- coNP co NTM free
- written complete

Michael Forbes
forbes@cs.illinois.edu
2019-01-17.4 → 2019-01-22.1
2019-01-22.5 cs 579

cs 579 Computational Complexity : Lecture 3 23 3/

admin : post standards ‖ for writing also ‖
- high level language ‖ make clear you could implement low level details ‖
- correctness ‖ emphasize low level details when required ‖
- runtime

last time : TIME
P
PATH ∈ P

today : nondeterministic TMs
NTIME
NP
HAMPATH ∈ NP?

Question ?

PATH = { ⟨G, s, t⟩ : G = (V, E) s, t ∈ V, G has $s \rightsquigarrow t$ path }
↑ any reasonable encoding into Σ^* ($= \{0,1\}^*$) $\forall x \in V$

Prop : PATH ∈ P ‖ efficient polynomial time algo ↳ in input size ‖ "Hamiltonian Path"

HAMPATH = { ⟨G, s, t⟩ : G = (V, E) has $s \rightsquigarrow t$ path that visits all nodes in V exactly once }
an "algo" : ‖ not known to be in P ‖

on input ⟨G = (V, E), s, t⟩ :

n^n 1) for $V = \{v_1, \dots, v_n\}$, guess $v_{i_1}, \dots, v_{i_n} \in V$, $n = |V|$ ‖ correctness clear

poly(|x|) 2) reject if any repeated in guess
3) reject if $v_{i_1} \neq s$, $v_{i_n} \neq t$ or $(v_{i_j}, v_{i_{j+1}}) \notin E$
4) else accept ‖ efficient if guess correct ‖

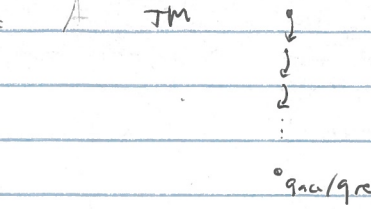
goal : model "guessing" as computational resource
‖ guess is not determined, never non-deterministic ‖
‖ not an implementable model, but helps understand problems we do want to solve ‖

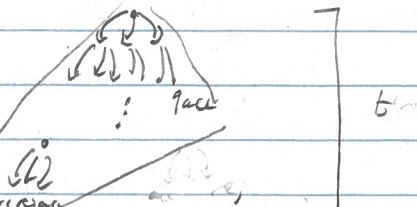
def : a nondeterministic Turing machine (NTM) is a TM with a transition function

$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,S,R\}}$ ‖ head move ‖ ‖ outputs a set of possible transitions ‖
‖ power set ‖ ‖ a guess for which are correct ‖

$|\delta(\cdot)| = 1$ ≡ deterministic TM ‖ state ‖ ‖ tape alphabet ‖
$|\delta(\cdot)| = 0$ ⇒ reject

A branch of a NTM computation is a sequence of valid transition functions.

eg : A TM


NTM


def : a branch accepts an input x if the transitions in the branch reach q_{acc} (q_{rej})
rejects

a NTM N accepts an input x if any branch of N on x accepts
rejects all rejects

the time of NTM N on x is the length of longest branch
‖ massive parallelism ‖

Michael Forbes
miforbes@illinois.edu

def. NTM N runs in time $t(n): \mathbb{N} \to \mathbb{N}$ if all $x \in \Sigma^*$, N acc/rej x in $\leq t(|x|)$ steps [all branches]

$\text{NTIME}(t(n)) = \{ L : L = L(N), N \text{ runs in } O(t(n)) \text{ steps} \}$
[those inputs accepted by N]

$NP = \bigcup_k \text{NTIME}(n^k)$ [nondeterministic polynomial time]

Rmk. - $P \subseteq NP$ [use single transition]
- NP is model invariant [eg changing # tapes in TM]

Prop. HAMPATH $\in NP$

idea: use many branches to try all paths

Pf. N = "on input $\langle G=(V,E), s, t \rangle$:
poly time
1) for $V = \{v_1, \dots, v_n\}$, $n = |V|$, write down $v_{i_1}, \dots, v_{i_n} \in V$
— as before —
accept if $s = v_{i_1} \to v_{i_2} \to \cdots \to v_{i_n} = t$ is valid hamiltonian path
reject else

Can guess with nondeterminism $\delta(q, \gamma) \to \{(q', 0, R), (q', 1, R)\}$
[sometimes unwieldy] [repeating yields stream of bits] [can turn into verifier w/ proper encoding]

def. a verifier for a language L is an algorithm V w/
$L = \{ w \mid V \text{ accepts } \langle x, w \rangle, \text{ some } w \in \Sigma^* \}$ [constant # needed to check]
the time of a verifier is a function of $|x|$, [eg $w = \bar{v}$]
$L = \{ x \mid \exists \langle x, w \rangle, w \in \Sigma^{\leq t(|x|)} \}$
$\iff$ poly time verifier has $|w| \leq \text{poly}(|x|)$

