Today.

Types of graphs.
Types of graphs.
  Complete Graphs.
  Trees.
  Hypercubes.
Today.

Types of graphs.
  Complete Graphs.
  Trees.
  Hypercubes.
Complete Graph.

$K_n$ complete graph on $n$ vertices.
Complete Graph.

$K_n$ complete graph on $n$ vertices. All edges are present.
Complete Graph.

$K_n$ complete graph on $n$ vertices.
All edges are present.
Everyone is my neighbor.
Complete Graph.

$K_n$ complete graph on $n$ vertices.
All edges are present.
Everyone is my neighbor.
Each vertex is adjacent to every other vertex.
Complete Graph.

$K_n$ complete graph on $n$ vertices.
All edges are present.
Everyone is my neighbor.
Each vertex is adjacent to every other vertex.
Complete Graph.

$K_n$ complete graph on $n$ vertices.
All edges are present.
Everyone is my neighbor.
Each vertex is adjacent to every other vertex.

How many edges?
Complete Graph.

$K_n$ complete graph on $n$ vertices.
All edges are present.
Everyone is my neighbor.
Each vertex is adjacent to every other vertex.

How many edges?
Each vertex is incident to $n - 1$ edges.
Complete Graph.

\[ K_n \] complete graph on \( n \) vertices.
All edges are present.
Everyone is my neighbor.
Each vertex is adjacent to every other vertex.

How many edges?
Each vertex is incident to \( n - 1 \) edges.
Sum of degrees is \( n(n - 1) \).
Complete Graph.

$K_n$ complete graph on $n$ vertices.
All edges are present.
Everyone is my neighbor.
Each vertex is adjacent to every other vertex.

How many edges?
Each vertex is incident to $n - 1$ edges.
Sum of degrees is $n(n - 1)$.
$\implies$ Number of edges is $n(n - 1)/2$. 
Complete Graph.

\( K_n \) complete graph on \( n \) vertices.
All edges are present.
Everyone is my neighbor.
Each vertex is adjacent to every other vertex.

How many edges?
Each vertex is incident to \( n - 1 \) edges.
Sum of degrees is \( n(n - 1) \).
\[ \implies \text{Number of edges is } n(n-1)/2. \]
Remember sum of degree is \( 2|E| \).
$K_4$ and $K_5$

$K_5$ is not planar.
$K_4$ and $K_5$

$K_5$ is not planar.
Cannot be drawn in the plane without an edge crossing!
\( K_4 \) and \( K_5 \)

\( K_5 \) is not planar.

Cannot be drawn in the plane without an edge crossing!

Prove it!
$K_4$ and $K_5$

$K_5$ is not planar.
Cannot be drawn in the plane without an edge crossing!
Prove it! Read Note 5!!
Graph $G = (V, E)$.

Binary Tree!

More generally.
Trees: Definitions

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- A connected graph where any edge addition creates a cycle.

Some trees:

- Yes and connected?
- Yes.

- Removing any edge disconnects it.

- Adding any edge creates a cycle.

Tree or not tree!
Trees: Definitions

Definitions:

A connected graph without a cycle.
Trees: Definitions

Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
Trees: Definitions

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
Trees: Definitions

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- A connected graph where any edge addition creates a cycle.
Trees: Definitions

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- A connected graph where any edge addition creates a cycle.
Trees: Definitions

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- A connected graph where any edge addition creates a cycle.

Some trees.

![Graphs]

no cycle and connected?
Trees: Definitions

Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
A connected graph where any edge addition creates a cycle.

Some trees.

no cycle and connected? Yes.
Trees: Definitions

Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
A connected graph where any edge addition creates a cycle.

Some trees.

no cycle and connected? Yes.
$|V| - 1$ edges and connected?
Trees: Definitions

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- A connected graph where any edge addition creates a cycle.

Some trees.

- no cycle and connected? Yes.
- $|V| - 1$ edges and connected? Yes.
Trees: Definitions

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- A connected graph where any edge addition creates a cycle.

Some trees.

- no cycle and connected? Yes.
- $|V| - 1$ edges and connected? Yes.
- removing any edge disconnects it.
Trees: Definitions

Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
A connected graph where any edge addition creates a cycle.

Some trees.

no cycle and connected? Yes.
$|V| - 1$ edges and connected? Yes.
removing any edge disconnects it. Harder to check.
Trees: Definitions

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- A connected graph where any edge addition creates a cycle.

Some trees.

- no cycle and connected? Yes.
- $|V| - 1$ edges and connected? Yes.
- removing any edge disconnects it. Harder to check. but yes.
Trees: Definitions

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- A connected graph where any edge addition creates a cycle.

Some trees.

- no cycle and connected? Yes.
- $|V| - 1$ edges and connected? Yes.
- removing any edge disconnects it. Harder to check. but yes.
- Adding any edge creates cycle.
Trees: Definitions

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- A connected graph where any edge addition creates a cycle.

Some trees.

no cycle and connected? Yes.
$|V| - 1$ edges and connected? Yes.
removing any edge disconnects it. Harder to check. but yes.
Adding any edge creates cycle. Harder to check.
Trees: Definitions

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- A connected graph where any edge addition creates a cycle.

Some trees.

- no cycle and connected? Yes.
- $|V| - 1$ edges and connected? Yes.
- removing any edge disconnects it. Harder to check. but yes.
- Adding any edge creates cycle. Harder to check. but yes.
Trees: Definitions

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- A connected graph where any edge addition creates a cycle.

Some trees.

- no cycle and connected? Yes.
- $|V| - 1$ edges and connected? Yes.
- removing any edge disconnects it. Harder to check. but yes.
- Adding any edge creates cycle. Harder to check. but yes.
Trees: Definitions

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- A connected graph where any edge addition creates a cycle.

Some trees.

- [Diagram of a tree with no cycle and connected]
- [Diagram of a tree with $|V| - 1$ edges and connected]
- [Diagram of a tree where removing any edge disconnects it]
- [Diagram of a tree where adding any edge creates a cycle]

no cycle and connected? Yes.
$|V| - 1$ edges and connected? Yes.
removing any edge disconnects it. Harder to check. but yes.
Adding any edge creates cycle. Harder to check. but yes.

Tree or not tree!
Thm:
“G connected and has \(|V| - 1\) edges” \(\equiv\)
“G is connected and has no cycles.”
Thm:
“G connected and has $|V| - 1$ edges” \(\equiv\) “G is connected and has no cycles.”

Proof of \(\implies\) (only if):

By induction on $|V|$. 

Base Case: $|V| = 1.$ $0$ = $|V| - 1$ edges and has no cycles.

Induction Step: Assume for $G$ with up to $k$ vertices. Prove for $k + 1$.

Consider some vertex $v$ in $G$. How is it connected to the rest of $G$? Might it be connected by just 1 edge? Is there a Degree 1 vertex? Is the rest of $G$ connected?
Thm:
“G connected and has $|V| - 1$ edges” $\equiv$
“G is connected and has no cycles.”

Proof of $\implies$ (only if): By induction on $|V|$. 
Thm:
“G connected and has $|V| - 1$ edges” $\equiv$
“G is connected and has no cycles.”

