Algorithms - Spring '25

NP-Hardness

Recap

- Have a good Easter break!
- Next HW: posted, due last days of classes
  - ↳ join a HW8 group, then sign up in calendar
- Readings: Resume next week
  - ↳ different book
- Final: ~~Thurs.~~ Thus. May 8, 10³⁰ am

"The Pattern"

In the same way a "word problem" can ask you to model a graph & then choose correct alg

→ NP-reductions require some imagination & practice!

- Find known NP-Hard problem
- Convert an instance of ~~NP-Hard~~ into your problem
- Prove "Yes" in NP-Hard Problem ⟺ "Yes" in ours

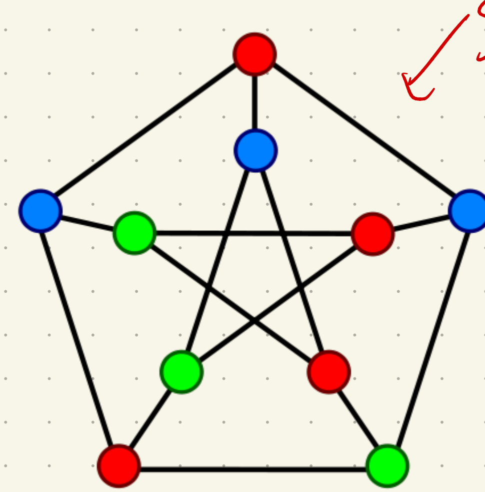
Next: Graph Coloring

A <u>k-coloring</u> of a graph G is a map:

$$c : V \to \{1, \ldots, k\}$$

that assigns one of k "colors" to each vertex so that every edge has 2 different colors at its endpoints

3-colorable



Goal: Use few colors

$k = V$ easy!

Thm: 3-colorability is NP-Complete.

(Decision version: Given G ~~& k~~ 3,
output yes/no)
if 3-colorable

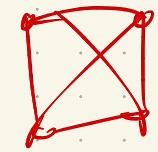
In NP:
certificate: color for each vertex

To check:
loop over every edge
& verify endpoints have
different colors

NP-Hard:

Reduction from 3SAT.

4 colors



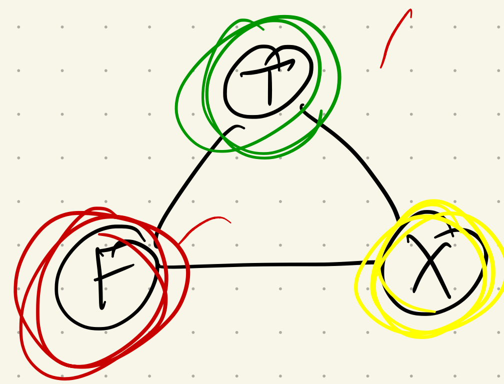
Given formula for 3SAT Φ,

we'll make a graph G_Φ.

Φ will be satisfiable

$\iff G_\Phi$ can be 3-colored.

Key notion: Build "gadgets"!

