Foundations of Computing

Lecture 22

Arkady Yerukhimovich

April 10, 2025

Outline

Lecture 21 Review

- $\mathcal{P}$ and $\mathcal{NP}$
- Polynomial-Time Reductions
- $\mathcal{NP}$-completeness of SAT

Outline

What We Already Know

1. SAT is $\mathcal{NP}$-complete
2. 3-SAT is $\mathcal{NP}$-complete
3. 3-SAT $\leq_P$ CLIQUE
4. CLIQUE $\leq_P$ Independent Set

What We Already Know

1. SAT is $\mathcal{NP}$-complete
2. 3-SAT is $\mathcal{NP}$-complete
3. 3-SAT $\leq_P$ CLIQUE
4. CLIQUE $\leq_P$ Independent Set

A Key Tool to Build Reductions

A Key Tool to Build Reductions

Gadgets

A Key Tool to Build Reductions

Gadgets

- Gadgets are structures in the target problem that can simulate structures in the source problem

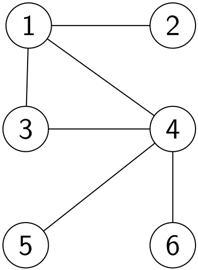
A Key Tool to Build Reductions

Gadgets

- Gadgets are structures in the target problem that can simulate structures in the source problem
- For example, in proof of 3SAT $\leq_P$ CLIQUE
  - We replaced each variable with a node
  - We replaced each clause with 3 nodes (1 for each variable)
  - Edges capture independent variables between clauses

Vertex Covers

Given a graph $G = (V, E)$, a vertex cover is a subset of the nodes $C \subseteq V$ s.t. each edge in E has an end-point in V.



Vertex Cover Problem

Vertex Cover Problem

$$\text{VERTEX-COVER} = \{\langle G, k\rangle \mid G \text{ has a vertex cover of size } \leq k\}$$

Vertex Cover Problem

$$\text{VERTEX-COVER} = \{\langle G, k\rangle \mid G \text{ has a vertex cover of size } \leq k\}$$

Goal: Prove that VC is $\mathcal{NP}$-Complete

Vertex Cover Problem

Vertex Cover Problem

$$\text{VERTEX-COVER} = \{\langle G, k\rangle \mid G \text{ has a vertex cover of size } \leq k\}$$

Goal: Prove that VC is $\mathcal{NP}$-Complete

1. Show that VC $\in \mathcal{NP}$

Vertex Cover Problem

Vertex Cover Problem

$$\text{VERTEX-COVER} = \{\langle G, k\rangle \mid G \text{ has a vertex cover of size } \leq k\}$$

Goal: Prove that VC is $\mathcal{NP}$-Complete

1. Show that VC $\in \mathcal{NP}$
2. Show that 3-SAT $\leq_p$ VC

3-SAT $\leq_p$ VC

Goal: Show reduction f from 3-SAT to VC s.t.

- if ϕ is satisfiable, $f(\phi) = \langle G, k \rangle$ s.t. G has VC of size $\leq k$
- if ϕ is not satisfiable, $f(\phi) = \langle G, k \rangle$ s.t. G has no VC of size $\leq k$

3-SAT $\leq_p$ VC

Goal: Show reduction f from 3-SAT to VC s.t.

- if ϕ is satisfiable, $f(\phi) = \langle G, k \rangle$ s.t. G has VC of size $\leq k$
- if ϕ is not satisfiable, $f(\phi) = \langle G, k \rangle$ s.t. G has no VC of size $\leq k$

Variable gadget:

3-SAT $\leq_p$ VC

Goal: Show reduction f from 3-SAT to VC s.t.

- if ϕ is satisfiable, $f(\phi) = \langle G, k \rangle$ s.t. G has VC of size $\leq k$
- if ϕ is not satisfiable, $f(\phi) = \langle G, k \rangle$ s.t. G has no VC of size $\leq k$

Variable gadget: For every variable x_1, draw pair of nodes

Clause gadget:

3-SAT $\leq_p$ VC

Goal: Show reduction f from 3-SAT to VC s.t.

- if ϕ is satisfiable, $f(\phi) = \langle G, k\rangle$ s.t. G has VC of size $\leq k$
- if ϕ is not satisfiable, $f(\phi) = \langle G, k\rangle$ s.t. G has no VC of size $\leq k$

Variable gadget: For every variable x_1, draw pair of nodes

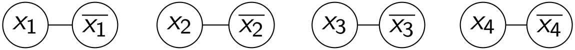
Clause gadget: For every (3-term) clause draw a triangle

Observations:

- For each variable need 1 node in cover
- For each triangle need at least 2 nodes
- Need to connect variables to clauses