Prop. $L \in NP$ iff L has poly time verifier

Pf. $\Rightarrow$: let N be NTM accepting L in $t(n)$ steps
[for polynomial can be upper bounded by this for all n]
$A = \{ \langle x, w \rangle \mid w$ is a description a sequence of $\leq n^k$ valid transitions that makes N accept $x \}$
hence $x \in L$ iff $|w| \leq O(t(|x|))$ witness exists

$\Leftarrow$: $\exists$ poly time verifier for L
$t(n)$
N = "on input x: - guess w w/ $|w| \leq t(n)$
- accept iff V accepts $\langle x, w \rangle$"

Prop. $t(n): \mathbb{N} \to \mathbb{N}$, $\text{TIME}(t(n)) \leq \text{NTIME}(t(n)) \subseteq \text{TIME}(2^{O(t(n))})$
$\uparrow$ easy
[both very pessimistic]

Michael Forbes
~~2314~~
miforbes@illinois.edu

2019-01-22.2 → 2019-01-22.3
2019-01-224 ← CS574

pf (verifier vers): $\forall t(n)$-time verifier for L

M = "on input x:
$|\Sigma|^{O(t(n))} = 2^{O(t)}$ 1) Run all $w \in \Sigma^{\leq t(n)}$
$t(n)$ - check if V accepts $\langle x, w\rangle$
2) accept if V ever accepts

← Questions?

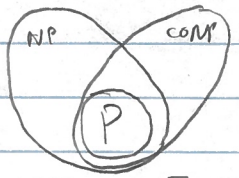
Q: $\overline{\text{HAMPATH}} := \Sigma^* \setminus \text{HAMPATH} \in NP$? open

Sl: $\{\langle G, s, t\rangle : G$ has no $s \rightsquigarrow t$ hamiltonian path$\}$
↳ up to (magic integer encoding)
[not clear how to verify no hampath]

def: $coNP = \{\overline{L} : L \in NP\}$

Rmk: $coP = P$ [switch acc/rej]

the world.



Conj: $P \neq NP$ [easier to recognize/verify than to create]
difficulty: many clever algorithms! [eg music, poets] [have to show that none work]

eg PRIMES = $\{\langle n\rangle : n$ is prime integer$\}$
↳ binary encoding

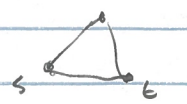
n prime iff no nontrivial factorization $n = a \cdot b$ $a, b < n$
n not prime iff exists "
$\Rightarrow \overline{\text{PRIMES}} \in coNP$, N = "on input $\langle n\rangle$: - guess $a, b < n$
$\Rightarrow$ PRIMES $\in coNP$ [in P?] - accept if $n = a \cdot b$ "

Thm [Pratt 75]: PRIMES $\in NP$ [uses some basic number theory]
Thm [Agrawal Kayal Saxena 02]: PRIMES $\in P$ [undergrad] [took entire] [had evidence of long before] [even more elementary number theory]

Questions:

def. undirected graph $G = (V, E)$. a $s \rightsquigarrow t$ path is Eulerian if all edges are use exactly once. [can repeat vertices]

eg:



no Eulerian path

is Eulerian

Michael Fahey
mjfahey@illinois.edu
2019-01-22.4 ← 2019-01-22.3
→ 2019-01-24.1
cs579

[can check connectedness in P]

G connected, $s \neq t$

EULER-PATH = $\{\langle G, s, t\rangle : G$ undirected graph w/ Eulerian $s \rightsquigarrow t$ path$\}$

Prop. $\in NP$ [easy] [vs HAMPATH]

Prop: G has $s \rightsquigarrow t$ Eulerian path iff G is connected

- deg s, deg t odd — deg v = # edges incident to v
- $v \notin \{s,t\} \Rightarrow$ deg v even

] check in P

Cor. EULER-PATH $\in P$

Pf. $\Rightarrow$: $p: s = v_0 \to v_1 \to v_2 \to \cdots \to v_{m-1} \to v_m = t$

adds 1 to deg s; adds 2 to deg v_1; adds 1 to deg t

$\deg_p v = \begin{cases} 2(\#\text{ occur at } v \text{ in } p) & v \neq s,t \quad [\text{even}] \\ 2(\#\text{ occur}) - 1 & v = s,t \quad [\text{odd}] \end{cases}$

p is Eulerian: $\deg_p v = \deg v$

$\Leftarrow$: not hard [but out of time]

[gurushlim: that there are many algs, want evidence problem is hard]

next time: NP-completeness