# Listing Bit Strings

List all bit strings of length 3.

# Listing Bit Strings

List all bit strings of length 3.

000, 001, 010, 011, 100, 101, 110, 111.

# Listing Bit Strings

List all bit strings of length 3.

000, 001, 010, 011, 100, 101, 110, 111.

Now do it while only flipping one bit at a time!

# Listing Bit Strings

List all bit strings of length 3.

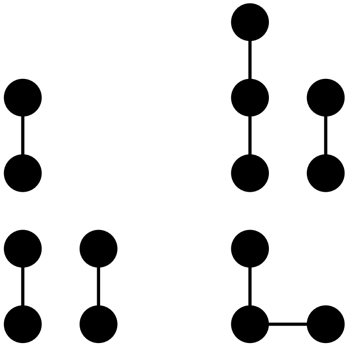
000, 001, 010, 011, 100, 101, 110, 111.

Now do it while only flipping one bit at a time!

Today: Finish graphs and talk about numbers.

# Forests

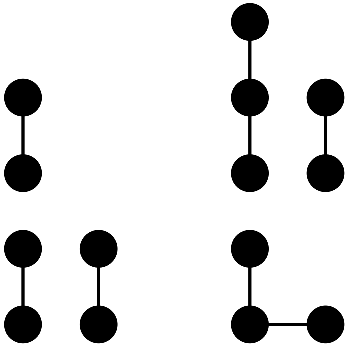
A **forest** is an acyclic graph.



Each connected component of a forest is a tree.

# Forests

A **forest** is an acyclic graph.

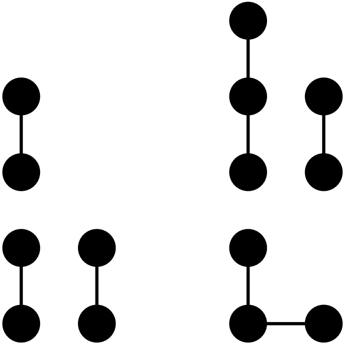


Each connected component of a forest is a tree.

How many connected components in this graph?

# Forests

A **forest** is an acyclic graph.

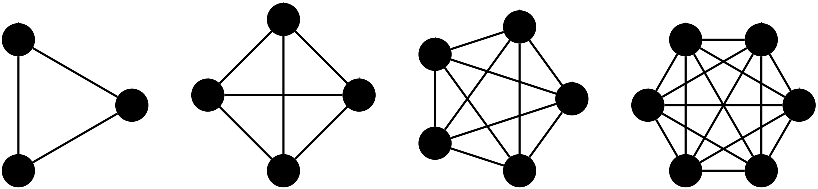


Each connected component of a forest is a tree.

How many connected components in this graph? 6.

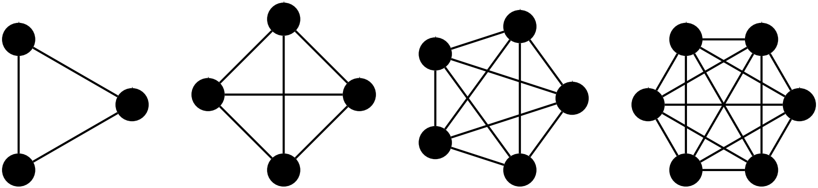
# Complete Graphs

The **complete graph** $K_n$ has $n$ vertices and *all* possible edges.



# Complete Graphs

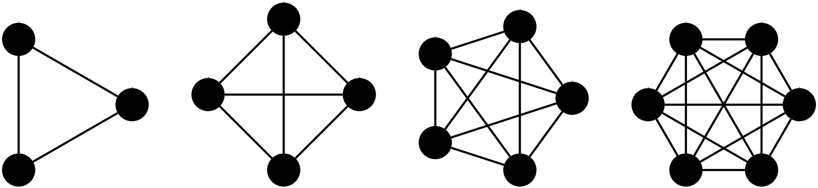
The **complete graph** $K_n$ has $n$ vertices and *all* possible edges.



A **bipartite graph** has **left nodes** $L$ and **right nodes** $R$.

# Complete Graphs

The **complete graph** $K_n$ has $n$ vertices and *all* possible edges.

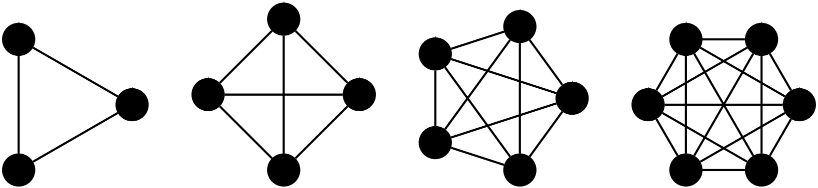


A **bipartite graph** has **left nodes** $L$ and **right nodes** $R$.

- The vertex set is $V = L \cup R$.

# Complete Graphs

The **complete graph** $K_n$ has $n$ vertices and *all* possible edges.

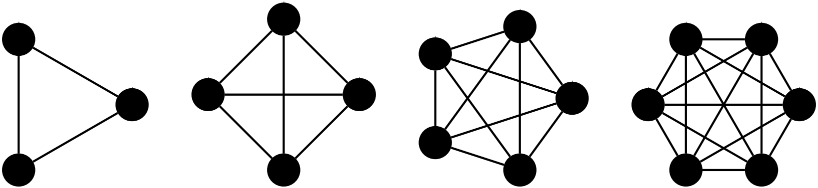


A **bipartite graph** has **left nodes** $L$ and **right nodes** $R$.

- The vertex set is $V = L \cup R$.
- Left nodes are only allowed to connect to right nodes; right nodes are only allowed to connect to left nodes.

# Complete Graphs

The **complete graph** $K_n$ has $n$ vertices and *all* possible edges.



A **bipartite graph** has **left nodes** $L$ and **right nodes** $R$.

- The vertex set is $V = L \cup R$.
- Left nodes are only allowed to connect to right nodes; right nodes are only allowed to connect to left nodes.

The **complete bipartite graph** $K_{m,n}$ has $m$ left nodes, $n$ right nodes, and *all* possible edges.

# Edge Sparsity

How many edges does $K_n$ have?

# Edge Sparsity

How many edges does $K_n$ have?

- Handshaking Lemma: $\sum_{v \in V} \deg v = 2|E|$.

# Edge Sparsity

How many edges does $K_n$ have?

- Handshaking Lemma: $\sum_{v \in V} \deg v = 2|E|$.
- $\sum_{v \in V} \deg v = n(n-1)$.

# Edge Sparsity

How many edges does $K_n$ have?

- Handshaking Lemma: $\sum_{v \in V} \deg v = 2|E|$.
- $\sum_{v \in V} \deg v = n(n-1)$.
- So $|E| = n(n-1)/2$.

# Edge Sparsity

How many edges does $K_n$ have?

- Handshaking Lemma: $\sum_{v \in V} \deg v = 2|E|$.
- $\sum_{v \in V} \deg v = n(n-1)$.
- So $|E| = n(n-1)/2$.

Asymptotic notation from CS 61A/B: $|E| = \Theta(n^2)$.

# Edge Sparsity

How many edges does $K_n$ have?

- Handshaking Lemma: $\sum_{v \in V} \deg v = 2|E|$.
- $\sum_{v \in V} \deg v = n(n-1)$.
- So $|E| = n(n-1)/2$.

Asymptotic notation from CS 61A/B: $|E| = \Theta(n^2)$.

For a tree on $n$ vertices, $|E| = n - 1 = \Theta(n)$.