3-SAT $\leq_p$ VC Example

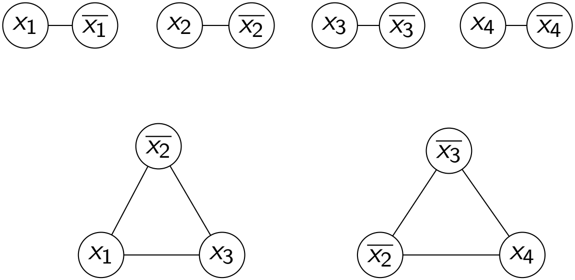
$$\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee x_4)$$



1. A satisfying assignment implies cover C, $|C| \leq 2c + v$
2. No satisfying assignment implies smallest cover needs $|C| > 2c + v$

3-SAT $\leq_p$ VC Example

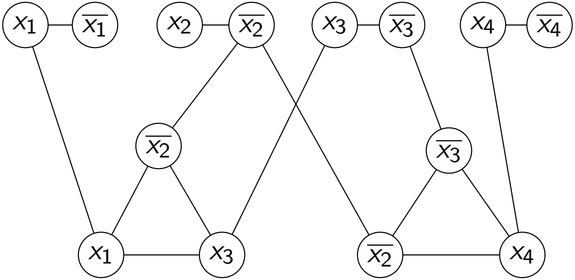
$$\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee x_4)$$



1. A satisfying assignment implies cover C, $|C| \leq 2c + v$
2. No satisfying assignment implies smallest cover needs $|C| > 2c + v$

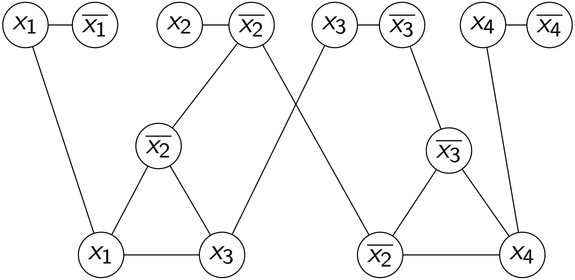
3-SAT $\leq_p$ VC Example

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



3-SAT $\leq_p$ VC Example

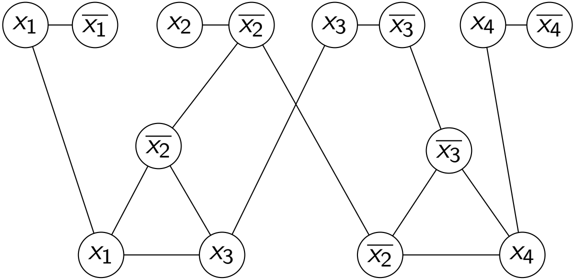
$$\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee x_4)$$



1. A satisfying assignment implies cover C, $|C| \leq 2c + v$

3-SAT $\leq_p$ VC Example

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



1. A satisfying assignment implies cover C, $|C| \leq 2c + v$
2. No satisfying assignment implies smallest cover needs $|C| > 2c + v$

Outline

3-Coloring

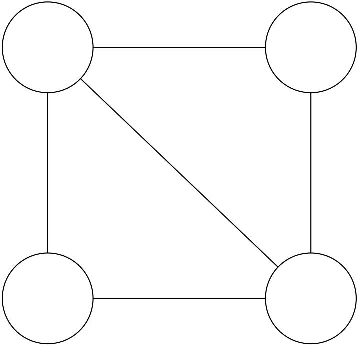
Definition

An undirected graph G is 3-colorable, if can assign colors $\{0, 1, 2\}$ to all nodes, such that no edges have the same color on both ends.

3-Coloring

Definition

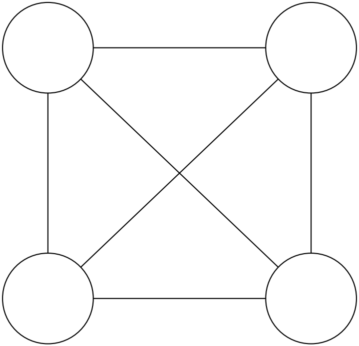
An undirected graph G is 3-colorable, if can assign colors $\{0, 1, 2\}$ to all nodes, such that no edges have the same color on both ends.



3-Coloring

Definition

An undirected graph G is 3-colorable, if can assign colors $\{0, 1, 2\}$ to all nodes, such that no edges have the same color on both ends.



