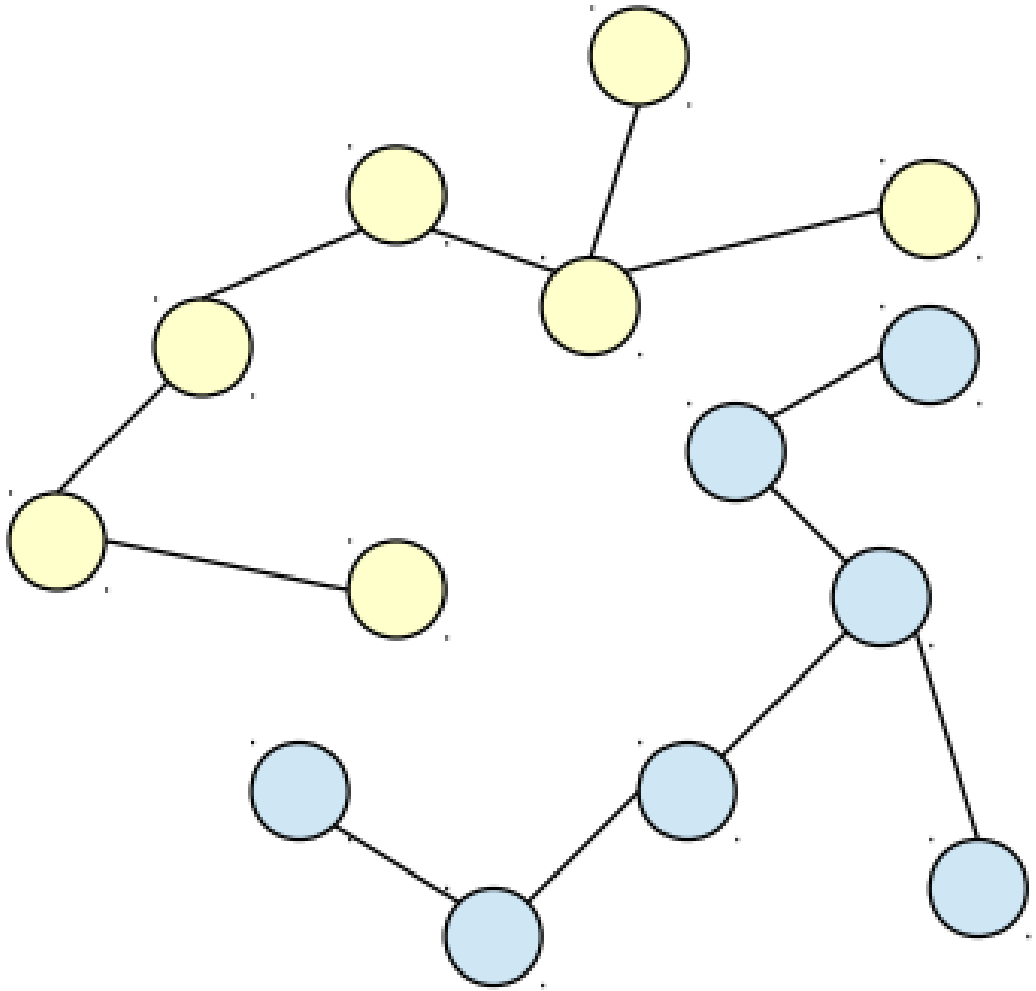
Assume any tree with at most k nodes has one more node than edge.

Consider an arbitrary tree with $k+1$ nodes.



Assume any tree with at most k nodes has one more node than edge.

