Graphs and Relations

Friday Four Square!
4:15PM, Outside Gates

Announcements

- Problem Set 1 due right now.
- Problem Set 2 out.
  - Checkpoint due Monday, October 8.
  - Assignment due Friday, October 12.
  - Play around with induction and its applications!

Mathematical Structures

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

Some Formalisms

The Cartesian Product

- Recall: The **power set** $\wp(S)$ of a set is the set of all its subsets.
- The **Cartesian Product** of $A \times B$ of two sets is defined as

$$A \times B \equiv \{ \ (a, b) \mid a \in A \text{ and } b \in B \ \}$$

$$\underset{\mathrm{A}}{\left\{ 0,\ 1,\ 2 \right\}} \times \underset{\mathrm{B}}{\left\{ \mathrm{a},\ \mathrm{b},\ \mathrm{c} \right\}} = \left\{ \begin{array}{l} \mathbf{(0,\ a),(0,\ b),(0,\ c),} \\ \mathbf{(1,\ a),(1,\ b),(1,\ c),} \\ \mathbf{(2,\ a),(2,\ b),(2,\ c)} \end{array} \right\}$$

The Cartesian Product

- Recall: The **power set** $\wp(S)$ of a set is the set of all its subsets.
- The **Cartesian Product** of $A \times B$ of two sets is defined as

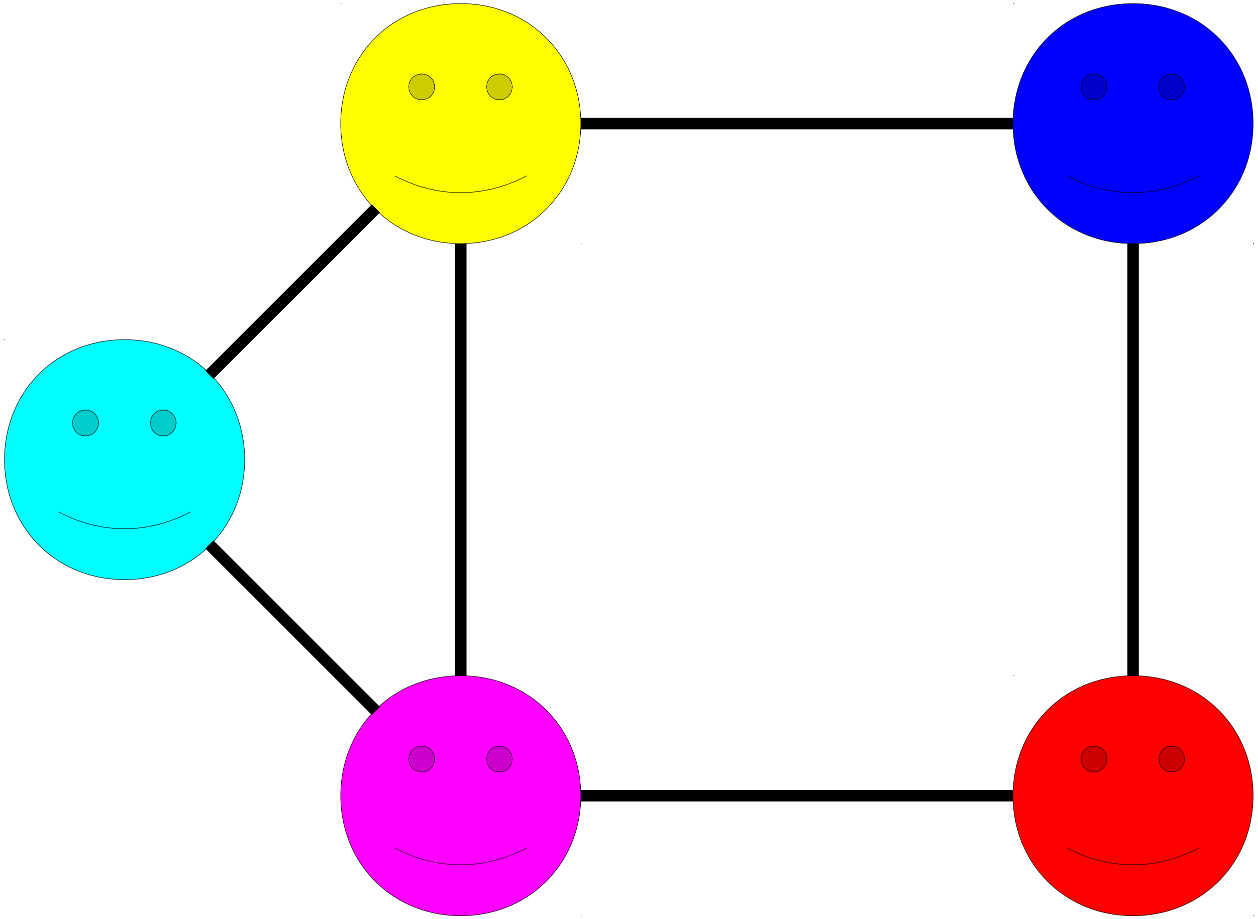
$$A \times B \equiv \{\ (a, b) \mid a \in A \text{ and } b \in B\ \}$$

- We denote $A^k \equiv \underbrace{A \times A \times \ldots \times A}_{k \text{ times}}$

$$\underset{\text{A}}{\left\{0, 1, 2\right\}}^2 = \left\{\begin{array}{l} (0, 0),(0, 1),(0, 2), \\ (1, 0),(1, 1),(1, 2), \\ (2, 0),(2, 1),(2, 2) \end{array}\right\}$$

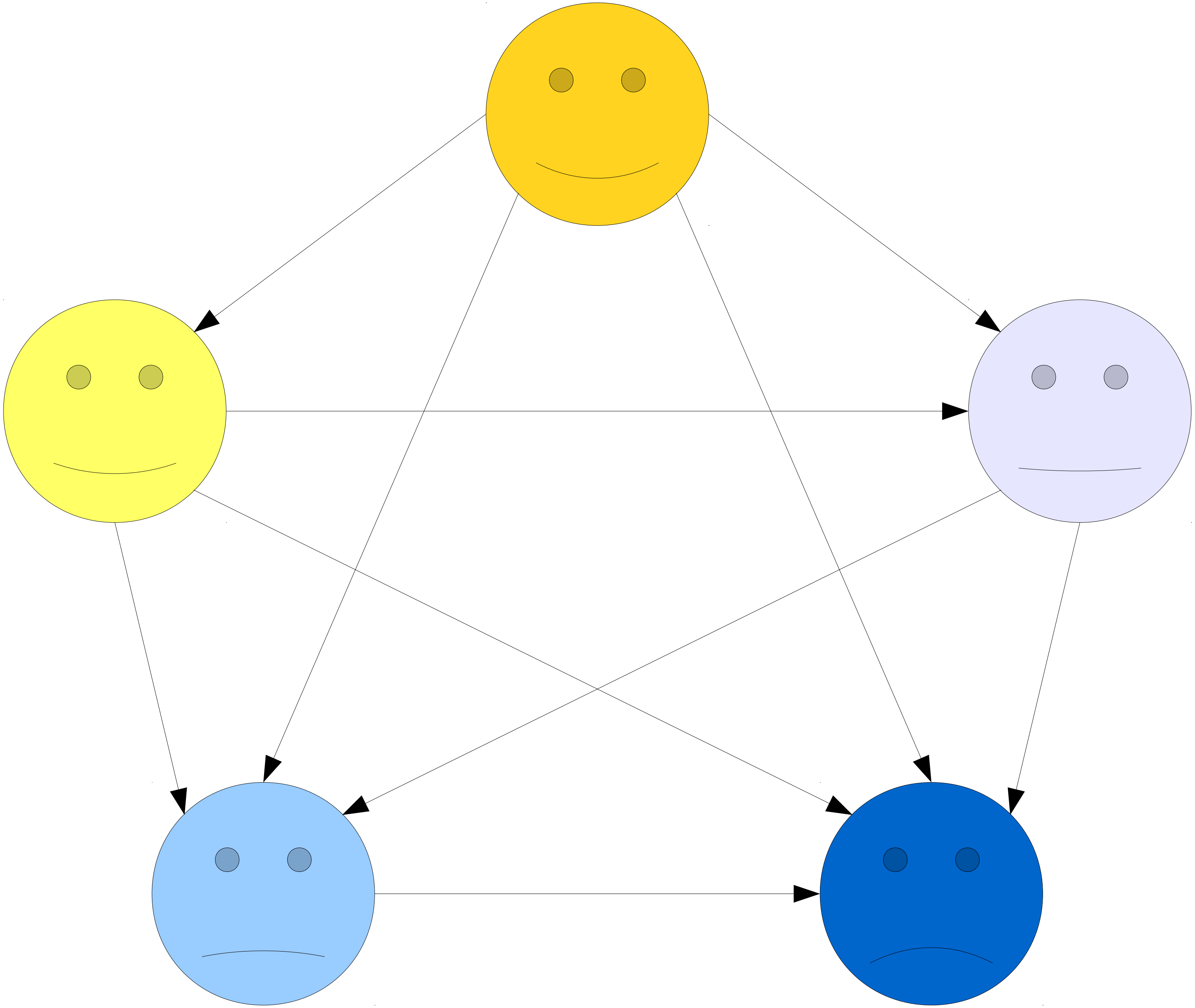
Graphs

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

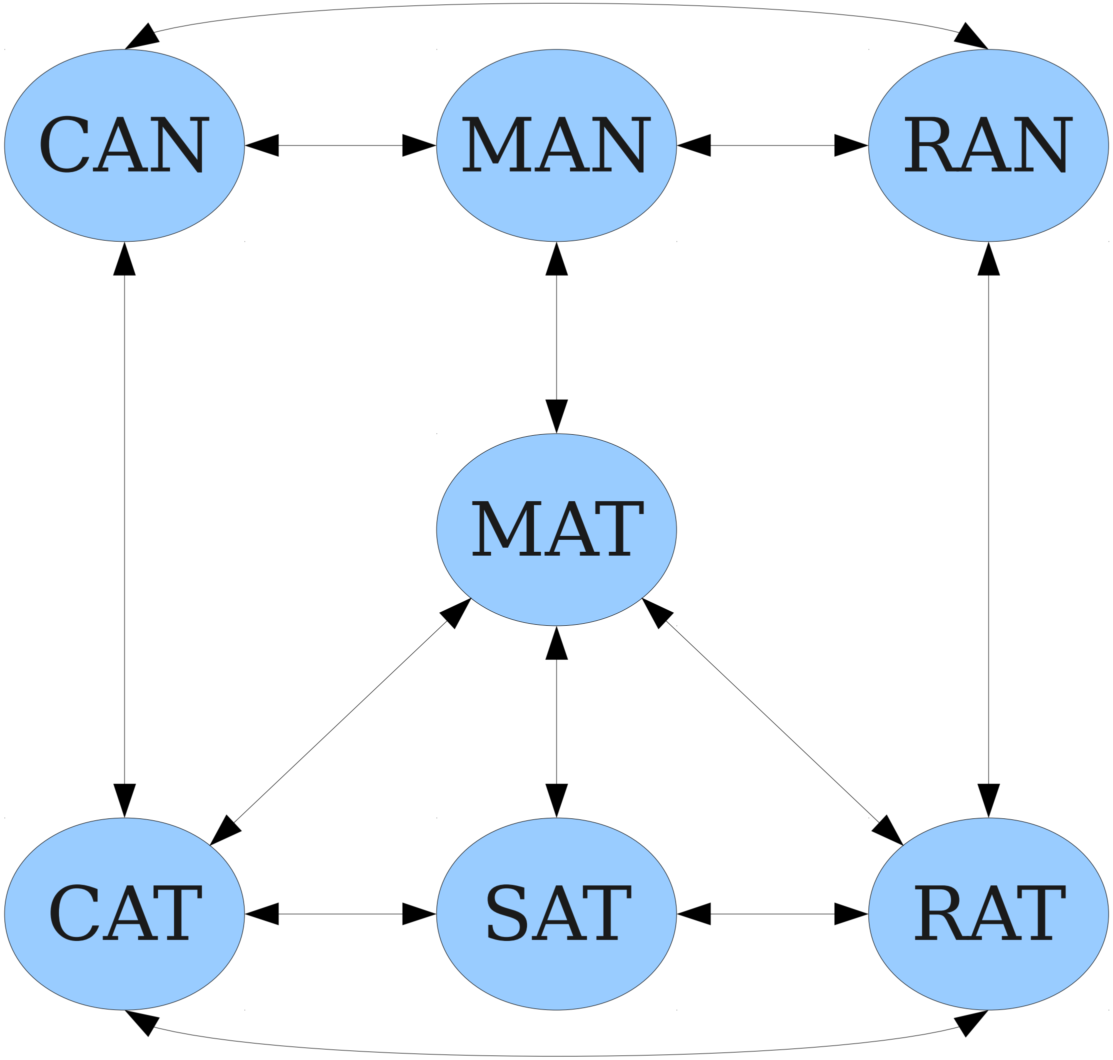


A graph consists of a set of **nodes** (or ***vertices***) connected by **edges** (or ***arcs***)

Some graphs are **directed**.

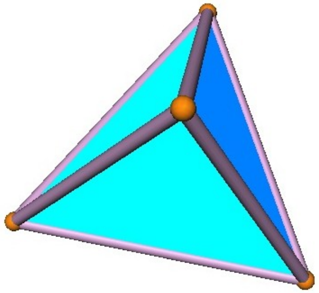


Some graphs are **undirected**.

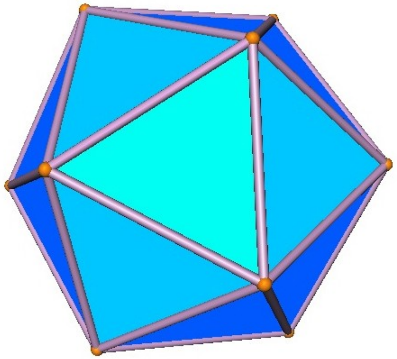


You can think of them as directed graphs with edges both ways.

