Today.

Types of graphs.

Today.

Types of graphs.

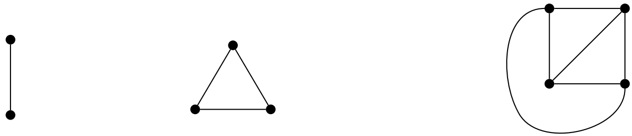
- Complete Graphs.
- Trees.
- Hypercubes.

Today.

Types of graphs.

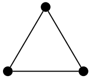
- Complete Graphs.
- Trees.
- Hypercubes.

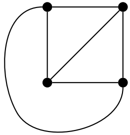
Complete Graph.



K_n complete graph on n vertices.

Complete Graph.

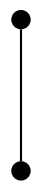


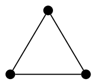


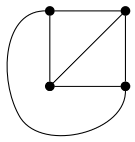
K_n complete graph on n vertices.

All edges are present.

Complete Graph.





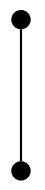


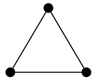
K_n complete graph on n vertices.

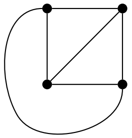
All edges are present.

Everyone is my neighbor.

Complete Graph.







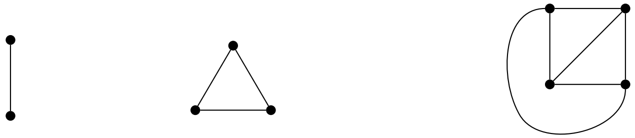
K_n complete graph on n vertices.

All edges are present.

Everyone is my neighbor.

Each vertex is adjacent to every other vertex.

Complete Graph.



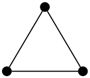
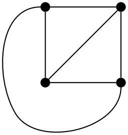
K_n complete graph on n vertices.

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.

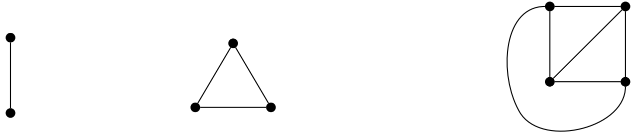
All edges are present.

Everyone is my neighbor.

Each vertex is adjacent to every other vertex.

How many edges?

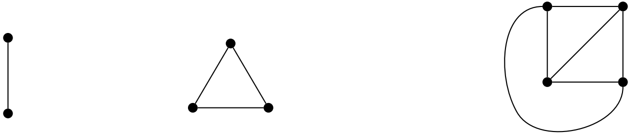
Complete Graph.



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.

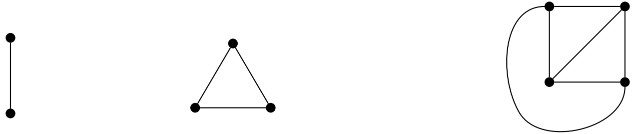
Each vertex is adjacent to every other vertex.

How many edges?

Each vertex is incident to $n-1$ edges.

Sum of degrees is $n(n-1)$.

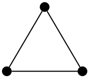
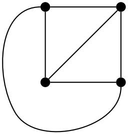
Complete Graph.



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?

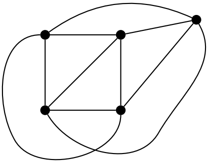
Each vertex is incident to $n-1$ edges.

Sum of degrees is $n(n-1)$.

$\implies$ Number of edges is $n(n-1)/2$.

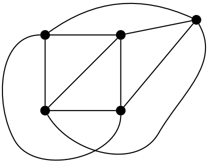
Remember sum of degree is $2|E|$.

K_4 and K_5



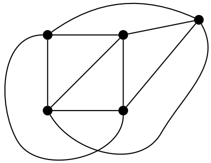
K_5 is not planar.

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

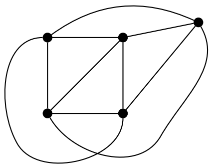


K_5 is not planar.

Cannot be drawn in the plane without an edge crossing!

Prove it!

K_4 and K_5



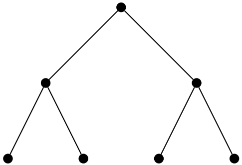
K_5 is not planar.

Cannot be drawn in the plane without an edge crossing!

Prove it! Read Note 5!!

Trees!

Graph $G = (V, E)$.
Binary Tree!



More generally.

Trees: Definitions

Definitions:

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:

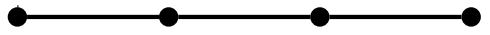
A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
A connected graph where any edge addition creates a cycle.

