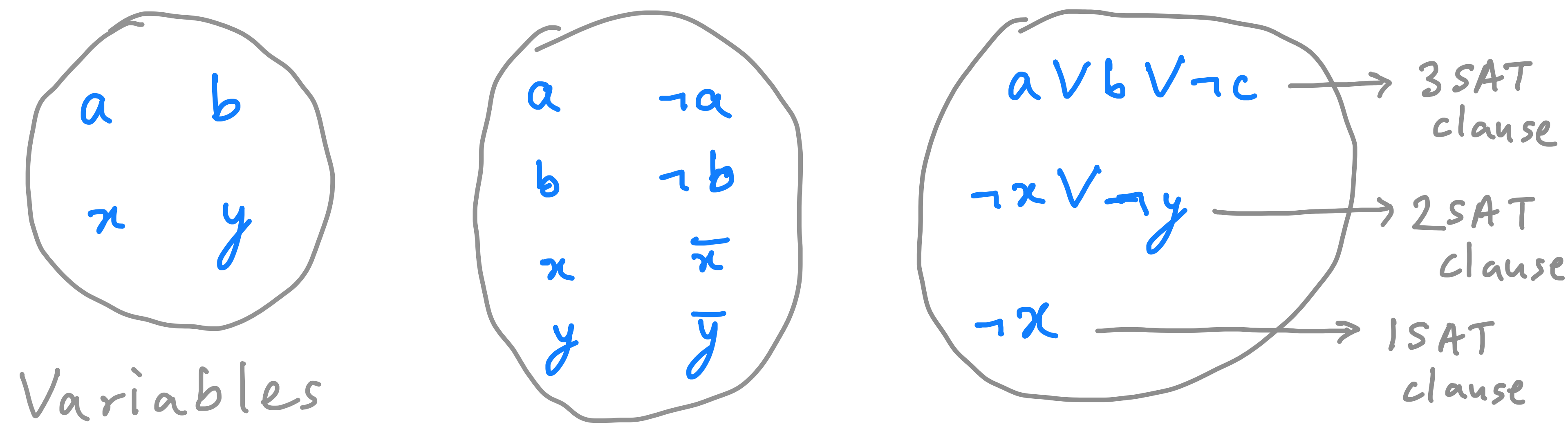
Graphsat + other decision problems

Recall SAT : This asks if we can find a truth assignment that can satisfy a boolean formula.



Variables (Domain = $\top/\bot$)   Literals   Clauses

SAT : $(a \vee b \vee \neg c) \wedge (b \vee \neg c) \wedge d$

3SAT: $(a \vee b \vee \neg c) \wedge (b \vee \neg c \vee d) \wedge (a \vee b \vee d)$

2SAT: $(a \vee b) \wedge (b \vee \neg c)$

If a satifying assignment exists, then we call the formula Satisfiable. Else call it Unsatisfiable.

Facts of life:

1. $2SAT \in P$. If our 2SAT instance has n variables then we can find a satisfying assignment in $p(n)$ time, where p is some polynomial. And if we know the assignment, then we can check that it is a satisfying assignment in polynomial time.

2. $3SAT \in NP$. We can check a satisfying assignment in poly. time. But to find? Exponential time.

Bruteforce strategy:

Given (x : Bformula), $\exists$ (a : Assignment) such that $assign(a, x) = T$

$\hookrightarrow$ This results in 2^n assignments being checked.

③ 3SAT $\notin$ co-NP. If a formula is satisfiable, we can check in poly. time given an assignment. If a formula is unsatisfiable, checking that will take exponential time.

Intuition:

① Solving analysis assignment: HARD

② Checking assignment if solution is correct: EASY

$$① + ② \Rightarrow \in NP$$

③ Checking assignment if solution is incorrect: HARD

$$① + ③ \Rightarrow \notin \text{co-NP}$$

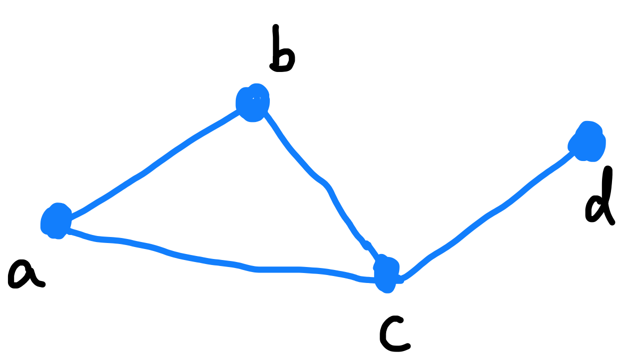
④ kSAT for $k \geq 4$ is not interesting.