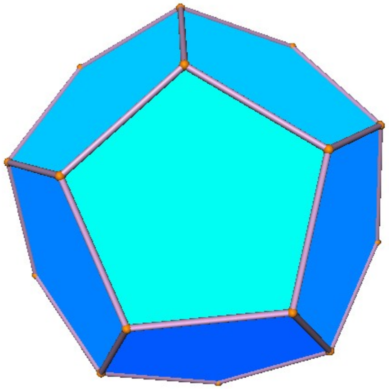
Graphs are Everywhere!



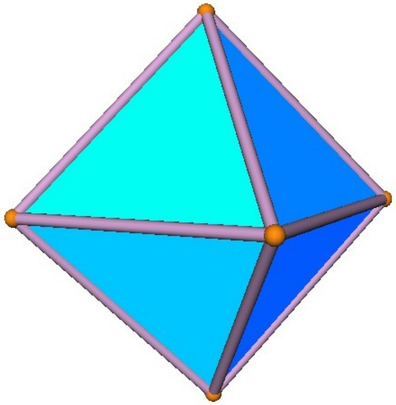
Tetrahedron



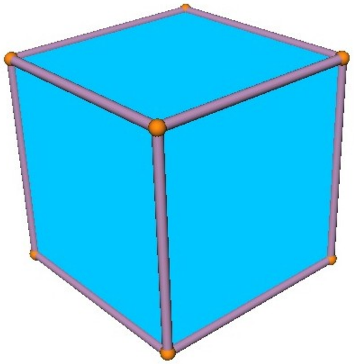
Icosahedron



Dodecahedron

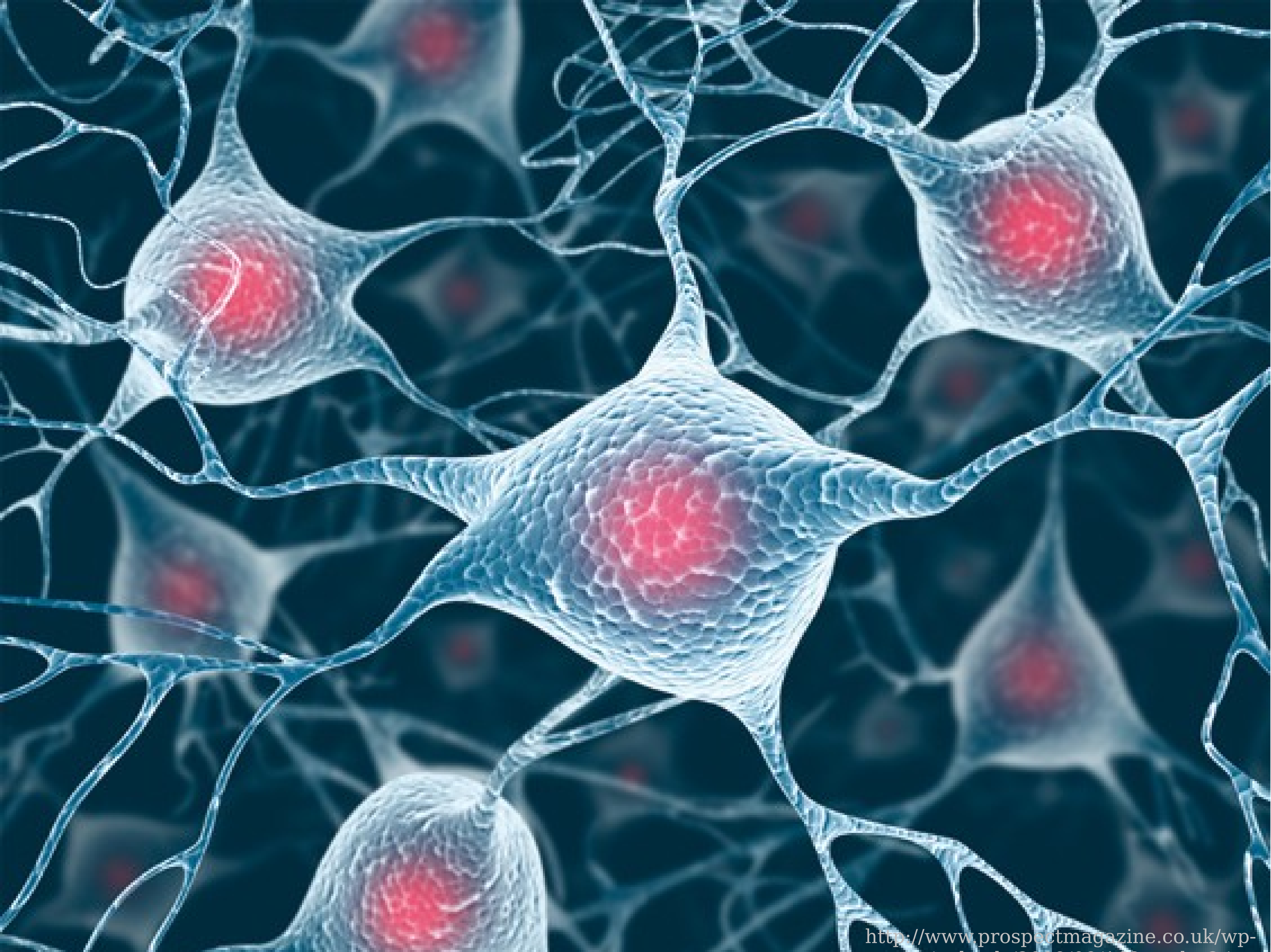


Octahedron



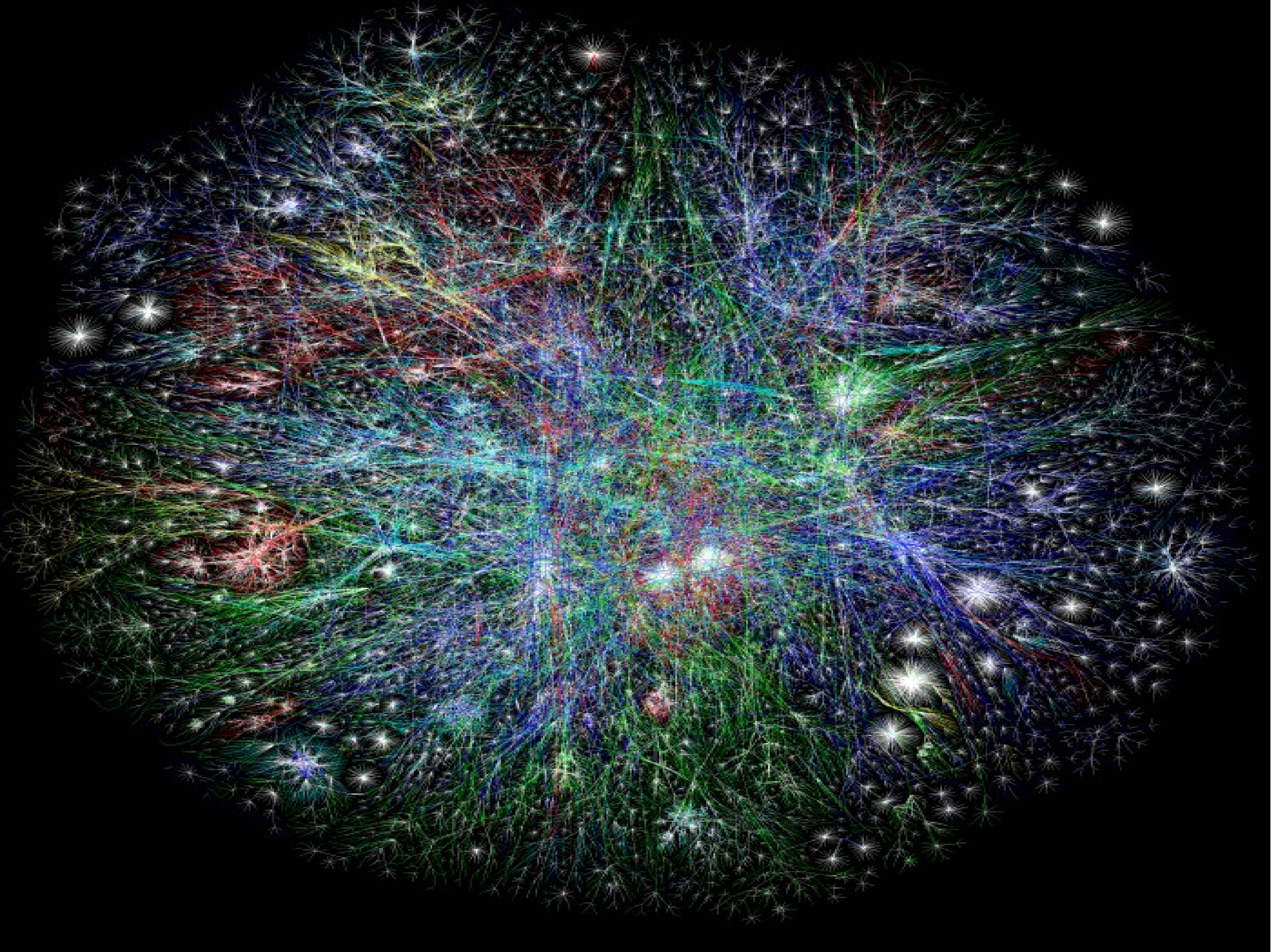
Cube

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



http://www.prospectmagazine.co.uk/wp-

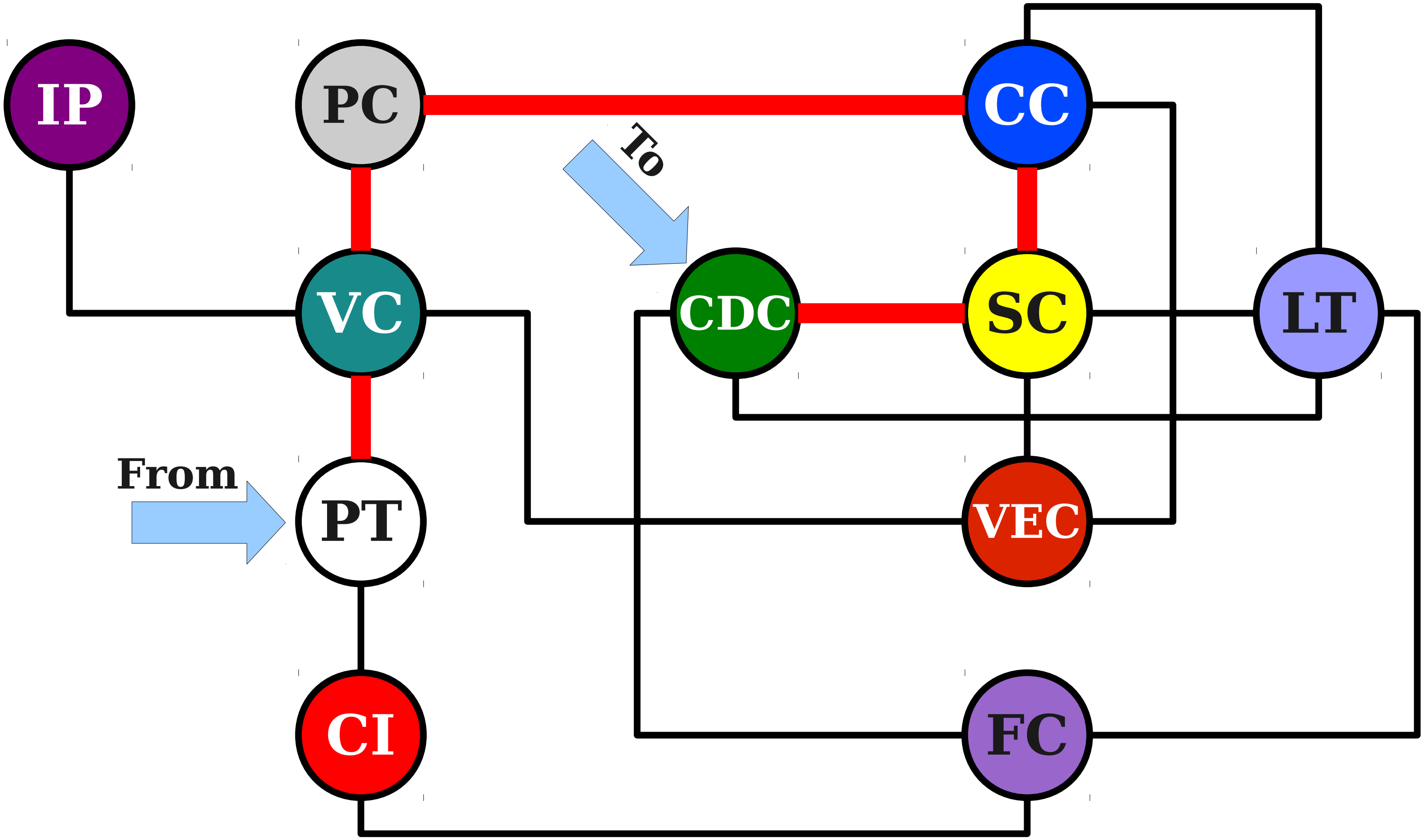
Me too!