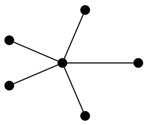
Trees: Definitions

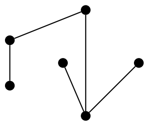
Definitions:

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.







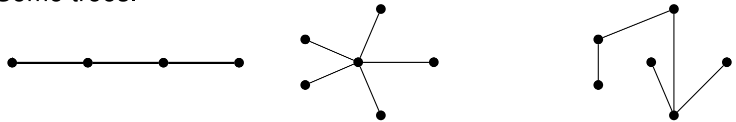
no cycle and connected?

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.



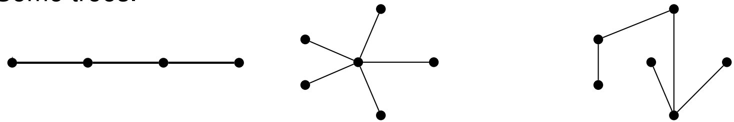
no cycle and connected? Yes.

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.



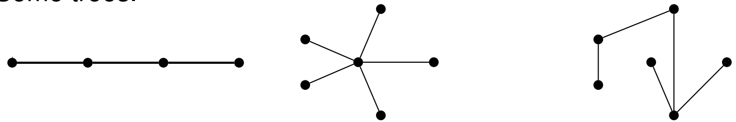
no cycle and connected? Yes.
$|V|-1$ edges and connected?

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.



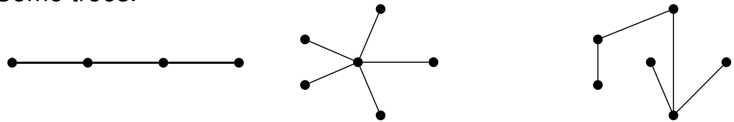
no cycle and connected? Yes.
$|V|-1$ edges and connected? Yes.

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.



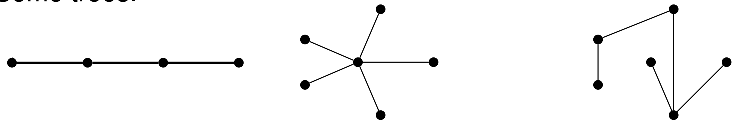
no cycle and connected? Yes.
$|V|-1$ edges and connected? Yes.
removing any edge disconnects it.

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.



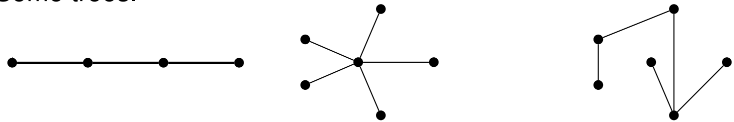
no cycle and connected? Yes.
$|V|-1$ edges and connected? Yes.
removing any edge disconnects it. Harder to check.

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.



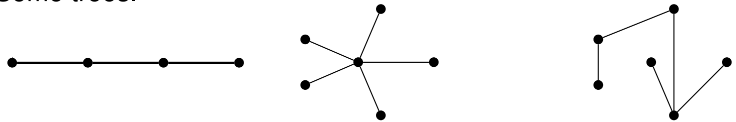
no cycle and connected? Yes.
$|V|-1$ edges and connected? Yes.
removing any edge disconnects it. Harder to check. but yes.

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.



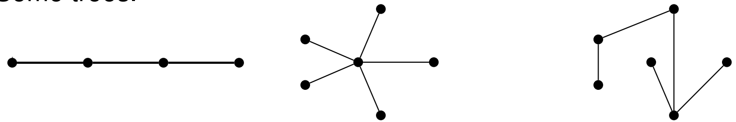
no cycle and connected? Yes.
$|V|-1$ edges and connected? Yes.
removing any edge disconnects it. Harder to check. but yes.
Adding any edge creates cycle.

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.



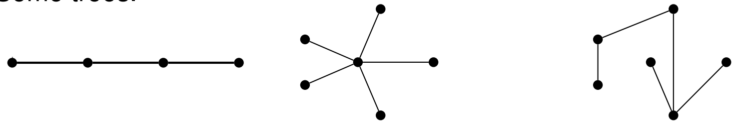
no cycle and connected? Yes.
$|V| - 1$ edges and connected? Yes.
removing any edge disconnects it. Harder to check. but yes.
Adding any edge creates cycle. Harder to check.