$$kSAT \xrightarrow[\text{algorithm}]{\text{poly. time}} 3SAT$$

$2SAT \in P$ | Not sure why this boundary exists! | $3SAT \in NP$

$$SAT(x) := \exists\, a,\quad assign(a, x) = T$$

$$GraphSAT(g) := \forall\, x \in g,\ \exists\, a,\quad assign(a, x) = T$$



$\leftrightsquigarrow$

$$(a \vee b) \wedge (b \vee c) \wedge (a \vee c) \wedge (c \vee d)$$
$$(a \vee b) \wedge (b \vee c) \wedge (a \vee c) \wedge (c \vee \bar{d})$$
$$(a \vee b) \wedge (b \vee c) \wedge (a \vee c) \wedge (\bar{c} \vee d)$$
$$(a \vee b) \wedge (b \vee c) \wedge (a \vee c) \wedge (\bar{c} \vee \bar{d})$$
$$(a \vee b) \wedge (b \vee c) \wedge (a \vee \bar{c}) \wedge (c \vee d)$$
$$\vdots$$
$$(\bar{a} \vee \bar{b}) \wedge (\bar{b} \vee \bar{c}) \wedge (\bar{a} \vee \bar{c}) \wedge (\bar{c} \vee \bar{d})$$

5 edges $\leftrightarrow$ 4^5 formulas.

A graph g is SATISFIABLE if every formula $x \in g$ is SATISFIABLE.

A graph g is UNSATISFIABLE if any formula $x \in g$ is UNSATISFIABLE.

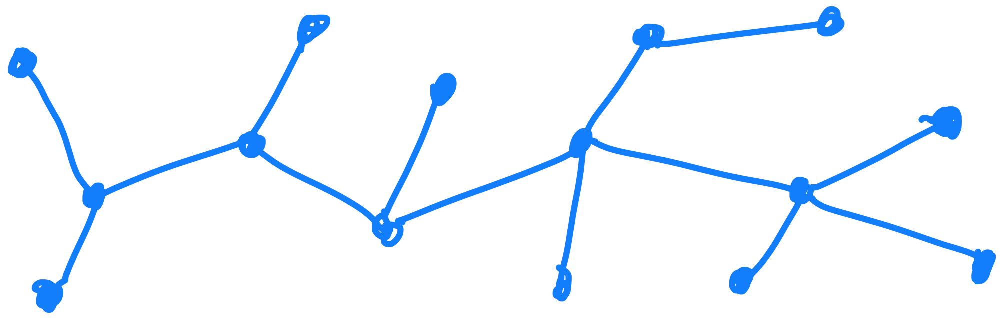
Q0. Which graphs are satisfiable/unsatisfiable?

Q1. How much more complicated is GraphSAT compared to SAT.

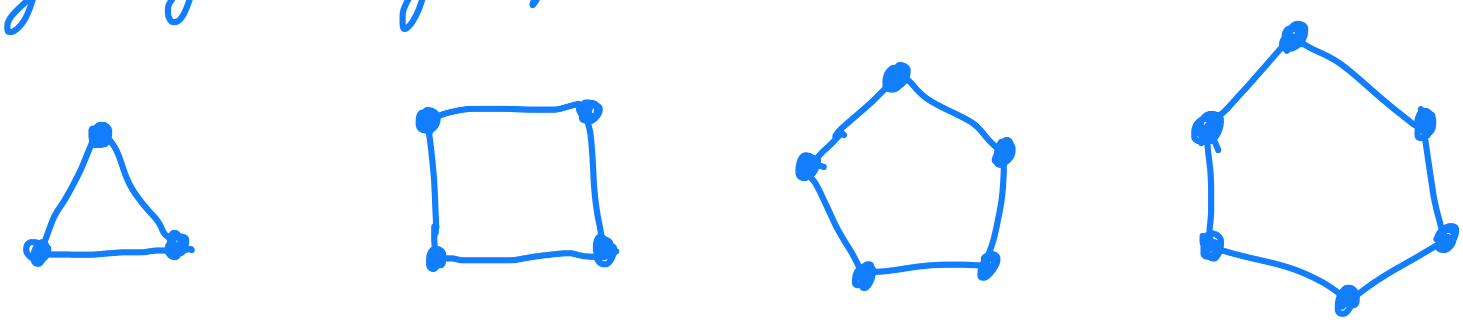
I will state the answers to Q0 as facts instead of results. $\longrightarrow$ More on Friday @ 4 pm CST

