18.404/6.840 Lecture 17

Last time:

- Cook-Levin Theorem: SAT is NP-complete
- $3SAT$ is NP-complete

Today: (Sipser §8.1 – §8.2)

- Space complexity
- SPACE$\big(f(n)\big)$, NSPACE$\big(f(n)\big)$
- PSPACE, NPSPACE
- Relationship with TIME classes
- Examples

SPACE Complexity

Defn: Let $f\colon \mathbb{N} \to \mathbb{N}$ where $f(n) \geq n$. Say TM M runs in space $f(n)$ if M always halts and uses at most $f(n)$ tape cells on all inputs of length n.

Check-in 17.1

We define space complexity for multi-tape TMs by taking the sum of the cells used on all tapes.

Do we get the same class PSPACE for multi-tape TMs?

(a) No.

(b) Yes, converting a multi-tape TM to single-tape only squares the amount of space used.

(c) Yes, converting a multi-tape TM to single-tape only increases the amount of space used by a constant factor.

Check-in 17.1

Relationships between Time and SPACE Complexity

Theorem: For $t(n) \geq n$

1) $\text{TIME}\big(t(n)\big) \subseteq \text{SPACE}\big(t(n)\big)$
2) $\text{SPACE}\big(t(n)\big) \subseteq \text{TIME}\big(2^{O(t(n))}\big)$
   $= \bigcup_c \text{TIME}\big(c^{t(n)}\big)$

Proof:

1) A TM that runs in $t(n)$ steps cannot use more than $t(n)$ tape cells.
2) A TM that uses $t(n)$ tape cells cannot use more than $c^{t(n)}$ time without repeating a configuration and looping (for some c).

Corollary: P $\subseteq$ PSPACE

Theorem: NP $\subseteq$ PSPACE [next slide]

NP ⊆ PSPACE

Theorem: NP ⊆ PSPACE

Proof:

1. $SAT \in$ PSPACE
2. If $A \leq_{\mathrm{P}} B$ and $B \in$ PSPACE then $A \in$ PSPACE

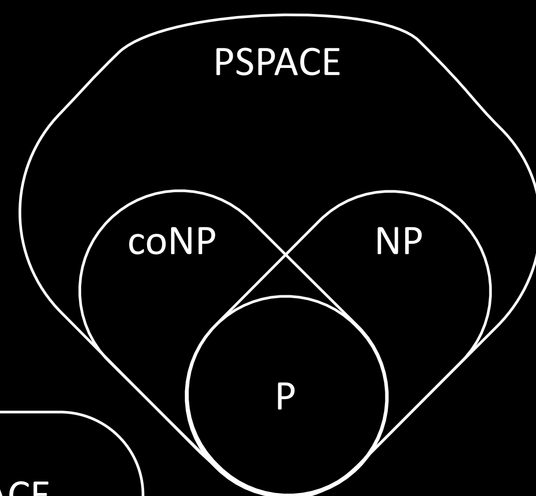
Defn: coNP $= \{\overline{A} \mid A \in \mathrm{NP}\}$

$\overline{HAMPATH} \in$ coNP

$TAUTOLOGY = \{\langle\phi\rangle \mid$ all assignments satisfy $\phi\} \in$ coNP

coNP ⊆ PSPACE (because PSPACE = coPSPACE)

P = PSPACE ? *Not known.*



Or possibly:

P = NP = coNP = PSPACE

Example: $TQBF$

Defn: A quantified Boolean formula (QBF) is a Boolean formula with leading exists ($\exists x$) and for all ($\forall x$) quantifiers. All variables must lie within the scope of a quantifier.

A QBF is TRUE or FALSE.

Examples: $\phi_1 = \forall x\, \exists y\, [(x \vee y) \wedge (\overline{x} \vee \overline{y})]$
$\phi_2 = \exists y\, \forall x\, [(x \vee y) \wedge (\overline{x} \vee \overline{y})]$

Defn: $TQBF = \{\langle \phi \rangle \mid \phi$ is a QBF that is TRUE$\}$

Thus $\phi_1 \in TQBF$ and $\phi_2 \notin TQBF$.

Theorem: $TQBF \in$ PSPACE

Check-in 17.2

How is SAT a special case of $TQBF$?

(a) Remove all quantifiers.

(b) Add $\exists$ and $\forall$ quantifiers.

(c) Add only $\exists$ quantifiers.

(d) Add only $\forall$ quantifiers.

Check-in 17.2

$TQBF \in$ PSPACE

Theorem: $TQBF \in$ PSPACE

Proof: "On input $\langle \phi \rangle$

1. If ϕ has no quantifiers, then ϕ has no variables so either $\phi =$ True or $\phi =$ False. Output accordingly.
2. If $\phi = \exists x\, \psi$ then evaluate ψ with $x =$ TRUE and $x =$ FALSE recursively. *Accept* if either accepts. *Reject* if not.
3. If $\phi = \forall x\, \psi$ then evaluate ψ with $x =$ TRUE and $x =$ FALSE recursively. *Accept* if both accept. *Reject* if not."

