BU CS 332 – Theory of Computation

Lecture 22:
• NP

Reading:
Sipser Ch 7.3-7.4

Mark Bun
November 23, 2021
Nondeterministic time and NP

Let \( f : \mathbb{N} \to \mathbb{N} \)

A NTM \( M \) runs in time \( f(n) \) if on every input \( w \in \Sigma^n \), \( M \) halts on \( w \) within at most \( f(n) \) steps on every computational branch.

\( \text{NTIME}(f(n)) \) is a class (i.e., set) of languages:

A language \( A \in \text{NTIME}(f(n)) \) if there exists an NTM \( M \) that

1) Decides \( A \), and
2) Runs in time \( O(f(n)) \)

**Definition**: NP is the class of languages decidable in polynomial time on a nondeterministic TM

\[
\text{NP} = \bigcup_{k=1}^{\infty} \text{NTIME}(n^k)
\]
Speeding things up with nondeterminism

HW 5 Problem 3:

\[ TRIANGLE = \{ \langle G \rangle | \text{digraph } G \text{ contains a triangle} \} \]

**Deterministic algorithm:**

\[ G = (V, E) \]

For each \( u \in V : \)

For each \( v \in V : \)

For each \( w \in V : \)

Check: \( (u,v), (v,w), (w,u) \in E \)

Runtime \( \geq |V|^3 \)

\# vertices = \( |V| \)

\( V = \{1, \ldots, n\} \)

Write down a \( \# \) between 1 and \( n \) takes \( \log_2 n \) bits

**Nondeterministic algorithm:**

1) Nondeterministically guess \( (u,v,w) \)

2) Check if \( (u,v), (v,w), (w,u) \in E \)

Guess \( O(\log |V|) \) bits
Hamiltonian Path

\[ HAMPATH = \{ \langle G, s, t \rangle \mid G \text{ is a directed graph and there is a path from } s \text{ to } t \text{ that passes through every vertex exactly once} \} \]
**HAMPATH ∈ NP**

The following **nondeterministic** algorithm decides **HAMPATH** in polynomial time:

On input \( \langle G, s, t \rangle \): (Vertices of \( G \) are numbers 1, \( ... \), \( k \))

1. **Nondeterministically** guess a sequence \( c_1, c_2, ... , c_k \) of numbers 1, \( ... \), \( k \)

2. Check that \( c_1, c_2, ... , c_k \) is a permutation: Every number 1, \( ... \), \( k \) appears exactly once

3. Check that \( c_1 = s \), \( c_k = t \), and there is an edge from every \( c_i \) to \( c_{i+1} \)

4. **Accept** if all checks pass, otherwise, **reject**.
Analyzing the algorithm

Need to check:

1) Correctness
   a) If \( (s,t) \in \text{HAMPATH} \), \( s \) a Hamiltonian path \( c_1, \ldots, c_k \)
      \( \Rightarrow \) branch of computation that guesses this path leads NTM to accept
   b) If \( (s,t) \notin \text{HAMPATH} \), then every \( c_1, \ldots, c_k \) is not a Hamiltonian path from \( s \) to \( t \).
      \( \Rightarrow \) Every computation branch rejects

2) Running time
   Guessing \( c_1, \ldots, c_k \) takes \( \Theta(k \log k) \) time
   Checking permutation takes \( O(k \log^2 k) \) time
   Checking \( c_1, \ldots, c_k \) is an s-t path takes \( O(k) \) lookups
to adj. matrix / adj. list of \( G \)
An alternative characterization of \textbf{NP}

“Languages with polynomial-time verifiers”

How did we design an NTM for HAMPATH?

• Given a candidate path, it is easy (poly-time) to check whether this path is a Hamiltonian path

• We designed a poly-time NTM by nondeterministically guessing this path and then checking it

• Lots of problems have this structure (CLIQUE, 3-COLOR, COMPOSITE,...). They might be hard to solve, but a candidate solution is easy to check.
An alternative characterization of NP

“Languages with polynomial-time verifiers”

A verifier for a language $L$ is a deterministic algorithm $V$ such that $w \in L$ iff there exists a string $c$ such that $V(\langle w, c \rangle)$ accepts.

Running time of a verifier is only measured in terms of $|w|$

$V$ is a polynomial-time verifier if it runs in time polynomial in $|w|$ on every input $\langle w, c \rangle$.

(Without loss of generality, $|c|$ is polynomial in $|w|$, i.e., $|c| = O(|w|^k)$ for some constant $k$)
**HAMPATH** has a polynomial-time verifier

Certificate $c$: $(c_1, \ldots, c_n)$ a candidate path

Verifier $V$:

On input $\langle G, s, t; c \rangle$: (Vertices of $G$ are numbers 1, ..., $k$)

1. Check that $c_1, c_2, \ldots, c_k$ is a permutation: Every number 1, ..., $k$ appears exactly once

2. Check that $c_1 = s$, $c_k = t$, and there is an edge from every $c_i$ to $c_{i+1}$

3. **Accept** if all checks pass, otherwise, **reject**.