Proof of $\implies$ (only if): By induction on $|V|$.
Base Case: $|V| = 1$. $0 = |V| - 1$ edges and has no cycles.
Equivalence of Definitions

Thm:
“G connected and has $|V| - 1$ edges” $\equiv$
“G is connected and has no cycles.”

Proof of $\implies$ (only if): By induction on $|V|$.
Base Case: $|V| = 1$. $0 = |V| - 1$ edges and has no cycles.
Thm:
“G connected and has $|V| - 1$ edges” $\equiv$
“G is connected and has no cycles.”

Proof of $\implies$ (only if): By induction on $|V|$.
Base Case: $|V| = 1$. $0 = |V| - 1$ edges and has no cycles.

Induction Step: Assume for $G$ with up to $k$ vertices. Prove for $k + 1$
Thm:
“G connected and has $|V| - 1$ edges” ≡
“G is connected and has no cycles.”

Proof of (only if): By induction on $|V|$.
Base Case: $|V| = 1$. $0 = |V| - 1$ edges and has no cycles.

Induction Step: Assume for $G$ with up to $k$ vertices. Prove for $k + 1$ vertices.
Equivalence of Definitions

**Thm:**
“G connected and has $|V| - 1$ edges” ≡
“G is connected and has no cycles.”

**Proof of $\implies$ (only if):** By induction on $|V|$.

**Base Case:** $|V| = 1$. $0 = |V| - 1$ edges and has no cycles.

**Induction Step:** Assume for $G$ with up to $k$ vertices. Prove for $k + 1$

Consider some vertex $v$ in $G$. How is it connected to the rest of $G$?
Might it be connected by just 1 edge?
Equivalence of Definitions

Thm:
“$G$ connected and has $|V| - 1$ edges” $≡$
“$G$ is connected and has no cycles.”

Proof of $\implies$ (only if): By induction on $|V|$.
Base Case: $|V| = 1$. $0 = |V| - 1$ edges and has no cycles.

Induction Step: Assume for $G$ with up to $k$ vertices. Prove for $k + 1$
Consider some vertex $v$ in $G$. How is it connected to the rest of $G$?
Might it be connected by just 1 edge?
Is there a Degree 1 vertex?
Thm:
“G connected and has $|V| - 1$ edges” $\equiv$
“G is connected and has no cycles.”

Proof of $\implies$ (only if): By induction on $|V|$.
Base Case: $|V| = 1$. $0 = |V| - 1$ edges and has no cycles.

Induction Step: Assume for $G$ with up to $k$ vertices. Prove for $k + 1$
Consider some vertex $v$ in $G$. How is it connected to the rest of $G$?
Might it be connected by just 1 edge?
Is there a Degree 1 vertex?
Is the rest of $G$ connected?
Theorem:
“G connected and has $|V| - 1$ edges” $\equiv$
“G is connected and has no cycles.”

Lemma: If $v$ is a degree 1 in connected graph $G$, $G - v$ is connected.
Proof:
For $x \neq v, y \neq v \in V$, 
Theorem: “G connected and has $|V| - 1$ edges” $\equiv$ “G is connected and has no cycles.”

Lemma: If $v$ is a degree 1 in connected graph $G$, $G - v$ is connected.

Proof: For $x \neq v, y \neq v \in V$,
there is path between $x$ and $y$ in $G$ since connected.
Equivalence of Definitions: Useful Lemma

**Theorem:**
“$G$ connected and has $|V| - 1$ edges” ≡
“$G$ is connected and has no cycles.”

**Lemma:** If $v$ is a degree 1 in connected graph $G$, $G - v$ is connected.

**Proof:**
For $x \neq v, y \neq v \in V$,
there is path between $x$ and $y$ in $G$ since connected.
and does not use $v$ (degree 1)
**Theorem:**
“$G$ connected and has $|V| - 1$ edges” $\equiv$
“$G$ is connected and has no cycles.”

**Lemma:** If $v$ is a degree 1 in connected graph $G$, $G - v$ is connected.

**Proof:**

For $x \neq v, y \neq v \in V$, there is path between $x$ and $y$ in $G$ since connected.
and does not use $v$ (degree 1)
$\implies G - v$ is connected.
Theorem:
“$G$ connected and has $|V| - 1$ edges” $\equiv$
“$G$ is connected and has no cycles.”

Lemma: If $v$ is a degree 1 in connected graph $G$, $G - v$ is connected.
Proof:
For $x \neq v, y \neq v \in V$,
there is path between $x$ and $y$ in $G$ since connected.
and does not use $v$ (degree 1)
$\implies G - v$ is connected.
Theorem: “\(G\) connected and has \(|V| - 1\) edges” ≡ “\(G\) is connected and has no cycles.”

Lemma: If \(v\) is a degree 1 in connected graph \(G\), \(G - v\) is connected.

Proof:
For \(x \neq v, y \neq v \in V\),
there is path between \(x\) and \(y\) in \(G\) since connected.
and does not use \(v\) (degree 1)
\(\implies G - v\) is connected.
Proof of only if.

Thm:
“G connected and has \(|V| - 1\) edges” ≡
“G is connected and has no cycles.”

Proof of $\implies$: By induction on \(|V|\).
Base Case: \(|V| = 1\). 0 = \(|V| - 1\) edges and has no cycles.
Induction Step: Assume for \(G\) with up to \(k\) vertices. Prove for \(k + 1\)
Proof of only if.

Thm:
“$G$ connected and has $|V| - 1$ edges” $\equiv$
“$G$ is connected and has no cycles.”

Proof of $\implies$: By induction on $|V|$.
Base Case: $|V| = 1$. $0 = |V| - 1$ edges and has no cycles.
Induction Step: Assume for $G$ with up to $k$ vertices. Prove for $k + 1$

Claim: There is a degree 1 node.
Thm:
“G connected and has $|V| - 1$ edges” $\equiv$
“G is connected and has no cycles.”

Proof of $\implies$: By induction on $|V|$.
Base Case: $|V| = 1$. $0 = |V| - 1$ edges and has no cycles.
Induction Step: Assume for $G$ with up to $k$ vertices. Prove for $k + 1$
Claim: There is a degree 1 node.
Proof: First, connected $\implies$ every vertex degree $\geq 1$. 
Proof of only if.

**Thm:**
“G connected and has $|V|−1$ edges” $\equiv$
“G is connected and has no cycles.”

**Proof of $\implies$:**
By induction on $|V|$.

**Base Case:** $|V|=1$. $0=|V|−1$ edges and has no cycles.

**Induction Step:** Assume for $G$ with up to $k$ vertices. Prove for $k+1$

**Claim:** There is a degree 1 node.

**Proof:** First, connected $\implies$ every vertex degree $\geq 1$.

Sum of degrees is $2|V|−2$
Proof of only if.

**Thm:**
“G connected and has \(|V| - 1\) edges” ≡
“G is connected and has no cycles.”

**Proof of \(\implies\):** By induction on \(|V|\).
Base Case: \(|V| = 1\). \(0 = |V| - 1\) edges and has no cycles.
Induction Step: Assume for \(G\) with up to \(k\) vertices. Prove for \(k + 1\)

**Claim:** There is a degree 1 node.

**Proof:** First, connected \(\implies\) every vertex degree \(\geq 1\).
- Sum of degrees is \(2|V| - 2\)
- Average degree \(2 - (2/|V|)\)
Proof of only if.

