# SPACE COMPLEXITY

This material is covered in Chapter 8 of the textbook.

For simplicity, we define the "space" used by a Turing machine computation only for machines that halt on all inputs.

**Definition.** (Def. 8.1)

1. Let $M$ be a deterministic Turing machine (DTM) that halts on all inputs. The *space used by* $M$ (space complexity of $M$) is the function $f : \mathbb{N} \longrightarrow \mathbb{N}$ where $f(n)$ is the maximum number of tape cells that $M$ scans on any input of length $n$.

2. Let $M$ be a nondeterministic Turing machine (NTM) where all computation branches halt on all inputs. The *space used by* $M$ is the function $f : \mathbb{N} \longrightarrow \mathbb{N}$ where $f(n)$ is the maximum number of tape cells that $M$ scans on any branch of its computation on any input of length $n$.

As with time complexity, when defining space complexity we ignore constant factors:

$$\text{SPACE}(f(n)) = \{L \mid L \text{ is decided by some DTM in space } O(f(n))\}$$

$$\text{NSPACE}(f(n)) = \{L \mid L \text{ is decided by some NTM in space } O(f(n))\}$$

The following observations are immediate (why?):

Let $f : \mathbb{N} \longrightarrow \mathbb{N}$ be any function. Then

- $\text{TIME}(f(n)) \subseteq \text{SPACE}(f(n))$
- $\text{NTIME}(f(n)) \subseteq \text{NSPACE}(f(n))$
- $\text{TIME}(f(n)) \subseteq \text{NTIME}(f(n))$
- $\text{SPACE}(f(n)) \subseteq \text{NSPACE}(f(n))$

An important fact concerning space bounded computations (and difference with time bounded computations) is that "space can be re-used" whereas time obviously cannot be re-used. This means that already a linear space bound is quite powerful. Turing machines with a linear space bound were considered earlier under the name linear bounded automaton (LBA).

**Example.** SAT $\in$ SPACE($n$)

Recall that comparing deterministic and nondeterministic time complexity classes was very difficult. For space complexity classes we have the following remarkable result:

**Theorem.** (Savitch's theorem) For any function $f$ where $f(n) \geq n$ we have:

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

*Note.* The above formulation of Savitch's theorem relies on the assumption that all computations of the Turing machines are guaranteed to halt. Recall that we defined the space complexity classes only for decider TM's. If the TM's are not guaranteed to halt, Savitch's theorem needs to impose some restrictions on the function $f(n)$ (that is, $f(n)$ has to be space constructible, as defined in Chapter 9).

Given an arbitrary Turing machine $M$, it is *undecidable* for example whether $M$ operates in time/space $p(n)$ where $p$ is a given polynomial.

On the other hand, for any "well behaving" function[1] $f$ and an arbitrary Turing machine $M$ we can construct a Turing machine $M_f$ with the following properties:

1. $L(M_f)$ consists of exactly all strings $w$ such that $M$ accepts $w$ in time (or space) $f(|w|)$.

[1] "well behaving" means here *time constructible* or *space constructible.* These notions will be defined in Chapter 9 of the textbook. For our present purposes it is sufficient to know that all naturally defined functions like the polynomial, exponential, logarithmic functions are "well behaved".

2. $M_f$ operates in time $f(n)$ (or space $f(n)$).

3. All computations of $M_f$ halt.

The above facts explain why it is reasonable that we can assume that all computations of a time-bounded (or space-bounded) Turing machine halt. This assumption does not change the class of recognized languages as long as we are dealing with "well behaved" time and space bounds.

The language families defined by deterministic/nondeterministic polynomial space Turing machines are:

$$\text{PSPACE} = \bigcup_k \text{SPACE}(n^k)$$

$$\text{NPSPACE} = \bigcup_k \text{NSPACE}(n^k)$$

By Savitch's theorem we have:

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

The deterministic exponential time class is

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

**Lemma.** PSPACE $\subseteq$ EXPTIME.

*Proof:* will be done in class.

The inclusions of the time/space complexity classes studied up to now are:

$$\text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME}$$

- Note that NP $\subseteq$ PSPACE follows from the equality PSPACE = NPSPACE.

- We will later prove that P $\subset$ EXPTIME (strict inclusion). However, it is not known whether any of the other inclusions in the above chain are strict. It is *conjectured* that all inclusion in the chain are likely to be strict.

Next we will consider complete problems for the class PSPACE.

**Definition.** A language $B$ is PSPACE-complete if

1. $B \in$ PSPACE, and
2. $A \leq_P B$ for every $A \in$ PSPACE.

A *quantified Boolean formula* allows (in addition to the Boolean operations) the use of universal ($\forall$, "for all") and existential ($\exists$, "exists") quantifiers. A *fully* quantified formula where each variable is in the scope of a quantifier has a value true or false.

The quantified Boolean formulas problem is encoded as the language:

$$\text{TQBF} = \{< \varphi > | \ \varphi \text{ is a true fully quantified Boolean formula } \}$$

**Theorem.** TQBF is PSPACE-complete.

- A straightforward recursive algorithm decides TQBF in polynomial space (even in linear space).
- The proof showing that any problem in PSPACE is reducible to TQBF uses an extension of the construction from the proof of the Cook-Levin theorem.

Other PSPACE-complete problems:

- Evaluating a quantified Boolean formula resembles determining a winning strategy in a two-person game, see section 8.3. Many games ("Geography", as well as, chess and Go on $n \times n$ boards) are PSPACE-complete.

- The following language is PSPACE-complete:

$$B = \{ < M, w > \quad | \quad M \text{ is a DTM and } M \text{ accepts } w \text{ without leaving the first } |w| + 1 \text{ squares of the tape } \}$$

  Note that $B$ can be accepted even in linear space.