Chapter 3 : Space complexity

Time and space are two major parameters for which we measure complexities of computational problems. This chapter is an introduction to the classification of problems based on their space (i.e., memory) requirements. The relationships between the time and space complexity classes will also be explored.

Unless otherwise stated, we take the one-tape one-head TM as our standard model of computation. However, it is meaningful to talk about sub-linear space complexities. Our usual TM model does not comply gracefully with this context. So we will make some relevant changes in the model, as and when necessary.

Notice that space and time differ dramatically in an important aspect: *Space can be reused, whereas time cannot be.* This leads to interesting differences between time and space complexity results.

3.1 Polynomial space complexity

The space complexity $f(n)$ of a DTM M is measured (as a function of the input size n) by the maximum number of tape cells scanned by M on an input of size n. For an NTM we have a similar definition, but now the maximum is taken over all possible branches of computations over all possible inputs of size n.

$$\begin{aligned} \text{SPACE}(f(n)) &:= \{L \mid L \text{ is decided by an } \mathrm{O}(f(n))\text{-space DTM}\}, \\ \text{NSPACE}(f(n)) &:= \{L \mid L \text{ is decided by an } \mathrm{O}(f(n))\text{-space NTM}\}. \end{aligned}$$

To start with we restrict our attention to polynomial space complexities:

$$\text{PSPACE} := \bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k) \qquad \text{and} \qquad \text{NPSPACE} := \bigcup_{k \in \mathbb{N}} \text{NSPACE}(n^k).$$

These classes are invariant for all reasonable encoding schemes and for all computational models that are polynomially equivalent to the one-tape one-head TM.

3.1 Example (1) The satisfiability problem SAT is decided by a DTM that evaluates an input for all possible truth assignments of the variables. The DTM needs the storage of only the current truth assignment of the variables and a mechanism to step through the list of all possible assignments. All these can be done in linear space (but exponential time), so SAT $\in$ PSPACE.

(2) Consider the language

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

An NFA with s states accepts all strings, if and only if it accepts all strings of length $\leqslant 2^s$. This fact allows us to design an NTM N that decides $\overline{\text{ALL}_{\text{NFA}}}$ by guessing a string of length $\leqslant 2^s$ *rejected* by the input NFA A. The simulation of A by N need only store a list of the states of A and a marker indicating the current state of A. Thus N requires nondeterministic linear space, i.e., $\overline{\text{ALL}_{\text{NFA}}} \in$ NPSPACE.

3.2 Lemma Let M be an $\mathrm{O}(f(n))$-space TM with s states and with a tape alphabet of size g. Then M has at most $sf(n)g^{f(n)}$ different configurations on an input of size n. For $f(n) \geqslant n$ this number is $2^{\mathrm{O}(f(n))}$.

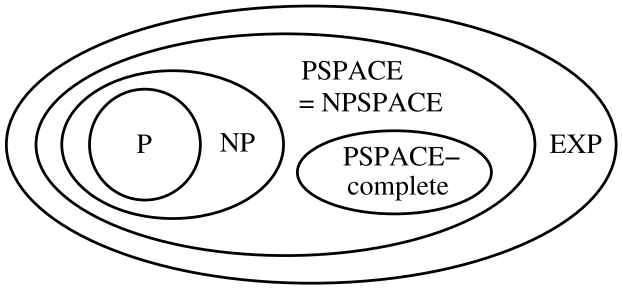
3.3 Theorem [Savitch's theorem] For $f(n) \geqslant n$ we have NSPACE$(f(n)) \subseteq$ SPACE$(f^2(n))$.

3.4 Corollary PSPACE = NPSPACE.

Thus the deterministic and nondeterministic polynomial space complexity classes are provably the same. The corresponding question for time complexity ($\mathrm{P} = \mathrm{NP}$) is yet unresolved.