**Correctness:**

a) If $\langle G, s, t \rangle \in L$, $\exists$ Hamiltonian path $c_1, \ldots, c_n$ from $s$ to $t$

   $\Rightarrow \exists$ $c$ s.t. $V(\langle G, s, t; c \rangle)$ accepts

b) If $\langle G, s, t \rangle \notin L$, then every $c$ is not a Hamilton path $\Rightarrow \forall c$

   $V(\langle G, s, t; c \rangle)$ rejects
NP is the class of languages with polynomial-time verifiers

**Theorem:** A language \( L \in \text{NP} \) iff there is a polynomial-time verifier for \( L \)
Alternative proof of $\text{NP} \subseteq \text{EXP}$

One can prove $\text{NP} \subseteq \text{EXP}$ as follows. Let $V$ be a verifier for a language $L$ running in time $T(n)$. We can construct a $2^{O(T(n))}$ time algorithm for $L$ as follows.

a) On input $\langle w, c \rangle$, run $V$ on $\langle w, c \rangle$ and output the result

b) On input $w$, run $V$ on all possible $\langle w, c \rangle$, where $c$ is a certificate. Accept if any run accepts.

c) On input $w$, run $V$ on all possible $\langle w, c \rangle$, where $c$ is a certificate of length at most $T(|w|)$. Accept if any run accepts.

d) On input $w$, run $V$ on all possible $\langle x, c \rangle$, where $x$ is a string of length $|w|$ and $c$ is a certificate of length at most $T(|w|)$. Accept if any run accepts.
NP is the class of languages with polynomial-time verifiers

Theorem: A language $L \in \text{NP}$ iff there is a polynomial-time verifier for $L$

Proof: $\iff$ Let $L$ have a poly-time verifier $V((w, c))$

Idea: Design NTM $N$ for $L$ that nondeterministically guesses a certificate

\[
\text{NTM } N: \\
\text{On input } w: \\
\begin{align*}
1) \text{Nondet. guess } c \text{ of length } \leq T(1|w|) \\
2) \text{Run } V((w, c)). \text{ If accepts, accept. If rejects, reject}
\end{align*}
\]

Correctness:

$w \in L \iff \exists$ $c$, $|c| \leq T(1|w|)$ s.t. $V((w, c))$ accepts

$\iff N$ accepts $w$ on some comp. branch

Runtime:

$|c|$ is polynomial in $|w|$

$\iff$ step 1 is poly-time

$V$ runs in poly-time $\iff$

step 2 poly-time
NP is the class of languages with polynomial-time verifiers

⇒ Let $L$ be decided by an NTM $N$ running in time $T(n)$ and making up to $b$ nondeterministic choices in each step.

Idea: Design verifier $V$ for $L$ where certificate is sequence of “good” nondeterministic choices.

Certificate: $C = (c_1, ..., c_{T(n)}) \in [b]^{T(n)}$

$C_i$ in the “good” nondet. choice at time step $i$

$V(<w, C>):$

1) Simulate $N$ on input $w$, where at every time step $i$, make nondet. choice $c_i$

2) If simulation reaches accept, accept; reject, reject.
WARNING: Don’t mix-and-match the NTM and verifier interpretations of NP
To show a language $L$ is in NP, do exactly one:

1) Exhibit a poly-time NTM for $L$
   $N = “On input w:"
   $<Do some nondeterministic stuff>...”$

   OR

2) Exhibit a poly-time (deterministic) verifier for $L$
   $V = “On input w and certificate c:"
   $<Do some deterministic stuff>...”$
Examples of **NP** languages: SAT

“Is there an assignment to the variables in a logical formula that make it evaluate to true?”

- **Boolean variable**: Variable that can take on the value true/false (encoded as 0/1) \( \text{Ex: } x_1, x_2, x_3 \)
  \( x, y, z \)

- **Boolean operations**: \( \land \) (AND), \( \lor \) (OR), \( \neg \) (NOT)

- **Boolean formula**: Expression made of Boolean variables and operations. \( \text{Ex: } (x_1 \lor \overline{x_2}) \land x_3 \)

- **An assignment** of 0s and 1s to the variables **satisfies** a formula \( \varphi \) if it makes the formula evaluate to 1 \( (1, 1, 1) \) is a **satisfying ass'mt**

- **A formula** \( \varphi \) is **satisfiable** if there exists an assignment that satisfies it
Examples of NP languages: SAT

Ex: \((x_1 \lor \overline{x_2}) \land x_3\)  
Yes because \((1,1,1)\) is a sat assmt

Ex: \((x_1 \lor x_2) \land \overline{x_1} \land \overline{x_2}\) = \((x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2})\)
Not satisfiable (no possible sat. assmt)

\[ SAT = \{(\varphi) | \varphi \text{ is a satisfiable formula}\} \]

Claim: \( SAT \in NP \)

<table>
<thead>
<tr>
<th>NTM for SAT</th>
<th>Poly-time verifier for SAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>On input (\varphi(x_1, \ldots, x_m)):</td>
<td></td>
</tr>
<tr>
<td>1) Nondelet. ques ((C_1, \ldots, C_m) \in {0,1}^m)</td>
<td></td>
</tr>
<tr>
<td>2) Evaluate (\varphi(C_1, \ldots, C_m)).</td>
<td></td>
</tr>
<tr>
<td>If True : accept</td>
<td></td>
</tr>
<tr>
<td>If False : reject</td>
<td></td>
</tr>
</tbody>
</table>

| Certificate \(C = (C_1, \ldots, C_m) \in \{0,1\}^m\) |
| On input \(\varphi, C\): |
| 1) Evaluate \(\varphi(C_1, \ldots, C_m)\). |
| If True : accept  |
| If False : reject |