Complexity & Algorithms - Spring '26

Complexity Hierarchies

Recap

- Midterm practice posted
- No reading next Tuesday
  - ↳ exam in class
- HW - due Friday
  (consider Saturday a hard deadline, so solutions can be posted)

Complexity Boundaries

Lots of questions this week and last about why some variants are hard & others aren't.

I'm not sure I have good answers!

Remember:

- P could = NP
- Difference: can I find an algorithm

An example: 2SAT

$\bar{a} \Leftrightarrow \neg a$

Why is it easy?

$\Phi = (x_1 \vee x_2) \wedge \cdots (x_i \vee x_j) \cdots$

Some logic: $(a \vee b)$: need a or b

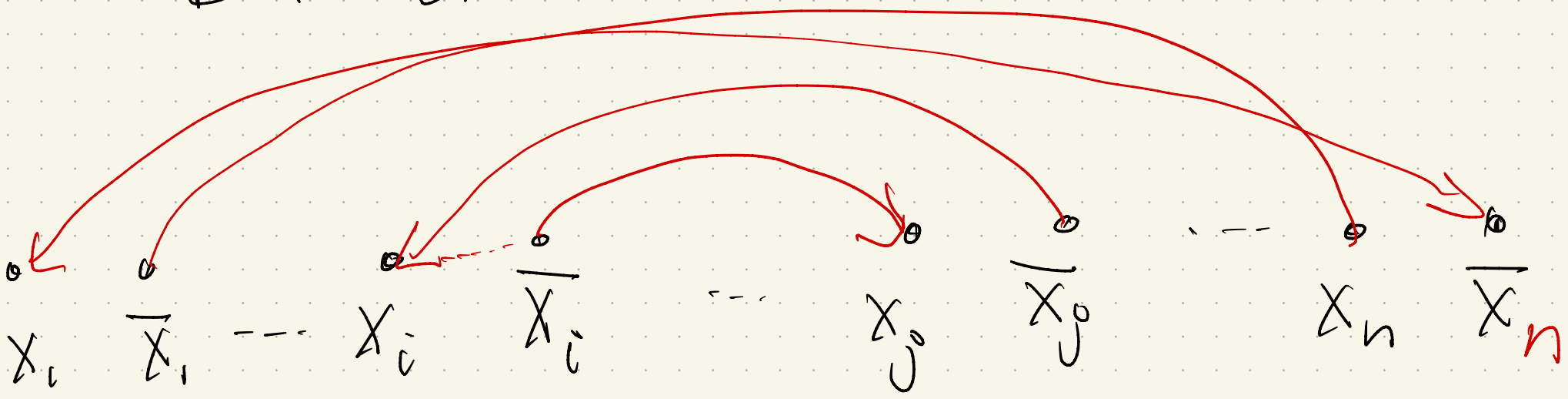
recall: $p \to q$ is false only if p = true, q = false

$\neg q$	a	b	$a \vee b$	$\neg a \to b$	$\neg b \to a$	$(\neg a \to b) \wedge (\neg b \to a)$
F	T	T	T	T	T	T
F	T	F	T	T	T	T
T	F	T	T	T	T	T
T	F	F	F	F	F	F

Build a graph!

$(x_1 \wedge \overline{x_n})$

Build 2n nodes:



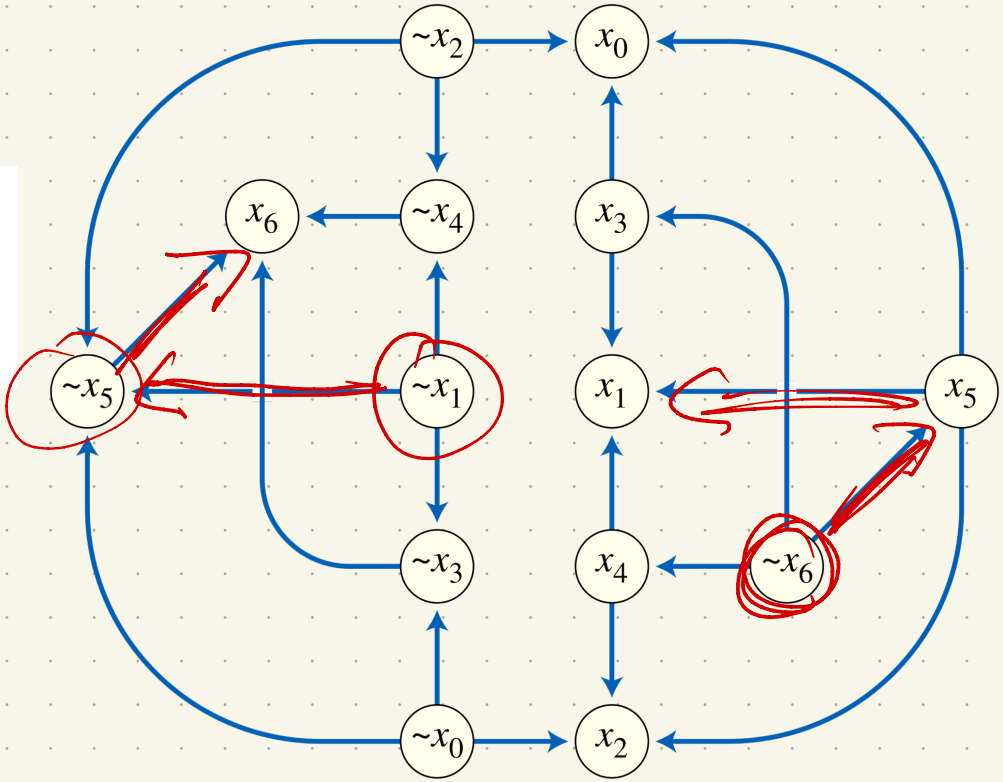
for $(x_i \vee x_j)$: add $\overline{x_i} \to x_j$, $\overline{x_j} \to x_i$

Identify implications

Note: Can't have $(x_i$

Example

$$(x_0 \vee x_2) \wedge (x_0 \vee \neg x_3) \wedge (x_1 \vee \neg x_3) \wedge (x_1 \vee \neg x_4) \wedge (x_2 \vee \neg x_4) \wedge (x_0 \vee \neg x_5) \wedge (x_1 \vee \neg x_5) \wedge (x_2 \vee \neg x_5) \wedge (x_3 \vee x_6) \wedge (x_4 \vee x_6) \wedge (x_5 \vee x_6).$$



This is SAT:
all $x_i = T$
(There are 15 more)

Consider two clauses: $(a \vee b)$ and $(\neg b \vee c)$
↳ also need $(a \vee c)$. Why?
So: if a is false → need c to be true
$(c \vee b)$

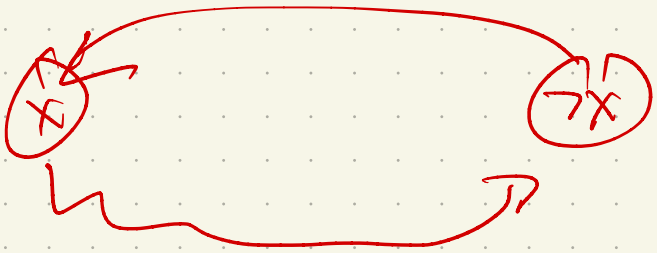
Paths in G:

In other words:

if we have a path $x \rightsquigarrow \neg x$,

and $\neg x \rightsquigarrow x$, problem!

