# BU CS 332 – Theory of Computation

https://forms.gle/nUoNjcqHxqVZm8nk6

## Lecture 21:

- NP: Nondeterminstic TMs vs.
  Deterministic Verifiers

Reading:

Sipser Ch 7.3-7.4

Mark Bun

April 22, 2024

# Nondeterministic time and NP

Let $f : \mathbb{N} \rightarrow \mathbb{N}$

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

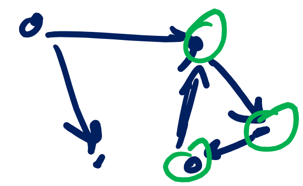
$$\mathrm{NP} = \bigcup_{k=1}^{\infty} \mathrm{NTIME}(n^k)$$

# Speeding things up with nondeterminism

$= (V, E)$

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

## Deterministic algorithm:



For each $u \in V$:
  For each $v \in V$:
    For each $w \in V$:
      Check if $(u,v)$, $(v,w)$, and $(w,u)$ are all in $E$.

Takes $O(|V|^3)$

## Nondeterministic algorithm:

Nondeterministically guess $u, v, w \in V$

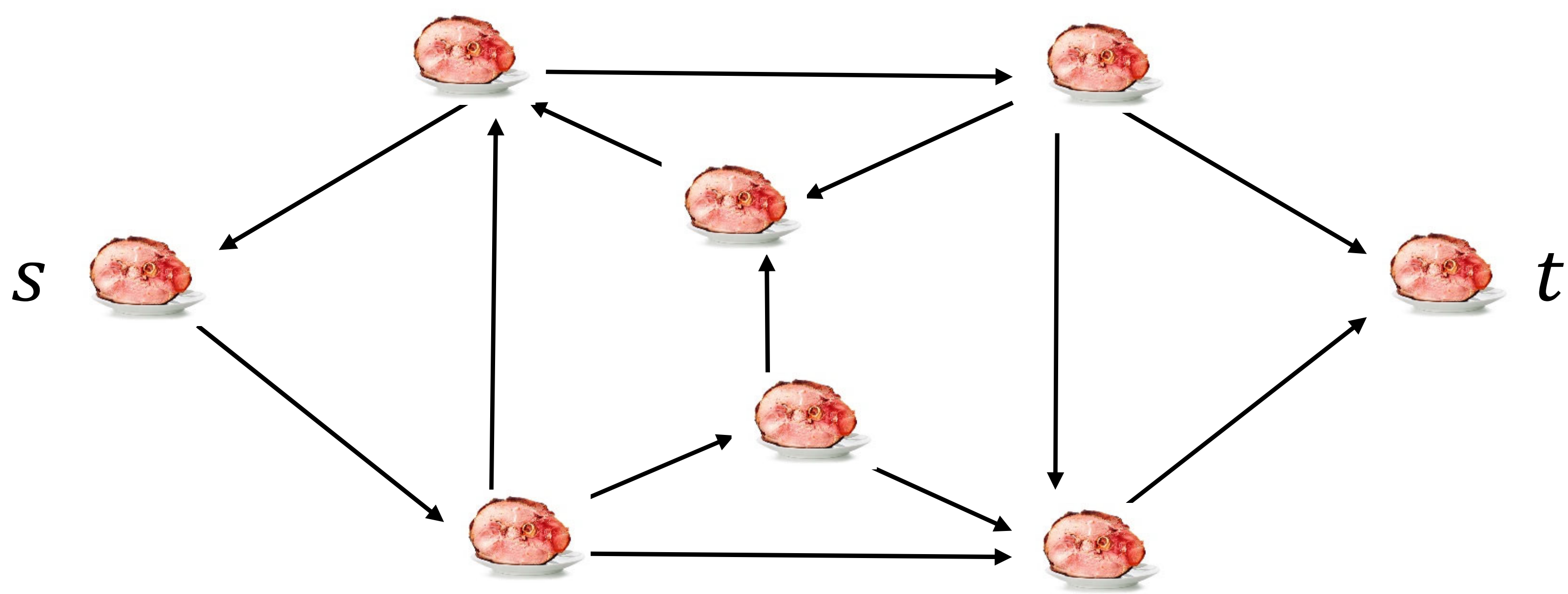
3 vertices, each takes $\log|V|$ bits to write down

