NP Hardness Reductions

Health Days This Week!

12Apr Flow Applications Reading: KT §7.6 | E §11	13Apr	14Apr P vs NP and NP-hardness Reading: KT §8.1, 8.3 | E §12.1–12.5 Assignment 7 out Assignment 6 due	15Apr	16Apr Problem Reductions Reading: KT §8.1, 8.3 | E §12.1–12.5
19Apr NP-hard Reductions Reading: Reading: KT §8.2, 8.4 | E §12.6–12.8	20Apr	21Apr Health Day	22Apr Health Day	23Apr Intractability Wrap Up Reading: KT §8.5–8.7; | E §12.6–12.8

You are here

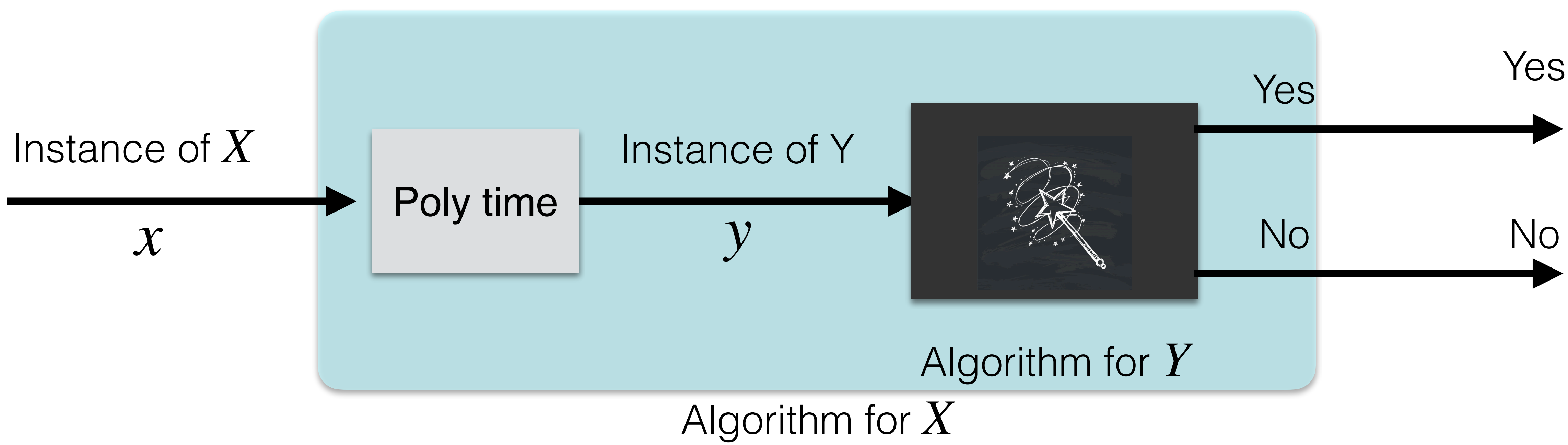
Rest and sunshine is here!

Reminders & Leftovers

- Assignment 7: aim to finish off the first 3 questions by tomorrow
- Please fill out TA survey! (**Only 5 out of 40** students filled it out)
- Reduction recordings available from CS256-S20
  - Linked in GLOW; use as readings or for review
  - Can find in Course Media Gallery-> Pre-recorded lectures
- Reduction from **Graph-2-Color** to **Graph-3-Color**:
  - Input G: graph whose nodes we are trying to color with 2 colors
  - Create G' be the graph with all the nodes and edges as G plus 3 more nodes: r, g, b in a triangle with an edge (r, v) $\forall v \in V$
  - G has a valid 2 coloring if and only if G' has a valid 3 coloring

