Reductions and Completeness

AA FIB, UPC

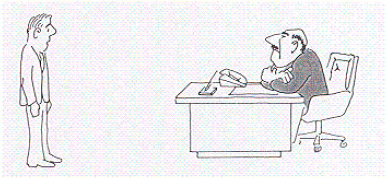
Spring 2025-2026

P vs NP

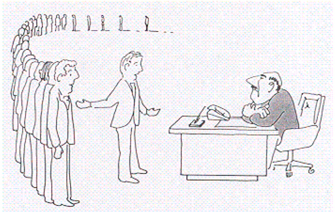
The US$ 10^6 Question: Is P $\neq$ NP or P $=$ NP ?

http://www.claymath.org/prizeproblems/pvsnp.html

Cartoons from Garey-Johnson's book



"I can't find an efficient algorithm,
I guess I'm just too dumb."



"I can't find an efficient algorithm,
but neither can all these famous people."

Bounded many-one reducibility

- *Problem A is* many-one reducible *to problem B ($A \leq_m B$)*
  if there is a computable function f so that $x \in L_A$ iff $f(x) \in L_B$.

Bounded many-one reducibility

- *Problem A is* many-one reducible *to problem B ($A \leq_m B$)*
  if there is a computable function f so that $x \in L_A$ iff $f(x) \in L_B$.
- f is called the reduction.

Bounded many-one reducibility

- *Problem A is* many-one reducible *to problem B ($A \leq_m B$)*
  if there is a computable function f so that $x \in L_A$ iff $f(x) \in L_B$.
- f is called the reduction.
- $\leq_m$ transfer the property of being decidable. To be able to have this kind of result for complexity classed we have to bound the resources that can be used in the reduction.

Bounded many-one reducibility

- *Problem A is* many-one reducible *to problem B ($A \leq_m B$)* if there is a computable function f so that $x \in L_A$ iff $f(x) \in L_B$.
- f is called the reduction.
- $\leq_m$ transfer the property of being decidable. To be able to have this kind of result for complexity classed we have to bound the resources that can be used in the reduction.
- *Problem A is* poly time many-one reducible *to problem B ($A \leq_m^p B$)* if there is a function f computable in polynomial time so that $x \in L_A$ iff $f(x) \in L_B$.

Bounded many-one reducibility

- *Problem A is* many-one reducible *to problem B* ($A \leq_m B$) if there is a computable function f so that $x \in L_A$ iff $f(x) \in L_B$.
- f is called the reduction.
- $\leq_m$ transfer the property of being decidable. To be able to have this kind of result for complexity classed we have to bound the resources that can be used in the reduction.
- *Problem A is* poly time many-one reducible *to problem B* ($A \leq_m^p B$) if there is a function f computable in polynomial time so that $x \in L_A$ iff $f(x) \in L_B$.
- *Problem A is* log space many-one reducible *to problem B* ($A \leq_m^L B$) if there is a function f computable in logarithmic space so that $x \in L_A$ iff $f(x) \in L_B$.

Completeness

- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems.$\mathcal{C}$ is closed under $\leq$ if $A \leq B$ and $B \in \mathcal{C}$ implies $A \in \mathcal{C}$.

Completeness

- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems.$\mathcal{C}$ is closed under $\leq$ if $A \leq B$ and $B \in \mathcal{C}$ implies $A \in \mathcal{C}$.
- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems closed under $\leq$

Completeness

- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems.$\mathcal{C}$ is closed under $\leq$ if $A \leq B$ and $B \in \mathcal{C}$ implies $A \in \mathcal{C}$.
- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems closed under $\leq$
  - Problem B is hard for class $\mathcal{C}$ under $\leq$ if, for each $A \in \mathcal{C}$, $A \leq B$.

Completeness

- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems.$\mathcal{C}$ is closed under $\leq$ if $A \leq B$ and $B \in \mathcal{C}$ implies $A \in \mathcal{C}$.
- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems closed under $\leq$
  - Problem B is hard for class $\mathcal{C}$ under $\leq$ if, for each $A \in \mathcal{C}$, $A \leq B$.
  - Problem B is complete for class $\mathcal{C}$ under $\leq$ if, B is hard for $\mathcal{C}$ and $B \in \mathcal{C}$.

