

CMPS 2200 -- Fall 2015

P and NP

Carola Wenk

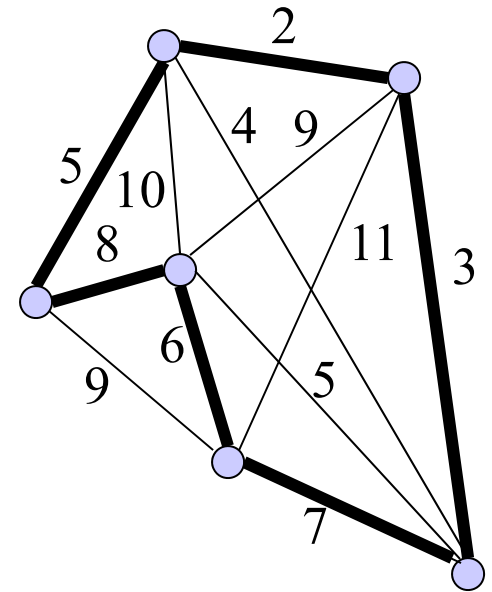
Slides courtesy of Piotr Indyk with additions by Carola Wenk

We have seen so far

- Algorithms for various problems
 - Running times $O(nm^2)$, $O(n^2)$, $O(n \log n)$, $O(n)$, etc.
 - I.e., polynomial in the input size
- Can we solve all (or most of) interesting problems in polynomial time ?
- Not really...

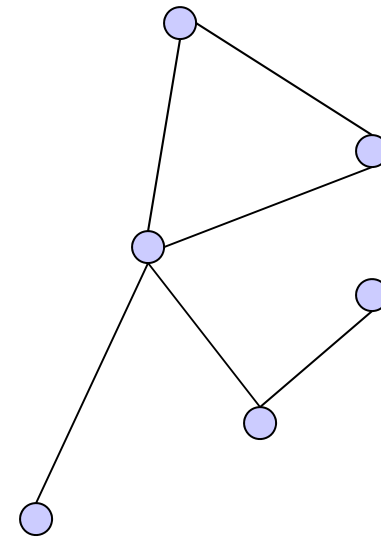
Example difficult problem

- Traveling Salesperson Problem (TSP; optimization variant)
 - **Input:** Undirected graph with lengths on edges
 - **Output:** Shortest tour that visits each vertex exactly once
- Best known algorithm: $O(n 2^n)$ time.



Another difficult problem

- Clique (optimization variant):
 - **Input:** Undirected graph $G=(V,E)$
 - **Output:** Largest subset C of V such that every pair of vertices in C has an edge between them (C is called a *clique*)
- Best known algorithm:
 $O(n 2^n)$ time



What can we do ?

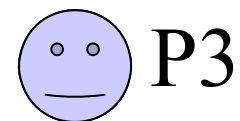
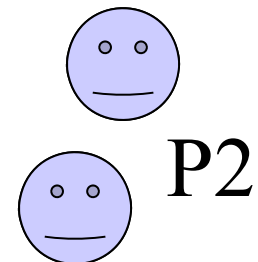
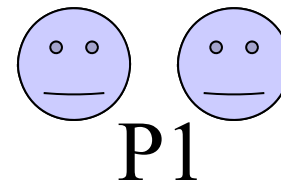
- Spend more time designing algorithms for those problems
 - People tried for a few decades, no luck
- Prove there is **no** polynomial time algorithm for those problems
 - Would be great
 - Seems *really* difficult
 - Best lower bounds for “natural” problems:
 - $\Omega(n^2)$ for restricted computational models
 - $4.5n$ for unrestricted computational models

What else can we do ?

