香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

SCHOOL OF DATA SCIENCE
數據科學學院

CSC 6011 - Theory of Computation

Lecture 15 - Space Complexity, PSPACE

Time: Mon & Wed, 4:00 - 5:20 PM

Location: Teaching A, 107

Instructor: Tao LIN (林涛)

TA: Yefan GAO (高叶繁)

Fall 2026

Teaching plan

Lecture	Topic
	Computability Theory
1–2	Introduction. Finite automata (DFA and NFA). Regular languages.
3–4	Nonregular languages, pumping lemma. Context-free languages, pushdown automata.
5–6	Turing machines, Church-Turing thesis.
7–8	Decidability, diagonalization.
9	Reduction.
	Complexity Theory - Part 1
10	Time complexity.
11–12	P and NP, NP-completeness.
13	Cook-Levin theorem.
14	Co-NP. Optimization problems.

Midterm

Lecture	Topic
	Complexity Theory - Part 2
Current 15–16	Space complexity, PSPACE, Savitch's theorem.
17	**[Midterm Exam] (Nov 9).**
18	L, NL. More on space complexity.
	Other Topics
19–20	[ICLR'26 paper "Transformers are Inherently Succinct"] & EXPSPACE-completeness.
21–22	Randomized computation, RP, BPP.
23–24	Interactive proofs, IP=PSPACE.
25–26	Approximation algorithms, hardness of approximation, PCP theorem.
27–28	Communication complexity.

Poly-Time Mapping vs Turing Reductions

Mapping reduction $A \leq_m^P B$

Compute $f(x)$ in poly time, with $x \in A \Leftrightarrow f(x) \in B$.

Can only call the solver for B once and return its answer unchanged (no reversal).

Turing reduction $A \leq_T^P B$ **(Lecture 14 & Homework 3)**

Can call the solver for B multiple times and reverse YES/NO answers.

Complexity notions depend on the type of reductions:

- Under $\leq_T^P$, NP = co-NP.
  - To solve A, query the solver for $\overline{A}$, and reverse the answer.
  - Thus, "optimization problem is NP-hard and co-NP-hard" = "optimization problem is NP-hard."
- Under $\leq_m^P$ (by default), whether NP = co-NP remains open.

Space Complexity

Defn: Let $f: \mathbb{N} \to \mathbb{N}$ where $f(n) \geq n$.

- A (deterministic) 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.
- An NTM M runs in space $f(n)$ if all branches halt and each branch uses at most $f(n)$ tape cells on all inputs of length n.

Defn:

- $\mathrm{SPACE}(f(n)) = \{ B \mid$ some 1-tape TM M decides B and M runs in space $O(f(n)) \}$
- $\mathrm{NSPACE}(f(n)) = \{ B \mid$ some 1-tape NTM M decides B and M runs in space $O(f(n)) \}$

$\mathrm{PSPACE} = \bigcup_k \mathrm{SPACE}(n^k)$ "polynomial space"

$\mathrm{NPSPACE} = \bigcup_k \mathrm{NSPACE}(n^k)$ "nondeterministic polynomial space"

Space Complexity

Defn: Let $f: \mathbb{N} \to \mathbb{N}$ where $f(n) \geq n$.

- A (deterministic) 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.
- An NTM M runs in space $f(n)$ if all branches halt and each branch uses at most $f(n)$ tape cells on all inputs of length n.

Defn:

- $\mathrm{SPACE}(f(n)) = \{ B \mid$ some 1-tape T
- $\mathrm{NSPACE}(f(n)) = \{ B \mid$ some 1-tape

PSPACE $= \bigcup_k \mathrm{SPACE}(n^k)$ "polynomi

NPSPACE $= \bigcup_k \mathrm{NSPACE}(n^k)$ "nond

Mini-Quiz 15.1

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.

Relationships between Time and Space Complexity

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

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

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 and more than $c^{t(n)}$ time must repeat some configuration and loop (for $c = |\Gamma \cup Q|$), but we assumed halting.