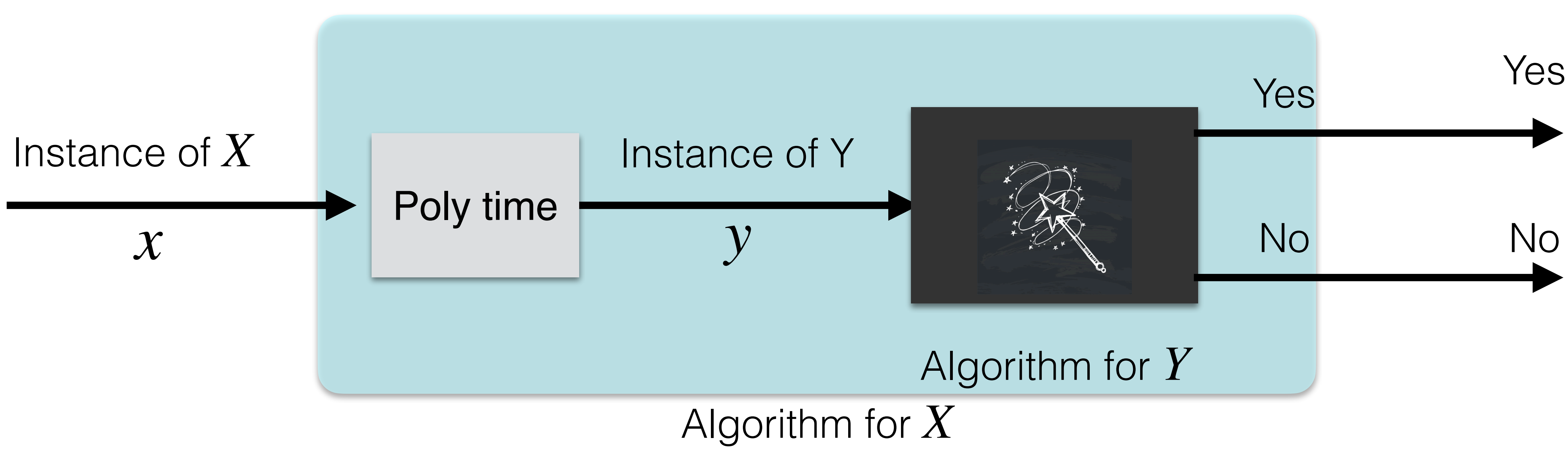
Reductions: General Pattern

- Describe a polynomial-time algorithm to transform an arbitrary instance x of Problem X into a special instance y of Problem Y
- Prove that:
  - If x is a "yes" instance of X, then y is a "yes" instance of Y
  - If y is a "yes" instance of Y, then x is a "yes" instance of X



Reductions: General Pattern

- Describe a polynomial-time algorithm to transform an arbitrary instance x of Problem X into a special instance y of Problem Y
- Notice that correctness of reductions are not symmetric:
    - the “if” proof needs to handle arbitrary instances of X
    - the “only if” needs to handle the special instance of Y



IND-SET is NP Complete:

$\text{3SAT} \leq_p \text{IND-SET}$

Problem Definition: 3-SAT

- **Literal**. A Boolean variable or its negation x_i or $\overline{x_i}$
- **Clause**. A disjunction of literals $C_j = x_1 \vee \overline{x_2} \vee x_3$
- **Conjunctive normal form (CNF).** A boolean formula ϕ that is a conjunction of clauses $\Phi = C_1 \wedge C_2 \wedge C_3$
- **SAT**. Given a CNF formula Φ, does it have a satisfying truth assignment?
- **3SAT.** A SAT formula where each clause contains exactly 3 literals (corresponding to different variables)
- $\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$
- **SAT**, **3SAT** are both NP complete
- We will use 3SAT to prove other problems are NP hard

IND-SET

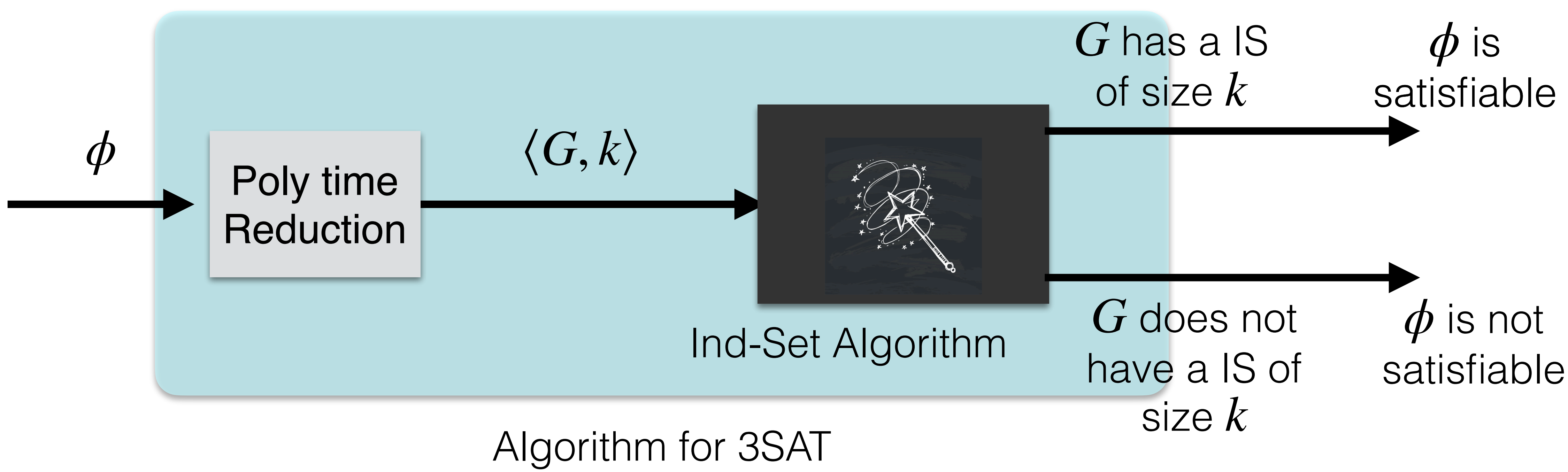
- Given a graph $G = (V, E)$, an independent set is a subset of vertices $S \subseteq V$ such that no two of them are adjacent, that is, for any $x, y \in S$, $(x, y) \notin E$
- IND-SET Problem.
  Given a graph $G = (V, E)$ and an integer k, does G have an independent set of size at least k?

