BU CS 332 – Theory of Computation

https://forms.gle/ERY3Hk4MWvvBzGxb7

Lecture 21:

- NP: Nondeterminstic TMs vs. Deterministic Verifiers

Reading:

Sipser Ch 7.3-7.4

Mark Bun

April 16, 2025

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

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

A language $A \in \mathrm{NTIME}(f(n))$ 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

$$\mathrm{NP} = \bigcup_{k=1}^{\infty} \mathrm{NTIME}(n^k) = \mathrm{NTIME}(n) \cup \mathrm{NTIME}(n^2) \cup \mathrm{NTIME}(n^3) \cup \ldots$$

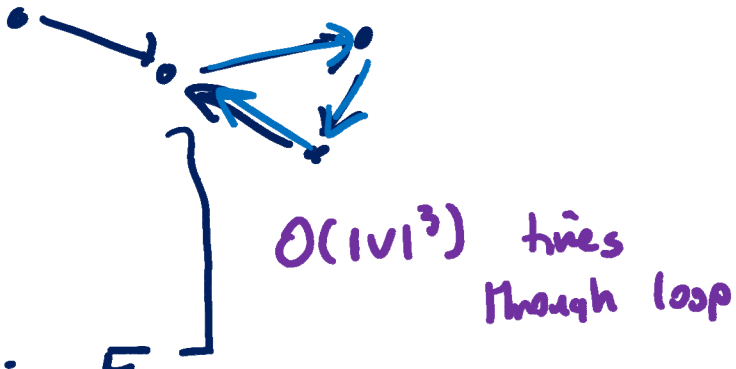
$$= \{ L \mid \exists \text{ NTM } M \text{ deciding } L \text{ in poly-time} \}$$

Speeding things up with nondeterminism

$$TRIANGLE = \{\langle G \rangle \mid \text{digraph } G \text{ contains a triangle}\}$$

$\langle G \rangle = (V, E)$

Deterministic algorithm:



For $u \in V$:
  For $v \in V$:
    For $w \in V$:
      Test if (u,v), (v,w), (w,u) all in E
      If so, accept.
Reject.

$O(|V|^3)$ times through loop

Nondeterministic algorithm:

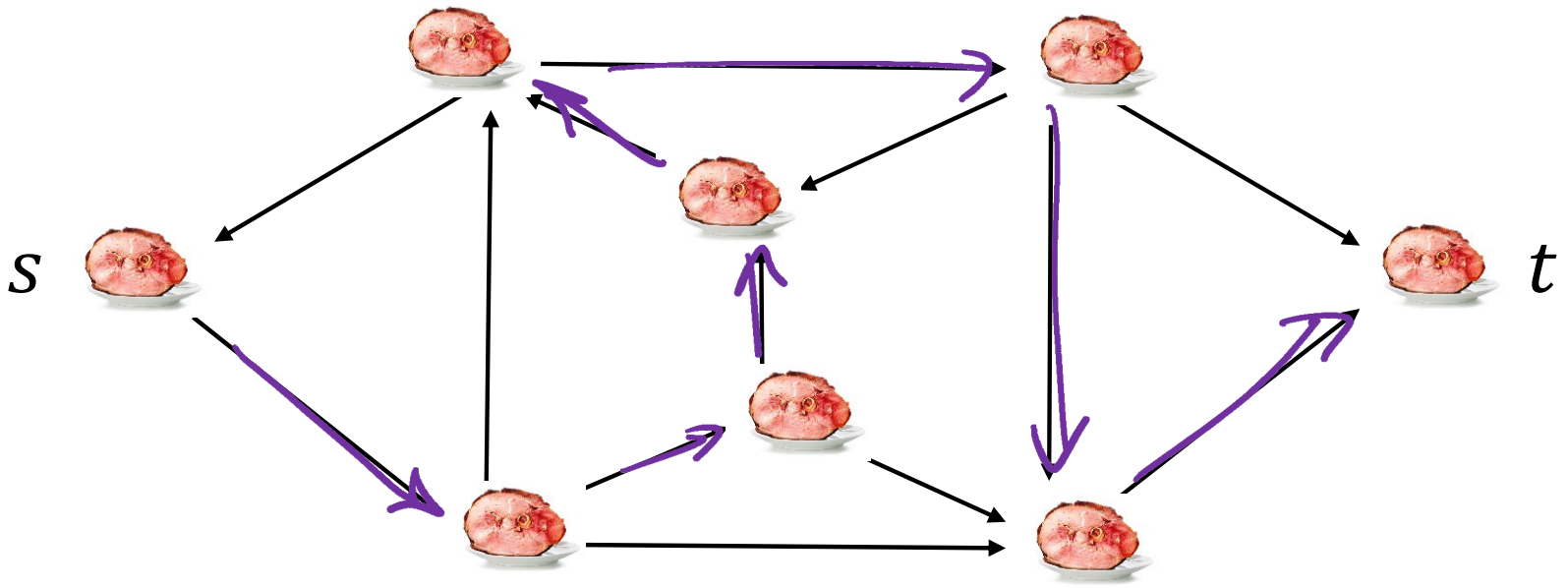
Nondeterministically guess $u, v, w \in V$ } $O(\log |V|)$