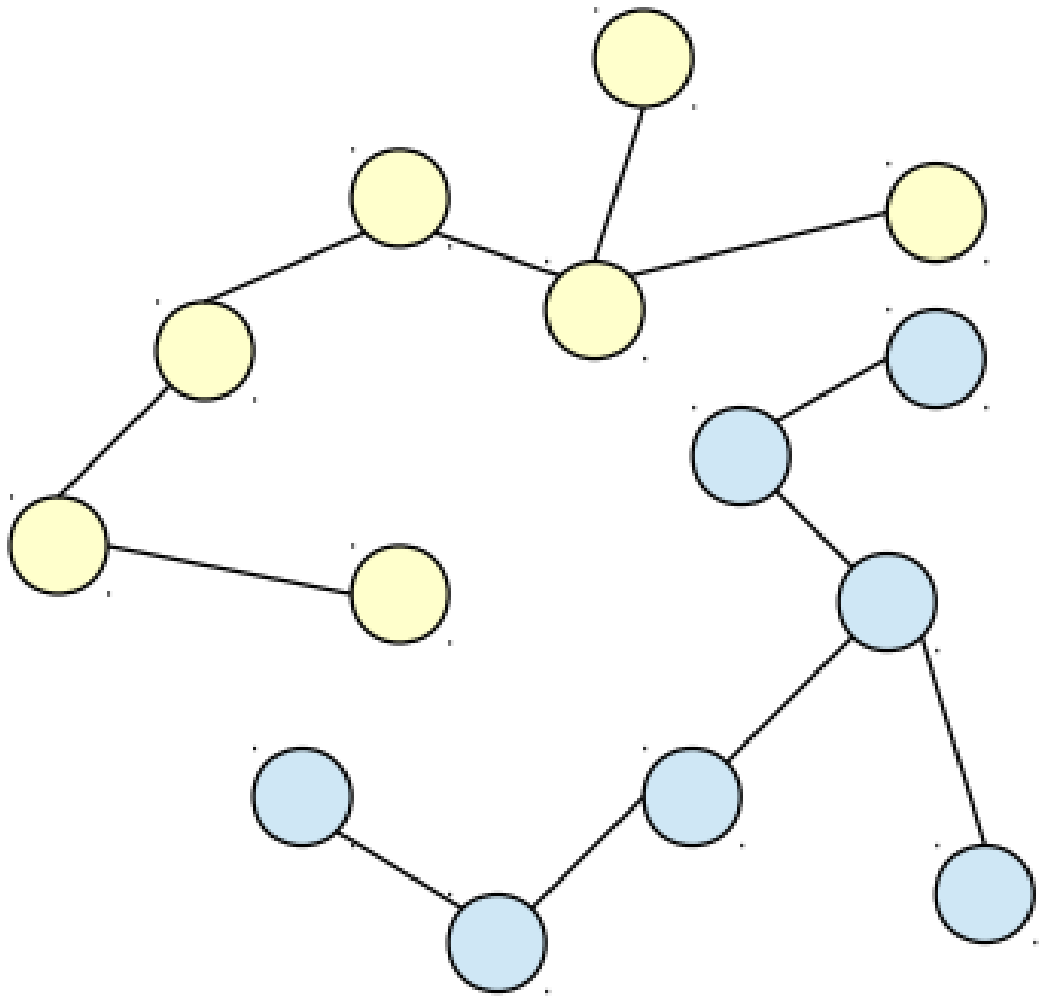
Consider an arbitrary tree with $k+1$ nodes.



Assume any tree with at most k nodes has one more node than edge.

Consider an arbitrary tree with $k+1$ nodes.

Suppose there are r nodes in the yellow tree.