Thm:
“G connected and has \(|V| – 1\) edges” ≡
“G is connected and has no cycles.”

Proof of \(\implies\):  By induction on \(|V|\).

Base Case: \(|V| = 1\). \(0 = |V| – 1\) edges and has no cycles.

Induction Step: Assume for \(G\) with up to \(k\) vertices. Prove for \(k + 1\)

Claim: There is a degree 1 node.

Proof: First, connected \(\implies\) every vertex degree \(\geq 1\).

Sum of degrees is \(2|V| – 2\).

Average degree \(2 – (2/|V|)\)

Not everyone is bigger than average!
Proof of only if.

**Thm:**
“G connected and has $|V| - 1$ edges” \(\equiv\)
“G is connected and has no cycles.”

**Proof of \(\implies\):** By induction on $|V|$.

**Base Case:** $|V| = 1$. $0 = |V| - 1$ edges and has no cycles.

**Induction Step:** Assume for $G$ with up to $k$ vertices. Prove for $k + 1$

**Claim:** There is a degree 1 node.

**Proof:** First, connected \(\implies\) every vertex degree $\geq 1$.

- Sum of degrees is $2|V| - 2$
- Average degree $2 - (2/|V|)$
- Not everyone is bigger than average! □

By degree 1 removal lemma, $G - v$ is connected.
Proof of only if.

Thm:
“G connected and has $|V| - 1$ edges” \(\equiv\)
“G is connected and has no cycles.”

Proof of \(\implies\): By induction on \(|V|\).

Base Case: \(|V| = 1\). \(0 = |V| - 1\) edges and has no cycles.

Induction Step: Assume for \(G\) with up to \(k\) vertices. Prove for \(k + 1\)

Claim: There is a degree 1 node.
Proof: First, connected \(\implies\) every vertex degree \(\geq 1\).

Sum of degrees is \(2|V| - 2\)
Average degree \(2 - (2/|V|)\)
Not everyone is bigger than average!

By degree 1 removal lemma, \(G - v\) is connected.
\(G - v\) has \(|V| - 1\) vertices and \(|V| - 2\) edges so by induction
Proof of only if.

**Thm:**
“G connected and has $|V| - 1$ edges” $\equiv$
“G is connected and has no cycles.”

**Proof of $\implies$:** By induction on $|V|$.
Base Case: $|V| = 1$. 0 = $|V| - 1$ edges and has no cycles.
Induction Step: Assume for $G$ with up to $k$ vertices. Prove for $k + 1$

**Claim:** There is a degree 1 node.

**Proof:** First, connected $\implies$ every vertex degree $\geq 1$.
Sum of degrees is $2|V| - 2$
Average degree $2 - (2/|V|)$
Not everyone is bigger than average!

By degree 1 removal lemma, $G - v$ is connected.
$G - v$ has $|V| - 1$ vertices and $|V| - 2$ edges so by induction $\implies$ no cycle in $G - v$. 
Proof of only if.

Thm: “G connected and has \( |V| - 1 \) edges” \( \equiv \) “G is connected and has no cycles.”

Proof of \( \implies \): By induction on \(|V|\).

Base Case: \(|V| = 1\). \( 0 = |V| - 1 \) edges and has no cycles.

Induction Step: Assume for \( G \) with up to \( k \) vertices. Prove for \( k + 1 \)

Claim: There is a degree 1 node.

Proof: First, connected \( \implies \) every vertex degree \( \geq 1 \).

Sum of degrees is \( 2|V| - 2 \)

Average degree \( 2 - (2/|V|) \)

Not everyone is bigger than average!

By degree 1 removal lemma, \( G - v \) is connected.

\( G - v \) has \( |V| - 1 \) vertices and \( |V| - 2 \) edges so by induction

\( \implies \) no cycle in \( G - v \).

And no cycle in \( G \) since degree 1 cannot participate in cycle.
Proof of only if.

**Thm:**
“$G$ connected and has $|V| - 1$ edges” \(\equiv\) “$G$ is connected and has no cycles.”

**Proof of \(\implies\):** By induction on $|V|$.

**Base Case:** $|V| = 1$. $0 = |V| - 1$ edges and has no cycles.

**Induction Step:** Assume for $G$ with up to $k$ vertices. Prove for $k+1$ vertices.

**Claim:** There is a degree 1 node.

**Proof:** First, connected \(\implies\) every vertex degree $\geq 1$.

- Sum of degrees is $2|V| - 2$
- Average degree $2 - (2/|V|)$
- Not everyone is bigger than average!

By degree 1 removal lemma, $G - v$ is connected.

$G - v$ has $|V| - 1$ vertices and $|V| - 2$ edges so by induction

\(\implies\) no cycle in $G - v$.

And no cycle in $G$ since degree 1 cannot participate in cycle.
Proof of “if part”

**Thm:**
“G is connected and has no cycles” \(\implies\) “G connected and has \(|V| - 1\) edges”

**Proof:**

Can we use the “degree 1” idea again?
Walk from a vertex using untraversed edges and vertices. Until get stuck. Why?
Finitely-many vertices, no cycle!

**Claim:**
Degree 1 vertex.

**Proof of Claim:**
Can’t visit more than once since no cycle. Entered. Didn’t leave. Only one incident edge.
Removing node doesn’t create cycle. New graph is connected. (from our Degree 1 lemma).
By induction \(G - v\) has \(|V| - 2\) edges. \(G\) has one more or \(|V| - 1\) edges.
Proof of “if part”

**Thm:**
“G is connected and has no cycles” $\implies$ “G connected and has $|V| - 1$ edges”

**Proof:** Can we use the “degree 1” idea again?
Proof of “if part”

**Thm:**
“$G$ is connected and has no cycles” $\implies$ “$G$ connected and has $|V| - 1$ edges”

**Proof:** Can we use the “degree 1” idea again?
  Walk from a vertex using untraversed edges and vertices.
Proof of “if part”

**Thm:**
“G is connected and has no cycles” $\implies$ “G connected and has $|V| - 1$ edges”

**Proof:** Can we use the “degree 1” idea again?
  Walk from a vertex using untraversed edges and vertices.
  Until get stuck. Why?
Proof of “if part”

**Thm:**
“$G$ is connected and has no cycles” $\implies$ “$G$ connected and has $|V| - 1$ edges”

**Proof:** Can we use the “degree 1” idea again?
Walk from a vertex using untraversed edges and vertices. Until get stuck. Why? Finitely-many vertices, no cycle!
Proof of “if part”

Thm:
“G is connected and has no cycles” $\implies$ “G connected and has $|V| - 1$ edges”

Proof: Can we use the “degree 1” idea again? 
   Walk from a vertex using untraversed edges and vertices. 
   Until get stuck. Why? Finitely-many vertices, no cycle!

