Robert Andrews
rgandre2@illinois
2018-01-29.4 → 2018-01-31
2018-01-31.2 ←
cs579

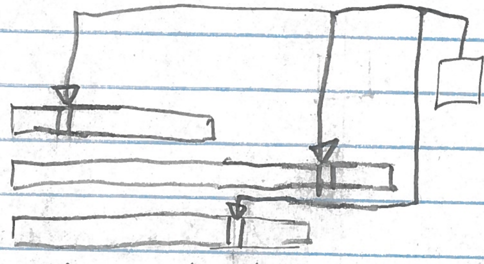
CS 579: Computational Complexity: Lecture 5

admin: Michael next week
pset 1 due Monday via email

today: - Logspace
- Circuits

Q: How to model $o(n)$ space? [big data, etc.].
A: Only pay for extra space

Def: 3-tape Turing machine
- Read-only input tape
- Read/write work tape
- Write-only output tape



M uses space $S(n)$ if $\leq S(n)$ work-tape cells used on input of length n.

$\mathrm{SPACE}(S(n)) = \{L : \exists M$ deciding L in $S(n)$ space$\}$.
$\mathrm{NSPACE}(\sim) = \{$ non-deterministic $\}$

$L = \mathrm{SPACE}(O(\log n))$ [$O(1)$ pointers to input].
$NL = \mathrm{NSPACE}(O(\log n))$.

Def: $\mathrm{PATH} = \{\langle G, s, t\rangle : G$ is a directed graph, $\exists s \to t$ path in $G\}$.

Thm: PATH is NL-complete.
($\mathrm{PATH} \in NL$).

$v \leftarrow s$
For $i \leftarrow 1$ to n:
  Guess $u \in N(v)$.
  Verify $(v \to u) \in E(G)$
  If $u = t$, accept
Reject

Space: $\log n$ to store counter
$2 \log n$ to keep vertices
$= O(\log n)$

$\Rightarrow \mathrm{PATH} \in NL$.

[R]obert Andrews
[r]gandre2@illinois
2018-01-31.1 → 2018-01-31.2
2018-01-31.3 ← cs579

For hardness, need new reduction
Def: $L_1 \le_{log} L_2$ ["L_1 reduces to L_2 in logspace"]
if ∃ logspace computable f such that
$x \in L_1 \Leftrightarrow f(x) \in L_2$. [f does not pay to write output]
Prop. If $A \le_{log} B$ and $B \in L$, then $A \in L$.

[P]ATH NL-hard

Given $L = L(N)$ for NTM running in space $s(n) = k \log n$
compute config graph G and output $\langle G, c_{start}, c_{accept} \rangle$
Each config requires
- input head position → $\log n$ bits
- work tape contents → $k \log n$ bits
- " " head position → $\log(k \log n) = O(\log n)$ bits

⇒ enumerate configs in logspace
Given config c_1, c_2, can check if $c_1 \to c_2$
by applying transitions of N
⇒ enumerate edges in logspace
⇒ compute $\langle G, c_{start}, c_{accept} \rangle$ in logspace
⇒ $L(N) \le_{log}$ PATH
⇒ PATH is NL-hard ⇒ NL-complete

Rmk - PATH has $O(n)$-time algo - BFS/DFS.
- $L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP$

Savitch ⇒ $NL \subseteq SPACE(\log^2 n) \subsetneq PSPACE = NPSPACE$
Time hierarchy ⇒ $P \subsetneq EXP$
- Certificate def of NL: read-once poly-size cert

Def: $coNL = \{L : \bar{L} \in NL\}$ (lang)
Q: NL vs coNL?
Thm [Immerman-Szelepcsényi]: NL = coNL.
Pf idea: show $\overline{PATH} \in NL$.
$C_i = \{v \in G : d(s,v) \le i\}$
Given $|C_i|$, compute $|C_{i+1}|$
Given $|C_{n-1}|$, compute $t \overset{?}{\in} C_{n-1}$.

Robert Andrews
rgandre2@illinois
2018-01-31.2 → 2018-01-31.
2018-01-31.4 ←
cs579

Pf

Step 1: Given $|C_i|$, compute $|C_{i+1}|$:

$|C_{i+1}| \leftarrow 0$
For $v \in V$:
  count ← 0
  reached? ← false
  For $u \in V$:
    Guess if $u \in C_i$
    If yes, guess & verify path
      count ← count + 1
      If $u = v$ or $(u \to v) \in E(G)$,
        reached? ← true
  If count $\neq |C_i|$, reject.
  If reached?, $|C_{i+1}| \leftarrow |C_{i+1}| + 1$
Output $|C_{i+1}|$

return $\leq |C_{i+1}|$: cannot overguess vertices
$\geq |C_{i+1}|$: must g... every vtx in C...

Step 2: Given $|C_{n-1}|$, compute $t \overset{?}{\in} C_{n-1}$

count ← 0
For $v \in V$:
  Guess if $v \in C_{n-1}$
  If yes, guess & verify path
  count ← count + 1
If count $\neq |C_{n-1}|$, reject
If t reached, reject
Else, accept.

Both algorithms keep $O(1)$ vertices on work tape, so they use $\leq O(\log n)$ space.

□

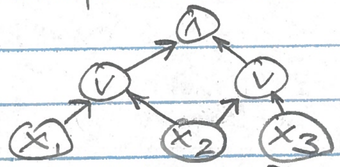
Rmk – Implies NSPACE hierarchy (can negate in NSPACE)
– Surprising: NP vs coNP major open problem

Robert Andrews
rgandre2 @ illinois
2018-01-31.3 → 2018-01-31.4
2018-02-02.1 ← cs579

Q: Case analysis as a resource?

Def: A Boolean circuit is a dag w/ vertices labeled as input, $\wedge$, $\vee$, or $\neg$.
- computes in natural way.

ex.



Def: $\text{Size}(C)$ = # of $\wedge$ and $\vee$ gates

Def: $f: \{0,1\}^* \to \{0,1\}$ computed by circuit family $\mathcal{C} = \{C_1, C_2, \ldots\}$ if $\forall x,\ f(x) = C_{|x|}(x)$.

$\text{Size}(f) = \text{Size}(C_n)$.

Def: $\text{SIZE}(s(n)) = \{f : \text{Size}(f) \le s(n)\}$

Def: $P/poly = \bigcup_{k>0} \text{SIZE}(n^k)$.

next time: more circuits

admin: - Michael back next week
- ps1 due Monday.