- Show that those hard problems are essentially equivalent. I.e., if we can solve one of them in polynomial time, then all others can be solved in polynomial time as well.
- Works for at least 10 000 hard problems

The benefits of equivalence

- Combines research efforts
- If one problem has a polynomial time solution, then all of them do
- More realistically: Once an exponential **lower bound** is shown for one problem, it holds for all of them



Summing up

- If we show that a problem Π is equivalent to ten thousand other well studied problems without efficient algorithms, then we get a very strong evidence that Π is hard.
- We need to:
 1. Identify the class of problems of interest
→ Decision problems, NP
 2. Define the notion of equivalence
→ Polynomial-time reductions
 3. Prove the equivalence(s)

Decision Problem

- Decision problems: answer YES or NO.
- Example: **Search Problem** Π_{Search}
Given an unsorted set S of n numbers and a number key , is key contained in S ?
- Input is $x=(S, key)$
- Possible algorithms that solve $\Pi_{\text{Search}}(x)$:
 - $A_1(x)$: Linear search algorithm. $O(n)$ time
 - $A_2(x)$: Sort the array and then perform binary search. $O(n \log n)$ time
 - $A_3(x)$: Compute all possible subsets of S (2^n many) and check each subset if it contains key. $O(n2^n)$ time.

Decision problem vs. optimization problem

3 variants of Clique:

- 1. Input:** Undirected graph $G=(V,E)$, and an integer $k \geq 0$.
Output: Does G contain a clique C such that $|C| \geq k$?
- 2. Input:** Undirected graph $G=(V,E)$
Output: Largest integer k such that G contains a clique C with $|C|=k$.
- 3. Input:** Undirected graph $G=(V,E)$
Output: Largest clique C of V .

3. is harder than **2.** is harder than **1.** So, if we reason about the decision problem (**1.**), and can show that it is hard, then the others are hard as well. Also, every algorithm for **3.** can solve **2.** and **1.** as well.

Decision problem vs. optimization problem (cont.)

Theorem:

- a) If **1.** can be solved in polynomial time, then **2.** can be solved in polynomial time.
- b) If **2.** can be solved in polynomial time, then **3.** can be solved in polynomial time.

Proof:

- a) Run **1.** for values $k = 1 \dots n$. Instead of linear search one could also do binary search.
- b) Run **2.** to find the size k_{opt} of a largest clique in G . Now check one edge after the other. Remove one edge from G , compute the new size of the largest clique in this new graph. If it is still k_{opt} then this edge is not necessary for a clique. If it is less than k_{opt} then it is part of the clique.

Class of problems: NP

- Decision problems: answer YES or NO. E.g., "is there a tour of length $\leq K$ " ?
- Solvable in *non-deterministic polynomial* time:
 - Intuitively: the solution can be **verified** in polynomial time
 - E.g., if someone gives us a tour T , we can verify in *polynomial* time if T is a tour of length $\leq K$.
- Therefore, the decision variant of TSP is in NP.

Formal definitions of P and NP

- A decision problem Π is solvable in **polynomial time** (or $\Pi \in P$), if there is a polynomial time algorithm $A(\cdot)$ such that for any input x :

$$\Pi(x)=\text{YES} \text{ iff } A(x)=\text{YES}$$

- A decision problem Π is solvable in **non-deterministic polynomial time** (or $\Pi \in NP$), if there is a polynomial time algorithm $A(\cdot, \cdot)$ such that for any input x :

$$\Pi(x)=\text{YES} \text{ iff there exists a certificate } y \text{ of size } \text{poly}(|x|) \text{ such that } A(x,y)=\text{YES}$$

Examples of problems in NP

- Is “Does there exist a clique in G of size $\geq K$ ” in NP ?

Yes: $A(x,y)$ interprets x as (G,K) , y as a set C , and checks if all vertices in C are adjacent and if $|C| \geq K$

- Is **Sorting** in NP ?

No, not a decision problem.