Test if (u,v), (v,w), (w,u) all in E. If so accept, else reject.

Hamiltonian Path

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

"Hamiltonian path from s to t"



$HAMPATH \in$ NP

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

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

1. **Nondeterministically** guess a sequence $c_1, c_2, \ldots, c_k$ of numbers $1, \ldots, k$
2. Check that $c_1, c_2, \ldots, c_k$ is a permutation: Every number $1, \ldots, k$ 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

- $\langle G, s, t \rangle \in$ HAMPATH $\Rightarrow$ $\exists$ $c_1, \ldots, c_k$ forming a Hamiltonian path from s to t in G

  $\Rightarrow$ branch of nondeterminism on which $c_1, \ldots, c_k$ was guessed
  1) hits every vertex once 2) forms a path from s to t
  $\Rightarrow$ all checks pass, NTM accepts

- $\langle G, s, t \rangle \notin$ HAMPATH $\Rightarrow$ $\forall$ $c_1, \ldots, c_k$ fail to form a Hamiltonian path
  $\Rightarrow$ all branches of nondeterminism lead to guessed $c_1, \ldots, c_k$ that fails at least one check $\Rightarrow$ all branches reject.

2) Running time

$O(k \log k)$ + $O((k \log k)^2)$ + $O(k^3 \log k) = O(k^3 \log k)$

$O(k \log k)$: guess candidate path

$O((k \log k)^2)$: check candidate path hits every vertex once

$O(k^3 \log k)$: check candidate path is a path

which is polynomial as a function of input length k^2

Nondeterministically guessing, then checking

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 deterministically checking it
- Lots of problems have this structure (CLIQUE, 3-COLOR, FACTOR,…). They might be hard to solve, but a candidate solution is easy to check.

General structure: $w \in L$ if and only if there exists a nondeterministically guessable, but deterministically checkable c

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

"certificate" "witness" "proof"

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: $c_1, c_2, \ldots, c_k$ a list of vertices representing a candidate path

Verifier V:

Runtime of algorithm: $O(k^3 \log k)$ is polynomial in $|w| \approx k^2 + 2\log k$

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

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

- $\langle G, s, t \rangle \in HAMPATH \Rightarrow \exists\ c_1, \ldots, c_k$ a Hamiltonian path from s to t
  $\Rightarrow \exists\ c = c_1, \ldots, c_k$ s.t. $V(\langle G, s, t; c \rangle)$ accepts
- $\langle G, s, t \rangle \notin HAMPATH \Rightarrow \forall\ c_1, \ldots, c_k$ fails to be a Hamiltonian path
  $\Rightarrow \forall\ c = c_1, \ldots, c_k$ causes $V(\langle G, s, t; c \rangle)$ to reject.

NP is the class of languages with polynomial-time verifiers

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

Alternative proof of NP ⊆ EXP

- $\forall w \in L$ $\exists c$ s.t. $V(\langle w, c\rangle)$ accepts
- $\forall w \notin L$ $\forall c$ $V(\langle w, c\rangle)$ reject

] time $T(n)$

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

- $\forall w \in L$ $M(w)$ accepts
- $\forall w \notin L$ $M(w)$ rejects

] in time $2^{O(T(n))}$

$L \in NP \Rightarrow \exists V$ a verifier for L running in time $O(n^k)$

$\Rightarrow L$ decidable in time $2^{O(n^k)}$

$\Rightarrow L \in EXP$

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 string. Accept if any run accepts.

(could have unbounded length)

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.

