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 $2, 3, \ldots, \sqrt{b}$.

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:

$$P = \bigcup_{k} \text{TIME}(n^k).$$
## Examples of languages in P

<table>
<thead>
<tr>
<th>Problem</th>
<th>Description</th>
<th>Algorithm</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULTIPLE</td>
<td>Is x a multiple of y?</td>
<td>Grade school division</td>
<td>51, 17</td>
<td>51, 16</td>
</tr>
<tr>
<td>RELPRIME</td>
<td>Are x and y relatively prime?</td>
<td>Euclid (300 BCE)</td>
<td>34, 39</td>
<td>34, 51</td>
</tr>
<tr>
<td>PRIMES</td>
<td>Is x prime?</td>
<td>AKS (2002)</td>
<td>53</td>
<td>51</td>
</tr>
<tr>
<td>all CFLs</td>
<td>(e.g. the language of balanced parentheses and brackets)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is the string in the given CFL? (e.g., is the string of parentheses and brackets balanced?)</td>
<td>Dynamic programming</td>
<td>Depends on the language; e.g. ([][][])</td>
<td>Depends on the language; e.g. ([]), ()</td>
</tr>
<tr>
<td>LSOLVE</td>
<td>Is there a vector x that satisfies Ax = b?</td>
<td>Gauss-Edmonds elimination</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4/7/2016
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, \text{ for int } p, q > 1 \}  
• certificate: integer \( p > 1 \) that divides \( x \)  
such a certificate exists iff \( x \) is composite  
• verifier  

\[ V = \text{``On input } \langle x, p \rangle, \text{ where } x \text{ and } p \text{ are integers:} \]

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 \text{ is an undirected graph that contains a cycle } C \text{ that visits each node exactly once}\}
- certificate C: Hamiltonian cycle (i.e., permutation of the nodes)

graph G

certificate C
Examples of languages in NP

- \text{UHamCycle} = \{ \langle G \rangle \mid G \text{ is an undirected graph that contains a cycle } C \text{ that visits each node exactly once} \}
- \text{certificate } C: \text{ Hamiltonian cycle (i.e., permutation of the nodes)}
- \text{verifier } V = ``\text{On input } \langle G, C \rangle:\text{''}
  1. \text{Accept if}
  2. each node of } G \text{ appears in } C \text{ exactly once}
  3. \text{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**: ∨, ∧, and ¬
- **Boolean formula**: expression with Boolean variables and ops
  Example: \((x_1 \lor \overline{x_2}) \land x_3\)
- An **assignment** of 0s and 1s to the variables **satisfies** formula \(\varphi\) if it makes it evaluate to 1.
- \(\varphi\) is **satisfiable** if there exists an assignment that satisfies it.

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

**Prove:** \(\text{SAT} \in \text{NP}\).
Classes P, NP, EXP

- **P.** Languages for which there is a poly-time algorithm. An algorithm that runs in time $O(n^k)$ for some $k$

- **EXP.** Languages for which there is an exponential-time algorithm. An 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.
Does $P = NP$? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

- Is the decision problem as easy as the verification problem?
- Clay $1$ million prize.

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.