Claim: Degree 1 vertex.
Proof of “if part”

Thm: “G is connected and has no cycles” $\implies$ “G connected and has $|V| - 1$ edges”
Proof: Can we use the “degree 1” idea again?
Walk from a vertex using untraversed edges and vertices.
Until get stuck. Why? Finitely-many vertices, no cycle!
Claim: Degree 1 vertex.
Proof of Claim: Can’t visit more than once since no cycle.
Proof of “if part”

**Thm:**
“$G$ is connected and has no cycles” $\implies$ “$G$ connected and has $|V| - 1$ edges”

**Proof:** Can we use the “degree 1” idea again?
Walk from a vertex using untraversed edges and vertices.
Until get stuck. Why? Finitely-many vertices, no cycle!

**Claim:** Degree 1 vertex.

**Proof of Claim:**
Can’t visit more than once since no cycle.
Entered.
Thm: “G is connected and has no cycles” $\implies$ “G connected and has $|V| - 1$ edges”

Proof: Can we use the “degree 1” idea again?
Walk from a vertex using untraversed edges and vertices.
Until get stuck. Why? Finitely-many vertices, no cycle!

Claim: Degree 1 vertex.

Proof of Claim:
Can’t visit more than once since no cycle.
Entered. Didn’t leave.
**Thm:**
“\(G\) is connected and has no cycles” \(\implies\) “\(G\) connected and has \(|V| - 1\) edges”

**Proof:** Can we use the “degree 1” idea again?
- Walk from a vertex using untraversed edges and vertices.
- Until get stuck. Why? Finitely-many vertices, no cycle!

**Claim:** Degree 1 vertex.

**Proof of Claim:**
- Can’t visit more than once since no cycle.
- Entered. Didn’t leave. Only one incident edge.
Proof of “if part”

**Thm:**
“G is connected and has no cycles” \(\implies\) “G connected and has \(|V|−1\) edges”

**Proof:** Can we use the “degree 1” idea again?
Walk from a vertex using untraversed edges and vertices.
Until get stuck. Why? Finitely-many vertices, no cycle!

**Claim:** Degree 1 vertex.

**Proof of Claim:**
Can’t visit more than once since no cycle.
Entered. Didn’t leave. Only one incident edge.
Removing node doesn’t create cycle.
Proof of “if part”

**Thm:**
“G is connected and has no cycles” $\implies$ “G connected and has $|V| - 1$ edges”

**Proof:** Can we use the “degree 1” idea again?
Walk from a vertex using untraversed edges and vertices.
Until get stuck. Why? Finitely-many vertices, no cycle!

**Claim:** Degree 1 vertex.

**Proof of Claim:**
Can’t visit more than once since no cycle.
Entered. Didn’t leave. Only one incident edge.

Removing node doesn’t create cycle.
New graph is connected. (from our Degree 1 lemma).
Proof of “if part”

**Thm:**
“G is connected and has no cycles” $\implies$ “G connected and has $|V| - 1$ edges”

**Proof:** Can we use the “degree 1” idea again?
Walk from a vertex using untraversed edges and vertices.
Until get stuck. Why? Finitely-many vertices, no cycle!

**Claim:** Degree 1 vertex.

**Proof of Claim:**
Can’t visit more than once since no cycle.
Entered. Didn’t leave. Only one incident edge.
Removing node doesn’t create cycle.
New graph is connected. (from our Degree 1 lemma).
By induction $G - v$ has $|V| - 2$ edges.
Proof of “if part”

**Thm:**
“G is connected and has no cycles” $\implies$ “G connected and has $|V| - 1$ edges”

**Proof:** Can we use the “degree 1” idea again?
Walk from a vertex using untraversed edges and vertices.
Until get stuck. Why? Finitely-many vertices, no cycle!

**Claim:** Degree 1 vertex.

**Proof of Claim:**
Can’t visit more than once since no cycle.
Entered. Didn’t leave. Only one incident edge.

Removing node doesn’t create cycle.
New graph is connected. (from our Degree 1 lemma).
By induction $G - v$ has $|V| - 2$ edges.
$G$ has one more or $|V| - 1$ edges.
**Proof of “if part”**

**Thm:**
“$G$ is connected and has no cycles” $\implies$ “$G$ connected and has $|V| - 1$ edges”

**Proof:** Can we use the “degree 1” idea again?
Walk from a vertex using untraversed edges and vertices.
Until get stuck. Why? Finitely-many vertices, no cycle!

**Claim:** Degree 1 vertex.

**Proof of Claim:**
Can’t visit more than once since no cycle.
Entered. Didn’t leave. Only one incident edge.
Removing node doesn’t create cycle.
New graph is connected. (from our Degree 1 lemma).
By induction $G - v$ has $|V| - 2$ edges.
$G$ has one more or $|V| - 1$ edges.
Hypercubes.

Complete graphs, really well connected!

$V = \binom{V}{n-1}/2$

$G = (V, E)$

$V = \{0, 1\}^n$

$E = \{(x, y) | x$ and $y differ in one bit position.$

$0 \ 1$

$00 \ 10$

$01 \ 11$

$000 \ 010$

$001 \ 011$

$100 \ 110$

$101 \ 111$

$2^n$ vertices.

number of $n$-bit strings!

$n^2 - 1$ edges.

$2^n$ vertices each of degree $n$.

total degree is $n^2$ and half as many edges!
Hypercubes.

Complete graphs, really well connected! Lots of edges. 
\[ |V|(|V| - 1)/2 \]
Hypercubes.

Complete graphs, really well connected! Lots of edges. $|V|(|V| - 1)/2$
Trees, connected, few edges.
Hypercubes.

Complete graphs, really well connected! Lots of edges.
\[ |V|(|V| - 1)/2 \]
Trees, connected, few edges.
\[ (|V| - 1) \]
Hypercubes.

Complete graphs, really well connected! Lots of edges.

$$|V|(|V| - 1)/2$$

Trees, connected, few edges.

$$(|V| - 1)$$
Hypercubes.

Complete graphs, really well connected! Lots of edges.
\[ |V|(|V| - 1)/2 \]
Trees, connected, few edges.
\[ (|V| - 1) \]
Hypercubes.

\[ \begin{align*}
00 & \quad 01 \\
01 & \quad 10 \\
10 & \quad 11 \\
000 & \quad 010 \\
001 & \quad 011 \\
100 & \quad 110 \\
101 & \quad 111 \\
\end{align*} \]

\[ 2^n \text{ vertices.} \]
\[ \text{number of } n \text{-bit strings!} \]
\[ 2^n - 1 \text{ edges.} \]
\[ 2^n \text{ vertices each of degree } n \text{, total degree is } n^2 \text{ and half as many edges!} \]
Hypercubes.

Complete graphs, really well connected! Lots of edges.
\[ |V|(|V| - 1)/2 \]
Trees, connected, few edges.
\[ (|V| - 1) \]

Hypercubes. Well connected.
Hypercubes.

Complete graphs, really well connected! Lots of edges. 
\[ |V|(|V| - 1)/2 \]
Trees, connected, few edges.  
\[ (|V| - 1) \]

Hypercubes. Well connected.  
\[ |V| \log |V| \text{ edges!} \]
Hypercubes.

