Computational Complexity

16 Jan 2014

Today

- Turing machines
- Languages
- Universal Turing Machine
- Polynomial time algorithms and class $\mathbf{P}$

Intuition behind what computation is

1. Read a bit of the input.
2. Read a bit (or possibly a symbol from a slightly larger alphabet, say a digit in the set $\{0, \ldots, 9\}$) from the "scratch pad" or working space we allow the algorithm to use.

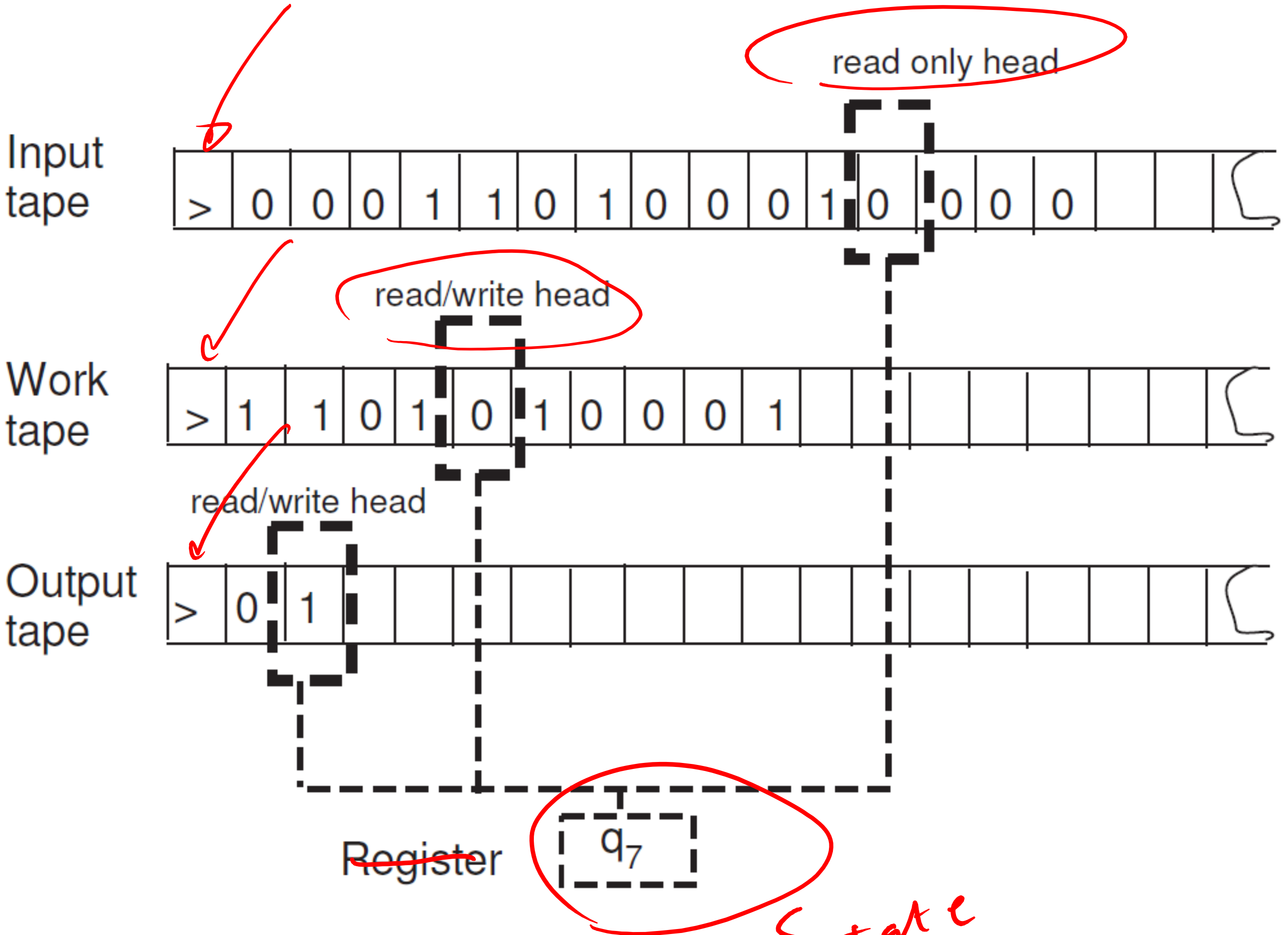
Based on the values read,

3. Write a bit/symbol to the scratch pad.
4. Either stop and output 0 or 1, or choose a new rule from the set that will be applied next.

