Space Complexity

Huan Long

Shanghai Jiao Tong University

Acknowledgements

Part of the slides comes from a similar course given by Prof. Yijia Chen.

http://basics.sjtu.edu.cn/~chen/

Textbook

Introduction to the theory of computation
Michael Sipser, MIT
Third edition, 2012

Outline

Space Complexity

Savitch's Theorem

The Class PSPACE

Space Complexity

Definition

Let M be a DTM that halts on all inputs. The space complexity of M is the function $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of tape cells that M scans on any input of length n. If the space complexity of M is $f(n)$, we also say that M runs in space $f(n)$.

If M is an NTM wherein all branches halt on all inputs, we define its space complexity $f(n)$ to be the maximum number of tape cells that M scans on any branch of its computation for any input of length n.

Definition

Let $f : \mathbb{N} \to \mathbb{R}^+$ be a function. The space complexity classes,

SPACE($f(n)$)=$\{L \mid L$ is a language decided by an $\mathcal{O}(f(n))$ space DTM$\}$

NSPACE($f(n)$)=$\{L \mid L$ is a language decided by an $\mathcal{O}(f(n))$ space NTM$\}$

SAT $\in$ SPACE(n)

M_1 on $\langle\varphi\rangle$, where φ is a Boolean formula:

1. For each truth assignment to the variable $x_1, \ldots, x_m$ of φ:
2. Evaluate φ on that truth assignment.
3. If φ ever evaluated to 1, then accept; if not, then reject.

M_1 runs in linear space.

$\overline{\text{ALL}_{\text{NFA}}} \in \text{NSPACE}(n)$

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

N on $\langle M \rangle$, where M is an NFA:

1. Place a marker on the start state of the NFA.
2. Repeat 2^q times, where q is the number of states of M:
3. Nondeterministically select an input symbol and change the positions of the markers on M's state to simulate reading the symbol.
4. Accept if stage 2 and 3 reveal some string that M rejects, that is if at some point none of the markers lie on accept state of M. Otherwise, reject.

Savitch's Theorem

Theorem (Savitch, 1969)

For any function $f : \mathbb{N} \to \mathbb{R}^+$, where $f(n) > n$,

$$\textit{NSPACE}(f(n)) \subseteq \textit{SPACE}(f^2(n)).$$

Proof (1)

CANYIELD on input c_1, c_2 and t:

1. If $t = 1$, then test whether $c_1 = c_2$ or whether c_1 yields c_2 in one step according to the rules of N. Accept if either test succeeds; reject if both fail.
2. If $t > 1$, then for each configuration c_m of N using space $f(n)$:
3. Run CANYIELD($c_1, c_m, t/2$).
4. Run CANYIELD($c_m, c_2, t/2$).
5. If step 3 and 4 both accept, then accept.
6. If haven't yet accept, then reject.

Proof (2)

- Modify N so that when it accepts, it clears its tape and moves the head to the leftmost cell - thereby entering a configuration c_{accept}.
- Let c_{start} be the start configuration of N on w.
- We select a constant d so that N has no more than $2^{d \cdot f(n)}$ configurations using $f(n)$ tape, where $n = |w|$.

Proof (3)

M on input w:

1. Output the result of CANYIELD$(c_{\text{start}}, c_{\text{accept}}, 2^{d \cdot f(n)})$.

M uses space

$$\mathcal{O}\left(\log 2^{d \cdot f(n)} \cdot f(n)\right) = \mathcal{O}\left(f^2(n)\right).$$

Where do we get $f(n)$?

$$M \text{ tries } f(n) = 1, 2, 3, \ldots.$$

The Class PSPACE

Definition

<u>PSPACE</u> is the class of languages that are decidable in polynomial space on a deterministic Turing machine. In other words,

$$\mathsf{PSPACE} = \bigcup_k \mathsf{SPACE}\left(n^k\right).$$

We could also define

$$\mathsf{NPSPACE} = \bigcup_k \mathsf{NSPACE}\left(n^k\right).$$

Then by Savitch's Theorem

$$\mathsf{NPSPACE} = \mathsf{PSPACE}.$$

The relationship of PSPACE with P and NP

A machine which runs in time t can use at most space t.