Now we look at the relation between time and space complexity classes. A poly-time TM cannot consume more than poly-space and so $\mathrm{P} \subseteq \mathrm{PSPACE}$ and $\mathrm{NP} \subseteq \mathrm{NPSPACE}$. Since a TM that halts must not repeat any configuration on a given input (so as to avoid infinite loops), Lemma 3.2 guarantees that $\mathrm{PSPACE} \subseteq \mathrm{EXP}$. So we have a situation as depicted in Figure 3.1. We can (and will) prove that $\mathrm{P} \subsetneqq \mathrm{EXP}$, but it is unknown which of the containment(s) in the chain $\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP}$ is/are proper.

Figure 3.1: Relation between time and space complexity classes



In order to identify the most difficult problems in the class PSPACE we should identify a suitable reduction mechanism among PSPACE problems. In order that this reduction can correctly portray the relative difficulty of different problems in this class, we will stick to poly-time reducibility. (Note that poly-time is (apparently) a more stringent constraint that poly-space.)

3.5 Definition A language L is called PSPACE-complete, if it satisfies the following two conditions:

(1) $L \in \mathrm{PSPACE}$, and

(2) $A \leqslant_P L$ for every $A \in \mathrm{PSPACE}$.

3.6 Theorem TQBF is $\mathrm{PSPACE\text{-}complete}$, where TQBF is defined to be the language

$$\mathrm{TQBF} := \{\langle\phi\rangle \mid \phi \text{ is a true fully quantified Boolean formula}\}.$$

Given a fully quantified Boolean formula $\phi = \exists x_1 \forall x_2 \exists x_3 \cdots \mathrm{Q} x_k[\psi]$, we may think of a game in which two players E and A take part and assign variables $x_1, x_2, \ldots, x_k$ in succession with E assigning those corresponding to the existential quantifier ($\exists$) and A those corresponding to the universal quantifier ($\forall$). If this assignment lets ψ evaluate to true, player E wins, otherwise player A wins. We say that player E has a *winning strategy* for formula ϕ, if irrespective of the assignments of player A, player E can assign the variables (bound to existential quantifiers) in such a way that E wins. It is then clear that E has a winning strategy for ϕ, if and only if ϕ is a true formula. It follows that:

3.7 Theorem $\mathrm{FORMULA\text{-}GAME}$ is $\mathrm{PSPACE\text{-}complete}$, where

$$\mathrm{FORMULA\text{-}GAME} := \{\langle\phi\rangle \mid \text{Player } E \text{ has a winning strategy for the formula-game associated with the fully quantified Boolean formula } \phi\}.$$

Consider another game: ‘Antakhshari’ on the name of cities (a.k.a. the *geography game*). This game can be modeled as a problem associated with a directed graph G. Each node in the graph is labeled by the name of a city and an edge from node u to node v implies that u’s label ends with the same letter as v’s label begins with. The game starts at a predetermined node in the graph and two players alternately moves in the graph to yet-unvisited vertices following the edges of the graph. The player who fails to make a next move loses. In a generalized version of this game, we are given an arbitrary directed graph (without any association of nodes with cities or of edges with matching letters) and a predetermined node in it. The problem is to determine if the first player has a winning strategy. Formally:

$$\mathrm{GA} := \{\langle G, s\rangle \mid \text{The first player has a winning strategy for the generalized Antakhshari game on the directed graph } G\text{, if the game starts at node } s\}.$$

3.8 Theorem GA is PSPACE-complete.

For the proof of the theorem one may reduce FORMULA-GAME to GA in poly-time. Since it is believed that $\mathrm{P} \subsetneqq \mathrm{PSPACE}$, there is seemingly no poly-time algorithm for the first player to check the existence of a winning strategy, let alone determining it.

Exercises for Section 3.1

1. Show that:

(a) Any PSPACE-hard language is also NP-hard.

(b) If every NP-hard language is also PSPACE-hard, then $\mathrm{PSPACE} = \mathrm{NP}$.