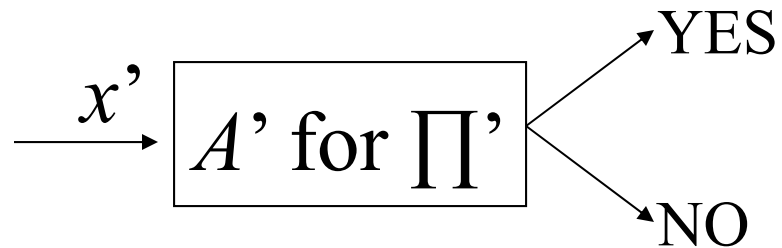
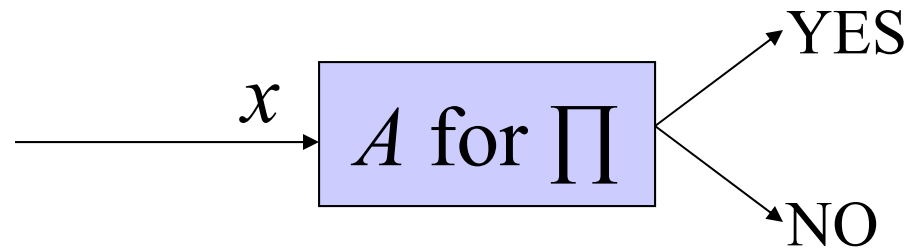
- Is “**Sortedness**” in NP ?

Yes: ignore y , and check if the input x is sorted.

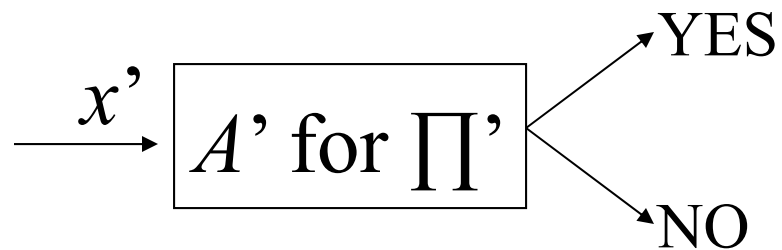
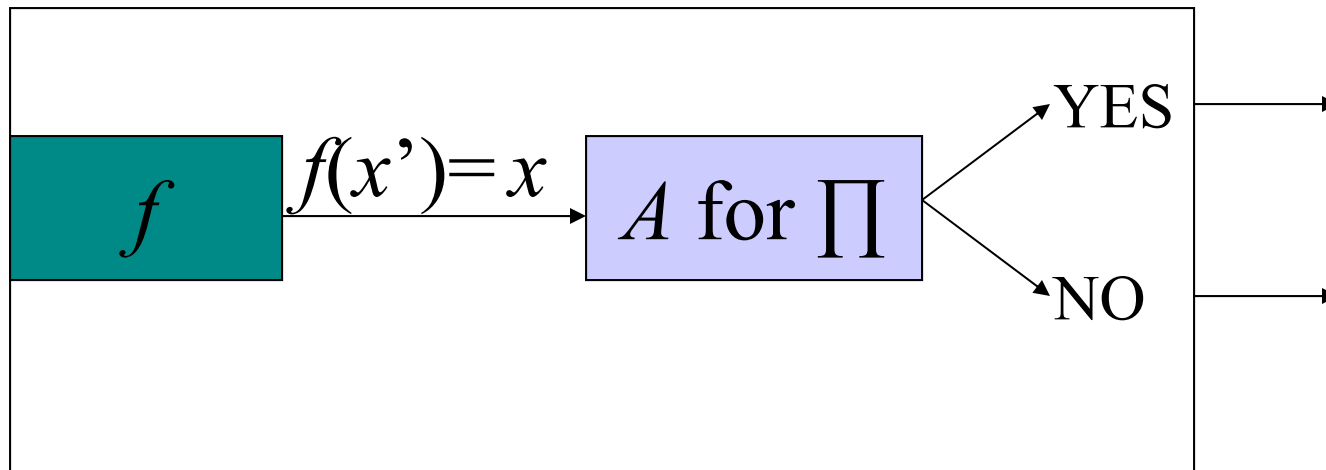
Summing up

- If we show that a problem Π is equivalent to ten thousand other well studied problems without efficient algorithms, then we get a very strong evidence that Π is hard.
- We need to:
 1. Identify the class of problems of interest
→ Decision problems, NP
 2. Define the notion of equivalence
→ Polynomial-time reductions
 3. Prove the equivalence(s)

