CSC363 Tutorial #1

Turing machines and stuff

January 18, 2023

I can't believe it's 2023 already

Things you should learn in this tutorial

Things you should learn in this tutorial

- What a Turing machine is, at a high level.
- How to trace the execution of a Turing machine.
- How to program a Turing machine at a low level: defining its states, defining the transition table.

Some administrative stuff

- Quiz 1 will be released on Jan 21, and due on Jan 23. Please remember to complete the quiz! Note that Quer*us sometimes doesn't send you notifications for quizzes.

A bit about myself?

Hi! I'm Paul Zhang, some 5th year student studying math/cs.

A bit about myself?

Hi! I'm Paul Zhang, some 5th year student studying math/cs.

- Contact: pol.zhang@utoronto.ca

A bit about myself?

Hi! I'm Paul Zhang, some 5th year student studying math/cs.

- Contact: pol.zhang@utoronto.ca
- Office Hours: Wednesdays 4:30-5:30pm, DH2034.
- Website (slides are posted here!): sjorv.github.io

A bit about myself?

Hi! I'm Paul Zhang, some 5th year student studying math/cs.

- Contact: pol.zhang@utoronto.ca
- Office Hours: Wednesdays 4:30-5:30pm, DH2034.
- Website (slides are posted here!): sjorv.github.io
- Hobbies: Gaming, taking naps at inappropriate times

A bit about myself?

Hi! I'm Paul Zhang, some 5th year student studying math/cs.

- Contact: pol.zhang@utoronto.ca
- Office Hours: Wednesdays 4:30-5:30pm, DH2034.
- Website (slides are posted here!): sjorv.github.io
- Hobbies: Gaming, taking naps at inappropriate times
  - Favourite food: sushi juice
  - D&D alignment: neutral good???? idk
  - horoscope: ummm... you can have my birthday i guess... december 11
  - mbti: hmmmm good question... you can try to guess...
  - facebook: nope :(
  - twi**er: 
  - discord: you can probably find my discord tag by word of mouth
  - favourite word: poggers
  - favourite fried rice topping: rice
  - where i would like to travel to: my fridge
  - most embarassing moment: when i was 11, i stole a connect-four set from a kid at burger king and made the kid cry :(why did i do that
  - favourite emoji: 💀
  - courses I'm TAing this semester: MAT157, CSC263, CSC363
  - favourite music genre: ummmm... can't really say anything specific... mostly edm related subgenres like breakcore, speedcore, happy hardcore, or dnb, something like that?
  - favourite historical person: nice try fbi
  - what i usually eat: pomeloes in the winter, watermelon in the summer
  - number of posters in my room: 1
  - furthest i've driven a car: 0 km
  - best gaming achievement: completing all super meat boy achievements, including the no death runs
  - how old i feel: simultaneously 15 years old and 63 years old
  - most recent book: tuesdays by morrie! it's a nice book...

What's a Turing Machine?

What's a Turing Machine?

There are various ways to formally define what a Turing Machine (TM) is.

What's a Turing Machine?

There are various ways to formally define what a Turing Machine (TM) is. Here's a definition from *Turing Computability* (Soare 2016).

1.4 ** Turing Machines

Definition 1.4.1. (Turing) A *Turing machine (automatic machine, a-machine)* M includes a two-way infinite *tape* divided into *cells*, a *reading head* which scans one cell of the tape at a time, and a finite set of internal *states* $Q = \{q_0, q_1, \dots, q_n\}$, $n \geq 1$. Each cell is either blank (B) or has written on it the symbol 1. In a single step the machine may simultaneously: (1) change from one state to another; (2) change the scanned symbol s to another symbol $s' \in S = \{1, B\}$; and (3) move the reading head one cell to the right (R) or left (L). The operation of M is controlled by a partial map $\delta : Q \times S \to Q \times S \times \{R, L\}$ (which may be undefined for some arguments).

The interpretation is that if $(q, s, q', s', X) \in \delta$ then the machine M in state q, scanning symbol s, changes to state q', replaces s by s', and moves to scan one square to the right if $X = R$ (or left if $X = L$). The map δ viewed as a finite set of quintuples is called a *Turing program*. The *input* integer x is represented by a string of $x + 1$ consecutive 1's (with all other cells blank). The Turing machine is pictured in Figure 1.1.

