Algorithms & Complexity: Lecture 1

Sam Barrett

February 1, 2021

1 Defining the Turing machine model

In order to talk about the time taken or the space used by an algorithm, we require a precise **model of computation**. There are many proposed models, we will focus on the Turing machine as defined by Arora and Barak in their book.

1.1 Arora-Barak Turing machines

1.1.1 Tapes

A Turing machine is defined as having k tapes where $k \geq 2$

- The first tape is the *input tape* and is **read-only**
- The $2..k-1$ tapes are *work* tapes and are **read-write**
- The k^{th} tape is the *output* tape.

Each tape has a leftmost cell and *potentially* infinitely many cells to the right of it. Potentially infinite meaning that at any given time, there are a finite number of cells but we can infinitely extend the tape over time.

Each tape has a **head** that sits on a cell and can move left and right.

1.1.2 Alphabet

A Turing machine also has an alphabet, denoted Γ. This is a **finite** set and it's elements are called *symbols*. There are 4 primary symbols: $\triangleright, \square, 0, 1$.

Here:

- $\{0,1\}^*$ is the set of bitstrings, the empty string is denoted with ε.
- $\triangleright$ is the left-of-tape symbol and $\square$ is the blank symbol

At any point in time, each cell of each tape contains a symbol. All but a finite number will be blank ($\square$)

1.1.3 Inital configuration

The input tape has $\triangleright$ on the leftmost cell, then a bitstring (the **input**) and the rest of the tape is blank. The work tapes (including the output tape) have $\triangleright$ on the leftmost cell and the rest are blank. Each tape starts with it's head on it's the leftmost cell.

1.1.4 Computation step

In a single step of computation the machine:

- reads the character at each tape head
- writes a character at each work tape head
- may move each tape head to the left or to the right. **note: our tapes are not recursive, if a head on the leftmost cell moves left, it stays put**

1.1.5 Formal definition

A **Turing machine** is defined as a (6) tuple, $M = (k, \Gamma, Q, q_{\texttt{start}}, q_{\texttt{halt}}, \delta)$ consists of the following data:

- the number of tapes, k, $k \geq 2$
- the alphabet $\{0, 1, \triangleright, \square\} \subseteq \Gamma$
- a finite set of Q states, including the start state $q_{\texttt{start}}$ and the halt state $q_{\texttt{halt}}$
- a transition function, $\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{L, R, S\}^k$ Where:
  - the initial Q is the state at the start of transition
  - Γ^k is the set of symbols read
  - the final Q is the state at the end of transition
  - Γ^{k-1} is the set of symbols written
  - $\{L, R, S\}^k$ is the set of movement instructions where:
    - L means *move left*
    - R means *move right*
    - S means *stay*

  Note: we read k symbols but only write $k-1$ symbols as we do not write on the input tape, we also have k movement instructions as we are able to move on all k of the tapes.

1.1.6 Example transition

Say we have $k =$, and $\Gamma = \{\triangleright, \square, 0, 1\}$ and $Q = \{4, 5, 6, 7, 8\}$ with $q_{\texttt{start}} = 4$ and $q_{\texttt{halt}} = 8$. We are currently in state 7 and the three tapes respectively say 1 (input), 1 (work) and $\square$ (output).

Say that $\delta(7, \langle 1, 1\square\rangle) = (5, \langle 0, \square\rangle, \langle L, L, S\rangle)$ then we:

- transition to state 5
- overwrite thje 1 on the work tape with 0
- overwrite the $\square$ on the output tape with $\square$ (no change)
- move left on the input tape (if possible)
- move left on the work tape (if possible)
- stay put on the output tape

We do not transition from the halt state (regardless of δ)

2 Computing with Turing machines

2.1 Computing a function

Given a function $f : \{0,1\}^* \to \{0,1\}^*$ a Turing machine $M = (k, \Gamma, Q, q_{\texttt{start}}, q_{\texttt{halt}}, \delta)$,

1. what does it mean to say that M **computes** f?

   It means that for every bitstring $x \in \{0,1\}^*$, if we start in state $q_{\texttt{start}}$ with the initial configuration showing x (meaning x appears on the input tape and and the work tapes are blank), when we run M, we eventually reach $q_{\texttt{halt}}$ with the output tape showing $\triangleright$ on the leftmost cell and then the bitstring $f(x)$ followed by all blanks.

Our initial configuration can be shown as:

Input tape:

$\triangleright$	1	0	1	1	

Work tapes:

$\square$	$\square$	$\square$	$\square$	$\square$	

Output tape: (also a work tape)

$\square$	$\square$	$\square$	$\square$	$\square$	

Given that $f(x) = 0110111$, our required output tape will then be as follows:

$\triangleright$	0	1	1	0	1	1	1	1	

If the machine, M does *this* for every bitstring x then we say it **computes** f. In the Arora-Barak definition, it does not matter what is on the work tape at the end of execution or the location of the work heads.