Graphs known to be SAT:

- Any finite tree is SATISFIABLE

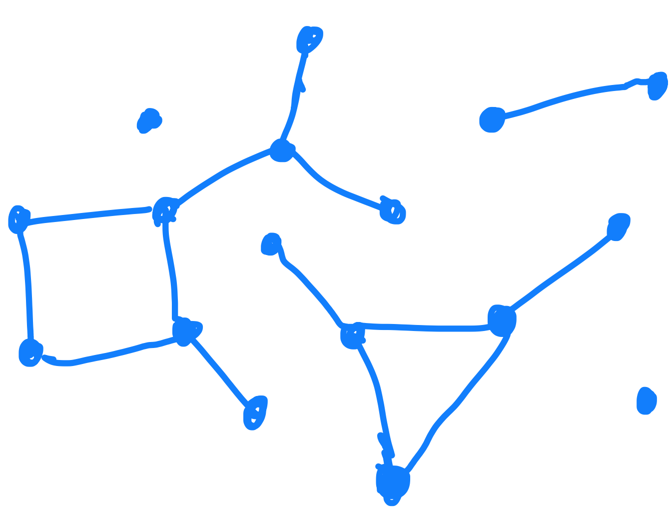


- Any cycle graph is SATISFIABLE



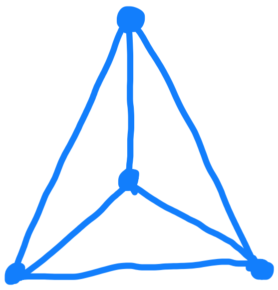
...

- "Combinations" of these are SATISFIABLE

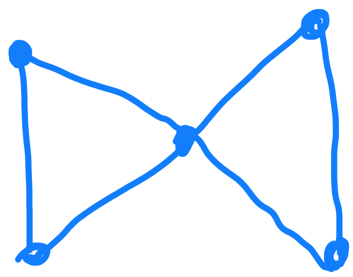


This whole thing is SATISFIABLE.

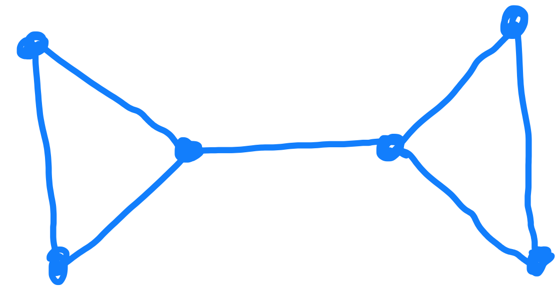
Graphs known to be UNSAT:



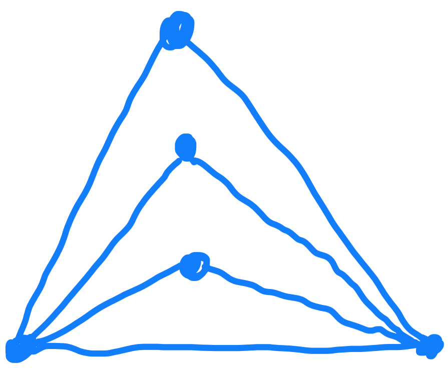
K_4



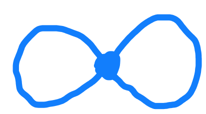
Butterfly



Bowtie



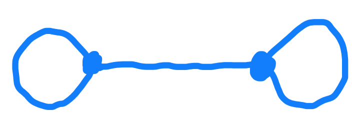
$K_{1,1,3}$



$a \wedge \neg a$

Double loop

"smooth butterfly"



$a \wedge (\neg a \vee \neg b) \wedge b$

"Smooth bowtie"

Dumbbell

"smooth $K_{1,1,3}$"

$(a \vee b) \wedge (a \vee \neg b) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg b)$

Structure theorem #0:

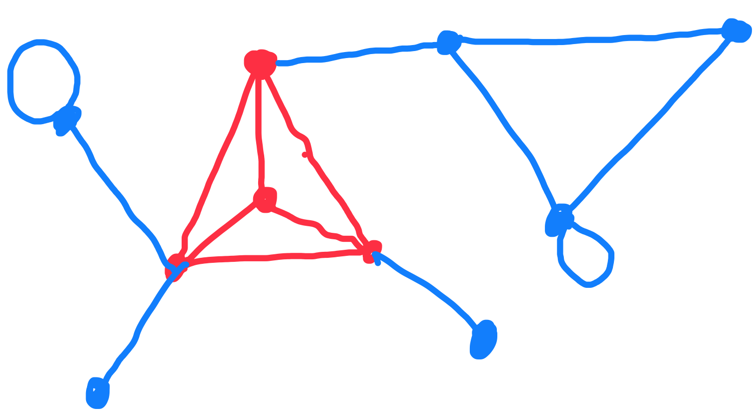
If g has two connected components g_1 and g_2 then

g is SATISFIABLE $\Longleftrightarrow$ g_1 is SATISFIABLE AND g_2 is SATISFIABLE.