Complete graphs, really well connected! Lots of edges.
\(|V|(|V| - 1)/2\)

Trees, connected, few edges.
\((|V| - 1)\)

Hypercubes. Well connected. \(|V|\log|V|\) edges!
Also represents bit-strings nicely.
Hypercubes.

Complete graphs, really well connected! Lots of edges.

\[ |V|(|V| - 1)/2 \]

Trees, connected, few edges.

\( (|V| - 1) \)

Hypercubes. Well connected. \(|V| \log |V|\) edges!

Also represents bit-strings nicely.
Hypercubes.

Complete graphs, really well connected! Lots of edges.
\[ |V|(|V| - 1)/2 \]
Trees, connected, few edges.
\[ (|V| - 1) \]

Hypercubes. Well connected. \(|V|\log|V|\) edges!
Also represents bit-strings nicely.

\[ G = (V, E) \]
Hypercubes.

Complete graphs, really well connected! Lots of edges.
\[ |V|(|V| - 1)/2 \]
Trees, connected, few edges.
\[ (|V| - 1) \]

Hypercubes. Well connected. \(|V| \log |V|\) edges!
Also represents bit-strings nicely.

\[ G = (V, E) \]
\[ |V| = \{0, 1\}^n, \]
Hypercubes.

Complete graphs, really well connected! Lots of edges.
\[ |V|(|V| - 1)/2 \]
Trees, connected, few edges.
\[ (|V| - 1) \]

Hypercubes. Well connected. \(|V| \log |V|\) edges!
Also represents bit-strings nicely.

\[ G = (V, E) \]
\[ |V| = \{0, 1\}^n, \]
\[ |E| = \{(x, y)|x \text{ and } y \text{ differ in one bit position.}\} \]
Hypercubes.

Complete graphs, really well connected! Lots of edges.

\[ |V|(|V| - 1)/2 \]

Trees, connected, few edges.

\( (|V| - 1) \)

Hypercubes. Well connected. \(|V| \log |V| \) edges!
Also represents bit-strings nicely.

\[ G = (V, E) \]

\[ |V| = \{0, 1\}^n, \]

\[ |E| = \{ (x, y) | x \text{ and } y \text{ differ in one bit position.} \} \]
Hypercubes.

Complete graphs, really well connected! Lots of edges.
\[ |V|(|V| - 1)/2 \]
Trees, connected, few edges.
\[ (|V| - 1) \]

Hypercubes. Well connected. \(|V| \log |V|\) edges!
Also represents bit-strings nicely.

\[ G = (V, E) \]
\[ |V| = \{0, 1\}^n, \]
\[ |E| = \{(x, y)|x \text{ and } y \text{ differ in one bit position.}\} \]

\[ 0 \quad 1 \]

\[ 00 \quad 01 \quad 10 \quad 11 \]

\[ 000 \quad 001 \quad 010 \quad 011 \]
\[ 100 \quad 101 \quad 110 \quad 111 \]

\[ 2^n \text{ vertices.} \]
Hypercubes.

Complete graphs, really well connected! Lots of edges.

\[ |V|(|V| - 1)/2 \]

Trees, connected, few edges.

\((|V| - 1)\)

Hypercubes. Well connected. \(|V| \log |V|\) edges!

Also represents bit-strings nicely.

\[ G = (V, E) \]

\(|V| = \{0, 1\}^n, \]

\(|E| = \{(x, y) | x \text{ and } y \text{ differ in one bit position.}\} \]

2\(^n\) vertices. number of \(n\)-bit strings!
Hypercubes.

Complete graphs, really well connected! Lots of edges.

\[ |V|(|V| - 1)/2 \]

Trees, connected, few edges.

\((|V| - 1)\)

Hypercubes. Well connected. \(|V|\log|V|\) edges!
Also represents bit-strings nicely.

\[ G = (V, E) \]
\[ |V| = \{0, 1\}^n, \]
\[ |E| = \{(x, y)|x \text{ and } y \text{ differ in one bit position.}\} \]

\[ 2^n \text{ vertices. number of } n\text{-bit strings!} \]
\[ n2^{n-1} \text{ edges.} \]
Hypercubes.

Complete graphs, really well connected! Lots of edges.

\[ |V|(|V| - 1)/2 \]

Trees, connected, few edges.

\((|V| - 1)\)

Hypercubes. Well connected. \(|V| \log |V| \) edges!

Also represents bit-strings nicely.

\[ G = (V, E) \]

\[ |V| = \{0, 1\}^n, \]

\[ |E| = \{(x, y) | x \text{ and } y \text{ differ in one bit position.}\} \]

\[ 2^n \text{ vertices. number of } n\text{-bit strings!} \]

\[ n2^{n-1} \text{ edges.} \]

\[ 2^n \text{ vertices each of degree } n \]
Hypercubes.

Complete graphs, really well connected! Lots of edges. 
\[|V|(|V| - 1)/2\]
Trees, connected, few edges. 
\((|V| - 1)\)

Hypercubes. Well connected. \(|V|\log|V|\) edges!
Also represents bit-strings nicely.

\[G = (V,E)\]
\[|V| = \{0,1\}^n,\]
\[|E| = \{(x,y)|x\ and\ y\ differ\ in\ one\ bit\ position.\}\]

\[2^n\] vertices. number of \(n\)-bit strings!
\[n2^{n-1}\] edges.

\[2^n\] vertices each of degree \(n\)
total degree is \(n2^n\)
Hypercubes.

Complete graphs, really well connected! Lots of edges.

$$|V|(|V| - 1)/2$$

Trees, connected, few edges.

$$(|V| - 1)$$

Hypercubes. Well connected. $|V|\log |V|$ edges!
Also represents bit-strings nicely.

$$G = (V, E)$$
$|V| = \{0, 1\}^n,$
$|E| = \{(x, y)| x and y differ in one bit position.\}$

2	extsuperscript{n} vertices. number of $n$-bit strings!

$n2^{n-1}$ edges.

$2^n$ vertices each of degree $n$

total degree is $n2^n$ and half as many edges!
Hypercubes.

Complete graphs, really well connected! Lots of edges.
\[ |V| (|V| - 1)/2 \]
Trees, connected, few edges.
\[(|V| - 1)\]

Hypercubes. Well connected. \(|V| \log |V| \) edges!
Also represents bit-strings nicely.

\[ G = (V, E) \]
\[ |V| = \{0, 1\}^n, \]
\[ |E| = \{(x, y) | x \text{ and } y \text{ differ in one bit position.}\} \]

\[ 2^n \text{ vertices. number of } n\text{-bit strings!} \]
\[ n2^{n-1} \text{ edges.} \]
\[ 2^n \text{ vertices each of degree } n \]
\[ \text{total degree is } n2^n \text{ and half as many edges!} \]
A 0-dimensional hypercube is a node labelled with the empty string of bits.
Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

An $n$-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n - 1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$. 
Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

An $n$-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$. 
Hypercube: Can’t cut me!