# Edge Sparsity

How many edges does $K_n$ have?

- Handshaking Lemma: $\sum_{v \in V} \deg v = 2|E|$.
- $\sum_{v \in V} \deg v = n(n-1)$.
- So $|E| = n(n-1)/2$.

Asymptotic notation from CS 61A/B: $|E| = \Theta(n^2)$.

For a tree on $n$ vertices, $|E| = n - 1 = \Theta(n)$.

The complete graph is called *dense*; trees are called *sparse*.

# Planar Graphs Are Sparse

**Theorem**: For a connected planar graph with $|V| \geq 3$, we have $e \leq 3v - 6$.

# Planar Graphs Are Sparse

**Theorem**: For a connected planar graph with $|V| \geq 3$, we have $e \leq 3v - 6$.

*Proof*.

# Planar Graphs Are Sparse

**Theorem**: For a connected planar graph with $|V| \geq 3$, we have $e \leq 3v - 6$.

*Proof*.

- Each edge has two "sides".

# Planar Graphs Are Sparse

**Theorem**: For a connected planar graph with $|V| \geq 3$, we have $e \leq 3v - 6$.

*Proof*.

- Each edge has two "sides". So, if we add up all of the sides, we get $2e$.

# Planar Graphs Are Sparse

**Theorem**: For a connected planar graph with $|V| \geq 3$, we have $e \leq 3v - 6$.

*Proof.*

- Each edge has two "sides". So, if we add up all of the sides, we get $2e$.
- Each face has at least three sides.

# Planar Graphs Are Sparse

**Theorem**: For a connected planar graph with $|V| \geq 3$, we have $e \leq 3v - 6$.

*Proof*.

- Each edge has two “sides”. So, if we add up all of the sides, we get $2e$.
- Each face has at least three sides. So the total number of sides is at least $3f$.

# Planar Graphs Are Sparse

**Theorem**: For a connected planar graph with $|V| \geq 3$, we have $e \leq 3v - 6$.

*Proof.*

- Each edge has two "sides". So, if we add up all of the sides, we get $2e$.
- Each face has at least three sides. So the total number of sides is at least $3f$.
- Thus, $2e \geq 3f$.

# Planar Graphs Are Sparse

**Theorem**: For a connected planar graph with $|V| \geq 3$, we have $e \leq 3v - 6$.

*Proof.*

- Each edge has two “sides”. So, if we add up all of the sides, we get $2e$.
- Each face has at least three sides. So the total number of sides is at least $3f$.
- Thus, $2e \geq 3f$.
- Euler’s Formula: $v + f = e + 2$.

# Planar Graphs Are Sparse

**Theorem**: For a connected planar graph with $|V| \geq 3$, we have $e \leq 3v - 6$.

*Proof.*

- Each edge has two "sides". So, if we add up all of the sides, we get $2e$.
- Each face has at least three sides. So the total number of sides is at least $3f$.
- Thus, $2e \geq 3f$.
- Euler's Formula: $v + f = e + 2$.
- Rearrange: $e \leq 3v - 6$. □

# Planar Graphs Are Sparse

**Theorem**: For a connected planar graph with $|V| \geq 3$, we have $e \leq 3v - 6$.

*Proof.*

- Each edge has two "sides". So, if we add up all of the sides, we get $2e$.
- Each face has at least three sides. So the total number of sides is at least $3f$.
- Thus, $2e \geq 3f$.
- Euler's Formula: $v + f = e + 2$.
- Rearrange: $e \leq 3v - 6$. $\square$

If the graph has $n$ vertices, then $|E| = \Theta(n)$.

# Planar Graphs Are Sparse

**Theorem**: For a connected planar graph with $|V| \geq 3$, we have $e \leq 3v - 6$.

*Proof*.

- Each edge has two “sides”. So, if we add up all of the sides, we get $2e$.
- Each face has at least three sides. So the total number of sides is at least $3f$.
- Thus, $2e \geq 3f$.
- Euler’s Formula: $v + f = e + 2$.
- Rearrange: $e \leq 3v - 6$. $\square$

If the graph has $n$ vertices, then $|E| = \Theta(n)$. Like trees.

# Planar Graphs Are Sparse

**Theorem**: For a connected planar graph with $|V| \geq 3$, we have $e \leq 3v - 6$.

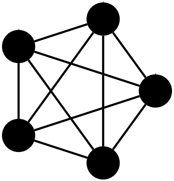
*Proof*.

- Each edge has two "sides". So, if we add up all of the sides, we get $2e$.
- Each face has at least three sides. So the total number of sides is at least $3f$.
- Thus, $2e \geq 3f$.
- Euler's Formula: $v + f = e + 2$.
- Rearrange: $e \leq 3v - 6$. $\square$

If the graph has $n$ vertices, then $|E| = \Theta(n)$. Like trees.

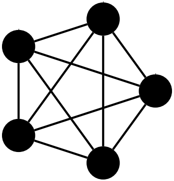
Planar graphs are sparse.

# $K_5$ Is Not Planar



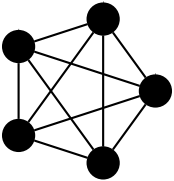
How many edges does $K_5$ have?

# $K_5$ Is Not Planar



How many edges does $K_5$ have? 10.

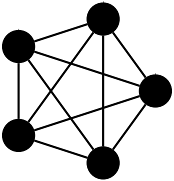
# $K_5$ Is Not Planar



How many edges does $K_5$ have? 10.

- $e = 10$.

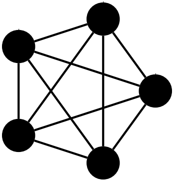
# $K_5$ Is Not Planar



How many edges does $K_5$ have? 10.

- $e = 10$.
- $3v - 6 = 9$.

# $K_5$ Is Not Planar

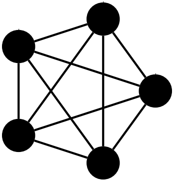


How many edges does $K_5$ have? 10.

- $e = 10$.
- $3v - 6 = 9$.

This violates $e \leq 3v - 6$ for planar graphs.

# $K_5$ Is Not Planar



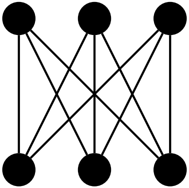
How many edges does $K_5$ have? 10.

- $e = 10$.
- $3v - 6 = 9$.

This violates $e \leq 3v - 6$ for planar graphs.

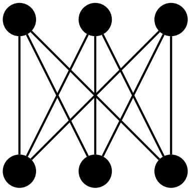
$K_5$ is not planar.

# $K_{3,3}$ Is Not Planar



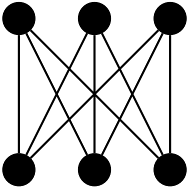
Consider $K_{3,3}$.

# $K_{3,3}$ Is Not Planar



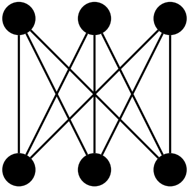
Consider $K_{3,3}$. Edges?

# $K_{3,3}$ Is Not Planar



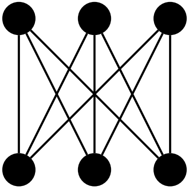
Consider $K_{3,3}$. Edges? 9.

# $K_{3,3}$ Is Not Planar



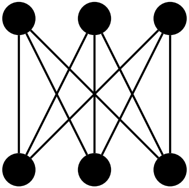
Consider $K_{3,3}$. Edges? 9. Vertices?