$O(\log|V|)$

Check if $(u,v)$, $(v,w)$, and $(w,u)$ are all in $E$

# Hamiltonian Path

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



# $HAMPATH \in \text{NP}$

$c_1, c_2, \ldots, c_k$

length $k \log k$

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

Adj. matrix takes $k^2$ bits to write down

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 $1, \ldots, k$ $O(k \log k)$
2. Check that $c_1, c_2, \ldots, c_k$ is a permutation: Every number $1, \ldots, k$ appears exactly once $O((k \log k)^2)$ time (or so)
3. Check that $c_1 = s$, $c_k = t$, and there is an edge from every $c_i$ to $c_{i+1}$ $O(k)$ edges to check $\cdot$ $O(k^2)$ time per edge
4. Accept if all checks pass, otherwise, reject.

# Analyzing the algorithm

Need to check:

1) Correctness

- If $\langle G, s, t\rangle \in HAMPATH$: $\exists$ a sequence of vertices $c_1, \ldots, c_n$ that form a Ham path from s to t in G

$\Rightarrow$ Branch of computation where $c_1, \ldots, c_n$ was guessed leads to acceptance

- If $\langle G, s, t\rangle \notin HAMPATH$: No sequence of vertices $c_1, \ldots, c_n$ forms a Ham path from s to t

$\Rightarrow$ Every sequence will either fail to be a path or fail to be a permutation of [k]

2) Running time

$\Rightarrow$ Every branch will lead to some check failing.

$$\underbrace{O(k \log k)}_{\text{Guess the candidate path}} + \underbrace{O(k^2}_{\text{Check path is a permutation}} \text{poly}(\log k)) + \underbrace{O(k) \cdot O(k^2)}_{\text{Check actually a path from } s \text{ to } t}$$

$= O(k^3)$ $=$ poly( input length )

$\approx k^2$

# Nondeterministically guessing, then checking

How did we design an NTM for HAMPATH? $w = \langle G, s, t \rangle$

- 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, COMPOSITE,…). 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 $\boxed{w \in L}$ 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 $c$: $c_1, \dots, c_k$ representing an alleged Ham path

Run time:

Same analysis as before, but note that valid certificate has length $k \log k = \text{poly}(n)$, so verification still takes time poly in $|\langle G,s,t\rangle|$.

Verifier $V$:

On input $\langle \underbrace{G, s, t}_{w \text{ (to be checked if in HAMPATH)}} ; \underbrace{c}_{\text{certificate}} \rangle$: (Vertices of $G$ are numbers $1, \dots, k$)

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

- If $\langle G,s,t\rangle \in HAMPATH$, then $c$ representing a Ham path from $s$ to $t$ in $G$ causes $V$ to accept on $\langle G,s,t;c\rangle$
- If $\langle G,s,t\rangle \notin HAMPATH$, then no sequence $c$ is a valid Ham path $\Rightarrow V$ will reject on $\langle G,s,t;c\rangle$.

# 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 $\mathrm{NP} \subseteq \mathrm{EXP}$

$w \in L \Rightarrow \exists$ certificate $c$ s.t. $V(\langle w, c \rangle)$ accepts

$w \notin L \Rightarrow \forall$ certificates $c$ $V(\langle w, c \rangle)$ rejects

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

$M$

$w \in L \Rightarrow M(w)$ accepts

$w \notin L \Rightarrow M(w)$ rejects

~~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.

Need to be careful that we don't try arbitrarily long strings

$\boxed{\text{c)}}$ 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.

Runtime $\underbrace{T(|w|)}_{\text{time per run of } V} \cdot \underbrace{2^{O(T(|w|))}}_{\text{\# of certificates to try}}$

~~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 $L \in \mathrm{NP}$ iff there is a polynomial-time verifier for $L$

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 appropriate poly length)
2. Run $V(\langle w, c\rangle)$. Accept if accepts. Reject if rejects.

