18.404/6.840 Lecture 18

Last time:

- Space complexity
- SPACENSPACE, PSPACE, NPSPACE
- Relationship with TIME classes

Today:

- Review   PSPACE
- Savitch's Theorem: NSPACESPACE
- PSPACE-completeness
- is PSPACE-complete

shrink me →

Posted: Pset 4 solutions, Pset 5

Review: SPACE Complexity

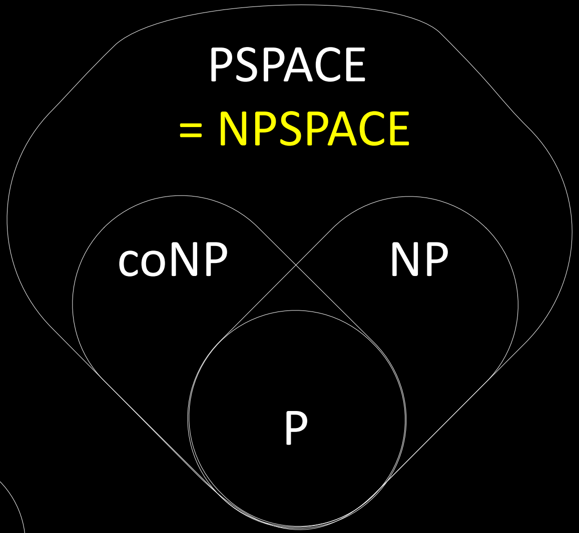
Defn: Let where . Say TM <u>runs in space</u> if always halts and uses at most tape cells on all inputs of length .

An NTM <u>runs in space</u> if all branches halt and each branch uses at most tape cells on all inputs of length .

SPACEsome 1-tape TM decides in space
NSPACEsome 1-tape NTM decides in space

PSPACE “polynomial space”
NPSPACE “nondeterministic polynomial space”

Today: PSPACE = NPSPACE



Or possibly: P = NP = coNP = PSPACE

Review: PSPACE

Theorem: SPACE

Proof: Write if there's a ladder from to of length .

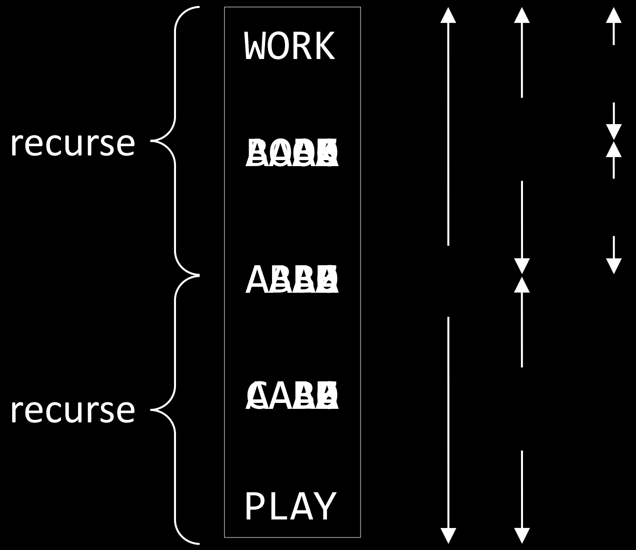
Here's a recursive procedure to solve the bounded DFA ladder problem:

- a DFA and by a ladder in

-"On input Let .

1. For , *accept* if and differ in place, else *reject*.
2. For , repeat for each of length
3. Recursively test and [division rounds up]
4. *Accept* both accept.
5. *Reject* [if all fail]."

Test with - procedure on input for



Space analysis:

Each recursive level uses space (to record).

Recursion depth is .

Total space used is .

PSPACE = NPSPACE

Savitch's Theorem: For , NSPACE SPACE

Proof: Convert NTM to equivalent TM , only squaring the space used.

For configurations and of , write if can get from to in steps.