Corollary: P $\subseteq$ PSPACE

Proof: $\mathrm{P} = \cup_k \mathrm{TIME}\left(n^k\right) \subseteq \cup_k \mathrm{SPACE}\left(n^k\right) = \mathrm{PSPACE}$

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

NP ⊆ PSPACE

Theorem: NP ⊆ PSPACE

Proof:

1) $\mathrm{SAT} \in$ PSPACE
2) If $A \leq_m^P B$ and $B \in$ PSPACE, then $A \in$ PSPACE

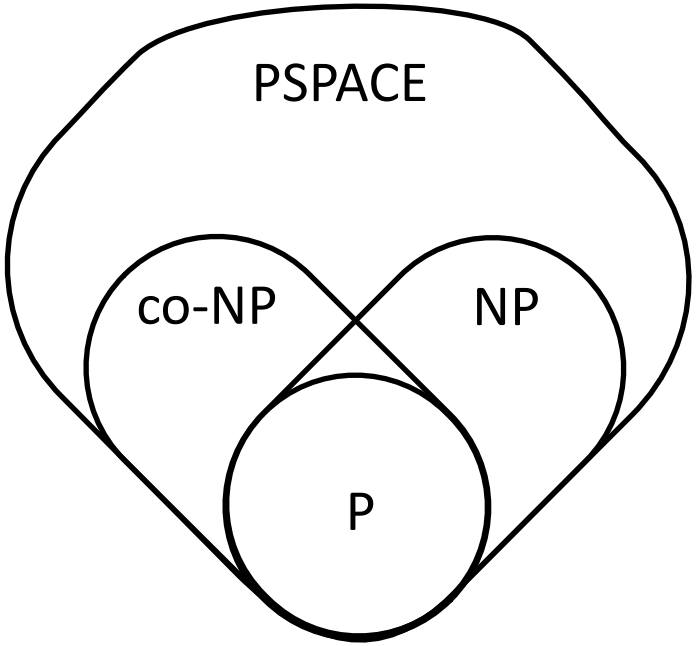
Recall: co-NP $= \{ \overline{A} \mid A \in \text{NP} \}$

Ex: $\overline{\mathrm{HAMPATH}}$, UNSAT, $\mathrm{TAUTOLOGY}$

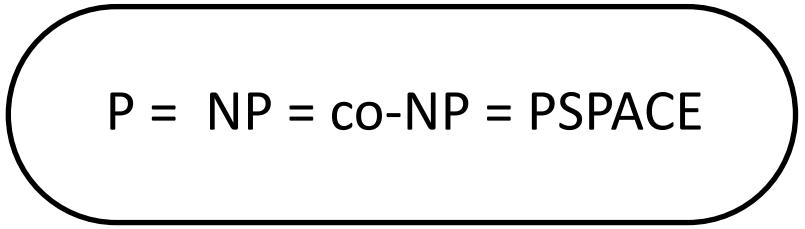
Corollary: co-NP ⊆ PSPACE

Proof: PSPACE = co-PSPACE

P = PSPACE ? *Not known.*



Or possibly:



Example: TQBF

"exists" "for all"

Defn: A quantified Boolean formula (QBF) is a Boolean formula with $\exists$ and $\forall$ quantifiers.

Every variable must be in the scope of a quantifier. A QBF has a fixed truth value: TRUE or FALSE.

Examples:

General form: $\phi = \forall x\,[x \vee (\exists y\,(\overline{x} \wedge y))]$ TRUE

Prenex form: all quantifiers appear at the front, followed by a quantifier-free formula.

$\phi' = \forall x\,\exists y\,[x \vee (\overline{x} \wedge y)]$ TRUE

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

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

Non-QBF example: $\forall y\,(x \vee y)$ value depends on "free variable" x

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

Thus ϕ, ϕ', $\phi_1 \in \mathrm{TQBF}$ and $\phi_2 \notin \mathrm{TQBF}$.

Theorem: $\mathrm{TQBF} \in$ PSPACE

Example: TQBF

"exists" "for all"

Defn: A quantified Boolean formula (QBF) is a Boolean formula with ∃ and ∀ quantifiers.
Every variable must be in the scope of a quantifier. A QBF has a fixed truth value: TRUE or FALSE.

Examples:

General form: $\phi = \forall x\,[x \vee (\exists y\,(\overline{x} \wedge y))]$ TRUE

Prenex form: all quantifiers appear at the front, followed by a quantifier-free formula.

$\phi' = \forall x\,\exists y\,[x \vee (\overline{x} \wedge y)]$ TRUE

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

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

Non-QBF example: $\forall y\,(x \vee y)$ value depends on "free varia

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

Thus ϕ, ϕ', $\phi_1 \in \mathrm{TQBF}$ and $\phi_2 \notin \mathrm{TQBF}$.