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.



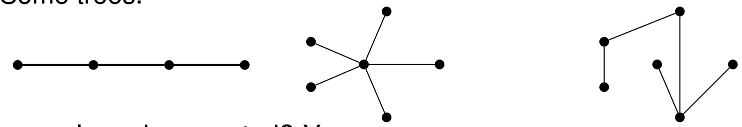
no cycle and connected? Yes.
$|V| - 1$ edges and connected? Yes.
removing any edge disconnects it. Harder to check. but yes.
Adding any edge creates cycle. Harder to check. but yes.

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.



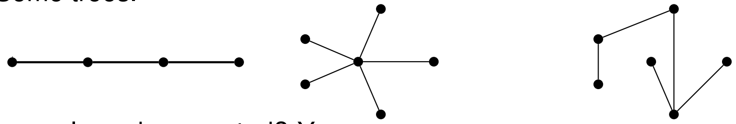
no cycle and connected? Yes.
$|V|-1$ edges and connected? Yes.
removing any edge disconnects it. Harder to check. but yes.
Adding any edge creates cycle. Harder to check. but yes.

Trees: Definitions

Definitions:

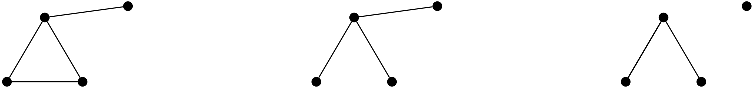
A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
A connected graph where any edge addition creates a cycle.

Some trees.



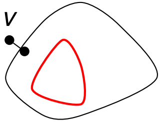
no cycle and connected? Yes.
$|V| - 1$ edges and connected? Yes.
removing any edge disconnects it. Harder to check. but yes.
Adding any edge creates cycle. Harder to check. but yes.

Tree or not tree!



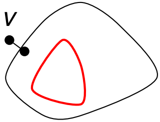
Equivalence of Definitions

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



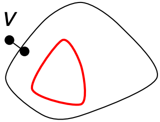
Equivalence of Definitions

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



Proof of $\implies$ (only if):

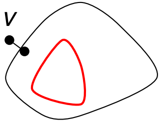
Equivalence of Definitions



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

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

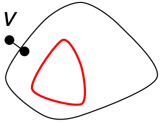
Equivalence of Definitions



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.

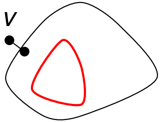
Equivalence of Definitions



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.

Equivalence of Definitions

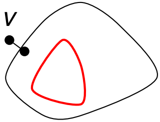


Thm:
"G connected and has $|V|-1$ edges" $\equiv$
"G is connected and has no cycles."

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

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

Equivalence of Definitions

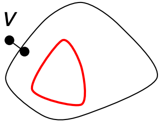


Thm:
"G connected and has $|V|-1$ edges" $\equiv$
"G is connected and has no cycles."

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

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?

Equivalence of Definitions



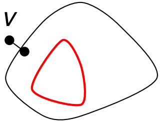
Thm:
"G connected and has $|V|-1$ edges" $\equiv$
"G is connected and has no cycles."

Proof of $\Longrightarrow$ (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

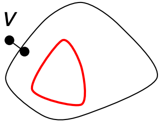
Thm:
"G connected and has $|V|-1$ edges" $\equiv$
"G is connected and has no cycles."



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?

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.

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?

Equivalence of Definitions: Useful Lemma

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 $x \neq v, y \neq v \in V$,

Equivalence of Definitions: Useful Lemma

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" $\equiv$
"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)

Equivalence of Definitions: Useful Lemma

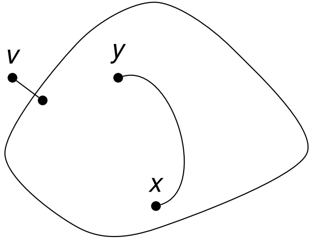
Theorem:
"G connected and has $|V|-1$ edges" $\equiv$
"G is connected and has no cycles."

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.

Equivalence of Definitions: Useful Lemma

Theorem:
"G connected and has $|V|-1$ edges" $\equiv$
"G is connected and has no cycles."

