BU CS 332 – Theory of Computation

Lecture 20:

- More on NP
- P vs. NP

HW 8 out, due April 21 (Tuesday)

Reading:

Sipser Ch 7.3-7.5

Mark Bun

April 13, 2020

Goals of complexity theory

Ultimate goal: Classify problems according to their feasibility and inherent computational difficulty

P $\approx$ Decision problems which can be solved efficiently

Can we exhibit general classes of problems which are either in P or provably not in P?

Some problems provably require exponential time! (Chapter 9)

NP: A fundamental and practically important class of problems which have defied classification, but nevertheless exhibits important structure (NP completeness)

Nondeterministic Time and NP

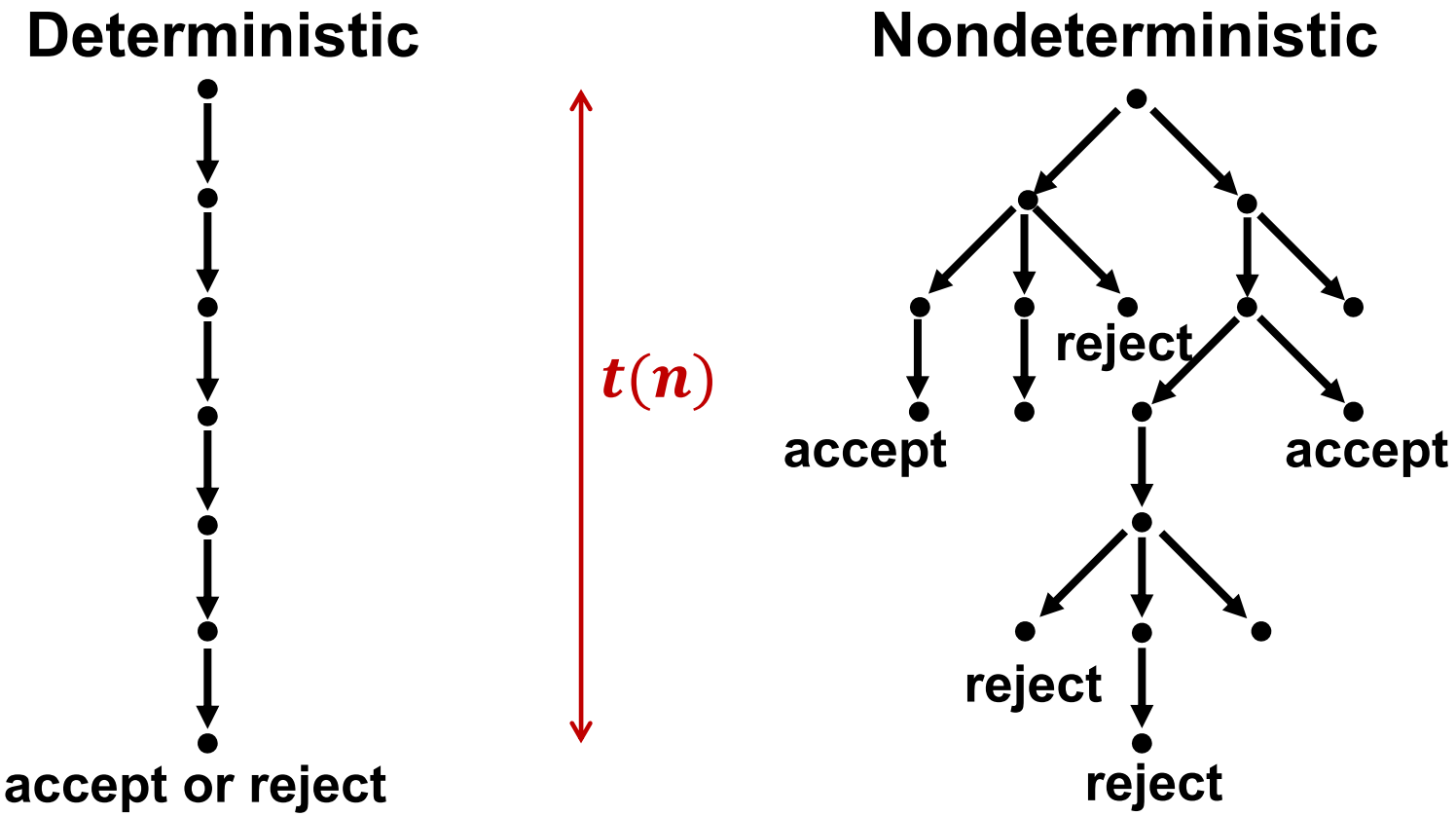
Nondeterministic time

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

