Introduction to Algorithms

Lecture 15

Last Time

- Introduction to Computational Geometry
- Computational Model
- Closest Pair problem
- Close Pair problem
- Segment intersection problem
- Orthogonal segments

Today's Topics

- Some difficult problems
- P and NP
- Polynomial-time reductions
- Cook's Theorem (SAT Problem)
- NP-complete problem
  - Clique
  - Independent set
  - Vertex cover

P vs NP (interconnectedness of all things)

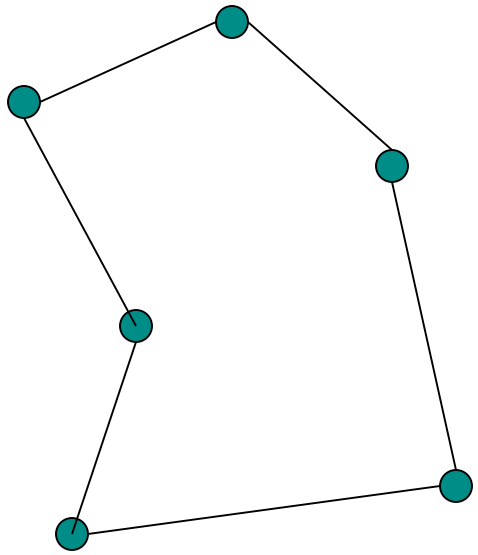
- A whole course by itself
- We'll do just one lecture
- More in "Theory of Computation", "Computational Complexity", etc.

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…

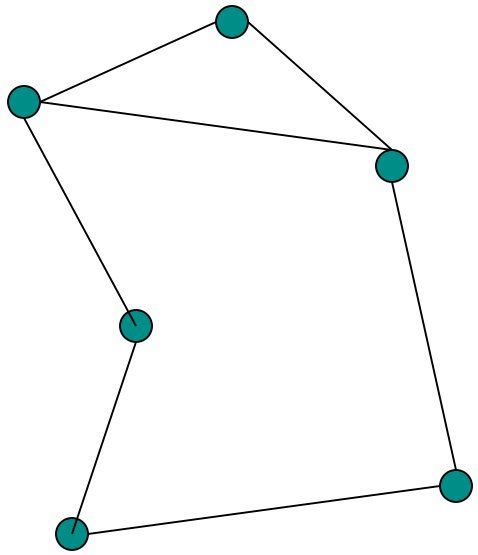
Example difficult problem

- Traveling Salesperson Problem (TSP)
  - 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:
  - 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
- Best known algorithm: $O(n\,2^n)$ time



What can we do ?

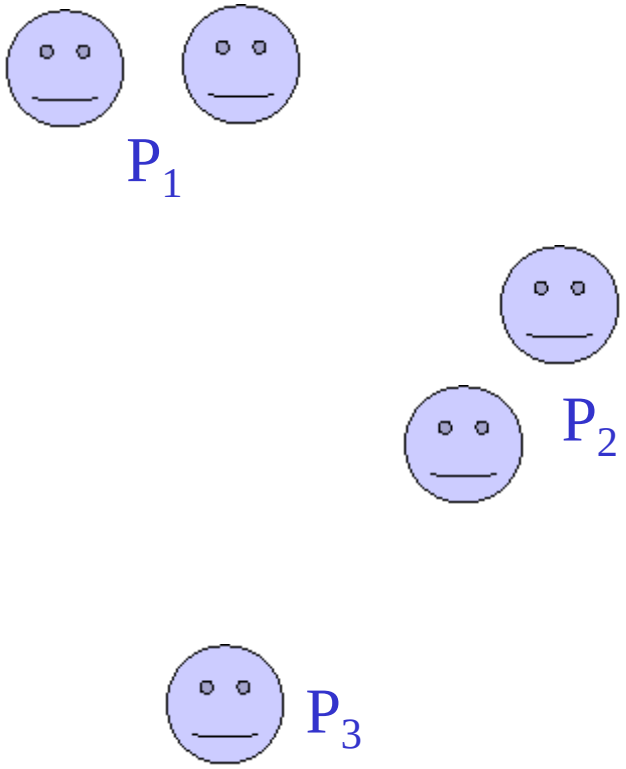
- Spend more time designing algorithms for those problems
  - People tried for a few decades, no luck
- Prove there is no polynomial time algorithm for those problems
  - Would be great
  - Seems *really* difficult
  - Best lower bounds for “natural” problems:
    - $\Omega(n^2)$ for restricted computational 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 poly time, then all others can be solved in poly time as well.
- Works for at least 10 000 hard problems

