Complexity IBC028, Lecture 4

H. Geuvers

Institute for Computing and Information Sciences
Radboud University Nijmegen

Version: spring 2021

Outline

Decision Problems

P and **NP**

NP-hard and **NP**-complete

Many algorithmic problems are decision problems

- A **Decision Problem** is the question whether some input i satisfies a specific property $Q(i)$. Its solution is a yes/no answer.
- Some examples:
  - Given a number n, is n prime?
  - Given a graph G, is there a Hamiltonian cycle in G?
  - Given a graph G, is there an Euler tour in G?
  - Given a graph G and two points p and q in G, are p and q connected?
- We can associate a decision problem Q with a **language** $L_Q \subseteq \{0,1\}^*$

$w \in L_Q \Leftrightarrow w$ is an encoding of a problem for which Q holds.

Encodings of decision problems

- The precise encoding is left implicit.
- We have the usual operations on languages: union, intersection, complement, concatenation, Kleene-star.

$$\begin{aligned}
\text{Ham} &\subseteq \{0,1\}^* \\
&:= \text{collection of strings } w \text{ that encode a graph } G \\
&\quad \text{ that has a Hamiltonian cycle} \\
\text{Path} &\subseteq \{0,1\}^* \\
&:= \text{collection of strings } w \text{ that encode } \langle G, p, q, n\rangle, \\
&\quad \text{where } G \text{ is a graph, } p, q \in G, \text{ such that} \\
&\quad \text{there is a path from } p \text{ to } q \text{ in } G \text{ with at most } n \text{ edges}
\end{aligned}$$

Polynomial Decision Problems

Definition

- An algorithm A is **polynomial** if we have for its time complexity T that $T(n) = \mathcal{O}(n^k)$ for some k.
- The algorithm $A : \{0,1\}^* \to \{0,1\}$ **decides** $L \subseteq \{0,1\}^*$ if $w \in L \iff A(w) = 1$.
- A decision problem Q is **polynomial** if we have a polynomial algorithm that decides L_Q.

What encoding?
Encodings $e_1, e_2 : I \to \{0,1\}^*$ are **polynomially related** if there are polynomial functions f and g such that $f(e_1(i)) = e_2(i)$ and $g(e_2(i)) = e_1(i)$ for all $i \in I$.
If e_1, e_2 are polynomially related, then, for a problem $Q \subseteq I$:

$$e_1(Q) \text{ is polynomial if and only if } e_2(Q) \text{ is polynomial.}$$

Closure operations for Polynomial Decision Problems

Lemma

Polynomial decision problems are closed under complement, intersection, union, concatenation

Proof

- If A decides $L \subseteq \{0,1\}^*$ in polynomial time, then $B(w) := 1 - A(w)$ decides $\overline{L}$ in polynomial time.
- If A_i decides L_i in polynomial time, then $B(w) := \mathrm{sg}(A_1(w) + A_2(w))$ decides $L_1 \cup L_2$ in polynomial time.
- If A_i decides L_i in polynomial time, then

$$B(w) := \mathrm{sg}(\Sigma_{w=uv} A_1(u) \cdot A_2(v))$$

decides $L_1 L_2$ in polynomial time.

The class **P**

Definition

$$\mathbf{P} := \{L \subseteq \{0,1\}^* \mid \quad \exists A, A \text{ polynomial}, A \text{ decides } L\}$$

- Path $\in$ **P**, EulerTour $\in$ **P**,
- Ham $\notin$ **P** (...everyone thinks)

For Ham, no polynomial algorithm is known, but there is a notion of **certificate** that can be checked in polynomial time.

$$w \in \text{Ham} \quad \Leftrightarrow \quad w \text{ encodes a graph } G \quad \wedge$$
$$\exists y (y \text{ encodes a Hamiltonian cycle in } G).$$

Non-deterministic Polynomial Decision Problems

Definition

- The algorithm A **verifies** $L \subseteq \{0,1\}^*$ if $A : \{0,1\}^* \to \{0,1\}$ and
$$w \in L \iff \exists y \in \{0,1\}^*(A(w,y) = 1).$$
- $L \subseteq \{0,1\}^*$ is **non-deterministic polynomial** (NP) if there is a polynomial algorithm A that verifies L with polynomial certificates, that is
$$w \in L \iff \exists y \in \{0,1\}^*(|y| \text{ polynomial in } |w| \wedge A(w,y) = 1).$$

- Ham is non-deterministic polynomial.
- NonPrime (determining whether a number is not prime) is non-deterministic polynomial.