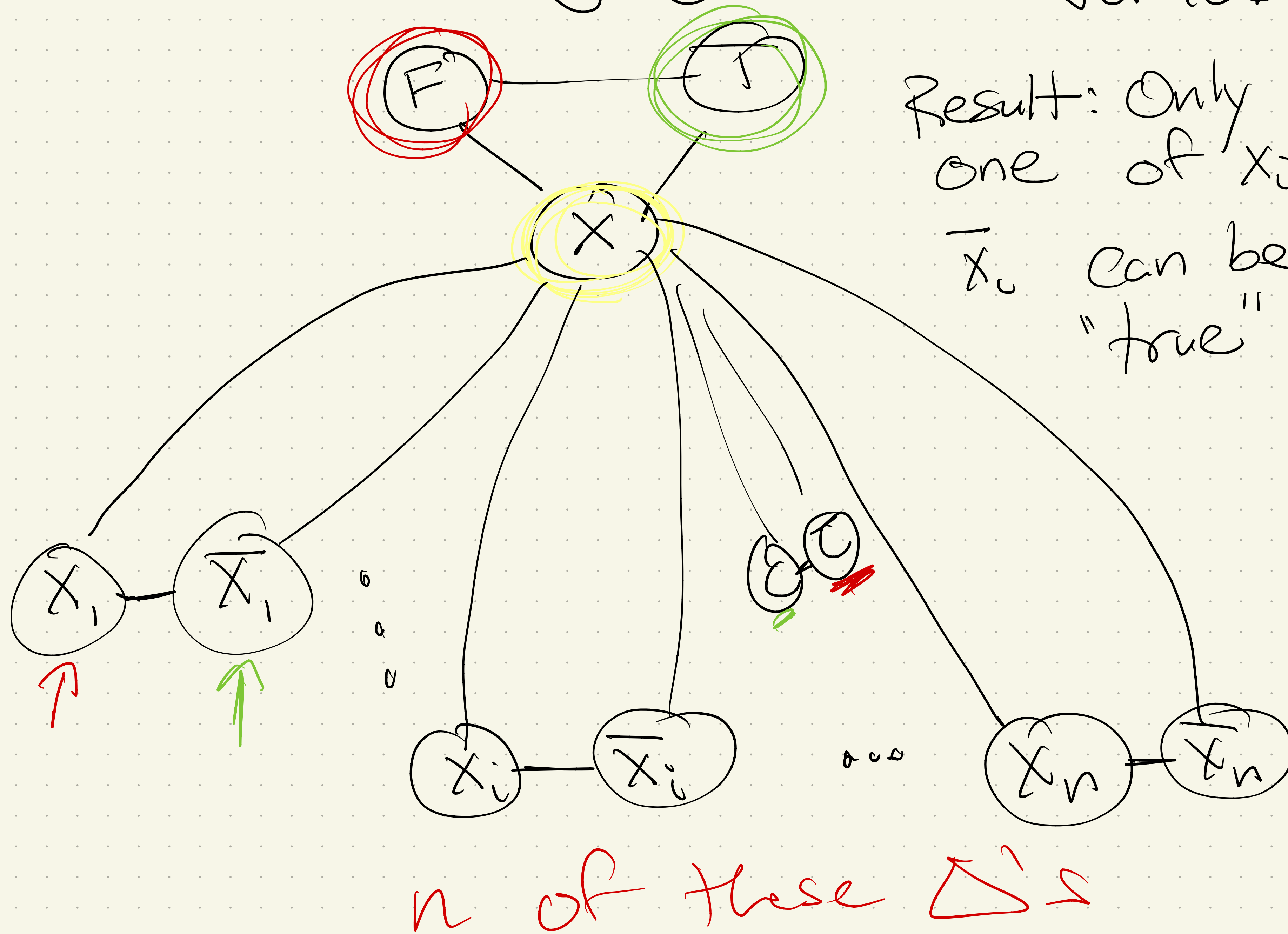


① Truth gadget –

Must use 3 colors –

so establishes a "true" color.

② Variable gadget: one per variable



Result: Only one of x_i + $\overline{x}_i$ can be "true"

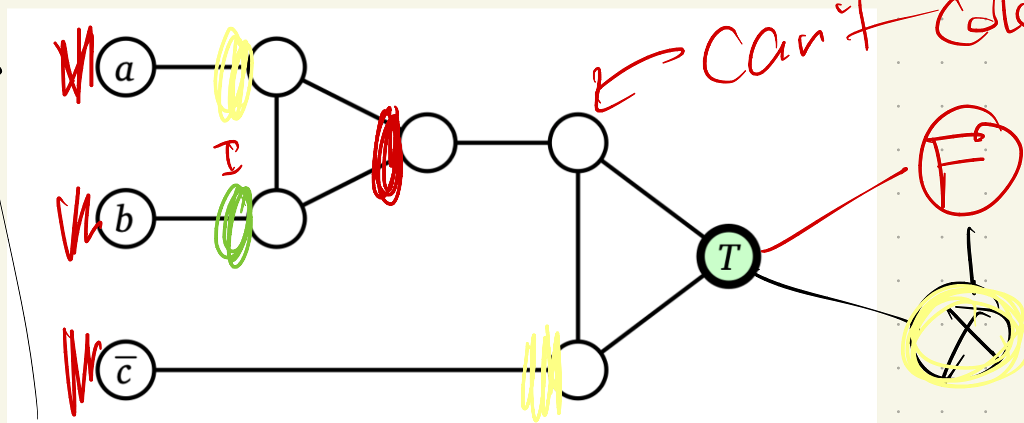
③ Clause gadget: $(x_i \vee \overline{x_j} \vee x_k)$

For each clause, join 3 of the variable vertices to the "true" vertex from the truth gadget.