Completeness

- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems.$\mathcal{C}$ is closed under $\leq$ if $A \leq B$ and $B \in \mathcal{C}$ implies $A \in \mathcal{C}$.
- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems closed under $\leq$
  - Problem B is hard for class $\mathcal{C}$ under $\leq$ if, for each $A \in \mathcal{C}$, $A \leq B$.
  - Problem B is complete for class $\mathcal{C}$ under $\leq$ if, B is hard for $\mathcal{C}$ and $B \in \mathcal{C}$.
- All complexity classes we mentioned earlier are closed under $\leq_m^L$.

Completeness

- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems.$\mathcal{C}$ is closed under $\leq$ if $A \leq B$ and $B \in \mathcal{C}$ implies $A \in \mathcal{C}$.
- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems closed under $\leq$
  - Problem B is hard for class $\mathcal{C}$ under $\leq$ if, for each $A \in \mathcal{C}$, $A \leq B$.
  - Problem B is complete for class $\mathcal{C}$ under $\leq$ if, B is hard for $\mathcal{C}$ and $B \in \mathcal{C}$.
- All complexity classes we mentioned earlier are closed under $\leq_m^L$.
- P, NP, PSPACE, EXP are closed under $\leq_m^p$.

Completeness

- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems.$\mathcal{C}$ is closed under $\leq$ if $A \leq B$ and $B \in \mathcal{C}$ implies $A \in \mathcal{C}$.
- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems closed under $\leq$
  - Problem B is hard for class $\mathcal{C}$ under $\leq$ if, for each $A \in \mathcal{C}$, $A \leq B$.
  - Problem B is complete for class $\mathcal{C}$ under $\leq$ if, B is hard for $\mathcal{C}$ and $B \in \mathcal{C}$.
- All complexity classes we mentioned earlier are closed under $\leq_m^L$.
- P, NP, PSPACE, EXP are closed under $\leq_m^p$.
- A P-complete problem is a problem complete for P under $\leq_m^L$.

Completeness

- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems.$\mathcal{C}$ is closed under $\leq$ if $A \leq B$ and $B \in \mathcal{C}$ implies $A \in \mathcal{C}$.
- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems closed under $\leq$
  - Problem B is hard for class $\mathcal{C}$ under $\leq$ if, for each $A \in \mathcal{C}$, $A \leq B$.
  - Problem B is complete for class $\mathcal{C}$ under $\leq$ if, B is hard for $\mathcal{C}$ and $B \in \mathcal{C}$.
- All complexity classes we mentioned earlier are closed under $\leq_m^L$.
- P, NP, PSPACE, EXP are closed under $\leq_m^p$.
- A P-complete problem is a problem complete for P under $\leq_m^L$.
- A NP-complete problem is a problem complete for NP under $\leq_m^p$.

Completeness

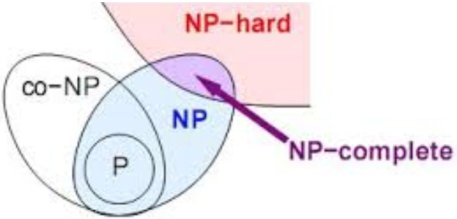
- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems.$\mathcal{C}$ is closed under $\leq$ if $A \leq B$ and $B \in \mathcal{C}$ implies $A \in \mathcal{C}$.
- Let $\leq$ be a reducibility among problems and $\mathcal{C}$ a class of problems closed under $\leq$
  - Problem B is hard for class $\mathcal{C}$ under $\leq$ if, for each $A \in \mathcal{C}$, $A \leq B$.
  - Problem B is complete for class $\mathcal{C}$ under $\leq$ if, B is hard for $\mathcal{C}$ and $B \in \mathcal{C}$.
- All complexity classes we mentioned earlier are closed under $\leq_m^L$.
- P, NP, PSPACE, EXP are closed under $\leq_m^p$.
- A P-complete problem is a problem complete for P under $\leq_m^L$.
- A NP-complete problem is a problem complete for NP under $\leq_m^p$.
- $\leq_m^p$ and $\leq_m^L$ are transitive relations.