Formalisms

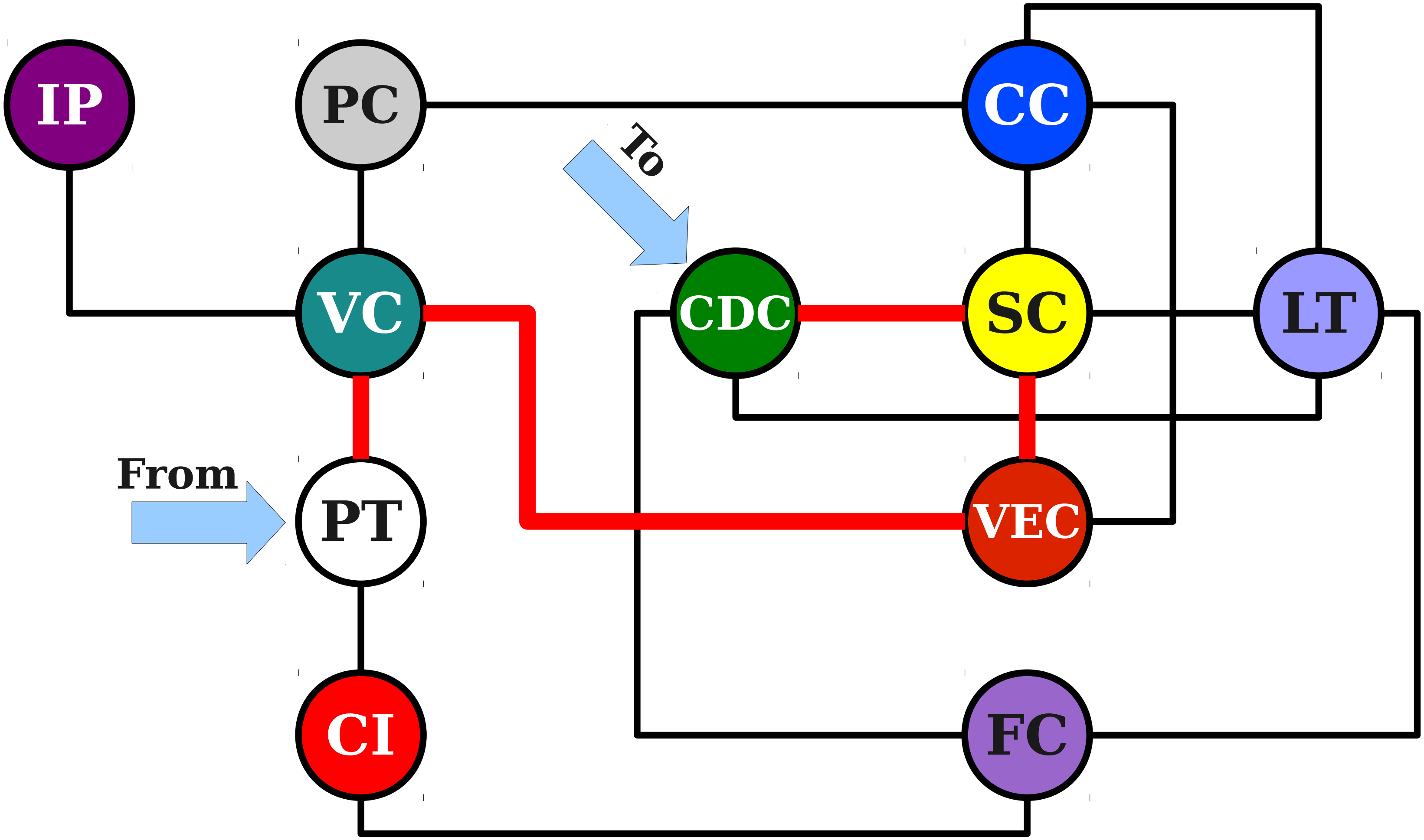
- A **graph** is an ordered pair $G = (V, E)$ where
  - V is a set of the **vertices** (nodes) of the graph.
  - E is a set of the **edges** (arcs) of the graph.
- E can be a set of ordered pairs or unordered pairs.
  - If E consists of ordered pairs, G is a **directed graph**.
  - If E consists of unordered pairs, G is an **undirected graph**.
- Each edge is an pair of the **start** and **end** (or **source** and **sink**) of the edge.

Navigating a Graph



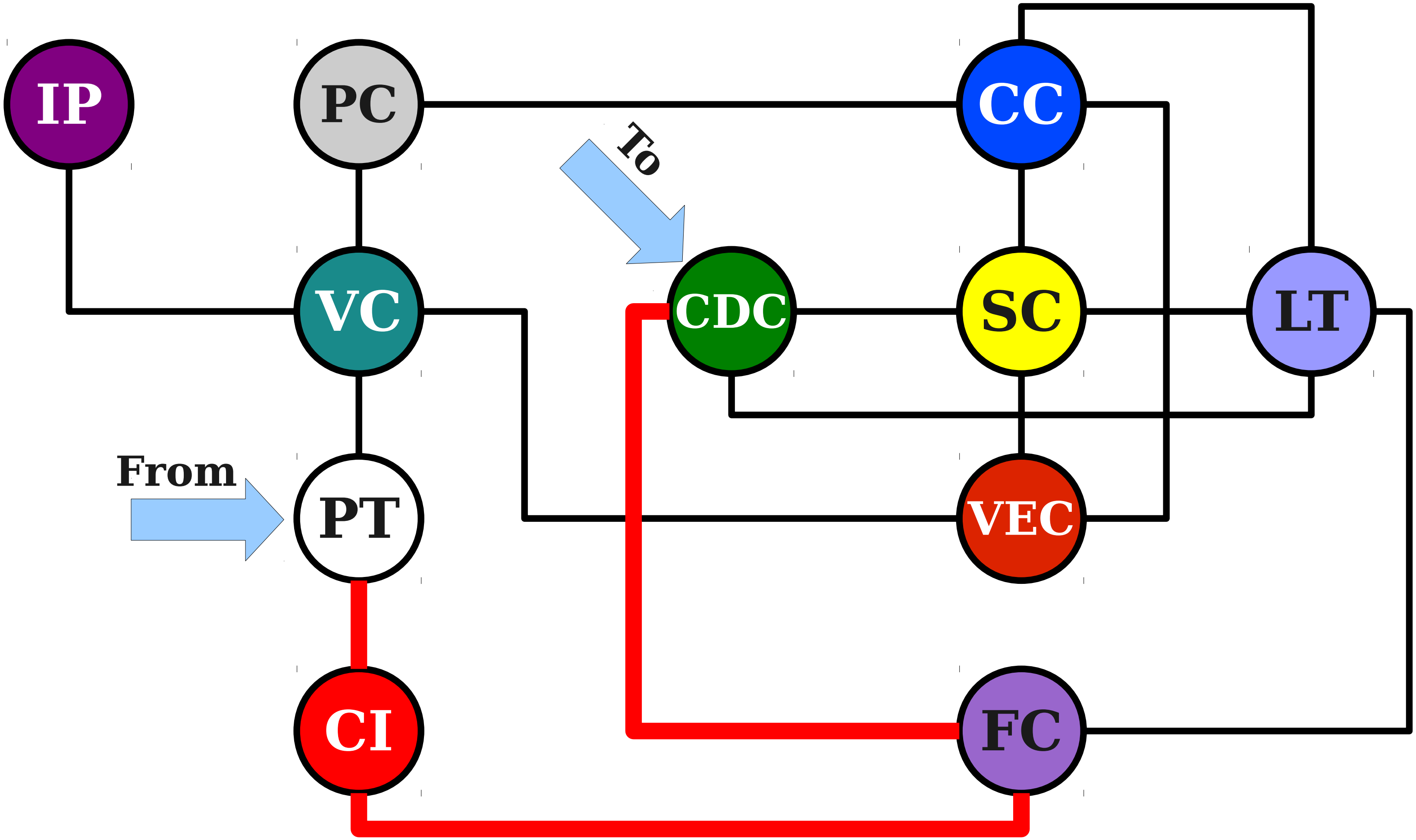
PT → VC → PC → CC → SC → CDC

Navigating a Graph



PT → VC → VEC → SC → CDC

Navigating a Graph

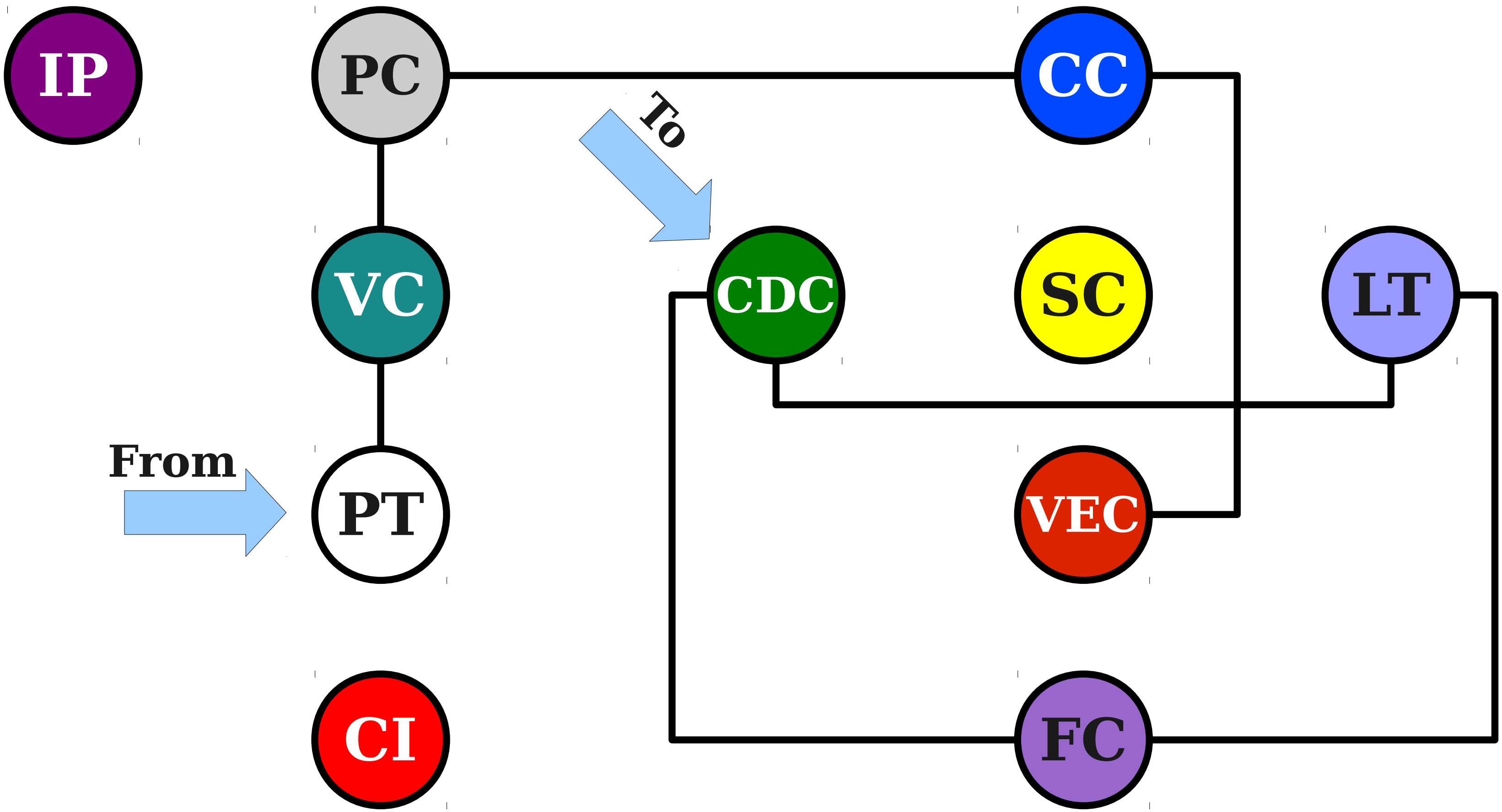


PT → CI → FC → CDC

A **path** from v_1 to v_n is a sequence of edges $((v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n))$.

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

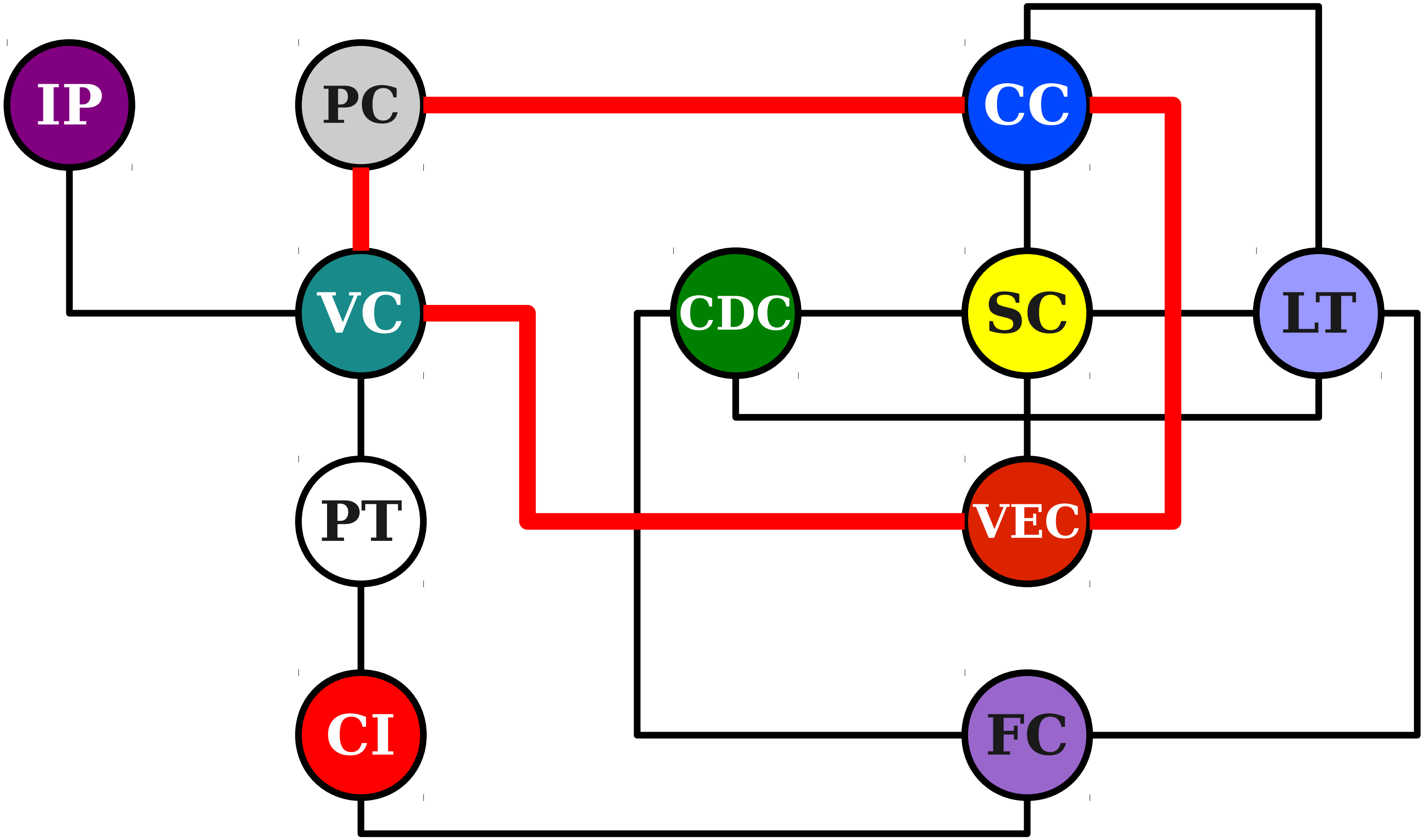
Navigating a Graph



A node v is **reachable** from node u iff there is a path from u to v.