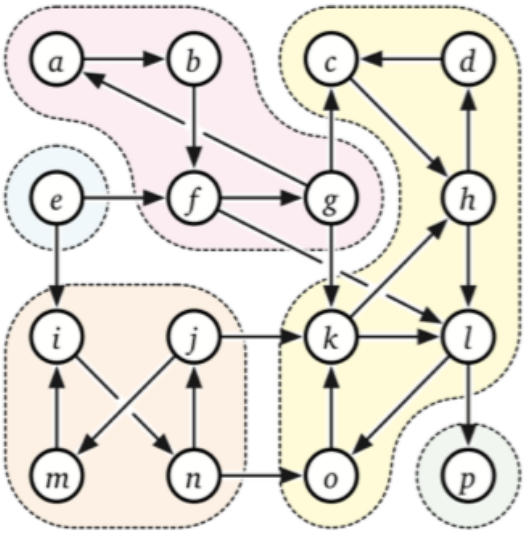
Why?

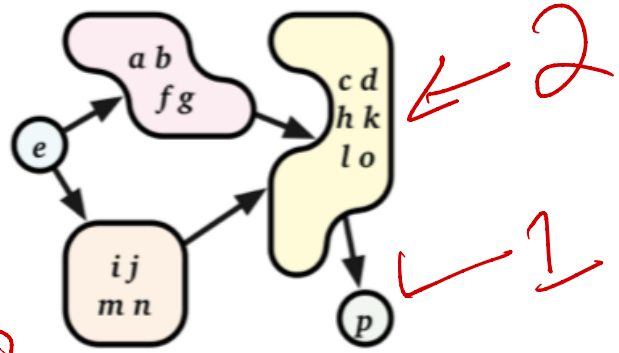


The other direction

Suppose no x & $\bar{x}$ are in the same strongly connected component.

↳ Dfn: $S \subseteq V$ is strongly connected if $\forall u, v \in S$, there is a path $u \rightsquigarrow v$ & $v \rightsquigarrow u$.





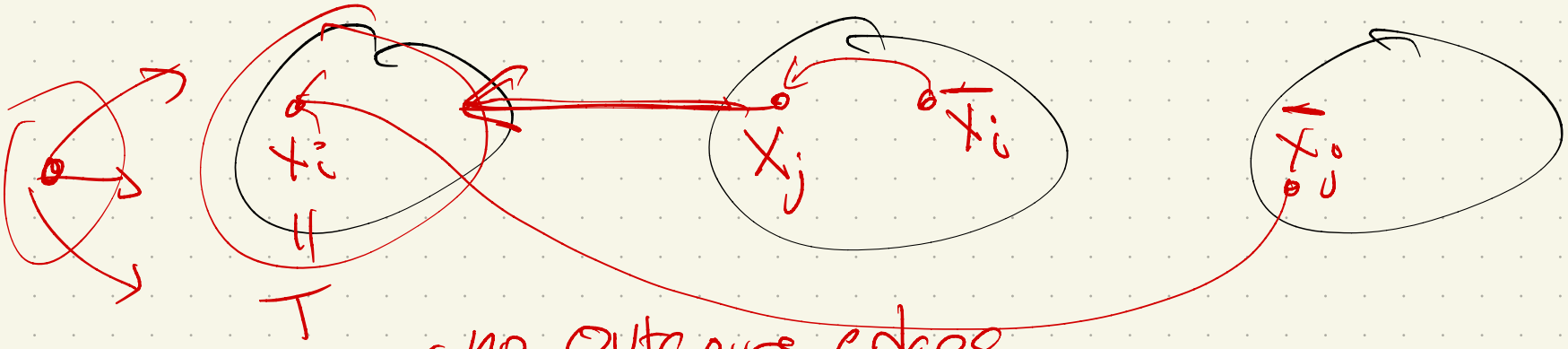
Directed cyclic graph

6.13. The strong components of a graph G and the strong component graph $scc(G)$.

Algorithm:

Collapse our graph into SCCs.

