COMP3630/6360: Theory of Computation
Semester 1, 2022
The Australian National University

Space Complexity

This lecture covers Chapter 11 of HMU: Other Complexity Classes

- The classes PS and NPS
- Relationship to other classes
- Savitch's Theorem
- Quantified Boolean Formulae
- PSpace completeness

Additional Reading: Chapter 11 of HMU.

Polynomial Space

Definition 10.1

A Turing machine M is *polyspace bounded* if there is a polynomial p so that M never uses more than $p(|w|)$ tape cells when started with input w.

Note.

- For deterministic machines, this refers to the unique computation path
- For non-deterministic machines, this refers to *all* computation paths starting with input w.

Definition 10.2

The class $\mathrm{PS} = \mathrm{PSPACE}$ is the class of languages L such that $L = L(M)$ for a polyspace bounded *deterministic* Turing machine.

The class $\mathrm{NPS} = \mathrm{NPSPACE}$ is the class of languages L such that $L = L(M)$ for a polyspace bounded *nondeterministic* Turing machine.

Example $\mathsf{ALL}_{\mathsf{NFA}}$

$$\mathsf{ALL}_{\mathsf{NFA}} = \{\{\langle A\rangle \mid A \text{ is an NFA and } L(A) = \Sigma^*\}$$

Currently, it's known neither whether $\mathsf{ALL}_{\mathsf{NFA}} \in NP$ nor whether $\mathsf{ALL}_{\mathsf{NFA}} \in co - NP$.

NPSPACE Algorithm for $\mathsf{ALL}^c_{\mathsf{NFA}}$

$M =$ "On input $\langle A\rangle$, where $A = (Q, \Sigma, \delta, q_0, F)$ is an NFA:

1. Place a marker on q_0. If $q_0 \notin F$, accept.
2. Repeat $2^{|Q|}$ times:
   1. Let $m \subseteq Q$ be the positions of markers.
   2. Pick any $a \in \Sigma$ and change m to $\bigcup_{q\in m} \delta(q, a)$.
   3. If $m \cap F = \emptyset$, *accept*.
3. *reject*."

- M may use exponential time but linear space only.
- $2^{|Q|}$ iterations are needed to ensure we can reach all possible patterns of markers on states.
- Hence $\mathsf{ALL}^c_{\mathsf{NFA}} \in \mathsf{NPSPACE}$.

Q. Why didn't we introduce coNSPACE?

Relationship to Other Classes

Easy Inclusions

- $P \subseteq PSPACE$ and $NP \subseteq NPSPACE$ (you cannot use more than polynomially many cells in polynomial time).

Unknown Inclusions

- We don't know whether P = PSPACE or NP = PSPACE.

Inclusions we will see

- PSPACE = NPSPACE
- this is remarkable, as we don't know whether P = NP.

Exponential Time

Definition 10.3

A deterministic or non-deterministic Turing machine runs in *exponential time* if it terminates in at most $c^{p(|w|)}$ steps for a constant c and polynomial p.

EXP is the class of languages L for which $L = L(M)$ for an exptime *deterministic* Turing machine.

NEXP is the class of languages L for which $L = L(M)$ for a *nondeterministic* exponential time Turing machine.

More Inclusions

$$P \subseteq NP \subseteq PSPACE \subseteq EXP$$

- $NP \subseteq PSPACE$ follows once we have shown that $NPSPACE = PSPACE$
- $PSPACE \subseteq EXP$ needs to be proved
- We know that $P \subsetneq EXP$, so that one inclusion is proper
- But we don't know which . . .

PSPACE vs EXP

Theorem 10.4

PSPACE $\subseteq$ *EXP*

Main Idea

A polyspace bounded machine only has $c^{p(n)}$ different ID's.