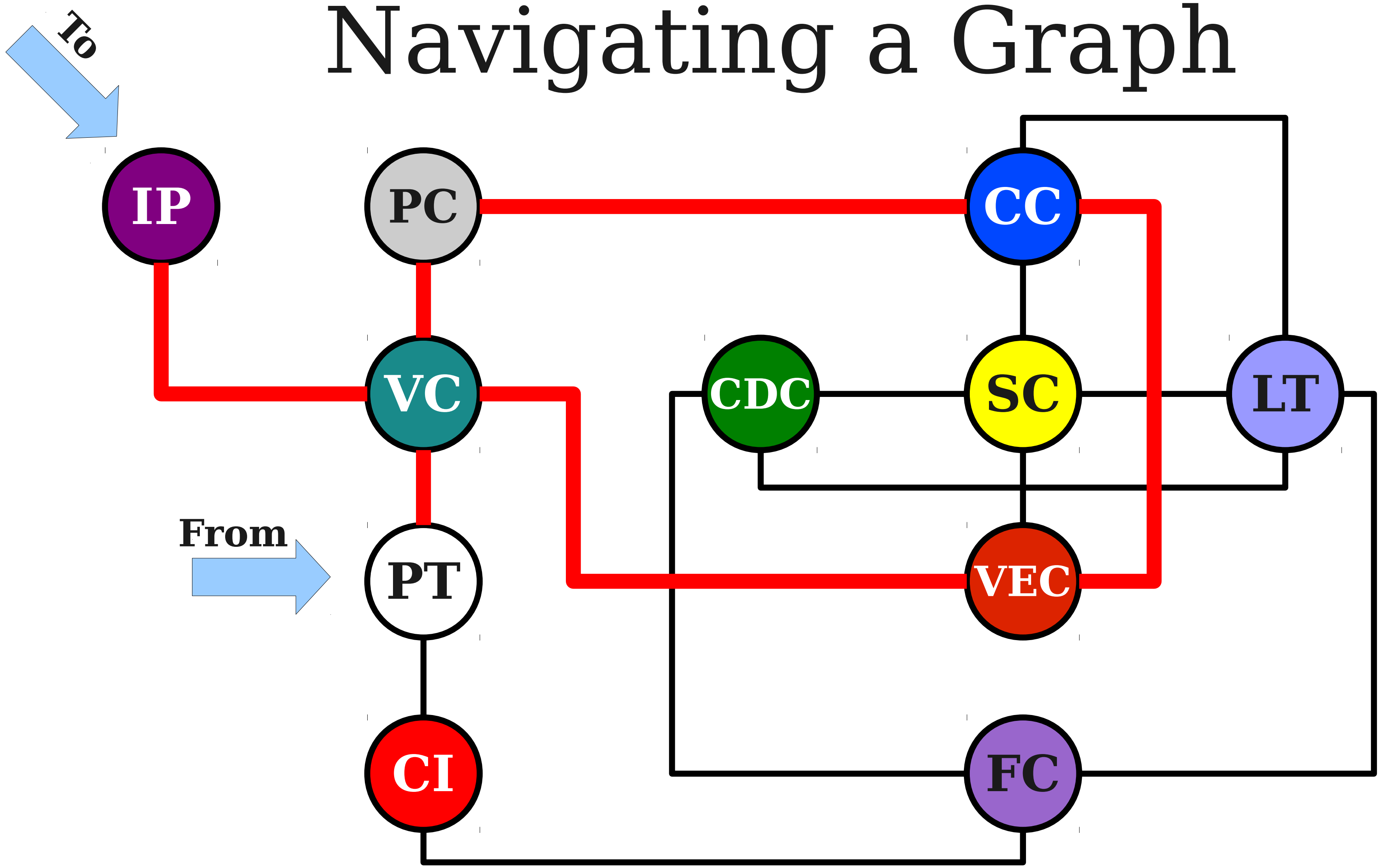
We denote this as $\boldsymbol{u \to v}$.

Navigating a Graph



PC → CC → VEC → VC → PC → CC → VEC → VC → PC

Navigating a Graph



PT → VC → PC → CC → VEC → VC → IP

A **cycle** in a graph is a path

$$((v_1, v_2), \ldots, (v_n, v_1))$$

that starts and ends at the same node.

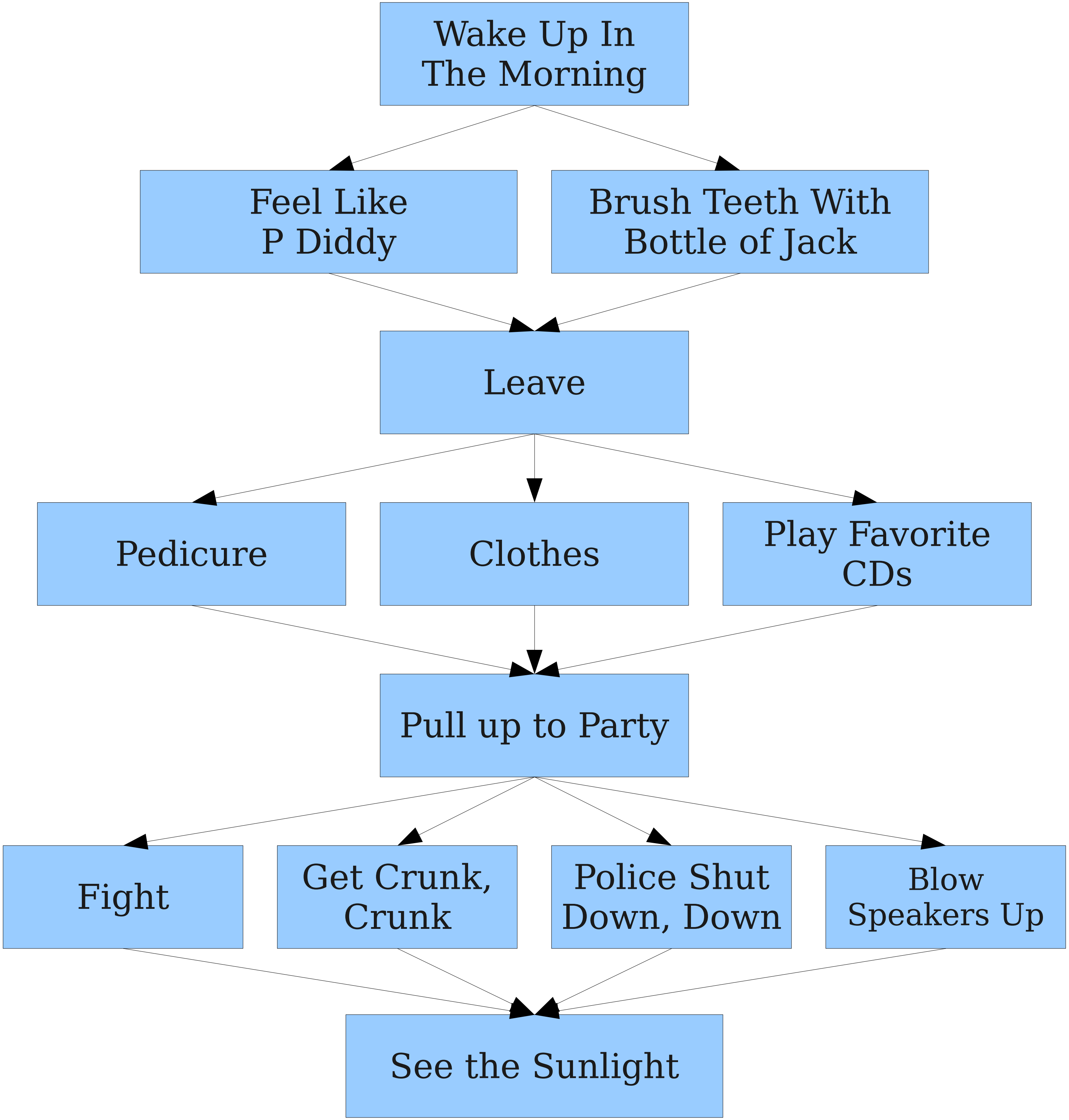
A **simple path** is a path that
does not repeat any nodes or edges.

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

Summary of Terminology

- A **path** is a series of edges connecting two nodes.
  - The **length** of a path is the number of edges in the path.
  - A node *v* is **reachable** from u if there is a path from *u* to *v*.
- A **cycle** is a path from a node to itself.
- A **simple path** is a path with no duplicate nodes or edges.
- A **simple cycle** is a cycle with no duplicate nodes or edges (except the start/end node).

Representing Prerequisites



Wake Up In The Morning

Feel Like P Diddy

Brush Teeth With Bottle of Jack

Leave

Pedicure

Clothes

Play Favorite CDs

Pull up to Party

Fight

Get Crunk, Crunk

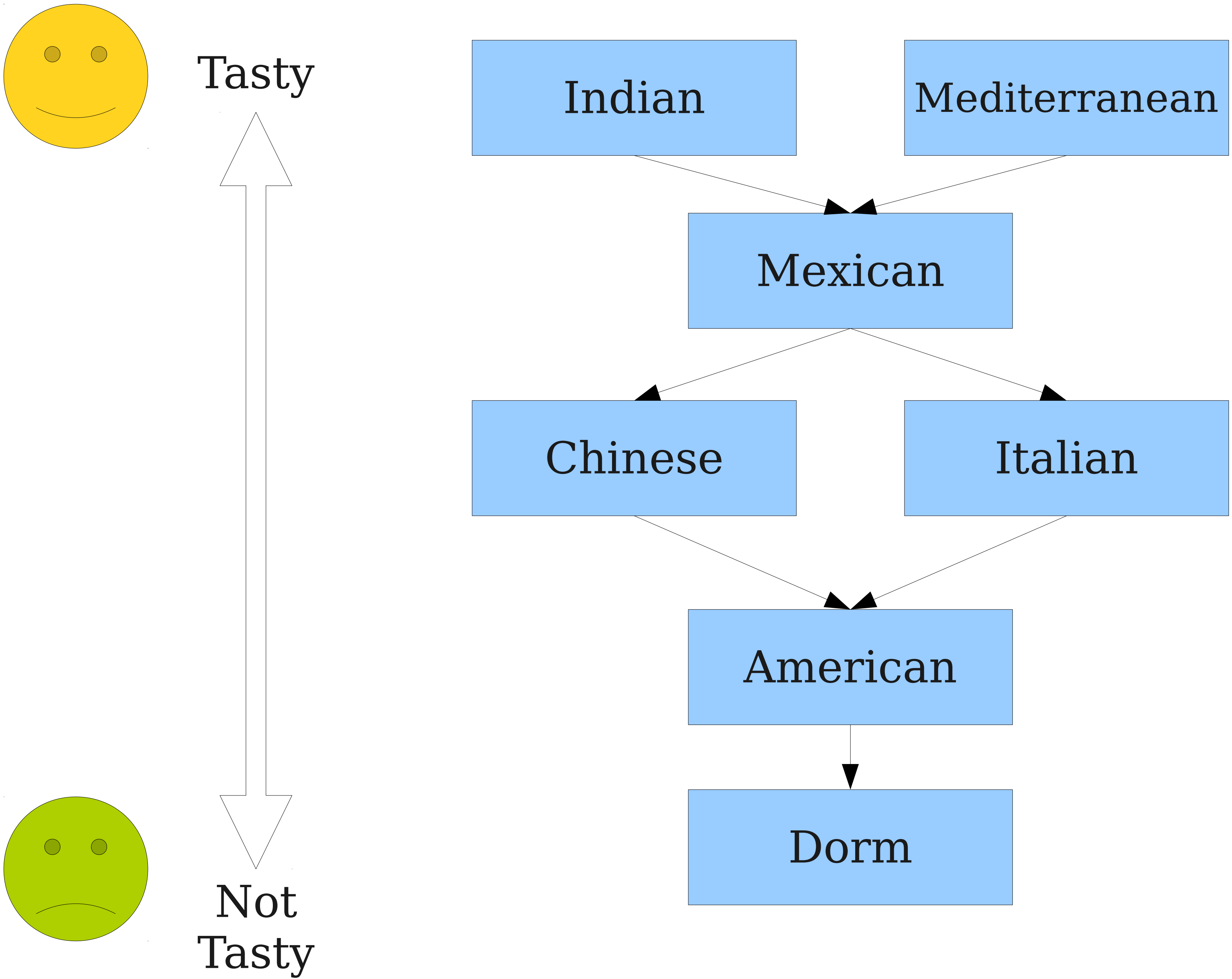
Police Shut Down, Down

Blow Speakers Up

See the Sunlight

A **directed acyclic graph** (DAG) is a directed graph with no cycles.

