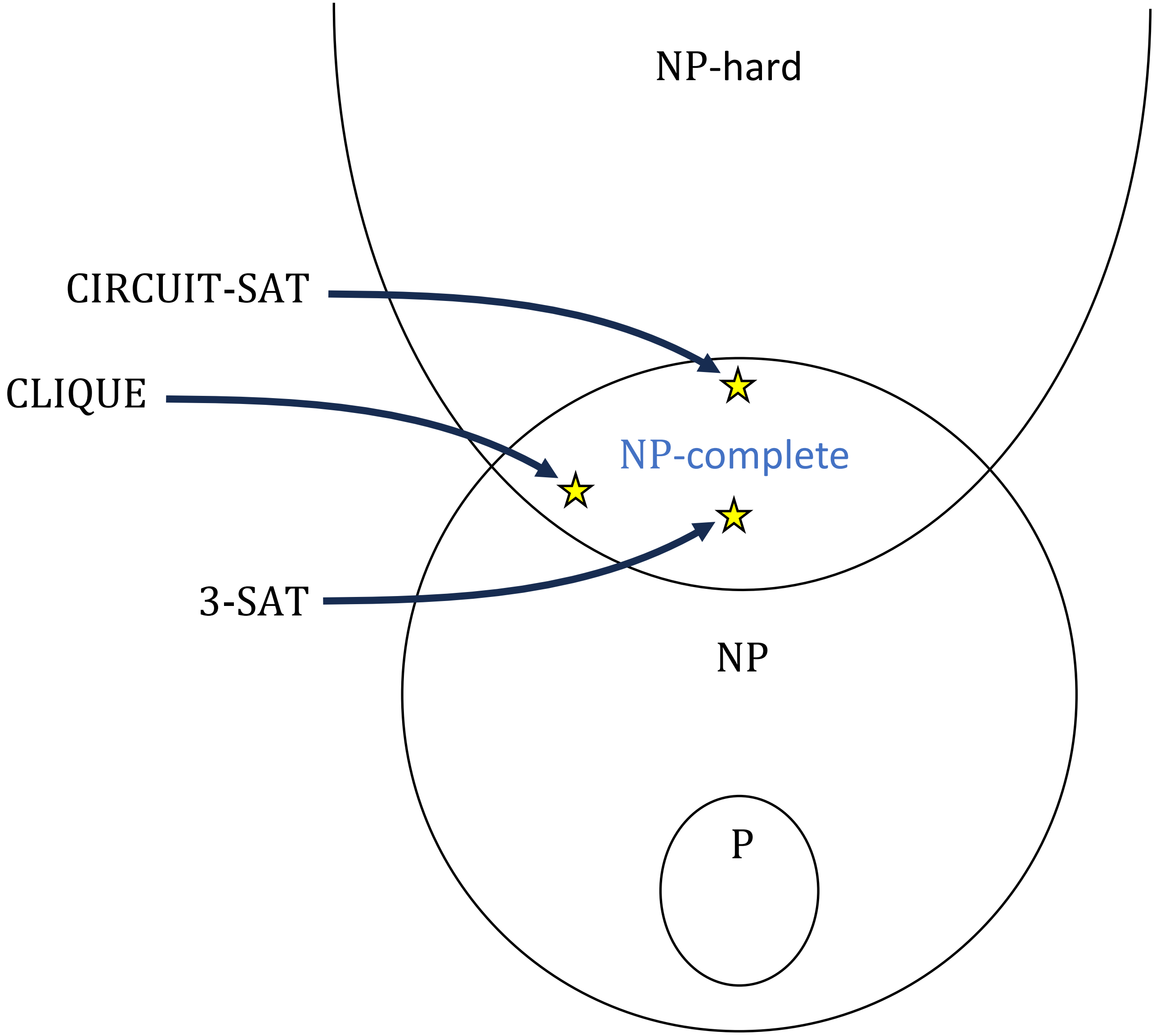
CMSC 28100

Introduction to Complexity Theory

Autumn 2025

Instructor: William Hoza


NP-hard
CIRCUIT-SAT
CLIQUE
NP-complete
3-SAT
NP
P

The subset sum problem

$$\text{SUBSET-SUM} = \left\{ \langle a_1, \ldots, a_k, T \rangle : \begin{array}{l} a_1, \ldots, a_k, T \in \mathbb{N} \ \text{ and there exists} \\ I \subseteq \{1, \ldots, k\} \text{ such that } \sum_{i \in I} a_i = T \end{array} \right\}$$

