CS 7800: Advanced Algorithms

Lecture 12: Intractability II

- More NP-completeness reductions
- Class coNP, PSPACE

Instructor: Jonathan Ullman
Lecturer: Lydia Zakynthinou
10-21-22

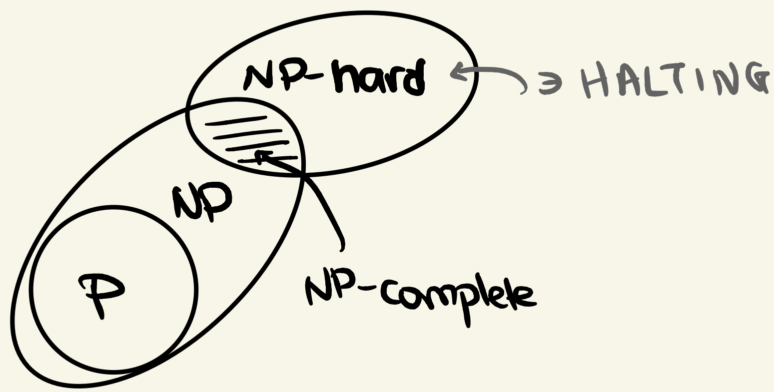
The Class NP

Non-deterministic Polynomial time

Def. NP is the class of problems for which $\exists$ an efficient certifier.
(For every "YES" instance s, $\exists$ certificate t of polynomial length w.r.t. s, such that given t and s, we can verify that s is a "YES" instance in polynomial time.)

Def. Y is NP-hard iff $\forall X \in NP \quad X \leq_p Y$

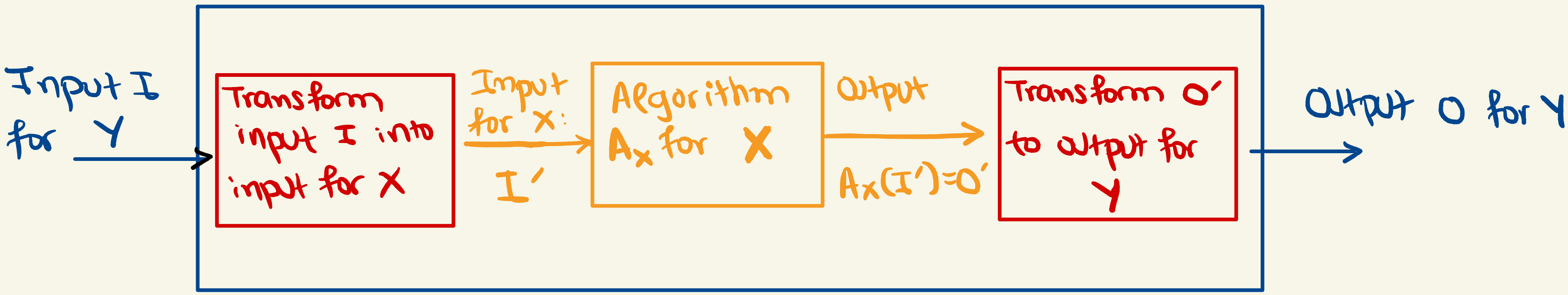
Def. Y is NP-complete iff Y is NP-hard and $Y \in NP$.



NP-complete problems

$$
\begin{tikzcd}
 & & & \text{SET COVER} \\
 & & \text{VERTEX COVER} \arrow[ur, "\le_p"] \arrow[r, "\le_p"] & \text{INTEGER LINEAR PROGRAMMING} \\
 & & \text{INDEPENDENT SET} \arrow[u, leftrightarrow, "\equiv_p"] \arrow[d, leftrightarrow, "\equiv_p"] & \\
 & & \text{CLIQUE} & \\
\begin{array}{l}\text{Any}\\ \text{problem}\\ Y \in NP\end{array} \arrow[r, "\le_p"] & \text{3-SAT} \arrow[uur, "\le_p"] & &
\end{tikzcd}
$$

Strategy to prove that X is NP-complete



(1) Prove $X \in NP$.

(2) Find problem Y that is known to be NP-complete, and prove $Y \leq_p X$ *:

(2a) Consider arbitrary input I to problem Y.

(2b) Construct a poly-time transformation of input I to a (special) instance I' of X

(2c) Prove correctness:

(i) If I is a YES instance for $Y \Rightarrow I'$ is a YES instance for X.

(ii) If I' is a YES instance for $X \Rightarrow I$ is a YES instance for Y

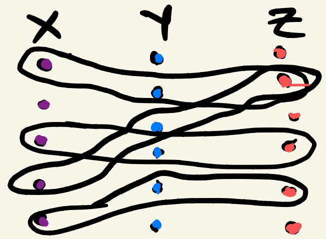
"packing", "covering", "sequencing", "partitioning", "numerical".