IND-SET: NP Complete

- To show Independent set is NP complete
    - Show it is in NP (already did in previous lectures)
    - Reduce a known NP complete problem to it
        - We will use 3-SAT
- Looking ahead: once we have shown 3-SAT $\leq_p$ IND-SET
    - Since **IND-SET** $\leq_p$ **Vertex Cover**
    - And **Vertex Cover** $\leq_p$ **Set Cover**
    - We can conclude they are also NP hard
    - As they are both in NP, they are also NP complete!

IND-SET: NP hard

- **Theorem.** 3-SAT $\leq_p$ IND-SET
- Given an instance Φ of 3-SAT, we construct an instance $\langle G, k \rangle$ of IND-SET s.t. G has an independent set of size k iff ϕ is satisfiable.



Map the Problems

3SAT	What is a possible solution?	**Ind-Set**
An assignment of T/F to variables		A selection of vertices to be an IS S

What is the requirement?

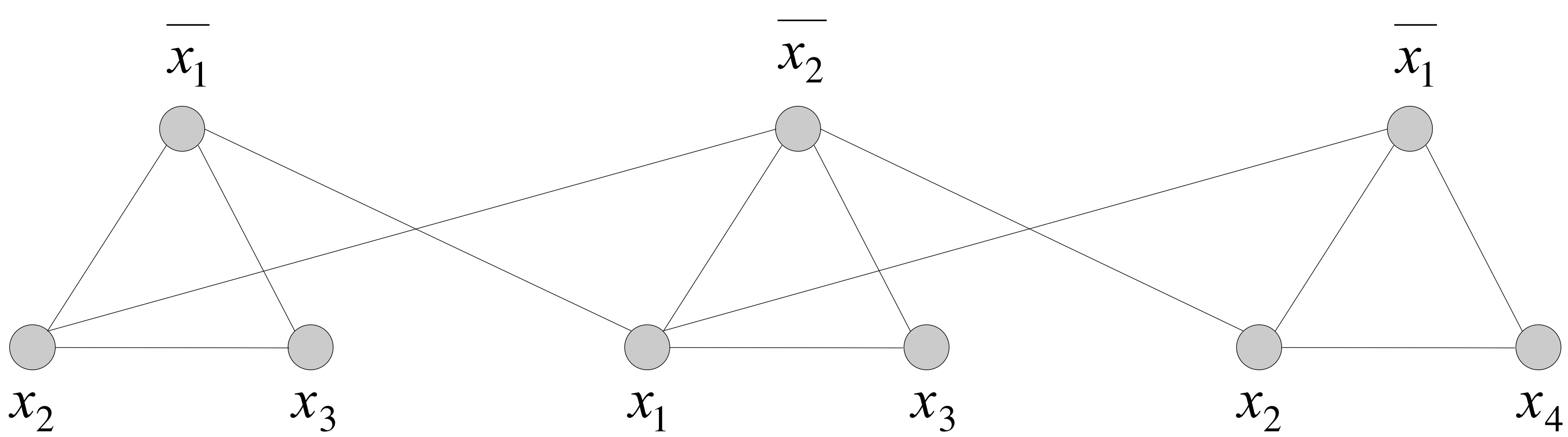
Each clause must contain at least one literal that is True	S must contain at least k vertices

What are the restrictions?

x can be true iff $\bar{x}$ is assigned false	If $(u, v) \in E$, then both u and v cannot be in S

3SAT$\leq_p$ IND-SET

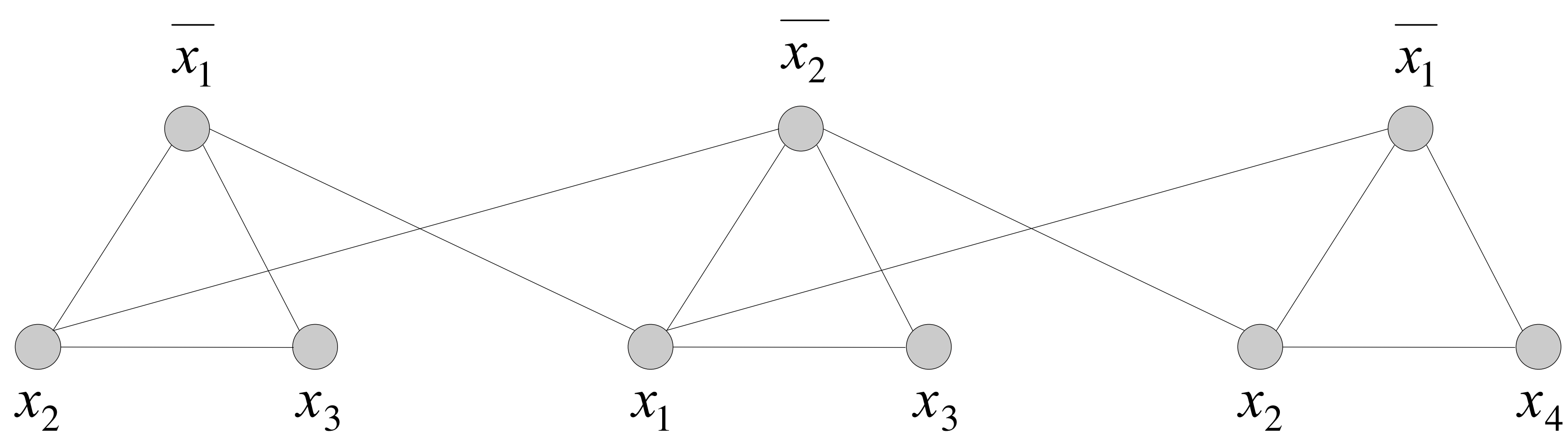
- **Reduction.** Let k be the number of clauses in Φ.
  - G has $3k$ vertices, one for each literal in Φ
  - (Clause gadget) For each clause, connect the three literals in a triangle
  - (Variable gadget) Each variable is connected to its negation