Goal: Prove than 3-Coloring is $\mathcal{NP}$-Complete

NAE-3SAT

NAE-kSAT Problem

$$\text{NAE-kSAT} = \{\langle\phi\rangle \quad | \quad \phi \text{ is in } k\text{-CNF and } \phi \text{ has a satisfying assignment s.t. each clause has at least one 0 and at least one 1}\}$$

NAE-3SAT

NAE-kSAT Problem

$$\text{NAE-kSAT} = \{\langle \phi \rangle \mid \phi \text{ is in } k\text{-CNF and } \phi \text{ has a satisfying assignment s.t. each clause has at least one 0 and at least one 1}\}$$

Definition:

- x is an NAE-assignment of ϕ if $\phi(x) = 1$ and x does not assign all 1s or all 0s to any clause

NAE-3SAT

NAE-kSAT Problem

$$\text{NAE-kSAT} = \{\langle\phi\rangle \mid \phi \text{ is in } k\text{-CNF and } \phi \text{ has a satisfying assignment s.t. each clause has at least one 0 and at least one 1}\}$$

Definition:

- x is an NAE-assignment of ϕ if $\phi(x) = 1$ and x does not assign all 1s or all 0s to any clause

Lemma: If x is NAE-assignment of ϕ then $\overline{x}$ is NAE-assignment of ϕ

Proof:

- x must assign at least one 1 and at least one 0 to every clause
- $\overline{x}$ must also have at least one 1 and one 0 in every clause
- This means every clause is satisfied, and ϕ is satisfied since it's CNF

Goal

Prove that NAE-3SAT is $\mathcal{NP}$-complete: 3SAT $\leq_P$ NAE-3SAT

3SAT $\leq_P$ NAE-3SAT

- Turns out it's not easy to directly prove this
- Instead we take two steps:

$$\text{3SAT} \ \leq_P \ \text{NAE-4SAT} \ \leq_P \ \text{NAE-3SAT}$$

3SAT $\leq_P$ NAE-4SAT

- We need a reduction f that takes 3SAT instance ϕ and converts it into NAE-4SAT instance ϕ'

3SAT $\leq_P$ NAE-4SAT

- We need a reduction f that takes 3SAT instance ϕ and converts it into NAE-4SAT instance ϕ'
  - If ϕ is satisfiable, ϕ' is NAE-satisfiable

3SAT $\leq_P$ NAE-4SAT

- We need a reduction f that takes 3SAT instance ϕ and converts it into NAE-4SAT instance ϕ'
  - If ϕ is satisfiable, ϕ' is NAE-satisfiable
  - If ϕ' is NAE-satisfiable, ϕ is satisfiable

3SAT $\leq_P$ NAE-4SAT

- We need a reduction f that takes 3SAT instance ϕ and converts it into NAE-4SAT instance ϕ'
  - If ϕ is satisfiable, ϕ' is NAE-satisfiable
  - If ϕ' is NAE-satisfiable, ϕ is satisfiable
  - Note that this must hold for every clause of ϕ, ϕ'

3SAT $\leq_P$ NAE-4SAT

- We need a reduction f that takes 3SAT instance ϕ and converts it into NAE-4SAT instance ϕ'
  - If ϕ is satisfiable, ϕ' is NAE-satisfiable
  - If ϕ' is NAE-satisfiable, ϕ is satisfiable
  - Note that this must hold for every clause of ϕ, ϕ'
- Idea: Can we build a "gadget" for each clause of ϕ to enforce this condition?

3SAT $\leq_P$ NAE-4SAT

- We need a reduction f that takes 3SAT instance ϕ and converts it into NAE-4SAT instance ϕ'
  - If ϕ is satisfiable, ϕ' is NAE-satisfiable
  - If ϕ' is NAE-satisfiable, ϕ is satisfiable
  - Note that this must hold for every clause of ϕ, ϕ'
- Idea: Can we build a "gadget" for each clause of ϕ to enforce this condition?

$$(x_1 \vee x_2 \vee x_3) \rightarrow (x_1 \vee x_2 \vee x_3 \vee S)$$

3SAT $\leq_P$ NAE-4SAT

- We need a reduction f that takes 3SAT instance ϕ and converts it into NAE-4SAT instance ϕ'
  - If ϕ is satisfiable, ϕ' is NAE-satisfiable
  - If ϕ' is NAE-satisfiable, ϕ is satisfiable
  - Note that this must hold for every clause of ϕ, ϕ'
- Idea: Can we build a "gadget" for each clause of ϕ to enforce this condition?

$$(x_1 \vee x_2 \vee x_3) \rightarrow (x_1 \vee x_2 \vee x_3 \vee S)$$

- Why this works:

3SAT $\leq_P$ NAE-4SAT

- We need a reduction f that takes 3SAT instance ϕ and converts it into NAE-4SAT instance ϕ'
  - If ϕ is satisfiable, ϕ' is NAE-satisfiable
  - If ϕ' is NAE-satisfiable, ϕ is satisfiable
  - Note that this must hold for every clause of ϕ, ϕ'
- Idea: Can we build a "gadget" for each clause of ϕ to enforce this condition?

$$(x_1 \vee x_2 \vee x_3) \rightarrow (x_1 \vee x_2 \vee x_3 \vee S)$$

- Why this works:
  - Want to argue:
    - A satisfying assignment to ϕ implies satisfying assignment to ϕ'
    - A satisfying assignment to ϕ' implies satisfying assignment to ϕ