2. Demonstrate that PSPACE is closed under union, intersection, complement and Kleene closure.

***3.** Show that the language

$$\mathrm{DTM}_{\text{IN-PLACE}} := \{\langle M, \alpha\rangle \mid M \text{ accepts } \alpha \text{ without leaving the first } |\alpha| \text{ cells of the tape}\}$$

is PSPACE-complete. (**Hint:** You may directly reduce any $L \in \mathrm{PSPACE}$ to $\mathrm{DTM}_{\text{IN-PLACE}}$.)

4. A l i n e a r b o u n d e d a u t o m a t o n (L B A) is a one-tape one-head nondeterministic TM with the tape finite and just big enough to hold the entire input. If an LBA tries to move its head outside its tape at either end, some mechanism prevents it from doing so, and the head remains in the old position. Deduce that the language

$$A_{\text{LBA}} := \{\langle M, \alpha\rangle \mid \text{The LBA } M \text{ accepts } \alpha\}$$

is PSPACE-complete.

5. The generalized tic-tac-toe game is played on an $n \times n$ board with player X starting and with players X and O making alternate moves. The player who first places his marker in five consecutive cells in a row, column or diagonal wins. The game ends in a draw if no such sequence is present when all of the n^2 cells are marked. Show that the language

$$\mathrm{GT} := \{\langle c\rangle \mid c \text{ is an intermediate configuration of the board with next move by X and with a winning strategy for X}\}$$

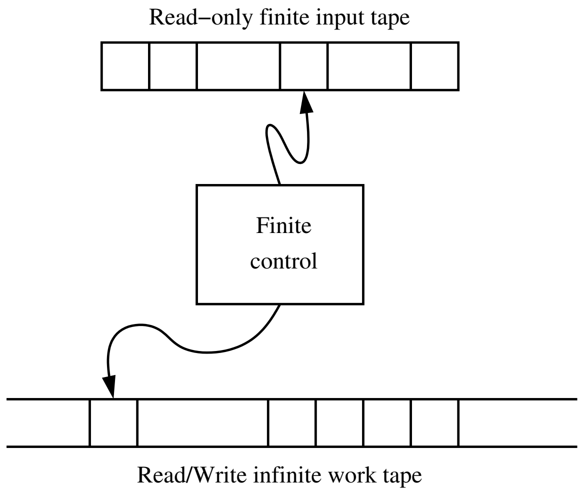
is in PSPACE. (**Hint:** Use recursion.)

3.2 Logarithmic space complexity

A TM is expected to read the full input given to it and just doing that takes time linear in the input size. Languages (like the set of all binary strings starting with 0) for which the entire input need not be scanned are uninteresting in complexity theory. So we do not bother to define sublinear time complexity classes.

With space complexity, on the other hand, the situation is different. Suppose that the input given in a read-only memory (like CD-ROM) and the objective is to process the input using a separate read/write memory.

Figure 3.2: The TM model for log-space computations



Even when the input is pretty big, we may require little additional space to process the data. In view of this it is expedient to define complexity classes comprising problems demanding sublinear (additional) space.

Our standard one-tape one-head TM requires linear space in its working memory just to store the input. It is, therefore, necessary now to alter this model a bit to cope with sublinear space complexity. A solution is provided in Figure 3.2. This TM has a read-only tape which is finite and just big enough to store the input. Some mechanism prevents the read head from falling off either end of the input. For the processing of the input the TM uses an infinite read/write work tape. The space complexity of this TM is measured by the number of cells in its work tape, visited by the read/write head.

The *configuration* of this new model is described by the following quantities:

- The state of the finite control.
- The content of the work tape.
- The positions of the two heads.

Note that the content of the input tape is not part of the configuration, since this tape contains precisely the (known) input α throughout the machine's computation.

3.9 Lemma Let M be an $f(n)$-space TM of the kind shown in Figure 3.2. The number of configurations of M on an input of size n is at most $n2^{O(f(n))}$. For $f(n) \geqslant \lg n$ this number is $2^{O(f(n))}$.