$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

3SAT $\leq_p$ IND-SET

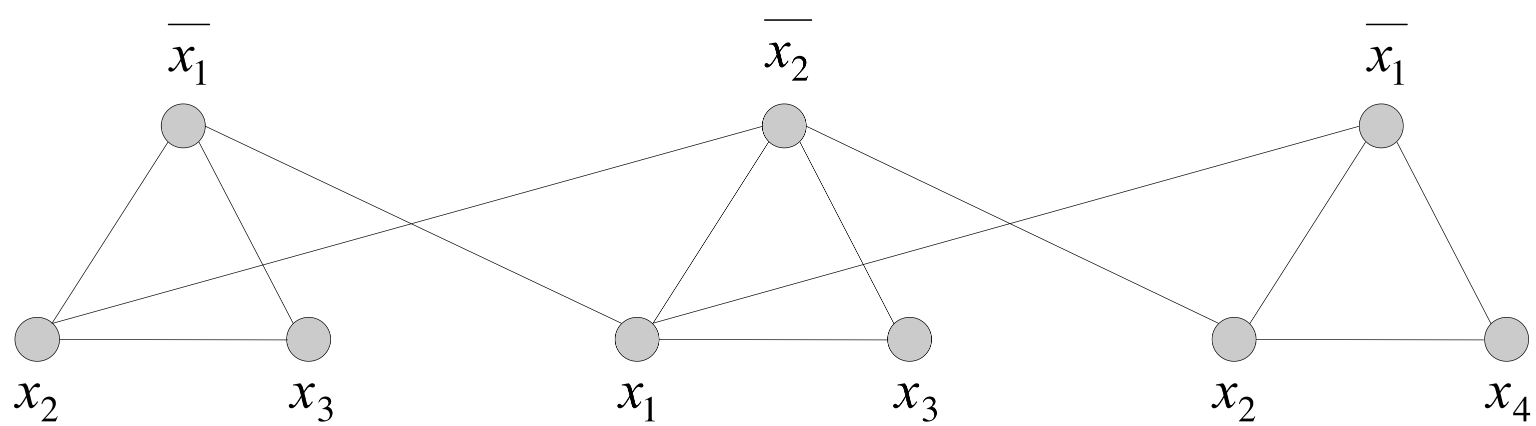
- **Observations.**
  - Any independent set is G can contain at most 1 vertex from each clause triangle
  - Only one of x_i or $\overline{x_i}$ can be in an independent set (*consistency*)



$$\Phi = \left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(\overline{x_1} \vee x_2 \vee x_4\right)$$

3SAT$\leq_p$ IND-SET

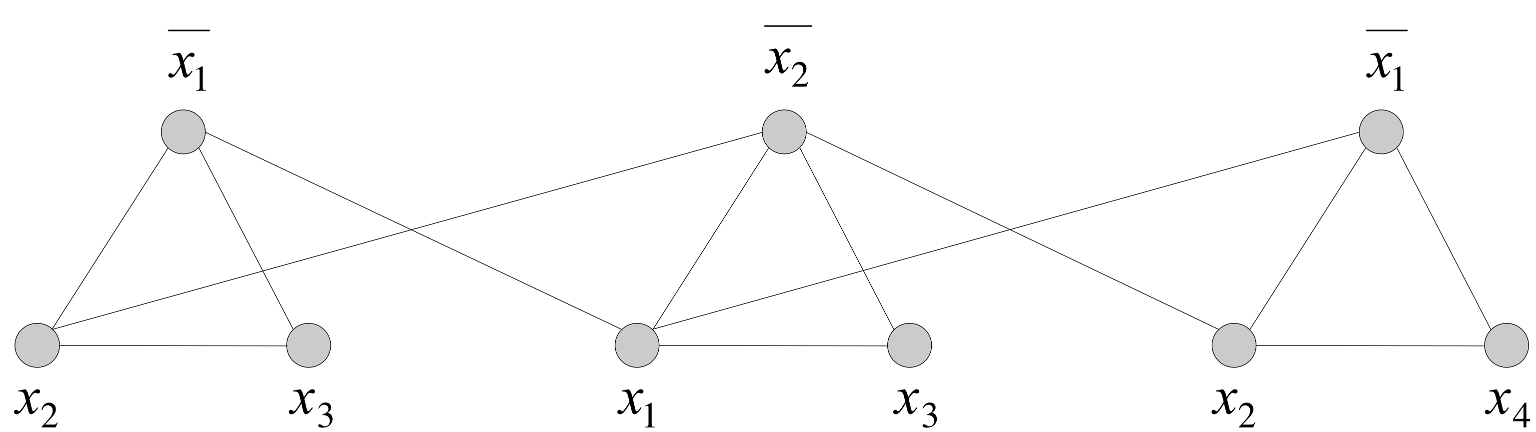
- **Claim.** Φ is satisfiable iff G has an independent set of size k
- $(\Rightarrow)$ Suppose Φ is satisfiable, consider a satisfying assignment
    - There is at least one true literal in each clause
    - Select one true literal from each clause/triangle
    - This is an independent set of size k