3SAT $\leq_P$ NAE-4SAT

- We need a reduction f that takes 3SAT instance ϕ and converts it into NAE-4SAT instance ϕ'
  - If ϕ is satisfiable, ϕ' is NAE-satisfiable
  - If ϕ' is NAE-satisfiable, ϕ is satisfiable
  - Note that this must hold for every clause of ϕ, ϕ'
- Idea: Can we build a "gadget" for each clause of ϕ to enforce this condition?

$$(x_1 \vee x_2 \vee x_3) \rightarrow (x_1 \vee x_2 \vee x_3 \vee S)$$

- Why this works:
  - Want to argue:
    - A satisfying assignment to ϕ implies satisfying assignment to ϕ'
    - A satisfying assignment to ϕ' implies satisfying assignment to ϕ
  - ($\Rightarrow$) If $(x_1 \vee x_2 \vee x_3) = 1$

3SAT $\leq_P$ NAE-4SAT

- We need a reduction f that takes 3SAT instance ϕ and converts it into NAE-4SAT instance ϕ'
  - If ϕ is satisfiable, ϕ' is NAE-satisfiable
  - If ϕ' is NAE-satisfiable, ϕ is satisfiable
  - Note that this must hold for every clause of ϕ, ϕ'
- Idea: Can we build a "gadget" for each clause of ϕ to enforce this condition?

$$(x_1 \vee x_2 \vee x_3) \rightarrow (x_1 \vee x_2 \vee x_3 \vee S)$$

- Why this works:
  - Want to argue:
    - A satisfying assignment to ϕ implies satisfying assignment to ϕ'
    - A satisfying assignment to ϕ' implies satisfying assignment to ϕ
  - ($\Rightarrow$) If $(x_1 \vee x_2 \vee x_3) = 1$ at least one $x_i = 1$, so $(x_1 \vee x_2 \vee x_3 \vee S) = 1$. Set $S = 0$ to make it NAE-assignment

3SAT $\leq_P$ NAE-4SAT

- We need a reduction f that takes 3SAT instance ϕ and converts it into NAE-4SAT instance ϕ'
  - If ϕ is satisfiable, ϕ' is NAE-satisfiable
  - If ϕ' is NAE-satisfiable, ϕ is satisfiable
  - Note that this must hold for every clause of ϕ, ϕ'
- Idea: Can we build a "gadget" for each clause of ϕ to enforce this condition?

$$(x_1 \vee x_2 \vee x_3) \rightarrow (x_1 \vee x_2 \vee x_3 \vee S)$$

- Why this works:
  - Want to argue:
    - A satisfying assignment to ϕ implies satisfying assignment to ϕ'
    - A satisfying assignment to ϕ' implies satisfying assignment to ϕ
  - ($\Rightarrow$) If $(x_1 \vee x_2 \vee x_3) = 1$ at least one $x_i = 1$, so $(x_1 \vee x_2 \vee x_3 \vee S) = 1$. Set $S = 0$ to make it NAE-assignment
  - ($\Leftarrow$) If $(x_1 \vee x_2 \vee x_3 \vee S) = 1$

3SAT $\leq_P$ NAE-4SAT

- We need a reduction f that takes 3SAT instance ϕ and converts it into NAE-4SAT instance ϕ'
  - If ϕ is satisfiable, ϕ' is NAE-satisfiable
  - If ϕ' is NAE-satisfiable, ϕ is satisfiable
  - Note that this must hold for every clause of ϕ, ϕ'
- Idea: Can we build a "gadget" for each clause of ϕ to enforce this condition?

$$(x_1 \vee x_2 \vee x_3) \rightarrow (x_1 \vee x_2 \vee x_3 \vee S)$$

- Why this works:
  - Want to argue:
    - A satisfying assignment to ϕ implies satisfying assignment to ϕ'
    - A satisfying assignment to ϕ' implies satisfying assignment to ϕ
  - ($\Rightarrow$) If $(x_1 \vee x_2 \vee x_3) = 1$ at least one $x_i = 1$, so $(x_1 \vee x_2 \vee x_3 \vee S) = 1$. Set $S = 0$ to make it NAE-assignment
  - ($\Leftarrow$) If $(x_1 \vee x_2 \vee x_3 \vee S) = 1$
    - If $S = 0$, then at least one $x_i = 1$, so $(x_1 \vee x_2 \vee x_3) = 1$

3SAT $\leq_P$ NAE-4SAT

- We need a reduction f that takes 3SAT instance ϕ and converts it into NAE-4SAT instance ϕ'
  - If ϕ is satisfiable, ϕ' is NAE-satisfiable
  - If ϕ' is NAE-satisfiable, ϕ is satisfiable
  - Note that this must hold for every clause of ϕ, ϕ'
- Idea: Can we build a "gadget" for each clause of ϕ to enforce this condition?