We begin with M in the *starting state* q_1 scanning the leftmost cell containing a 1, called the *starting cell*. If the machine ever reaches the *halting state* q_0, after say s steps, then we say M *halts* and the *output* y is the total number of 1's on the tape. (Note that $f(x) = \max\{x + 1, s\}$ bounds the maximum distance from the starting cell to any cell which is either scanned or contains an input symbol. Hence the determination of y is effective.)

We may assume that M never makes any further moves after reaching state q_0, i.e., that the domain of δ contains no element of the form (q_0, s). We say that M *computes* the partial function ψ provided that $\psi(x) = y$

Figure: Soare's definition of a Turing Machine

What's a Turing Machine?

There are various ways to formally define what a Turing Machine (TM) is. Here's a definition from *Turing Computability* (Soare 2016).

1.4 ** Turing Machines

Definition 1.4.1. (Turing) A *Turing machine (automatic machine, a-machine)* M includes a two-way infinite *tape* divided into *cells*, a *reading head* which scans one cell of the tape at a time, and a finite set of internal *states* $Q = \{q_0, q_1, \ldots, q_n\}$, $n \geq 1$. Each cell is either blank (B) or has written on it the symbol 1. In a single step the machine may simultaneously: (1) change from one state to another; (2) change the scanned symbol s to another symbol $s' \in S = \{1, B\}$; and (3) move the reading head one cell to the right (R) or left (L). The operation of M is controlled by a partial map $\delta : Q \times S \to Q \times S \times \{R, L\}$ (which may be undefined for some arguments).

The interpretation is that if $(q, s, q', s', X) \in \delta$ then the machine M in state q, scanning symbol s, changes to state q', replaces s by s', and moves to scan one square to the right if $X = R$ (or left if $X = L$). The map δ viewed as a finite set of quintuples is called a *Turing program*. The *input* integer x is represented by a string of $x + 1$ consecutive 1's (with all other cells blank). The Turing machine is pictured in Figure 1.1.

We begin with M in the *starting state* q_1 scanning the leftmost cell containing a 1, called the *starting cell*. If the machine ever reaches the *halting state* q_0, after say s steps, then we say M *halts* and the *output* y is the total number of 1's on the tape. (Note that $f(x) = \max\{x + 1, s\}$ bounds the maximum distance from the starting cell to any cell which is either scanned or contains an input symbol. Hence the determination of y is effective.)

We may assume that M never makes any further moves after reaching state q_0, i.e., that the domain of δ contains no element of the form (q_0, s). We say that M *computes* the partial function ψ provided that $\psi(x) = y$

Figure: Soare's definition of a Turing Machine

The definition is hard to penetrate! We'll give some analogies and

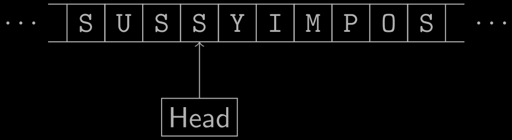
What's a Turing Machine?

Think of a Turing Machine (TM) as a ✨magical computer ✨(doesn't exist in real life!).

What's a Turing Machine?

Think of a Turing Machine (TM) as a ✨magical computer ✨(doesn't exist in real life!).

The TM has infinite memory (yay!). However, the memory is difficult to access, because it is in the form of cells on a linear "tape", with only one read/write head. The read/write head can only move left/right one cell at a time...



What's a Turing Machine?

Note: for the people who remember CSC236 content, it might be helpful to recall what a DFA is before you proceed.

What's a Turing Machine?

Note: for the people who remember CSC236 content, it might be helpful to recall what a DFA is before you proceed.

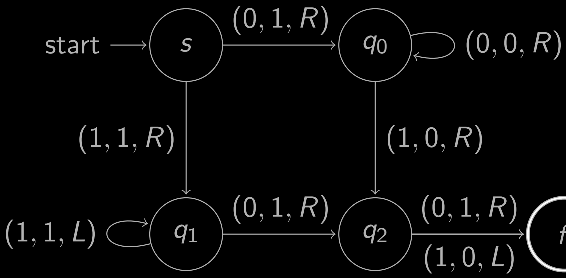
The "processor" of a Turing machine is just a DFA. The TM reads one character from its read/write head, and based on the current state, determines the following behaviour:

What's a Turing Machine?