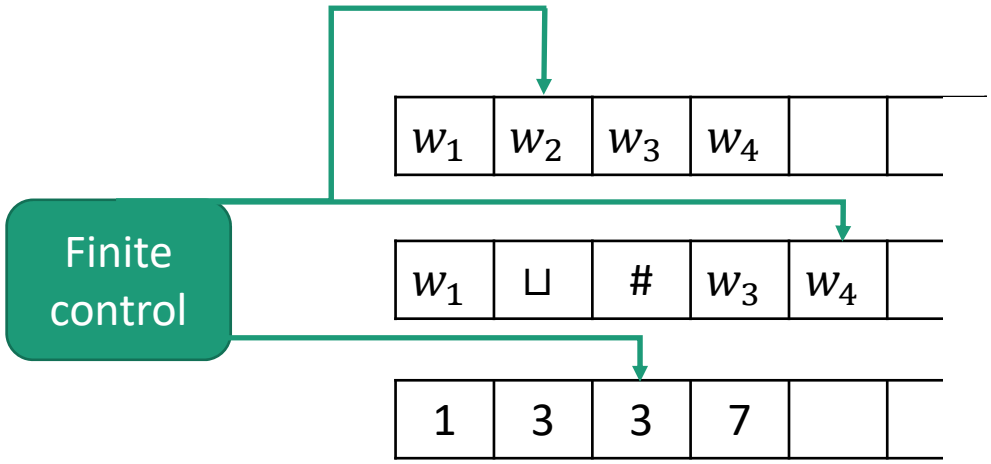
Deterministic vs. nondeterministic time



Deterministic vs. nondeterministic time

Theorem: Let $t(n) \geq n$ be a function. Every NTM running in time $t(n)$ has an equivalent single-tape TM running in time $2^{O(t(n))}$

Proof: Simulate NTM by 3-tape TM



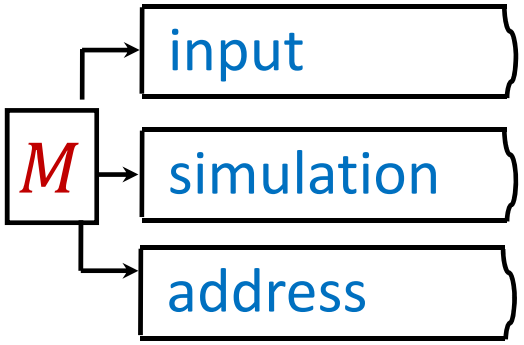
Input w to N (read-only)

Simulation tape (run N on w using nondeterministic choices from tape 3)

Address in computation tree

Deterministic vs. nondeterministic time

Theorem: Let $t(n) \geq n$ be a function. Every NTM running in time $t(n)$ has an equivalent single-tape TM running in time $2^{O(t(n))}$



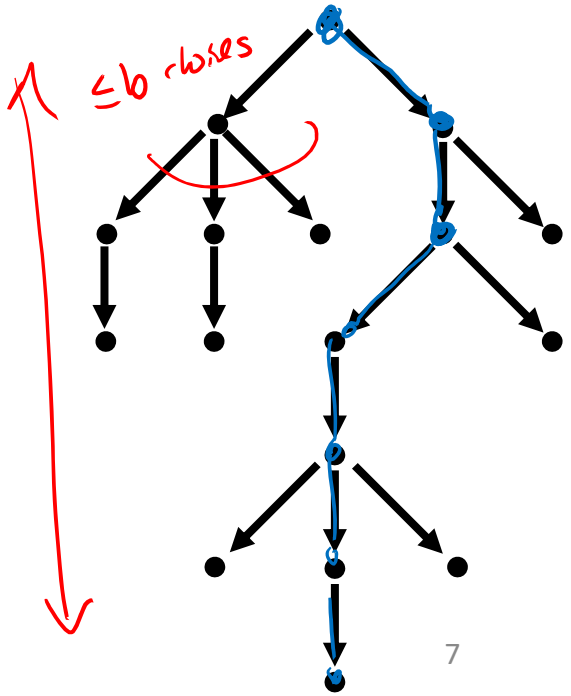
Proof: Simulate NTM by 3-tape TM

- # leaves: $b^{t(n)}$
- # nodes: $1 + b + b^2 + \ldots + b^{t(n)} \leq 2\, b^{t(n)}$