Examples of DAGs



Tasty

Indian

Mediterranean

Mexican

Chinese

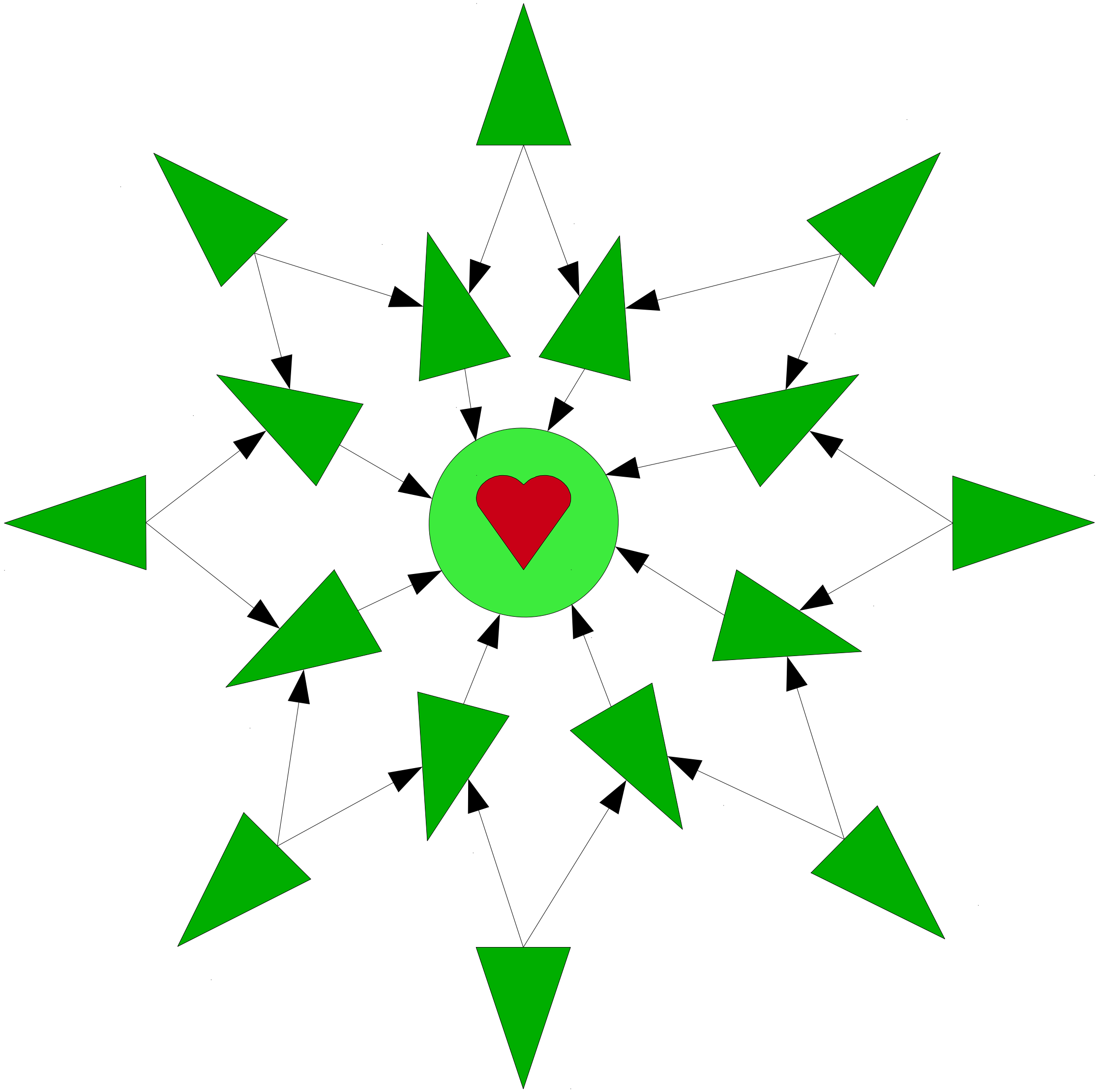
Italian

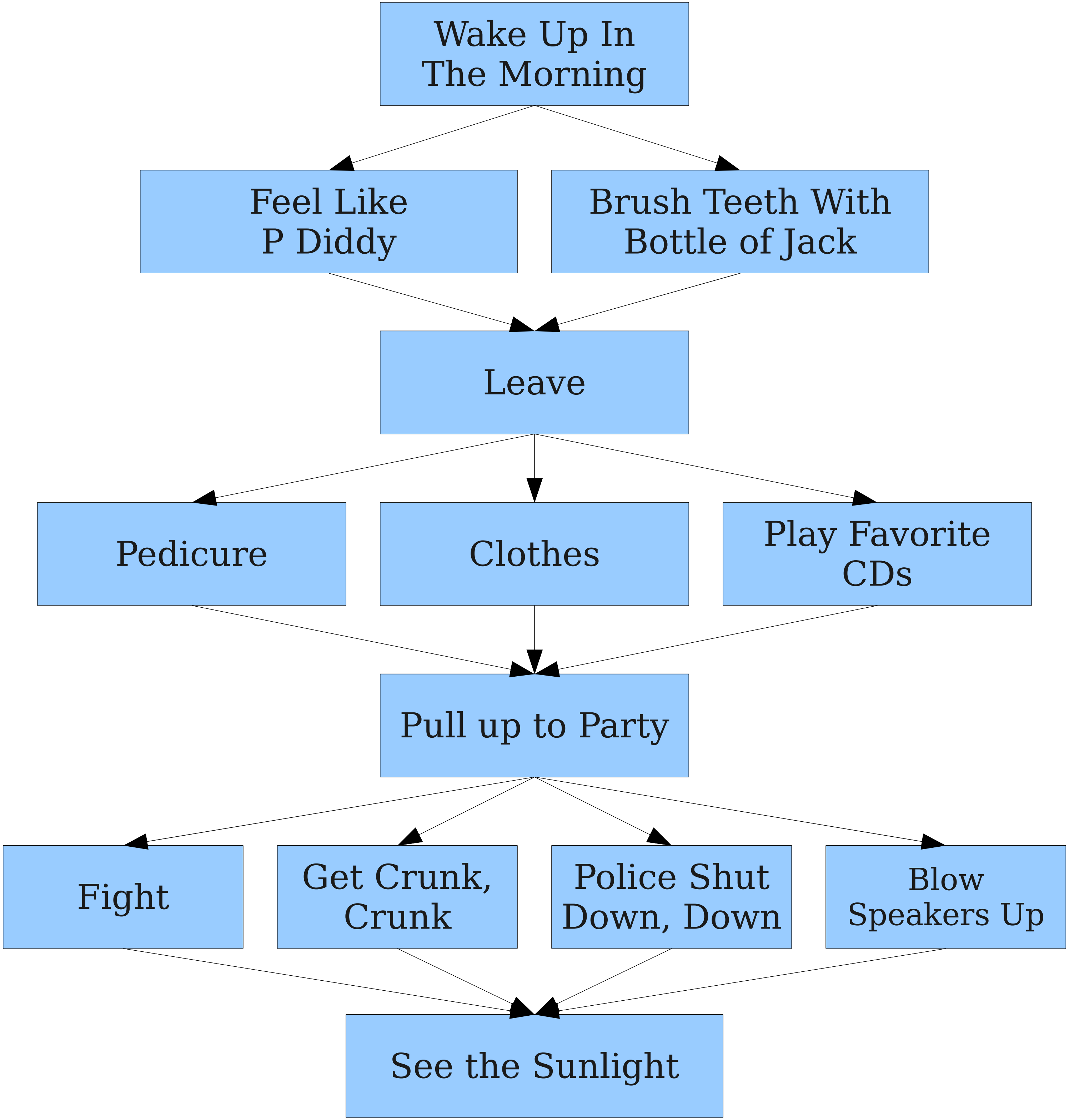
American

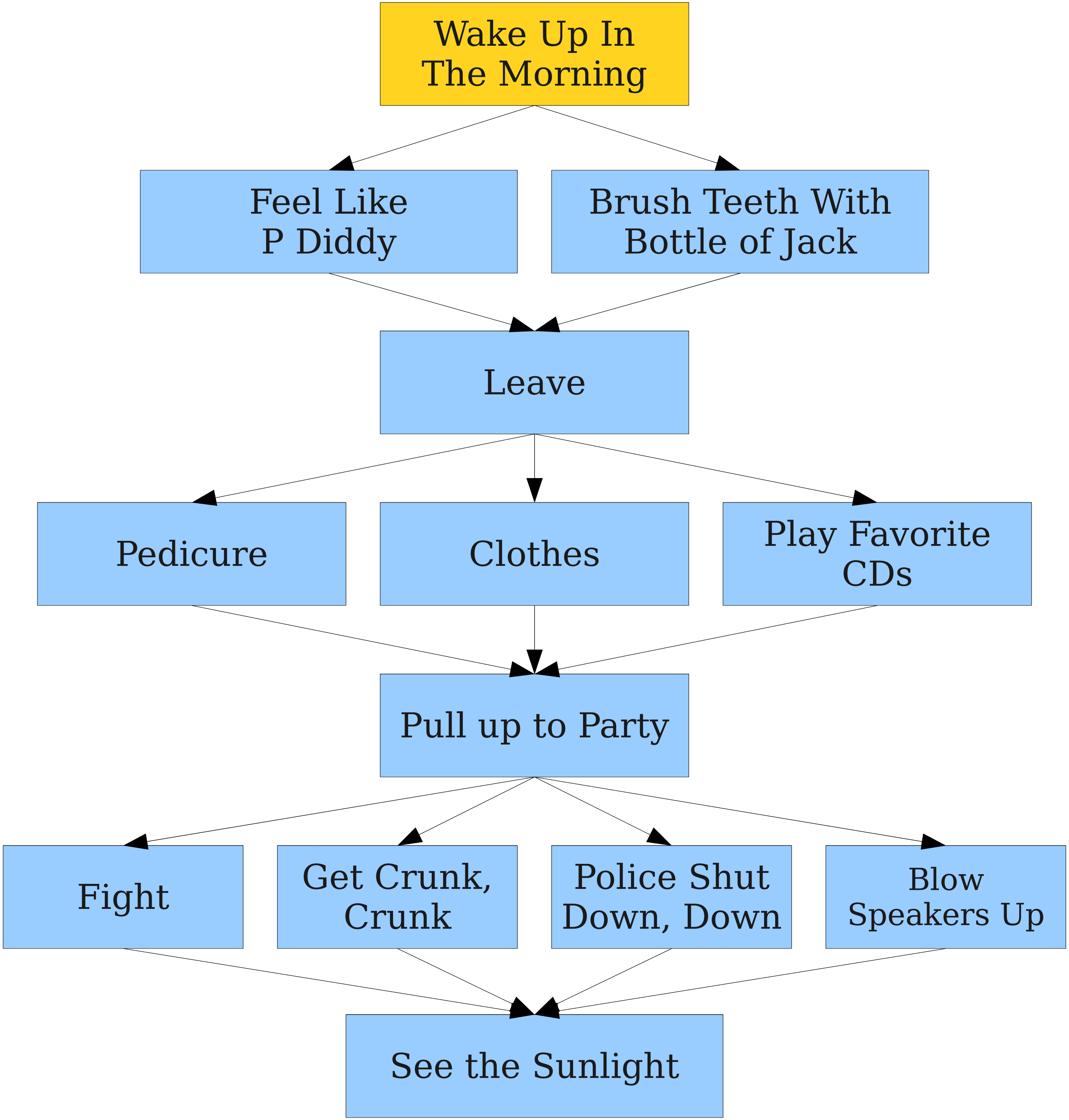
Dorm

Not Tasty

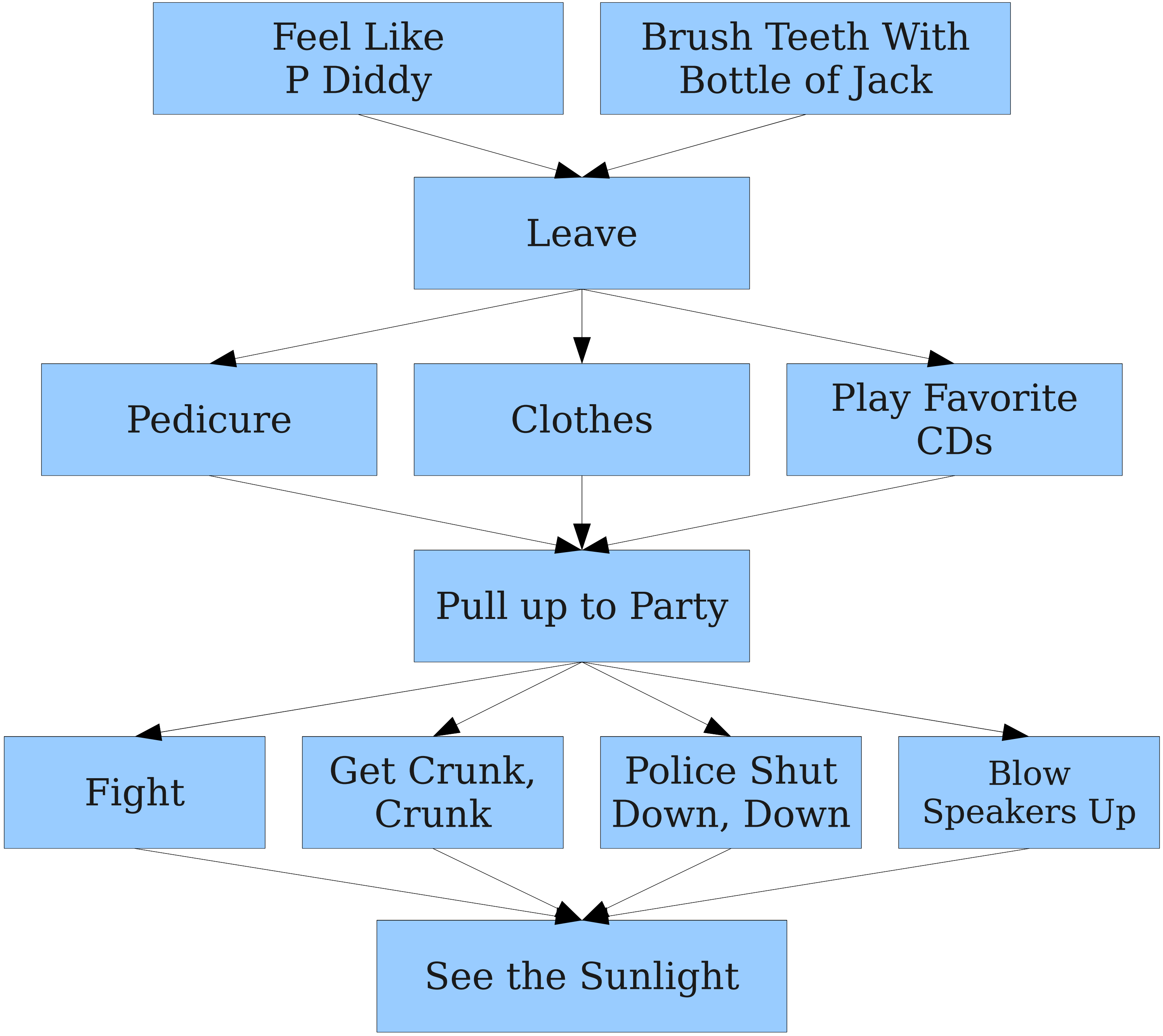
Examples of DAGs




Wake Up In The Morning
Feel Like P Diddy
Brush Teeth With Bottle of Jack
Leave
Pedicure
Clothes
Play Favorite CDs
Pull up to Party
Fight
Get Crunk, Crunk
Police Shut Down, Down
Blow Speakers Up
See the Sunlight