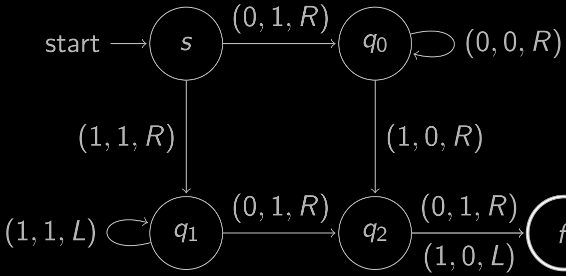
Note: for the people who remember CSC236 content, it might be helpful to recall what a DFA is before you proceed.

The "processor" of a Turing machine is just a DFA. The TM reads one character from its read/write head, and based on the current state, determines the following behaviour:

- What to write back to the cell (if anything).
- Which direction (left or right) to move the read/write head.
- The next state of the DFA.



What's a Turing Machine?



Consider the above DFA, for example. Suppose the TM's processor was in state s.

- If a '0' is read through the read/write head, the following happens:
  - The '0' is overwritten with a '1'.
  - The read/write head moves one cell to the right.
  - The processor transitions to state q_0.
- If instead a '1' is read through the read/write head:
  - The '1' is overwritten with a '1' (i.e. the tape doesn't change).
  - The read/write head moves one cell to the right.
  - The processor transitions to state q_1.

What's a Turing Machine?

Some more notes (also on worksheet):

[1] Some other conventions for the blank symbol are '␣' or '0'.

What's a Turing Machine?

Some more notes (also on worksheet):

- Each Turing Machine must designate a blank character. In our course, the blank character is usually denoted '$\square$'.[1]

[1] Some other conventions for the blank symbol are '␣' or '0'.

What's a Turing Machine?

Some more notes (also on worksheet):

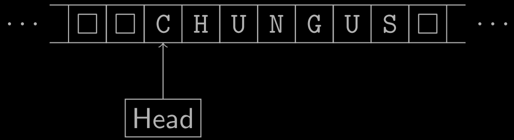
- Each Turing Machine must designate a blank character. In our course, the blank character is usually denoted '□'.[1] The TM starts with blank characters
- Turing Machines can receive string inputs. On input *w* (*w* being some string), place *w* on the tape (at the read/write head) before executing. For example, the TM should look as follows after accepting the input 'CHUNGUS':

[1] Some other conventions for the blank symbol are '␣' or '0'.

What's a Turing Machine?

Some more notes (also on worksheet):

- Each Turing Machine must designate a blank character. In our course, the blank character is usually denoted '□'.[1] The TM starts with blank characters
- Turing Machines can receive string inputs. On input *w* (*w* being some string), place *w* on the tape (at the read/write head) before executing. For example, the TM should look as follows after accepting the input 'CHUNGUS':



- Each Turing Machine has a designated finish state. For us, the finishing state is called 'f'. Upon reaching the finishing state, execution stops, and whatever string remains on the tape (excluding □ characters) is called the output.

[1] Some other conventions for the blank symbol are '␣' or '0'.

Exercise time!

[2] Back in 1936, that is...

Exercise time!

Turing machines were the ✨state of the art ✨(imaginary) computing technology.[2]

[2] Back in 1936, that is...

Exercise time!

Turing machines were the ✨state of the art ✨(imaginary) computing technology.[2]

Now you can cosplay as a Turing machine! **Task:** Complete Exercise 1 on the worksheet.

[2] Back in 1936, that is...

Exercise time!

Let $f : \Sigma^* \to \Sigma^*$ be a function that accepts a string as input, and outputs another string.

Exercise time!

Let $f : \Sigma^* \to \Sigma^*$ be a function that accepts a string as input, and outputs another string.
We say that a Turing machine M computes the function f if given any string $w \in \Sigma^*$:

- $M(w)$ halts,
- $M(w)$ outputs $f(w)$.

Exercise time!