Structure theorem #1:

Let g be a subgraph of h.

g is UNSATISFIABLE $\Longrightarrow$ h is UNSATISF.



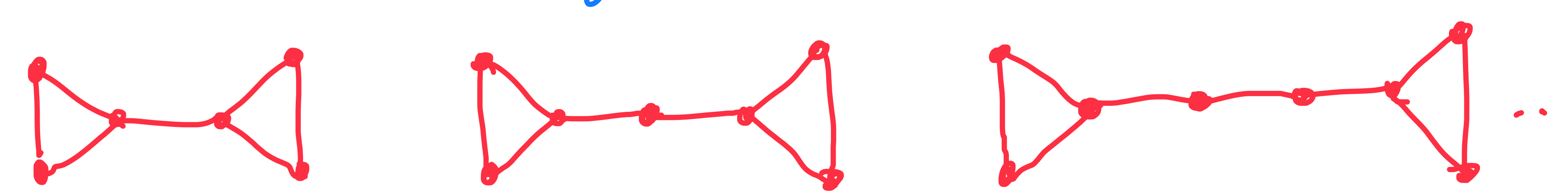
is unsatisfiable.

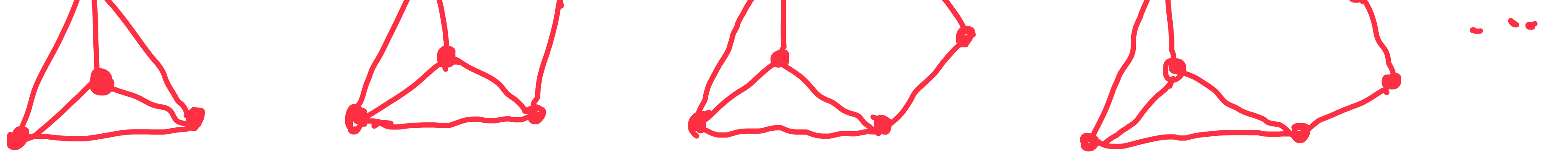
Algorithm idea for GraphSAT:

0. Let g be an arbitrary graph.
1. Search for all of the known unsatisfiable graphs as subgraphs in g.
2. If none exist, then g is SAT.
   If some exist, then g is UNSAT.

Problems:

- SUBGRAPH MATCHING is NP-complete.
- We need to search for infinitely many subgraphs!



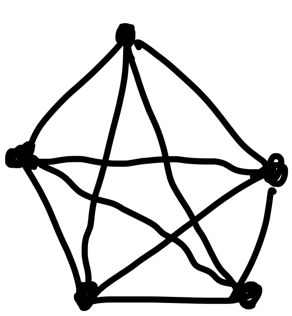


Structure theorem #2:

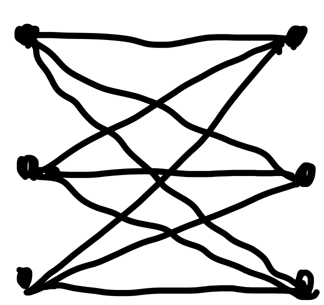
All paths can be smoothed without changing SATISFIABILITY.

Algorithm idea: Check for UNSATISFIABLE subgraphs while also checking for possible "smoothing" operations.

Kuratowski's result: A graph g is planar iff it does not "contain"



K_5



$K_{3,3}$

- these should not be subgraphs of g.
- Cannot obtain K_5 or $K_{3;3}$ by smoothing paths in g.
- Cannot obtain by "merging edges in g"

Q. How quickly can we decide if an arbitrary graph g is "contained" in an arbitrary graph h?
(ala Kuratowski)

GRAPH MINOR PROBLEM $\in$ NP-complete.