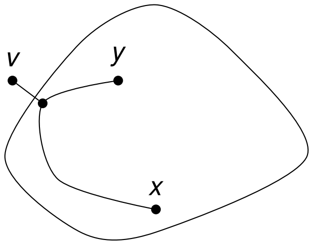
Lemma: If v is a degree 1 in connected graph G, $G-v$ is connected.
Proof:
For $x \neq v, y \neq v \in V$,
there is path between x and y in G since connected.
and does not use v (degree 1)
$\implies$ $G-v$ is connected. $\square$



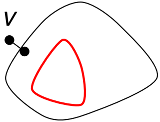
Equivalence of Definitions: Useful Lemma

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. $\square$



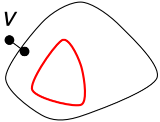
Proof of only if.



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 $k+1$

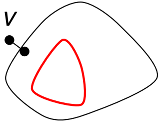
Proof of only if.



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.

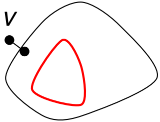
Proof of only if.



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 ≥ 1.

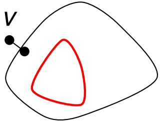
Proof of only if.



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 ≥ 1.
Sum of degrees is $2|V|-2$

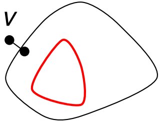
Proof of only if.



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 ≥ 1.
Sum of degrees is $2|V|-2$
Average degree $2-(2/|V|)$

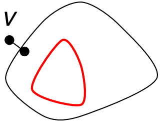
Proof of only if.



Thm:
"G connected and has $|V|-1$ edges" $\equiv$
"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 ≥ 1.
Sum of degrees is $2|V|-2$
Average degree $2-(2/|V|)$
Not everyone is bigger than average! $\square$

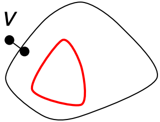
Proof of only if.



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 ≥ 1.
Sum of degrees is $2|V|-2$
Average degree $2-(2/|V|)$
Not everyone is bigger than average! $\square$
By degree 1 removal lemma, $G-v$ is connected.

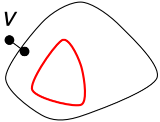
Proof of only if.



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 ≥ 1.
Sum of degrees is $2|V|-2$
Average degree $2-(2/|V|)$
Not everyone is bigger than average! $\square$
By degree 1 removal lemma, $G-v$ is connected.
$G-v$ has $|V|-1$ vertices and $|V|-2$ edges so by induction

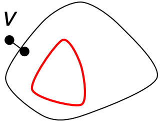
Proof of only if.



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 ≥ 1.
Sum of degrees is $2|V|-2$
Average degree $2-(2/|V|)$
Not everyone is bigger than average! $\square$
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 $G-v$.

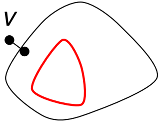
Proof of only if.



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 ≥ 1.
Sum of degrees is $2|V|-2$
Average degree $2-(2/|V|)$
Not everyone is bigger than average! $\square$
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 $k+1$
Claim: There is a degree 1 node.
Proof: First, connected $\implies$ every vertex degree ≥ 1.
Sum of degrees is $2|V|-2$
Average degree $2-(2/|V|)$
Not everyone is bigger than average! $\square$
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. $\square$

Proof of “if part”

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

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” $\Longrightarrow$ “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” $\Longrightarrow$ “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” $\Longrightarrow$ “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.

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.

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.

Proof of "if part"

Thm:
"G is connected and has no cycles" $\Longrightarrow$ "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. $\square$
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. $\square$
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. $\square$
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. $\square$
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. $\square$

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. $\square$

Hypercubes.

Complete graphs, really well connected!

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.

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. $|V|\log|V|$ 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.

$G=(V,E)$

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

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

Hypercubes.

Complete graphs, really well connected! Lots of edges.

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

Trees, connected, few edges.

$(|V|-1)$

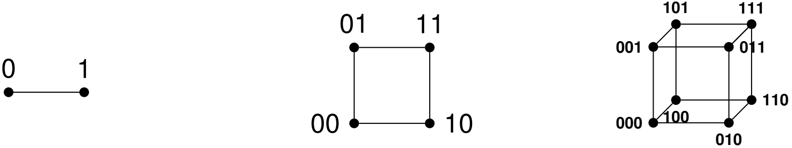
Hypercubes. Well connected. $|V|\log|V|$ edges!

Also represents bit-strings nicely.

$G = (V,E)$

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

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



Hypercubes.

Complete graphs, really well connected! Lots of edges.

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

Trees, connected, few edges.

$(|V|-1)$

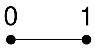
Hypercubes. Well connected. $|V|\log|V|$ edges!