Let $f : \Sigma^* \to \Sigma^*$ be a function that accepts a string as input, and outputs another string.

We say that a Turing machine M computes the function f if given any string $w \in \Sigma^*$:

- $M(w)$ halts,
- $M(w)$ outputs $f(w)$.

Note: Not every Turing machine M computes a function!

Exercise time!

Let $f : \Sigma^* \to \Sigma^*$ be a function that accepts a string as input, and outputs another string.

We say that a Turing machine M computes the function f if given any string $w \in \Sigma^*$:

- $M(w)$ halts,
- $M(w)$ outputs $f(w)$.

Note: Not every Turing machine M computes a function! If for some input $w \in \Sigma^*$ $M(w)$ does not halt (in other words, $M(w)$ loops), then M does not compute a function.

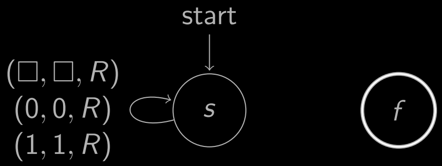
Exercise time!

Let $f : \Sigma^* \rightarrow \Sigma^*$ be a function that accepts a string as input, and outputs another string.

We say that a Turing machine M computes the function f if given any string $w \in \Sigma^*$:

- $M(w)$ halts,
- $M(w)$ outputs $f(w)$.

Note: Not every Turing machine M computes a function! If for some input $w \in \Sigma^*$ $M(w)$ does not halt (in other words, $M(w)$ loops), then M does not compute a function.



Our favourite website

Click here for free ihpone 4 📱📱📱giveaway (WORKING 2023 📅) (NO VIRUS 😷😷😷) (FREE SHIPPING 🚢INTERNATIONAL 🌍) (+200 robux free 🤑🤑) [SOCIAL EXPERIMENT GONE WRONG 🚨🚨POLICE CALLED] (free fortnite dance

and paste e1.txt into the website for receive free ihpone 📱📱📱.

Accepting and rejecting

Now, instead of having a finishing state f, some Turing machines may have an accepting state (denoted q_{acc}) and a rejecting state (denoted q_{rej}).

Accepting and rejecting

Now, instead of having a finishing state f, some Turing machines may have an accepting state (denoted q_{acc}) and a rejecting state (denoted q_{rej}).

For such machines M, upon input w, the following behaviour can happen:

- $M(w)$ reaches the accepting state q_{acc}. In this case, we say M accepts w.
- $M(w)$ reaches a rejecting state q_{rej}. In this case, we say M rejects w.

Accepting and rejecting

Now, instead of having a finishing state f, some Turing machines may have an accepting state (denoted q_{acc}) and a rejecting state (denoted q_{rej}).

For such machines M, upon input w, the following behaviour can happen:

- $M(w)$ reaches the accepting state q_{acc}. In this case, we say M accepts w.
- $M(w)$ reaches a rejecting state q_{rej}. In this case, we say M rejects w.

Accepting and rejecting

Now, instead of having a finishing state f, some Turing machines may have an accepting state (denoted q_{acc}) and a rejecting state (denoted q_{rej}).

For such machines M, upon input w, the following behaviour can happen:

- $M(w)$ reaches the accepting state q_{acc}. In this case, we say M accepts w.
- $M(w)$ reaches a rejecting state q_{rej}. In this case, we say M rejects w.
- $M(w)$ never reaches an accepting or rejecting state. In this case, we say M loops on w.

Exercise time!

Task: Complete Exercise 2 on the worksheet.

Our favourite website

Please return to free ihpone 4 📱📱📱giveaway (WORKING 2023 📅) (NO VIRUS 😷😷😷) (FREE SHIPPING 🚢INTERNATIONAL 🌍) (+200 robux free 🤑🤑) [SOCIAL EXPERIMENT GONE WRONG 🚨🚨POLICE CALLED] (free fortnite dance , and paste e2.txt into the website for receive free ihpone 📱📱📱.

Formal definition of a Turing machine

Before giving the formal definition of a TM, please take some time to prepare yourself psychologically. You can have a brief moment to reflect on this photo that I took somewhere in the world.

Figure: Guess where this photo was taken!

