Undergrad Complexity: Lecture 2

Ben Chaplin

Contents

1 Background **1**
- 1.1 Languages . . . 1
- 1.2 Algorithms . . . 1

2 Turing Machines **1**
- 2.1 Model . . . 1
- 2.2 Formal definition . . . 2

1 Background

1.1 Languages

Definition. A **language L** over an alphabet Σ is a subset of strings $L \subseteq \Sigma^*$.

Example. PRIMES $= \{\langle x \rangle : x \in \mathbb{N}, x \text{ is prime}\}$, equivalent to the decision problem $\{0,1\}^* \to \{\text{no, yes}\}$.

1.2 Algorithms

What is an algorithm? We can define the concept using programming languages, or better, models of computation. Turing machines are a good choice as they're easy to formalize. Furthermore:

Church-Turing Thesis. Any real-world algorithm can be simulated by Turing Machines.

However, this is a statement about computability. In terms of complexity theory, it's important to note that an algorithm running in time T in C-like pseudocode, it can compiled to a Turing machine running in time roughly T^4.

2 Turing Machines

2.1 Model

Official model of a Turing Machine:

- one tape
- two-way infinite

Roughly, the tape holds symbols from an alphabet and reads/writes with a "head" pointer. The source code tells us how to move and write.

2.2 Formal definition

Definition. A **Turing machine M** is a 7-tuple $M = (\Gamma, b, \Sigma, Q, q_0, F, \delta)$, where

- Γ is a finite, non-empty set of tape symbols
- $b \in \Gamma$ is the blank symbol (allowed to occur infinitely on tape)
- $\Sigma \subseteq (\Gamma - \{b\})$ is the input alphabet
- Q is a finite, non-empty set of states
- q_0 is the initial state
- $F \subseteq Q$ is the set of accepting states
- $\delta : (Q - F) \times \Gamma \nrightarrow Q \times \Gamma \times \{\text{left, right}\}$ is the transition function. If δ is not defined on the current tape symbol, the machine halts.