Theorem: $\mathrm{TQBF} \in$ PSPACE

Mini-Quiz 15.2

How is SAT a special case of TQBF?

(a) Remove all quantifiers.

(b) Add ∃ and ∀ quantifiers.

(c) Add only ∃ quantifiers.

(d) Add only ∀ quantifiers.

TQBF ∈ PSPACE

Theorem: $\mathrm{TQBF} \in \mathrm{PSPACE}$

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 $O(n)$ space (to write down the ψ with the assigned x value).

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

So $\mathrm{TQBF} \in \mathrm{SPACE}(n^2)$ (In fact, $\mathrm{TQBF} \in \mathrm{SPACE}(n)$ by modifying the formula in place).

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: $\text{LADDER}_{\text{DFA}} = \{\langle B, u, v\rangle \mid B$ 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: $\text{LADDER}_{\text{DFA}} \in$ NPSPACE

WORK
PORK
PORT
SORT
SOOT
SLOT
PLOT
PLOY
PLAY

$\text{LADDER}_{\text{DFA}} \in$ NPSPACE

Theorem: $\text{LADDER}_{\text{DFA}} \in$ NPSPACE

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

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

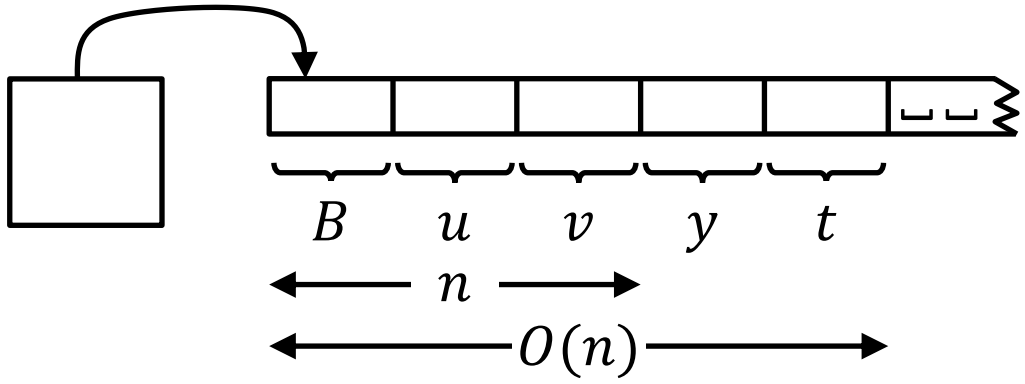
Proof: "On input $\langle B, u, v \rangle$

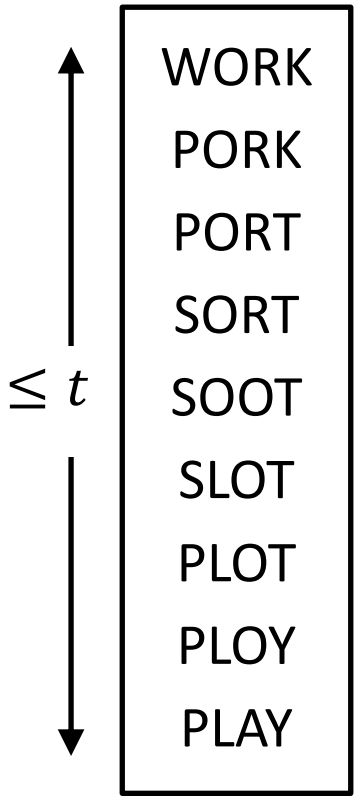
1. Let $y = u$ and let $m = |u|$.
2. Repeat at most $t = |\Sigma|^m$ times (number of all strings of length m).
3. Nondeterministically change one symbol in y.
4. *Reject* if $y \notin L(B)$.
5. *Accept* if $y = v$.
6. *Reject* [exceeded t steps].

Space used is for storing y and t.