# $K_{3,3}$ Is Not Planar



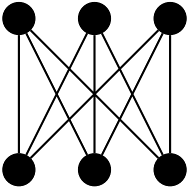
Consider $K_{3,3}$. Edges? 9. Vertices? 6.

# $K_{3,3}$ Is Not Planar



Consider $K_{3,3}$. Edges? 9. Vertices? 6. So $3v - 6 = 12$.

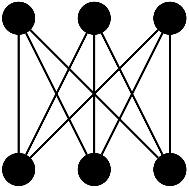
# $K_{3,3}$ Is Not Planar



Consider $K_{3,3}$. Edges? 9. Vertices? 6. So $3v-6=12$.

The previous proof fails.

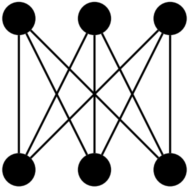
# $K_{3,3}$ Is Not Planar



Consider $K_{3,3}$. Edges? 9. Vertices? 6. So $3v - 6 = 12$.

The previous proof fails. Make it stronger!

# $K_{3,3}$ Is Not Planar

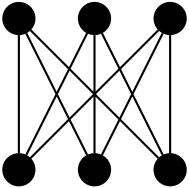


Consider $K_{3,3}$. Edges? 9. Vertices? 6. So $3v - 6 = 12$.

The previous proof fails. Make it stronger!

- The total number of sides is $2e$.

# $K_{3,3}$ Is Not Planar

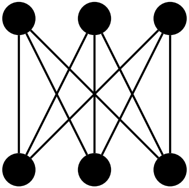


Consider $K_{3,3}$. Edges? 9. Vertices? 6. So $3v - 6 = 12$.

The previous proof fails. Make it stronger!

- The total number of sides is $2e$.
- Each face has at least three sides.

# $K_{3,3}$ Is Not Planar

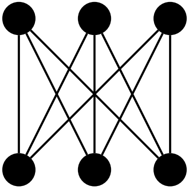


Consider $K_{3,3}$. Edges? 9. Vertices? 6. So $3v - 6 = 12$.

The previous proof fails. Make it stronger!

- The total number of sides is $2e$.
- Each face has at least three sides. Actually, at least four!

# $K_{3,3}$ Is Not Planar

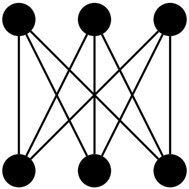


Consider $K_{3,3}$. Edges? 9. Vertices? 6. So $3v - 6 = 12$.

The previous proof fails. Make it stronger!

- The total number of sides is $2e$.
- Each face has at least three sides. Actually, at least four!
- In a bipartite graph, cycles are of even length.

# $K_{3,3}$ Is Not Planar

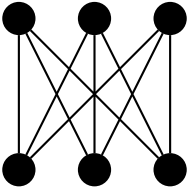


Consider $K_{3,3}$. Edges? 9. Vertices? 6. So $3v - 6 = 12$.

The previous proof fails. Make it stronger!

- The total number of sides is $2e$.
- Each face has at least three sides. Actually, at least four!
- In a bipartite graph, cycles are of even length.
- So, $2e \geq 4f$ and $v + f = e + 2$, so rearranging gives $e \leq 2v - 4$ for bipartite planar graphs.

# $K_{3,3}$ Is Not Planar



Consider $K_{3,3}$. Edges? 9. Vertices? 6. So $3v - 6 = 12$.

The previous proof fails. Make it stronger!

- The total number of sides is $2e$.
- Each face has at least three sides. Actually, at least four!
- In a bipartite graph, cycles are of even length.
- So, $2e \geq 4f$ and $v + f = e + 2$, so rearranging gives $e \leq 2v - 4$ for bipartite planar graphs.

Conclusion: $K_{3,3}$ is not planar.

# Why $K_5$ and $K_{3,3}$?

Why did we show that $K_5$ and $K_{3,3}$ are non-planar?

# Why $K_5$ and $K_{3,3}$?

Why did we show that $K_5$ and $K_{3,3}$ are non-planar?

**Kuratowski's Theorem**: A graph is non-planar if and only if it "contains" $K_5$ or $K_{3,3}$.

# Why $K_5$ and $K_{3,3}$?

Why did we show that $K_5$ and $K_{3,3}$ are non-planar?

**Kuratowski's Theorem**: A graph is non-planar if and only if it "contains" $K_5$ or $K_{3,3}$.

- The word "contains" is tricky...

# Why $K_5$ and $K_{3,3}$?

Why did we show that $K_5$ and $K_{3,3}$ are non-planar?

**Kuratowski's Theorem**: A graph is non-planar if and only if it "contains" $K_5$ or $K_{3,3}$.

- The word "contains" is tricky…do not worry about the details.

# Why $K_5$ and $K_{3,3}$?

Why did we show that $K_5$ and $K_{3,3}$ are non-planar?

**Kuratowski's Theorem**: A graph is non-planar if and only if it "contains" $K_5$ or $K_{3,3}$.

- The word "contains" is tricky. . . do not worry about the details. Not important for the course.

# Why $K_5$ and $K_{3,3}$?

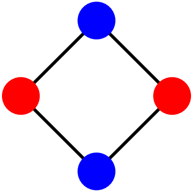
Why did we show that $K_5$ and $K_{3,3}$ are non-planar?

**Kuratowski's Theorem**: A graph is non-planar if and only if it "contains" $K_5$ or $K_{3,3}$.

- The word "contains" is tricky. . . do not worry about the details. Not important for the course.
- Content of theorem: essentially $K_5$ and $K_{3,3}$ are the only obstructions to non-planarity.

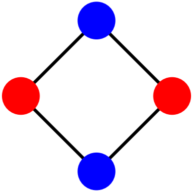
# Graph Coloring

A **(vertex) coloring** of a graph *G* is an assignment of colors to vertices so that no two colors are joined by an edge.



# Graph Coloring

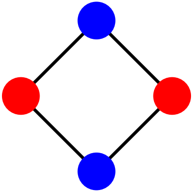
A **(vertex) coloring** of a graph $G$ is an assignment of colors to vertices so that no two colors are joined by an edge.



Why do we care about graph coloring?

# Graph Coloring

A **(vertex) coloring** of a graph $G$ is an assignment of colors to vertices so that no two colors are joined by an edge.

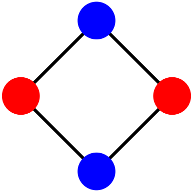


Why do we care about graph coloring?

- Edges are used to encode *constraints*.

# Graph Coloring

A **(vertex) coloring** of a graph $G$ is an assignment of colors to vertices so that no two colors are joined by an edge.



Why do we care about graph coloring?

- Edges are used to encode *constraints*.
- Graph colorings can be used for scheduling, etc.

# Coloring with Maximum Degree $+1$

**Theorem.** Let $d_{\max}$ be the maximum degree of any vertex in $G$. Then $G$ can be colored with $d_{\max} + 1$ colors.

# Coloring with Maximum Degree $+1$

**Theorem.** Let $d_{\mathrm{max}}$ be the maximum degree of any vertex in $G$. Then $G$ can be colored with $d_{\mathrm{max}}+1$ colors.

*Proof*.

# Coloring with Maximum Degree $+1$

**Theorem.** Let $d_{\text{max}}$ be the maximum degree of any vertex in $G$. Then $G$ can be colored with $d_{\text{max}} + 1$ colors.

*Proof.*

- Use induction on $|V|$.

# Coloring with Maximum Degree $+1$

**Theorem.** Let $d_{\max}$ be the maximum degree of any vertex in $G$. Then $G$ can be colored with $d_{\max} + 1$ colors.