NP-completeness

A problem A is NP-complete if:

1. $A \in$ NP, and
2. for every $B \in$ NP, $B \leq_m^p A$.

If for each $B \in$ NP, $B \leq_m^p A$ but $A \notin$ NP then A is said to be NP-hard.



Lemma

If A is NP-complete, then $A \in$P iff P=NP.

Lemma

If A is NP-complete, then $A \in$P iff P=NP.

- Once we prove that a problem is NP-complete, either A has no efficient algorithm or all NP problems are in P.

Lemma

If A is NP-complete, then $A \in$P iff P=NP.

- Once we prove that a problem is NP-complete, either A has no efficient algorithm or all NP problems are in P.
- P $=$ NP? is one of the Millennium Problems Clay Mathematics Institute

Lemma

If A is NP-complete, then $A \in$P iff P=NP.

- Once we prove that a problem is NP-complete, either A has no efficient algorithm or all NP problems are in P.
- P $=$ NP? is one of the Millennium Problems Clay Mathematics Institute
- *Majority conjecture:* P $\neq$ NP

Lemma

If A is NP-complete, then $A \in$P iff P=NP.

- Once we prove that a problem is NP-complete, either A has no efficient algorithm or all NP problems are in P.
- P $=$ NP? is one of the Millennium Problems Clay Mathematics Institute
- *Majority conjecture:* P $\neq$ NP
- To prove a problem is NP-complete, we just have to find a reduction from a problem known to be NP-complete.

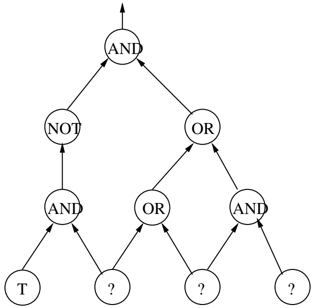
Lemma

If A is NP-complete, then $A \in$P iff P=NP.

- Once we prove that a problem is NP-complete, either A has no efficient algorithm or all NP problems are in P.
- P = NP? is one of the Millennium Problems Clay Mathematics Institute
- *Majority conjecture:* P $\neq$ NP
- To prove a problem is NP-complete, we just have to find a reduction from a problem known to be NP-complete.
- We need as a seed a first NP-complete problem.

CIRCUIT SAT.

CIRCUIT SAT: Given a Boolean circuit with gates *AND*, *OR*, *NOT*, and the input gates *T*, *F* and *?*, and one output gate. Is there an an assignment to the input gates (?), such that the circuit evaluates to T?



For example if the input to ? is T,F,T, the output is F
if the input is F,T,T, the output is T

Any NP problem can be ”expressed” as CIRCUIT SAT.

- We want to show that if $A \in \mathrm{NP}$, then $A \leq_m^p$ CIRCUIT SAT.

Any NP problem can be ”expressed” as CIRCUIT SAT.

- We want to show that if $A \in$ NP, then $A \leq_m^p$ CIRCUIT SAT.
- A is a decision NP problem $\Rightarrow$ there is a polynomial-time algorithm $\mathcal{A}$ which given an instance x and a witness solution c of A, checks in polynomial time (in the length of $|x|$) if c is a valid certificate for x.

Any NP problem can be "expressed" as CIRCUIT SAT.

- We want to show that if $A \in \mathrm{NP}$, then $A \leq_m^p$ CIRCUIT SAT.
- A is a decision NP problem $\Rightarrow$ there is a polynomial-time algorithm $\mathcal{A}$ which given an instance x and a witness solution c of A, checks in polynomial time (in the length of $|x|$) if c is a valid certificate for x.
- Any polynomial-time algorithm (TM) can be expressed as a polynomial-size circuit, whose input gates encode the input to the algorithm, and the ? input gates are feeding the witness c. It mimics the tape (memory) evolution. If the algorithm solves a decision problem (Y/N), the output of the circuit will be 1/0.

Any NP problem can be "expressed" as CIRCUIT SAT.