Hence

$$\text{P} \subseteq \text{PSAPCE} \;\text{ and }\; \text{NP} \subseteq \text{NPSPACE} = \text{PSPACE}.$$

PSPACE and EXPTIME

Let $f : \mathbb{N} \to \mathbb{R}^+$ satisfy $f(n) \geq n$. Then a TM uses $f(n)$ space can have at most

$$f(n) \cdot 2^{\mathcal{O}(f(n))}$$

configurations. Therefore it must run in time $f(n) \cdot 2^{\mathcal{O}(f(n))}$.

Hence

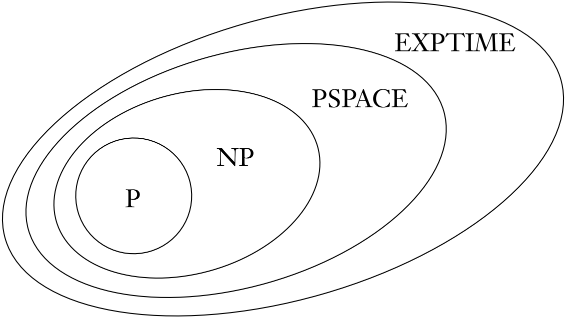
$$\text{PSAPCE} \subseteq \text{EXPTIME} = \bigcup_k \text{TIME}\left(2^{n^k}\right).$$

We know

$$\text{P} \subseteq \text{NP} \subseteq \text{PSAPCE} = \text{NPSPACE} \subseteq \text{EXPTIME}$$

and it is easy to show P$\neq$EXPTIME.

The general consensus is



PSPACE Completeness

Definition

A language B is PSPACE-complete if

1. B is in PSPACE, and
2. every A $\in$PSAPCE is polynomial time reducible to B.

If B merely satisfies condition 2, then it is PSPACE-hard.

Why not polynomial space reducibility?

Let $A, B \subseteq \Sigma^*$. Then A is polynomial space reducible to B, if a polynomial space computable function $f : \Sigma^* \to \Sigma^*$ exists, where for every w

$$w \in A \iff f(w) \in B.$$

Remark

1. Let B be a language with $\emptyset \neq B \neq \Sigma^*$. Then every $A \in$PSPACE is polynomial space reducible to B.
2. Let $B \in P$ and A be polynomial space reducible to B. It is not known that $A \in$P too.

The TQBF problem

- A Boolean formula contains Boolean variables, the constant 0 and 1, and the Boolean operations $\wedge$, $\vee$, and $\neg$.
- The universal quantifiers $\forall$ in $\forall\varphi$ means that φ is true for every value of x in the universe.
- The existential quantifiers $\exists$ in $\exists\varphi$ means that φ is true for some value of x in the universe.
- Boolean formulas with quantifiers are quantified Boolean formulas. For such formulas, the universe is $\{0, 1\}$. E.g.

$$\forall x \exists y \left[(x \vee y) \wedge (\overline{x} \vee \overline{y}) \right].$$

- When each variable of a formula appears within the scope of some quantifier, the formula is fully quantified. A fully quantified Boolean formula is always either true or false.

The TQBF problem

The TQBF problem is to determine whether a fully quantified Boolean formula is true or false, i.e.,

TQBF $= \{\langle\varphi\rangle \mid$ TQBF is a true fully quantifier Boolean formula$\}$.

Theorem

TQBF is PSPACE-complete.

Proof (1)

T on $\langle\varphi\rangle$, a fully quantifier Boolean formula:

1. If φ contains no quantifiers, then it contains no variables, so evaluate φ and accept if it is true; otherwise reject.
2. If $\varphi = \exists x \psi$, recursively call T on ψ first with x:=0 and second with x:=1. If either result is accept, then accept; otherwise reject.
3. If $\varphi = \forall x \psi$, recursively call T on ψ first with x:=0 and second with x:=1. If both results are accept, then accept; otherwise reject.

T runs in polynomial space.