$$(x_1 \vee x_2 \vee x_3) \rightarrow (x_1 \vee x_2 \vee x_3 \vee S)$$

- Why this works:
  - Want to argue:
    - A satisfying assignment to ϕ implies satisfying assignment to ϕ'
    - A satisfying assignment to ϕ' implies satisfying assignment to ϕ
  - ($\Rightarrow$) If $(x_1 \vee x_2 \vee x_3) = 1$ at least one $x_i = 1$, so $(x_1 \vee x_2 \vee x_3 \vee S) = 1$. Set $S = 0$ to make it NAE-assignment
  - ($\Leftarrow$) If $(x_1 \vee x_2 \vee x_3 \vee S) = 1$
    - If $S = 0$, then at least one $x_i = 1$, so $(x_1 \vee x_2 \vee x_3) = 1$
    - If $S = 1$, then $(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee 0)$ is also NAE-assignment. So, $(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) = 1$, so just flip all assignments in ϕ'

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE
- Observations: Starting with $(x_1 \vee x_2 \vee x_3 \vee x_4)$

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE
- Observations: Starting with $(x_1 \vee x_2 \vee x_3 \vee x_4)$
  - We know that not all x_i have the same value

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE
- Observations: Starting with $(x_1 \vee x_2 \vee x_3 \vee x_4)$
  - We know that not all x_i have the same value
  - At least one of x_i is a 1 and one is a 0

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE
- Observations: Starting with $(x_1 \vee x_2 \vee x_3 \vee x_4)$
  - We know that not all x_i have the same value
  - At least one of x_i is a 1 and one is a 0
  - Idea: Let's split the variables into two clauses:

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE
- Observations: Starting with $(x_1 \vee x_2 \vee x_3 \vee x_4)$
  - We know that not all x_i have the same value
  - At least one of x_i is a 1 and one is a 0
  - Idea: Let's split the variables into two clauses:

$$(x_1 \vee x_2 \vee x_3 \vee x_4) \rightarrow (x_1 \vee x_2 \vee z_i) \wedge (x_3 \vee x_4 \vee \overline{z_i})$$

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE
- Observations: Starting with $(x_1 \vee x_2 \vee x_3 \vee x_4)$
  - We know that not all x_i have the same value
  - At least one of x_i is a 1 and one is a 0
  - Idea: Let's split the variables into two clauses:

$$(x_1 \vee x_2 \vee x_3 \vee x_4) \to (x_1 \vee x_2 \vee z_i) \wedge (x_3 \vee x_4 \vee \overline{z_i})$$

- Why this works:

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE
- Observations: Starting with $(x_1 \vee x_2 \vee x_3 \vee x_4)$
  - We know that not all x_i have the same value
  - At least one of x_i is a 1 and one is a 0
  - Idea: Let's split the variables into two clauses:
    $$(x_1 \vee x_2 \vee x_3 \vee x_4) \to (x_1 \vee x_2 \vee z_i) \wedge (x_3 \vee x_4 \vee \overline{z_i})$$
- Why this works:
  - $(\Leftarrow)$ If $(x_1 \vee x_2 \vee z_i)$ and $(x_3 \vee x_4 \vee \overline{z_i})$ are both NAE

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE
- Observations: Starting with $(x_1 \vee x_2 \vee x_3 \vee x_4)$
  - We know that not all x_i have the same value
  - At least one of x_i is a 1 and one is a 0
  - Idea: Let's split the variables into two clauses:

$$(x_1 \vee x_2 \vee x_3 \vee x_4) \to (x_1 \vee x_2 \vee z_i) \wedge (x_3 \vee x_4 \vee \overline{z_i})$$

- Why this works:
  - $(\Leftarrow)$ If $(x_1 \vee x_2 \vee z_i)$ and $(x_3 \vee x_4 \vee \overline{z_i})$ are both NAE, then $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is NAE

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE
- Observations: Starting with $(x_1 \vee x_2 \vee x_3 \vee x_4)$
  - We know that not all x_i have the same value
  - At least one of x_i is a 1 and one is a 0
  - Idea: Let's split the variables into two clauses:

$$(x_1 \vee x_2 \vee x_3 \vee x_4) \rightarrow (x_1 \vee x_2 \vee z_i) \wedge (x_3 \vee x_4 \vee \overline{z_i})$$

- Why this works:
  - $(\Leftarrow)$ If $(x_1 \vee x_2 \vee z_i)$ and $(x_3 \vee x_4 \vee \overline{z_i})$ are both NAE, then $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is NAE
  - $(\Rightarrow)$ If $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is NAE