$2^{O(T(|w|))}$ possible c's of this length

Each run takes $O(T(|w|))$ time

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

Think $T(n) = \text{poly}(n)$

Proof: $\Leftarrow$ Let L have a time-$T(n)$ verifier $V(\langle w, c \rangle)$

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

NTM N:

On input w:

1) Nondeterministically guess c of length $\leq T(|w|)$
2) Run $V(\langle w, c \rangle)$. If accepts, accept. If rejects, reject.

Runtime: $O(T(|w|))$ time to guess c
+ $O(T(n))$ time to run V
$\Rightarrow O(T(|w|))$ total time.

Correctness:

- If $w \in L$:
  Correctness of $V \Rightarrow \exists c$ s.t. $V(\langle w, c \rangle)$ accepts
  WLOG $|c| \leq T(|w|)$ since V only has time to read $T(|w|)$ symbols
  $\Rightarrow$ branch where c is guessed leads N to accept.
- If $w \notin L$: $\forall c$ $V(\langle w, c \rangle)$ rejects
  $\Rightarrow$ all branches N rejects.

NP is the class of languages with polynomial-time verifiers

$\Rightarrow$ 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

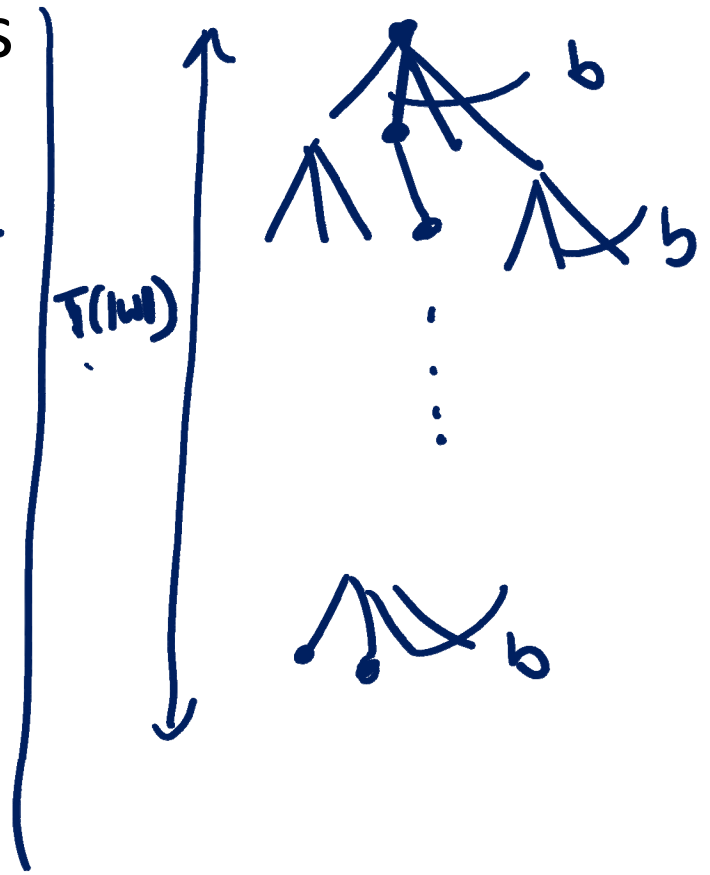
Certificate $c \in [b]^{T(|w|)}$

where each c_i represents "take the c_ith choice at time step i"

Verifier V:

On input $\langle w, c \rangle$:

1) Simulate running N on input w using nondeterministic choices represented by c
2) If N accepts, accept, if rejects, 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)
- Boolean operations: $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT)
- Boolean formula: Expression made of Boolean variables and operations. Ex: $\varphi(x_1, x_2, x_3) = (x_1 \vee \overline{x_2}) \wedge x_3$
- An assignment of 0s and 1s to the variables satisfies a formula φ if it makes the formula evaluate to 1
- A formula φ is satisfiable if there exists an assignment that satisfies it

Examples of NP languages: SAT

Ex: $(x_1 \vee \overline{x_2}) \wedge x_3$ Satisfiable?

YES, since $x_1=1, x_2=1, x_3=1$ is a satisfying assignment

Ex: $(x_1 \vee x_2) \wedge \overline{x_1} \wedge \overline{x_2}$ Satisfiable?

Not Satisfiable

$$SAT = \{\langle \varphi \rangle | \varphi \text{ is a satisfiable formula}\}$$

Claim: $SAT \in \text{NP}$