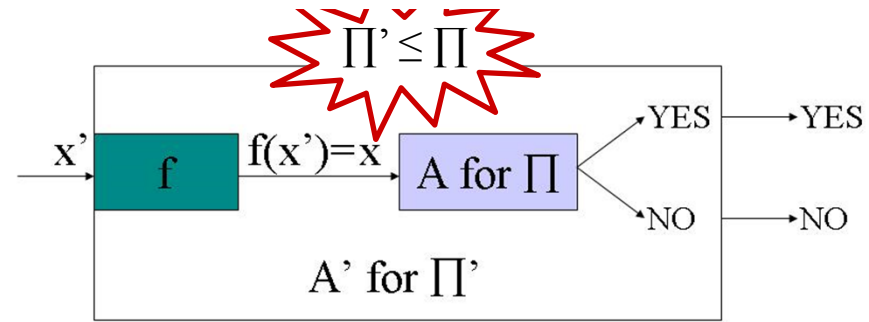
$\Pi' \leq \Pi$: Reduce Π' to Π



$\Pi' \leq \Pi$: Reduce Π' to Π



Reductions



- Π' is *polynomial time reducible* to Π ($\Pi' \leq \Pi$) iff
 1. there is a polynomial time function f that maps inputs x' for Π' into inputs x for Π ,
 2. such that for any x' :

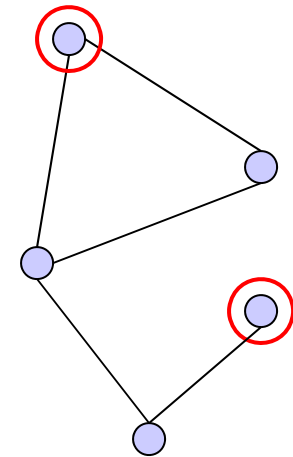
$$\Pi'(x') = \Pi(f(x'))$$

(or in other words $\Pi'(x') = \text{YES}$ iff $\Pi(f(x')) = \text{YES}$)

- Fact 1: if $\Pi \in P$ and $\Pi' \leq \Pi$ then $\Pi' \in P$
- Fact 2: if $\Pi \in NP$ and $\Pi' \leq \Pi$ then $\Pi' \in NP$
- Fact 3 (transitivity):
if $\Pi'' \leq \Pi'$ and $\Pi' \leq \Pi$ then $\Pi'' \leq \Pi$

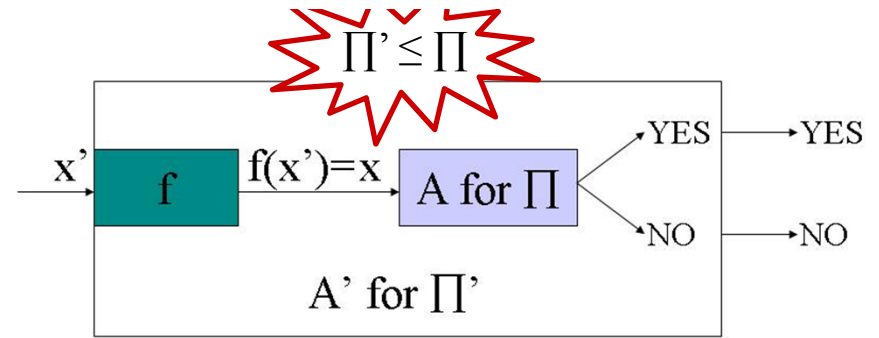
Independent set (IS)

- **Input:** Undirected graph $G=(V,E)$, K
- **Output:** Is there a subset S of V , $|S| \geq K$ such that **no** pair of vertices in S has an edge between them? (S is called an *independent set*)



Clique \leq **IS**

$\underbrace{\text{Clique}}_{\Pi'}$ \leq $\underbrace{\text{IS}}_{\Pi}$

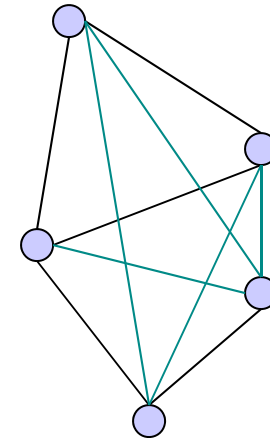


- Given an input $G=(V,E), K$ to Clique, need to construct an input $G'=(V',E'), K'$ to IS,

$$f(x')=x$$

such that G has clique of size $\geq K$ iff G' has IS of size $\geq K'$.