Run time:

1. Takes poly time because we can take c to have poly length
2. Takes poly time because V does

Correctness:

If $w \in L \Rightarrow \exists$ c (of poly length) s.t. $V(\langle w, c\rangle)$ accepts [defn of correctness for V]

$\Rightarrow$ Branch of nondet. guessing where c is guessed leads N to accept.

If $w \notin L \Rightarrow \forall c$ $V(\langle w, c\rangle)$ rejects

$\Rightarrow$ Every branch of NTM computation leads to reject.

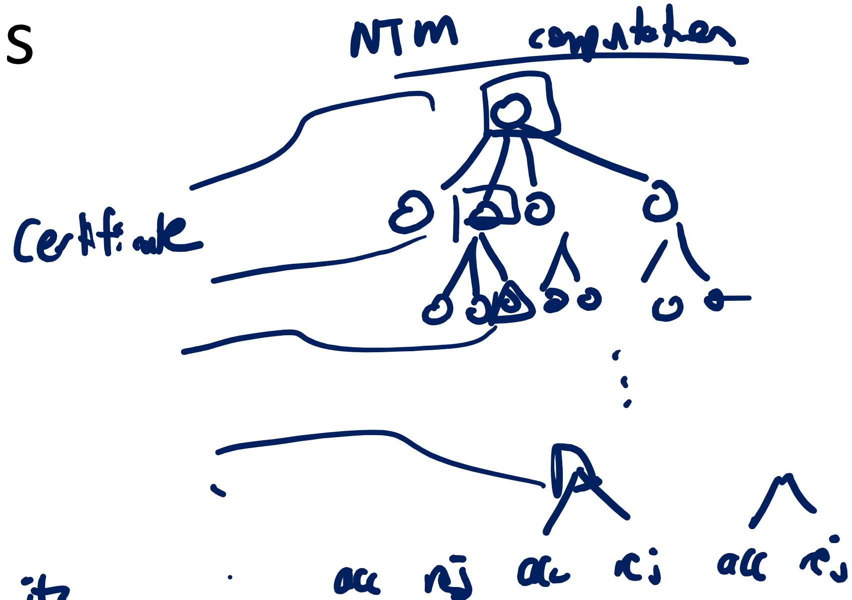
# NP is the class of languages with polynomial-time verifiers

Correctness: $w \in L \Rightarrow \exists$ a branch s.t. $N$ accepts $\Rightarrow V$ accepts when certificate encodes this branch

$w \notin L \Rightarrow \forall$ branches cause $N$ to reject $\Rightarrow V$ always rejects

$\Rightarrow$ 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: $c_1, \dots, c_{T(|w|)} \in [b]^{T(|w|)}$

$c_i$ indicates which of $b$ choices to make at time step $i$

Verifier $V$:

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

1. Simulate $N$ on input $w$ using $c$ as its nondeterministic choices
2. Accept if accepts, reject if rejects.

Runtime: Step 1 takes $T(|w|)$ time, which is polynomial when $T$ is.

# 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) $x, y, z$ $x_1, x_2, x_3$
- 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 $\varphi$ if it makes the formula evaluate to 1

  $x_1 = 0$ $x_3 = 1$ $\Rightarrow \varphi(0,0,1) = (0 \vee 1) \wedge 1 = 1$
  $x_2 = 0$

- A formula $\varphi$ is satisfiable if there exists an assignment that satisfies it $\varphi$ is satisfiable because 0,0,1 satisfies it.

# Examples of **NP** languages: SAT

Ex: $(x_1 \vee \overline{x_2}) \wedge x_3$ $x_1 = 0,\ x_2 = 0,\ x_3 = 1$ satisfies this Satisfiable?

Ex: $(x_1 \vee x_2) \wedge \overline{x_1} \wedge \overline{x_2}$ Not satisfiable Satisfiable?

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

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