$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

3SAT $\leq_p$ IND-SET

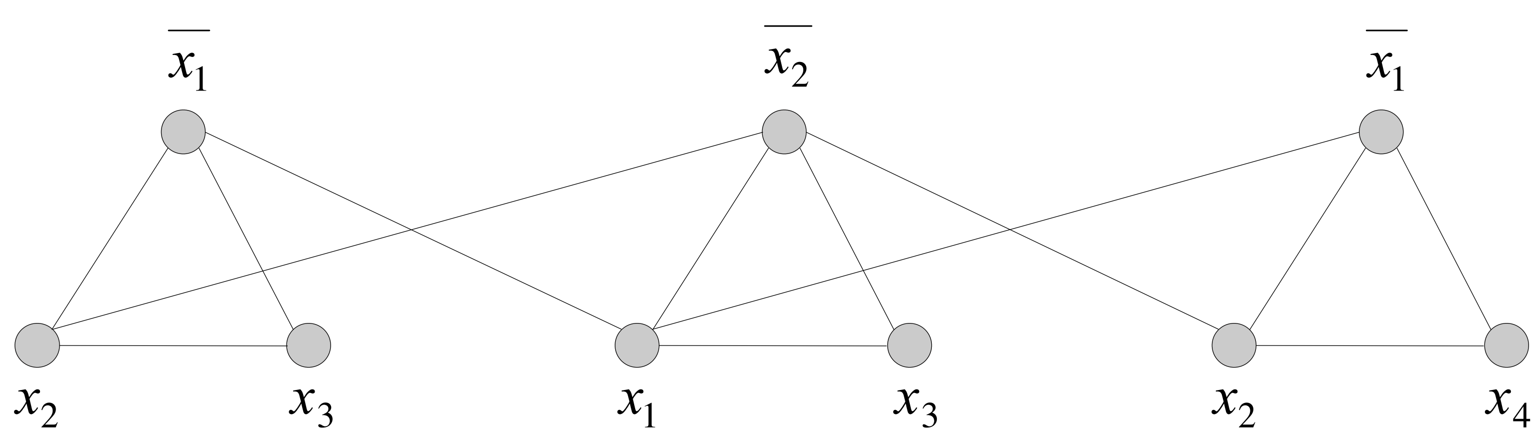
- **Claim.** Φ is satisfiable iff G has an independent set of size k
- $(\Leftarrow)$ Let S be in an independent set in G of size k
  - S must contain exactly one node in each triangle
  - Set the corresponding literals to *true*
  - Set remaining literals consistently
  - All clauses are satisfied — Φ is satisfiable ∎



$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

3SAT$\leq_p$ IND-SET

- Our reduction is clearly polynomial time in the input
  - G has $3k$ nodes, where k is #clauses, and n edges (one for each variable in G)
- Thus, independent is NP hard
- Since independent set is in NP (shown previously)
  - Independent set is NP complete



$$\Phi = \left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(\overline{x_1} \vee x_2 \vee x_4\right)$$

Reduction Strategies

- Equivalence
  - **VERTEX-COVER** $\equiv_p$ **IND-SET**
- Special case to general case
  - **VERTEX-COVER** $\leq_p$ **SET-COVER**
- Encoding with gadgets
  - **3-SAT** $\leq_p$ **IND-SET**
- Transitivity
  - **3-SAT** $\leq_p$ **IND-SET** $\leq_p$ **VERTEX-COVER** $\leq_p$ **SET-COVER**
  - Thus, **IND-SET**, **VERTEX-COVER** and **SET-COVER** are NP hard
  - Since they are all in NP, also NP - complete

List of NPC Problems So Far

- **3-SAT**
- **INDEPENDENT SET**
- **VERTEX COVER**
- **SET COVER**
- **CLIQUE**
- More to come:
    - **Subset Sum**
    - **Knapsack**
    - **3-COLOR**
    - **Hamiltonian cycle / path**
    - **TSP**