- count up to $c^{p(n)}$ – exponentially many steps
- must surely have repeated an ID by then

Proof PSPACE $\subseteq$ EXP

- Assume that a TM M only uses $p(n)$ space, and has semi-infinite tape.
- M has $s \times p(n) \times t^{p(n)}$ many ID's:
  - s are the states, $p(n)$ are the different head positions, $t^{p(n)}$ is tape contents.
- We have $(t+1)^{p(n)+1} \geq p(n)t^{p(n)}$
- ...and $s = (t+1)^c$ where $c = \log_{t+1} s$
- hence number of IDs is $\leq (t+1)^{p(n)+1+c}$.
- count ID's on a separate tape in base $t+1$ and $p(n)+c+1$ tape cells.
- have two-tape machine M' counting up to max IDs on extra tape
- halt if counter exceeds maximal value (M accepts if no IDs are repeated)
- converting to a single-tape machines gives polynomial blowup, hence exptime overall.

Savitch's Theorem: PSPACE=NPSPACE

Theorem 10.5

PSPACE=NPSPACE

- Let M be nondeterministic and polyspace bounded by $p(n)$
- M has $c^{p(n)}$ different IDs.
- Decide $P(I, J, K) = I \vdash^* J$ for IDs I and J in $\leq K$ steps
- This gives $w \in L(M)$ iff $I \vdash^* J$ for initial ID I and accepting ID J in at most $c^{p(n)}$ steps
- The bound on steps is valid, as M accepts iff M accepts without repeating IDs

Remains. Implement $P(I, J, K)$ taking polynomial space.

Recursive Doubling

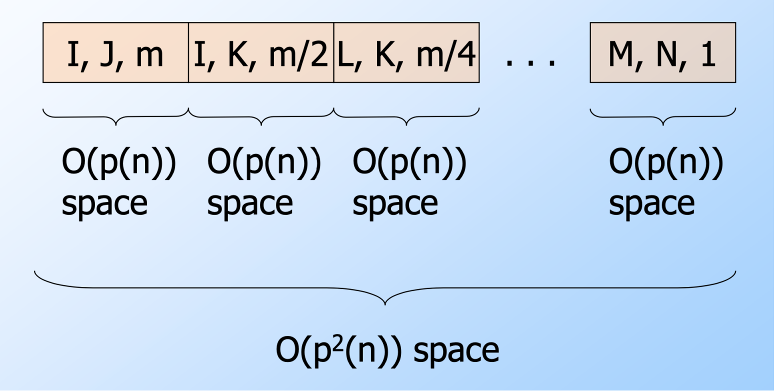
Goal. Implement $P(I, J, N) = I \vdash^{\leq m} J$ in deterministic polyspace

```
P(I, J, m): for all IDs K with length <= p(n) do {
  if P(I, K, m/2) and P(K, J, m/2) then return true
}
return false
```

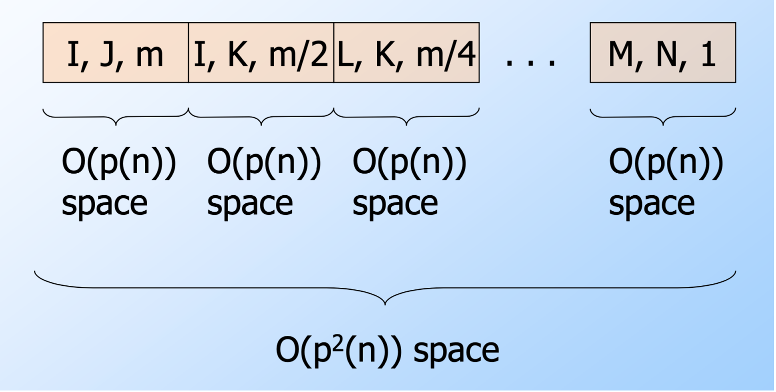
Q. How much space does this implementation need?

Recursive Doubling

```
P(I, J, m): for all IDs K with length <= p(n) do {
  if P(I, K, m/2) and P(K, J, m/2) then return true
}
return false
```



Recursive Doubling



- $m = c^{p(n)}$, so $\log c^{p(n)} = p(n)$ recursive calls till base case
- calls with argument m only induce one call with argument $m/2$
- only need to remember current ID for function return
- $\mathcal{O}(p^2(n))$ space in total