Robert Andrew
rgandre2@ill
2018-01-24.5 → 2018-01-2
2018-01-29.2 ← cs57

cs579: Computational Complexity: Lecture 4
admin: Michael out of town
Extra OH - T2, R1

today: NP-intermediate
Space

Q: How to make problems easier?
A: Longer input.

Lemma (Padding):
Given L ← language
f ← time-constructible
Let $L_{pad} = \{x\#^{f(|x|)-|x|} : x \in L\}$.

$L_{pad} \in TIME(T(n)) \Leftrightarrow L \in TIME(T(f(n)))$

Pf: ⇒ Given x
Pad to $x\#^{f(|x|)-|x|}$
Run algo for L_{pad} on
] both take $O(T(f(|x|)))$ time

⇐ Given y.
Check if $y = x\#^{f(|x|)-|x|}$ for some x
If no, reject
O.w., run algo for L on x.
running time: $O(|y|)$ for syntax check
$O(T(f(|x|)))$ for L algo
overall $T(|y|)$ time □

Q: If $P \neq NP$, does $NP \setminus P = NP$-complete?
A: No. [Ladner's thm]
[ETH]
Exponential Time Hypothesis: 3SAT requires $2^{\Omega(n)}$ time

ert Andrews
andre2@illinois
-01-29.1 → 2018-01-29.2
01-29.3 ← CS579

Rmk: ETH ⇒ P ≠ NP

Thm [Baby Ladner]: ETH ⇒ NP \ P ≠ NP-complete.

Idea: Pad 3SAT with $f(n) = \omega(\text{poly}(n))$ and $2^{o(n)}$.

Pf: $3SAT_{pad}$ = 3SAT padded with $f(n) = n^{\log n}$.

$3SAT \in TIME(2^n) \Rightarrow 3SAT_{pad} \in TIME\left(2^{2^{\sqrt{\log n}}}\right)$.

$3SAT_{pad}$ NP-complete $\Rightarrow 3SAT \leq_p 3SAT_{pad} \Rightarrow 3SAT \in TIME\left(2^{2^{\sqrt{\log p(n)}}}\right)$ (reduction run-time)

$$2^{\sqrt{\log p(n)}} = 2^{O(\sqrt{\log n})} = 2^{o(\log n)} = o(n)$$

$\Rightarrow 3SAT \in TIME(2^{o(n)}) \Rightarrow\Leftarrow ETH$.

$3SAT_{pad} \in P$: $3SAT \in TIME(\text{poly}(n^{\log n})) \subseteq TIME(2^{O(n)}) \Rightarrow\Leftarrow ETH$ □.

Rmk NP-intermediate langs are constructible

Full Ladner computes padding via diagonalization.

Same proof works if SAT requires $n^{f(n)}$, $f(n) = \omega(1)$, time-constructible.

Space complexity

Q: Time vs space as resources - can reuse space

Thm [Savitch]: $NSPACE(S(n)) \subseteq SPACE(S(n)^2)$.

$S(n) \geq \log n$, space-constructible. (will prove $S(n) \geq n$ today)

Def: A configuration of M on x is a string describing
- state of M
- contents of work tape
- head position

Assume wlog one accept config (erase tape & move head all the way left)

Robert Andrejko
rgandre2@illino
2018-01-29.2 → 2018-01-2
2018-01-29.4 ← CS57

Pf [Switch]

$L \in NSPACE(S(n))$, NTM N. $L(N) = L$.

CanReach(c_1, c_2, t): [reach c_2 from c_1 in t steps].
If $t=0$ & $c_1 \neq c_2$, reject.
If $t=1$, check if $c_1 \to c_2$ in 1 step.
Else for all config c_3 using $S(n)$ space:
If CanReach$(c_1, c_3, t/2)$
and CanReach$(c_3, c_2, t/2)$, accept
Reject

On input x, run CanReach$(c_{start}, c_{accept}, 2^{O(S(n))})$
Space usage: each level uses $O(S(n))$ space
$\log(2^{O(S(n))}) = O(S(n))$ levels
$\Rightarrow O(S(n)^2)$ space $\square$.

Cor: PSPACE = NPSPACE.
Rmk: - Switch is a pathfinding algo for config graph
- Space-constructibility not necessary.

Q: Hard problems for space?

Def: TQBF $= \{\langle\varphi\rangle : \varphi$ is true, fully quantified Boolean formula$\}$
ex. $\varphi_1 = \forall x \exists y [(x \wedge y) \vee (\neg x \vee \neg y)]$. $\varphi_1 \in$ TQBF
$\varphi_2 = \exists x \forall y$ ——— // ——— $\varphi_2 \notin$ //

Thm: TQBF is PSPACE-complete.
Pf [$\in$ PSPACE).
Assume prenex form (quantifiers first).
IsTrue(φ):
If $\forall$ quantifier, return IsTrue$(\varphi(0,\ldots)) \wedge$ IsTrue$(\varphi(1,\ldots))$
$\exists$ ——— // ——— $\vee$ ——— //
no quantifier, evaluate φ

(PSPACE-hard)

$L \in$ PSPACE, TM M, $L(M) = L$.
(runs in $O(n^k)$ space)

Given x, want $\varphi_{c_1, c_2, t}$ = "M goes from c_1 to c_2 in $\leq t$ steps"

Config $c \mapsto$ Boolean vars

$\varphi_{c_1, c_2, 1} = (c_1 = c_2) \vee (c_1 \to c_2)$: on details next time

$\varphi_{c_1, c_2, t} = \exists c_3 \, \forall (a, b) \in \{(c_1, c_3), (c_3, c_2)\} \, [\varphi_{a, b, t/2}]$.

Final formula: $\varphi_{c_{start}, c_{accept}, 2^{O(n^k)}}$.

Each level adds $O(n^k)$ to φ
$\log(2^{O(n^k)}) = O(n^k)$ levels
$\Rightarrow$ final formula has length $O(n^{2k})$.

$L \leq_p$ TQBF, so TQBF is PSPACE-hard
$\Rightarrow$ -complete □.

next time: - Small space
- Circuits

admin: no Michael this week.