SUBSET-SUM is NP Complete:

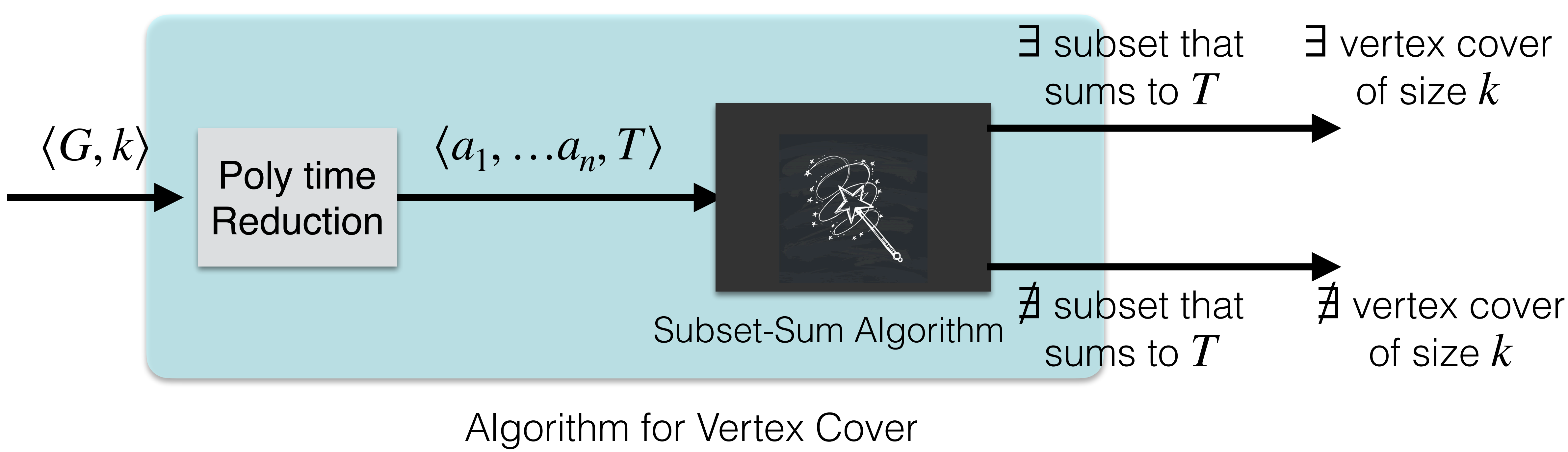
Vertex-Cover $\leq_p$ SUBSET-SUM

Subset Sum Problem

- **SUBSET-SUM.**
  Given n positive integers $a_1, \ldots, a_n$ and a target integer T, is there a subset of numbers that adds up to exactly T
- **SUBSET-SUM** $\in$ NP
  - Certificate: a subset of numbers
  - Poly-time verifier: checks if subset is from the given set and sums exactly to T
- Problem has a pseudo-polynomial $O(nT)$-time dynamic programming algorithm similar to Knapsack
- Will prove **SUBSET-SUM** is **NP hard:** reduction from vertex cover

Vertex Cover to Subset Sum

- **Theorem.** VERTEX-COVER $\leq_p$ SUBSET-SUM
- Proof. Given a graph G with n vertices and m edges and a number k, we construct a set of numbers $a_1, \ldots, a_t$ and a target sum T such that G has a vertex cover of size k iff there is a subset of numbers that sum to T



Map the Problems

Vertex Cover		Subset Sum
	What is a possible solution?	
A selection of vertices to be in VC C		A selection of numbers in subset S
	What is the requirement?	
C must contain at most k vertices		numbers in S must sum to T
	What are the restrictions?	
If $(u, v) \in E$, then either u or v must be in S		S must be a subset of input integers

Vertex Cover to Subset Sum

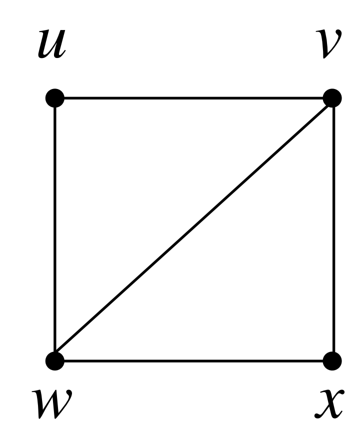
- **Theorem.** VERTEX-COVER $\leq_p$ SUBSET-SUM
- **Proof.** Label the edges of G as $0, 1, \ldots, m - 1$.
- **Reduction**.
    - We'll create one integer for every vertex, and one integer for every edge
    - Force selection of k vertex integers: so will make sure that we can't sum to T unless we have that
    - Force edge covering: for every edge (u, v), we will force that number can't sum to T unless either u or v is picked