Wake Up In The Morning
Feel Like P Diddy
Brush Teeth With Bottle of Jack
Leave
Pedicure
Clothes
Play Favorite CDs
Pull up to Party
Fight
Get Crunk, Crunk
Police Shut Down, Down
Blow Speakers Up
See the Sunlight

Wake Up In
The Morning



Feel Like
P Diddy

Brush Teeth With
Bottle of Jack

Leave

Pedicure

Clothes

Play Favorite
CDs

Pull up to Party

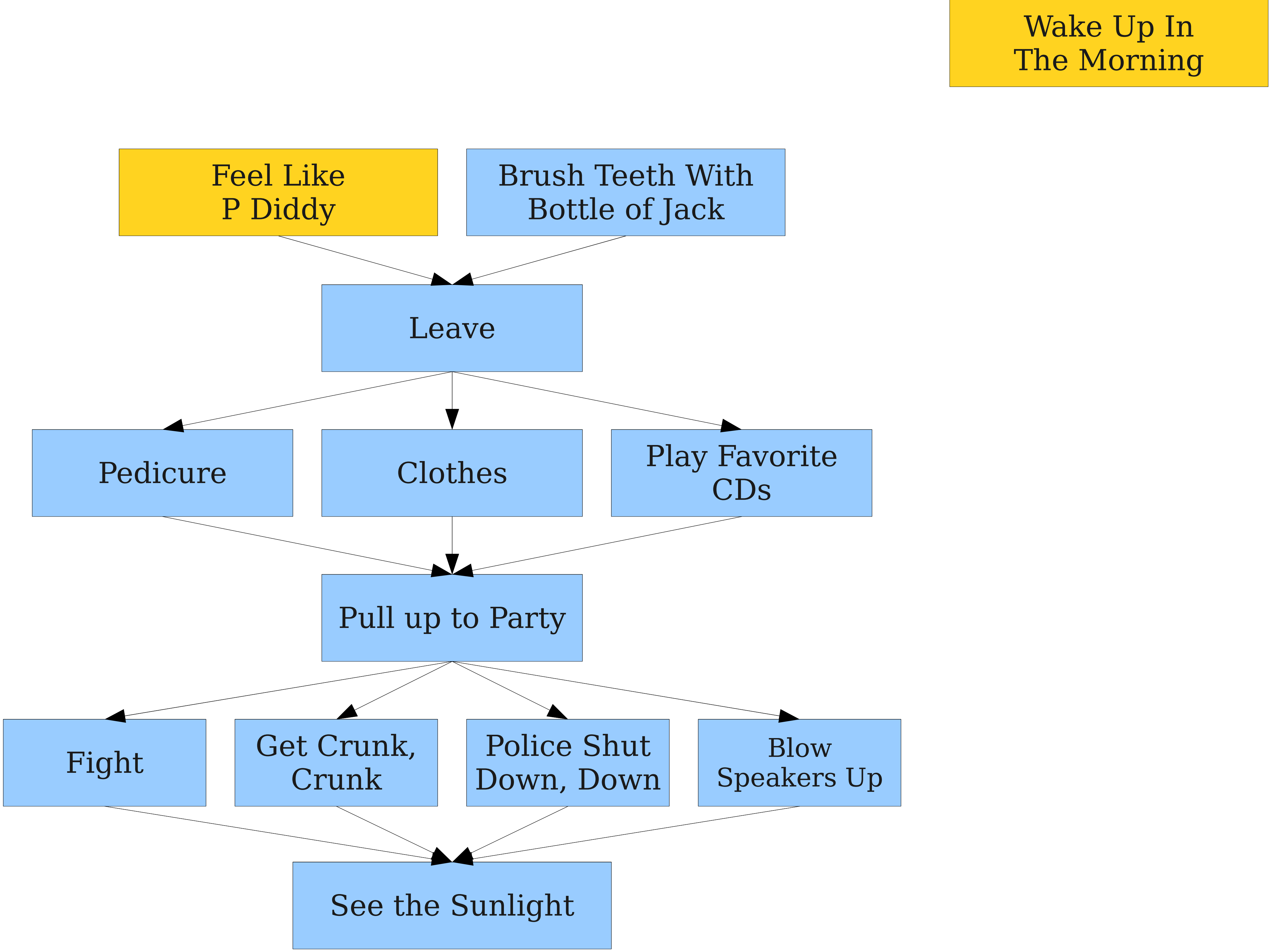
Fight

Get Crunk,
Crunk

Police Shut
Down, Down

Blow
Speakers Up

See the Sunlight



Wake Up In The Morning

Feel Like P Diddy

Brush Teeth With Bottle of Jack

Leave

Pedicure

Clothes

Play Favorite CDs

Pull up to Party

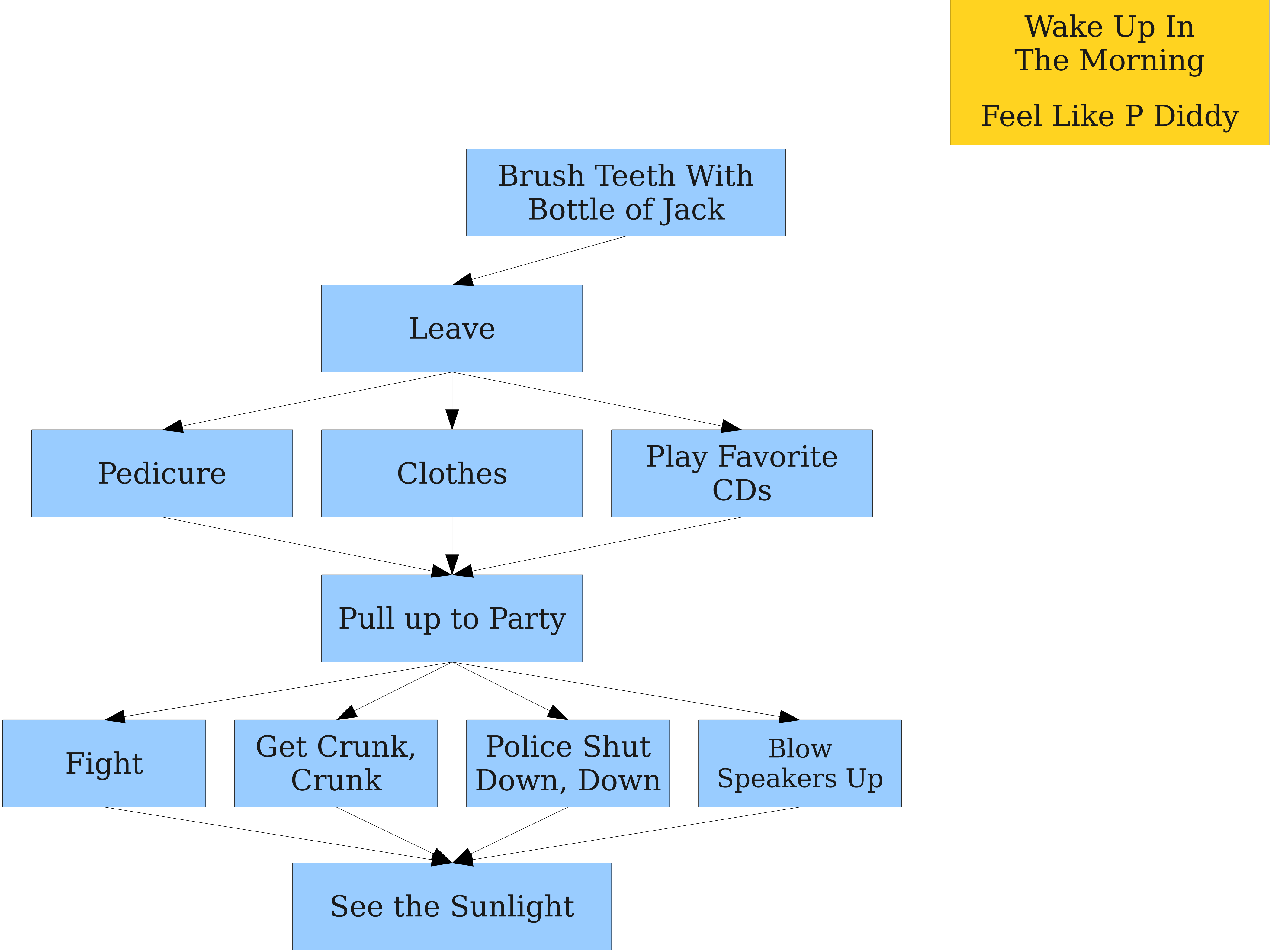
Fight

Get Crunk, Crunk

Police Shut Down, Down

Blow Speakers Up

See the Sunlight


Wake Up In The Morning
Feel Like P Diddy
Brush Teeth With Bottle of Jack
Leave
Pedicure
Clothes
Play Favorite CDs
Pull up to Party
Fight
Get Crunk, Crunk
Police Shut Down, Down
Blow Speakers Up
See the Sunlight

Wake Up In
The Morning

Feel Like P Diddy

Brush Teeth With
Bottle of Jack

Leave

Clothes

Play Favorite CDs

Pedicure

Pull up to Party

Fight

Blow Speakers Up

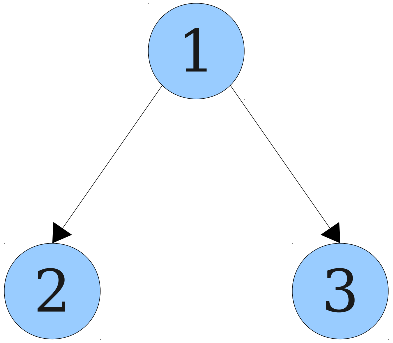
Police Shut
Down, Down

Get Crunk, Crunk

See the Sunlight

Topological Sort

- A **topological ordering** of the nodes of a DAG is one where no node is listed before its predecessors.
- Algorithm:
  - Find a node with no incoming edges.
  - Remove it from the graph.
  - Add it to the resulting ordering.
- There may be many valid orderings:



1 2 3

1 3 2

Theorem: A graph has a topological ordering iff it is a DAG.

Relations

Relations

- A **binary relation** is a property that describes whether two objects are related in some way.
- Examples:
  - Less-than: $x < y$
  - Divisibility: x divides y evenly
  - Friendship: x is a friend of y
  - Tastiness: x is tastier than y
- Given binary relation R, we write aRb iff a is **related** to b.
  - $a = b$
  - $a < b$
  - a "is tastier than" b
  - $a \equiv_k b$

Relations as Sets

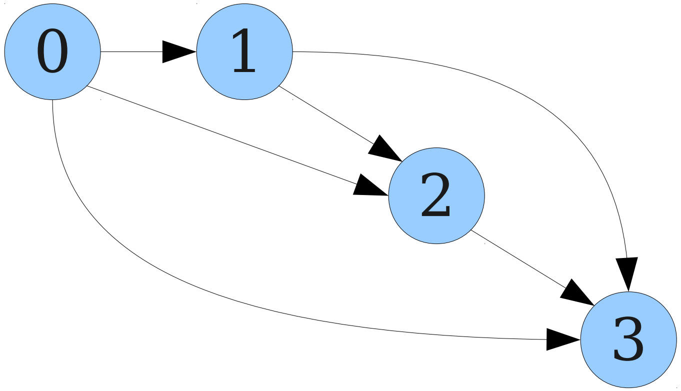
- Formally, a relation is a set of ordered pairs representing the pairs for which the relation is true.
  - Equality: { (0, 0), (1, 1), (2, 2), … }
  - Less-than: { (0, 1), (0, 2), …, (1, 2), (1, 3), … }
- Formally, we have that

$$\boldsymbol{aRb \equiv (a, b) \in R}$$

- The binary relations we'll discuss today will be binary relations over a set A.
  - Each relation is a subset of A^2.

Binary Relations and Graphs

- Each (directed) graph defines a binary relation:
  - aRb iff (a, b) is an edge.
- Each binary relation defines a graph:
  - (a, b) is an edge iff aRb.
- Example: Less-than



An Important Question

- Why study binary relations and graphs separately?
- **Simplicity**:
  - Certain operations feel more "natural" on binary relations than on graphs and vice-versa.
  - Converting a relation to a graph might result in an overly complex graph (or vice-versa).
- **Terminology**:
  - Vocabulary for graphs often different from that for relations.

Equivalence Relations

“x and y have the same color”

“x and y have the same shape”

“$x = y$”

“x and y have the same area”

“x and y are programs that produce the same output”

Informally

An **equivalence relation** is a relation that indicates when objects have some trait in common.

Do <u>not</u> use this definition in proofs!
It's just an intuition!