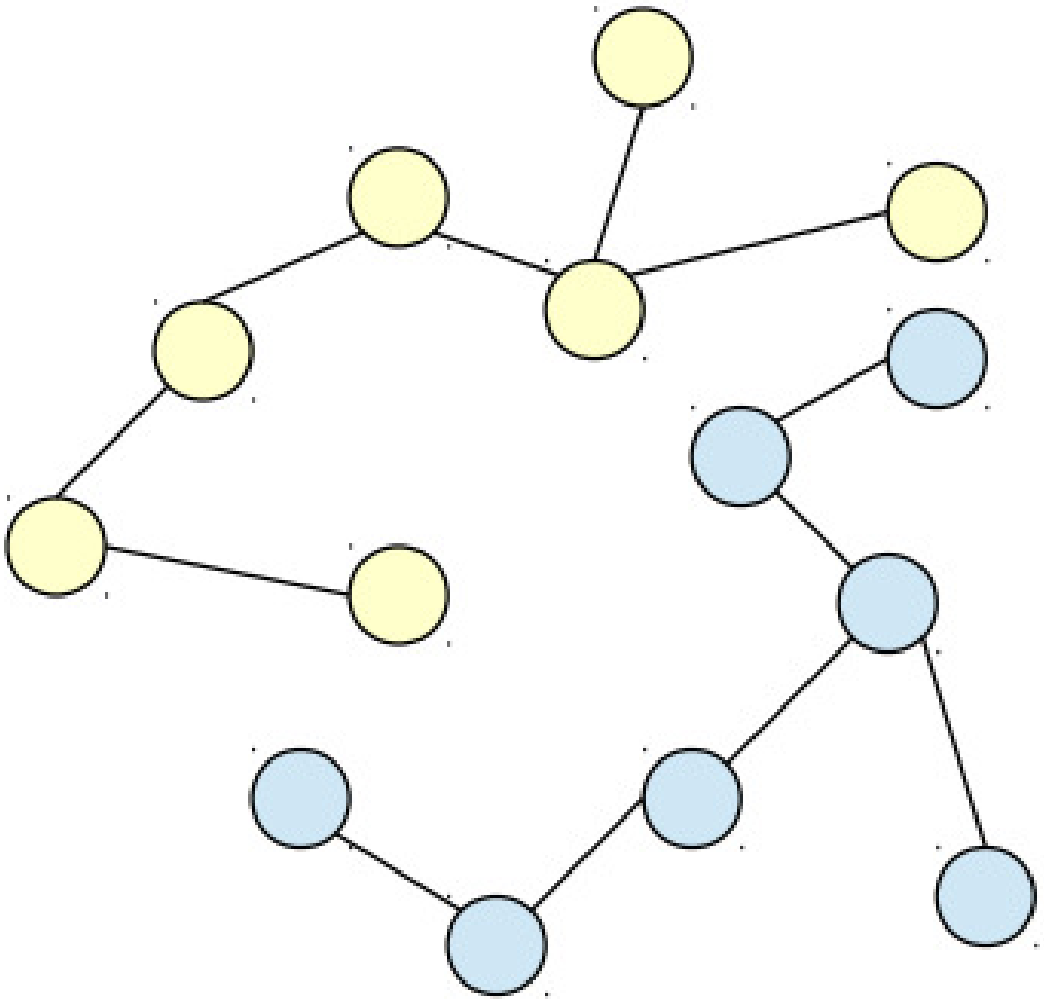


Assume any tree with at most k nodes has one more node than edge.