Give recursive algorithm to test :

"On input [goal is to check]

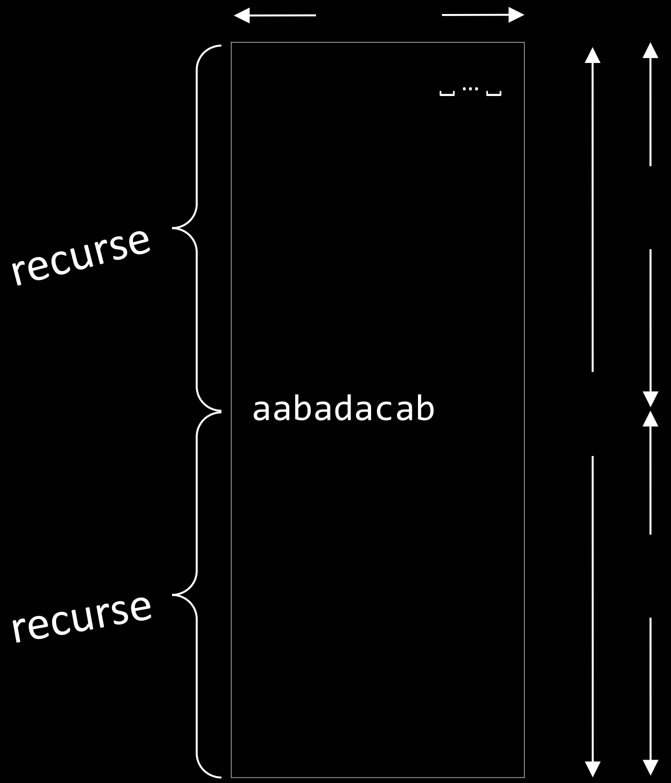
1. If , check directly by using 's program and answer accordingly.
2. If , repeat for all configurations that use space.
3. Recursively test and
4. If both are true, *accept*. If not, continue.
5. *Reject* if haven't yet accepted."

Test if accepts by testing where = number of configurations

=

Each recursion level stores 1 config = space.

Number of levels = . Total space.



PSPACE-completeness

Defn: is <u>PSPACE-complete</u> if
1) PSPACE
2) For all PSPACE,

If is PSPACE-complete and P then P PSPACE.

Check-in 18.1

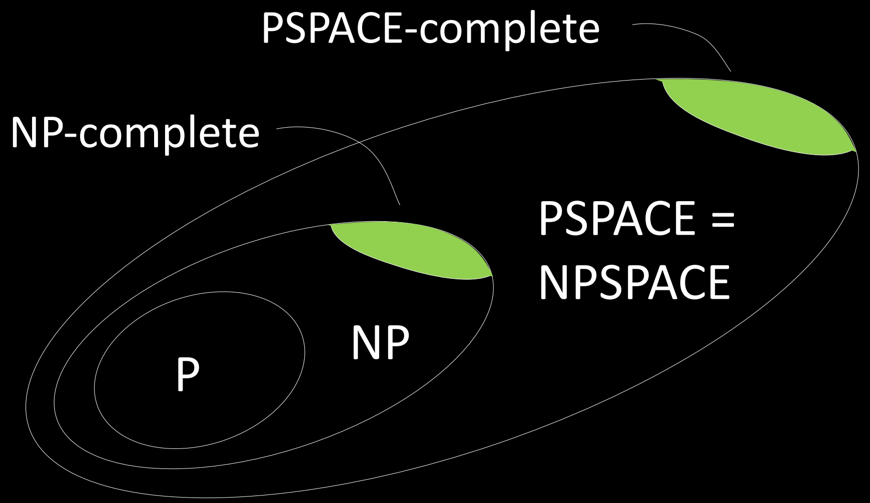
Knowing that is PSPACE-complete, what can we conclude if NP? Check all that apply.

(a) P = PSPACE

(b) NP = PSPACE

(c) P = NP

(d) NP = coNP



Think of complete problems as the “hardest” in their associated class.