- Construction: $K'=K, V'=V, E'=\overline{E}$
- Reason: C is a clique in G iff it is an IS in G' 's complement.

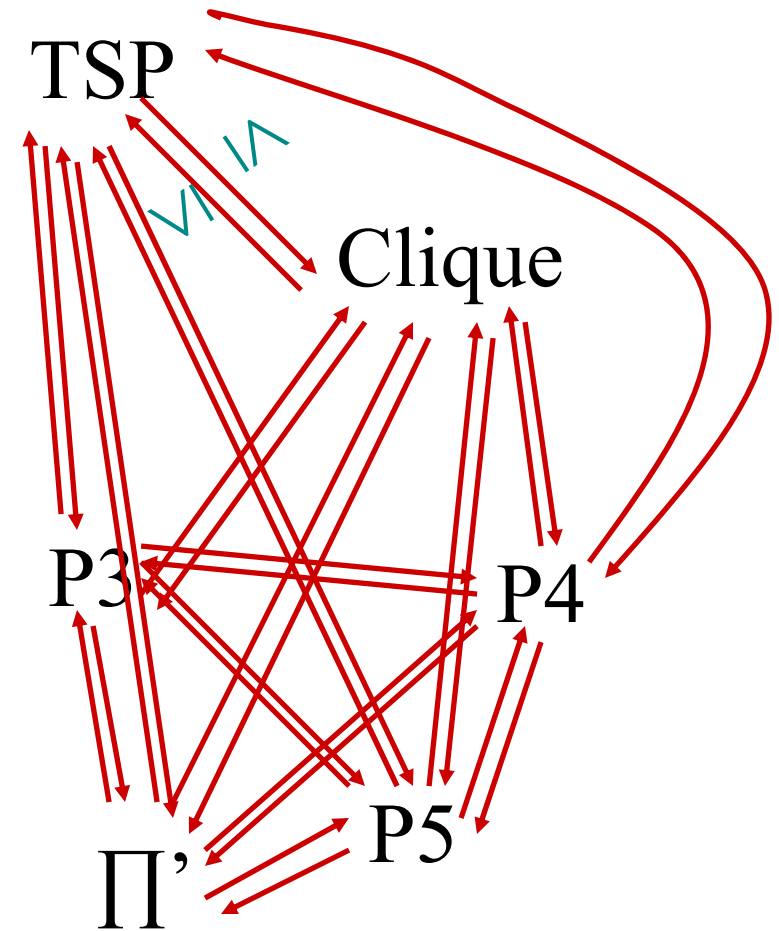


Recap

- We defined a large class of interesting problems, namely NP
- We have a way of saying that one problem is not harder than another ($\Pi' \leq \Pi$)
- Our goal: show equivalence between hard problems

Showing equivalence between difficult problems

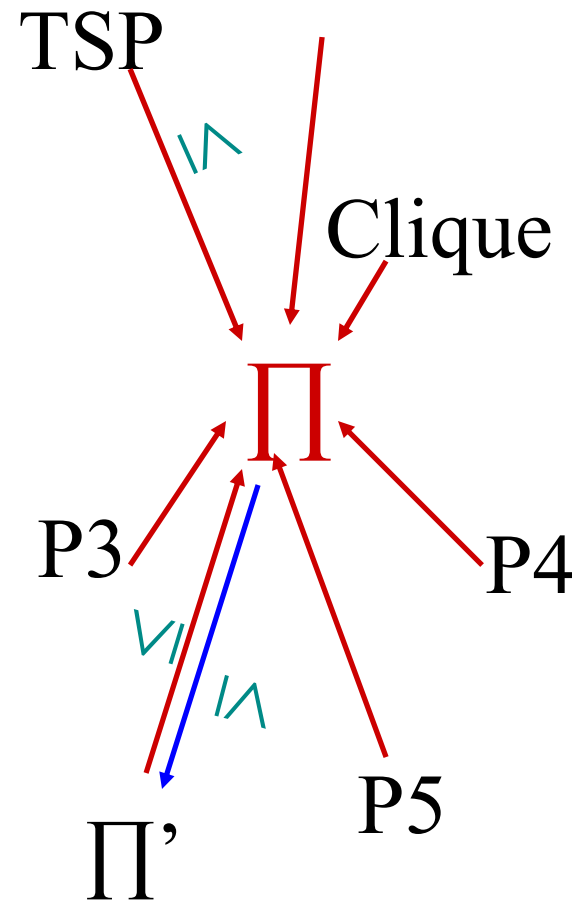
- Options:
 - Show reductions between all pairs of problems
 - Reduce the number of reductions using transitivity of “ \leq ”