<u>Goal</u>: If all 3 are false, no valid 3-coloring

variable B's

can't color



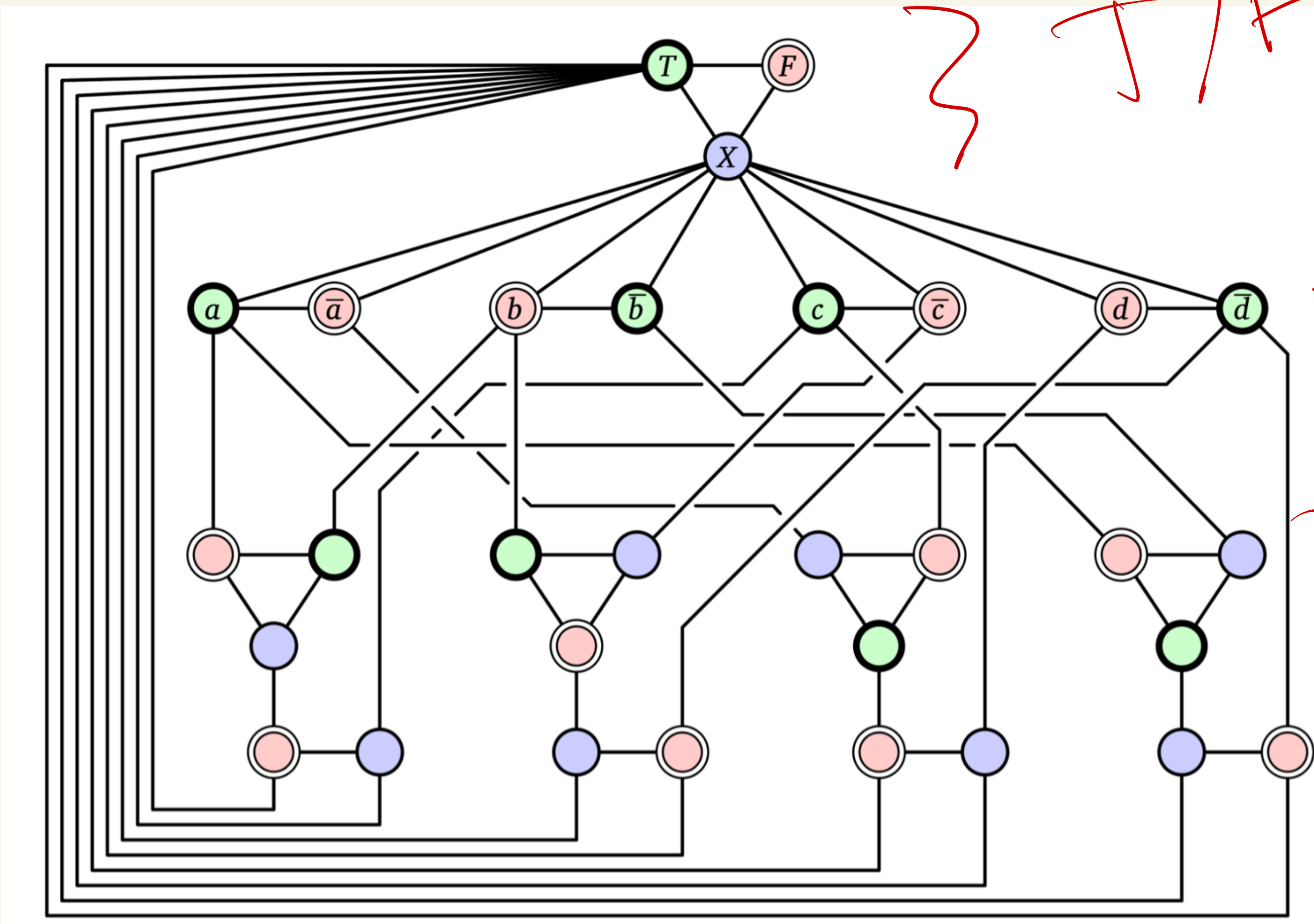
A clause gadget for $(a \vee b \vee \bar{c})$.

Why?? try to color all "false"

↪ can't 3-color

If any vertex is green, we can 3-color

Final reduction image:



A 3-colorable graph derived from the satisfiable 3CNF formula
$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$

T/F gadget

← variables

Clause gadgets

$V = 3 + 2n + 5m$

$E = 3 + 3n + 9m$

poly.