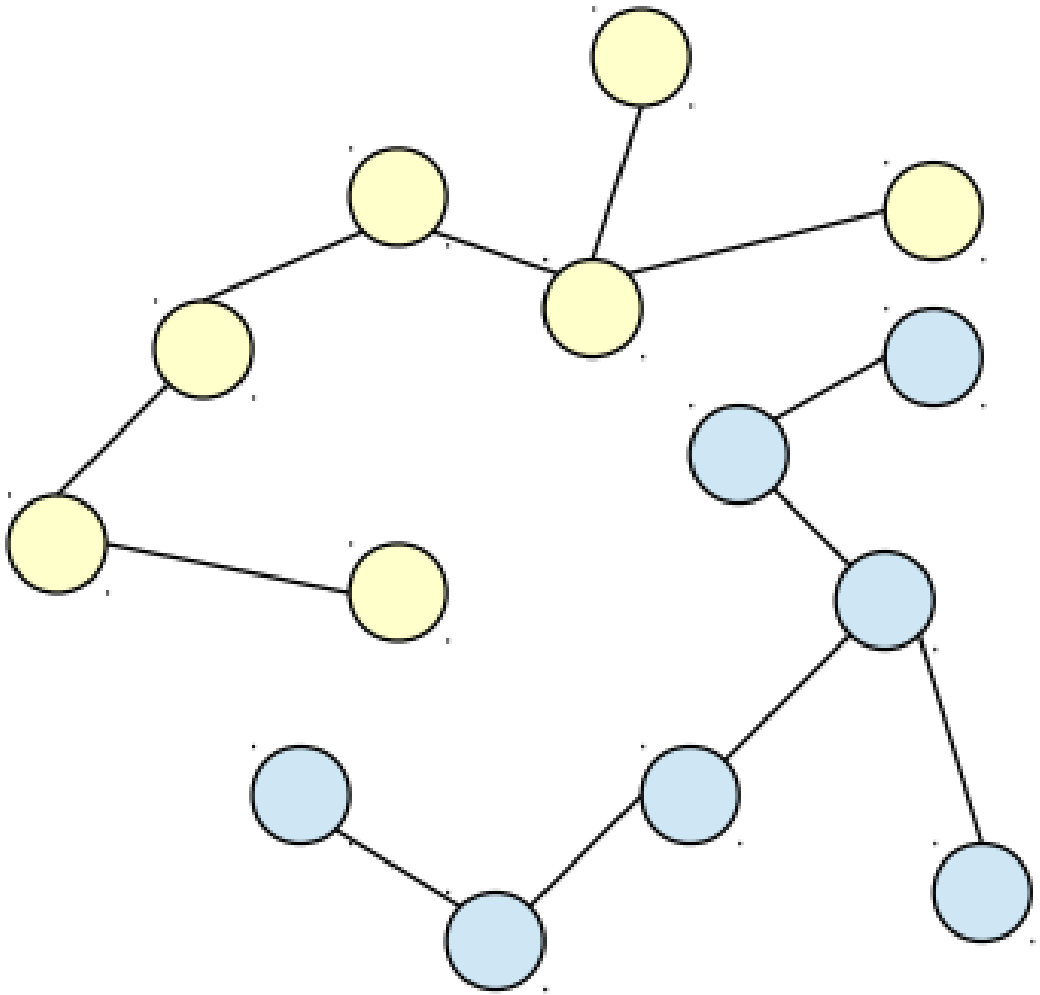
Consider an arbitrary tree with $k+1$ nodes.

Suppose there are r nodes in the yellow tree.

Then there are $(k+1)-r$ nodes in the blue tree.



Assume any tree with at most k nodes has one more node than edge.



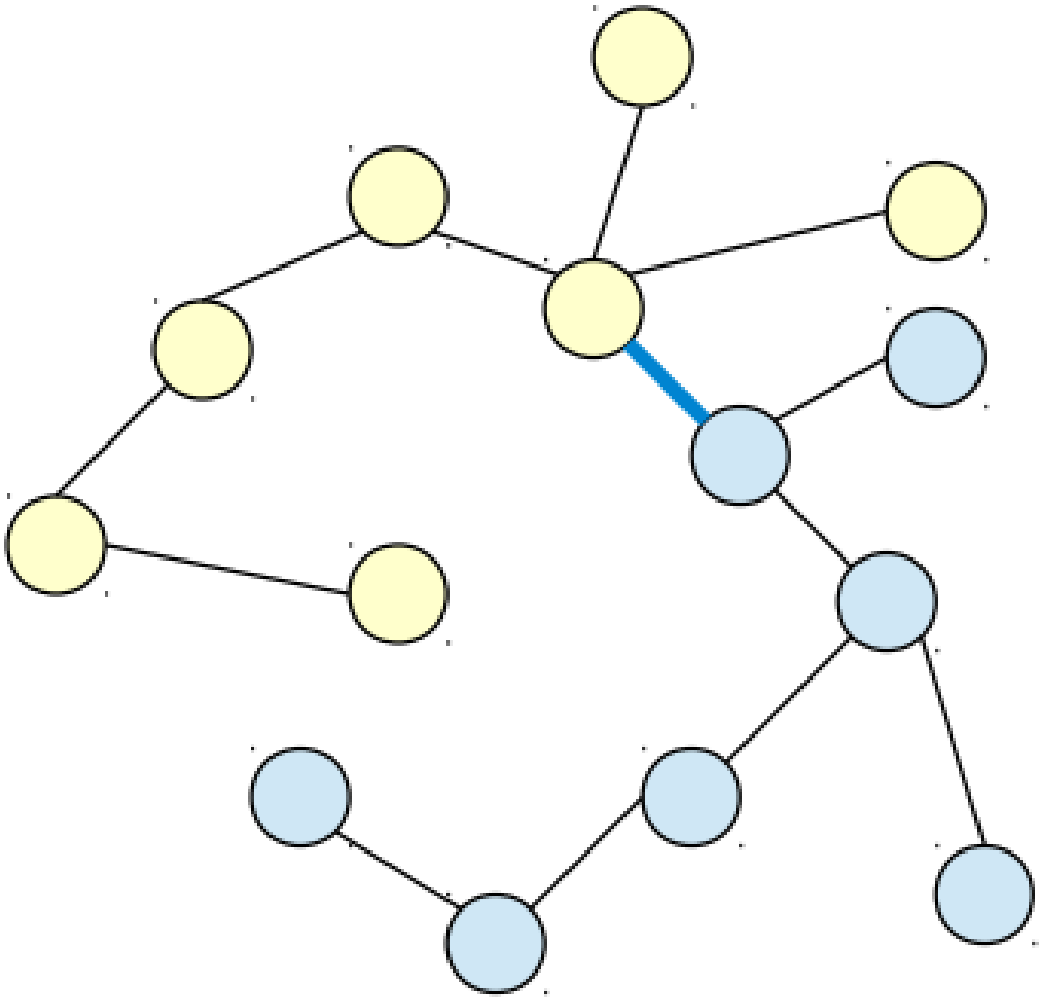
Consider an arbitrary tree with $k+1$ nodes.