- We want to show that if $A \in \mathrm{NP}$, then $A \leq_m^p$ CIRCUIT SAT.
- A is a decision NP problem $\Rightarrow$ there is a polynomial-time algorithm $\mathcal{A}$ which given an instance x and a witness solution c of A, checks in polynomial time (in the length of $|x|$) if c is a valid certificate for x.
- Any polynomial-time algorithm (TM) can be expressed as a polynomial-size circuit, whose input gates encode the input to the algorithm, and the ? input gates are feeding the witness c. It mimics the tape (memory) evolution. If the algorithm solves a decision problem (Y/N), the output of the circuit will be 1/0.
- There is a way to feed c and get output 1 iff there is a valid certificate.

The seminal theorem: Cook-Levin's Theorem

Therefore, given any instance x for A, we can construct in poly-time instance C of CIRCUIT SAT whose known inputs are the bits of x, and whose unknown inputs are the bits of c, and such that the output of C is 1 iff $\mathcal{A}$ outputs YES on input (x, c).

The seminal theorem: Cook-Levin's Theorem

Therefore, given any instance x for A, we can construct in poly-time instance C of CIRCUIT SAT whose known inputs are the bits of x, and whose unknown inputs are the bits of c, and such that the output of C is 1 iff $\mathcal{A}$ outputs YES on input (x, c).

Theorem (Cook-Levin's theorem)

CIRCUIT-SAT is NP-complete.

Boolean formulas

- A Boolean variable x can take values 0,1.

Boolean formulas

- A Boolean variable x can take values 0,1.
- A Boolean formula is an expression constructed from Boolean variables and connectives, negation ($\neg\phi$ or $\overline{\phi}$), disjunction ($\vee$) and conjunction ($\wedge$).

Boolean formulas

- A Boolean variable x can take values 0,1.
- A Boolean formula is an expression constructed from Boolean variables and connectives, negation ($\neg\phi$ or $\overline{\phi}$), disjunction ($\vee$) and conjunction ($\wedge$).
- A Boolean formula ϕ is satisfiable if there is a truth assignment $T : X \to \{0, 1\}$ to the variables in ϕ such that $T(\phi) = 1$.

Boolean formulas

- A Boolean variable x can take values 0,1.
- A Boolean formula is an expression constructed from Boolean variables and connectives, negation ($\neg\phi$ or $\overline{\phi}$), disjunction ($\vee$) and conjunction ($\wedge$).
- A Boolean formula ϕ is satisfiable if there is a truth assignment $T : X \to \{0,1\}$ to the variables in ϕ such that $T(\phi) = 1$.
- For example, for $X = \{x_1, x_2, x_3\}$,

$$\phi = (x_1 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_2 \vee x_3)$$

is satisfiable, take $T(x_1) = T(x_2) = 1$, $T(x_3) = 0$ then $T(\phi) = 0$.

Boolean formulas: normal forms

Boolean formulas: normal forms

- A literal is a Boolean variable x or a negation of a Boolean variable$\bar{x}$.

Boolean formulas: normal forms

- A literal is a Boolean variable x or a negation of a Boolean variable$\bar{x}$.
- A clause is a disjunction (conjunction) of literals.

Boolean formulas: normal forms

- A literal is a Boolean variable x or a negation of a Boolean variable$\bar{x}$.
- A clause is a disjunction (conjunction) of literals.
- A Boolean formula ϕ in Conjunctive Normal Form (CNF) is a conjunction of clauses, $\phi = \bigwedge_{i=1}^{m}(C_i)$, where each clause $C_i = \bigvee_{j=1}^{k_i}\{l_j\}$.

  For example, for $X = \{x_1, x_2, x_3\}$, a CNF formula is

$$\phi = (x_1 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_2 \vee x_3)$$

Boolean formulas: normal forms

- A literal is a Boolean variable x or a negation of a Boolean variable$\bar{x}$.
- A clause is a disjunction (conjunction) of literals.
- A Boolean formula ϕ in Conjunctive Normal Form (CNF) is a conjunction of clauses, $\phi = \bigwedge_{i=1}^{m}(C_i)$, where each clause $C_i = \bigvee_{j=1}^{k_i}\{l_j\}$.

  For example, for $X = \{x_1, x_2, x_3\}$, a CNF formula is

$$\phi = (x_1 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_2 \vee x_3)$$

- A Boolean formula ϕ in Disjunctive Normal Form (DNF) is expressed as a disjunction of clauses, $\phi = \bigvee_{i=1}^{m}(C_i)$, where each clause $C_i = \bigwedge_{j=1}^{k_i}\{l_j\}$.

