18.404/6.840 Lecture 17

Last time:

- Cook-Levin Theorem: is NP-complete
- is NP-complete

Today:

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

SPACE Complexity

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

ape

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

1) TIME SPACE
2) SPACE TIME

Proof:

1) A TM that runs in steps cannot use more than tape cells.
2) A TM that uses tape cells cannot use more than time
   without repeating a configuration and looping (for some).

Corollary: P PSPACE

Theorem: NP PSPACE [next slide]

NP PSPACE

Theorem: NP PSPACE

Proof:

1. PSPACE
2. If and PSPACE then PSPACE

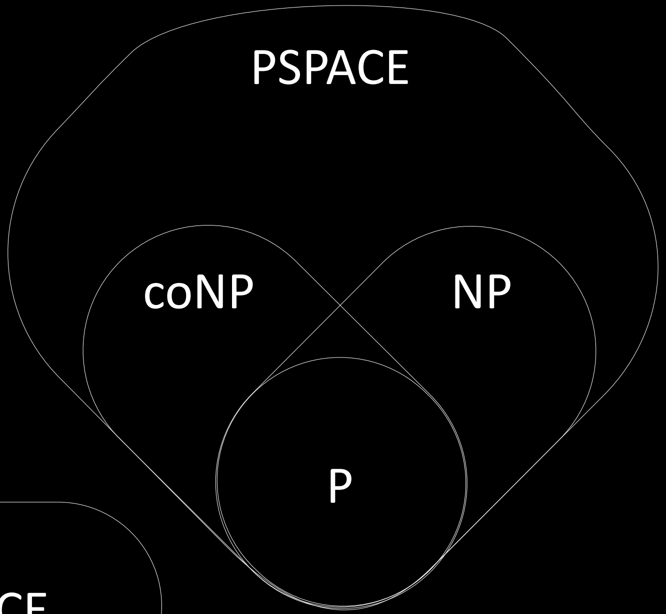
Defn: coNP NP}

coNP

all assignments satisfy

coNP PSPACE (because PSPACE = coPSPACE)

P = PSPACE ? *Not known.*



Or possibly:

P = NP = coNP = PSPACE

Example:

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

A QBF is TRUE or FALSE.

Examples: TRUE

FALSE

Defn: is a QBF that is TRUE

Thus and .

Theorem: PSPACE

Check-in 17.2

How is a special case of ?

(a) Remove all quantifiers.

(b) Add and quantifiers.

(c) Add only quantifiers.

(d) Add only quantifiers.

Check-in 17.2

PSPACE

Theorem: PSPACE

Proof: “On input

1. If has no quantifiers, then has no variables
   so either True or False. Output accordingly.
2. If then evaluate with TRUE and FALSE recursively.
   Accept if either accepts. *Reject* if not.
3. If then evaluate with TRUE and FALSE recursively.
   Accept if both accept. *Reject* if not.”

Space analysis:

Each recursive level uses constant space (to record the value).
The recursion depth is the number of quantifiers, at most .

So SPACE

Coffee
Break

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

Defn: is a DFA and contains a ladder where and }.

Theorem: NPSPACE

WORK
PORK
PORT
SORT
SOOT
SLOT
PLOT
PLOY
PLAY

NPSPACE

Theorem: NPSPACE

Proof idea: Nondeterministically guess the sequence from to .

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

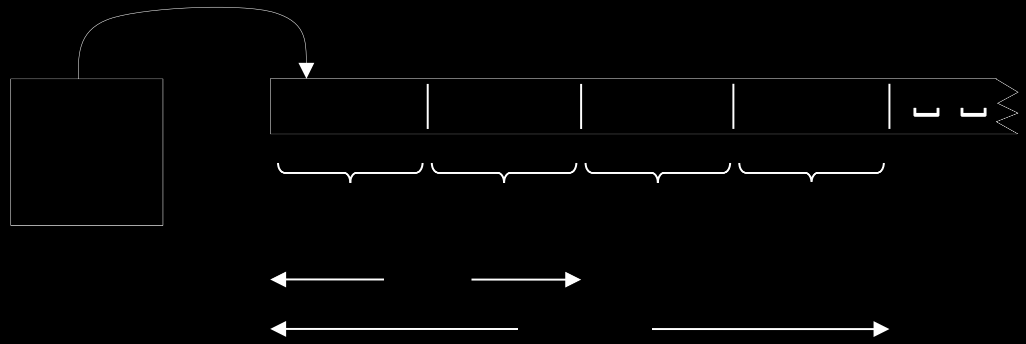
Proof: “On input

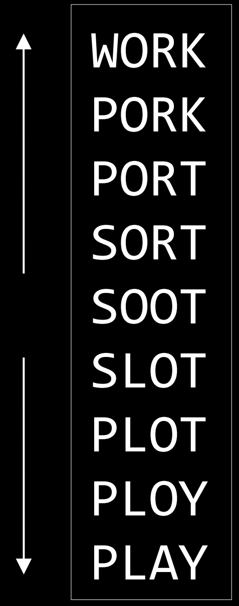
```
1. Let  and let .
2. Repeat at most  times where .
3.     Nondeterministically change one symbol in .
4.     Reject if .
5.     Accept if .
6. Reject  [exceeded  steps].
```

Space used is for storing and .

NSPACE.

Theorem: PSPACE (!)





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 .

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. SPACENSPACE
3. PSPACE, NPSPACE
4. Relationship with TIME classes
5. PSPACE
6. NSPACE
7. SPACE