Theorem: SUBSET-SUM is NP-complete

- **Proof:** SUBSET-SUM $\in$ NP. (Why?)
- We will prove it is NP-hard by reduction from 3-SAT

Proof that 3-SAT $\leq_P$ SUBSET-SUM

If ϕ is a 3-CNF formula with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$, then $\Psi(\langle \phi \rangle) =$ the following:

Integers represented in base 8

	x_1	x_2	$\cdots$	x_n	c_1	c_2	$\cdots$	c_m
$a_{x_1} =$	1	0	$\cdots$	0	1	0	$\cdots$	0
$a_{\bar{x}_1} =$	1	0	$\cdots$	0	0	0	$\cdots$	0
$a_{x_2} =$		1	$\cdots$	0	0	$\boxed{1}$	$\cdots$	0
$a_{\bar{x}_2} =$		1	$\cdots$	0	1	0	$\cdots$	0
$\vdots$			$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_{x_n} =$				1	1	0	$\cdots$	1
$a_{\bar{x}_n} =$				1	0	1	$\cdots$	$\boxed{1}$
$a_{c_1} =$					1	0	$\cdots$	0
$a'_{c_1} =$					1	0	$\cdots$	0
$a_{c_2} =$						1	$\cdots$	0
$a'_{c_2} =$						1	$\cdots$	0
$\vdots$							$\ddots$	$\vdots$
$a_{c_m} =$								1
$a'_{c_m} =$								1
$T =$	1	1	$\cdots$	1	3	3	3	3

Does x_2 appear in c_2?

Does $\bar{x}_n$ appear in c_m?

- Suppose $\phi(x) = 1$
  - If $x_i = 1$, select a_{x_i}
  - If $x_i = 0$, select $a_{\bar{x}_i}$
  - If only two literals in c_j are satisfied, select a_{c_j}
  - If only one literal in c_j is satisfied, select a_{c_j} and a'_{c_j}
  - Selected numbers sum to T ✔

Proof that 3-SAT $\leq_P$ SUBSET-SUM

If ϕ is a 3-CNF formula with variables $x_1, \dots, x_n$ and clauses $c_1, \dots, c_m$, then $\Psi(\langle\phi\rangle) =$ the following:

Integers represented in base 8

$$
\begin{array}{rcccc|cccc}
 & x_1 & x_2 & \cdots & x_n & c_1 & c_2 & \cdots & c_m \\
\hline
a_{x_1} = & 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\
a_{\bar{x}_1} = & 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
a_{x_2} = & & 1 & \cdots & 0 & 0 & \boxed{1} & \cdots & 0 \\
a_{\bar{x}_2} = & & 1 & \cdots & 0 & 1 & 0 & \cdots & 0 \\
\vdots & & & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
a_{x_n} = & & & & 1 & 1 & 0 & \cdots & 1 \\
a_{\bar{x}_n} = & & & & 1 & 0 & 1 & \cdots & \boxed{1} \\
\hline
a_{c_1} = & & & & & 1 & 0 & \cdots & 0 \\
a'_{c_1} = & & & & & 1 & 0 & \cdots & 0 \\
a_{c_2} = & & & & & & 1 & \cdots & 0 \\
a'_{c_2} = & & & & & & 1 & \cdots & 0 \\
\vdots & & & & & & & \ddots & \vdots \\
a_{c_m} = & & & & & & & & 1 \\
a'_{c_m} = & & & & & & & & 1 \\
\hline
T = & 1 & 1 & \cdots & 1 & 3 & 3 & 3 & 3
\end{array}
$$

Does x_2 appear in c_2?

Does $\bar{x}_n$ appear in c_m?

- Suppose a subset of the numbers sum to T
  - There are no "carries," because each column has at most five ones
  - If a_{x_i} is selected, set $x_i = 1$
  - If $a_{\bar{x}_i}$ is selected, set $x_i = 0$
  - Each clause must have at least one satisfied literal ✔

Proof that 3-SAT $\leq_P$ SUBSET-SUM

If ϕ is a 3-CNF formula with variables $x_1, \dots, x_n$ and clauses $c_1, \dots, c_m$, then $\Psi(\langle\phi\rangle) =$ the following:

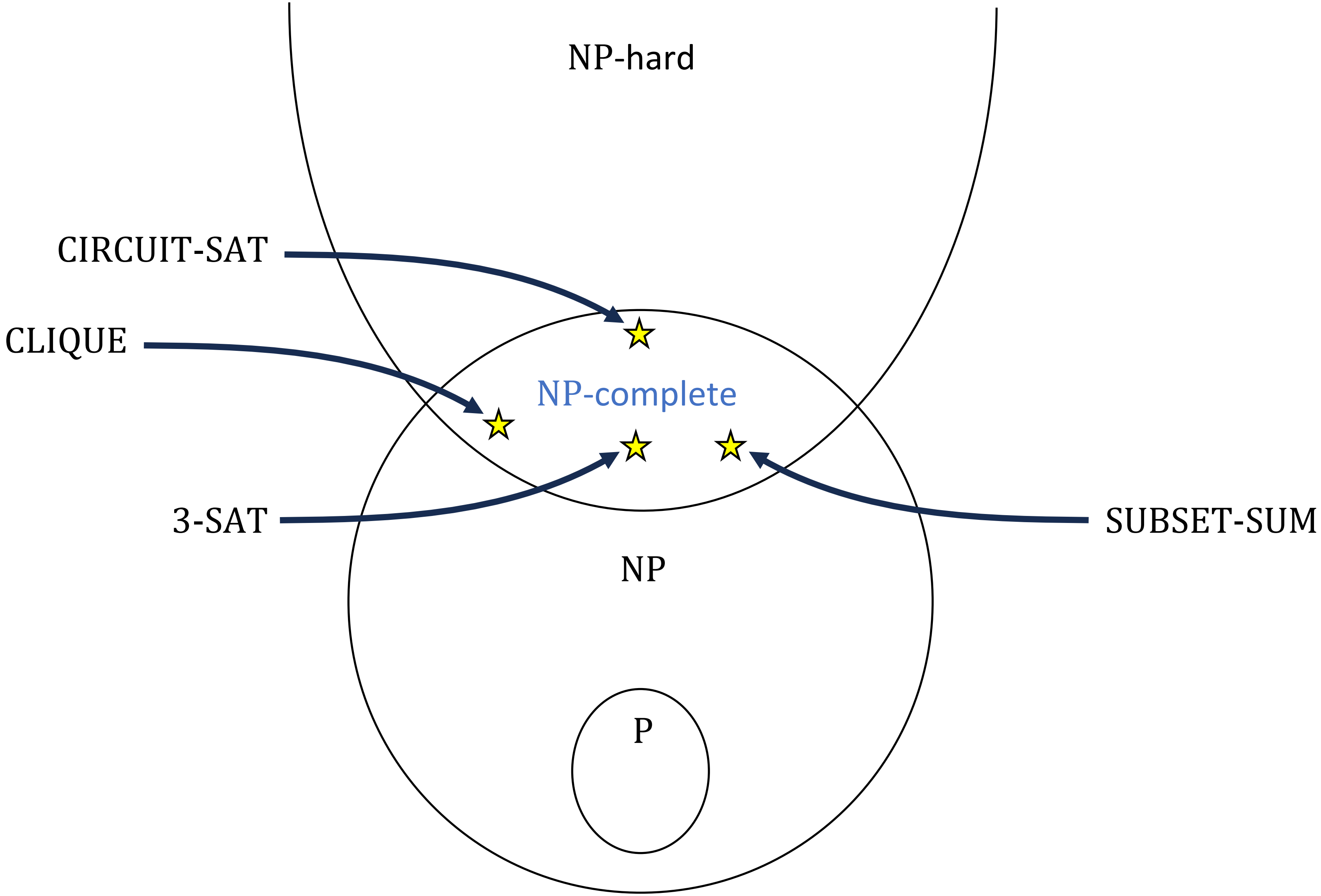
Integers represented in base 8

	x_1	x_2	$\cdots$	x_n	c_1	c_2	$\cdots$	c_m
$a_{x_1} =$	1	0	$\cdots$	0	1	0	$\cdots$	0
$a_{\bar{x}_1} =$	1	0	$\cdots$	0	0	0	$\cdots$	0
$a_{x_2} =$		1	$\cdots$	0	0	1	$\cdots$	0
$a_{\bar{x}_2} =$		1	$\cdots$	0	1	0	$\cdots$	0
$\vdots$			$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_{x_n} =$				1	1	0	$\cdots$	1
$a_{\bar{x}_n} =$				1	0	1	$\cdots$	1
$a_{c_1} =$					1	0	$\cdots$	0
$a'_{c_1} =$					1	0	$\cdots$	0
$a_{c_2} =$						1	$\cdots$	0
$a'_{c_2} =$						1	$\cdots$	0
$\vdots$							$\ddots$	$\vdots$
$a_{c_m} =$								1
$a'_{c_m} =$								1
$T =$	1	1	$\cdots$	1	3	3	3	3

Does x_2 appear in c_2?