Take topological ordering: $(x_i \vee x_j)$



Start at end: make each variable true.
↙ no outgoing edges

Process backwards:

each clause has 2 edges

↳ any clause w/ variable in later SCC is already true

Runtime: top sort: $O(V+E)$

↳ n vars, m claves

$V = 2n$

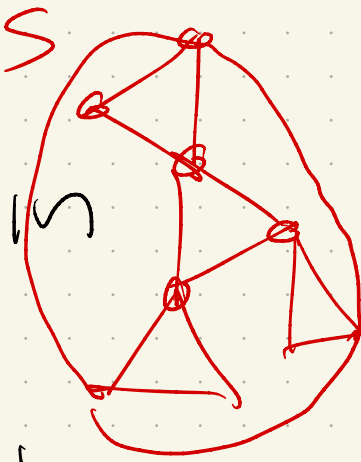
$E = 2m$

$\Rightarrow O(m+n)$

Why is 3SAT harder?

The alg doesn't work!

Another example Hamiltonian vs



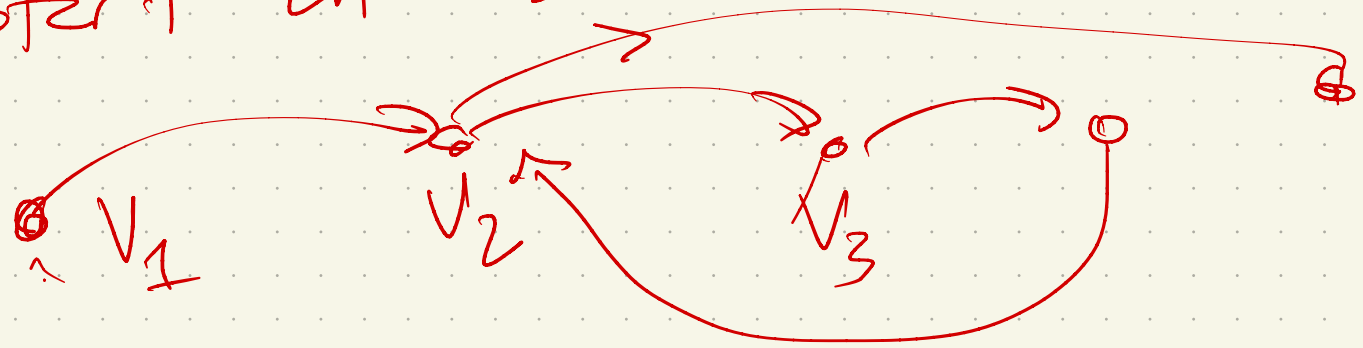
Euler ~~tours~~ "cycles": visit each edge in G with no repeats

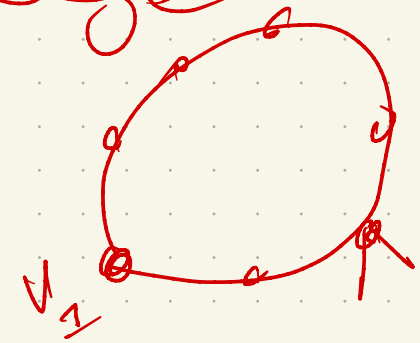
Theorem: G has an Euler cycle $\iff$ every vertex in G has even degree. (& G is connected)

Proof: $\Rightarrow$: Spps G has Euler tour: trace tour → every time I enter v, I must also leave it. ⇝ +2 to degree except first/last vertex ⇝ +2 to deg.

⇐: $\forall v \in G$, $d(v)$ is even.

start at some vertex v_1 → has some edge





If each v_i has outgoing edges ✓

continue.

Stuck when no outgoing edge:

Must be back at v_1.

- If I've used every edge, done.
- If not: Start at some vertex with edges left.

So: a "local" condition can be checked in polynomial time.

Algorithm:

for each $v \in V$
check if $d(v)$ is even
if not
fail
it is Eulerian

In contrast, for Hamiltonian tours:

Need global ordering of vertices.
A local "bad choice" can get you stuck.

How many orderings?

$V!$ orders

↳ "feels" exponential

Hardness of approximation

2 weeks ago: TSP is NP-Hard.

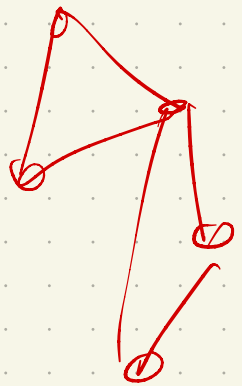
Reduction: From Hamiltonian cycle. ⇝ known NP-Hard

Input: $G = (V, E)$, unweighted.