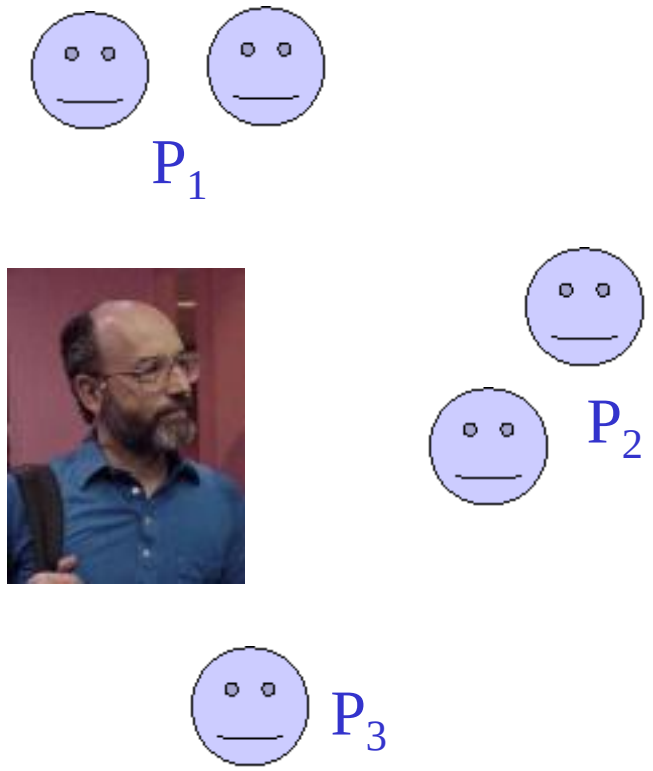
The benefits of equivalence

- Combines research efforts
- If one problem has polytime solution, then all of them do



A more realistic scenario

- Once an exponential lower bound is shown for one problem, it holds for all of them
- But someone *is* happy…



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:
  - Identify the class of problems of interest
  - Define the notion of equivalence
  - Prove the equivalence(s)

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 as a tour T, we can verify if T is a tour of length $\leq K$.
- Therefore, TSP is in NP.

Formal definitions of P and NP

- A problem Π is solvable in poly time (or $\Pi \in P$), if there is a poly time algorithm $V(.)$ such that for any input x:

  $\Pi(x) = YES$ iff $V(x) = YES$

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

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

Examples of problems in NP

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

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

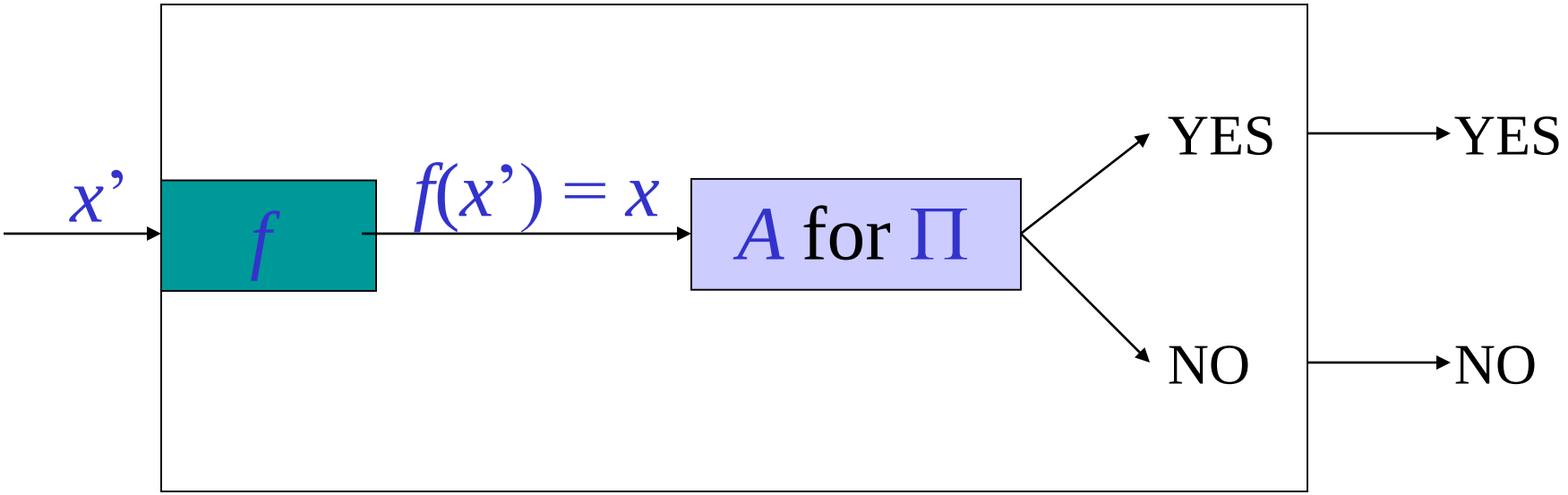
- Is Sorting in NP ?