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE
- Observations: Starting with $(x_1 \vee x_2 \vee x_3 \vee x_4)$
  - We know that not all x_i have the same value
  - At least one of x_i is a 1 and one is a 0
  - Idea: Let's split the variables into two clauses:
    $$(x_1 \vee x_2 \vee x_3 \vee x_4) \rightarrow (x_1 \vee x_2 \vee z_i) \wedge (x_3 \vee x_4 \vee \overline{z_i})$$
- Why this works:
  - $(\Leftarrow)$ If $(x_1 \vee x_2 \vee z_i)$ and $(x_3 \vee x_4 \vee \overline{z_i})$ are both NAE, then $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is NAE
  - $(\Rightarrow)$ If $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is NAE
    - If $x_1 \neq x_2$: Set $z_i = x_3$

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE
- Observations: Starting with $(x_1 \vee x_2 \vee x_3 \vee x_4)$
  - We know that not all x_i have the same value
  - At least one of x_i is a 1 and one is a 0
  - Idea: Let's split the variables into two clauses:
    $$(x_1 \vee x_2 \vee x_3 \vee x_4) \to (x_1 \vee x_2 \vee z_i) \wedge (x_3 \vee x_4 \vee \overline{z_i})$$
- Why this works:
  - $(\Leftarrow)$ If $(x_1 \vee x_2 \vee z_i)$ and $(x_3 \vee x_4 \vee \overline{z_i})$ are both NAE, then $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is NAE
  - $(\Rightarrow)$ If $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is NAE
    - If $x_1 \neq x_2$: Set $z_i = x_3$
    - If $x_1 \neq x_3$: Set $z_i = x_3$

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE
- Observations: Starting with $(x_1 \vee x_2 \vee x_3 \vee x_4)$
  - We know that not all x_i have the same value
  - At least one of x_i is a 1 and one is a 0
  - Idea: Let's split the variables into two clauses:
    $$(x_1 \vee x_2 \vee x_3 \vee x_4) \rightarrow (x_1 \vee x_2 \vee z_i) \wedge (x_3 \vee x_4 \vee \overline{z_i})$$
- Why this works:
  - $(\Leftarrow)$ If $(x_1 \vee x_2 \vee z_i)$ and $(x_3 \vee x_4 \vee \overline{z_i})$ are both NAE, then $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is NAE
  - $(\Rightarrow)$ If $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is NAE
    - If $x_1 \neq x_2$: Set $z_i = x_3$
    - If $x_1 \neq x_3$: Set $z_i = x_3$
    - If $x_1 \neq x_4$: Set $z_i = x_4$

NAE-4SAT $\leq_P$ NAE-3SAT

- Need a gadget to convert 4-CNF clause to 3-CNF clauses that preserves NAE
- Observations: Starting with $(x_1 \vee x_2 \vee x_3 \vee x_4)$
  - We know that not all x_i have the same value
  - At least one of x_i is a 1 and one is a 0
  - Idea: Let's split the variables into two clauses:
    $$(x_1 \vee x_2 \vee x_3 \vee x_4) \rightarrow (x_1 \vee x_2 \vee z_i) \wedge (x_3 \vee x_4 \vee \overline{z_i})$$
- Why this works:
  - $(\Leftarrow)$ If $(x_1 \vee x_2 \vee z_i)$ and $(x_3 \vee x_4 \vee \overline{z_i})$ are both NAE, then $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is NAE
  - $(\Rightarrow)$ If $(x_1 \vee x_2 \vee x_3 \vee x_4)$ is NAE
    - If $x_1 \neq x_2$: Set $z_i = x_3$
    - If $x_1 \neq x_3$: Set $z_i = x_3$
    - If $x_1 \neq x_4$: Set $z_i = x_4$

Theorem

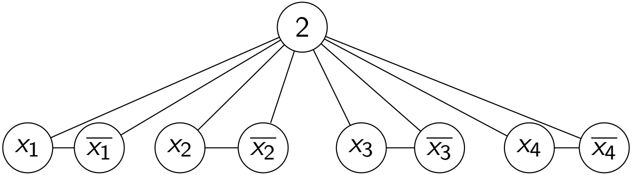
$$\text{3SAT} \leq_P \text{NAE-4SAT} \leq_P \text{NAE-3SAT}$$

NAE-3SAT $\leq_p$ 3-Coloring

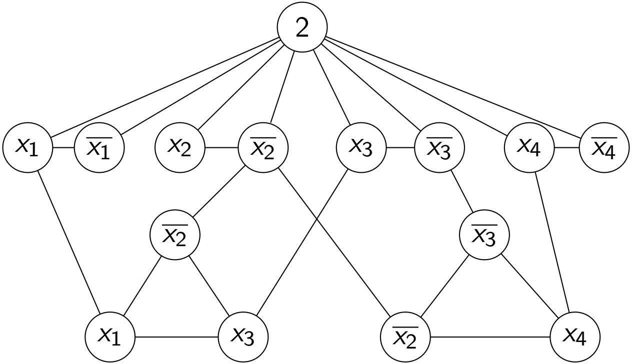
$$\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee x_4)$$