Build TSP instance: let $k = |V|$

& $G' = K_{|V|}$ with weights:

$$\ell(e) = \begin{cases} 1 & \text{if } e \in G \\ 2 & \text{if } e \notin G \end{cases}$$



If G has Hamiltonian cycle:

Build a TSP with weight = $|V|$ by using the cycle

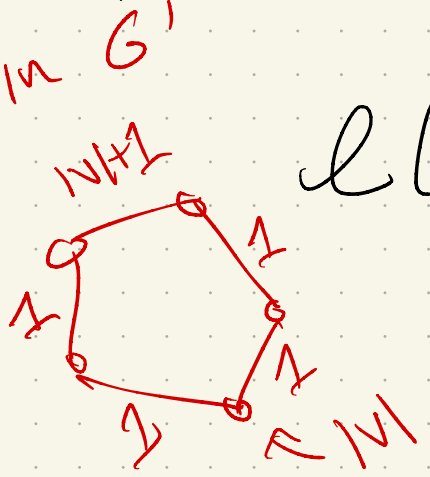
If G' has a TSP with weight $|V|$:

→ need $|V|$ edges, so can't use any weight 2 edges

⇒ also a Ham cycle in G

Think about approximation

New weight function for G':



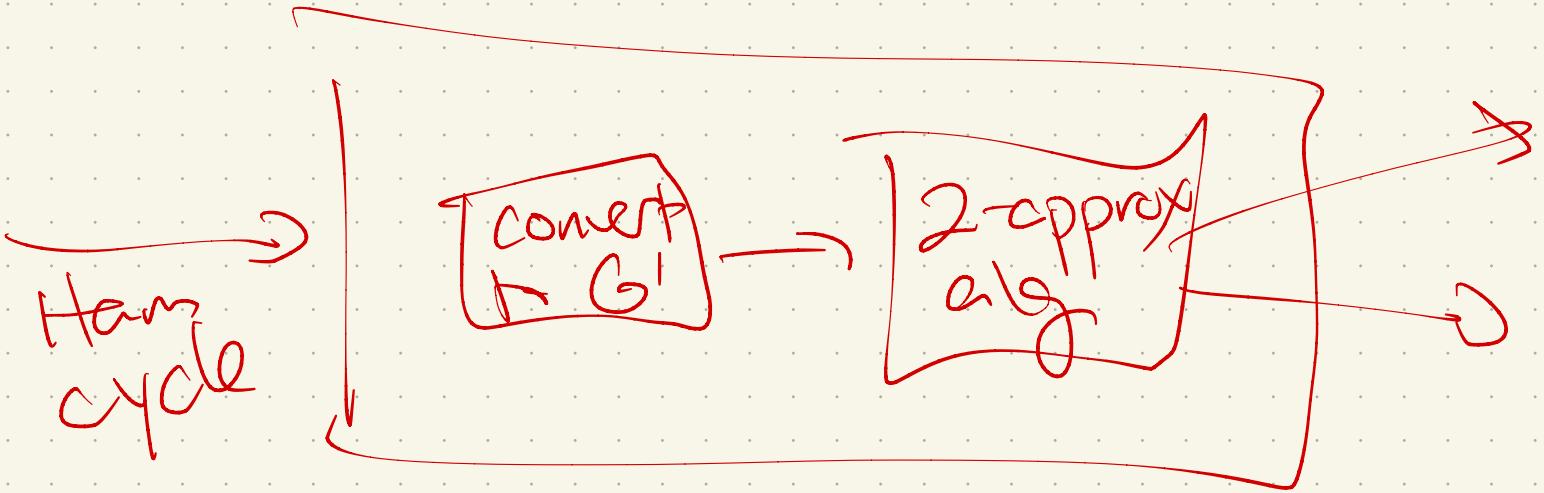
$$\ell(e) = \begin{cases} 1 & \text{if } e \in G \\ |V|+1 & \text{if } e \notin G \end{cases}$$