Running time:

To simulate one root-to-node path: $O(t(n))$

Total time: $2\, b^{t(n)} \cdot O(t(n)) = O(t(n) \cdot b^{t(n)}) = 2^{O(t(n))}$ (b is constant)



Deterministic vs. nondeterministic time

Theorem: Let $t(n) \geq n$ be a function. Every NTM running in time $t(n)$ has an equivalent single-tape TM running in time $2^{O(t(n))}$

Proof: Simulate NTM by 3-tape TM in time $2^{O(t(n))}$

We know that a 3-tape TM can be simulated by a single-tape TM with quadratic overhead, hence we get running time

$$(2^{O(t(n))})^2 = 2^{2 \cdot O(t(n))} = 2^{O(t(n))}$$

Therefore:

$$NTIME(t(n)) \subseteq TIME(2^{O(t(n))})$$

Difference in time complexity

Extended Church-Turing Thesis:

At most polynomial difference in running time between all (reasonable) deterministic models

At most exponential difference in running time between deterministic and nondeterministic models

Nondeterministic time

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

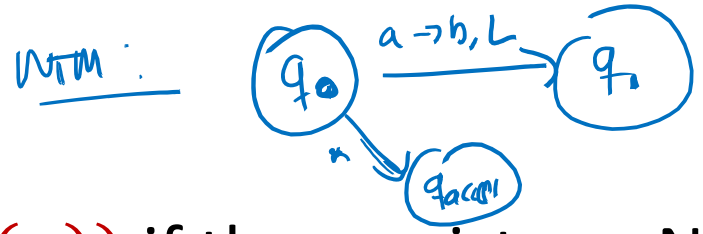
$\mathrm{NTIME}(f(n))$ is a class (i.e., set) of languages:

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

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

$\mathrm{NTIME}(f(n)) \subseteq \mathrm{TIME}(2^{O(f(n))})$

NTIME explicitly



A language $A \in \text{NTIME}(f(n))$ if there exists an NTM M such that, on every input $w \in \Sigma^*$

1. Every computational branch of M halts in either the accept or reject state within $f(|w|)$ steps
   [M runs in the f(n)]
   (or computational branch dies w/ no outgoing transitions)
2. $w \in A$ iff there exists an accepting computational branch of M on input w
3. $w \notin A$ iff every computational branch of M rejects on input w (or dies with no applicable transitions)

Together = M decides A

Complexity class NP

[Nondeterministic polynomial time]

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

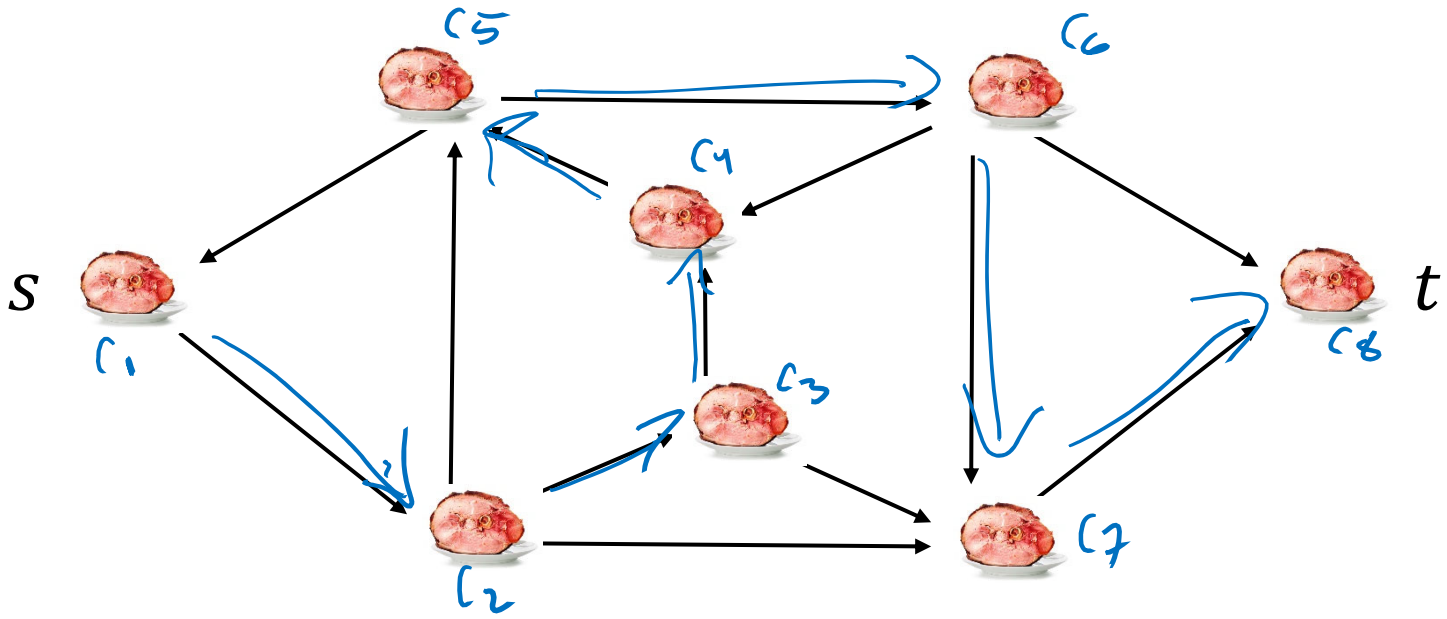
$$\mathrm{NP} = \bigcup_{k=1}^{\infty} \mathrm{NTIME}(n^k)$$