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

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





$\text{LADDER}_{\text{DFA}} \in \text{PSPACE}$

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

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

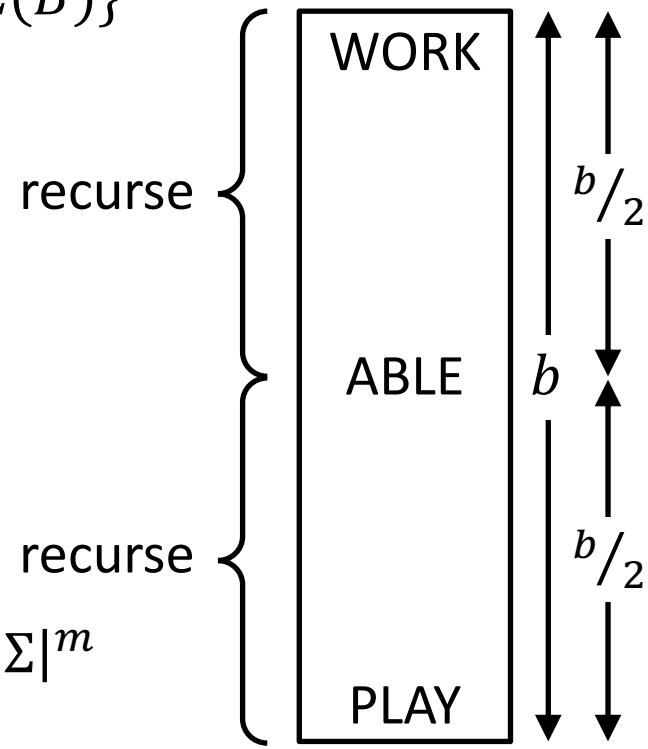
Here's a recursive procedure to solve the bounded DFA ladder problem:

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

B-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* if both accept.
5. *Reject* [if all fail]."

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



$\text{LADDER}_{\text{DFA}} \in \text{PSPACE}$

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

Mini-Quiz 15.3

Find an English word ladder from HEAD to FEET.

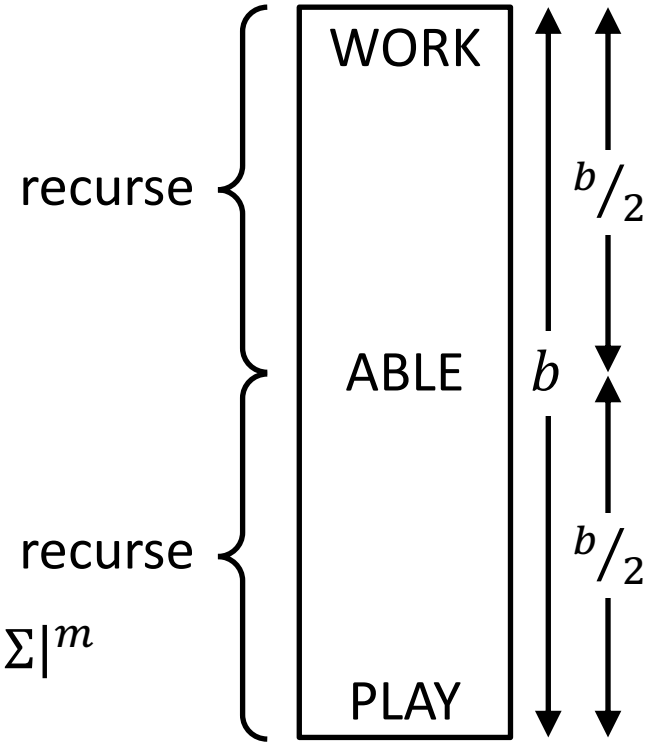
Find a Chinese word ladder from 火山 to 海洋.

Space analysis:

Each recursive level uses space $O(n)$ (to record u, v, b, w).

Recursion depth is $\log_2 t = O(m) = O(n)$.

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



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

Quick review of today

1. Space complexity
2. $\mathrm{SPACE}\big(f(n)\big), \mathrm{NSPACE}\big(f(n)\big)$
3. PSPACE, NPSPACE
4. Relationship with TIME classes
5. $\mathrm{TQBF} \in$ PSPACE
6. $\mathrm{LADDER}_{\mathrm{DFA}} \in \mathrm{NSPACE}(n)$
7. $\mathrm{LADDER}_{\mathrm{DFA}} \in \mathrm{SPACE}(n^2)$

These slides are adapted from Michael Sipser, *18.404J Theory of Computation*, Fall 2020. Massachusetts Institute of Technology:

- Source: MIT OpenCourseWare https://ocw.mit.edu

Except for separately identified third-party material, this adapted presentation is licensed under CC BY-NC-SA 4.0.