NAE-3SAT $\leq_p$ 3-Coloring

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

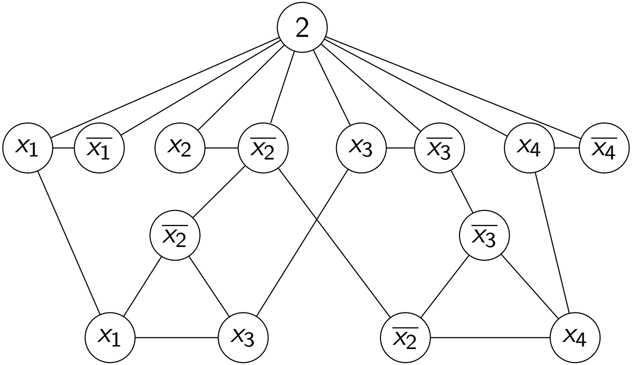


$$\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee x_4)$$



NAE-3SAT $\leq_p$ 3-Coloring

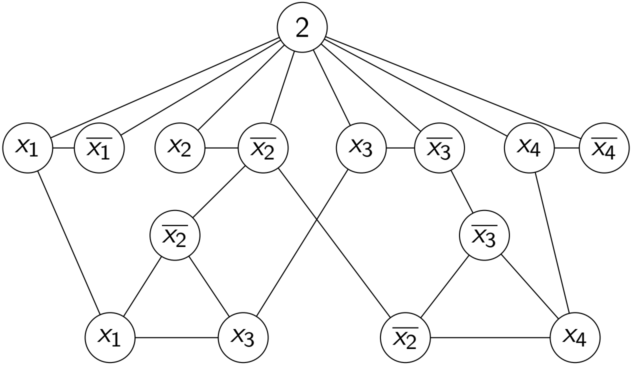
$$\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee x_4)$$



1. If ϕ is NAE-SAT, then not all variables are all 0 or all 1. So, enough colors to color clauses

NAE-3SAT $\leq_p$ 3-Coloring

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



1. If ϕ is NAE-SAT, then not all variables are all 0 or all 1. So, enough colors to color clauses
2. If G is 3-colorable, colors indicate a NAE-SAT assignment

Conclusions

- Many useful problems are $\mathcal{NP}$-complete

Conclusions

- Many useful problems are $\mathcal{NP}$-complete
- But, as long as $\mathcal{P} \neq \mathcal{NP}$, these are hard

Conclusions

- Many useful problems are $\mathcal{NP}$-complete
- But, as long as $\mathcal{P} \neq \mathcal{NP}$, these are hard
- Given a problem L, you should:

Conclusions

- Many useful problems are $\mathcal{NP}$-complete
- But, as long as $\mathcal{P} \neq \mathcal{NP}$, these are hard
- Given a problem L, you should:
  1. Try to solve it ($L \in \mathcal{P}$)

Conclusions

- Many useful problems are $\mathcal{NP}$-complete
- But, as long as $\mathcal{P} \neq \mathcal{NP}$, these are hard
- Given a problem L, you should:
  1. Try to solve it ($L \in \mathcal{P}$)
  2. Try to prove $\mathcal{NP}$-complete

Conclusions

- Many useful problems are $\mathcal{NP}$-complete
- But, as long as $\mathcal{P} \neq \mathcal{NP}$, these are hard
- Given a problem L, you should:
  1. Try to solve it ($L \in \mathcal{P}$)
  2. Try to prove $\mathcal{NP}$-complete
- But, you must be careful

Conclusions

- Many useful problems are $\mathcal{NP}$-complete
- But, as long as $\mathcal{P} \neq \mathcal{NP}$, these are hard
- Given a problem L, you should:
    1. Try to solve it ($L \in \mathcal{P}$)
    2. Try to prove $\mathcal{NP}$-complete
- But, you must be careful

$$\text{3-Coloring is } \mathcal{NP}\text{-complete, but 2-Coloring} \in \mathcal{P}$$

Outline

Are All Problems in $\mathcal{NP}$?

Question

Do all languages have poly-size proofs?

Are All Problems in $\mathcal{NP}$?

Question

Do all languages have poly-size proofs?

Consider the following language:

UNSAT

$$\text{UNSAT} = \{\langle\phi\rangle \mid \phi \text{ is not satisfiable}\}$$

Are All Problems in $\mathcal{NP}$?

Question

Do all languages have poly-size proofs?