$P \subseteq NP \subseteq EXP$

why? $NTIME(n^4) \subseteq TIME(2^{O(n^4)})$

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$\}$



$HAMPATH \in$ NP

Analysis:
$\langle G,s,t\rangle \in$ HAMPATH $\Rightarrow \exists$ a nondeterministic guess of $c_1,\dots,c_n$ that passes all checks
$\langle G,s,t\rangle \notin$ HAMPATH $\Rightarrow$ Every guess of $c_1,\dots,c_n$ either fails to visit every vertex exactly once from s to t or fails to be a directed path

The following nondeterministic algorithm accepts in time $O(n^{1.5})$, where $n = |\langle G, s, t\rangle|$, adjacency matrix encoding

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

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

= check we visit every vertex exactly once

= check $c_1,\dots,c_n$ forms a path from s to t

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" "proof" "witness"

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: $c_1, \ldots, c_k \in \{1, \ldots, k\}$

Verifier V: V is deterministic, runs in time $O(n^{1.5})$ as function of $|\langle G, s, t \rangle|$

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.

why a verifier?

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

$\langle G, s, t \rangle \notin HAMPATH \Rightarrow \forall\, c$ causes $V(\langle G, s, t; c \rangle)$ to reject

NP is the class of languages with polynomial-time verifiers

(poly-time on an NTM)

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

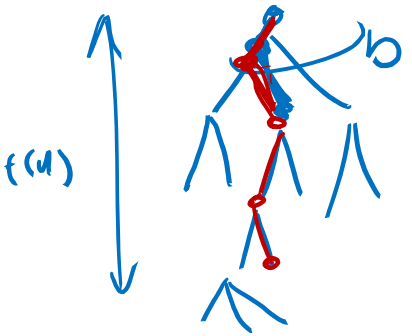
Proof: $\Rightarrow$ Let $L \in NP$, w/ NTM N deciding L in time $t(n)$, a polynomial, b nondeterministic choices in every step

Construct verifier $V(\langle w, c \rangle)$ w input, c certificate

Idea: Use c to encode one of $b^{t(n)}$ possible nondeterministic choices of machine N

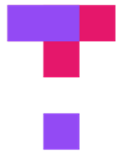
"On input $\langle w, c \rangle$:

1. Simulate N on w using c as sequence of nondeterministic choices
2. If N accepts, accept; ow. reject"



c = path through tree to "accept"

$\Leftarrow$ Let $V(\langle w, c\rangle)$ be a verifier for L running in time $T = O(n^k)$ for some const. k



Goal: Design a poly-time NTM deciding L

Idea: Nondeterministically guess c and use verifier to check it

NTM

"On input w:

1) Nondeterministically guess c of length $|c| \leq T$

2) Run $V(\langle w, c\rangle)$, accept if accepts, reject o.w."

$w \in L \Rightarrow \exists\ |c| \leq T.\ V(\langle w, c\rangle)$ accepts $\Rightarrow$ NTM accepts

$w \notin L \Rightarrow \forall\ |c| \leq T$ $V(\langle w, c\rangle)$ rejects $\Rightarrow$ NTM rejects on all branches