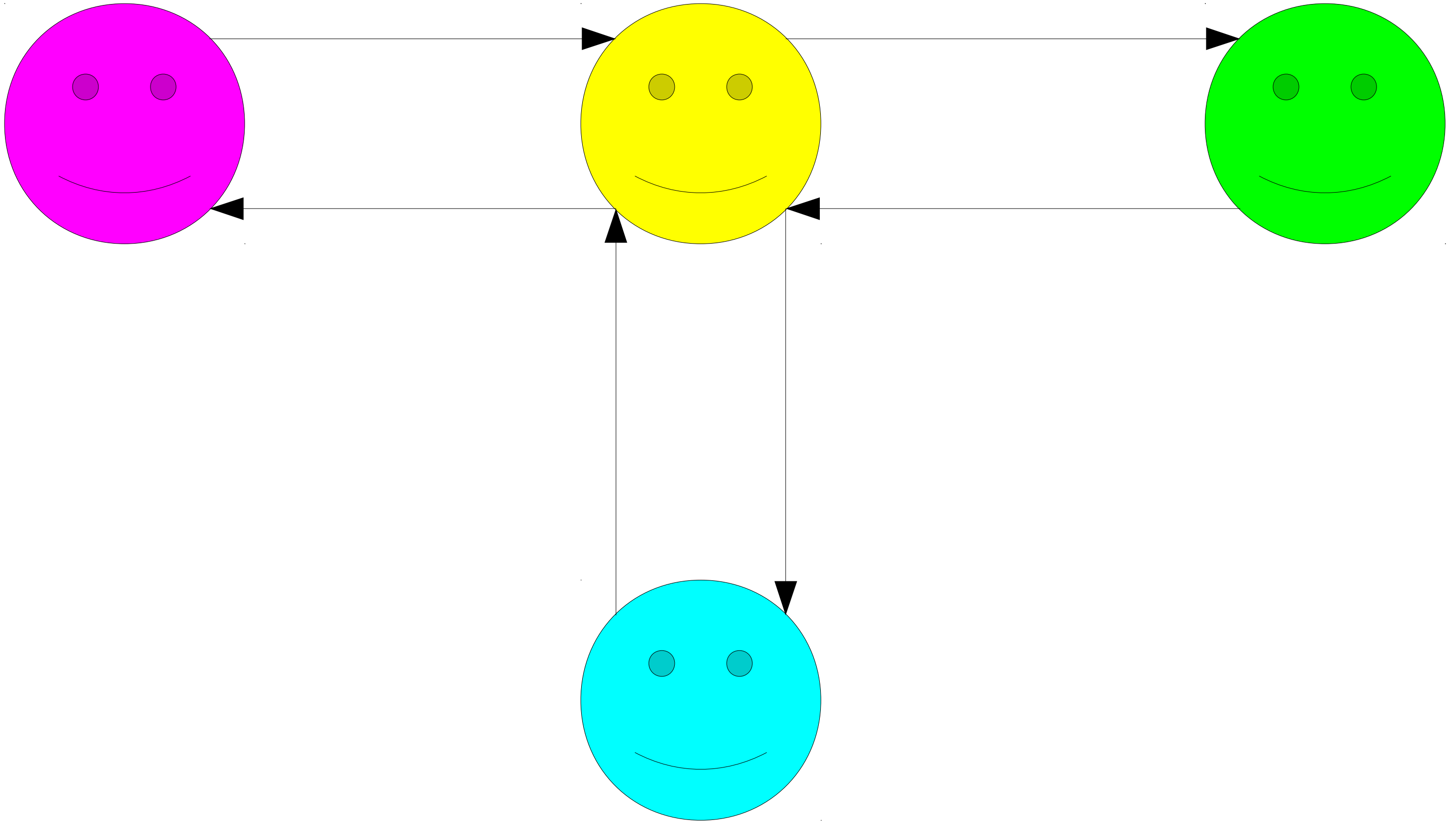
Symmetry

A binary relation R over a set A
is called **symmetric** iff

For any $x \in A$ and $y \in A$, if $x\mathrm{R}y$, then $y\mathrm{R}x$.

This definition (and others like it) can be used
in formal proofs.

An Intuition for Symmetry



For any $x \in A$ and $y \in A$,
if xRy, then yRx.

Reflexivity

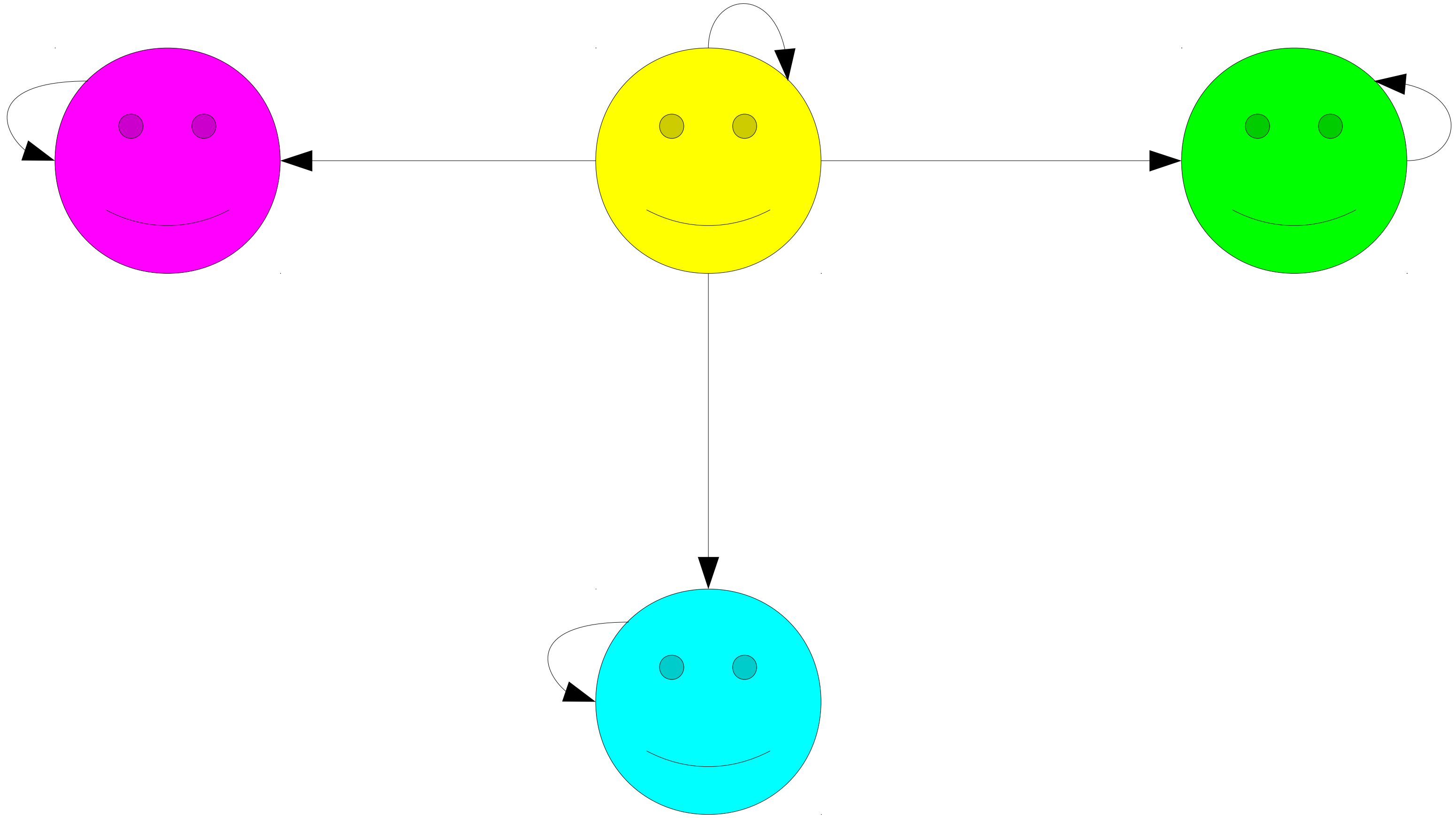
A binary relation R over a set A is called **reflexive** iff

For any $x \in A$, we have $x\mathrm{R}x$.

Some Reflexive Relations

- Equality:
  - For any x, we have $x = x$.
- Not greater than:
  - For any integer x, we have $x \leq x$.
- Subset:
  - For any set S, we have $S \subseteq S$.

An Intuition for Reflexivity



For any $x \in A$,
xRx

Transitivity

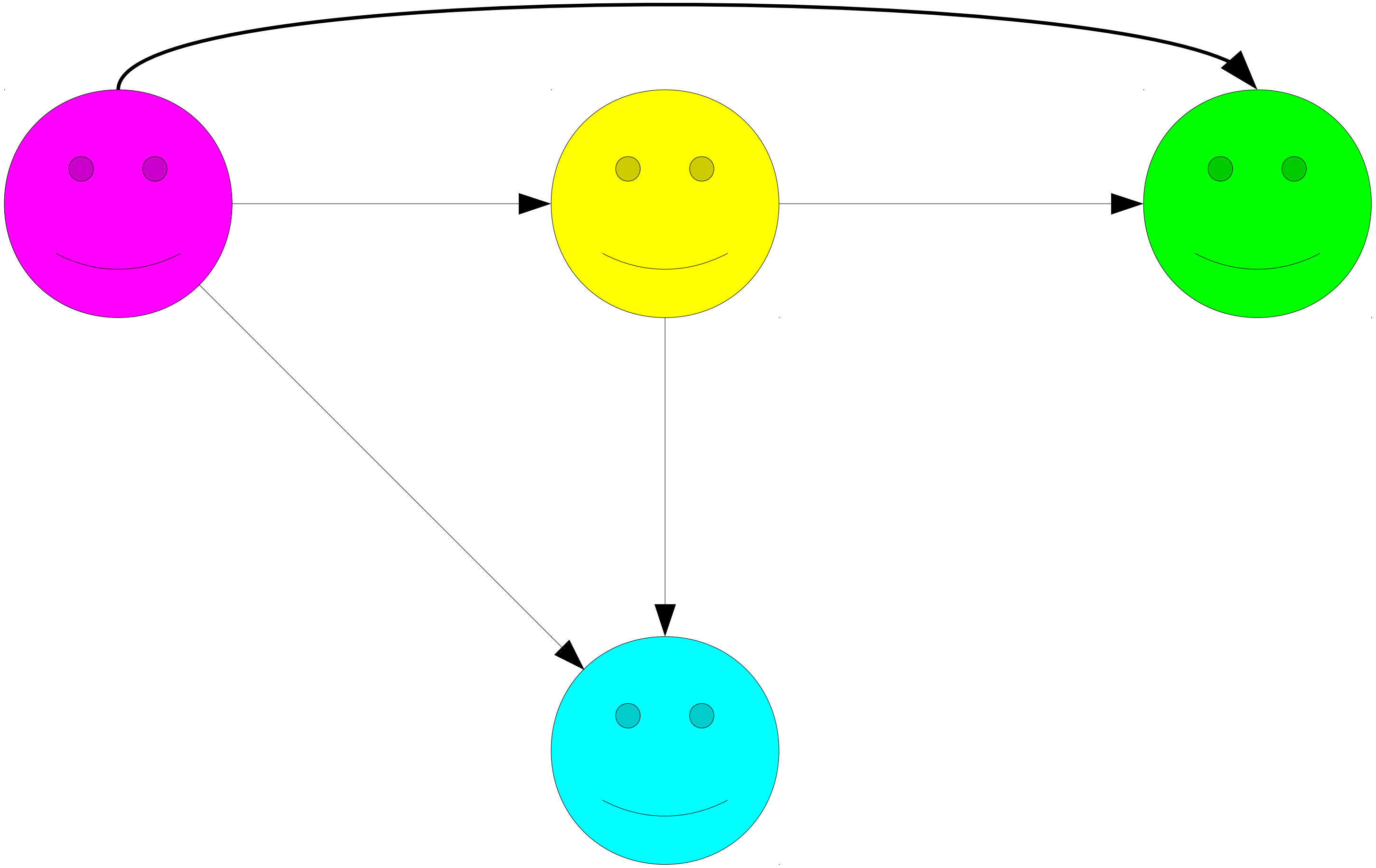
A binary relation R over a set A
is called **transitive** iff

For any x, y, $z \in A$,
if xRy and yRz,
then xRz.

Some Transitive Relations

- Equality:
  - $x = y$ and $y = z$ implies $x = z$.
- Less-than:
  - $x < y$ and $y < z$ implies $x < z$.
- Subset:
  - $S \subseteq T$ and $T \subseteq U$ implies $S \subseteq U$.

An Intuition for Transitivity



For any x, y, $z \in A$,
if xRy and yRz,
then xRz.