Savitch's theorem continues to hold for this new model for space complexities $f(n) \geqslant \lg n$. Moreover, for space complexities $f(n) \geqslant n$ the new model is equivalent to our previous one.

Sublinear spaces like $\sqrt{n}$ or $n/\lg n$ are not really interesting. We start with logarithmic space. There is a class of problems which requires just $O(\lg n)$ space (and not a polynomial of $\lg n$).

$$\mathrm{L} := \mathrm{SPACE}(\lg n) \qquad \text{and} \qquad \mathrm{NL} := \mathrm{NSPACE}(\lg n).$$

These classes are robust against change of (reasonable) encoding schemes (but not necessarily against change of the computational model). It is worthwhile to mention here that logarithmic space is sufficient to store pointers to the input. Thus we may expect a decent bunch of problems in the classes L and NL. These classes (more correctly, log-space reduction algorithms) are also useful to differentiate among 'easy' problems, like those in the class P.

3.10 Example (1) The context-free language $\{0^k1^k \mid k \geqslant 0\} \subseteq \{0,1\}^*$ is in L. Given an input α a deterministic TM may first check if α is in the correct form – this stage requires no extra space. Once a string of the format 0^k1^l is confirmed, the numbers k and l are counted in the work tape and subsequently compared with one another. Since k and l can be stored in binary using $\mathrm{O}(\lg n)$ bits, the work tape requires only logarithmic space.

(2) We have seen that the language

$$\mathrm{PATH} := \{\langle G, s, t\rangle \mid \text{ There is a path from vertex } s \text{ to vertex } t \text{ in a directed graph } G\}$$

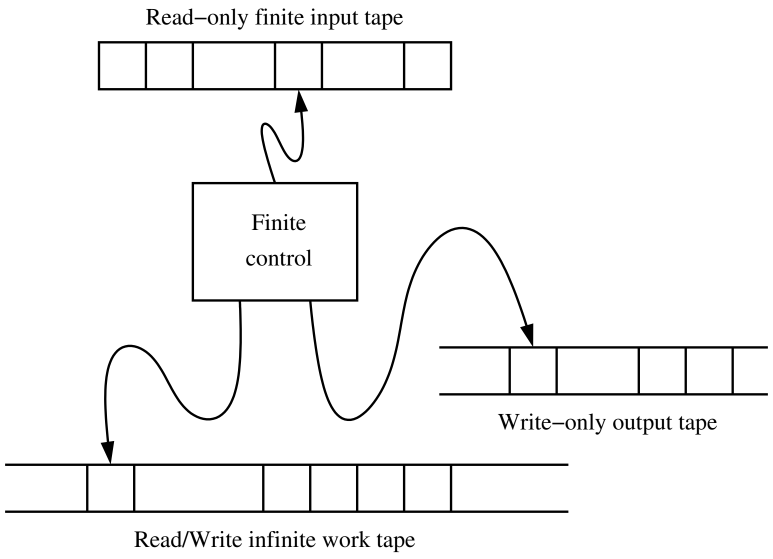
is in P. We do not know if PATH $\in$ L. But we can design an NTM that nondeterministically explores paths from s of length $\leqslant m$ (where $m = n(G)$) and accepts if and only if t is reachable from s. At any point of the computation it is sufficient to store only the current node in a path being explored. Thus PATH $\in$ NL.

PATH illustrates the possibility that L $\neq$ NL. In order to tackle this (yet unsolved) question, it is worthwhile to look at complete problems in the class NL. By Lemma 3.9 L $\subseteq$ P and so poly-time reducibility is inadequate to compare relative difficulties of problems in NL. We must use 'easier' reductions.

3.11 Definition A log-space transducer is a Turing machine as described in Figure 3.3. It consists of a finite read-only input tape that holds the input (and nothing else), a write-only output tape (possibly infinite) and a usual infinite read/write work tape. On an input α of size n the transducer uses (i.e., visits or scans) $\mathrm{O}(\lg n)$ symbols in its work tape and ends its computation with $f(\alpha)$ (and nothing else) on the output tape. The function $f : \Sigma^* \to \Sigma^*$ that the transducer computes is called a log-space computable function.