Showing equivalence between difficult problems

- Options:
 - Show reductions between all pairs of problems
 - Reduce the number of reductions using transitivity of “ \leq ”
 - Show that *all* problems in NP are reducible to a *fixed* Π .

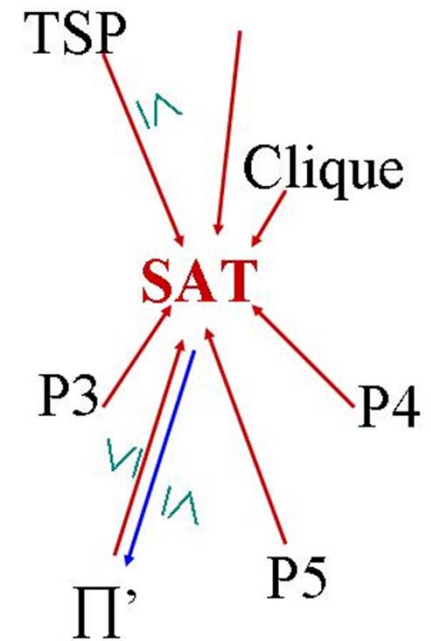
To show that some problem $\Pi' \in \text{NP}$ is equivalent to all difficult problems, we only show $\Pi \leq \Pi'$.



The first problem \square

- Satisfiability problem (SAT):
 - Given: a formula φ with m clauses over n variables, e.g., $x_1 \vee x_2 \vee x_5, x_3 \vee \neg x_5$
 - Check if there exists TRUE/FALSE assignments to the variables that makes the formula satisfiable

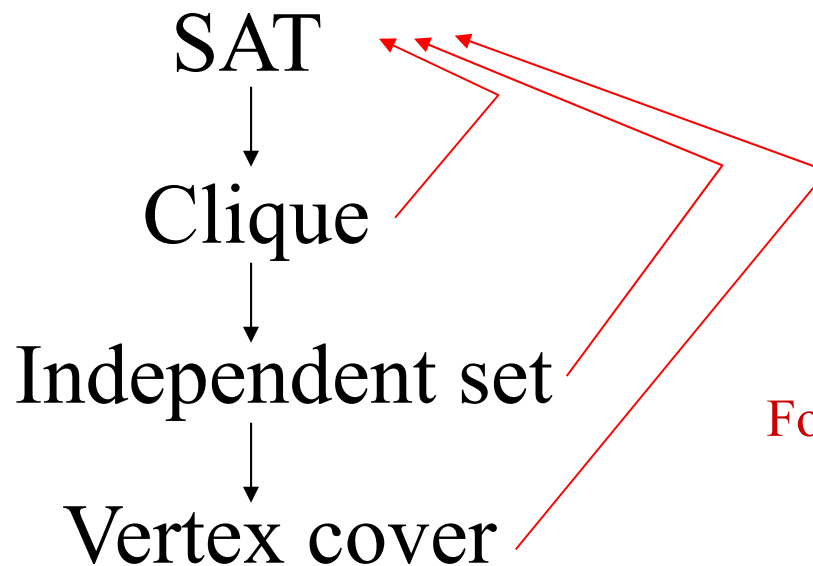
SAT is NP-complete



- **Fact:** $SAT \in NP$
- **Theorem [Cook'71]:** For any $\Pi' \in NP$ we have $\Pi' \leq SAT$.
- **Definition:** A problem Π such that for any $\Pi' \in NP$ we have $\Pi' \leq \Pi$, is called *NP-hard*
- **Definition:** An NP-hard problem that belongs to NP is called *NP-complete*
- **Corollary:** SAT is NP-complete.