  No, not a decision problem.

- Is "Sortedness" in NP ?

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

Reductions: Π' to Π



A' for Π'

Reductions

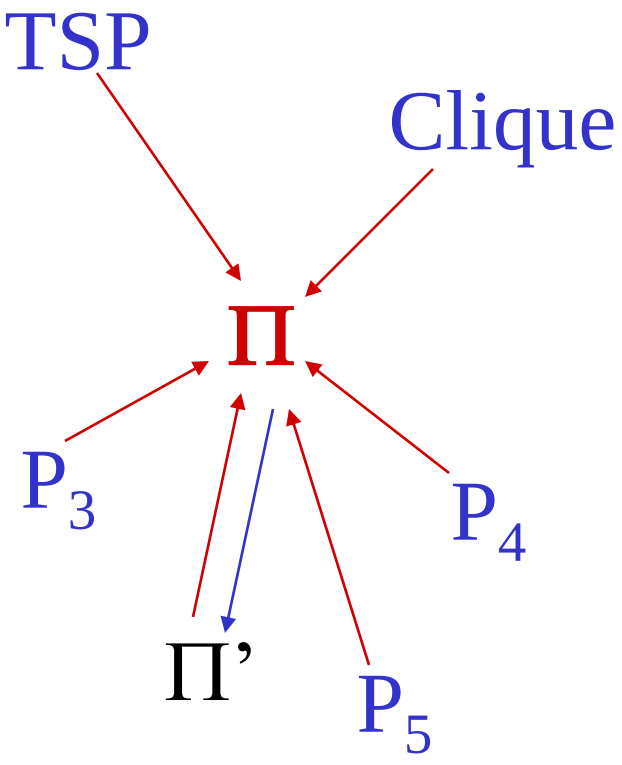
- Π' is poly time reducible to Π ($\Pi' \leq \Pi$) iff there is a poly time function f that maps inputs x' of Π' into inputs x of Π, such that for any x'

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

- 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: if $\Pi' \leq \Pi$ and $\Pi'' \leq \Pi'$ then $\Pi'' \leq \Pi$

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 a reducible to a *fixed* Π. To show that some problem $\Pi' \in NP$ is equivalent to all difficult problems, we only show $\Pi \leq \Pi'$.



The first problem Π

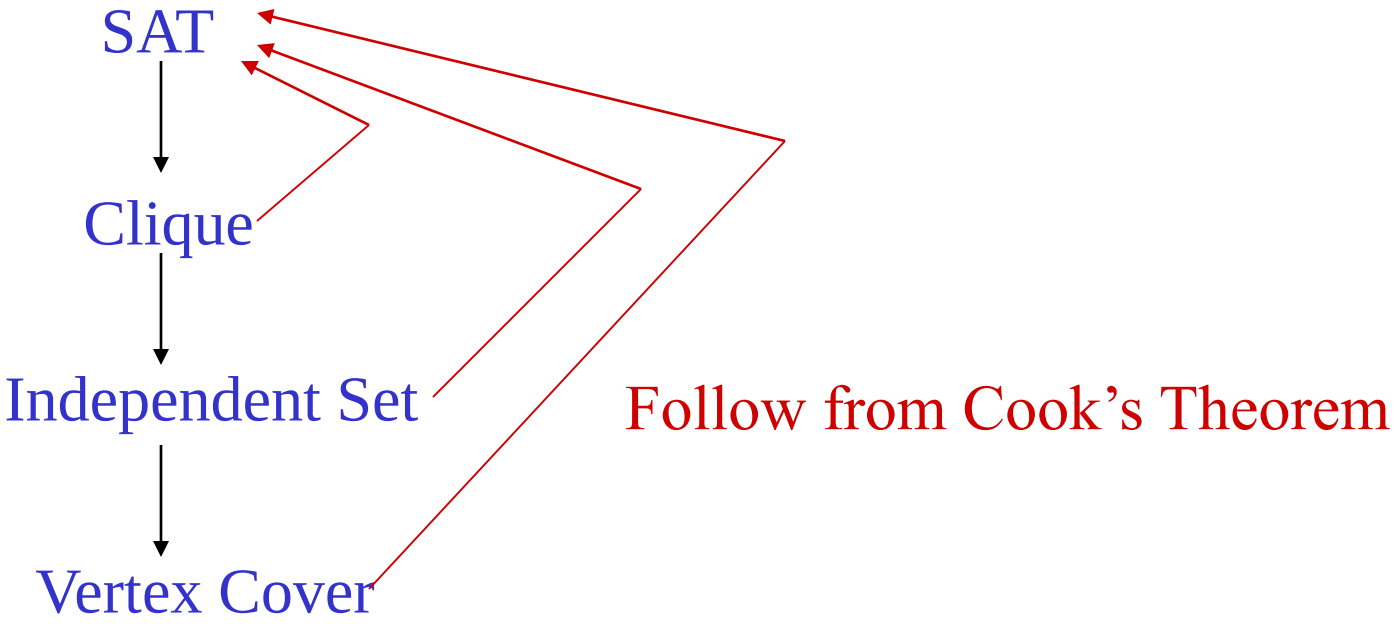
- Satisfiability problem (SAT):
  - Given: a formula φ with m clauses $C_1,\ldots, C_m$ over n variables.