SAT problem and variations

Given a formula CNF formula ϕ on a set X of n variables,

- SAT: Is ϕ satisfiable?

SAT problem and variations

Given a formula CNF formula ϕ on a set X of n variables,

- SAT: Is ϕ satisfiable?
- k-SAT. Given a boolean formula in CNF $\phi = \bigwedge_{i=1}^{m}(C_i)$ in where each clause has exactly k literals, is ϕ satisfiable?
  Ex. 3-SAT
  $\phi = (x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \bar{x_2} \vee x_4) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (x_1 \vee \bar{x_3} \vee \bar{x_4}) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3}) \wedge (\bar{x_2} \vee x_3 \vee x_4) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3}) \wedge (\bar{x_2} \vee x_3 \vee \bar{x_4}) \wedge (x_2 \vee \bar{x_3} \vee \bar{x_4}) \wedge (\bar{x_2} \vee \bar{x_3} \vee \bar{x_4})$
  $T(x_1) = 1$, $T(x_2) = T(x_3) = T(x_4) = 0$, satisfies the previous instance.

SAT problem and variations

Given a formula CNF formula ϕ on a set X of n variables,

- SAT: Is ϕ satisfiable?
- k-SAT. Given a boolean formula in CNF $\phi = \bigwedge_{i=1}^{m}(C_i)$ in where each clause has exactly k literals, is ϕ satisfiable?
  Ex. 3-SAT
  $\phi = (x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \bar{x}_2 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_3 \vee x_4) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_3 \vee \bar{x}_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4)$
  $T(x_1) = 1$, $T(x_2) = T(x_3) = T(x_4) = 0$, satisfies the previous instance.

Given a DNF formula ϕ on a set X of n variables,

- DNF-SAT: Is ϕ satisfiable?

SAT problem and variations

Given a formula CNF formula ϕ on a set X of n variables,

- SAT: Is ϕ satisfiable?
- k-SAT. Given a boolean formula in CNF $\phi = \bigwedge_{i=1}^{m}(C_i)$ in where each clause has exactly k literals, is ϕ satisfiable?
  Ex. 3-SAT
  $\phi = (x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \bar{x}_2 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_3 \vee x_4) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_3 \vee \bar{x}_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4)$
  $T(x_1) = 1$, $T(x_2) = T(x_3) = T(x_4) = 0$, satisfies the previous instance.

Given a DNF formula ϕ on a set X of n variables,

- DNF-SAT: Is ϕ satisfiable?
- k-DNF-SAT: Given a boolean formula in CNF $\phi = \bigwedge_{i=1}^{m}(C_i)$ in where each clause has exactly k literals, is ϕ satisfiable?

SAT problems

SAT problems

Theorem

SAT is NP-complete.

SAT problems

Theorem

SAT is NP-complete.
k-SAT is NP-complete for $k \geq 3$ and in P for $k \leq 2$.

SAT problems

SAT problems

SAT $\in$ NP

- Given a CNF Boolean formula F on n variables, having n, $C_1, \ldots, C_m$ clauses, F is satisfiable iff there is a truth assignment that satisfies F.

SAT problems

SAT $\in$ NP

- Given a CNF Boolean formula F on n variables, having n, $C_1, \ldots, C_m$ clauses, F is satisfiable iff there is a truth assignment that satisfies F.
- Formula size: the sum of the lengths of its clauses. $|F| = \sum_{i=1}^{m} |C_i|$.

SAT problems

SAT $\in$ NP

- Given a CNF Boolean formula F on n variables, having n, $C_1, \dots, C_m$ clauses, F is satisfiable iff there is a truth assignment that satisfies F.
- Formula size: the sum of the lengths of its clauses. $|F| = \sum_{i=1}^{m} |C_i|$. If F is a k-SAT formula $|F| \leq km$

SAT problems

SAT $\in$ NP

- Given a CNF Boolean formula F on n variables, having n, $C_1, \dots, C_m$ clauses, F is satisfiable iff there is a truth assignment that satisfies F.
- Formula size: the sum of the lengths of its clauses. $|F| = \sum_{i=1}^{m} |C_i|$. If F is a k-SAT formula $|F| \leq km$
- A truth assignment can be modeled by a a string $\sigma \in \{0,1\}^n$ which has size n.