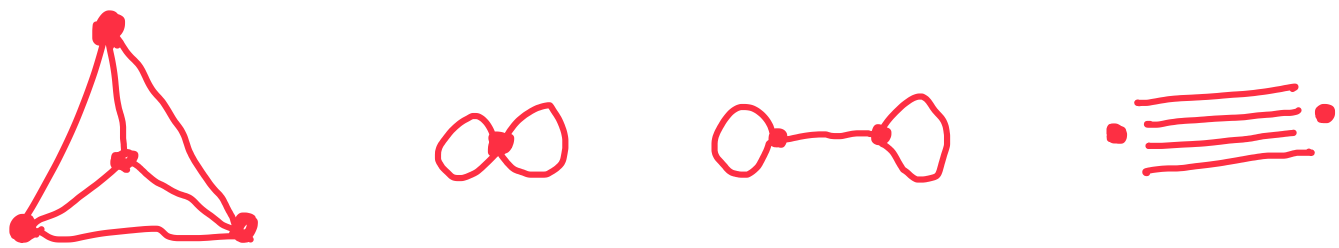
Q. How quickly can we decide if a fixed graph g is "contained" in an arbitrary graph h?

$\in P$

This means finiteness of Kuratowski's result is the key!

Our result: A graph g is SATISFIABLE iff it does not "contain"

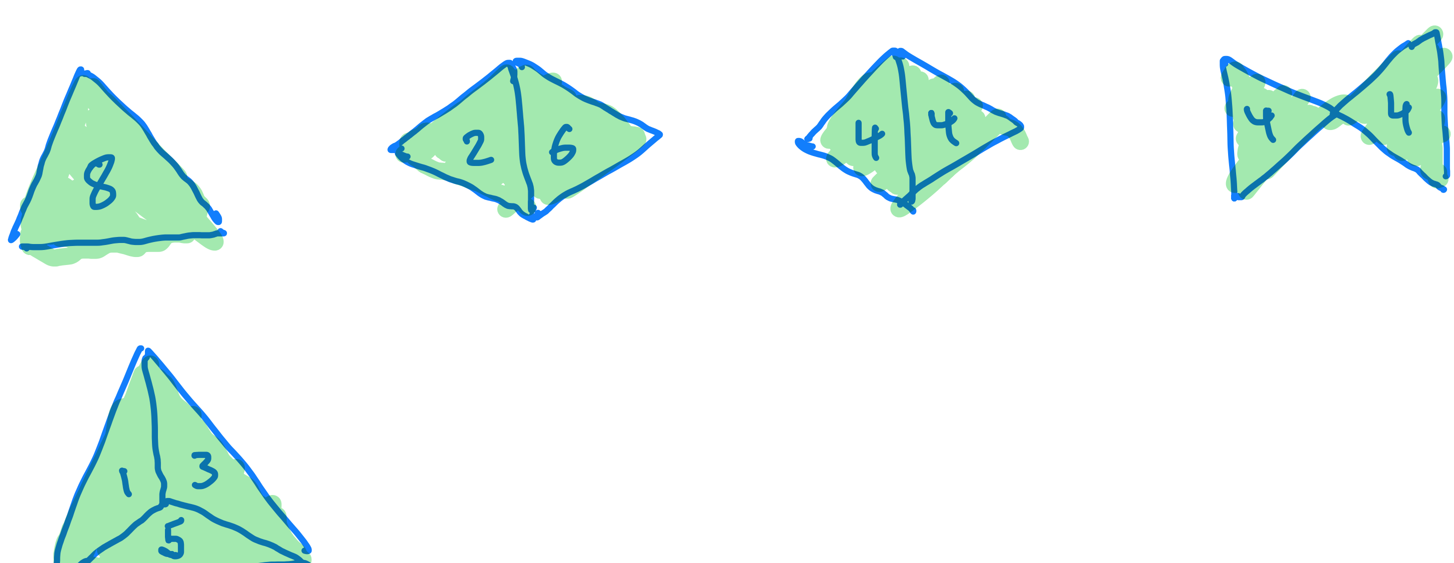
more relaxed than Kuratowski's condition