**Thm:** Any subset $S$ of the hypercube where $|S| \leq |V|/2$ has $\geq |S|$ edges connecting it to $V - S$:

$$|E \cap S \times (V - S)| \geq |S|$$

**Terminology:**
- $(S, V - S)$ is cut.
- $(E \cap S \times (V - S))$ - cut edges.

**Restatement:** for any cut in the hypercube, the number of cut edges is at least the size of the small side.
Thm: Any subset $S$ of the hypercube where $|S| \leq |V|/2$ has $\geq |S|$ edges connecting it to $V - S$
Thm: Any subset $S$ of the hypercube where $|S| \leq |V|/2$ has $\geq |S|$ edges connecting it to $V - S$: $|E \cap S \times (V - S)| \geq |S|$
**Thm**: Any subset $S$ of the hypercube where $|S| \leq |V|/2$ has $|S|$ edges connecting it to $V - S$: $|E \cap S \times (V - S)| \geq |S|$

**Terminology**: 

$(S, V - S)$ is cut. $(E \cap S \times (V - S))$ - cut edges. 

Restatement: for any cut in the hypercube, the number of cut edges is at least the size of the small side.
Theorem: Any subset $S$ of the hypercube where $|S| \leq |V|/2$ has $\geq |S|$ edges connecting it to $V - S$: $|E \cap S \times (V - S)| \geq |S|$

Terminology:
$(S, V - S)$ is cut.
**Thm:** Any subset $S$ of the hypercube where $|S| \leq |V|/2$ has $\geq |S|$ edges connecting it to $V - S$: $|E \cap S \times (V - S)| \geq |S|$

**Terminology:**
- $(S, V - S)$ is cut.
- $(E \cap S \times (V - S))$ - cut edges.
**Thm:** Any subset \( S \) of the hypercube where \( |S| \leq |V|/2 \) has \( \geq |S| \) edges connecting it to \( V - S \): \(|E \cap S \times (V - S)| \geq |S|\)

**Terminology:**
- \((S, V - S)\) is cut.
- \((E \cap S \times (V - S))\) - cut edges.
**Thm:** Any subset $S$ of the hypercube where $|S| \leq |V|/2$ has $\geq |S|$ edges connecting it to $V - S$: $|E \cap S \times (V - S)| \geq |S|$

Terminology:

- $(S, V - S)$ is cut.
- $(E \cap S \times (V - S))$ - cut edges.

Restatement: for any cut in the hypercube, the number of cut edges is at least the size of the small side.
Proof of Large Cuts.

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side.

**Proof:**
Proof of Large Cuts.

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side.

**Proof:**

Base Case: \(n = 1\)
Proof of Large Cuts.

**Thm:** For any cut \( (S, V - S) \) in the hypercube, the number of cut edges is at least the size of the small side.

**Proof:**
Base Case: \( n = 1 \ V = \{0, 1\} \).
Proof of Large Cuts.

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side.

**Proof:**
Base Case: \(n = 1\) \(V = \{0, 1\}\).
\(S = \{0\}\) has one edge leaving.
Proof of Large Cuts.

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side.

**Proof:**

Base Case: \(n = 1\) \(V = \{0, 1\}\).
- \(S = \{0\}\) has one edge leaving.
- \(S = \emptyset\) has 0.
Thm: For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side.

Proof:
Base Case: \(n = 1\) \(V = \{0, 1\}\).
- \(S = \{0\}\) has one edge leaving.
- \(S = \emptyset\) has 0.
Proof of Large Cuts.

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side.

**Proof:**
Base Case: \(n = 1\) \(V = \{0, 1\}\).
- \(S = \{0\}\) has one edge leaving.
- \(S = \emptyset\) has 0.
Induction Step Idea

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side.
**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side.

Use recursive definition into two subcubes.
**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side.

Use recursive definition into two subcubes.

Two cubes connected by edges.
Induction Step Idea

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side.

Use recursive definition into two subcubes.

Two cubes connected by edges.

Case 1: Count edges inside subcube inductively.
**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side.

Use recursive definition into two subcubes.

Two cubes connected by edges.

**Case 1:** Count edges inside subcube inductively.
**Thm:** For any cut $(S, V - S)$ in the hypercube, the number of cut edges is at least the size of the small side.

Use recursive definition into two subcubes.

Two cubes connected by edges.

**Case 1:** Count edges inside subcube inductively.

**Case 2:** Count inside and across.
**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side.

Use recursive definition into two subcubes.

Two cubes connected by edges.

Case 1: Count edges inside subcube inductively.

Case 2: Count inside and across.
Thm: For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).
Thm: For any cut ($S, V - S$) in the hypercube, the number of cut edges is at least the size of the small side, $|S|$.

Proof: Induction Step.
Induction Step

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step.
Recursive definition:
**Induction Step**

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof: Induction Step.**

Recursive definition:

\[
H_0 = (V_0, E_0), H_1 = (V_1, E_1), \text{ edges } E_x \text{ that connect them.}
\]
**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof: Induction Step.**

Recursive definition:

\[ H_0 = (V_0, E_0), H_1 = (V_1, E_1), \text{ edges } E_x \text{ that connect them.} \]

\[ H = (V_0 \cup V_1, E_0 \cup E_1 \cup E_x) \]
Induction Step

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step.

Recursive definition:

\[
\begin{align*}
H_0 &= (V_0, E_0), H_1 = (V_1, E_1), \text{ edges } E_x \text{ that connect them,} \\
H &= (V_0 \cup V_1, E_0 \cup E_1 \cup E_x) \\
S &= S_0 \cup S_1 \text{ where } S_0 \text{ in first, and } S_1 \text{ in other.}
\end{align*}
\]
Induction Step

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof: Induction Step.**

Recursive definition:

\[
H_0 = (V_0, E_0), H_1 = (V_1, E_1), \text{ edges } E_x \text{ that connect them.}
\]

\[
H = (V_0 \cup V_1, E_0 \cup E_1 \cup E_x)
\]

\[
S = S_0 \cup S_1 \text{ where } S_0 \text{ in first, and } S_1 \text{ in other.}
\]

**Case 1:** \(|S_0| \leq |V_0|/2, |S_1| \leq |V_1|/2\)
**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof: Induction Step.**

Recursive definition:

\[
H_0 = (V_0, E_0), H_1 = (V_1, E_1), \text{ edges } E_x \text{ that connect them.}
\]

\[
H = (V_0 \cup V_1, E_0 \cup E_1 \cup E_x)
\]

\[
S = S_0 \cup S_1 \text{ where } S_0 \text{ in first, and } S_1 \text{ in other.}
\]

**Case 1:** \(|S_0| \leq |V_0|/2, |S_1| \leq |V_1|/2

Both \(S_0\) and \(S_1\) are small sides.
Thm: For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

Proof: Induction Step.

Recursive definition:

\[
H_0 = (V_0, E_0), H_1 = (V_1, E_1), \text{edges } E_x \text{ that connect them.}
\]

\[
H = (V_0 \cup V_1, E_0 \cup E_1 \cup E_x)
\]

\[
S = S_0 \cup S_1 \text{ where } S_0 \text{ in first, and } S_1 \text{ in other.}
\]

Case 1: \(|S_0| \leq |V_0|/2, |S_1| \leq |V_1|/2\)

Both \(S_0\) and \(S_1\) are small sides. So by induction.
Induction Step

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step.