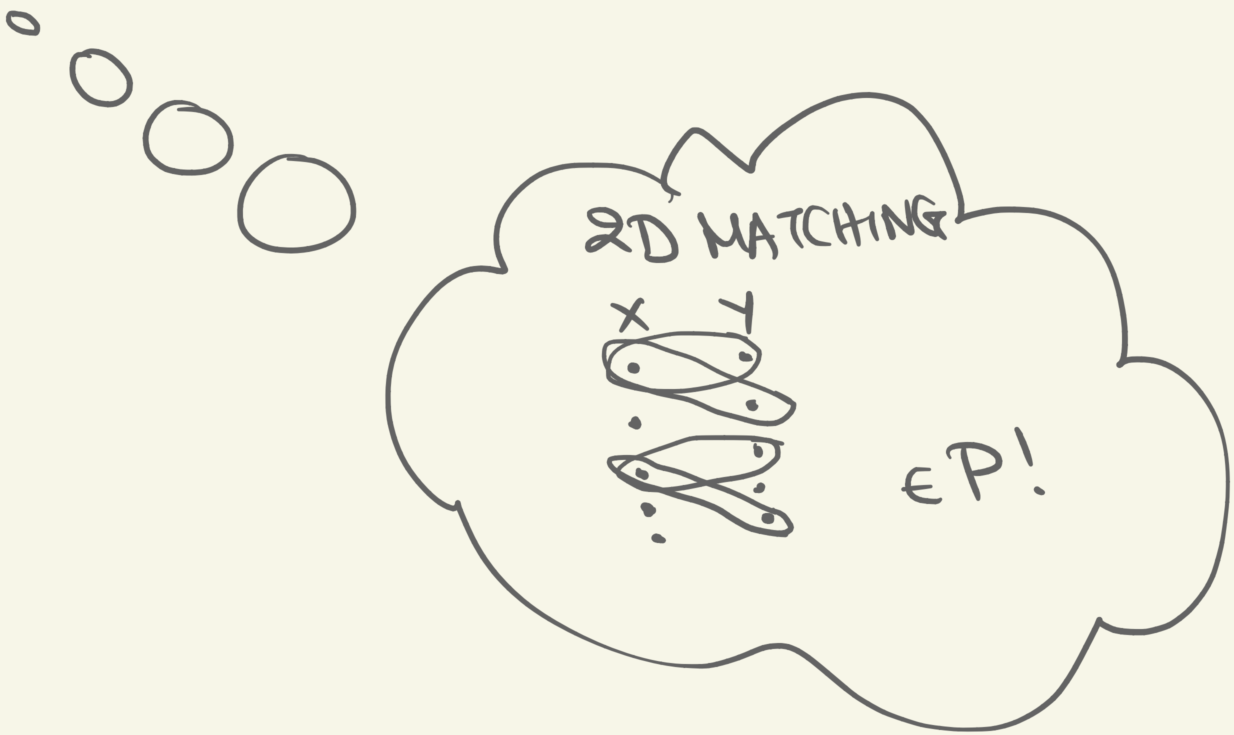
* Karp reduction. More general reductions are Cook reductions.

3D MATCHING is NP-complete

Input: Disjoint sets X, Y, Z, $|X| = |Y| = |Z| = n$. Set $T \subseteq X \times Y \times Z$ of ordered triples.

Output: YES iff $\exists$ set of n triples $S \subseteq T$ s.t. each element in $X \cup Y \cup Z$ is contained in exactly one of the triples.

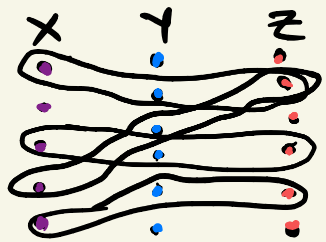




3D MATCHING is NP-complete

<u>Input</u>: Disjoint sets X, Y, Z, $|X| = |Y| = |Z| = n$. Set $T \subseteq X \times Y \times Z$ of ordered triples.

<u>Output</u>: YES iff $\exists$ set of n triples $S \subseteq T$ s.t. each element in $X \cup Y \cup Z$ is contained in exactly one of the triples.



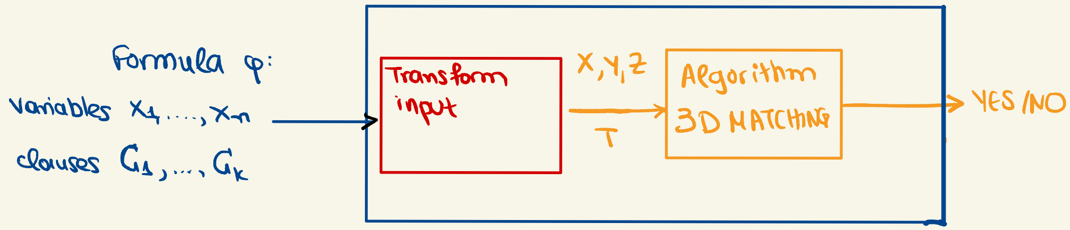
(1) 3D MATCHING $\in$ NP: A collection of n triples from T that covers every element in $X \cup Y \cup Z$ exactly once is a poly-length certificate that can be checked in poly-time.

(2) Find known NP-complete problem Y and prove $Y \leq_p$ 3D MATCHING.

3-SAT $\leq_p$ 3D MATCHING

Input: Disjoint sets X, Y, Z, $|X| = |Y| = |Z| = n$. Set $T \subseteq X \times Y \times Z$ of ordered triples.

Output: YES iff $\exists$ set of n triples $S \subseteq T$ s.t. each element in $X \cup Y \cup Z$ is contained in exactly one of the triples.