*Proof.*

- Use induction on $|V|$.
- For $|V| \geq 2$, remove a vertex $v$.

# Coloring with Maximum Degree $+1$

**Theorem.** Let $d_{\max}$ be the maximum degree of any vertex in $G$. Then $G$ can be colored with $d_{\max} + 1$ colors.

*Proof.*

- Use induction on $|V|$.
- For $|V| \geq 2$, remove a vertex $v$.
- Inductively color the resulting graph with $d_{\max} + 1$ colors.

# Coloring with Maximum Degree $+1$

**Theorem.** Let $d_{\max}$ be the maximum degree of any vertex in $G$. Then $G$ can be colored with $d_{\max} + 1$ colors.

*Proof.*

- Use induction on $|V|$.
- For $|V| \geq 2$, remove a vertex $v$.
- Inductively color the resulting graph with $d_{\max} + 1$ colors.
- Add $v$ back in.

# Coloring with Maximum Degree $+1$

**Theorem.** Let $d_{\max}$ be the maximum degree of any vertex in $G$. Then $G$ can be colored with $d_{\max} + 1$ colors.

*Proof.*

- Use induction on $|V|$.
- For $|V| \geq 2$, remove a vertex $v$.
- Inductively color the resulting graph with $d_{\max} + 1$ colors.
- Add $v$ back in.
- Since $v$ has at most $d_{\max}$ neighbors which use at most $d_{\max}$ colors, use an unused color to color $v$. $\square$

# Coloring with Maximum Degree $+1$

**Theorem.** Let $d_{\max}$ be the maximum degree of any vertex in $G$. Then $G$ can be colored with $d_{\max} + 1$ colors.

*Proof.*

- Use induction on $|V|$.
- For $|V| \geq 2$, remove a vertex $v$.
- Inductively color the resulting graph with $d_{\max} + 1$ colors.
- Add $v$ back in.
- Since $v$ has at most $d_{\max}$ neighbors which use at most $d_{\max}$ colors, use an unused color to color $v$. $\square$

For some types of graphs, this bound is very bad.

# Bipartite Graphs Are 2-Colorable

**Theorem**: $G$ is bipartite $\iff$ $G$ can be 2-colored.

# Bipartite Graphs Are 2-Colorable

**Theorem**: $G$ is bipartite $\iff$ $G$ can be 2-colored.

*Proof*.

# Bipartite Graphs Are 2-Colorable

**Theorem**: $G$ is bipartite $\iff$ $G$ can be 2-colored.

*Proof.*

- If $G$ is bipartite with $V = L \cup R$, color vertices in $L$ blue and vertices in $R$ red.

# Bipartite Graphs Are 2-Colorable

**Theorem**: $G$ is bipartite $\iff$ $G$ can be 2-colored.

*Proof*.

- If $G$ is bipartite with $V = L \cup R$, color vertices in $L$ blue and vertices in $R$ red.
- Conversely, suppose $G$ is 2-colorable.

# Bipartite Graphs Are 2-Colorable

**Theorem**: $G$ is bipartite $\iff$ $G$ can be 2-colored.

*Proof*.

- If $G$ is bipartite with $V = L \cup R$, color vertices in $L$ blue and vertices in $R$ red.
- Conversely, suppose $G$ is 2-colorable.
- In the 2-coloring of $G$, the red vertices have no edges between them, and similarly for blue vertices.

# Bipartite Graphs Are 2-Colorable

**Theorem**: $G$ is bipartite $\iff$ $G$ can be 2-colored.

*Proof.*

- If $G$ is bipartite with $V = L \cup R$, color vertices in $L$ blue and vertices in $R$ red.
- Conversely, suppose $G$ is 2-colorable.
- In the 2-coloring of $G$, the red vertices have no edges between them, and similarly for blue vertices.
- So the graph is bipartite. $\square$

# Bipartite Graphs Are 2-Colorable

**Theorem**: $G$ is bipartite $\iff$ $G$ can be 2-colored.

*Proof*.

- If $G$ is bipartite with $V = L \cup R$, color vertices in $L$ blue and vertices in $R$ red.
- Conversely, suppose $G$ is 2-colorable.
- In the 2-coloring of $G$, the red vertices have no edges between them, and similarly for blue vertices.
- So the graph is bipartite. $\square$

Consider $K_{n,n}$.

# Bipartite Graphs Are 2-Colorable

**Theorem**: $G$ is bipartite $\iff$ $G$ can be 2-colored.

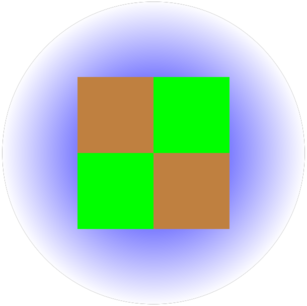
*Proof.*

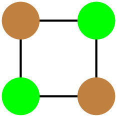
- If $G$ is bipartite with $V = L \cup R$, color vertices in $L$ blue and vertices in $R$ red.
- Conversely, suppose $G$ is 2-colorable.
- In the 2-coloring of $G$, the red vertices have no edges between them, and similarly for blue vertices.
- So the graph is bipartite. $\square$

Consider $K_{n,n}$. Then $d_{\max} + 1 = n + 1$, but it can be 2-colored.

# Graph Coloring & Planarity

Consider a colored map and its planar dual:

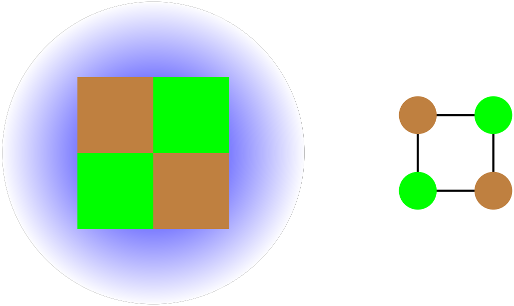




(Ignore the infinite face.)

# Graph Coloring & Planarity

Consider a colored map and its planar dual:



(Ignore the infinite face.)

Coloring a map so no adjacent regions have the same color is equivalent to coloring a planar graph.

# Four Color Theorem

**Four Color Theorem**: Every planar graph can be 4-colored.

# Four Color Theorem

**Four Color Theorem**: Every planar graph can be 4-colored.

- The proof required a human to narrow down the cases, and a computer to exhaustively check the remaining cases.

# Four Color Theorem

**Four Color Theorem**: Every planar graph can be 4-colored.

- The proof required a human to narrow down the cases, and a computer to exhaustively check the remaining cases.
- The proof has not yet been simplified to the point where a human can easily read over it.

# Four Color Theorem

**Four Color Theorem**: Every planar graph can be 4-colored.

- The proof required a human to narrow down the cases, and a computer to exhaustively check the remaining cases.
- The proof has not yet been simplified to the point where a human can easily read over it.
- Note: $K_5$ requires 5 colors.

# Hypercubes

The **hypercube** of dimension $d$, $Q_d$, where $d$ is a positive integer, has:

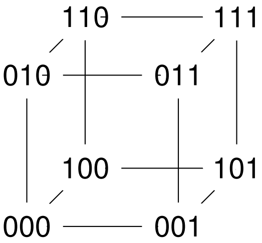
- vertices which are labeled by length-$d$ bit strings, and
- an edge between two vertices if and only if they differ in exactly one bit.