    Example: $x_1 \vee x_2 \vee x_5$, $x_3 \vee \neg x_5$
  - 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.

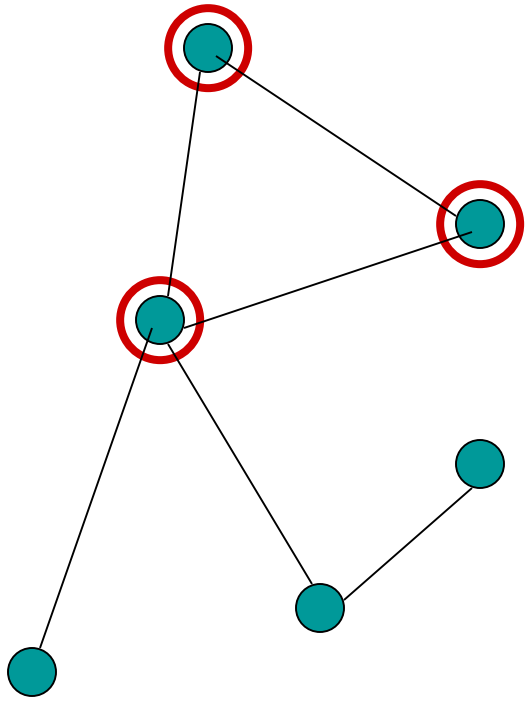
What we will prove



Conclusion: all of the above problems are NP-complete

Clique again

- Clique:
  - Input: undirected graph $G = (V, E)$, K
  - Output: is there a subset C of V, $|C| \geq K$, such that every pair of vertices in C has an edge between them



SAT ≤ Clique

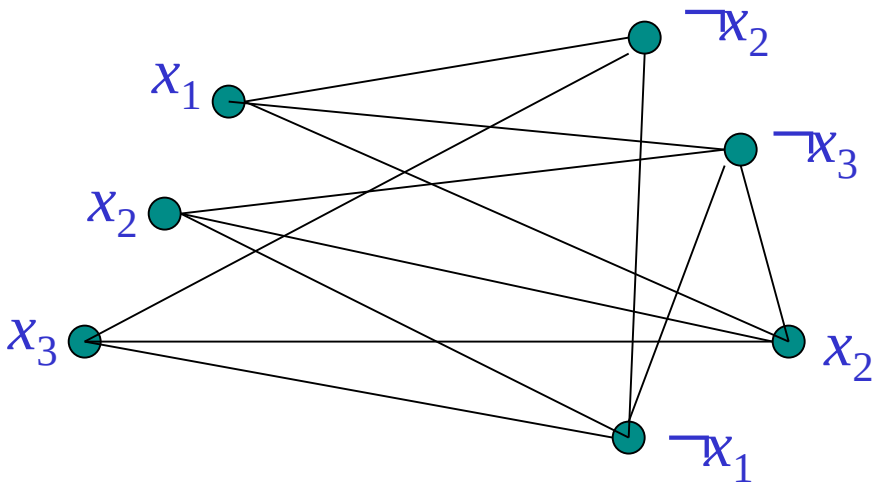
- Given a SAT formula $\varphi = C_1, \ldots, C_m$ over $x_1, \ldots, x_n$, we need to produce $G = (V, E)$ and K, such that φ satisfiable iff G has a clique of size $\geq K$.
- Notation: a literal is either x_i or $\neg x_i$