Space analysis:

Each recursive level uses constant space (to record the x value).

The recursion depth is the number of quantifiers, at most $n = |\langle \phi \rangle|$.

So $TQBF \in$ SPACE(n)

Example: Ladder Problem

A ladder is a sequence of strings of a common length where consecutive strings differ in a single symbol.

A word ladder for English is a ladder of English words.

Let A be a language. A ladder in A is a ladder of strings in A.

Defn: $LADDER_{\mathrm{DFA}} = \{\langle B, u, v\rangle |\ B$ is a DFA and $L(B)$ contains a ladder $y_1, y_2, \ldots, y_k$ where $y_1 = u$ and $y_k = v\}$.

Theorem: $LADDER_{\mathrm{DFA}} \in$ NPSPACE

WORK
PORK
PORT
SORT
SOOT
SLOT
PLOT
PLOY
PLAY

$LADDER_{\text{DFA}} \in$ NPSPACE

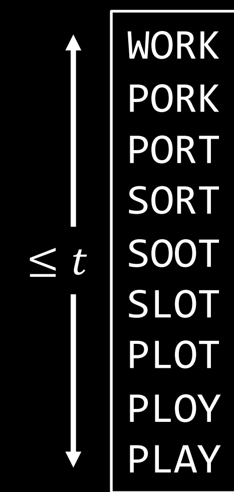
Theorem: $LADDER_{\text{DFA}} \in$ NPSPACE

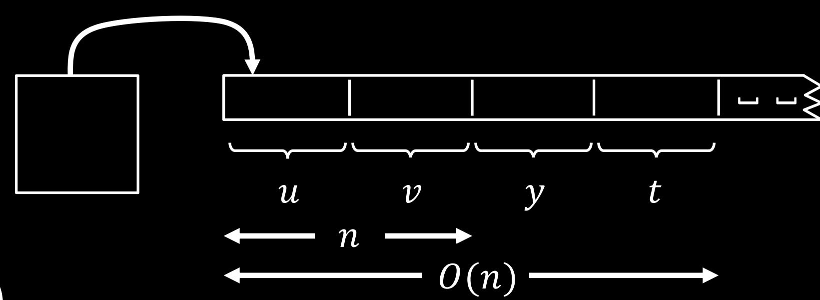
Proof idea: Nondeterministically guess the sequence from u to v.

Careful- (a) cannot store sequence, (b) must terminate.

Proof: "On input $\langle B, u, v\rangle$

1. Let $y = u$ and let $m = |u|$.
2. Repeat at most t times where $t = |\Sigma|^m$.
3. Nondeterministically change one symbol in y.
4. *Reject* if $y \notin L(B)$.
5. *Accept* if $y = v$.
6. *Reject* [exceeded t steps].





Space used is for storing y and t.

$LADDER_{\text{DFA}} \in \text{NSPACE}(n)$.

Theorem: $LADDER_{\text{DFA}} \in$ PSPACE (!)

$LADDER_{\text{DFA}} \in$ PSPACE

Theorem: $LADDER_{\text{DFA}} \in$ SPACE(n^2)

Proof: Write $u \xrightarrow{b} v$ if there's a ladder from u to v of length $\leq b$.

Here's a recursive procedure to solve the bounded DFA ladder problem:

$BOUNDED\text{-}LADDER_{\text{DFA}} = \{\langle B, u, v, b\rangle \mid B$ a DFA and $u \xrightarrow{b} v$ by a ladder in $L(B)\}$

$B\text{-}L =$ "On input $\langle B, u, v, b\rangle$ Let $m = |u| = |v|$.

1. For $b = 1$, *accept* if $u, v \in L(B)$ and differ in ≤ 1 place, else *reject*.
2. For $b > 1$, repeat for each w of length $|u|$
3. Recursively test $u \xrightarrow{b/2} w$ and $w \xrightarrow{b/2} v$ [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."

Test $\langle B, u, v\rangle \in LADDER_{\text{DFA}}$ with $B\text{-}L$ procedure on input $\langle B, u, v, t\rangle$ for $t = |\Sigma|^m$

Space analysis:

Each recursive level uses space $O(n)$ (to record w).

Recursion depth is $\log t = O(m) = O(n)$.

Total space used is $O(n^2)$.

Check-in 17.3

Find an English word ladder connecting MUST and VOTE.

(a) Already did it.

(b) I will.

Check-in 17.3

Quick review of today

1. Space complexity
2. SPACE$\left(f(n)\right)$, NSPACE$\left(f(n)\right)$
3. PSPACE, NPSPACE
4. Relationship with TIME classes
5. $TQBF$ ∈ PSPACE
6. $LADDER_{\text{DFA}} \in$ NSPACE(n)
7. $LADDER_{\text{DFA}} \in$ SPACE(n^2)