Does $\bar{x}_n$ appear in c_m?

- Reduction can be performed in polynomial time



NP-hard
CIRCUIT-SAT
CLIQUE
NP-complete
3-SAT
SUBSET-SUM
NP
P

Proving that Y_{NEW} is NP-complete ("cheat sheet")

1. Prove that $Y_{\mathrm{NEW}} \in \mathrm{NP}$
   - What is the certificate? How can you verify a purported certificate in polynomial time?
2. Prove that Y_{NEW} is NP-hard
   - Which NP-complete language Y_{OLD} are you reducing from?
   - What is the reduction? $\Psi(w) = w'$. How is w' defined? Polynomial time?
   - YES maps to YES: Assume there is a certificate x showing $w \in Y_{\mathrm{OLD}}$. In terms of x, describe a certificate y showing that $w' \in Y_{\mathrm{NEW}}$.
   - NO maps to NO: (Contrapositive) Assume there is a certificate y showing $w' \in Y_{\mathrm{NEW}}$. In terms of y, describe a certificate x showing that $w \in Y_{\mathrm{OLD}}$.

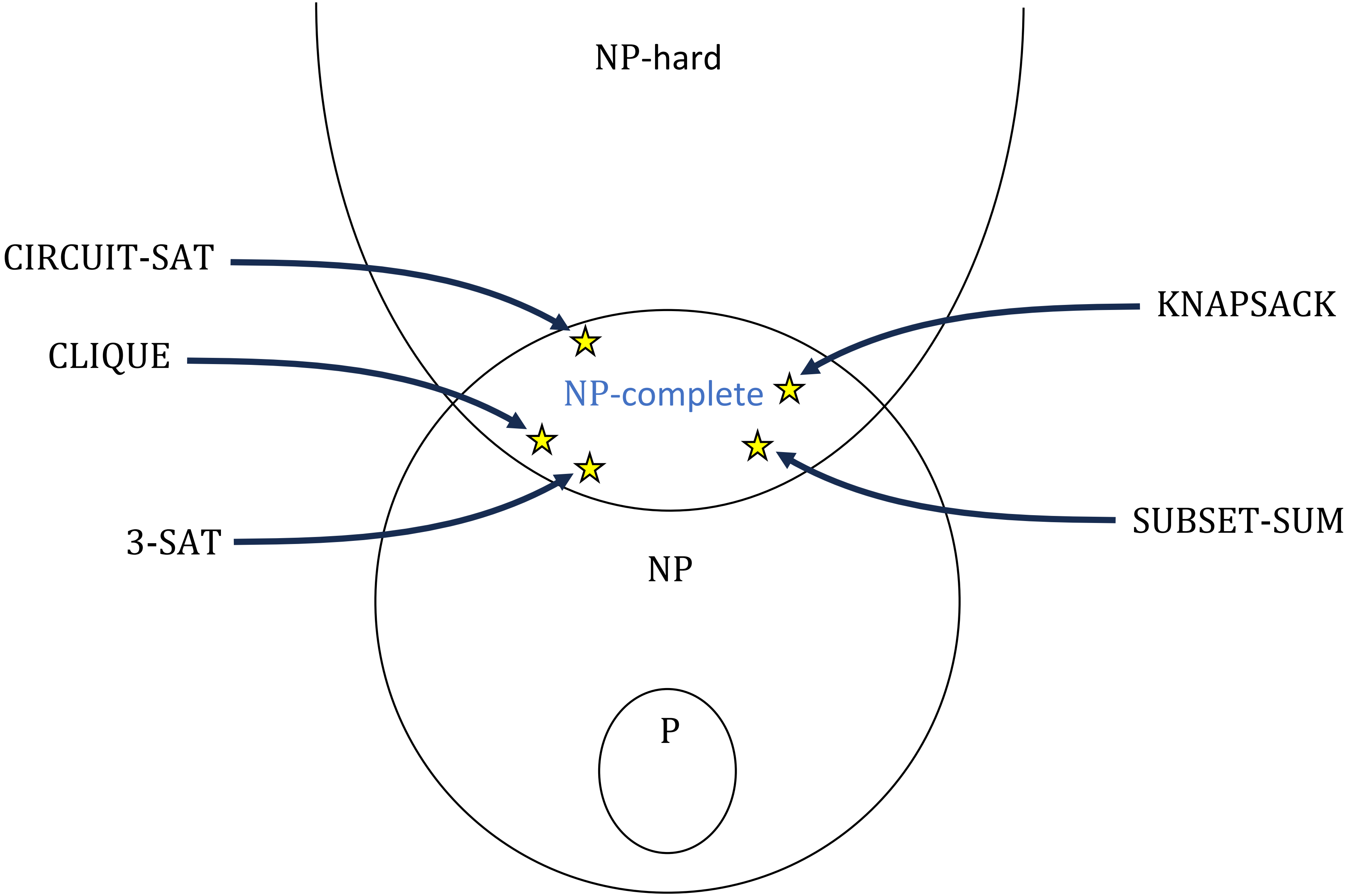
The Knapsack problem

- KNAPSACK $= \{\langle w_1, \dots, w_k, v_1, \dots, v_k, W, V\rangle : \text{there exists } S \subseteq \{1, 2, \dots, k\}$
  $\text{such that } \Sigma_{i \in S}\, w_i \leq W \text{ and } \Sigma_{i \in S}\, v_i \geq V\}$

Theorem: KNAPSACK is NP-complete

- **Proof:** It's in NP ✔
- Reduction from SUBSET-SUM:

$$\Psi(\langle a_1, \dots, a_k, T\rangle) = \langle a_1, \dots, a_k, a_1, \dots, a_k, T, T\rangle$$


NP-hard
CIRCUIT-SAT
CLIQUE
3-SAT
KNAPSACK
SUBSET-SUM
NP-complete
NP
P

NP-completeness is everywhere

- There are thousands of known NP-complete problems!
- These problems come from many different areas of study
  - Logic, graph theory, number theory, scheduling, optimization, economics, physics, chemistry, biology, …
- P vs. NP is one of the most important open questions in theoretical computer science and mathematics

NP-complete languages stand or fall together

- If you design a poly-time algorithm for one NP-complete language, then $\mathrm{P} = \mathrm{NP}$, so all NP-complete languages can be decided in poly time!