(2a,b)

Transform Input

Consider arbitrary input with n variables $x_1, \dots, x_n$ and k clauses $C_1, \dots, C_k$.

$$\text{e.g.: } \varphi = \underbrace{(\bar{x}_1 \vee x_2 \vee \bar{x}_3)}_{C_1} \wedge \underbrace{(x_1 \vee \bar{x}_2 \vee x_3)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_3)}_{C_3}$$

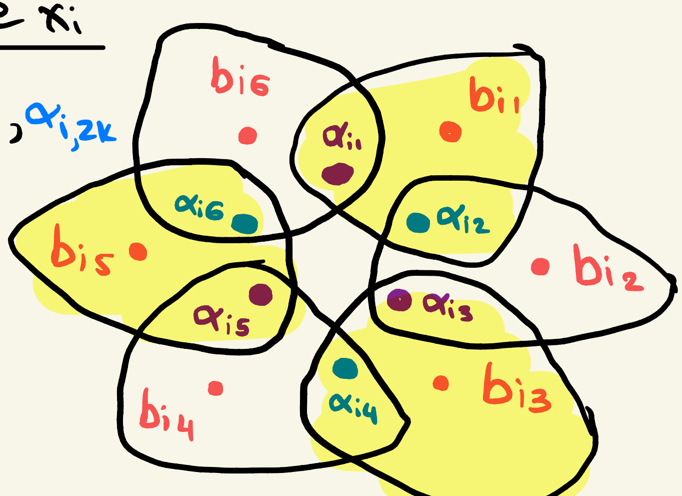
Gadget for variable x_i

$2k$ core $\quad A_i = \alpha_{i1}, \alpha_{i2}, \dots, \alpha_{i,2k}$

$2k$ tips $\quad B_i = b_{i1}, \dots, b_{i,2k}$

$2k$ "TF" triples:

$t_{ij} = (a_{ij}, a_{i,j+1}, b_{ij})$

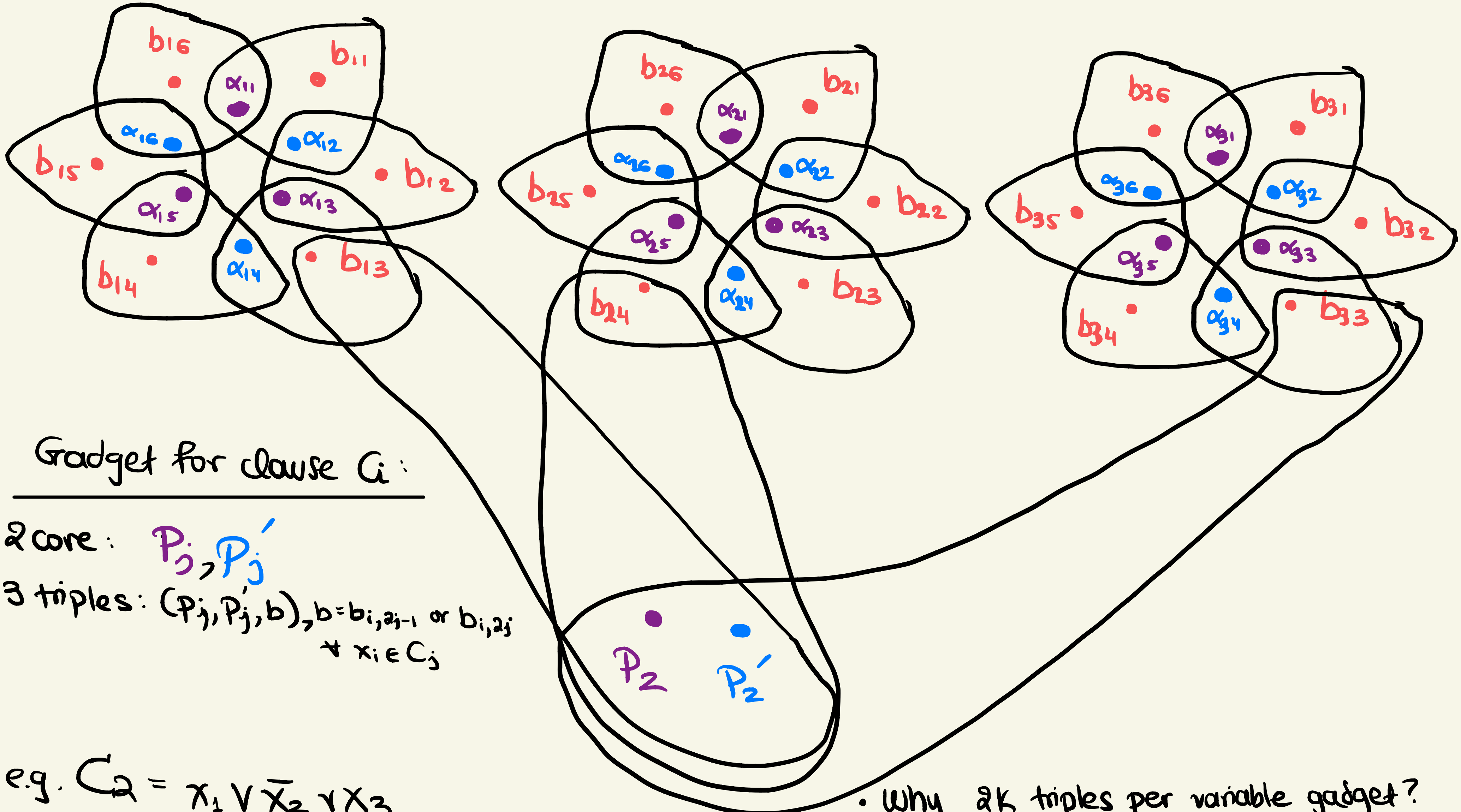


select even triples + free odd triples $\iff x_i = 1$

select odd triples + free even triples $\iff x_i = 0$

Can only cover the core exactly once if I select all odd or all even TF triples.