# Hypercubes

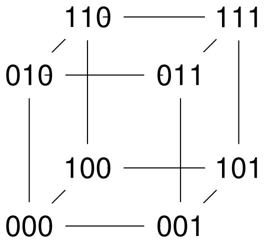
The **hypercube** of dimension $d$, $Q_d$, where $d$ is a positive integer, has:

- vertices which are labeled by length-$d$ bit strings, and
- an edge between two vertices if and only if they differ in exactly one bit.

Here is a picture of $Q_3$.

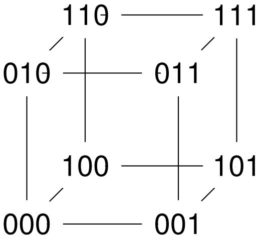


# Hypercube Facts



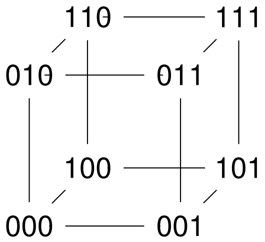

The 0-**face** is the part of the hypercube whose vertices begin with 0.

# Hypercube Facts




The 0**-face** is the part of the hypercube whose vertices begin with 0. Similarly for the 1**-face.**

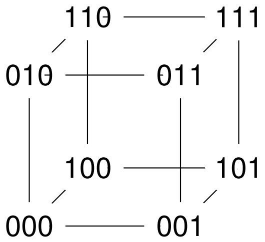
# Hypercube Facts




The 0-**face** is the part of the hypercube whose vertices begin with 0. Similarly for the 1-**face**.

The 0-face is a lower-dimensional hypercube.

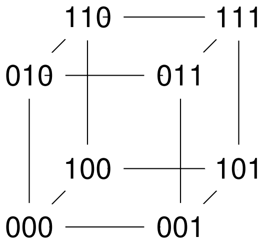
# Hypercube Facts




The 0-**face** is the part of the hypercube whose vertices begin with 0. Similarly for the 1-**face.**

The 0-face is a lower-dimensional hypercube. Induction!

# Hypercube Facts

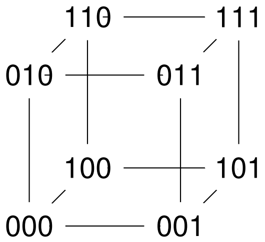



The 0-**face** is the part of the hypercube whose vertices begin with 0. Similarly for the 1-**face.**

The 0-face is a lower-dimensional hypercube. Induction!

Number of vertices?

# Hypercube Facts

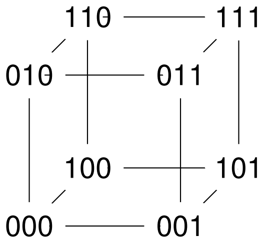



The 0-**face** is the part of the hypercube whose vertices begin with 0. Similarly for the 1-**face**.

The 0-face is a lower-dimensional hypercube. Induction!

Number of vertices? $2^d$.

# Hypercube Facts

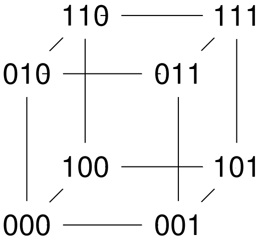



The 0-**face** is the part of the hypercube whose vertices begin with 0. Similarly for the 1-**face.**

The 0-face is a lower-dimensional hypercube. Induction!

Number of vertices? $2^d$.
Number of edges?

# Hypercube Facts

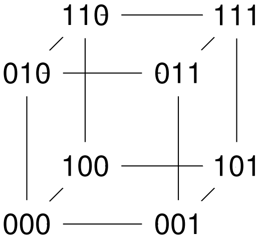



The 0**-face** is the part of the hypercube whose vertices begin with 0. Similarly for the 1**-face**.

The 0-face is a lower-dimensional hypercube. Induction!

Number of vertices? $2^d$.
Number of edges? $\sum_{v \in V} \deg v = d2^d$, so $|E| = d2^{d-1}$.

# Hypercube Facts




The 0-**face** is the part of the hypercube whose vertices begin with 0. Similarly for the 1-**face**.

The 0-face is a lower-dimensional hypercube. Induction!

Number of vertices? $2^d$.
Number of edges? $\sum_{v \in V} \deg v = d2^d$, so $|E| = d2^{d-1}$.

So for a hypercube with $n$ vertices, $|E| = \Theta(n \log n)$.

# Hypercubes Are Bipartite

**Theorem**: Hypercubes are 2-colorable.

# Hypercubes Are Bipartite

**Theorem**: Hypercubes are 2-colorable.

*Proof*.

# Hypercubes Are Bipartite

**Theorem**: Hypercubes are 2-colorable.

*Proof*.

- Color all vertices with an even number of 0s blue and an odd number of 0s orange.

# Hypercubes Are Bipartite

**Theorem**: Hypercubes are 2-colorable.

*Proof.*

- Color all vertices with an even number of 0s blue and an odd number of 0s orange.
- Since each edge flips a bit, edges only connect vertices of different parity. □

# Hypercubes Are Bipartite

**Theorem**: Hypercubes are 2-colorable.

*Proof*.

- Color all vertices with an even number of 0s blue and an odd number of 0s orange.
- Since each edge flips a bit, edges only connect vertices of different parity. □

*Inductive Proof*.

# Hypercubes Are Bipartite

**Theorem**: Hypercubes are 2-colorable.

*Proof*.

- Color all vertices with an even number of 0s blue and an odd number of 0s orange.
- Since each edge flips a bit, edges only connect vertices of different parity. □

*Inductive Proof*.

- Check the base case.

# Hypercubes Are Bipartite

**Theorem**: Hypercubes are 2-colorable.

*Proof*.

- Color all vertices with an even number of 0s blue and an odd number of 0s orange.
- Since each edge flips a bit, edges only connect vertices of different parity. □

*Inductive Proof*.

- Check the base case.
- Inductively color the 0-face.

# Hypercubes Are Bipartite

**Theorem**: Hypercubes are 2-colorable.

*Proof*.

- Color all vertices with an even number of 0s blue and an odd number of 0s orange.
- Since each edge flips a bit, edges only connect vertices of different parity. □

*Inductive Proof*.

- Check the base case.
- Inductively color the 0-face.
- If $0x$ is a vertex colored blue, color the vertex $1x$ orange and if $0x$ is orange, color $1x$ blue. □

# Hamiltonian Paths

Recall: List all bit strings of length 3, flipping one bit at a time.

# Hamiltonian Paths

Recall: List all bit strings of length 3, flipping one bit at a time.

A **Hamiltonian cycle** is a cycle that includes every vertex exactly once.

# Hamiltonian Paths

Recall: List all bit strings of length 3, flipping one bit at a time.

A **Hamiltonian cycle** is a cycle that includes every vertex exactly once.

Listing the bit strings while flipping one bit at a time is exactly a Hamiltonian cycle on the hypercube.

# Hamiltonian Paths

Recall: List all bit strings of length 3, flipping one bit at a time.

A **Hamiltonian cycle** is a cycle that includes every vertex exactly once.

Listing the bit strings while flipping one bit at a time is exactly a Hamiltonian cycle on the hypercube.

Inductive construction:

# Hamiltonian Paths

Recall: List all bit strings of length 3, flipping one bit at a time.

A **Hamiltonian cycle** is a cycle that includes every vertex exactly once.

Listing the bit strings while flipping one bit at a time is exactly a Hamiltonian cycle on the hypercube.

Inductive construction:

- Length 1: 0, 1.

# Hamiltonian Paths

Recall: List all bit strings of length 3, flipping one bit at a time.