Plan of attack:

- Show that the problems below are in NP, and show a sequence of reductions:



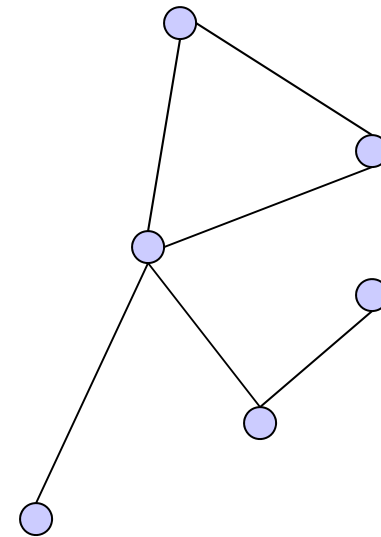
(thanks, Steve ☺)

Follow from Cook's Theorem

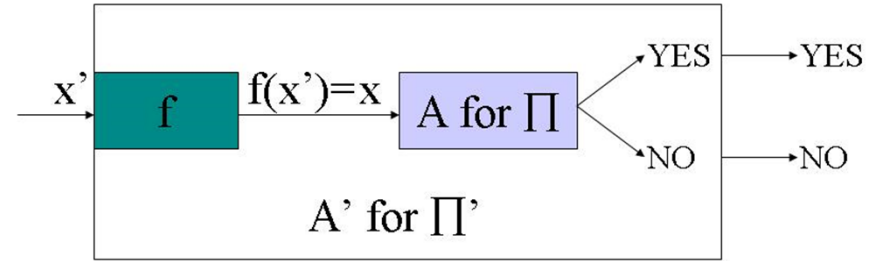
- Conclusion: all of the above problems are NP-complete

Clique again

- Clique (decision variant):
 - **Input:** Undirected graph $G=(V,E)$, and an integer $K \geq 0$
 - **Output:** Is there a clique C , i.e., a subset C of V such that every pair of vertices in C has an edge between them, such that $|C| \geq K$?



SAT \leq Clique



- Given a SAT formula $\varphi = C_1, \dots, C_m$ over x_1, \dots, x_n , we need to produce $G = (V, E)$ and K ,
 $f(x') = x$

such that φ satisfiable iff G has a clique of size $\geq K$.

- Notation: a **literal** is either x_i or $\neg x_i$

SAT \leq Clique reduction

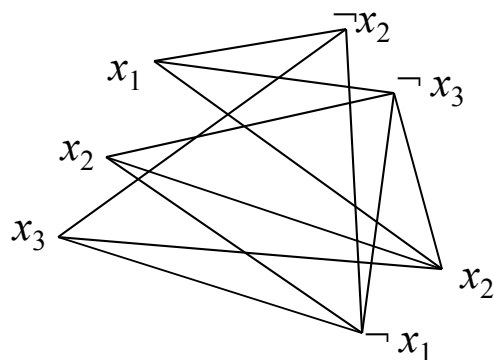
- For each literal t occurring in φ , create a vertex v_t
- Create an edge $v_t - v_{t'}$, iff:
 - t and t' are not in the same clause, and
 - t is not the negation of t'

SAT \leq Clique example

Edge $v_t - v_{t'}$ \Leftrightarrow

- t and t' are not in the same clause, and
- t is not the negation of t'

- Formula: $x_1 \vee x_2 \vee x_3, \neg x_2 \vee \neg x_3, \neg x_1 \vee x_2$
- Graph:



- Claim: φ satisfiable iff G has a clique of size $\geq m$

Proof

Edge $v_t - v_{t'}$ \Leftrightarrow

- t and t' are not in the same clause, and
- t is not the negation of t'

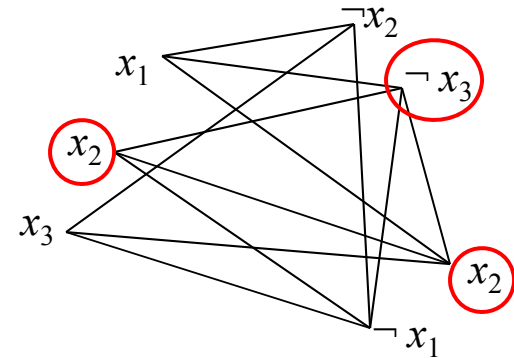
- “ \rightarrow ” part of Claim:

- Take any assignment that satisfies φ .