Suppose there are r nodes in the yellow tree.

Then there are $(k+1)-r$ nodes in the blue tree.

There are $r-1$ edges in the yellow tree and $k-r$ edges in the blue tree.

Assume any tree with at most k nodes has one more node than edge.



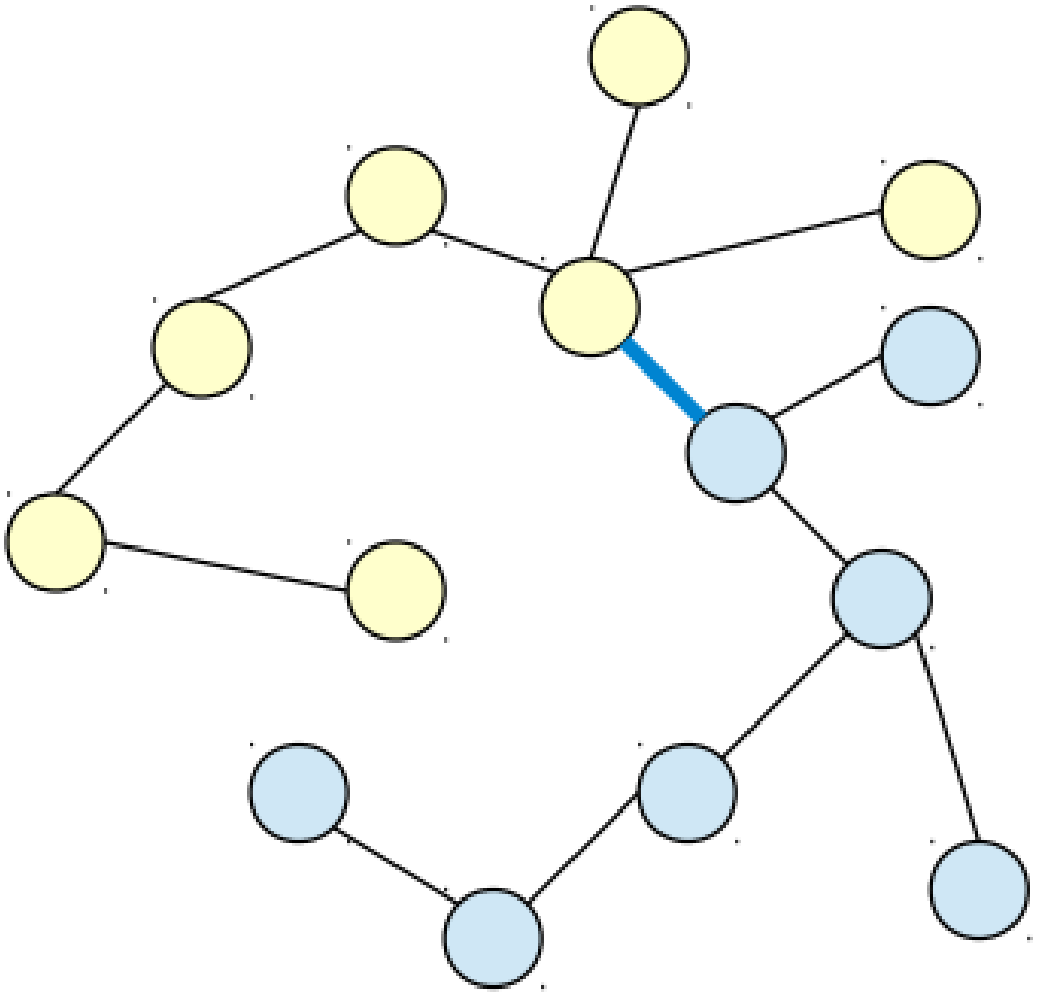
Consider an arbitrary tree with $k+1$ nodes.