A **Hamiltonian cycle** is a cycle that includes every vertex exactly once.

Listing the bit strings while flipping one bit at a time is exactly a Hamiltonian cycle on the hypercube.

Inductive construction:

- Length 1: 0, 1.
- Length 2: Length-1 sequence with 0s prepended.

# Hamiltonian Paths

Recall: List all bit strings of length 3, flipping one bit at a time.

A **Hamiltonian cycle** is a cycle that includes every vertex exactly once.

Listing the bit strings while flipping one bit at a time is exactly a Hamiltonian cycle on the hypercube.

Inductive construction:

- Length 1: 0, 1.
- Length 2: Length-1 sequence with 0s prepended. 00, 01.

# Hamiltonian Paths

Recall: List all bit strings of length 3, flipping one bit at a time.

A **Hamiltonian cycle** is a cycle that includes every vertex exactly once.

Listing the bit strings while flipping one bit at a time is exactly a Hamiltonian cycle on the hypercube.

Inductive construction:

- Length 1: 0, 1.
- Length 2: Length-1 sequence with 0s prepended. 00, 01. Length-1 sequence *backwards* with 1s prepended.

# Hamiltonian Paths

Recall: List all bit strings of length 3, flipping one bit at a time.

A **Hamiltonian cycle** is a cycle that includes every vertex exactly once.

Listing the bit strings while flipping one bit at a time is exactly a Hamiltonian cycle on the hypercube.

Inductive construction:

- Length 1: 0, 1.
- Length 2: Length-1 sequence with 0s prepended. 00, 01. Length-1 sequence *backwards* with 1s prepended. 11, 10.

# Hamiltonian Paths

Recall: List all bit strings of length 3, flipping one bit at a time.

A **Hamiltonian cycle** is a cycle that includes every vertex exactly once.

Listing the bit strings while flipping one bit at a time is exactly a Hamiltonian cycle on the hypercube.

Inductive construction:

- Length 1: 0, 1.
- Length 2: Length-1 sequence with 0s prepended. 00, 01.
  Length-1 sequence *backwards* with 1s prepended. 11, 10.
  Put it together: 00, 01, 11, 10.

# Hamiltonian Paths

Recall: List all bit strings of length 3, flipping one bit at a time.

A **Hamiltonian cycle** is a cycle that includes every vertex exactly once.

Listing the bit strings while flipping one bit at a time is exactly a Hamiltonian cycle on the hypercube.

Inductive construction:

- Length 1: 0, 1.
- Length 2: Length-1 sequence with 0s prepended. 00, 01.
  Length-1 sequence *backwards* with 1s prepended. 11, 10.
  Put it together: 00, 01, 11, 10.
- Length 3: 000, 001, 011, 010, 110, 111, 101, 100.

# Hamiltonian Paths

Recall: List all bit strings of length 3, flipping one bit at a time.

A **Hamiltonian cycle** is a cycle that includes every vertex exactly once.

Listing the bit strings while flipping one bit at a time is exactly a Hamiltonian cycle on the hypercube.

Inductive construction:

- Length 1: 0, 1.
- Length 2: Length-1 sequence with 0s prepended. 00, 01.
  Length-1 sequence *backwards* with 1s prepended. 11, 10.
  Put it together: 00, 01, 11, 10.
- Length 3: 000, 001, 011, 010, 110, 111, 101, 100.

Hypercubes have Hamiltonian cycles.

# Clock Mathematics

If it is 2:00 right now, what time is it in 24 hours?

# Clock Mathematics

If it is 2:00 right now, what time is it in 24 hours? Still 2:00.

# Clock Mathematics

If it is 2:00 right now, what time is it in 24 hours? Still 2:00.

In the clock mathematics, the numbers *wrap around*: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, . . .

# Clock Mathematics

If it is 2:00 right now, what time is it in 24 hours? Still 2:00.

In the clock mathematics, the numbers *wrap around*: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, . . .

We will do the same thing for bases other than 12.

# Clock Mathematics

If it is 2:00 right now, what time is it in 24 hours? Still 2:00.

In the clock mathematics, the numbers *wrap around*: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, . . .

We will do the same thing for bases other than 12.
Also, we will typically use the representatives $\{0, 1, \ldots, 11\}$ rather than $\{1, \ldots, 12\}$.

# Clock Mathematics

If it is 2:00 right now, what time is it in 24 hours? Still 2:00.

In the clock mathematics, the numbers *wrap around*: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, ...

We will do the same thing for bases other than 12.
Also, we will typically use the representatives $\{0, 1, \ldots, 11\}$ rather than $\{1, \ldots, 12\}$.

Question to ponder: What time will it be in $2^{1000000}$ hours from now?

# Clock Mathematics

If it is 2:00 right now, what time is it in 24 hours? Still 2:00.

In the clock mathematics, the numbers *wrap around*: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, ...

We will do the same thing for bases other than 12.
Also, we will typically use the representatives $\{0, 1, \ldots, 11\}$ rather than $\{1, \ldots, 12\}$.

Question to ponder: What time will it be in $2^{1000000}$ hours from now? Can this even be computed?

# Modular Equivalence

Let $m$ be a positive integer.

# Modular Equivalence

Let $m$ be a positive integer.

For the next few lectures, $m$ will be called the **modulus**.

# Modular Equivalence

Let $m$ be a positive integer.
For the next few lectures, $m$ will be called the **modulus**.

Say that $x \equiv y \pmod{m}$ if $m \mid x - y$.

# Modular Equivalence

Let $m$ be a positive integer.

For the next few lectures, $m$ will be called the **modulus**.

Say that $x \equiv y \pmod{m}$ if $m \mid x - y$.

Read this as "$x$ is equivalent to $y$, modulo $m$."

# Modular Equivalence

Let $m$ be a positive integer.
For the next few lectures, $m$ will be called the **modulus.**

Say that $x \equiv y \pmod{m}$ if $m \mid x - y$.
Read this as "$x$ is equivalent to $y$, modulo $m$."

Examples: What numbers are equivalent to 0, modulo 6?

# Modular Equivalence

Let $m$ be a positive integer.
For the next few lectures, $m$ will be called the **modulus**.

Say that $x \equiv y \pmod{m}$ if $m \mid x - y$.
Read this as "$x$ is equivalent to $y$, modulo $m$."

Examples: What numbers are equivalent to 0, modulo 6?

- $\ldots, -18, -12, -6, 0, 6, 12, 18, \ldots$.

# Modular Equivalence

Let $m$ be a positive integer.
For the next few lectures, $m$ will be called the **modulus**.

Say that $x \equiv y \pmod{m}$ if $m \mid x - y$.
Read this as "$x$ is equivalent to $y$, modulo $m$."

Examples: What numbers are equivalent to 0, modulo 6?

- $\ldots, -18, -12, -6, 0, 6, 12, 18, \ldots$

In the "modulo 6" system, think of these numbers as the same.

# Modular Equivalence: Addition, Multiplication

**Theorem**: If $a, b, c, d \in \mathbb{Z}$ with

$$a \equiv c \pmod{m} \qquad \text{and} \qquad b \equiv d \pmod{m},$$

then $a + b \equiv c + d \pmod{m}$ and $ab \equiv cd \pmod{m}$.

# Modular Equivalence: Addition, Multiplication

**Theorem**: If $a, b, c, d \in \mathbb{Z}$ with

$$a \equiv c \pmod{m} \qquad \text{and} \qquad b \equiv d \pmod{m},$$

then $a + b \equiv c + d \pmod{m}$ and $ab \equiv cd \pmod{m}$.