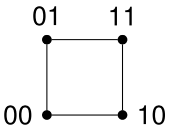
Also represents bit-strings nicely.

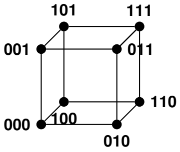
$G=(V,E)$

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

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







2^n vertices.

Hypercubes.

Complete graphs, really well connected! Lots of edges.

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

Trees, connected, few edges.

$(|V|-1)$

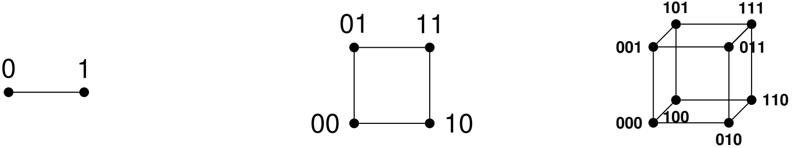
Hypercubes. Well connected. $|V|\log|V|$ edges!

Also represents bit-strings nicely.

$G = (V, E)$

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

$|E| = \{(x,y)|x$ and y 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.

$(|V|-1)$

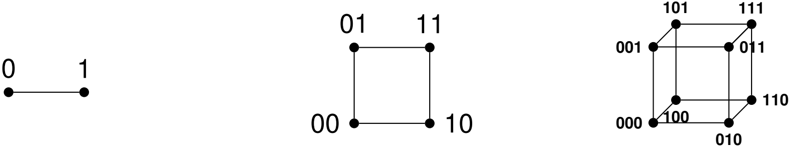
Hypercubes. Well connected. $|V|\log|V|$ edges!

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 vertices. number of n-bit strings!

$n2^{n-1}$ edges.

Hypercubes.

Complete graphs, really well connected! Lots of edges.

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

Trees, connected, few edges.

$(|V|-1)$

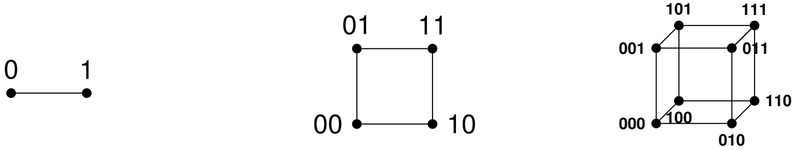
Hypercubes. Well connected. $|V|\log|V|$ edges!

Also represents bit-strings nicely.

$G=(V,E)$

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

$|E|=\{(x,y)|x$ 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

Hypercubes.

Complete graphs, really well connected! Lots of edges.

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

Trees, connected, few edges.

$(|V|-1)$

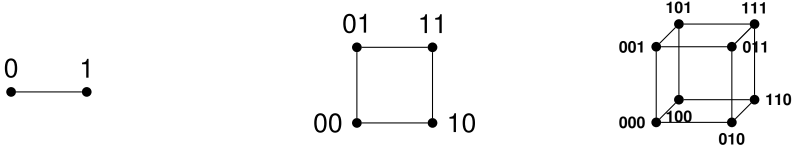
Hypercubes. Well connected. $|V|\log|V|$ edges!

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.

$(|V|-1)$

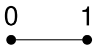
Hypercubes. Well connected. $|V|\log|V|$ edges!

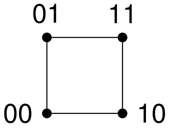
Also represents bit-strings nicely.

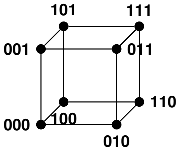
$G=(V,E)$

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

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







2^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.

$(|V|-1)$

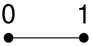
Hypercubes. Well connected. $|V|\log|V|$ edges!

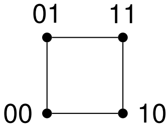
Also represents bit-strings nicely.

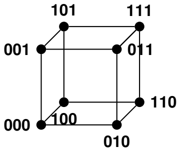
$G=(V,E)$

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

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







2^n 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!

Recursive Definition.

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

Recursive Definition.

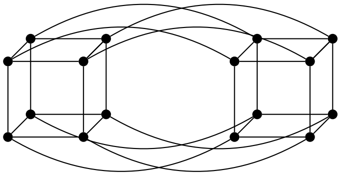
A 0-dimensional hypercube is a node labelled with the empty string of bits.

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!

Hypercube: Can't cut me!

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

Hypercube: Can't cut me!