3-SAT $\leq_p$ 3D MATCHING



Gadget for clause C_j:

2 core: P_j, P_j'

3 triples: (P_j, P_j', b), $b = b_{i,2j-1}$ or $b_{i,2j}$ $\forall x_i \in C_j$

e.g. $C_2 = x_1 \vee \bar{x}_2 \vee x_3$

Need $x_1 = 1$ or $x_2 = 0$ or $x_3 = 1$ to satisfy C_2.

→ Connect to an odd/even/odd free tip in variable $x_1/x_2/x_3$'s gadget.

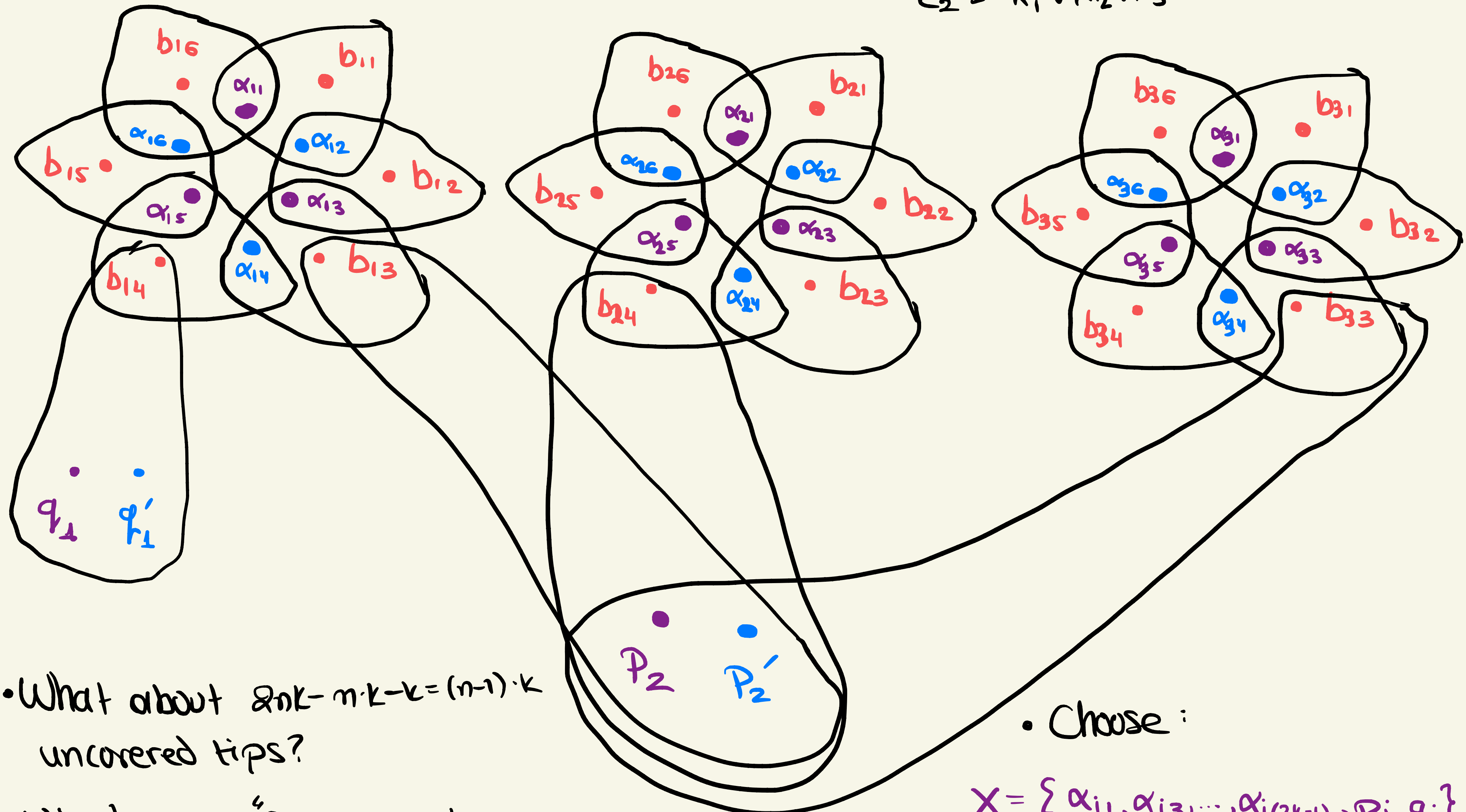
- Why 2k triples per variable gadget?