Consider the following language:

UNSAT

$$\text{UNSAT} = \{\langle\phi\rangle \mid \phi \text{ is not satisfiable}\}$$

Problems like UNSAT are in co-$\mathcal{NP}$

$\mathcal{P}$, $\mathcal{NP}$ and co-$\mathcal{NP}$

$\mathcal{P}$

$L \in \mathcal{P}$ if there exists poly-time DTM M s.t $M(x) = [x \in L]$

$\mathcal{P}$, $\mathcal{NP}$ and co-$\mathcal{NP}$

$\mathcal{P}$

$L \in \mathcal{P}$ if there exists poly-time DTM M s.t $M(x) = [x \in L]$

$\mathcal{NP}$

$L \in \mathcal{NP}$ if there exists poly-time DTM V s.t. for $x \in L$ there exists a witness w s.t. $V(x, w) = 1$

$\mathcal{P}$, $\mathcal{NP}$ and co-$\mathcal{NP}$

$\mathcal{P}$

$L \in \mathcal{P}$ if there exists poly-time DTM M s.t $M(x) = [x \in L]$

$\mathcal{NP}$

$L \in \mathcal{NP}$ if there exists poly-time DTM V s.t. for $x \in L$ there exists a witness w s.t. $V(x, w) = 1$

co-$\mathcal{NP}$

$L \in$ co-$\mathcal{NP}$ if there exists poly-time DTM V s.t. for $x \in L$ for all w, $V(x, w) = 0$

$\mathcal{P}$, $\mathcal{NP}$ and co-$\mathcal{NP}$

$\mathcal{P}$

$L \in \mathcal{P}$ if there exists poly-time DTM M s.t $M(x) = [x \in L]$

$\mathcal{NP}$

$L \in \mathcal{NP}$ if there exists poly-time DTM V s.t. for $x \in L$ there exists a witness w s.t. $V(x, w) = 1$

co-$\mathcal{NP}$

$L \in$ co-$\mathcal{NP}$ if there exists poly-time DTM V s.t. for $x \in L$ for all w, $V(x, w) = 0$

Question:

$$\text{Is } \mathcal{NP} = \text{co-}\mathcal{NP}?$$

Polynomial Hierarchy (PH)

We can continue in this way to define more powerful classes of languages:

Polynomial Hierarchy (PH)

We can continue in this way to define more powerful classes of languages:

Σ_2^p (Generalization of $\mathcal{NP}$

$L \in \Sigma_2^p$ if there exists poly-time DTM V s.t. for $x \in L$, there exists a w_1 s.t. for all w_2, $V(x, w_1, w_2) = 1$

$$\exists w_1 \forall w_2 \text{ s.t. } V(x, w_1, w_2) = 1$$

Polynomial Hierarchy (PH)

We can continue in this way to define more powerful classes of languages:

Σ_2^p (Generalization of $\mathcal{NP}$

$L \in \Sigma_2^p$ if there exists poly-time DTM V s.t. for $x \in L$, there exists a w_1 s.t. for all w_2, $V(x, w_1, w_2) = 1$

$$\exists w_1 \forall w_2 \text{ s.t. } V(x, w_1, w_2) = 1$$

Π_2^p (Generalization of co-$\mathcal{NP}$)

$L \in \Pi_2^p$ if there exists poly-time DTM V s.t. for $x \in L$, for all w_1 there exists w_2 s.t. $V(x, w_1, w_2) = 1$

$$\forall w_1 \exists w_2 \text{ s.t. } V(x, w_1, w_2) = 1$$

Polynomial Hierarchy (PH)

We can continue in this way to define more powerful classes of languages:

Σ_2^p (Generalization of $\mathcal{NP}$

$L \in \Sigma_2^p$ if there exists poly-time DTM V s.t. for $x \in L$, there exists a w_1 s.t. for all w_2, $V(x, w_1, w_2) = 1$

$$\exists w_1 \forall w_2 \text{ s.t. } V(x, w_1, w_2) = 1$$

Π_2^p (Generalization of co-$\mathcal{NP}$)

$L \in \Pi_2^p$ if there exists poly-time DTM V s.t. for $x \in L$, for all w_1 there exists w_2 s.t. $V(x, w_1, w_2) = 1$

$$\forall w_1 \exists w_2 \text{ s.t. } V(x, w_1, w_2) = 1$$

We believe that there are infinitely many levels of the polynomial hierarchy and that $\Pi_i^p \neq \Sigma_i^p$ for $i > 0$, but can't prove it.

The Complexity Zoo

- There are many other complexity classes

The Complexity Zoo

- There are many other complexity classes
- We know some relationships between classes

The Complexity Zoo

- There are many other complexity classes
- We know some relationships between classes
- But, most big questions (e.g., $\mathcal{P} = \mathcal{NP}$, $\mathcal{NP} =$ co-$\mathcal{NP}$, does PH collapse) are still not known!!!

The Complexity Zoo

- There are many other complexity classes
- We know some relationships between classes
- But, most big questions (e.g., $\mathcal{P} = \mathcal{NP}$, $\mathcal{NP} = \text{co-}\mathcal{NP}$, does PH collapse) are still not known!!!

Complexity Zoo

The complexity zoo (`https://complexityzoo.net/Complexity_Zoo`) now has 546 complexity classes.