SAT ≤ 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 ≤ Clique example

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

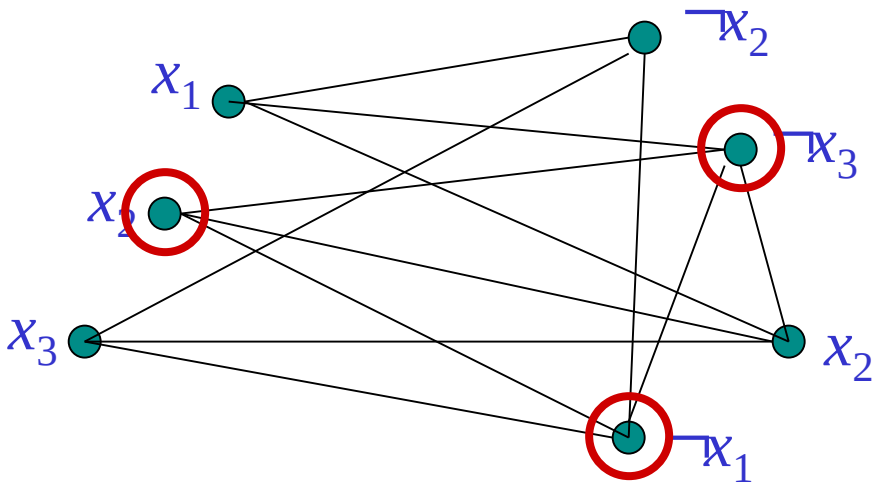


- Claim: φ satisfiable iff G has a clique of size $\geq m$

Proof

- "$\rightarrow$" part:
  - Take any assignment that satisfies φ.

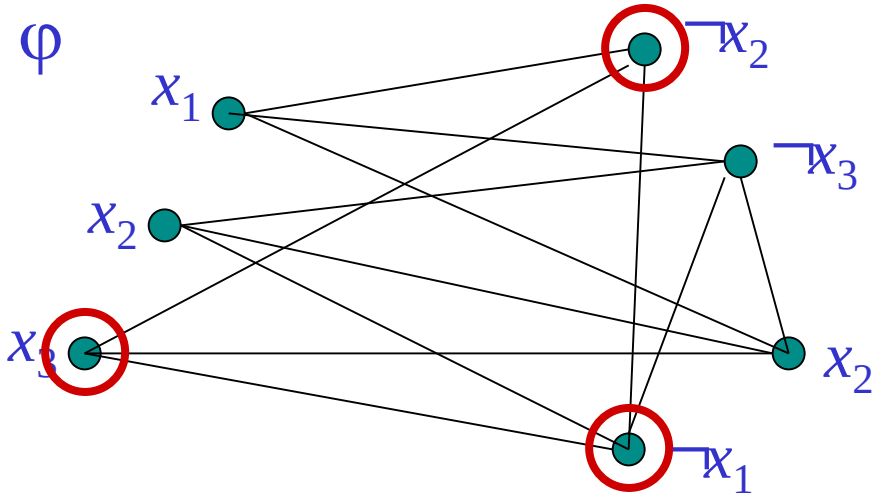
    E.g., x_1 = F, x_2 = T, x_3 = F
  - Let the set C contain one satisfied literal per clause
  - C is a clique



Proof

- "←" part:
  - Take any clique C of size m (i.e., $= m$)
  - Create a set of equations that satisfies selected literals.

    E.g., $x_3 = \mathrm{T}, x_2 = \mathrm{F}, x_1 = \mathrm{F}$
  - The set of equations is consistent and the solution satisfies φ

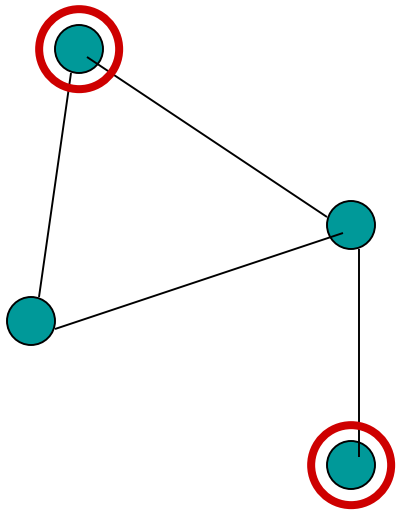


Altogether

- 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 poly time
- Therefore, SAT $\leq$ Clique$\rightarrow$ Clique NP-hard
- Clique is in NP $\rightarrow$ Clique is NP-complete

