Fundamental Computer Science
Lecture 4: Complexity
NP-completeness

Denis Trystram
MoSIG1 and M1Info – University Grenoble-Alpes

March, 2021
Definition of time complexity classes
  - $\mathcal{P}$: problems solvable in $O(n^k)$ time
  - $\mathcal{NP}$: problems verifiable in $O(n^k)$ time
  - space complexity

Prove that a problem belongs to $\mathcal{NP}$
  - provide a polynomial-time verifier

Reduction from problem $A$ to problem $B$ ($A \leq_P B$)
1. transform an instance $I_A$ of $A$ to an instance $I_B$ of $B$
2. show that the reduction is of polynomial size
3. prove that:
   "there is a solution for the problem $A$ on the instance $I_A$
   if and only if
   there is a solution for the problem $B$ on the instance $I_B"
Agenda

- Definition of the class NP-complete
- The SAT problem
- Cook-Levin theorem
- Use reductions to prove NP-completeness
- A detailed example: Vertex Cover
- Variants of SAT
Let $C$ be a set of languages.

**Definition**

A language $B$ is $C$-complete if

- $B \in C$, and
- every language $A$ in $C$ is polynomially reducible to $B$. 
**NP-completeness**

**Definition**

A language $B$ is **NP-complete** if

- $B \in \mathcal{NP}$, and
- every language $A$ in $\mathcal{NP}$ is polynomially reducible to $B$. 

**Theorem**

If $B$ is NP-complete and $B \in \mathcal{P}$, then $\mathcal{P} = \mathcal{NP}$

**Proof:**

Direct from the definition of reducibility
**NP-completeness**

**Definition**
A language $B$ is **NP-complete** if
- $B \in \mathcal{NP}$, and
- every language $A$ in $\mathcal{NP}$ is polynomially reducible to $B$.

**Theorem**

*If $B$ is NP-complete and $B \in \mathcal{P}$, then $\mathcal{P} = \mathcal{NP}$*

**Proof:**
- Direct from the definition of reducibility
NP-COMPLETENESS

Definition

A language $B$ is $\text{NP-complete}$ if

$B \in \mathcal{NP}$, and

every language $A \in \mathcal{NP}$ is polynomially reducible to $B$.

Theorem

If $B$ is $\text{NP-complete}$ and $B \leq_p C$ for $C \in \mathcal{NP}$, then $C$ is $\text{NP-complete}$

Proof:

- initially, $C \in \mathcal{NP}$
- we need to show: every $A \in \mathcal{NP}$ polynomially reduces to $C$
  - every language $\in \mathcal{NP}$ polynomially reduces to $B$
  - $B$ polynomially reduces to $C$
The next step

Prove that there are some problems in \texttt{NP-complete}

Stephen Cook proved in 1971 that \texttt{SAT} \in \texttt{NP-complete}
Recall on Logic: Boolean formulas

- $x_i$: a Boolean variable, values TRUE or FALSE
- $\overline{x}_i$: negation of $x_i$ – logical NOT
- $x_i, \overline{x}_i$: literals
Recall on Logic: Boolean formulas

- $x_i$: a Boolean variable, values TRUE or FALSE
- $\bar{x}_i$: negation of $x_i$ – logical NOT
- $x_i, \bar{x}_i$: literals
- $\lor$: logical OR
- $\land$: logical AND
Recall on Logic: Boolean formulas

- $x_i$: a Boolean variable, values TRUE or FALSE
- $\overline{x_i}$: negation of $x_i$ – logical NOT
- $x_i, \overline{x_i}$: literals
- $\lor$: logical OR
- $\land$: logical AND
- $(x_1 \lor \overline{x_3} \lor x_4)$: clause, a set of literals in disjunction

Every formula can be written in CNF, thus, focus on CNF formulas.
Recall on Logic: Boolean formulas

- $x_i$: a Boolean variable, values TRUE or FALSE
- $\overline{x}_i$: negation of $x_i$ – logical NOT
- $x_i, \overline{x}_i$: literals
- $\lor$: logical OR
- $\land$: logical AND
- $(x_1 \lor \overline{x}_3 \lor x_4)$: clause, a set of literals in disjunction
- $\mathcal{F} = (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_4) \land (x_1 \lor x_4)$: a Boolean formula in Conjunctive Normal Form (CNF), a set of clauses in conjunction
  - every formula can be written in CNF (thus, focus on CNF formulas)
Recall on Logic: Boolean formulas

- \( x_i \): a Boolean variable, values TRUE or FALSE
- \( \bar{x}_i \): negation of \( x_i \) – logical NOT
- \( x_i, \bar{x}_i \): literals
- \( \lor \): logical OR
- \( \land \): logical AND
- \( (x_1 \lor \bar{x}_3 \lor x_4) \): clause, a set of literals in disjunction
- \( \mathcal{F} = (x_1 \lor x_2 \lor \bar{x}_3) \land (\bar{x}_4) \land (x_1 \lor x_4) \): a Boolean formula in Conjunctive Normal Form (CNF), a set of clauses in conjunction
  - every formula can be written in CNF (thus, focus on CNF formulas)
- assignment: give TRUE or FALSE value to variables
Recall on Logic: Boolean formulas

- \( x_i \): a Boolean variable, values TRUE or FALSE
- \( \bar{x}_i \): negation of \( x_i \) – logical NOT
- \( x_i, \bar{x}_i \): literals
- \( \lor \): logical OR
- \( \land \): logical AND
- \( (x_1 \lor \bar{x}_3 \lor x_4) \): clause, a set of literals in disjunction
- \( F = (x_1 \lor x_2 \lor \bar{x}_3) \land (\bar{x}_4) \land (x_1 \lor x_4) \): a Boolean formula in Conjunctive Normal Form (CNF), a set of clauses in conjunction
  - every formula can be written in CNF (thus, focus on CNF formulas)
- assignment: give TRUE or FALSE value to variables
- a formula is satisfiable if there is an assignment evaluating to TRUE
  - i.e, \((x_1, x_2, x_3, x_4) = (T, T, T, F)\) for the above formula \( F \)
The satisfiability problem SAT

- $X = \{x_1, x_2, \ldots, x_n\}$: set of variables
- $C = \{C_1, C_2, \ldots, C_m\}$: set of clauses
- $F = C_1 \land C_2 \land \ldots \land C_m$

$\text{SAT} = \{\langle F \rangle \mid F \text{ is a satisfiable Boolean formula} \}$

The problem version of SAT:

- SAT
- **Instance.** $m$ clauses $C_i$ expressed using $n$ literals
- **Question.** Is the formula $F = C_1 \land C_2 \land \ldots \land C_m$ satisfiable?
Example: Vertex Cover

We will show in a separate lesson that $\text{VC} \in \text{NP-complete}$