Now, need reduction proof:

3 colorings of G_Φ $\Longleftrightarrow$ Φ is satisfiable

Pf:

$\Rightarrow$: Consider a 3-coloring of G:

Know each clause gadget must connect to a green variable.

↳ set these variables True, & other false.

Must have 1 true var. in each clause, so Φ is sat.

$\Leftarrow$ Consider a satisfying assignment to Φ:

↳ each x_i is T/F

↳ color according in G_Φ

Since each clause has $\geq$ one green vertex, we can 3 color each clause gadget

↳ 7 possibilites, I can draw each out.

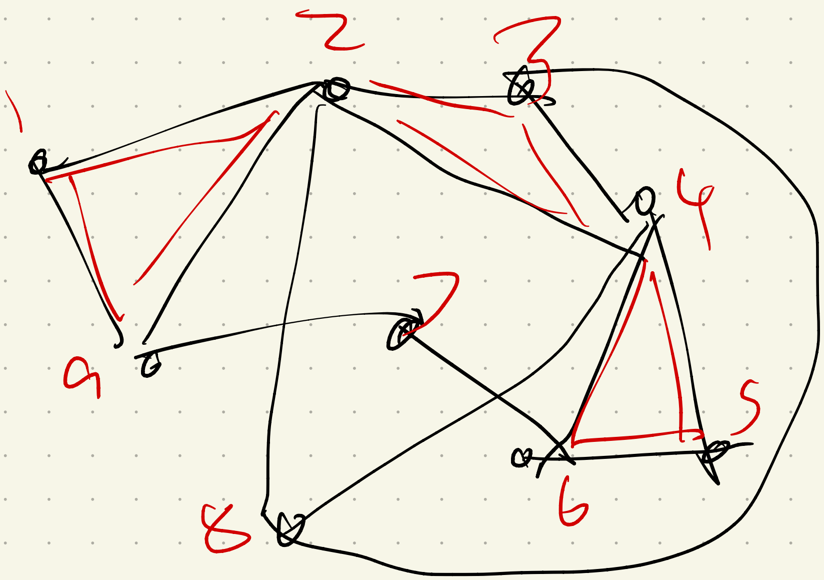
Clique (from ind set) $(x_i \vee x_j \vee x_k)$

Some reductions can feel a bit "easier":

Input: $G=(V,E)$ and $k \in \mathbb{Z}^+$

Is there a complete subgraph in G of size k?

G



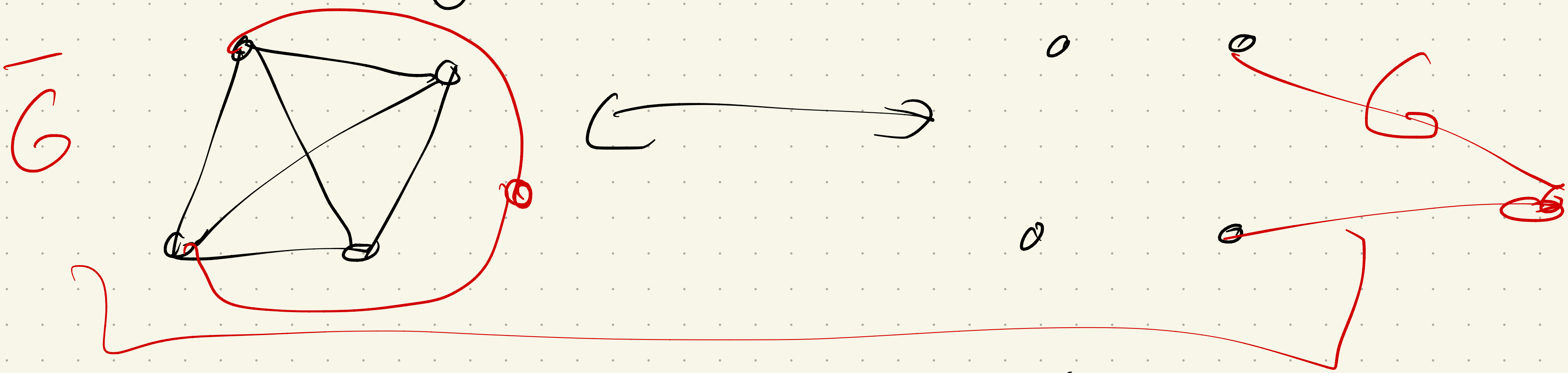
$k=3$? yes (lots)

$k=4$? Yes (one)

$k=5$? No

Feels a bit like opposite of ind set!

all edges no edges



Reduction: From ind. set.