Suppose there are r nodes in the yellow tree.

Then there are $(k+1)-r$ nodes in the blue tree.

There are $r-1$ edges in the yellow tree and $k-r$ edges in the blue tree.

Assume any tree with at most k nodes has one more node than edge.



Consider an arbitrary tree with $k+1$ nodes.

Suppose there are r nodes in the yellow tree.

Then there are $(k+1)-r$ nodes in the blue tree.

There are $r-1$ edges in the yellow tree and $k-r$ edges in the blue tree.

Adding in the initial edge we cut, there are $r-1 + k-r + 1 = k$ edges in the original tree.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges."

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge. We will prove $P(k+1)$, that any tree with $k+1$ nodes has k edges.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge. We will prove $P(k+1)$, that any tree with $k+1$ nodes has k edges.

Consider any tree T with $k+1$ nodes.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge. We will prove $P(k+1)$, that any tree with $k+1$ nodes has k edges.

Consider any tree T with $k+1$ nodes. Since T has at least two nodes and is connected, it must contain at least one edge.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge. We will prove $P(k+1)$, that any tree with $k+1$ nodes has k edges.

Consider any tree T with $k+1$ nodes. Since T has at least two nodes and is connected, it must contain at least one edge. Choose any edge in T and delete it.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge. We will prove $P(k+1)$, that any tree with $k+1$ nodes has k edges.

Consider any tree T with $k+1$ nodes. Since T has at least two nodes and is connected, it must contain at least one edge. Choose any edge in T and delete it. This splits T into two nonempty trees T_1 and T_2.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge. We will prove $P(k+1)$, that any tree with $k+1$ nodes has k edges.

Consider any tree T with $k+1$ nodes. Since T has at least two nodes and is connected, it must contain at least one edge. Choose any edge in T and delete it. This splits T into two nonempty trees T_1 and T_2. Every edge in T is part of T_1, is part of T_2, or is the initial edge we deleted.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge. We will prove $P(k+1)$, that any tree with $k+1$ nodes has k edges.

Consider any tree T with $k+1$ nodes. Since T has at least two nodes and is connected, it must contain at least one edge. Choose any edge in T and delete it. This splits T into two nonempty trees T_1 and T_2. Every edge in T is part of T_1, is part of T_2, or is the initial edge we deleted.

Let r be the number of nodes in T_1.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge. We will prove $P(k+1)$, that any tree with $k+1$ nodes has k edges.

Consider any tree T with $k+1$ nodes. Since T has at least two nodes and is connected, it must contain at least one edge. Choose any edge in T and delete it. This splits T into two nonempty trees T_1 and T_2. Every edge in T is part of T_1, is part of T_2, or is the initial edge we deleted.

Let r be the number of nodes in T_1. Since every node in T belongs to either T_1 or T_2, we see that T_2 has $(k+1)$-r nodes.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge. We will prove $P(k+1)$, that any tree with $k+1$ nodes has k edges.

Consider any tree T with $k+1$ nodes. Since T has at least two nodes and is connected, it must contain at least one edge. Choose any edge in T and delete it. This splits T into two nonempty trees T_1 and T_2. Every edge in T is part of T_1, is part of T_2, or is the initial edge we deleted.