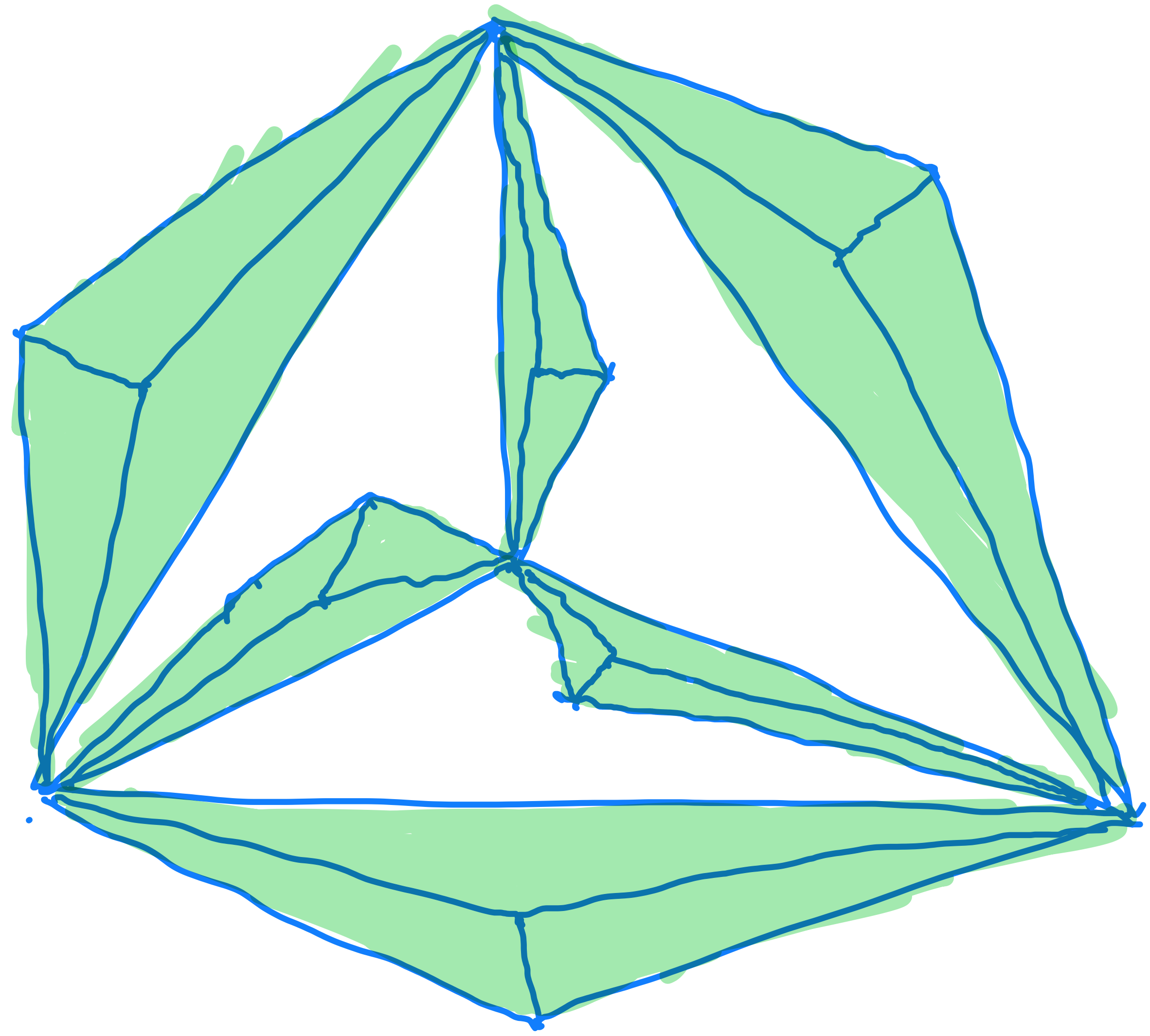


This gives us an algorithm in P.

New problem: The above result only holds for simple graphs (edge size ≤ 2).
For hypergraphs, we don't yet have a finite list.

Known UNSATISFIABLE hypergraphs:





2SAT $\in$ P and 2GraphSAT $\in$ P

3SAT $\in$ NP-complete. 3GraphSAT $\in$???

3GraphSAT (bruteforce) $\notin$ NP

- Given an arbitrary graph that is SATISFIABLE.
- To verify that it is SATISFIABLE, we need to check all boolean formulae in that graph are SATISFIABLE.
- Each check can be done in P time $\because$ 3SAT $\in$ NP.
- But there are exponentially many formulae to check. $\therefore$ Checking will take more than P time.
- 3GraphSAT $\notin$ NP

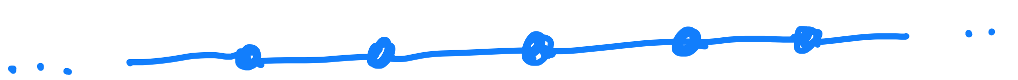
3GraphSAT $\notin$ co-NP

- Given an arbitrary unsatisfiable graph.
- To check unsatisfiability we need to check one of the formulae is unsatisfiable.
- This boils down to checking a single unsatisfiable formula.
- But 3SAT $\notin$ co-NP $\therefore$ 3GraphSAT $\notin$ co-NP.