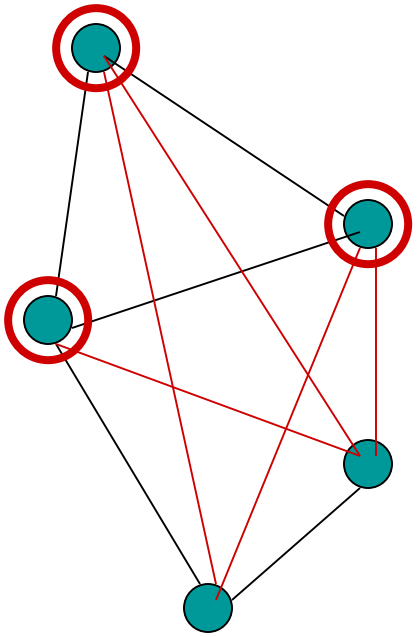
Independent set (IS)

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



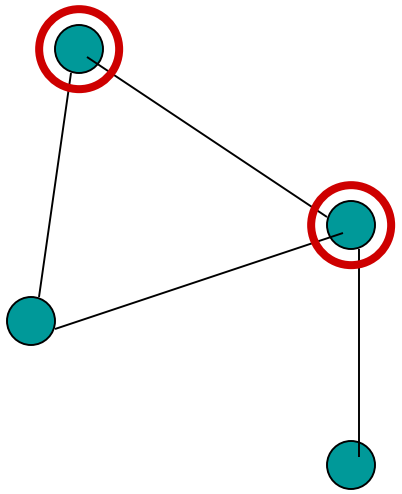
Clique ≤ IS

- Given an input $G = (V, E)$, K to Clique, need to construct an input $G'=(V',E')$, K' to IS, such that G has clique of size K iff G' has IS of size K'.
- Construction: $K'= K$, $V'= V$, $E'= E^c$
- Reason: C is a clique in G iff it is an IS in G's complement.



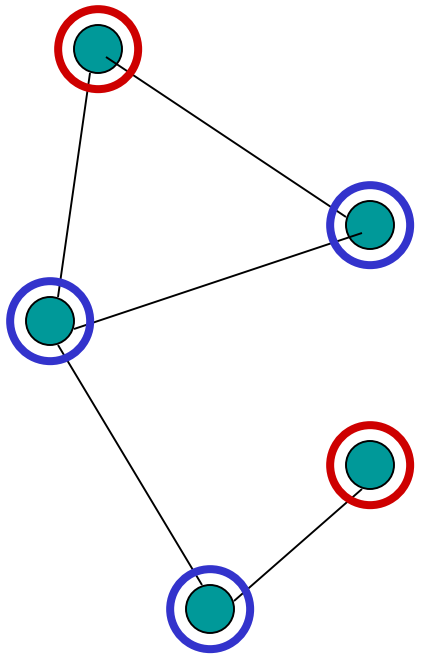
Vertex cover (VC)

- Input: undirected graph $G = (V, E)$
- 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.



IS ≤ VC

- Given an input $G=(V, E)$, K to IS, need to construct an input $G'=(V', E')$, K' to VC, such that G has an IS of size K iff G' has VC of size K'.
- Construction: $V' = V$, $E' = E$, $K' = |V| - K$
- Reason: S is an IS in G iff $V - S$ is a VC in G.



3SAT

- 3SAT:
  - Given: a formula φ with m clauses $C_1,\ldots, C_m$ over n variables such that $|C_i| = 3$ for $1 \leq i \leq m$.

    Example: $x_1 \vee x_2 \vee x_3$, $\neg x_1 \vee \neg x_2 \vee \neg x_3$, $\neg x_1 \vee \neg x_2 \vee \neg x_3$
  - Check if there exists TRUE/FALSE assignments to the variables that makes the formula satisfiable.

SAT $\leq$ 3SAT

Further Reading

- Michael Garey and David Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness.
- Michael Sipser, Introduction to the Theory of Computation, 中信出版社, 2002.
- Christos Papadimitriou, Computational Complexity, 清华大学出版社, 2004.