SAT problems

SAT $\in$ NP

- Given a CNF Boolean formula F on n variables, having n, $C_1, \dots, C_m$ clauses, F is satisfiable iff there is a truth assignment that satisfies F.
- Formula size: the sum of the lengths of its clauses. $|F| = \sum_{i=1}^{m} |C_i|$. If F is a k-SAT formula $|F| \leq km$
- A truth assignment can be modeled by a a string $\sigma \in \{0,1\}^n$ which has size n.
- Given F and $\sigma \in \{0,1\}^n$, σ satisfies F iff every clause has a literal that evaluates to 1. This property can be checked in time $O(n + |F|)$.

SAT problems

SAT $\in$ NP

- Given a CNF Boolean formula F on n variables, having n, $C_1, \ldots, C_m$ clauses, F is satisfiable iff there is a truth assignment that satisfies F.
- Formula size: the sum of the lengths of its clauses. $|F| = \sum_{i=1}^{m} |C_i|$. If F is a k-SAT formula $|F| \leq km$
- A truth assignment can be modeled by a a string $\sigma \in \{0,1\}^n$ which has size n.
- Given F and $\sigma \in \{0,1\}^n$, σ satisfies F iff every clause has a literal that evaluates to 1. This property can be checked in time $O(n + |F|)$.
- $SAT = \{F \mid \exists \sigma \in \{0,1\}^n F(\sigma) = 1\}$

SAT problems

SAT is NP-hard

SAT problems

SAT is NP-hard

- Let us construct a reduction from CIRCUIT SAT. Let C be a Boolean circuit on n inputs. C can be described as a labeled directed acyclic graph, (V, E), one vertex per gate and one per input and another for the output.

SAT problems

SAT is NP-hard

- Let us construct a reduction from CIRCUIT SAT. Let C be a Boolean circuit on n inputs. C can be described as a labeled directed acyclic graph, (V, E), one vertex per gate and one per input and another for the output.
- The formula F has one variable for each edge in E.

SAT problems

SAT is NP-hard

- Let us construct a reduction from CIRCUIT SAT. Let C be a Boolean circuit on n inputs. C can be described as a labeled directed acyclic graph, (V, E), one vertex per gate and one per input and another for the output.
- The formula F has one variable for each edge in E.
- Use $A \leftrightarrow B \equiv (\neg A \vee B) \wedge (A \vee \neg B)$ to express the gate equivalence in clause form.

SAT problems

SAT is NP-hard

- Let us construct a reduction from CIRCUIT SAT. Let C be a Boolean circuit on n inputs. C can be described as a labeled directed acyclic graph, (V, E), one vertex per gate and one per input and another for the output.
- The formula F has one variable for each edge in E.
- Use $A \leftrightarrow B \equiv (\neg A \vee B) \wedge (A \vee \neg B)$ to express the gate equivalence in clause form.
- For an and gate, $z \leftrightarrow x \wedge y$, we have $(\neg(x \wedge y) \vee z), ((x \wedge y) \vee \neg z)$ giving $(\neg x \vee \neg y \vee z), (x \vee \neg z), (x \vee \neg z)$.

SAT problems

SAT is NP-hard

- Let us construct a reduction from CIRCUIT SAT. Let C be a Boolean circuit on n inputs. C can be described as a labeled directed acyclic graph, (V, E), one vertex per gate and one per input and another for the output.
- The formula F has one variable for each edge in E.
- Use $A \leftrightarrow B \equiv (\neg A \vee B) \wedge (A \vee \neg B)$ to express the gate equivalence in clause form.
- For an and gate, $z \leftrightarrow x \wedge y$, we have $(\neg(x \wedge y) \vee z), ((x \wedge y) \vee \neg z)$ giving $(\neg x \vee \neg y \vee z), (x \vee \neg z), (x \vee \neg z)$.
- By construction, C is SAT iff F is SAT. Besides, each gate is replaced by a constant number of clauses, so the construction requires polynomial time.

