**Planar non-planar**

A finite graph is planar iff it does not contain a subgraph that is (a subdivision of) $K_5$ or $K_{3,3}$.

**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.

**Equivalence of Definitions.**

**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 $x \neq y \neq z \in V$, there is path between $x$ and $y$ in $G$ since connected, and does not use $v$ (degree 1) $\implies G - v$ is connected.

By induction on $|V| - 1$ 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.

**Trees.**

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.
**Proof of if**

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

**Proof:**
Walk from a vertex using untraversed edges.
Until get stuck.

**Claim:**
Must stuck at a degree 1 vertex.

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

**Tree’s fall apart.**

**Thm:**
Can always find a node such that the largest connected component we get by removing it has size at most \(|V|/2\)

**Idea of proof.**
Point edge toward bigger side.
Remove center node.

**Hypercubes.**

Complete graphs, really connected! But lots of edges.
\(|V|(|V|−1)/2\)
Trees, But few edges. \((|V|−1)\)
just falls apart!

**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 \(V−S\); \(|E∩S×(V−S)|\geq|S|\)

**Terminology:**
\((S, V−S)\) is cut.
a partition of the vertices of a graph into two disjoint subsets.
\((E∩S×(V−S))\) - 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:**
Base Case: \(n=1\) \(V=\{0,1\}\).
**Case 2: Count inside and across.**

Yes

Yes

How many colors do we need?

2!

just look at one first

So by induction.

**Bipartite?**

Which of the following graphs are bipartite?

No

Yes

No

A graph is a bipartite graph if and only if it does not contain any odd-length cycles.

**Proof**

Only if: trivial

Start at a node \( v \) in one part, say \( V \), the cycle must be like leaving \( V \), entering \( V \), . . . Also the cycle must end at \( v \), so the cycle must end with “entering \( V \)” All paired up, even length.

No odd-length cycle \( \implies \) bipartite:

Different connected components does not influence each other, just look at one first

Pick one arbitrary vertex \( v \), split all vertices into two groups

**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.

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

**Case 2:** Count inside and across.

Recursive definition:

\[ H_0 = (V_0, E_0), H_1 = (V_1, E_1) \]

edges \( E_0 \) that connect them.

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

\( S = S_0 \cup S_1 \) 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 \( \geq |S_0| + |S_1| = |S| \).

**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. Case 2.

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

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

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

\(|S_1| \leq |V_1| - |S_1| \leq |V_1|/2\)

Edges in \( E_0 \) connect corresponding nodes.

\(|S_0| \leq |S_1| \leq |V_1|/2\)

Total cut edges:

\(|S_0| + |S_1| + |V_0| - |S_0| + |S_0| - |S_1| = |V_0|\)

\(|V_0| = |V|/2 \geq |S|\)

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

**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, \(|S|\).

**Use recursive definition into two subcubes.**

Two cubes connected by edges.

Case 1: Count edges inside subcube inductively.

Case 2: Count inside and across.
What have we done?!

Graphs!

- Eulerian tour: DNA sequence reconstructing
- Coloring: Cellular tower frequency assignment
- Trees: Immense applications

Modeling reality:

- Internet? Giant directed graph
- Dark net? A separate connected component

......