Let r be the number of nodes in T_1. Since every node in T belongs to either T_1 or T_2, we see that T_2 has $(k+1)$-r nodes. Additionally, since T_1 and T_2 are nonempty, neither T_1 nor T_2 contains all the nodes from T.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge. We will prove $P(k+1)$, that any tree with $k+1$ nodes has k edges.

Consider any tree T with $k+1$ nodes. Since T has at least two nodes and is connected, it must contain at least one edge. Choose any edge in T and delete it. This splits T into two nonempty trees T_1 and T_2. Every edge in T is part of T_1, is part of T_2, or is the initial edge we deleted.

Let r be the number of nodes in T_1. Since every node in T belongs to either T_1 or T_2, we see that T_2 has $(k+1)$-r nodes. Additionally, since T_1 and T_2 are nonempty, neither T_1 nor T_2 contains all the nodes from T. Therefore, T_1 and T_2 each have between 1 and k nodes.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge. We will prove $P(k+1)$, that any tree with $k+1$ nodes has k edges.

Consider any tree T with $k+1$ nodes. Since T has at least two nodes and is connected, it must contain at least one edge. Choose any edge in T and delete it. This splits T into two nonempty trees T_1 and T_2. Every edge in T is part of T_1, is part of T_2, or is the initial edge we deleted.

Let r be the number of nodes in T_1. Since every node in T belongs to either T_1 or T_2, we see that T_2 has $(k+1)$-r nodes. Additionally, since T_1 and T_2 are nonempty, neither T_1 nor T_2 contains all the nodes from T. Therefore, T_1 and T_2 each have between 1 and k nodes. We can then apply our inductive hypothesis to see that T_1 has r-1 edges and T_2 has k-r edges.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge. We will prove $P(k+1)$, that any tree with $k+1$ nodes has k edges.

Consider any tree T with $k+1$ nodes. Since T has at least two nodes and is connected, it must contain at least one edge. Choose any edge in T and delete it. This splits T into two nonempty trees T_1 and T_2. Every edge in T is part of T_1, is part of T_2, or is the initial edge we deleted.

Let r be the number of nodes in T_1. Since every node in T belongs to either T_1 or T_2, we see that T_2 has $(k+1)$-r nodes. Additionally, since T_1 and T_2 are nonempty, neither T_1 nor T_2 contains all the nodes from T. Therefore, T_1 and T_2 each have between 1 and k nodes. We can then apply our inductive hypothesis to see that T_1 has r-1 edges and T_2 has k-r edges. Thus the total number of edges in T is $1 + (r-1) + (k-r) = k$, as required.

Theorem: If T is a tree with $n \geq 1$ nodes, then T has n-1 edges.

Proof: Let $P(n)$ be the statement "any tree with n nodes has n-1 edges." We will prove by induction that $P(n)$ holds for all $n \geq 1$, from which the theorem follows.

As a base case, we will prove $P(1)$, that any tree with 1 node has 0 edges. Any such tree has single node, so it cannot have any edges.

Now, assume for some arbitrary $k \geq 1$ that $P(1)$, $P(2)$, ..., and $P(k)$ are true, so any tree with between 1 and k nodes has one more node than edge. We will prove $P(k+1)$, that any tree with $k+1$ nodes has k edges.

Consider any tree T with $k+1$ nodes. Since T has at least two nodes and is connected, it must contain at least one edge. Choose any edge in T and delete it. This splits T into two nonempty trees T_1 and T_2. Every edge in T is part of T_1, is part of T_2, or is the initial edge we deleted.

Let r be the number of nodes in T_1. Since every node in T belongs to either T_1 or T_2, we see that T_2 has $(k+1)$-r nodes. Additionally, since T_1 and T_2 are nonempty, neither T_1 nor T_2 contains all the nodes from T. Therefore, T_1 and T_2 each have between 1 and k nodes. We can then apply our inductive hypothesis to see that T_1 has r-1 edges and T_2 has k-r edges. Thus the total number of edges in T is $1 + (r-1) + (k-r) = k$, as required. Therefore, $P(k+1)$ is true, completing the induction. ■