Vertex Cover to Subset Sum

- **Theorem.** VERTEX-COVER $\leq_p$ SUBSET-SUM
- Label the edges of G as $0,1,\ldots,m-1$.
- **Reduction**. Create $n+m$ integers and a target value T as follows
- Each integer is a $m+1$-bit number in base four
- Vertex integer a_v : mth (most significant) bit is 1 and for $i<m$, the ith bit is 1 if ith edge is incident to vertex v
- Edge integer b_{uv} : mth digit is 0 and for $i<m$, the ith bit is 1 if this integer represents an edge $i=(u,v)$
- Target value $T = k \cdot 4^m + \sum_{i=0}^{m-1} 2 \cdot 4^i$

Vertex Cover to Subset Sum

- Example: consider the graph $G = (V, E)$ where $V = \{u, v, w, x\}$ and $E = \{(u, v),\ (u, w),\ (v, w),\ (v, x),\ (w, x)\}$



	5th	4th : (wx)	3rd : (vx)	2nd : (vw)	1st : (uw)	0th: (uv)
a_u	1	0	0	0	1	1
a_v	1	0	1	1	0	1
a_w	1	1	0	1	1	0
a_x	1	1	1	0	0	0
b_{uv}	0	0	0	0	0	1
b_{uw}	0	0	0	0	1	0
b_{vw}	0	0	0	1	0	0
b_{vx}	0	0	1	0	0	0
b_{wx}	0	1	0	0	0	0

$$a_u := 111000_4 = 1344$$
$$a_v := 110110_4 = 1300$$
$$a_w := 101101_4 = 1105$$
$$a_x := 100011_4 = 1029$$

$$b_{uv} := 010000_4 = 256$$
$$b_{uw} := 001000_4 = 64$$
$$b_{vw} := 000100_4 = 16$$
$$b_{vx} := 000010_4 = 4$$
$$b_{wx} := 000001_4 = 1$$

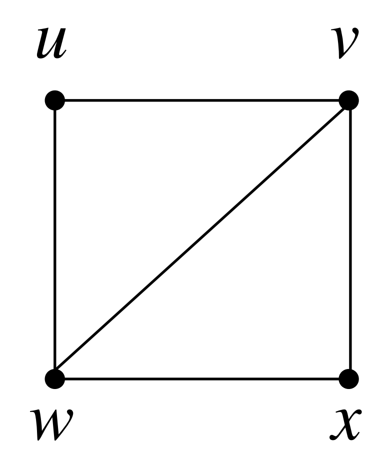
- If $k = 2$ then $T = 222222_4 = 2730$

Correctness

- **Claim.** G has a vertex cover of size k if and only there is a subset X of corresponding integers that sums to value T
- ($\Rightarrow$) Let C be a vertex cover of size k in G, define X as
  $X := \{a_v \mid v \in C\} \cup \{b_i \mid \text{edge } i \text{ has exactly one endpoint in } C\}$

	5^{th}	4^{th} : (wx)	3^{rd} : (vx)	2^{nd}: (vw)	1^{st} : (uw)	0^{th}: (uv)
a_u	1	0	0	0	1	1
a_v	1	0	1	1	0	1
a_w	1	1	0	1	1	0
a_x	1	1	1	0	0	0
b_{uv}	0	0	0	0	0	1
b_{uw}	0	0	0	0	1	0
b_{vw}	0	0	0	1	0	0
b_{vx}	0	0	1	0	0	0
b_{wx}	0	1	0	0	0	0

$C = \{v, w\}$



$T = 222222_4 = 2730$

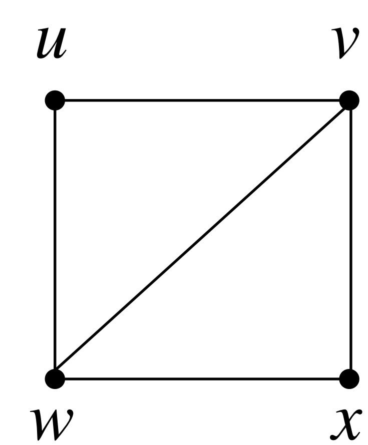
$$T = k \cdot 4^m + \sum_{i=0}^{m-1} 2 \cdot 4^i$$

Correctness

- **Claim.** G has a vertex cover of size k if and only there is a subset X of corresponding integers that sums to value T
- $(\Rightarrow)$ Let C be a vertex cover of size k in G, define X as
  $X := \{a_v \mid v \in C\} \cup \{b_i \mid$ edge i has exactly one endpoint in $C\}$