↳ So all clauses can connect to odd/even free tips of same variable.

Can only cover the core by selecting triple which includes free tip (setting variable to 0/1).

3-SAT $\leq_p$ 3D MATCHING

$C_2 = x_1 \vee \bar{x}_2 \vee x_3$



- What about $2nk - n \cdot k - k = (n-1) \cdot k$ uncovered tips?

Add dummy "cleanup" gadgets

$(q_\ell, q'_\ell, b_{ij})$ $\forall b_{ij}$, $\forall \ell \in [(n-1)k]$

- Choose:

$X = \{\alpha_{i1}, \alpha_{i3}, \ldots, \alpha_{i(2k-1)}, p_i, q_i\}$

$Y = \{\alpha_{i2}, \ldots, \alpha_{i,2k}, p'_i, q'_i\}$

$Z = \{b_{ij} \ \forall i \in [n], j \in [2k]\}$

T = "TF" + "clause" + "cleanup" triples

3-SAT $\leq_p$ 3D MATCHING

Our transformation is polynomial time. ✓

(2c) Remains to prove the transformation is correct:

- If φ has a satisfying assignment, X, Y, Z, T have a perfect 3D Matching:
  Given sat. assignment, choose odd/even "TF" triples based on values of the assignment on each $x_i = 0/1$ $\forall i \in [n]$. The core elements A_i, B_i $\forall i \in [n]$ are covered exactly once.
  Choose on triple per clause corresponding to free tip of variable that makes it True ($\exists \geq 1$). The core elements p_j, p_j' $\forall j \in [k]$ are covered exactly once.
  Choose $(n-1) \cdot k$ "cleanup" triples that correspond to $(n-1) \cdot k$ uncovered tips to cover them and the core of the cleanup triples, q_ℓ, q_ℓ' $\forall \ell \in [(n-1)k]$

- If X, Y, Z, T as defined above have a perfect 3D Matching, then φ is satisfiable.
  Setting variable $x_i = 0/1$ based on whether odd/even "TF" triples were selected in the matching results in a satisfying assignment.
  Why? Clause cores p_j, p_j' must be covered so mathing must include one triple with a free tip from one of the gadgets of the variables involved in the clause. For this tip to be free, it must have been that the variable has been set to a value that satisfies the clause.

SUBSET SUM is NP-complete

Input: $w_1, \dots, w_n \in \mathbb{N}$. Target W.

Output: YES iff $\exists$ subset $S \subseteq [n]$ such that $\sum_{i \in S} w_i = W$.

(1) SUBSET SUM $\in$ NP. ✓

(2) Find NP-complete problem Y and prove $Y \leq_p$ SUBSET SUM.

3D MATCHING $\leq_p$ SUBSET SUM

Input: $w_1, \dots, w_n \in \mathbb{N}$. Target W.

Output: YES iff $\exists$ subset S of $\{w_1, \dots, w_n\}$ that adds up to W.

(2ab) **Transform input**

Consider X, Y, Z, $|X| = |Y| = |Z| = n$ and m triples $T \subseteq X \times Y \times Z$.

$\forall$ triple $t = (x_i, y_j, z_k)$ $i, j, k \in [n] \times [n] \times [n]$, construct $3n$-bit vector with 1 in position $i, n+j, 2n+k$

$$w_t = \overbrace{\underset{z_n \dots z_k \dots z_1}{0 \dots 1 \dots 0}}^{Z} \;\Big|\; \overbrace{\underset{y_n \dots y_j \dots y_1}{0 \dots 1 \dots 0}}^{Y} \;\Big|\; \overbrace{\underset{x_n \dots x_i \dots x_1}{0 \dots 1 \dots 0}}^{X}$$

$$w_t = d^{i-1} + d^{n+j-1} + d^{2n+k-1} \quad \text{for base } d > 1$$

Union of triples $\approx$ addition where each position's digit corresponds to the number of times that element has been included in a triple... if addition does not incur any carries.

3D MATCHING $\leq_p$ SUBSET SUM

(2a,b) Transform input

Consider X, Y, Z, $|X| = |Y| = |Z| = n$ and m triples $T \subseteq X \times Y \times Z$.

$\forall$ triple $t = (x_i, y_j, z_k)$ $i,j,k \in [n]\times[n]\times[n]$, construct $3n$-bit vector with 1 in position $i, n+j, 2n+k$

$$w_t = \overbrace{0\ldots1\ldots\ldots0}^{Z} \mid \overbrace{0\ldots\ldots1\ldots0}^{Y} \mid \overbrace{0\ldots1\ldots0}^{X}$$

$$ z_n \ldots z_k \ldots\ldots z_1 \mid y_n \ldots\ldots y_j \ldots y_1 \mid x_n \ldots x_i \ldots x_1$$

$$w_t = d^{i-1} + d^{n+j-1} + d^{2n+k-1}$$ for base $d > 1$.

Choose base $d = m+1$ (no carries even if an element is included in all m triples and we select (add) all of them).

Choose $$W = \sum_{i=0}^{3n-1} (m+1)^i \longrightarrow W = \overbrace{\underbrace{11\ldots\ldots1}_{n}}^{Z} \mid \overbrace{\underbrace{1\ldots\ldots1}_{n}}^{Y} \mid \overbrace{\underbrace{1\ldots\ldots1}_{n}}^{X}$$

each element selected exactly once.