P and NP

$\mathbf{P} :=$
$\{L \subseteq \{0,1\}^* \mid \ \exists A, A \text{ polynomial}, \ w \in L \Longleftrightarrow A(w) = 1\}$

$\mathbf{NP} :=$
$\{L \subseteq \{0,1\}^* \mid \ \exists A, A \text{ polynomial},$
$w \in L \Longleftrightarrow \exists y \in \{0,1\}^*(|y| \text{ polynomial in } |w| \ \wedge \ A(w,y) = 1)\}$

- **P** = the class of polynomial time decision problems.
- **NP** = the class of non-deterministic polynomial time decision problems.
- First property: $\mathbf{P} \subseteq \mathbf{NP}$.

What is the non-determinism in **NP**?

Polynomial algorithm for L	$=$	a deterministic Turing Machine M that halts on every input w in a number of steps polynomial in $\|w\|$ such that $w \in L$ iff $M(w)$ halts in q_f.
Non-deterministic polynomial algorithm for L	$=$	a non-deterministic Turing Machine M that halts on every input w in a number of steps polynomial in $\|w\|$ such that $w \in L$ iff $M(w)$ has a computation that halts in q_f.

A non-deterministic TM can be turned into a deterministic TM by making choices. The “certificate” is the succesful choice from the list of possible choices.

Polynomial Reducibility

Definition

L_1 (polynomially) reduces to L_2, notation $L_1 \leq_P L_2$ if there is a polynomial function $f : \{0,1\}^* \to \{0,1\}^*$ such that

$$x \in L_1 \iff f(x) \in L_2$$

Lemma

- $\leq_P$ is transitive: if $L_1 \leq_P L_2$ and $L_2 \leq_P L_3$ then $L_1 \leq_P L_3$.
- If $L_1 \leq_P L_2$ and $L_2 \in \mathbf{P}$, then $L_1 \in \mathbf{P}$.

NP-hard and **NP**-complete

Definition

- L is called **NP-hard** if

$$\forall L' \in \mathbf{NP}(L' \leq_P L).$$

  That is: all **NP**-problems can be reduced to L.
- $\mathbf{NPH} := \{L \mid L \text{ is } \mathbf{NP}\text{-hard}\}$.
- L is called **NP-complete** if $L \in \mathbf{NP}$ and L is **NP**-hard.
- $\mathbf{NPC} := \mathbf{NP} \cap \mathbf{NPH}$.

Theorem

If $L' \leq_P L$ and $L' \in \mathbf{NPH}$, then $L \in \mathbf{NPH}$.

Proof: Let $L'' \in \mathbf{NP}$. Then $L'' \leq_P L' \leq_P L$, so $L'' \leq_P L$. □

NP-hard and **NP**-complete problems

How to prove that L is **NP**-complete?

- First prove that $L \in$ **NP**: give a polynomial algorithm and a certificate for each input.
- Pick a well-known $L' \in$ **NPH** and show that $L' \leq_P L$.

There are very many known **NP**-hard problems.

- SAT $\in$ **NPH** (Cook-Levin, 1970), to be discussed further in the next lecture. In the final lecture we will prove that SAT $\in$ **NPH**.
- Ham $\in$ **NPH** and so is "traveling salesman problem" (TSP)
- "Clique" and "vertex cover" are graph-problems in **NPH**.

$$\mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXPTIME} \subseteq \mathbf{EXPSPACE}$$

All these inclusions are known; for none of them it is known if they are strict inclusions.

Satisfiability

DEFINITION

The boolean formulas are built from

- Atoms, $p, q, r, \ldots$
- Boolean connectives $\wedge$, $\vee$, $\neg$.

A formula is **satisfiable** if we can assign values (from $\{0, 1\}$) to the atoms such that the formula is true.
SAT is the problem of deciding if a boolean formula is satisfiable.

SAT was the first problem shown to be **NP**-complete.
A (seemingly simpler) variant of SAT is already **NP**-complete:
CNF-SAT: satisfiabilty of **conjunctive normal forms** (CNF):

- A CNF is a **conjunction of clauses**
- A clause is a **disjunction of literals**
- a literal is an atom or a negated atom.

NP and co-NP

Definition

co-NP $:= \{L \mid \overline{L} \in \mathbf{NP}\}$. ($\overline{L}$ is the complement of L.)

- Some well-known problems are in **co-NP**, for example Prime, which tests if a number n is a prime number.
- Prime $\in$ **NP** $\cap$ **co-NP**, and it is unknown if Prime $\in$ **P** (but not to be expected).
- The precise relations between **P**, **NP** and **co-NP** are a major open question in Computer Science, most notably:

$$\mathbf{P} \stackrel{??}{=} \mathbf{NP}.$$