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

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:

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

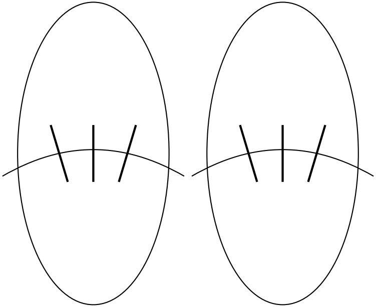
Induction Step Idea

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.



Induction Step Idea

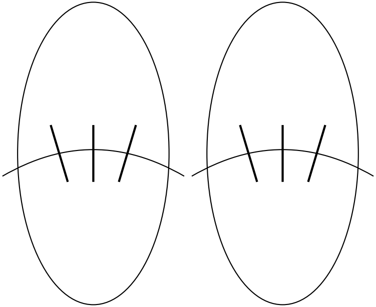
Thm: For any cut $(S, V - S)$ in the hypercube, the number of cut edges is at least the size of the small side.

Use recursive definition into two subcubes.

Two cubes connected by edges.

Case 1: Count edges inside subcube inductively.

Case 2: Count inside and across.



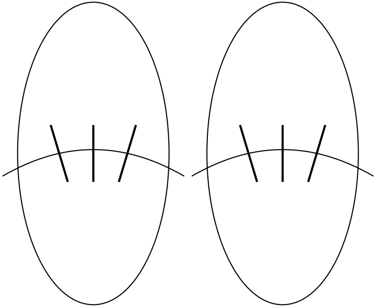
Induction Step Idea

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

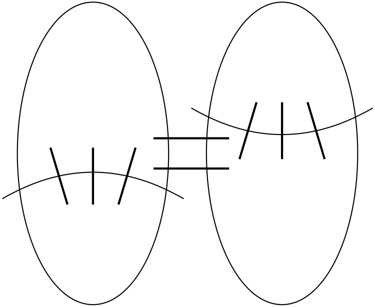
Use recursive definition into two subcubes.

Two cubes connected by edges.

Case 1: Count edges inside subcube inductively.



Case 2: Count inside and across.



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, $|S|$.

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.

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:

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

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.
$H = (V_0 \cup V_1, E_0 \cup E_1 \cup E_x)$

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.

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: $|S_0| \leq |V_0|/2, |S_1| \leq |V_1|/2$

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.

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.

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: $|S_0| \le |V_0|/2, |S_1| \le |V_1|/2$
Both S_0 and S_1 are small sides. So by induction.
Edges cut in $H_0 \ge |S_0|$.

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:

$H_0 = (V_0, E_0)$,$H_1 = (V_1, E_1)$, 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 $H_1 \geq |S_1|$.

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 $H_1 \geq |S_1|$.

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: $|S_0| \le |V_0|/2, |S_1| \le |V_1|/2$
Both S_0 and S_1 are small sides. So by induction.
Edges cut in $H_0 \ge |S_0|$.
Edges cut in $H_1 \ge |S_1|$.

Total cut edges $\ge |S_0| + |S_1| = |S|$.

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 $\geq |S_0| + |S_1| = |S|$. $\square$

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. $|S_0| \geq |V_0|/2$.

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 $|S| \leq |V|/2$.

$\implies \geq |S_1|$ 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.

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.

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.

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.

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:

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:

$\geq$

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:

$\geq \ |S_1|$

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:

$\geq \ |S_1| + |V_0| - |S_0|$

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:
$\geq \ |S_1| + |V_0| - |S_0| + |S_0| - |S_1|$

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 $|S| \leq |V|/2$.
$\implies \geq |S_1|$ 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:
$\geq \; |S_1| + |V_0| - |S_0| + |S_0| - |S_1| = |V_0|$
$|V_0|$

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 $|S| \leq |V|/2$.

$\implies \geq |S_1|$ 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:

$\geq \; |S_1| + |V_0| - |S_0| + |S_0| - |S_1| = |V_0|$

$|V_0| = |V|/2 \geq |S|$.

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 $|S| \leq |V|/2$.

$\implies \geq |S_1|$ 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:

$\geq \ |S_1| + |V_0| - |S_0| + |S_0| - |S_1| = |V_0|$

$|V_0| = |V|/2 \geq |S|$. $\square$

Also, case 3 where $|S_1| \geq |V|/2$ is symmetric.

Hypercubes and Boolean Functions.

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

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!

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 $\{0,1\}^n$

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 $\{0,1\}^n$

Central object of study.