Recursive definition:

\[
H_0 = (V_0, E_0), H_1 = (V_1, E_1), \text{edges } E_x \text{ that connect them.}
\]

\[
H = (V_0 \cup V_1, E_0 \cup E_1 \cup E_x)
\]

\[
S = S_0 \cup S_1 \text{ where } S_0 \text{ in first, and } S_1 \text{ in other.}
\]

**Case 1:** \(|S_0| \leq |V_0|/2, |S_1| \leq |V_1|/2\)

Both \(S_0\) and \(S_1\) are small sides. So by induction.

Edges cut in \(H_0 \geq |S_0|\).
**Induction Step**

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step.

Recursive definition:

\[ H_0 = (V_0, E_0), H_1 = (V_1, E_1), \text{ edges } E_x \text{ that connect them.} \]

\[ H = (V_0 \cup V_1, E_0 \cup E_1 \cup E_x) \]

\[ S = S_0 \cup S_1 \text{ where } S_0 \text{ in first, and } S_1 \text{ in other.} \]

**Case 1:** \(|S_0| \leq |V_0|/2, |S_1| \leq |V_1|/2\)

Both \(S_0\) and \(S_1\) are small sides. So by induction.

- Edges cut in \(H_0 \geq |S_0|\).
- Edges cut in \(H_1 \geq |S_1|\).
**Induction Step**

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step.

Recursive definition:

\[ H_0 = (V_0, E_0), H_1 = (V_1, E_1), \text{ edges } E_x \text{ that connect them.} \]

\[ H = (V_0 \cup V_1, E_0 \cup E_1 \cup E_x) \]

\[ S = S_0 \cup S_1 \text{ where } S_0 \text{ in first, and } S_1 \text{ in other.} \]

**Case 1:** \(|S_0| \leq |V_0|/2, |S_1| \leq |V_1|/2\)

Both \(S_0\) and \(S_1\) are small sides. So by induction.

Edges cut in \(H_0 \geq |S_0|\).

Edges cut in \(H_1 \geq |S_1|\).
Thm: For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

Proof: Induction Step.
Recursive definition:

\[
\begin{align*}
H_0 &= (V_0, E_0), H_1 = (V_1, E_1), \text{ edges } E_x \text{ that connect them.} \\
H &= (V_0 \cup V_1, E_0 \cup E_1 \cup E_x) \\
S &= S_0 \cup S_1 \text{ where } S_0 \text{ in first, and } S_1 \text{ in other.}
\end{align*}
\]

Case 1: \(|S_0| \leq |V_0|/2, |S_1| \leq |V_1|/2\)
Both \(S_0\) and \(S_1\) are small sides. So by induction.

- Edges cut in \(H_0 \geq |S_0|\).
- Edges cut in \(H_1 \geq |S_1|\).

Total cut edges \(\geq |S_0| + |S_1| = |S|\).
Induction Step

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof: Induction Step.**

Recursive definition:

\[H_0 = (V_0, E_0), H_1 = (V_1, E_1),\] edges \(E_x\) that connect them.

\[H = (V_0 \cup V_1, E_0 \cup E_1 \cup E_x)\]

\[S = S_0 \cup S_1\] where \(S_0\) in first, and \(S_1\) in other.

**Case 1:** \(|S_0| \leq |V_0|/2, |S_1| \leq |V_1|/2\)

Both \(S_0\) and \(S_1\) are small sides. So by induction.

- Edges cut in \(H_0\) \(\geq |S_0|\).
- Edges cut in \(H_1\) \(\geq |S_1|\).

Total cut edges \(\geq |S_0| + |S_1| = |S|\).
Thm: For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

Proof: Induction Step. Case 2. \(|S_0| \geq |V_0|/2\).
Induction Step. Case 2.

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step. Case 2. \(|S_0| \geq |V_0|/2.

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2

\(|S_1| \leq |V_1|/2 \) since \(|S| \leq |V|/2.\)
Induction Step. Case 2.

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step. Case 2. \(|S_0| \geq |V_0|/2\).

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2\)

\(|S_1| \leq |V_1|/2\) since \(|S| \leq |V|/2\).

\(\implies \geq |S_1|\) edges cut in \(E_1\).
Induction Step. Case 2.

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step. Case 2. \(|S_0| \geq |V_0|/2\).

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2\)

\(|S_1| \leq |V_1|/2\) since \(|S| \leq |V|/2\).

\[\implies \geq |S_1| \text{ edges cut in } E_1.\]

\(|S_0| \geq |V_0|/2 \implies |V_0 - S_0| \leq |V_0|/2\)
Induction Step. Case 2.

Thm: For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

Proof: Induction Step. Case 2. \(|S_0| \geq |V_0|/2\).

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2\).
\(|S_1| \leq |V_1|/2\) since \(|S| \leq |V|/2\).
\(\implies \geq |S_1|\) edges cut in \(E_1\).
\(|S_0| \geq |V_0|/2\) \(\implies |V_0 - S_0| \leq |V_0|/2\).
\(\implies \geq |V_0| - |S_0|\) edges cut in \(E_0\).
**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step. Case 2. \(|S_0| \geq |V_0|/2\).

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2\)

\(|S_1| \leq |V_1|/2\) since \(|S| \leq |V|/2\).

\(\implies \geq |S_1|\) edges cut in \(E_1\).

\(|S_0| \geq |V_0|/2 \implies |V_0 - S_0| \leq |V_0|/2\)

\(\implies \geq |V_0| - |S_0|\) edges cut in \(E_0\).

Edges in \(E_x\) connect corresponding nodes.
Induction Step. Case 2.

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step. Case 2. \(|S_0| \geq |V_0|/2\).

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2 \quad |S_1| \leq |V_1|/2\) since \(|S| \leq |V|/2\).

\[\Rightarrow \geq |S_1| \text{ edges cut in } E_1.\]

\[|S_0| \geq |V_0|/2 \Rightarrow |V_0 - S_0| \leq |V_0|/2 \Rightarrow \geq |V_0| - |S_0| \text{ edges cut in } E_0.\]

Edges in \(E_x\) connect corresponding nodes.

\[\Rightarrow = |S_0| - |S_1| \text{ edges cut in } E_x.\]
Induction Step. Case 2.

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof: Induction Step. Case 2.** \(|S_0| \geq |V_0|/2\).

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2\)

\(|S_1| \leq |V_1|/2\) since \(|S| \leq |V|/2\).

\[\implies \geq |S_1| \text{ edges cut in } E_1.\]

\[|S_0| \geq |V_0|/2 \implies |V_0 - S_0| \leq |V_0|/2\]

\[\implies \geq |V_0| - |S_0| \text{ edges cut in } E_0.\]

Edges in \(E_x\) connect corresponding nodes.

\[\implies = |S_0| - |S_1| \text{ edges cut in } E_x.\]
Thm: For any cut $(S, V - S)$ in the hypercube, the number of cut edges is at least the size of the small side, $|S|$.

Proof: Induction Step. Case 2. $|S_0| \geq |V_0|/2$.

Recall Case 1: $|S_0|, |S_1| \leq |V|/2$

$|S_1| \leq |V_1|/2$ since $|S| \leq |V|/2$.

$\implies \geq |S_1|$ edges cut in $E_1$.