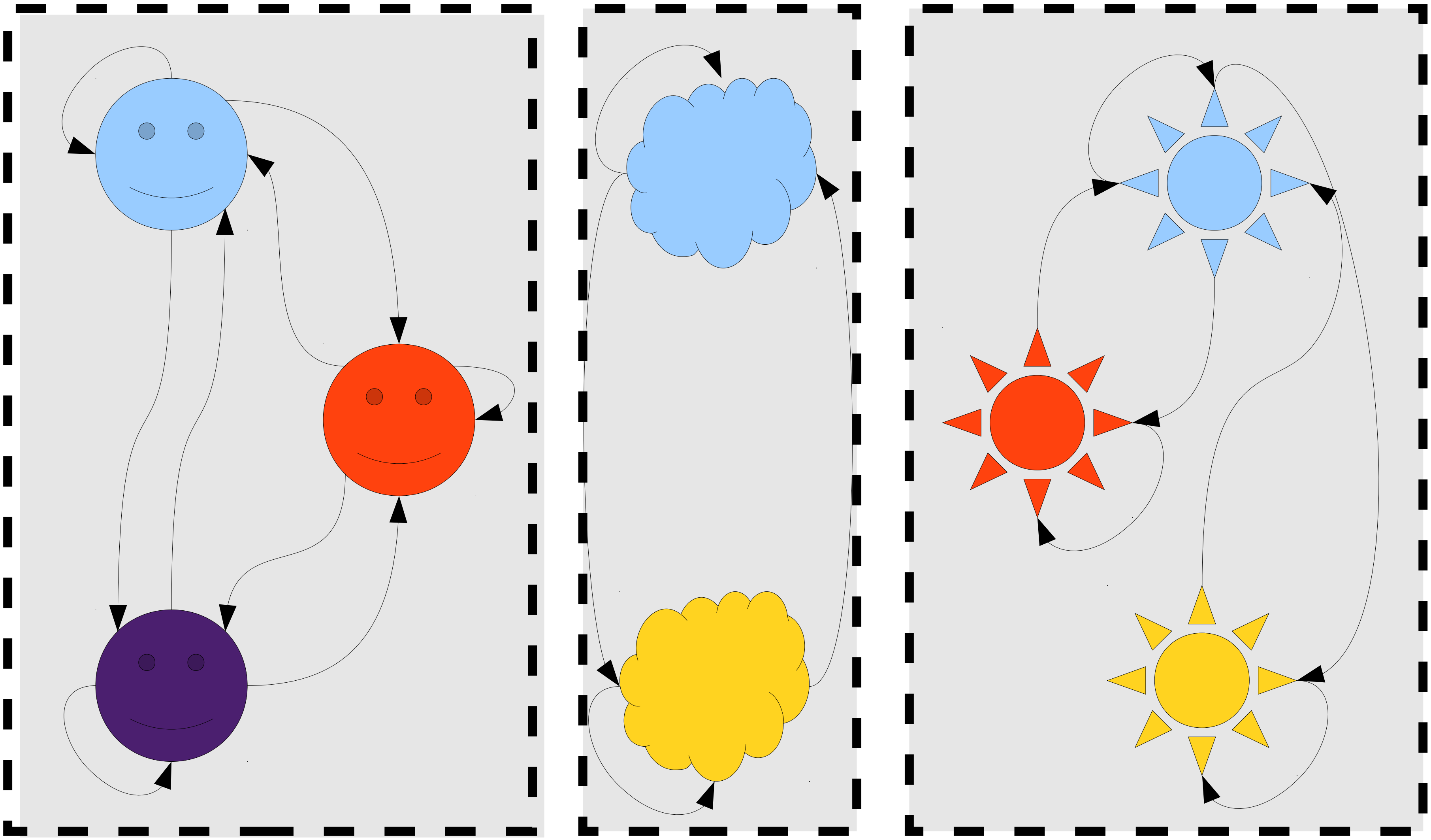
Equivalence Relations

A binary relation R over a set A is called an **equivalence relation** if it is

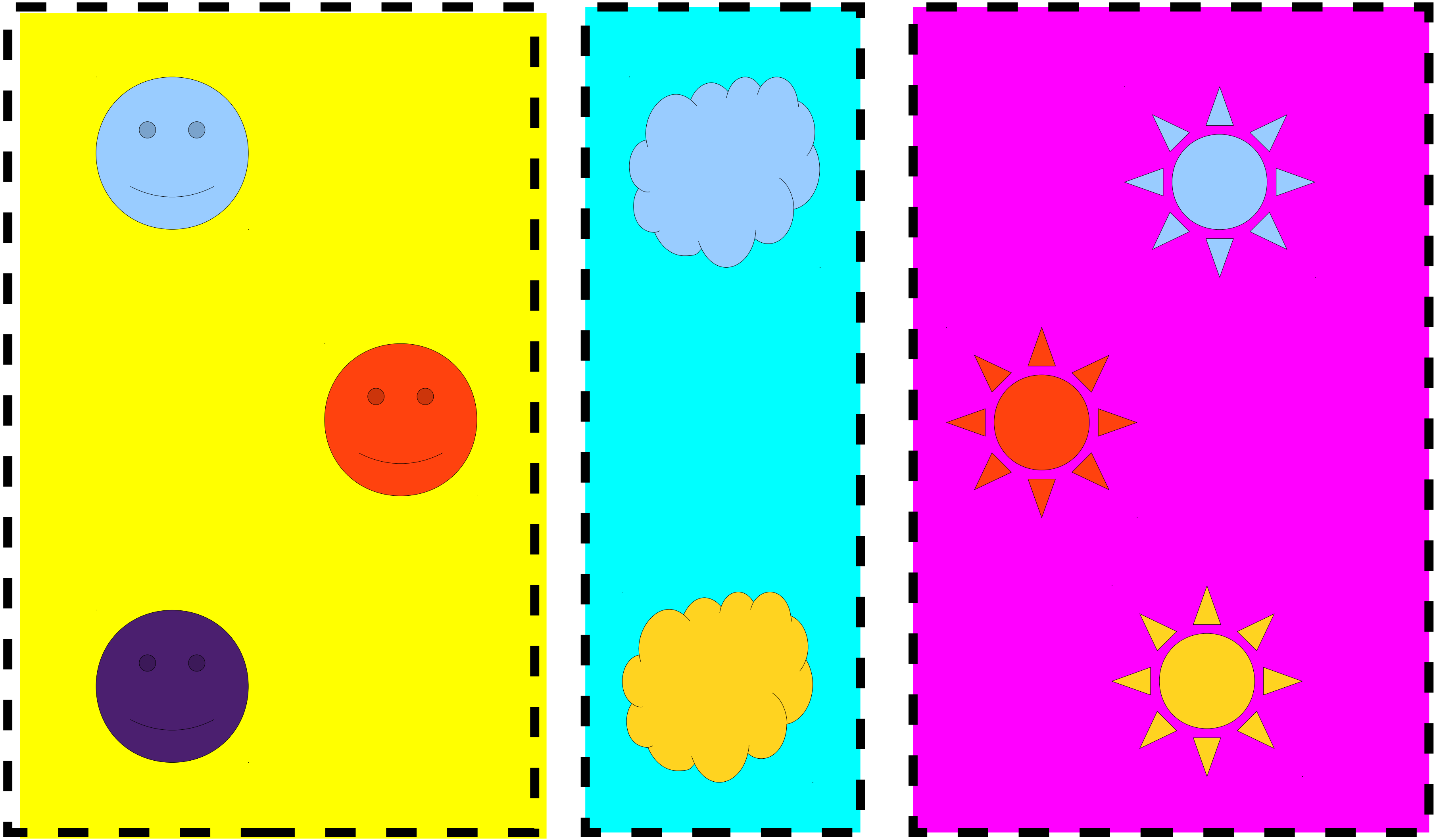
- **reflexive**,
- **symmetric**, and
- **transitive**.

Sample Equivalence Relations

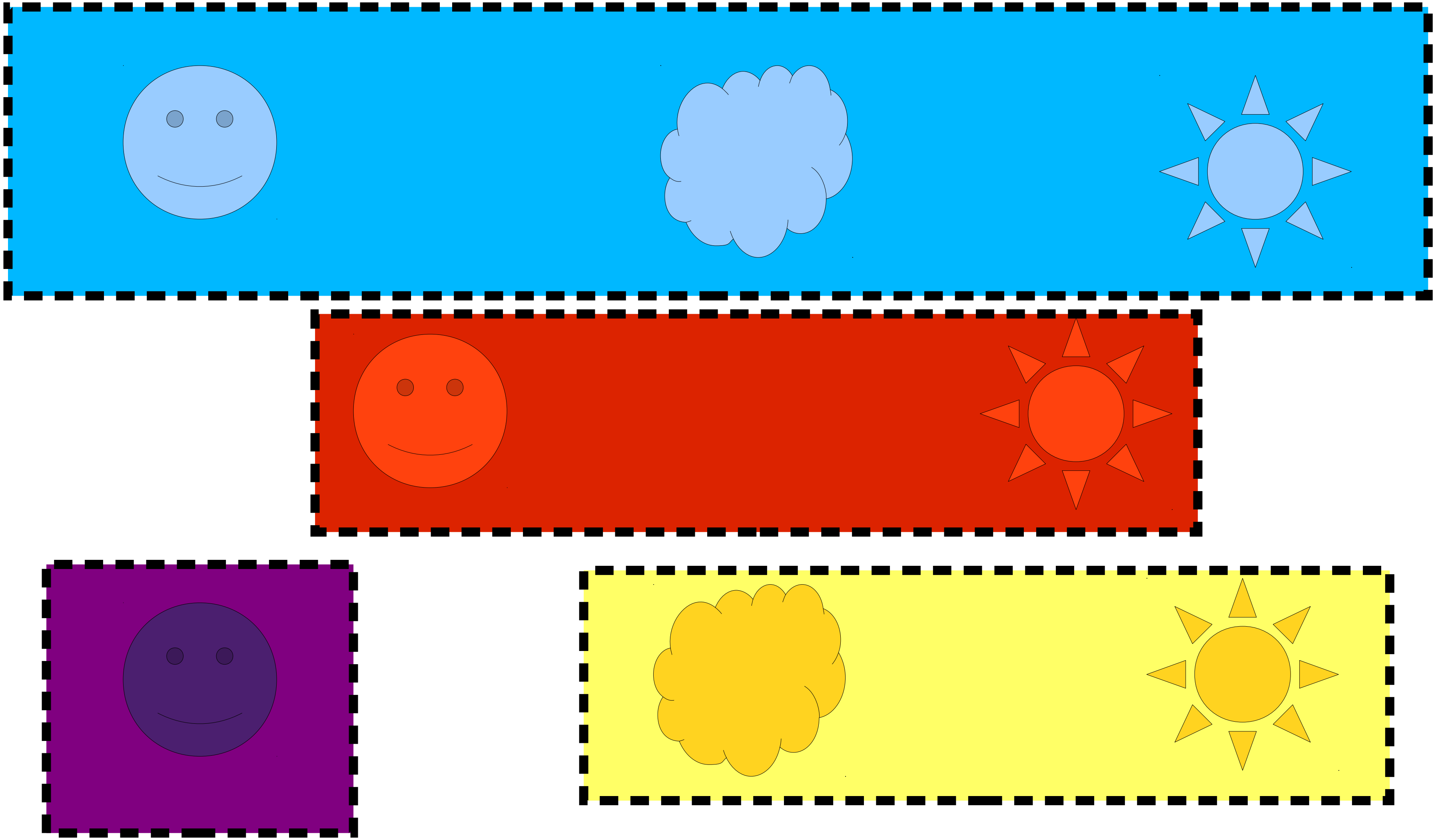
- Equality: $x = y$.
- For any graph G, the relation $x \leftrightarrow y$ meaning "x and y are mutually reachable."
- For any integer k, the relation $x \equiv_k y$ of modular congruence.



$xRy \equiv x$ and y have the same shape.



$xRy \equiv x$ and y have the same shape.



$xRy \equiv x$ and y have the same **color**.

Equivalence Classes

- Given an equivalence relation *R* over a set *A*, for any $a \in A$, the **equivalence class of *a*** is the set

$$[a]_R \equiv \{ x \mid x \in A \text{ and } aRx \}$$

- Informally, the set of all elements equal to *a*.
- *R* **partitions** the set *A* into a set of equivalence classes.

Theorem: Let R be an equivalence relation over a set A. Then every element of A belongs to exactly one equivalence class.

Theorem: Let R be an equivalence relation over a set A. Then every element of A belongs to exactly one equivalence class.

How do we prove this?

Existence and Uniqueness

- The proof we are attempting is a type of proof called an **existence and uniqueness** proof.
- We need to show that for any $a \in A$, there **exists** an equivalence class containing a and that this equivalence class is **unique**.
- These are two completely separate steps.

Proving Existence

- To prove **existence**, we need to show that for any $a \in A$, that a belongs to at least one equivalence class.
- This is just a proof of an existential statement.
- Can we find an equivalence class containing a?

Theorem: Let R be an equivalence relation over a set A. Then every element of A belongs to exactly one equivalence class.

Proof: We will show that every $a \in A$ belongs to at least one equivalence class and to at most one equivalence class.

To see that every $a \in A$ belongs to at least one equivalence class, consider any $a \in A$ and the equivalence class $[a]_R = \{ x \mid x \in A \text{ and } aRx \}$. Since R is an equivalence relation, R is reflexive, so aRa. Thus $a \in [a]_R$. Since our choice of a was arbitrary, this means every $a \in A$ belongs to at least one equivalence class – namely, $[a]_R$.

To see that every $a \in A$ belongs to at most one equivalence class, **???**

How do we prove this?

Proving Uniqueness

- To prove that there is a **unique** object with some property, we can do the following:
  - Consider any two arbitrary objects x and y with that property.
  - Show that $x = y$.
  - Conclude, therefore, that there is only one object with that property, and we just gave it two different names.

Theorem: Let R be an equivalence relation over a set A. Then every element of A belongs to exactly one equivalence class.

Proof: We will show that every $a \in A$ belongs to at least one equivalence class and to at most one equivalence class.

To see that every $a \in A$ belongs to at least one equivalence class, consider any $a \in A$ and the equivalence class $[a]_R = \{ x \mid x \in A \text{ and } aRx \}$. Since R is an equivalence relation, R is reflexive, so aRa. Thus $a \in [a]_R$. Since our choice of a was arbitrary, this means every $a \in A$ belongs to at least one equivalence class – namely, $[a]_R$.

To see that every $a \in A$ belongs to at most one equivalence class, we show that for any $a \in A$, if $a \in [x]_R$ and $a \in [y]_R$, then $[x]_R = [y]_R$. To do this, we prove that if $a \in [x]_R$ and $a \in [y]_R$, then $[x]_R \subseteq [y]_R$. By swapping $[x]_R$ and $[y]_R$, we can conclude that $[y]_R \subseteq [x]_R$, meaning that $[x]_R = [y]_R$.

Assume that $a \in [x]_R$ and $a \in [y]_R$. Consider any $t \in [x]_R$. We will show that $t \in [y]_R$. Since $t \in [x]_R$, we know xRt. Since $a \in [x]_R$, we have xRa. Since R is an equivalence relation, R is symmetric and transitive. By symmetry, from xRa we have aRx. By transitivity, from aRx and xRt we have aRt. Since $a \in [y]_R$, we have yRa. By transitivity, from yRa and aRt we have yRt. Thus, $t \in [y]_R$. Since our choice of t was arbitrary, we have $[x]_R \subseteq [y]_R$. Therefore, by our earlier reasoning, $[x]_R = [y]_R$. ■

Theorem: Let R be an equivalence relation over a set A. Then every element of A belongs to exactly one equivalence class.

Proof: We will show that every $a \in A$ belongs to at least one equivalence class and to at most one equivalence class.

To see that every $a \in A$ belongs to at least one equivalence class, consider any $a \in A$ and the equivalence class $[a]_R = \{ x \mid x \in A \text{ and } aRx \}$. Since R is an equivalence relation, R is reflexive, so aRa. Thus $a \in [a]_R$. Since our choice of a was arbitrary, this means every $a \in A$ belongs to at le

To ve
sh do
th pping
[x $[y]_R$.

This proof helps to justify our definition of equivalence relations. We need all three of the properties we've listed in order for this proof to work, and we don't need any others.

Assume that $a \in [x]_R$ and $a \in [y]_R$. Consider any $t \in [x]_R$. We will show that $t \in [y]_R$. Since $t \in [x]_R$, we know xRt. Since $a \in [x]_R$, we have xRa. Since R is an equivalence relation, R is symmetric and transitive. By symmetry, from xRa we have aRx. By transitivity, from aRx and xRt we have aRt. Since $a \in [y]_R$, we have yRa. By transitivity, from yRa and aRt we have yRt. Thus, $t \in [y]_R$. Since our choice of t was arbitrary, we have $[x]_R \subseteq [y]_R$. Therefore, by our earlier reasoning, $[x]_R = [y]_R$. ■