Input: $G = (V, E)$ & k : does G have ind set of size k?

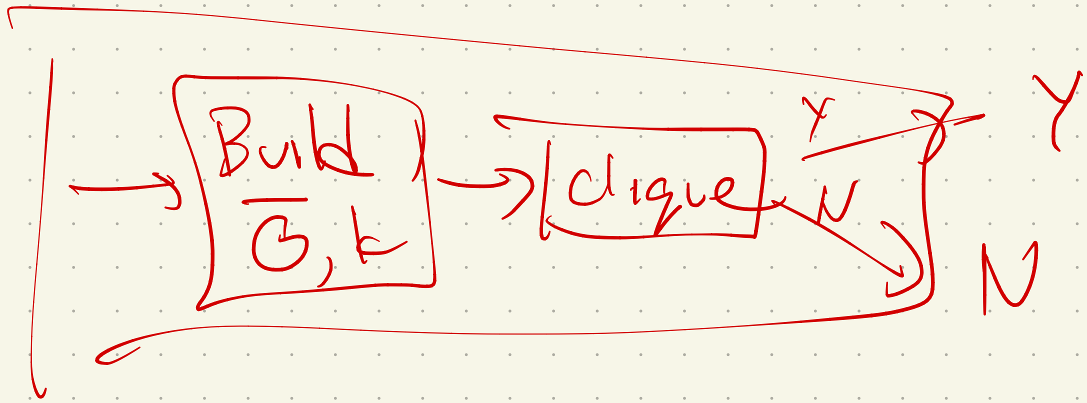
↳ Build a new graph $\overline{G} = (\overline{V}, \overline{E})$

$\overline{V} = V$

uses G

$\overline{E} = \{ \{u,v\} \mid \{u,v\} \notin E \}$

Ind Set:
G, k



Claim: G has ind. set of size k

$\iff \overline{G}$ has clique of size k.

proof:

$\Rightarrow$ Consider ind. set in G: k vertices $\{v_1, \dots, v_k\} \subseteq V$, s.t. $\forall$ pairs v_i, v_j, $\{v_i v_j\} \notin \cancel{\overline{E}} E$

By dfn of $\overline{G}$, that means $\{v_i, v_j\} \in \overline{E}$

$\hookrightarrow \{v_1, \dots, v_k\}$ are clique in $\overline{G}$

$\Leftarrow$: Consider k vertices in $\overline{G}$, $\{u_1, \dots, u_k\}$,
s.t. $\forall u_i \neq u_j$, $\{u_i u_j\} \in \overline{E}$
(a clique)

Then by dfn of $\overline{G}$, we know
$\{u_i u_j\} \notin E$
$\hookrightarrow$ $\{u_1, \dots, u_k\}$ in ind. set in G. $\boxtimes$

Subset Sum:

Given a set of numbers $X = \{x_1, \ldots, x_n\}$ and a target t, does some subset of X sum to t?

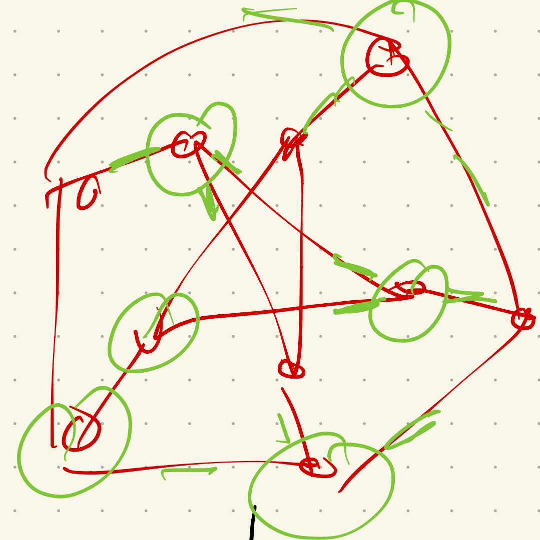
Ex: Actually did this one! See lecture from Ch. 2

Runtime: $O(n \cdot T)$

Why is this not polymonial? exp?