$|S_0| \geq |V_0|/2 \implies |V_0 - S_0| \leq |V_0|/2$

$\implies \geq |V_0| - |S_0|$ edges cut in $E_0$.

Edges in $E_x$ connect corresponding nodes.

$\implies = |S_0| - |S_1|$ edges cut in $E_x$.

Total edges cut:
**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step. Case 2. \(|S_0| \geq |V_0|/2\).

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2\)
\(|S_1| \leq |V_1|/2\) since \(|S| \leq |V|/2\).

\[\implies \geq |S_1| \text{ edges cut in } E_1.\]

\[|S_0| \geq |V_0|/2 \implies |V_0 - S_0| \leq |V_0|/2\]
\[\implies \geq |V_0| - |S_0| \text{ edges cut in } E_0.\]

Edges in \(E_x\) connect corresponding nodes.
\[\implies = |S_0| - |S_1| \text{ edges cut in } E_x.\]

Total edges cut:
\[\geq\]
**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** **Induction Step. Case 2.** \(|S_0| \geq |V_0|/2.

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2

\(|S_1| \leq |V_1|/2\) since \(|S| \leq |V|/2.

\(\implies \geq |S_1|\) edges cut in \(E_1\).

\(|S_0| \geq |V_0|/2 \implies |V_0 - S_0| \leq |V_0|/2

\(\implies \geq |V_0| - |S_0|\) edges cut in \(E_0\).

Edges in \(E_x\) connect corresponding nodes.

\(\implies = |S_0| - |S_1|\) edges cut in \(E_x\).

Total edges cut:

\(\geq |S_1|\)
**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step. Case 2. \(|S_0| \geq |V_0|/2\).

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2\)

\(|S_1| \leq |V_1|/2\) since \(|S| \leq |V|/2\).

\[\implies \geq |S_1| \text{ edges cut in } E_1.\]

\(|S_0| \geq |V_0|/2 \implies |V_0 - S_0| \leq |V_0|/2\)

\[\implies \geq |V_0| - |S_0| \text{ edges cut in } E_0.\]

Edges in \(E_x\) connect corresponding nodes.

\[\implies = |S_0| - |S_1| \text{ edges cut in } E_x.\]

Total edges cut:

\[\geq |S_1| + |V_0| - |S_0|\]
**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step. Case 2. \(|S_0| \geq |V_0|/2.\)

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2\)
\(|S_1| \leq |V_1|/2\) since \(|S| \leq |V|/2.\)
\[\implies \geq |S_1|\] edges cut in \(E_1.\)
\(|S_0| \geq |V_0|/2 \implies |V_0 - S_0| \leq |V_0|/2\)
\[\implies \geq |V_0| - |S_0|\] edges cut in \(E_0.\)

Edges in \(E_x\) connect corresponding nodes.
\[\implies = |S_0| - |S_1|\] edges cut in \(E_x.\)

Total edges cut:
\[\geq |S_1| + |V_0| - |S_0| + |S_0| - |S_1|\]
Induction Step. Case 2.

**Thm:** For any cut \((S, V – S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step. Case 2. \(|S_0| \geq |V_0|/2.\)

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2\)

\(|S_1| \leq |V_1|/2\) since \(|S| \leq |V|/2.\)

\[\implies \geq |S_1|\] edges cut in \(E_1.\)

\(|S_0| \geq |V_0|/2 \implies |V_0 – S_0| \leq |V_0|/2\)

\[\implies \geq |V_0| – |S_0|\] edges cut in \(E_0.\)

Edges in \(E_x\) connect corresponding nodes.

\[\implies = |S_0| – |S_1|\] edges cut in \(E_x.\)

Total edges cut:

\[\geq |S_1| + |V_0| – |S_0| + |S_0| – |S_1| = |V_0|\]
Induction Step. Case 2.

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step. Case 2. \(|S_0| \geq |V_0|/2\).

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2\)
\(|S_1| \leq |V_1|/2\) since \(|S| \leq |V|/2\).
\[\implies \geq |S_1| \text{ edges cut in } E_1.\]
\[|S_0| \geq |V_0|/2 \implies |V_0 - S_0| \leq |V_0|/2\]
\[\implies \geq |V_0| - |S_0| \text{ edges cut in } E_0.\]

Edges in \(E_x\) connect corresponding nodes.
\[\implies = |S_0| - |S_1| \text{ edges cut in } E_x.\]

Total edges cut:
\[\geq |S_1| + |V_0| - |S_0| + |S_0| - |S_1| = |V_0| \]
\[|V_0|\]
**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step. Case 2. \(|S_0| \geq |V_0|/2\).

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2\)
\(|S_1| \leq |V_1|/2\) since \(|S| \leq |V|/2\).
\[\implies \geq |S_1| \text{ edges cut in } E_1.\]
\[|S_0| \geq |V_0|/2 \implies |V_0 - S_0| \leq |V_0|/2\]
\[\implies \geq |V_0| - |S_0| \text{ edges cut in } E_0.\]

Edges in \(E_x\) connect corresponding nodes.
\[\implies = |S_0| - |S_1| \text{ edges cut in } E_x.\]

Total edges cut:
\[\geq |S_1| + |V_0| - |S_0| + |S_0| - |S_1| = |V_0| \]
\[|V_0| = |V|/2 \geq |S|.\]
Induction Step. Case 2.

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof: Induction Step. Case 2.** \(|S_0| \geq |V_0|/2.\)

Recall Case 1: \(|S_0|, |S_1| \leq |V|/2 \)
\(|S_1| \leq |V_1|/2 \) since \(|S| \leq |V|/2.\)

\[ \Rightarrow \geq |S_1| \text{ edges cut in } E_1. \]
\[ |S_0| \geq |V_0|/2 \Rightarrow |V_0 - S_0| \leq |V_0|/2 \]

\[ \Rightarrow \geq |V_0| - |S_0| \text{ edges cut in } E_0. \]

Edges in \(E_x\) connect corresponding nodes.

\[ \Rightarrow = |S_0| - |S_1| \text{ edges cut in } E_x. \]

**Total edges cut:**
\[ \geq |S_1| + |V_0| - |S_0| + |S_0| - |S_1| = |V_0| \]
\[ |V_0| = |V|/2 \geq |S|. \]

Also, case 3 where \(|S_1| \geq |V|/2\) is symmetric.
The cuts in the hypercubes are exactly the transitions from 0 sets to 1 set on boolean functions on \(\{0, 1\}^n\).
The cuts in the hypercubes are exactly the transitions from 0 sets to 1 set on boolean functions on \( \{0, 1\}^n \).

Central area of study in computer science!
Hypercubes and Boolean Functions.

The cuts in the hypercubes are exactly the transitions from 0 sets to 1 set on boolean functions on \( \{0, 1\}^n \).

Central area of study in computer science!

Yes/No Computer Programs \( \equiv \) Boolean function on \( \{0, 1\}^n \)
The cuts in the hypercubes are exactly the transitions from 0 sets to 1 set on boolean functions on \( \{0, 1\}^n \).

Central area of study in computer science!

Yes/No Computer Programs \( \equiv \) Boolean function on \( \{0, 1\}^n \)

Central object of study.