Proof (2)

Let A be a language decided by a TM M in space n^k. We need to show $A \leq_P$ TQBF.
Using two collections of variables c_1, c_2 and $t > 0$, we define a formula $\varphi_{c_1,c_2,t}$. If we assign c_1 and c_2 to actual configurations, the formula is true if and only if M can go from c_1 to c_2 in at most t steps.

Then we can let φ be the formula

$$\varphi_{c_{\text{start}},c_{\text{accetp}},h},$$

where $h = 2^{d \cdot n^k}$, where d is chosen so that M has no more than $2^{d \cdot n^k}$ configurations on an input of length n.

Proof (3)

The formula encodes the contents of configuration cells as in the proof of the Cook-Levin theorem.

- Each cell has several variables associated with it, one for each tape symbol and state, corresponding to the possible settings of that cell.
- Each configuration has n^k cells and so is encoded by $\mathcal{O}\left(n^k\right)$ variables.

Proof (4)

Let $t = 1$. We define $\varphi_{c_1,c_2,1}$ to say that either $c_1 = c_2$, or c_2 follows from c_1 in a single step of M.

- We express $c_1 = c_2$ by saying that each of the variables representing c_1 contains the same Boolean value as the corresponding variables representing c_2.
- We express the second possibility by using the technique presented in the proof of the Cook-Levin theorem.

Proof (5)

Let $t > 1$. Our first try is to define

$$\varphi_{c_1,c_2,t} = \exists m_1 \left[\varphi_{c_1,m_1,t/2} \wedge \varphi_{m_1,c_2,t/2}\right]$$

where m_1 represents a configuration of M.

- $\varphi_{c_1,c_2,t}$ is true if and only if M can go from c_1 to c_2 within t steps.
- But it is of size roughly t, which could be $2^{d \cdot n^k}$, exponential in n.

Proof (6)

Instead we define

$$\varphi_{c_1,c_2,t} = \exists m_1 \forall (c_3, c_4) \in \{(c_1, m_1), (m_1, c_2)\} \left[\varphi_{c_3,c_4,t/2}\right].$$

Then the size of $\varphi_{c_1,c_2,t}$ is $\log t$, bounded by

$$\log 2^{d \cdot n^k} = n^{\mathcal{O}(k)}.$$

Winning strategies for games

The formula game

Let

$$\varphi = \exists x_1 \forall x_2 \exists x_3 \cdots Q x_k \, [\psi]$$

We associate a game with φ.

1. Two players, Player A and Player E, take turns selecting the values of the variables $x_1, \ldots, x_k$.
2. Player A selects values for the variable bounded to $\forall$.
3. Player E selects values for the variable bounded to $\exists$.
4. At the end, if ψ is true, then Player E wins; otherwise Player A wins.

Example

Player E has a winning strategy for

$$\exists x_1 \forall x_2 \exists x_3 \left[(x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3}) \right].$$

Example

Player A has a winning strategy for

$$\exists x_1 \forall x_2 \exists x_3 \left[(x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (x_2 \vee \overline{x_3}) \right].$$

Let

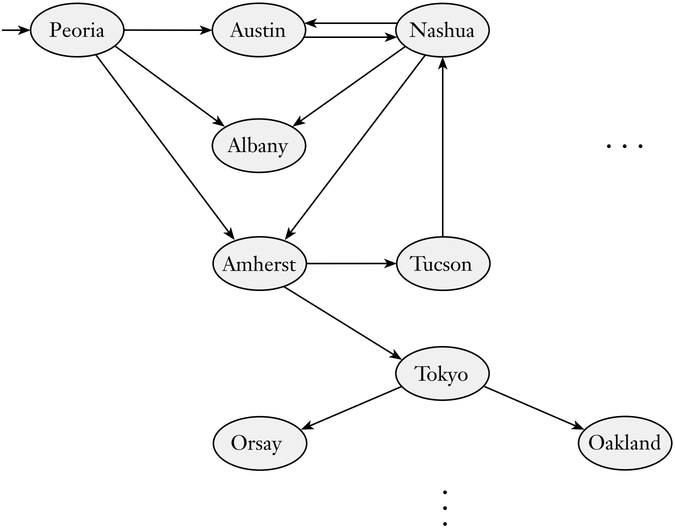
FORMULA-GAME$=\{\langle\varphi\rangle \mid$ Player E has a winning strategy
the formula game associated with $\varphi\}$

Theorem

FORMULA-GAME is PSPACE-complete.