Input: $\underbrace{X}_{\text{size } n} + T$ ← $\log T$ bits to write

Subset Sum is NP-Hard.



Reduction: Vertex Cover

Input: Graph G & size k

Goal: find k vertices, such that every edge in G is incident to at least one vertex in set

Challenge: Construct a set of numbers, s.t. we can hit a target value

$\iff$ G has ~~indep set~~ Vertex Cover of size k

Recall: base 4

$(32012)_4 = 3\cdot 4^4 + 2\cdot 4^3 + 0\cdot 4^2 + 1\cdot 4^1 + 2\cdot 4^0$

... 16's 4's 1's

$$\begin{array}{r} 1\,0\,3\,2 \\ +\,2\,3\,1\,2 \\ \hline 1\,0\,0\,1\,0 \end{array}$$

Idea: Use base 4:
force a target T that requires you to use only vertices, but to "cover" edges

Number edges $0 \ldots E-1$, & create a number for subset sum w/ E "digits"

$e_0 : b_0 = 0\,0\,\_\,\_\,\_\,\_\,\_\,0\,1$

$e_1 : b_1 = 0\,0\,\_\,\_\,\_\,\_\,0\,1\,0$

$\vdots$

$e_{E-1} : b_{E-1} = 0\,1\,0\,\_\,\_\,0$

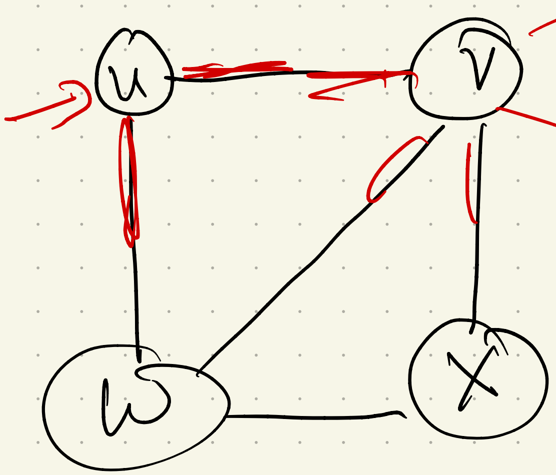
$\underbrace{\qquad\qquad}_{E \text{ spots}} \rightarrow E+1$

For each vertex, make another #:

$$a_v := \underbrace{1[\qquad\qquad]}_{E+1 \text{ digits}}$$

$d(v)$ 1's, for its endpoints (all else 0)

Think of base 4 representation



{u, v}
{x, v}
(w, v)

$a_u := 111000_4 = 1344$	$b_{uv} := 010000_4 = 256$
$a_v := 110110_4 = 1300$	$b_{uw} := 001000_4 = 64$
$a_w := 101101_4 = 1105$	$b_{vw} := 000100_4 = 16$
$a_x := 100011_4 = 1029$	$b_{vx} := 000010_4 = 4$
	$b_{wx} := 000001_4 = 1$

k of these 4's

$0\underbrace{11\cdots1}_{E}$

any value > 0

→ set of #s,

Now, set $T = \underset{\text{input}}{k} \cdot 4^{E} + \boxed{\sum_{i=0}^{E-1} 2 \cdot 4^{i}}$ 2 each

<u>why?</u>

Area k vertices

<u>Proof</u>: size k VC $\Longrightarrow$ sum to T

<u>$\Rightarrow$ VC</u>: $\exists$ k vertices $v_1, v_2, \dots, v_k$

s.t. $\forall e \in E$, e is incident to some

$v_i \in \{v_1, \dots, v_k\}$

$\Rightarrow$ (cont)

Pick a subset:

$\Leftarrow$: suppose some subset of #s sums to T. options?

Recall: $T = k \cdot 4^E + \sum_{i=0}^{E-1} 2 \cdot 4^i$

$+ a_r = \underline{\qquad\qquad}$

$+ b_e = \underline{\qquad\qquad}$

Plus:

Each digit position has only 3 1's across all #s:

So!

Partition:

Given $X = \{x_1, \ldots, x_n\}$, can we partition X into A & B (so $A \cup B = X$, $A \cap B = \emptyset$, & $A, B \neq \emptyset$)

s.t.

$$\sum_{x_i \in A} x_i = \sum_{x_j \in B} x_j \;?$$

Reduction?

