Intro to Theory of Computation

CS 464

LECTURE 24

Last time

- Relationship between models: deterministic/nondeterministic
- Class **P**

Today

- Class **NP**

Homework 9 due
Homework 10 out

Sofya Raskhodnikova

I-clicker question (frequency: AC)

Consider the following algorithm A for PRIMES.

Given b, try to divide b by 2,3,…,$\sqrt{b}$.

If one of them divides b, accept; o.w. reject.

If n = input length, # of divisions A performs is

A. $\Theta(\sqrt{n})$

B. $\Theta(n)$

C. $2^{\Theta(n)}$

D. $2^{\Theta(\sqrt{n})}$

E. None of the above.

Central ideas

- Poly-time as "feasible"
  - most natural problems either are easy (say in TIME(n^3)) or have no known poly-time algorithms

- P =languages that can be decided in poly-time
- NP = languages for which the membership in the language is easy to verify given a hint
- EXP = languages that can be decided in exponential time

- Poly-time Reductions: X is no harder than Y for poly-time TMs

The class P

P is the class of languages decidable in polynomial time on a ***deterministic*** 1-tape TM:

$$\boldsymbol{P} = \bigcup_{k} TIME(n^k).$$

Examples of languages in P

Problem	Description	Algorithm	Yes	No
MULTIPLE	Is x a multiple of y?	Grade school division	`51, 17`	`51, 16`
RELPRIME	Are x and y relatively prime?	Euclid (300 BCE)	`34, 39`	`34, 51`
PRIMES	Is x prime?	AKS (2002)	`53`	`51`
all CFLs (e.g. the language of balanced parentheses and brackets)	Is the string in the given CFL? (e.g., is the string of parentheses and brackets balanced?)	Dynamic programming	Depends on the language; e.g. `(([])[])`	Depends on the language; e.g. `([)], (()`
LSOLVE	Is there a vector x that satisfies Ax = b?	Gauss-Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 2 \\ 4 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

Class NP

- Verification algorithm intuition
  - Verifier views things from "managerial" viewpoint.
  - Verifier doesn't determine whether w $\in$ L on its own; rather, it checks with a proposed hint whether w $\in$ L.

- Algorithm V($\langle w, c \rangle$) is a **verifier** for language L if for every string w, **w $\in$ L** iff **there *exists* a string c such that V($\langle w, c \rangle$) accepts**.

"certificate" or "witness"

- The running time of a verifier is measured only in terms of length of w. A **polynomial-time verifier** runs in time polynomial in |w| and has certificate c of length polynomial in w:
  i.e., |c|=O($|w|^k$) for some constant k.

The class NP

NP is the class of languages that have polynomial-time verifiers.

Examples of languages in NP

- COMPOSITES = $\{\langle x \rangle \mid x = pq$, for int $p, q > 1\}$
- certificate: integer $p > 1$ that divides x

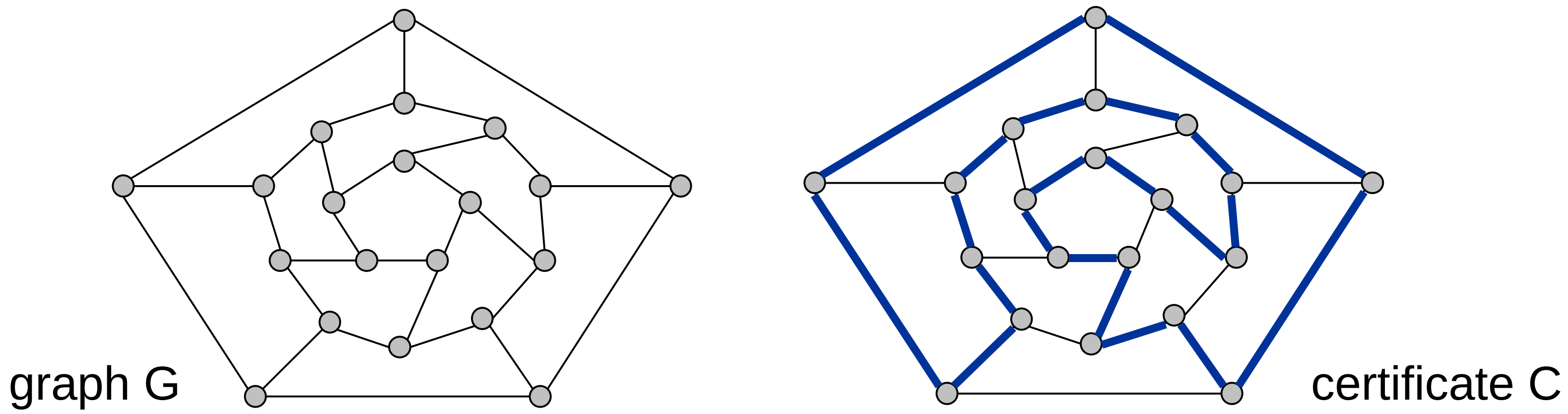
  such a certificate exists iff x is composite
- verifier

  V = `` On input $\langle x, p \rangle$, where x and p are integers:

  1. **If $p \leq 1$ or $p \geq x$, reject.**
  2. **Else if (x is a multiple of p, accept. O.w. reject."**

Examples of languages in NP

- UHamCycle = $\{\langle G\rangle \mid G$ is an undirected graph that contains a cycle C that visits each node exactly once$\}$
- certificate C: Hamiltonian cycle (i.e., permutation of the nodes)



Examples of languages in NP

- UHamCycle = $\{\langle G\rangle \mid G$ is an undirected graph that contains a cycle C that visits each node exactly once$\}$
- certificate C: Hamiltonian cycle (i.e., permutation of the nodes)
- verifier **V = `` On input $\langle G, C\rangle$:**

  1. **Accept if**
  2. **each node of G appears in C exactly once**
  3. **there is an edge between every pair of adjacent nodes in C**
  4. **O.w. reject.”**

Examples of languages in NP: SAT

- **Boolean variables:** variables that can take on values T/F (or 1/0)
- **Boolean operations:** $\vee$, $\wedge$, and $\neg$
- **Boolean formula:** expression with Boolean variables and ops

  Example: $(x_1 \vee \overline{x_2}) \wedge x_3$
- An **assignment** of 0s and 1s to the variables **satisfies** formula φ if it makes it evaluate to 1.
- φ is **satisfiable** if there exists an assignment that satisfies it.

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

Prove: SAT $\in$ **NP**.

Classes P, NP, EXP

- **P**. **Languages for which there is a poly-time algorithm.**

  algorithm that runs in time $O(n^k)$ for some k
- **EXP**. **Languages for which there is an exponential-time algorithm.**

  algorithm that runs in time $O(2^{n^k})$ for some k
- **NP**. **Languages for which there is a poly-time verifier.**

- **Lemma.** P $\subseteq$ NP.
- **Lemma.** NP $\subseteq$ EXP.
- **Lemma.** A language L is in NP iff L can be decided by a polynomial-time nondeterministic TM.

P vs. NP

- Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
  - Is the decision problem as easy as the verification problem?
  - Clay $1 million prize.

would break RSA cryptography (and potentially collapse economy)

- If yes: Efficient algorithms for UHamPath, SAT, TSP, factoring
- If no: No efficient algorithms possible for these problems.

- Consensus opinion on P = NP? Probably no.