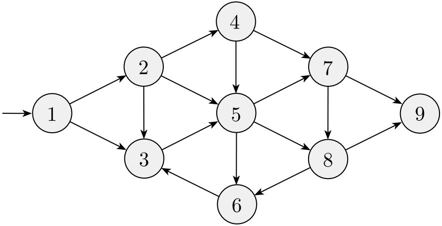
Generalized geography

1. Two players take turns to name cities from anywhere in the world.
2. Each city chosen must begin with the same letter that ended the previous city's name.
3. Repetition isn't permitted.
4. The game starts with some designated starting city and ends when some player can't continue and thus loses the game.


Peoria
Austin
Nashua
Albany
. . .
Amherst
Tucson
Tokyo
Orsay
Oakland

Generalized Geography

1. Take an arbitrary directed graph with a designated start node.



2. Player 1 starts by selecting the designated start node.
3. Then the players take turns picking nodes that form a simple path in the graph.
4. The first player unable to extend the path loses the game.

Let

GG=$\{\langle G, b\rangle|$ Player I has a winning strategy for the geography game played on graph G starting at $b\}$.

Theorem

GG is PSPACE-complete.

Proof (1)

M on $\langle G, b \rangle$:

1. If b has outdegree 0, then reject.
2. Remove node b and all connected arrows to get a new graph G'.
3. For each of the nodes $b_1, b_2, \ldots, b_k$ that b originally pointed at, recursively call M on $\langle G', b_i \rangle$.
4. If all of these accept, then Player II has a winning strategy in the original game, so reject. Otherwise, accept.

M runs in linear space.

Proof (2)

To show the hardness, we give a reduction from *FORMULA-GAME*.

Let

$$\varphi = \exists x_1 \forall x_2 \exists x_3 \cdots Q x_k \, [\psi],$$

where

- the quantifier begin and end with $\exists$, and alternate between $\exists$ and $\forall$,
- ψ is in conjunctive normal form.

We constructs a geography game on a graph G where optimal play mimics optimal play of the formula game on φ, in which Player I in the geography game takes the role of Player E in the formula game, and Player II takes the role of Player A.

Proof (3)

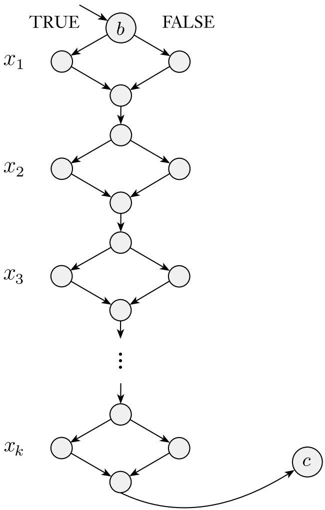
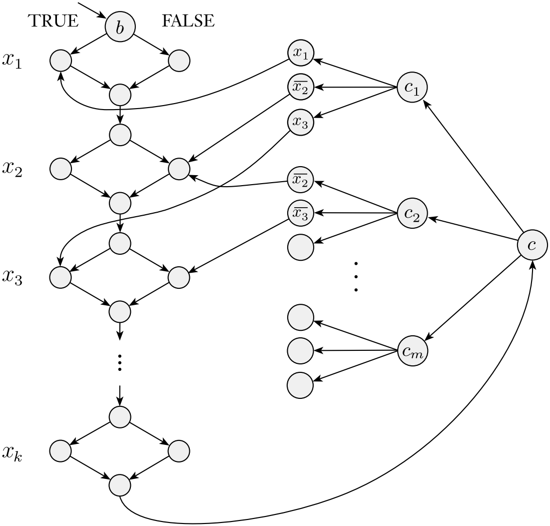


Figure: At the last node of the left diamonds, it is Player I's move.

Proof (4)



$$\exists x_1 \forall x_2 \cdots \exists x_k \left[(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee \cdots) \wedge \cdots \right].$$