- Our transformation is polynomial time.

3D MATCHING $\leq_p$ SUBSET SUM

(2c) Remains to prove our reduction is correct.

- If $\exists$ perfect 3D MATCHING (triples $t_1, \dots, t_n$) then $\exists$ numbers $(w_{t_1}, \dots, w_{t_n})$ that have a 1 in every position so $\sum_{i=1}^{n} w_{t_i} = W$.

- If $\exists$ $w_{t_1} + \dots + w_{t_k} = W$, then it must be that $k = n$ and each position has a 1 so each element is covered by 1 triple (not more because we would have $j > 1$ in that position).

∎

value(i) = size(i)

SUBSET SUM is also a special case of KNAPSACK (SUBSET SUM $\leq_p$ KNAPSACK)

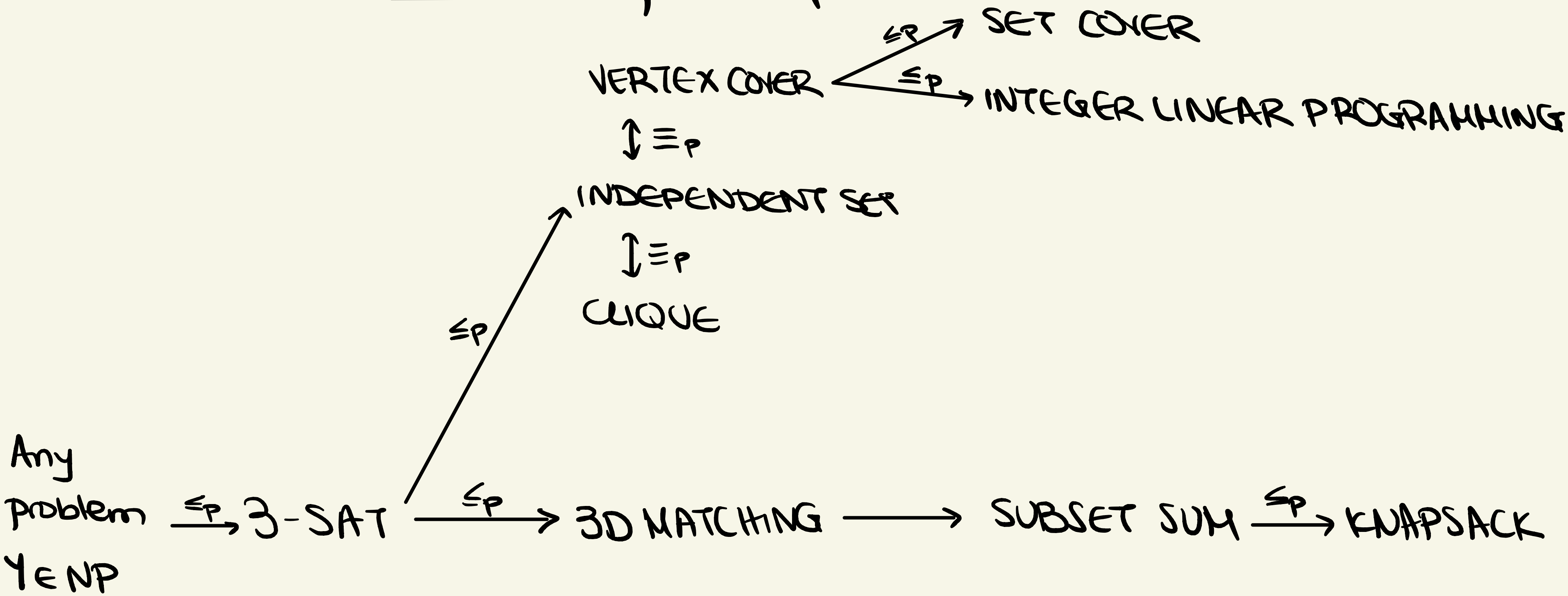
$\Rightarrow$ KNAPSACK is also NP-hard!

But also we have an algorithm that runs in $O(n \cdot W)$ time...