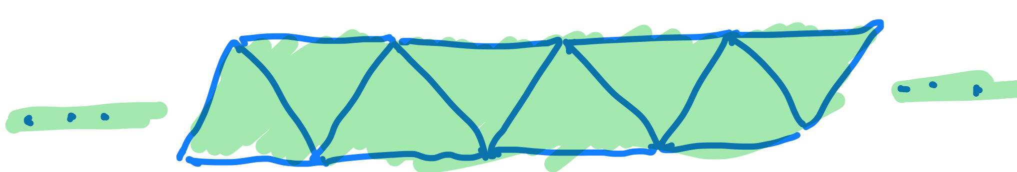
Bonus: Infinite graphs

Graphsat can be extended to infinite graphs! Even though infinite SAT doesn't make complete sense. And we cannot talk about complexity classes for infinite inputs.

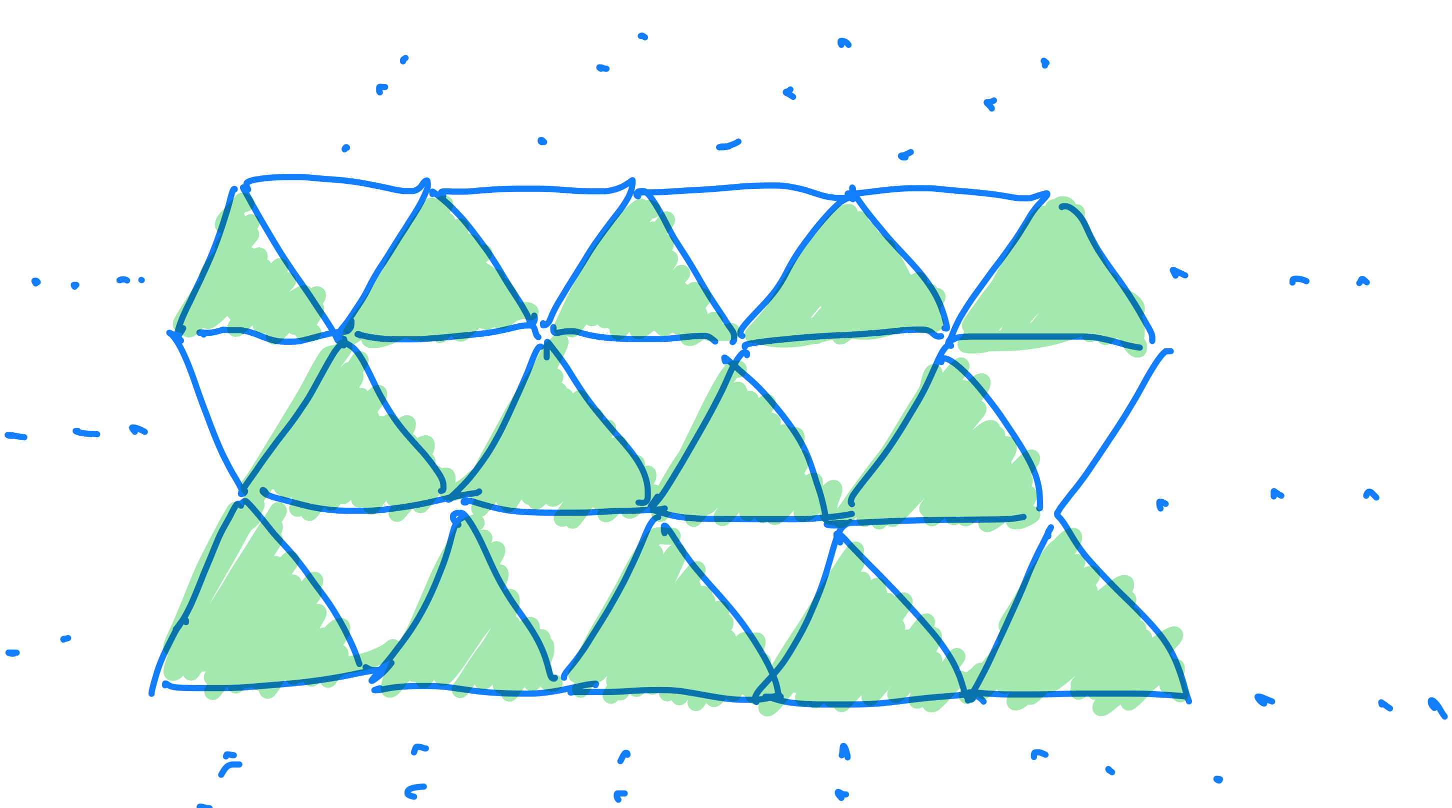
Known SATISFIABLE infinite graphs:



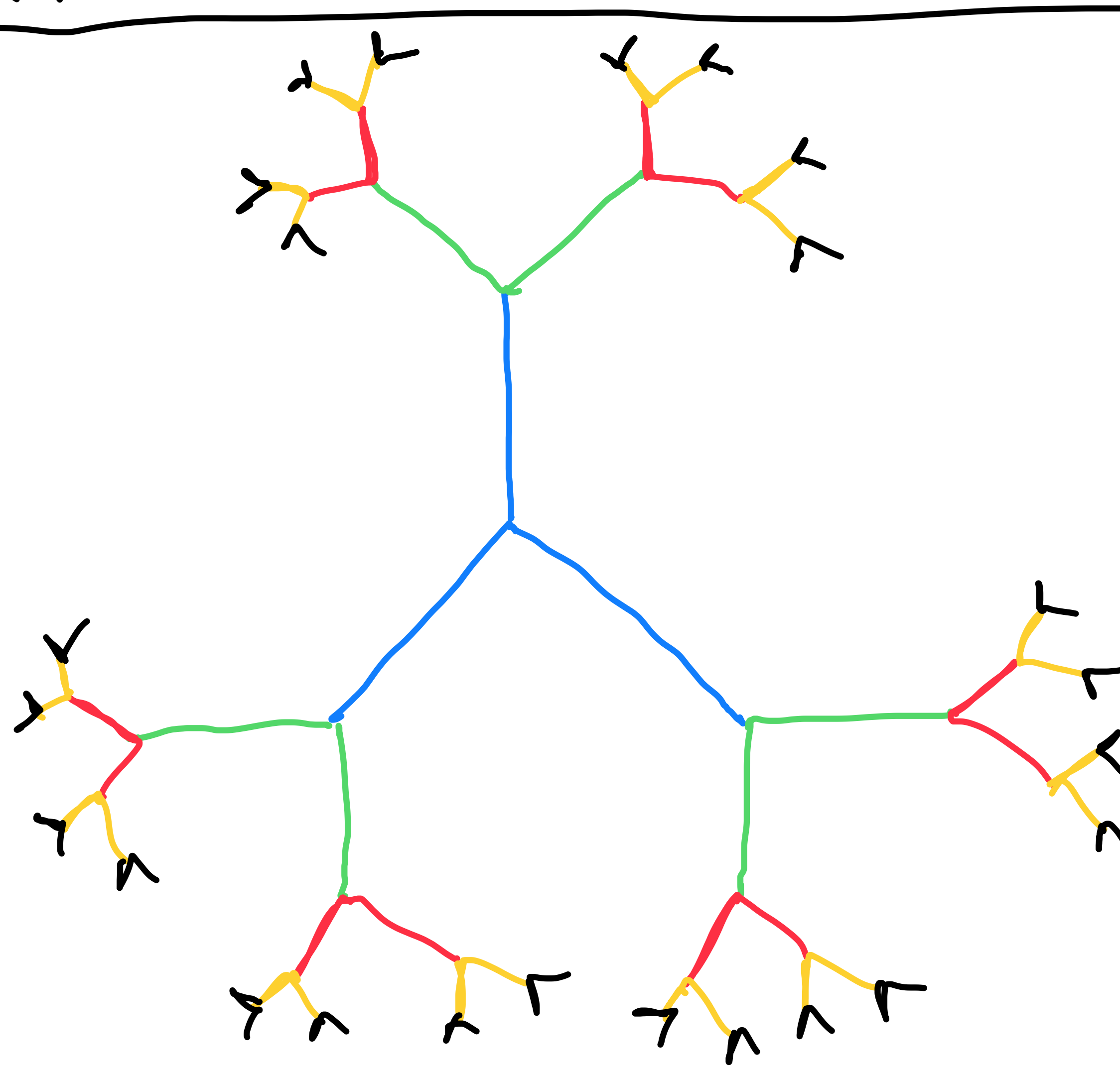
infinite line graph.



infinite strip of triangles



Known UNSATISFIABLE infinite graph:



Infinite tree graph with uniform degree 3.