Check-in 18.1

Coffee
Break

is PSPACE-complete

Recall: is a QBF that is TRUE

Examples: [TRUE]
[FALSE]

Theorem: is PSPACE-complete

Proof: 1) PSPACE ✓

2) For all PSPACE,

Let PSPACE be decided by TM in space .

Give a polynomial-time reduction mapping to .

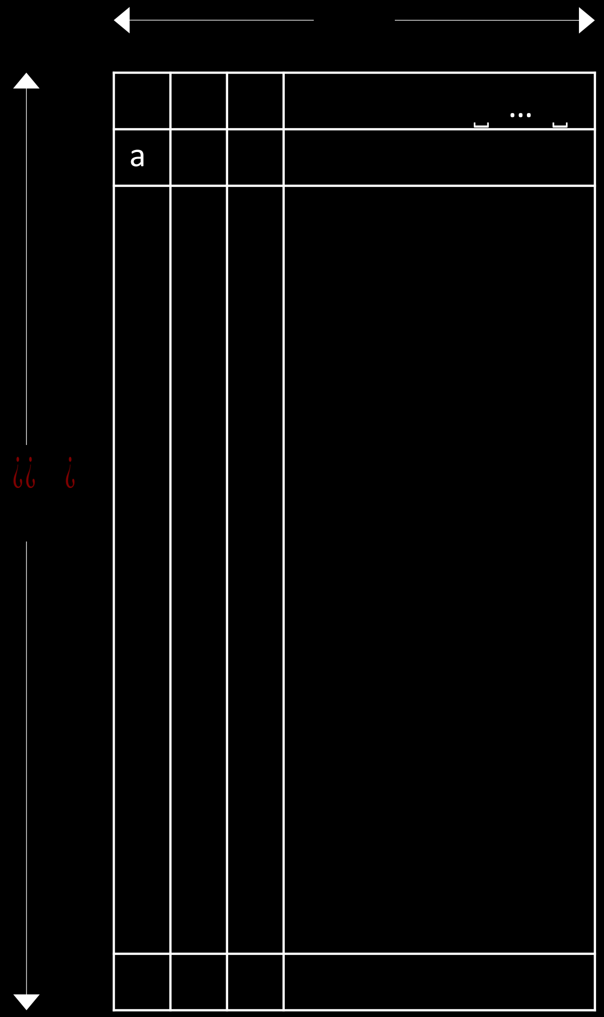
QBFs

iff is TRUE

Plan: Design to “say” accepts . simulates on .

Constructing : 1st try

Tableau for on



Recall: A tableau for on represents
a computation history for on
when accepts .
Rows of that tableau are configurations.

 runs in space , its tableau has:
- columns (max size of a configuration)
- rows (max number of steps)

Constructing . Try Cook-Levin method.
Then will be as big as tableau.
But that is exponential: .
Too big! ☹

Constructing : 2nd try

hide →

For configs and construct which "says" recursively.

$\underbrace{\qquad\qquad}$

as in Cook-Levin

$$\begin{tikzcd} & \cdot \arrow[dl] \arrow[d] & & \cdot \arrow[d] \arrow[dr] & \\ \cdot & \cdot & & \cdot & \cdot \end{tikzcd}$$

$$\begin{tikzcd} & \cdot \arrow[dl] \arrow[dr] & \\ \cdot & & \cdot \end{tikzcd}$$

defined as in Cook-Levin

Check-in 18.2

Why shouldn't we be surprised that this construction fails?

(a) We can't define a QBF by using recursion.

(b) It doesn't use anywhere.

(c) We know that P.

Size analysis:
Each recursive level doubles number of QBFs.
Number of levels is .
→ Size is exponential. ☹

Check-in 18.2

Constructing : 3rd try

is equivalent to

defined as in Cook-Levin

Size analysis:
Each recursive level <u>adds</u> to the QBF.
Number of levels is .
→ Size is ☺

Check-in 18.3
Would this construction still work if were nondeterministic?
(a) Yes.
(b) No.

Check-in 18.3

Quick review of today

1. PSPACE
2. Savitch's Theorem: NSPACESPACE
3. is PSPACE-complete