Addition and multiplication work as usual in modular arithmetic.

# Modular Equivalence: Addition, Multiplication

**Theorem**: If $a, b, c, d \in \mathbb{Z}$ with

$$a \equiv c \pmod{m} \qquad \text{and} \qquad b \equiv d \pmod{m},$$

then $a + b \equiv c + d \pmod{m}$ and $ab \equiv cd \pmod{m}$.

Addition and multiplication work as usual in modular arithmetic.

*Proof*.

# Modular Equivalence: Addition, Multiplication

**Theorem**: If $a, b, c, d \in \mathbb{Z}$ with

$$a \equiv c \pmod{m} \qquad \text{and} \qquad b \equiv d \pmod{m},$$

then $a + b \equiv c + d \pmod{m}$ and $ab \equiv cd \pmod{m}$.

Addition and multiplication work as usual in modular arithmetic.

*Proof*.

- By definition, $m \mid a - c$ and $m \mid b - d$.

# Modular Equivalence: Addition, Multiplication

**Theorem**: If $a, b, c, d \in \mathbb{Z}$ with

$$a \equiv c \pmod{m} \qquad \text{and} \qquad b \equiv d \pmod{m},$$

then $a + b \equiv c + d \pmod{m}$ and $ab \equiv cd \pmod{m}$.

Addition and multiplication work as usual in modular arithmetic.

*Proof*.

- By definition, $m \mid a - c$ and $m \mid b - d$.
- So, $m \mid a + b - (c + d)$.

# Modular Equivalence: Addition, Multiplication

**Theorem**: If $a, b, c, d \in \mathbb{Z}$ with

$$a \equiv c \pmod{m} \qquad \text{and} \qquad b \equiv d \pmod{m},$$

then $a + b \equiv c + d \pmod{m}$ and $ab \equiv cd \pmod{m}$.

Addition and multiplication work as usual in modular arithmetic.

*Proof*.

- By definition, $m \mid a - c$ and $m \mid b - d$.
- So, $m \mid a + b - (c + d)$.
- Also $a = km + c$ and $b = \ell m + d$ for some $k, \ell \in \mathbb{Z}$.

# Modular Equivalence: Addition, Multiplication

**Theorem**: If $a, b, c, d \in \mathbb{Z}$ with

$$a \equiv c \pmod{m} \qquad \text{and} \qquad b \equiv d \pmod{m},$$

then $a + b \equiv c + d \pmod{m}$ and $ab \equiv cd \pmod{m}$.

Addition and multiplication work as usual in modular arithmetic.

*Proof*.

- By definition, $m \mid a - c$ and $m \mid b - d$.
- So, $m \mid a + b - (c + d)$.
- Also $a = km + c$ and $b = \ell m + d$ for some $k, \ell \in \mathbb{Z}$.
- So, $ab = k\ell m^2 + dkm + c\ell m + cd$.

# Modular Equivalence: Addition, Multiplication

**Theorem**: If $a,b,c,d \in \mathbb{Z}$ with

$$a \equiv c \pmod{m} \qquad \text{and} \qquad b \equiv d \pmod{m},$$

then $a+b \equiv c+d \pmod{m}$ and $ab \equiv cd \pmod{m}$.

Addition and multiplication work as usual in modular arithmetic.

*Proof*.

- By definition, $m \mid a-c$ and $m \mid b-d$.
- So, $m \mid a+b-(c+d)$.
- Also $a = km+c$ and $b = \ell m+d$ for some $k,\ell \in \mathbb{Z}$.
- So, $ab = k\ell m^2 + dkm + c\ell m + cd$.
- Hence $m \mid ab-cd$. $\square$

# Representatives

**Theorem**: Each integer $x$ is equivalent to a unique member of $\{0, 1, \ldots, m-1\}$ modulo $m$.

# Representatives

**Theorem**: Each integer $x$ is equivalent to a unique member of $\{0, 1, \ldots, m-1\}$ modulo $m$.

*Proof*.

# Representatives

**Theorem**: Each integer $x$ is equivalent to a unique member of $\{0, 1, \ldots, m-1\}$ modulo $m$.

*Proof*.

- By Division Algorithm, $x = qm + r$ for some $q \in \mathbb{Z}$ and $r \in \{0, 1, \ldots, m-1\}$.

# Representatives

**Theorem**: Each integer $x$ is equivalent to a unique member of $\{0, 1, \ldots, m-1\}$ modulo $m$.

*Proof*.

- By Division Algorithm, $x = qm + r$ for some $q \in \mathbb{Z}$ and $r \in \{0, 1, \ldots, m-1\}$.
- Thus $m \mid x - r$, i.e., $x \equiv r \pmod{m}$.

# Representatives

**Theorem**: Each integer $x$ is equivalent to a unique member of $\{0, 1, \ldots, m-1\}$ modulo $m$.

*Proof*.

- By Division Algorithm, $x = qm + r$ for some $q \in \mathbb{Z}$ and $r \in \{0, 1, \ldots, m-1\}$.
- Thus $m \mid x - r$, i.e., $x \equiv r \pmod{m}$.
- If $x \equiv r_1 \pmod{m}$ and $x \equiv r_2 \pmod{m}$, then (by subtracting) $r_1 - r_2 \equiv 0 \pmod{m}$.

# Representatives

**Theorem**: Each integer $x$ is equivalent to a unique member of $\{0, 1, \ldots, m-1\}$ modulo $m$.

*Proof*.

- By Division Algorithm, $x = qm + r$ for some $q \in \mathbb{Z}$ and $r \in \{0, 1, \ldots, m-1\}$.
- Thus $m \mid x - r$, i.e., $x \equiv r \pmod{m}$.
- If $x \equiv r_1 \pmod{m}$ and $x \equiv r_2 \pmod{m}$, then (by subtracting) $r_1 - r_2 \equiv 0 \pmod{m}$.
- But this is impossible if $r_1, r_2 \in \{0, 1, \ldots, m-1\}$ are distinct. $\square$

# Representatives

**Theorem**: Each integer $x$ is equivalent to a unique member of $\{0,1,\ldots,m-1\}$ modulo $m$.

*Proof*.

- By Division Algorithm, $x = qm + r$ for some $q \in \mathbb{Z}$ and $r \in \{0,1,\ldots,m-1\}$.
- Thus $m \mid x - r$, i.e., $x \equiv r \pmod{m}$.
- If $x \equiv r_1 \pmod{m}$ and $x \equiv r_2 \pmod{m}$, then (by subtracting) $r_1 - r_2 \equiv 0 \pmod{m}$.
- But this is impossible if $r_1, r_2 \in \{0,1,\ldots,m-1\}$ are distinct. $\square$

Now we can think of the numbers $\{0,1,\ldots,m-1\}$ with addition and multiplication (modulo $m$) as a number system.

# Representatives

**Theorem**: Each integer $x$ is equivalent to a unique member of $\{0, 1, \ldots, m-1\}$ modulo $m$.

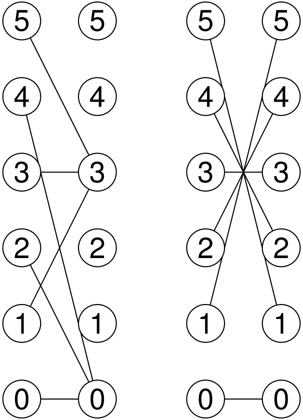
*Proof*.