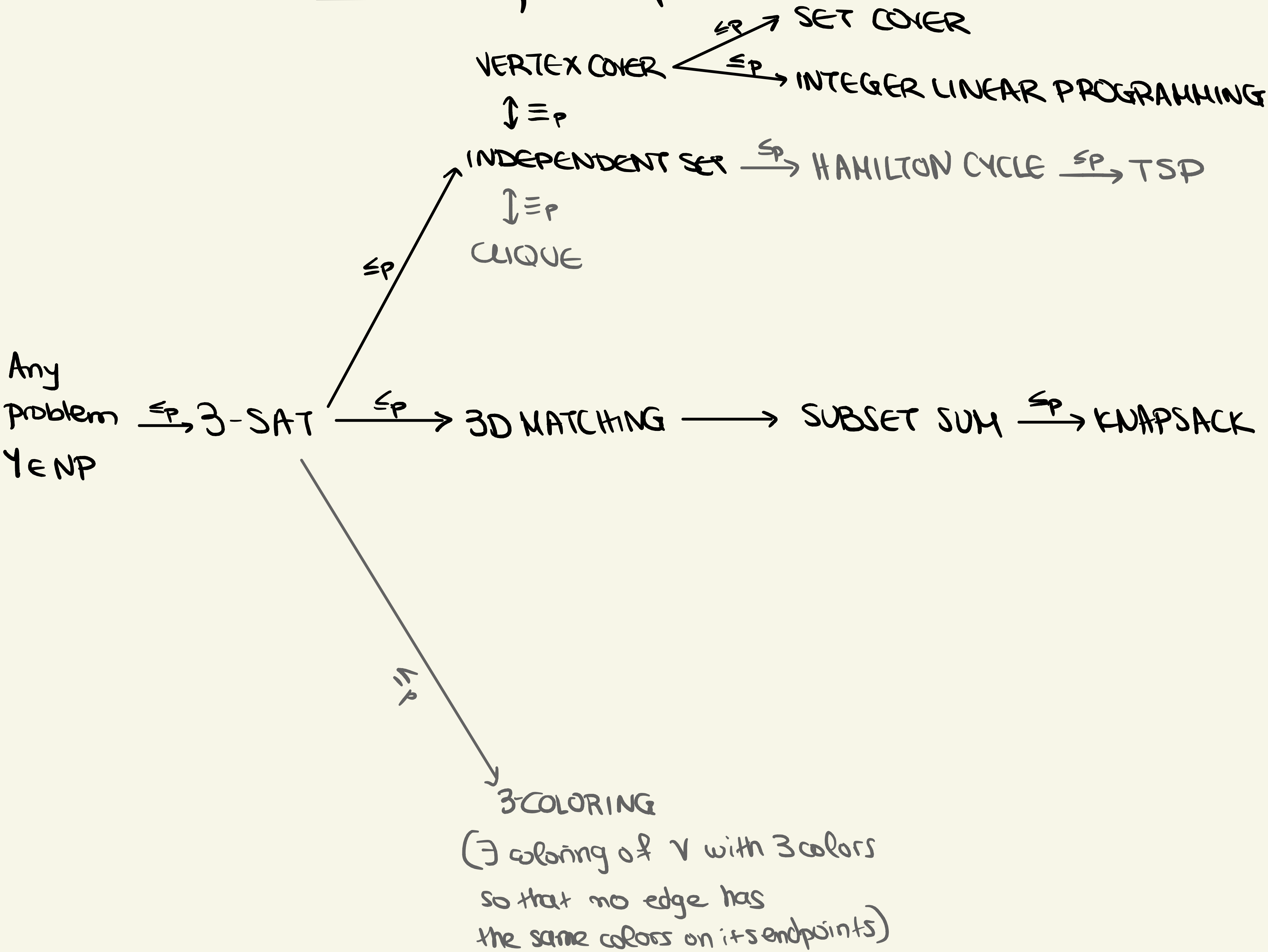
e.g. $n \cdot \log W$ e.g. W

<u>Def.</u> Algorithms whose running time depends on a quantity of the problem that could be exponentially larger than the length of the input are pseudo-polynomial time.
NP-hard problems with such an algorithm are weakly NP-hard.

NP-Complete problems



NP-Complete problems



The Class co-NP

Non-deterministic polynomial time

Def. NP is the class of problems for which $\exists$ an efficient certifier.

Def. Algorithm B is an efficient certifier for problem X if:

1. It is a polynomial time algorithm that takes input s and certificate t.
2. $\exists$ polynomial p so that $s \in X$ (YES instance) iff $\exists t$ with length $|t| \leq p(|s|)$ for which $B(s,t) = \text{YES}$.

Asymmetric definition: $s \notin X$ (NO instance) iff $\forall$ short t $B(s,t) = \text{NO}$.

Def. co-NP is the class of all problems X, whose complementary problem $\bar{X} \in NP$.

$$\text{complementary problem} \;\Vert\; s \in X \Leftrightarrow s \notin \bar{X}$$

The Class co-NP

Def. co-NP is the class of all problems X, whose complementary problem $\bar{X} \in NP$.

$s \in X \Leftrightarrow s \notin \bar{X}$

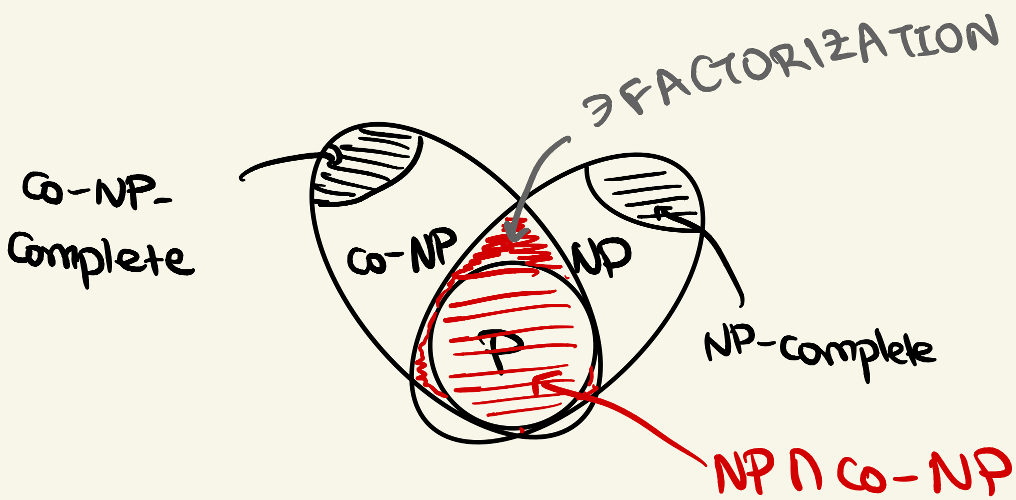
Obs. $X \in P \Leftrightarrow \bar{X} \in P$.

