Notes on Complexity Theory

Richard Willie

September 9, 2024

Contents

1 Space Complexity

In this chapter, we consider the complexity of computational problems in terms of the amount of space, or memory, that they require. As we did with time complexity, we need to select a model for measuring the space used by an algorithm. We continue with the Turing machine model for the same reason that we used it to measure time. Turing machines are mathematically simple and close enough to real computers to give meaningful results.

Definition 1.0.1 (Space complexity)

Let M be a deterministic Turing machine 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 M is a nondeterministic Turing machine 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.

We typically estimate the space complexity of Turing machines by using asymptotic notation.

Definition 1.0.2 (The classes **SPACE** and **NSPACE**)

Let $f : \mathbb{N} \to \mathbb{R}^+$ be a function. The space complexity classes, $\textbf{SPACE}(f(n))$ and $\textbf{NSPACE}(f(n))$, are defined as follows.

$$\begin{aligned}\textbf{SPACE}(f(n)) = \{L \mid\ & L \text{ is a language decided by an } O(f(n)) \text{ space} \\ & \text{deterministic Turing machine}\} \\ \textbf{NSPACE}(f(n)) = \{L \mid\ & L \text{ is a language decided by an } O(f(n)) \text{ space} \\ & \text{nondeterministic Turing machine}\}\end{aligned}$$

§1.1 Savitch's Theorem

Savitch's theorem is an early and significant result in space complexity. It demonstrates that deterministic machines can simulate nondeterministic machines with a surprisingly small increase in space. While time complexity simulations typically require an exponential increase in time, Savitch's theorem proves that any nondeterministic TM using $f(n)$ space can be converted into a deterministic TM that only requires $f^2(n)$ space.

Theorem 1.1.1 (Savitch's theorem, preliminary version)
For any[a] function $f : \mathbb{N} \to \mathbb{R}^+$, where $f(n) \in \Omega(n)$,

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

[a]Later on, we show that Savitch's theorem also holds whenever $f(n) \in \Omega(\log n)$.

Proof Idea. We need to simulate an $f(n)$ space NTM deterministically. A naive approach is to proceed by trying all the branches of the NTM's computation, one by one. The simulation needs to keep track of which branch it is currently trying so that is able to go onto the next one. But a branch that uses $f(n)$ may run for $2^{O(f(n))}$ steps and each step may be a nondeterministic choice. Exploring the branches sequentially would require recording all the choices used on a particular branch in order to be able to find the next branch. Therefore, this approach may use $2^{O(f(n))}$ space, exceeding our goal of $O(f^2(n))$ space.

Instead, we take a different approach by considering the following more general problem. We are given two configurations of the NTM, c_1 and c_2, together with a number t, and we test whether the NTM can get from c_1 to c_2 within t steps using only $f(n)$ space. We call this problem the **yieldability problem**. By solving the yieldability problem, where c_1 is the start configuration, c_2 is the accept configuration, and t is the maximum number of steps that the nondeterministic machine can use, we can determine whether the machine accepts its input.

We give a deterministic, recursive algorithm that solves the yieldability problem. It operates by searching for an intermediate configuration c_m, and recursively testing whether

1. c_1 can get to c_m within $t/2$ steps, and
2. whether c_m can get to c_2 within $t/2$ steps.

Reusing the space for each of the two recursive tests allows a significant savings of space.

This algorithm needs space for storing the recursion stack. Each level of the recursion uses $O(f(n))$ space to store a configuration. The depth of the recursion is $\log t$, where t is the maximum time that the nondeterministic machine may use on any branch. We have $t = 2^{O(f(n))}$, so $\log t = O(f(n))$. Hence the deterministic simulation uses $O(f^2(n))$ space. □

§1.2 The Class PSPACE

§1.3 PSPACE-completeness

§1.4 The Classes L and NL

§1.5 NL-completeness

§1.6 NL $=$ coNL