Proof:

Some fun examples

arXiv.org > cs > arXiv:1203.1895

Search or Article ID inside arXiv | All papers | Broaden your search usir

(Help | Advanced search)

Computer Science > Computational Complexity

Classic Nintendo Games are (Computationally) Hard

Greg Aloupis, Erik D. Demaine, Alan Guo, Giovanni Viglietta

(Submitted on 8 Mar 2012 (v1), last revised 8 Feb 2015 (this version, v3))

We prove NP-hardness results for five of Nintendo's largest video game franchises: Mario, Donkey Kong, Legend of Zelda, Metroid, and Pokemon. Our results apply to generalized versions of Super Mario Bros. 1-3, The Lost Levels, and Super Mario World; Donkey Kong Country 1-3; all Legend of Zelda games; all Metroid games; and all Pokemon role-playing games. In addition, we prove PSPACE-completeness of the Donkey Kong Country games and several Legend of Zelda games.

Comments: 36 pages, 36 figures. Fixed some typos. Added NP-hardness results (with proofs and figures) for American SMB2 and Zelda 2

Subjects: **Computational Complexity (cs.CC)**; Computer Science and Game Theory (cs.GT)

Cite as: arXiv:1203.1895 [cs.CC]

(or arXiv:1203.1895v3 [cs.CC] for this version)

Submission history

From: Alan Guo [view email]

[v1] Thu, 8 Mar 2012 19:37:20 GMT (627kb,D)

[v2] Thu, 6 Feb 2014 18:24:15 GMT (3330kb,D)

[v3] Sun, 8 Feb 2015 19:45:26 GMT (3425kb,D)

Which authors of this paper are endorsers? | Disable MathJax (What is MathJax?)

Link back to: arXiv, form interface, contact.

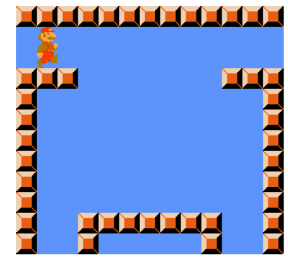


Figure 10: Variable gadget for Super Mario Bros.

shes until it is collected by Mario.

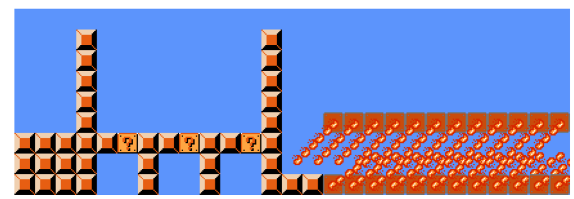


Figure 11: Clause gadget for Super Mario Bros.

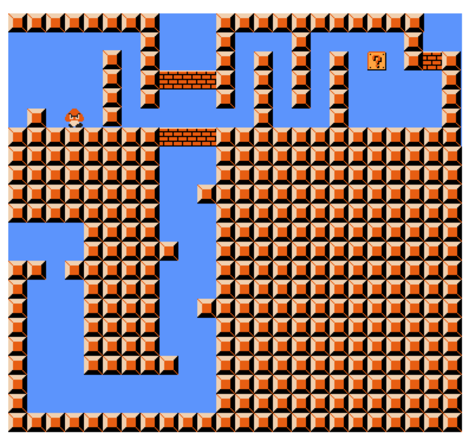
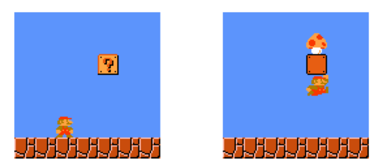


Figure 12: Crossover gadget for Super Mario Bros.



Left: Start gadget for Super Mario Bros. Right: The item block contains a

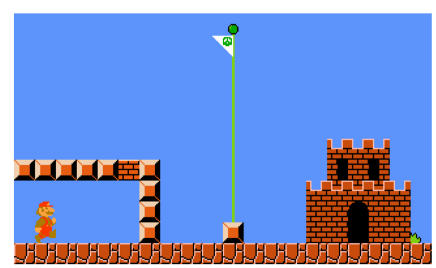
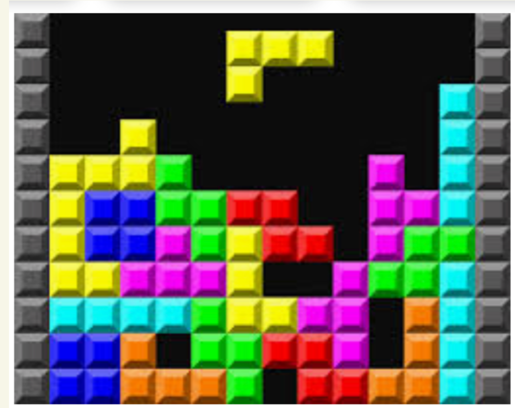


Figure 9: Finish gadget for Super Mario Bros.