But we don't know $\boxed{\text{co-NP} \overset{?}{=} NP}$

Obs. $P \subseteq \underbrace{NP \cap \text{co-NP}}_{}$. But we don't know $\boxed{P \overset{?}{=} NP \cap \text{co-NP}.}$

problems that have "good characterizations"

e.g. Bipartite matching, Max Flow (both $\in P$)



The Class PSPACE

Def. PSPACE is the class of all problems that can be solved by an algorithm that uses an amount of space polynomial in the size of the input.

Obs. 1. P $\subseteq$ PSPACE. But we don't know $\boxed{P \stackrel{?}{=} PSPACE}$.

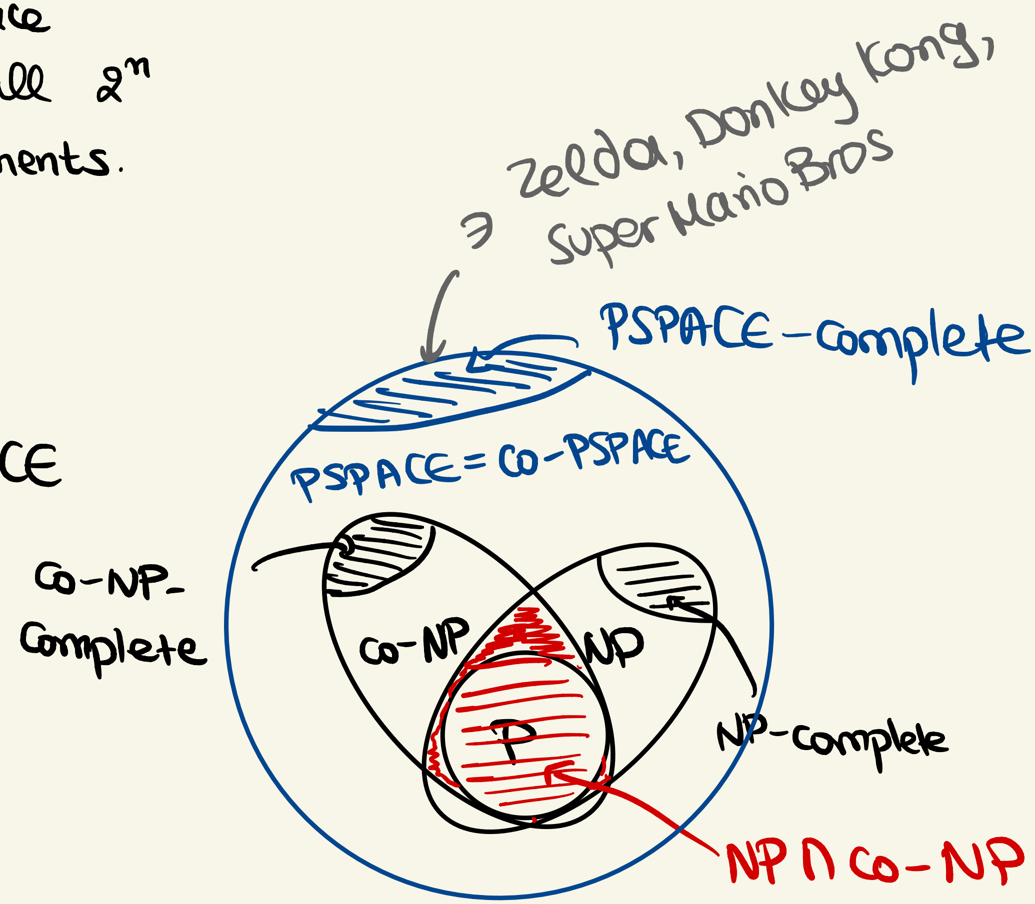
Obs. 2 3-SAT $\in$ PSPACE

x_1 x_2 ... x_n

Reuse space to try all 2^n assignments.

$\Rightarrow$ NP $\subseteq$ PSPACE

Obs. 3. $\overline{\text{3-SAT}} \in$ PSPACE $\Rightarrow$ co-NP $\in$ PSPACE



Theorem (Stockmeyer, Meyer '73)

QSAT is PSPACE-complete.

$\exists x_1 \forall x_2 \exists x_3 \ldots \forall x_n \; \varphi(x_1, \ldots, x_n) = 1$?

Are we doomed?

If $P \neq NP$, then a lot of interesting problems do not have a poly-time algorithm.

We can still ask for polynomial time algorithms:

↳ Approximation algorithms

↳ Randomized algorithms

We can also ask for algorithms that use small amount of memory:

↳ Sketching /Streaming algorithms