- By Division Algorithm, $x = qm + r$ for some $q \in \mathbb{Z}$ and $r \in \{0, 1, \ldots, m-1\}$.
- Thus $m \mid x - r$, i.e., $x \equiv r \pmod{m}$.
- If $x \equiv r_1 \pmod{m}$ and $x \equiv r_2 \pmod{m}$, then (by subtracting) $r_1 - r_2 \equiv 0 \pmod{m}$.
- But this is impossible if $r_1, r_2 \in \{0, 1, \ldots, m-1\}$ are distinct. $\square$

Now we can think of the numbers $\{0, 1, \ldots, m-1\}$ with addition and multiplication (modulo $m$) as a number system.
This system is usually called $\mathbb{Z}/m\mathbb{Z}$.

# Multiplication in Modular Arithmetic

Modulo 6:




Left: Going from left to right is multiplication by 3.

Right: Going from left to right is multiplication by 5.

# Bijections

A function $f : A \rightarrow B$ is:

# Bijections

A function $f : A \rightarrow B$ is:

- **injective** (or **one-to-one**) if for $x_1 \neq x_2$, $f(x_1) \neq f(x_2)$

# Bijections

A function $f : A \to B$ is:

- **injective** (or **one-to-one**) if for $x_1 \neq x_2$, $f(x_1) \neq f(x_2)$ (different inputs mapped to different outputs);

# Bijections

A function $f : A \to B$ is:

- **injective** (or **one-to-one**) if for $x_1 \neq x_2$, $f(x_1) \neq f(x_2)$ (different inputs mapped to different outputs);
- **surjective** (or **onto**) if for every $y \in B$, there is an $x \in A$ with $f(x) = y$

# Bijections

A function $f : A \to B$ is:

- **injective** (or **one-to-one**) if for $x_1 \neq x_2$, $f(x_1) \neq f(x_2)$ (different inputs mapped to different outputs);
- **surjective** (or **onto**) if for every $y \in B$, there is an $x \in A$ with $f(x) = y$ (every element of $B$ is hit);

# Bijections

A function $f : A \to B$ is:

- **injective** (or **one-to-one**) if for $x_1 \neq x_2$, $f(x_1) \neq f(x_2)$ (different inputs mapped to different outputs);
- **surjective** (or **onto**) if for every $y \in B$, there is an $x \in A$ with $f(x) = y$ (every element of $B$ is hit);
- **bijective** if it is both injective and surjective.

# Bijections

A function $f : A \to B$ is:

- **injective** (or **one-to-one**) if for $x_1 \neq x_2$, $f(x_1) \neq f(x_2)$ (different inputs mapped to different outputs);
- **surjective** (or **onto**) if for every $y \in B$, there is an $x \in A$ with $f(x) = y$ (every element of $B$ is hit);
- **bijective** if it is both injective and surjective.

A bijection is like relabeling the elements of $A$.

# Bijections

A function $f : A \to B$ is:

- **injective** (or **one-to-one**) if for $x_1 \neq x_2$, $f(x_1) \neq f(x_2)$ (different inputs mapped to different outputs);
- **surjective** (or **onto**) if for every $y \in B$, there is an $x \in A$ with $f(x) = y$ (every element of $B$ is hit);
- **bijective** if it is both injective and surjective.

A bijection is like relabeling the elements of $A$.

Consider the map "multiplication by $a$, modulo $m$".

# Bijections

A function $f : A \to B$ is:

- **injective** (or **one-to-one**) if for $x_1 \neq x_2$, $f(x_1) \neq f(x_2)$ (different inputs mapped to different outputs);
- **surjective** (or **onto**) if for every $y \in B$, there is an $x \in A$ with $f(x) = y$ (every element of $B$ is hit);
- **bijective** if it is both injective and surjective.

A bijection is like relabeling the elements of $A$.

Consider the map "multiplication by $a$, modulo $m$". That is, $f(x) := ax \bmod m$.

# Bijections

A function $f : A \to B$ is:

- **injective** (or **one-to-one**) if for $x_1 \neq x_2$, $f(x_1) \neq f(x_2)$ (different inputs mapped to different outputs);
- **surjective** (or **onto**) if for every $y \in B$, there is an $x \in A$ with $f(x) = y$ (every element of $B$ is hit);
- **bijective** if it is both injective and surjective.

A bijection is like relabeling the elements of $A$.

Consider the map "multiplication by $a$, modulo $m$". That is, $f(x) := ax \bmod m$.

When is this map bijective?

# Greatest Common Divisor

For two integers $a, b \in \mathbb{Z}$, the **greatest common divisor (GCD)** of $a$ and $b$ is the largest number that divides both $a$ and $b$.

# Greatest Common Divisor

For two integers $a, b \in \mathbb{Z}$, the **greatest common divisor (GCD)** of $a$ and $b$ is the largest number that divides both $a$ and $b$.

**Fact**: Any common divisor of $a$ and $b$ also divides $\gcd(a, b)$.

# Greatest Common Divisor

For two integers $a, b \in \mathbb{Z}$, the **greatest common divisor (GCD)** of $a$ and $b$ is the largest number that divides both $a$ and $b$.

**Fact**: Any common divisor of $a$ and $b$ also divides $\gcd(a, b)$.

(Proof: Next time!)

# Existence of Multiplicative Inverses

**Theorem**: $f(x) = ax \bmod m$ is bijective if and only if $\gcd(a, m) = 1$.

# Existence of Multiplicative Inverses

**Theorem**: $f(x) = ax \bmod m$ is bijective if and only if $\gcd(a, m) = 1$.

For $a \in \mathbb{Z}/m\mathbb{Z}$, a **multiplicative inverse** $x$ is an element of $\mathbb{Z}/m\mathbb{Z}$ for which $ax \equiv 1 \pmod{m}$.

# Existence of Multiplicative Inverses

**Theorem**: $f(x) = ax \bmod m$ is bijective if and only if $\gcd(a, m) = 1$.

For $a \in \mathbb{Z}/m\mathbb{Z}$, a **multiplicative inverse** $x$ is an element of $\mathbb{Z}/m\mathbb{Z}$ for which $ax \equiv 1 \pmod{m}$.

**Corollary**: For all $a \in \mathbb{Z}/m\mathbb{Z}$, $a$ has a multiplicative inverse (necessarily unique) if and only if $\gcd(a, m) = 1$.

# Existence of Multiplicative Inverses

**Theorem**: $f(x) = ax \bmod m$ is bijective if and only if $\gcd(a, m) = 1$.

For $a \in \mathbb{Z}/m\mathbb{Z}$, a **multiplicative inverse** $x$ is an element of $\mathbb{Z}/m\mathbb{Z}$ for which $ax \equiv 1 \pmod{m}$.

**Corollary**: For all $a \in \mathbb{Z}/m\mathbb{Z}$, $a$ has a multiplicative inverse (necessarily unique) if and only if $\gcd(a, m) = 1$.

(Proof: Next time!)

# Summary

Graphs.

- Consequences of Euler's Formula: non-planarity of $K_5$ and $K_{3,3}$; planar graphs are sparse.
- Types of graphs: forests, hypercubes.
- Graph colorings: $\leq d_{\max} + 1$ for general graphs, 2 for bipartite graphs.
- Hypercubes have Hamiltonian cycles.

Modular arithmetic.

- $a \equiv b \pmod{m}$ if $m \mid a - b$.
- Each number modulo $m$ has a representative in $\{0, 1, \ldots, m-1\}$.
- Injections, surjections, bijections. . .
- $a$ has a multiplicative inverse modulo $m$ if and only if $\gcd(a, m) = 1$.