Another: Tetris



NP-Hard: Reduce 3-partition

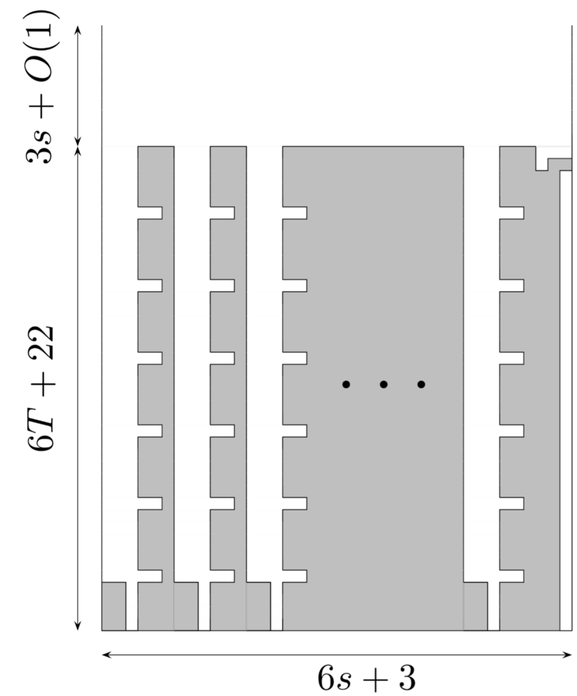


Fig. 2. *The initial gameboard for a Tetris game mapped from an instance of* 3-Partition.

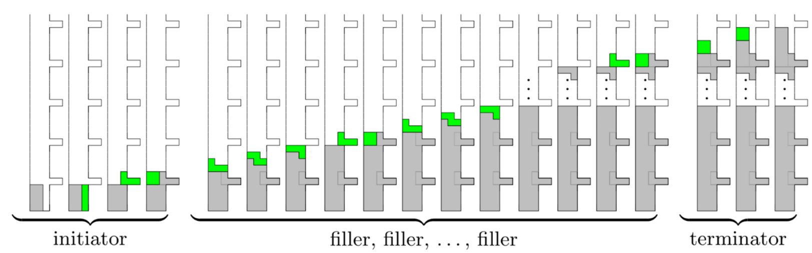


Fig. 3. *A valid sequence of moves within a bucket.*

Again: An active area of research!

arXiv.org > cs > arXiv:1711.00788

Search...

Help | Adva

Computer Science > Computational Geometry

On the complexity of optimal homotopies

Erin Wolf Chambers, Arnaud de Mesmay, Tim Ophelders

(Submitted on 2 Nov 2017)

In this article, we provide new structural results and algorithms for the Homotopy Height problem. In broad terms, this problem quantifies how much a curve on a surface needs to be stretched to sweep continuously between two positions. More precisely, given two homotopic curves γ_1 and γ_2 on a combinatorial (say, triangulated) surface, we investigate the problem of computing a homotopy between γ_1 and γ_2 where the length of the longest intermediate curve is minimized. Such optimal homotopies are relevant for a wide range of purposes, from very theoretical questions in quantitative homotopy theory to more practical applications such as similarity measures on meshes and graph searching problems.

We prove that Homotopy Height is in the complexity class NP, and the corresponding exponential algorithm is the best one known for this problem. This result builds on a structural theorem on monotonicity of optimal homotopies, which is proved in a companion paper. Then we show that this problem encompasses the Homotopic Fréchet distance problem which we therefore also establish to be in NP, answering a question which has previously been considered in several different settings. We also provide an O(log n)-approximation algorithm for Homotopy Height on surfaces by adapting an earlier algorithm of Har-Peled, Nayyeri, Salvatipour and Sidiropoulos in the planar setting.

Almost any problem in AI is NP-Hard.

Not impossible! Just exponential to solve perfectly:

- heuristics
- approximation
- pruning strategies