- If you prove that one NP-complete language cannot be decided in poly time, then $\mathrm{P} \neq \mathrm{NP}$, so no NP-complete language can be decided in poly time!

Intractability

- **This course so far:** How to identify intractability
- **Up next:** How to cope with intractability

Coping with intractability

- Suppose you really want to decide Y
- You find proof/evidence that $Y \notin \mathrm{P}$ 😟
  - Undecidability, EXP-hardness, NP-hardness…
- That doesn't necessarily mean you're out of luck…
- There are several approaches for coping with the fact that $Y \notin \mathrm{P}$

Coping with intractability

Nontrivial exponential-time algorithms

- Even if $Y \notin \mathrm{P}$, it still might have a nontrivial algorithm. Example:

Theorem: There is an algorithm that computes the size of the largest clique in a given n-vertex graph in time $O(1.189^n)$.

  - (Proof omitted. Not on exercises / exams)

- If your inputs happen to be relatively small, then maybe an exponential time complexity is tolerable

Pseudo-polynomial time algorithms

- If you have numeric inputs, you could try a pseudo-poly-time algorithm
- UNARY-VAL-KNAPSACK $= \{\langle w_1, \ldots, w_k, 1^{v_1}, \ldots, 1^{v_k}, W, 1^V \rangle$: there exists $S \subseteq \{1, 2, \ldots, k\}$ such that $\Sigma_{i \in S}\, w_i \leq W$ and $\Sigma_{i \in S}\, v_i \geq V\}$

Theorem: UNARY-VAL-KNAPSACK $\in$ P

Approximation algorithms

- Next approach for coping with intractability: approximation algorithms
- Example: Knapsack

Approximation algorithm for Knapsack

- For every $w_1, \ldots, w_k, v_1, \ldots, v_k, W$, define

$$\text{OPT} = \max\left\{\sum_{i \in S} v_i \,:\, S \subseteq \{1, \ldots, k\} \text{ and } \sum_{i \in S} w_i \leq W\right\}$$

Theorem: For every $\epsilon > 0$, there exists a poly-time algorithm such that given $w_1, \ldots, w_k, v_1, \ldots, v_k, W$, the algorithm outputs $S \subseteq \{1, \ldots, k\}$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i \geq (1 - \epsilon) \cdot \text{OPT}$