$+2$

If we use any $e \notin G$, length of TSP: $\geq 2|V|$

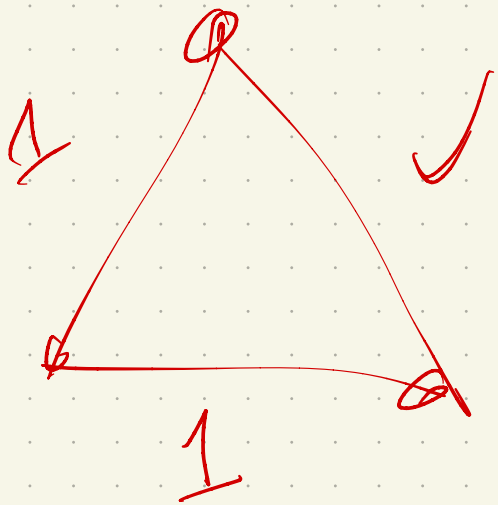
So if we have < 2-approx:

could use that to solve ham cycle


convert to G'
2-approx alg
Ham cycle

Wait: we did approx TSP!

but assumed $\triangle$-ineq



huge # $|V|+1$

does not satisfy $\triangle$-ineq

Even more:

$$\ell(e) = \begin{cases} 1 & \text{if } e \in G \\ f(n) & \text{if } e \notin G \end{cases}$$

Then: $f(n)$-approximation would imply $P = NP$, since could solve Hamiltonian cycle.

Strong versus weak NP-Hardness

A problem is weakly NP-Hard if

- it is NP-Hard
- but becomes polynomial-time when all inputs are bounded by a polynomial of input size.

Also call "pseudo-polynomial" time, polynomial in n + numeric values

Example: Subset sum

$O(nT)$, with n numbers & T the target

T has size $\log T$ when written

DP

$O(nT)$ is polynomial if

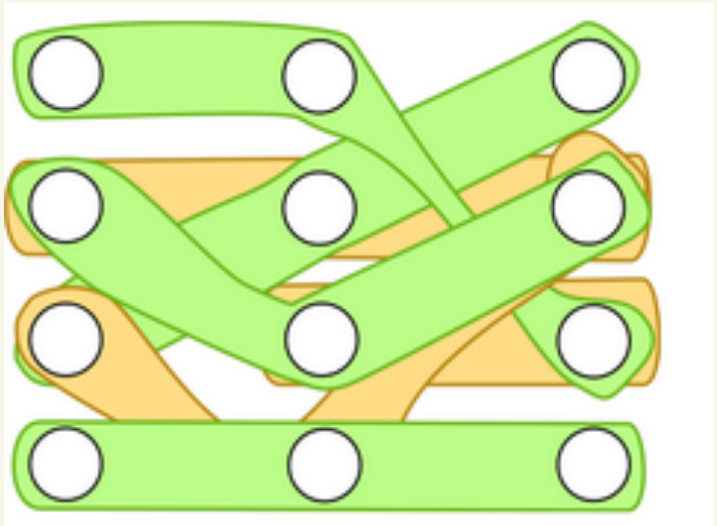
$T = O(n^c)$

$O(n^{c+1})$

In other words:
easy when #s stay small!

Strong NP Hardness: 3-dim matching

Given 3 disjoints sets X, Y & Z, each size n, along with a set $T \subseteq X \times Y \times Z$ of ordered triples, can we find a set $S \subseteq T$ s.t. each element of $X \cup Y \cup Z$ is in exactly 1 triple?



Note:

- generalization of bipartite matching
- No #s in sight!

Reduction: from 3SAT

Input: boolean function Φ with n variables & m clauses.