Formal definition of a Turing machine

A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}})$, where

Formal definition of a Turing machine

A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$, where

- Q is a finite set of states.

Formal definition of a Turing machine

A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$, where

- Q is a finite set of states.
- Σ is a finite collection of characters called the input alphabet, with $\square \notin \Sigma$.

Formal definition of a Turing machine

A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}})$, where

- Q is a finite set of states.
- Σ is a finite collection of characters called the input alphabet, with $\square \notin \Sigma$.
- Γ is a finite collection of characters called the tape alphabet, with $\square \in \Gamma$ and $\Sigma \subseteq \Gamma$.

Formal definition of a Turing machine

A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$, where

- Q is a finite set of states.
- Σ is a finite collection of characters called the input alphabet, with $\square \notin \Sigma$.
- Γ is a finite collection of characters called the tape alphabet, with $\square \in \Gamma$ and $\Sigma \subseteq \Gamma$.
- $\delta : Q \times \Gamma \to Q \times \gamma \times \{L, R\}$ is the transition function. Given a state $q \in Q$ and a character $c \in Q$, $\delta(q, c)$ specifies the new state (from Q), the character to write back (from Γ), and the direction to move the read/write head (from $\{L, R\}$).

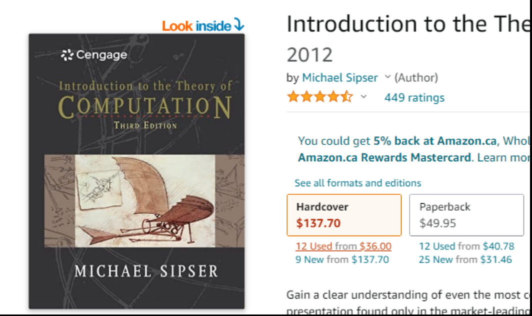
Formal definition of a Turing machine

A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$, where

- Q is a finite set of states.
- Σ is a finite collection of characters called the input alphabet, with $\square \notin \Sigma$.
- Γ is a finite collection of characters called the tape alphabet, with $\square \in \Gamma$ and $\Sigma \subseteq \Gamma$.
- $\delta : Q \times \Gamma \to Q \times \gamma \times \{L, R\}$ is the transition function. Given a state $q \in Q$ and a character $c \in Q$, $\delta(q, c)$ specifies the new state (from Q), the character to write back (from Γ), and the direction to move the read/write head (from $\{L, R\}$).
- q_0 is the start state.
- q_{acc} is the accept state.
- q_{rej} is the reject state.

Our favourite book (well, one of them)

The above definition of a Turing machine can be found in Michael Sipser's Introduction to the Theory of Computation. This is one of the reference books that we will use.



wtf

Please legally purchase a physical copy of this book so that the Cenegage Group can outperform its 2022 yearly revenue of $1.37 billion.

Formal definition of a Turing machine

A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}})$.

Formal definition of a Turing machine

A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}})$.

The TM in Exercise 2 can be formally defined as follows:

- $Q = \{q_0, q_1, q_2, c_1, c_2, r, q_{\mathrm{acc}}, q_{\mathrm{rej}}\}$;
- $\Sigma = \{0, 1\}$;
- $\Gamma = \{\square, 0, 1\}$;
- δ is defined with the following table:

(q, l)	(State, Letter, Direction)
$(q_0, 0)$	$(q_1, \square, R)$
$(q_0, 1)$	$(q_2, \square, R)$
$(q_0, \square)$	$(q_{\mathrm{acc}}, \square, L)$
$(q_0, \#)$	doesn't matter!
$(q_1, 0)$	$(q_1, 0, R)$
$(q_1, 1)$	$(q_1, 1, R)$
$\dots$	$\dots$

Why Turing Machines?

Any function that can be computed by a modern computer (in whatever language, e.g. Python or Assembly) can also be computed by a Turing machine.

Why Turing Machines?

Any function that can be computed by a modern computer (in whatever language, e.g. Python or Assembly) can also be computed by a Turing machine.

Why Turing Machines?

Any function that can be computed by a modern computer (in whatever language, e.g. Python or Assembly) can also be computed by a Turing machine.

Turing machines are a nice way to formalize the idea of an "algorithm"!