We say that a language A is log-space reducible to a language B, if there exists a log-space computable function f, such that $\alpha \in A$ if and only if $f(\alpha) \in B$. In this case we write $A \leqslant_L B$.

Figure 3.3: A log-space transducer



3.12 Definition A language L is called NL-complete, if it satisfies the following two conditions:

(1) $L \in$ NL, and

(2) $A \leqslant_L L$ for every $A \in$ NL.

3.13 Theorem If $A \leqslant_L B$ and $B \in \mathrm{L}$, then $A \in \mathrm{L}$ too.

3.14 Theorem If $A \leqslant_L B$ and A in $\mathrm{NL\text{-}complete}$, then B is $\mathrm{NL\text{-}complete}$ too.

3.15 Theorem If some $\mathrm{NL\text{-}complete}$ language is in L, then $\mathrm{L} = \mathrm{NL}$.

3.16 Theorem PATH is $\mathrm{NL\text{-}complete}$.

3.17 Corollary $\mathrm{NL} \subseteq \mathrm{P}$.

We end our study of logarithmic space complexity by introducing some other complexity classes. First we define coNL to be the class of all languages that are complements of languages in NL. One can design a nondeterministic log-space algorithm for $\overline{\mathrm{PATH}}$. Since PATH is $\mathrm{NL\text{-}complete}$, it follows that:

3.18 Theorem $\mathrm{NL} = \mathrm{coNL}$.

We can also define super-linear log-space classes:

$$\mathrm{L}^k := \mathrm{SPACE}(\lg^k n), \quad \mathrm{NL}^k := \mathrm{NSPACE}(\lg^k n), \quad \mathrm{PolyL} := \bigcup_{k \in \mathbb{N}} \mathrm{L}^k = \bigcup_{k \in \mathbb{N}} \mathrm{NL}^k.$$

These classes apparently turn out to be not so useful as the classes $\mathrm{L} = \mathrm{L}^1$ and $\mathrm{NL} = \mathrm{NL}^1$. We have the following containments:

L^k	$\subsetneqq$	L^{k+1}	[By the space hierarchy theorem]
NL^k	$\subsetneqq$	NL^{k+1}	[By the space hierarchy theorem]
L^k	$\subseteq$	NL^k	[It's not known if this containment is proper.]
NL^k	$\subseteq$	L^{2k}	[By Savitch's theorem]
PolyL	$\subsetneqq$	PSPACE	[By the space hierarchy theorem]

Exercises for Section 3.2

1. Demonstrate that NL is closed under union, intersection and Kleene closure.

2. Show that:
(a) The language consisting of strings with properly nested parentheses is in L.
* **(b)** The language consisting of strings with properly nested parentheses and (square) brackets is also in L.

3. Prove that $\mathrm{BIPARTITE} \in \mathrm{NL}$.

4. If f and g are log-space computable functions, show that so also is $g \circ f$. Conclude that if $A \leqslant_L B$ and $B \leqslant_L C$, then $A \leqslant_L C$.

** **5.** Demonstrate that 2SAT is $\mathrm{NL\text{-}complete}$. (**Hint:** You may show that $\mathrm{PATH} \leqslant_L \overline{\mathrm{2SAT}}$.)

6. A directed graph G is called *strongly connected*, if for every pair of vertices u, v in G there exists a (directed) u, v path in G. Prove that the language

$$\mathrm{STRONGLY\text{-}CONNECTED} := \{\langle G \rangle \mid G \text{ is a strongly connected digraph}\}$$

is $\mathrm{NL\text{-}complete}$. (**Hint:** You may use reduction from PATH.)

* **7.** Prove that TQBF and GA are $\mathrm{PSPACE\text{-}complete}$ even under log-space reductions. Conclude that these languages are not in NL.