E.g., $x_1 = \text{F}$, $x_2 = \text{T}$, $x_3 = \text{F}$

- Let the set C contain one satisfied literal per clause

- C is a clique

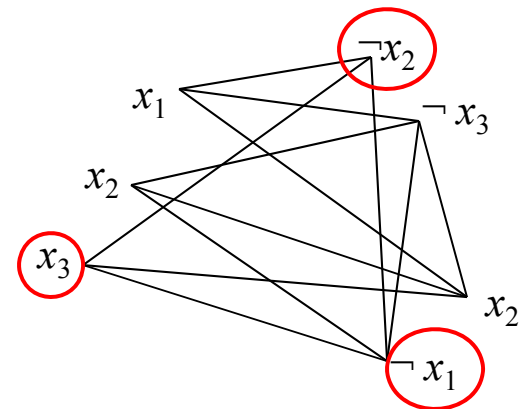


Proof

Edge $v_t - v_{t'} \Leftrightarrow$

- t and t' are not in the same clause, and
- t is not the negation of t'

- “ \leftarrow ” part of Claim:
 - Take any clique C of size $\geq m$ (i.e., $= m$)
 - Create a set of equations that satisfies selected literals.
- E.g., $x_3 = \text{T}$, $x_2 = \text{F}$, $x_1 = \text{F}$
- The set of equations is consistent and the solution satisfies φ

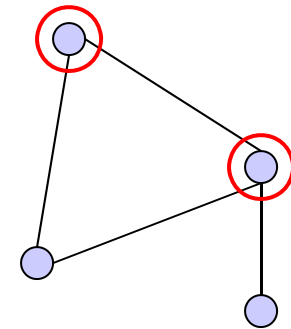


Altogether

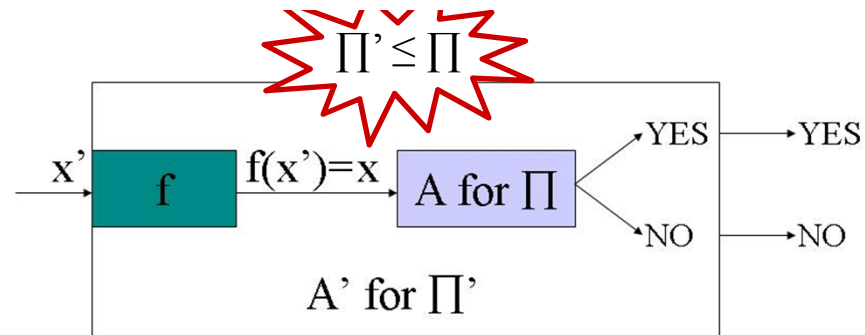
- We constructed a reduction that maps:
 - YES inputs to SAT to YES inputs to Clique
 - NO inputs to SAT to NO inputs to Clique
- The reduction works in polynomial time
- Therefore, $SAT \leq Clique \rightarrow Clique$ NP-hard
- Clique is in NP \rightarrow Clique is NP-complete

Vertex cover (VC)

- Input: undirected graph $G=(V,E)$, and $K \geq 0$
- Output: is there a subset C of V , $|C| \leq K$, such that each edge in E is incident to at least one vertex in C .



$$\underbrace{\text{IS}}_{\Pi'} \leq \underbrace{\text{VC}}_{\Pi}$$



- Given an input $G=(V,E)$, K to IS, need to construct an input $G'=(V',E')$, K' to VC, such that

$$f(x')=x$$

G has an IS of size $\geq K$ iff G' has VC of size $\leq K'$.

- Construction: $V'=V$, $E'=E$, $K'=|V|-K$
- Reason: S is an IS in G iff $V-S$ is a VC in G .