SAT problems

k-SAT, $k > 2$

- Membership in NP follows from the general result for SAT.

SAT problems

k-SAT, $k > 2$

- Membership in NP follows from the general result for SAT.
- In the previous reduction some clauses have 2 literals.

SAT problems

k-SAT, $k > 2$

- Membership in NP follows from the general result for SAT.
- In the previous reduction some clauses have 2 literals.
- Let C with ℓ literals and z a new variable, $C \leftrightarrow (C \vee z) \wedge (C \vee \neg z)$.

SAT problems

k-SAT, $k > 2$

- Membership in NP follows from the general result for SAT.
- In the previous reduction some clauses have 2 literals.
- Let C with ℓ literals and z a new variable, $C \leftrightarrow (C \vee z) \wedge (C \vee \neg z)$.
- Using the reduction from CIRCUIT SAT to SAT and this construction we can obtain a 3-SAT formula with the same properties.
  In total, we add no more than m variables and m clauses.

SAT problems

k-SAT, $k > 2$

- Membership in NP follows from the general result for SAT.
- In the previous reduction some clauses have 2 literals.
- Let C with ℓ literals and z a new variable, $C \leftrightarrow (C \vee z) \wedge (C \vee \neg z)$.
- Using the reduction from CIRCUIT SAT to SAT and this construction we can obtain a 3-SAT formula with the same properties.
  In total, we add no more than m variables and m clauses.
- The last trick allow us to reduce k-SAT to $k+1$-SAT in polynomial time.

Graph problems

Clique: INPUT: $G(V, E)$ and $k \in \mathbb{Z}$.
QUESTION: Decide if there is a complete subgraph of G with at least k vertices.

Graph problems

Clique: INPUT: $G(V,E)$ and $k \in \mathbb{Z}$.
QUESTION: Decide if there is a complete subgraph of G with at least k vertices.
Independent set (IS): INPUT: $G(V,E)$ and $k \in \mathbb{Z}$.
QUESTION: Decide if there is $S \subseteq G$ with $|S| \geq k$, such that $G[S]$ has no edges.

Graph problems

Clique: INPUT: $G(V,E)$ and $k \in \mathbb{Z}$.
QUESTION: Decide if there is a complete subgraph of G with at least k vertices.
Independent set (IS): INPUT: $G(V,E)$ and $k \in \mathbb{Z}$.
QUESTION: Decide if there is $S \subseteq G$ with $|S| \geq k$, such that $G[S]$ has no edges.
Vertex cover (VC): INPUT: $G(V,E)$ and $g \in \mathbb{Z}$.
QUESTION: Decide if there is $S \subseteq G$ with $|S| \geq g$, such that each edge in E has at least one of its vertices in S.

Graph problems

Clique: INPUT: $G(V, E)$ and $k \in \mathbb{Z}$.
QUESTION: Decide if there is a complete subgraph of G with at least k vertices.
Independent set (IS): INPUT: $G(V, E)$ and $k \in \mathbb{Z}$.
QUESTION: Decide if there is $S \subseteq G$ with $|S| \geq k$, such that $G[S]$ has no edges.
Vertex cover (VC): INPUT: $G(V, E)$ and $g \in \mathbb{Z}$.
QUESTION: Decide if there is $S \subseteq G$ with $|S| \geq g$, such that each edge in E has at least one of its vertices in S.

Theorem

Clique, Independent set and Vertex cover are NP-complete.

Problems on numbers

Subset Sum: INPUT: sequence of positive integers $S = \{a_1, \ldots, a_n\}$, let $I = \{1, \ldots, n\}$, and $t \in \mathbb{Z}$.
QUESTION: Decide if it exists a $I' \subseteq \{1, \ldots, n\}$ s.t. $\sum_{i \in I'} a_i = t$.

Problems on numbers

Subset Sum: INPUT: sequence of positive integers $S = \{a_1, \dots, a_n\}$, let $I = \{1, \dots, n\}$, and $t \in \mathbb{Z}$.
QUESTION: Decide if it exists a $I' \subseteq \{1, \dots, n\}$ s.t. $\sum_{i \in I'} a_i = t$.

Theorem

Subset sum is NP-complete.