	5^{th}	4^{th}: (wx)	3^{rd}: (vx)	2^{nd}: (vw)	1^{st}: (uw)	0^{th}: (uv)
[redacted]						
a_v	1	0	1	1	0	1
a_w	1	1	0	1	1	0
[redacted]						
b_{uv}	0	0	0	0	0	1
b_{uw}	0	0	0	0	1	0
[redacted]						
b_{vx}	0	0	1	0	0	0
b_{wx}	0	1	0	0	0	0

$$C = \{v, w\}$$



$$T = 222222_4 = 2730$$

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

Correctness

- **Claim.** G has a vertex cover of size k if and only there is a subset X of corresponding integers that sums to value T
- $(\Rightarrow)$ Let C be a vertex cover of size k in G, define X as
  $X := \{a_v \mid v \in C\} \cup \{b_i \mid$ edge i has exactly one endpoint in $C\}$
- Sum of the most significant bits of X is k
- All other bit must sum to 2, why?
- Consider column for edge (u, v):
  - Either both endpoints are in C, then we get two 1's from a_v and a_u and none from b_{uv}
  - Exactly one endpoint is in C: get 1 bit from b_{uv} and 1 bit from a_u or a_v
- Thus the elements of X sum to exactly T

Vertex Cover to Subset Sum

- **Claim.** G has a vertex cover of size k if and only there is a subset X of corresponding integers that sums to value T
- $(\Leftarrow)$ Let X be the subset of numbers that sum to T
- That is, there is $V' \subseteq V, E' \subseteq E$ s.t.

$$X := \sum_{v \in V'} a_v + \sum_{i \in E'} b_i = T = k \cdot 4^m + \sum_{i=0}^{m-1} 2 \cdot 4^i$$

- These numbers are base 4 and there are no carries
- Each b_i only contributes 1 to the ith digit, which is 2
- Thus, for each edge i, at least one of its endpoints must be in V'
    - V' is a vertex cover
- Size of V' is k: only vertex-numbers have a 1 in the mth position

Subset Sum: Final Thoughts

- Polynomial time reduction?
    - $O(nm)$ since we check vertex/edge incidence for each vertex/edge when creating $n + m$ numbers
- Does a $O(nT)$ subset-sum algorithm mean vertex cover can be solved in polynomial time?
    - No! $T \approx 4^m$
- NP hard problems that have pseudo-polynomial algorithms are called *weakly NP hard*

Steps to Prove X is NP Complete

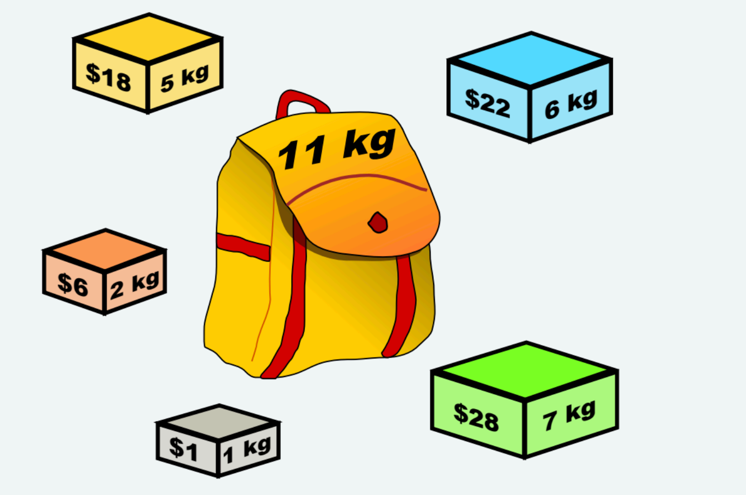
- Step 1. Show X is in **NP**
- Step 2. Pick a known NP hard problem Y from class
- Step 3. Show that $Y \leq_p X$
  - Show both sides of reduction are correct: if and only if directions
  - State that reduction runs in polynomial time in input size of problem Y

Class Exercise:

SUBSET-SUM $\leq_p$ Knapsack

Subset Sum to Knapsack

- **Knapsack**. Given n elements $a_1, \ldots, a_n$ where each element has a weight $w_i \geq 0$ and a value $v_i \geq 0$ and target weight W and value K. Does there exist a subset X of numbers such that
  - $\sum_{a_i \in X} w_i \leq W$
  - $\sum_{a_i \in X} v_i \geq K$



Creative Commons Attribution-Share Alike 2.5
by Dake

- **Knapsack** $\in$ NP
  - Can check if given subset satisfies the above conditions
- **Exercise.** Show **Subset-Sum** $\leq_p$ **Knapsack.**

Subset Sum to Knapsack

- **Knapsack.** Given n elements $a_1, \ldots, a_n$ where each element has a weight $w_i \geq 0$ and a value $v_i \geq 0$ and target weight W and value K. Does there exist a subset X of numbers such that
  - $\sum_{a_i \in X} w_i \leq W$ and $\sum_{a_i \in X} v_i \geq K$
- **Subset-Sum $\leq_p$ Knapsack Proof idea**:
  - $K = W = T$ and $w_i = v_i = a_i$ for all i
- If $\exists$ subset S s.t. $\sum_{i \in S} a_i = T$, then pick those S to be in Knapsack
- If $\exists$ a subset X s.t. weight of items less than W and value less than K, then X is exactly the subset of items that sum to T

Acknowledgments

- Some of the material in these slides are taken from
  - Kleinberg Tardos Slides by Kevin Wayne (https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/04GreedyAlgorithmsI.pdf)
  - Jeff Erickson's Algorithms Book (http://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE.pdf)