$m = 4$

Variable gadget:

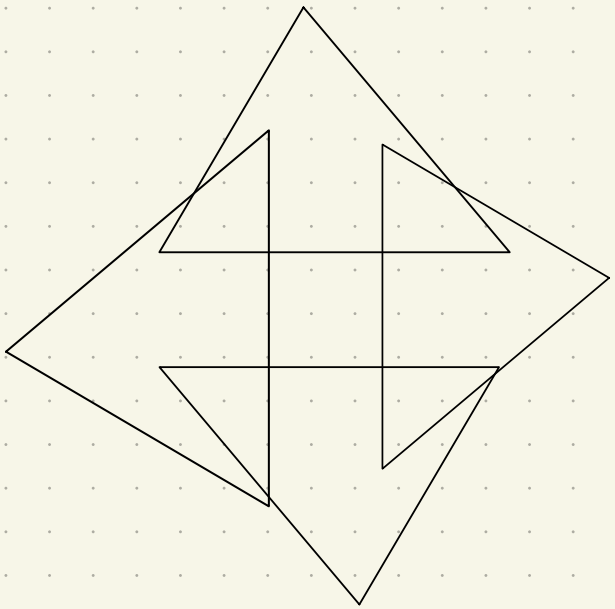
- m elements in core
- m elements in "tips"

Even/odd:

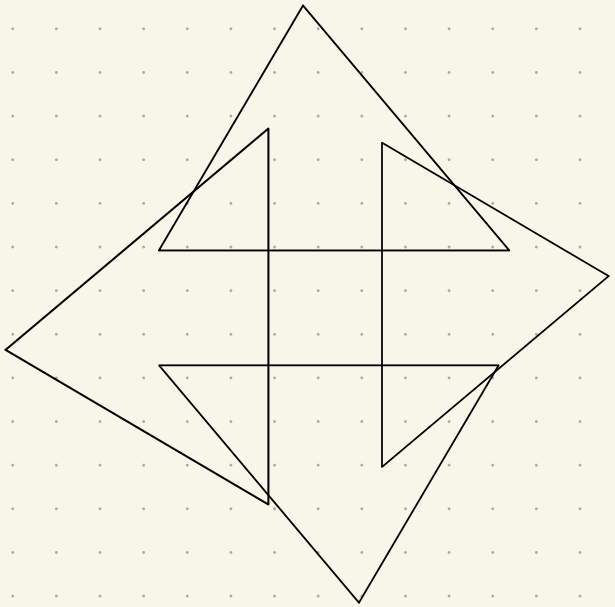
Clause gadget

- x_i: "even" covered $\Rightarrow$ true
- "odd" covered $\Rightarrow$ false

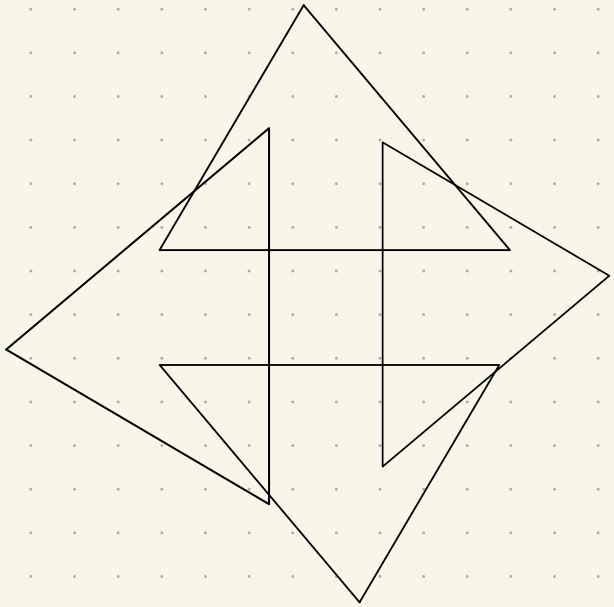
$x_i \vee \overline{x_j} \vee x_k$:



x_i



x_j



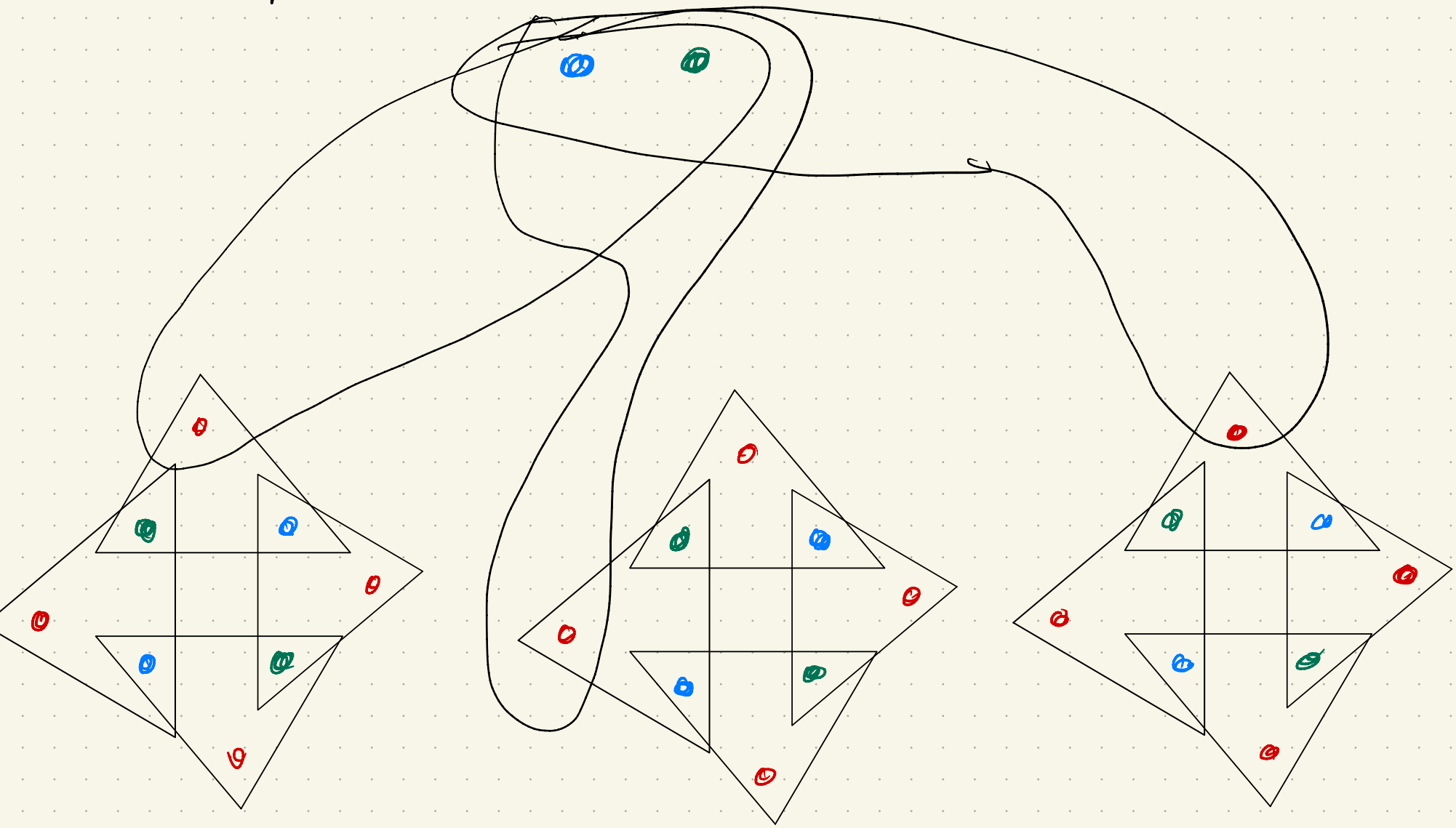
x_k

So: almost there!

Problem: some uncovered "tips".

What if x_i is in only 1 clause?

Finally: X, Y + Z!



Construction time:

Note: No numbers really anywhere!

So: nothing polynomially bounded.

Really about combinatorial structure of inclusion in the solution.