- Example: multiplying numbers:

abcd
× efgh

Turing Machines



State

always starts in State q_{start}

transition:

① Content of the heads
② internal State

new Content +
new State
new content of heads.

IF			THEN			
input symbol read	work/ output tape symbol read	current state	move input head	new work/ output tape symbol	move work/ output tape	new state
⋮	⋮	⋮	⋮	⋮	⋮	⋮
a	b	q	Right $\rightarrow$	b'	Left $\leftarrow$	q'
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Turing Machines

set of symbols

table of previous slide.

Formal definition. Formally, a TM M is described by a tuple (Γ, Q, δ) containing:

- A finite set Γ of the symbols that M's tapes can contain. We assume that Γ contains a designated "blank" symbol, denoted $\square$, a designated "start" symbol, denoted $\triangleright$ and the numbers 0 and 1. We call Γ the *alphabet* of M.

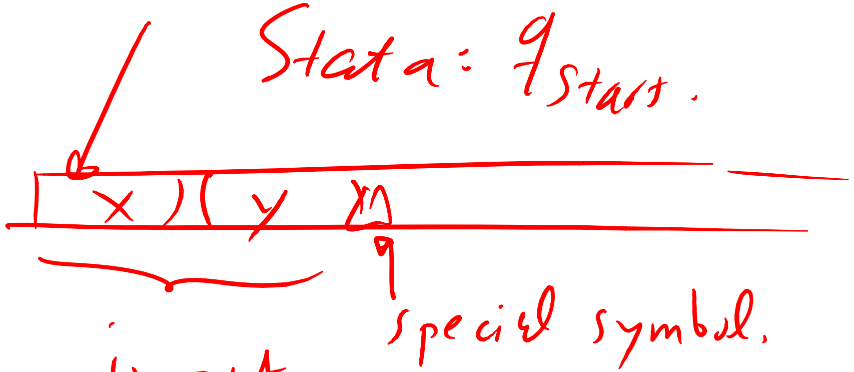
- A finite set Q of possible states M's register can be in. We assume that Q contains a designated start state, denoted q_{start} and a designated halting state, denoted q_{halt}.

- A function $\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{\mathsf{L}, \mathsf{S}, \mathsf{R}\}^k$, where $k \geq 2$, describing the rules M use in performing each step. This function is called the *transition function* of M (see Figure 1.2.)

Variants of Turing Machines

- K tape Turing Machine
  - 1 input tape
  - 1 output tape
  - $K-2$ work tape

- 1 tape Turing Machine:
  - A single tape for input, work, and output



- Amazingly, all the above are "equivalent" (can simulate each other)

Definition 1.3 *(Computing a function and running time)* Let $f : \{0,1\}^* \to \{0,1\}^*$ and let $T : \mathbb{N} \to \mathbb{N}$ be some functions, and let M be a Turing machine. We say that M *computes* f if for every $x \in \{0,1\}^*$, whenever M is initialized to the start configuration on input x, then it halts with $f(x)$ written on its output tape. We say M *computes* f *in* $T(n)$*-time* if its computation on every input x requires at most $T(|x|)$ steps.

e.g., M runs in time $\leq n^2$: the output for every input of length n is computed in time $O(n^2)$

Languages

- Simplest form of computational "tasks" / "problems":
  Given an input $x \in \{0,1\}^n$ decide if x is "acceptable" or not

$f:\{0,1\}^* \to \{0,1\}$

An important special case of functions mapping strings to strings is the case of *Boolean* functions, whose output is a single bit. We identify such a function f with the subset $L_f = \{x : f(x) = 1\}$ of $\{0,1\}^*$ and call such sets *languages* or *decision problems* (we use these terms interchangeably).[1] We identify the computational problem of computing f (i.e., given x compute $f(x)$) with the problem of deciding the language L_f (i.e., given x, decide whether $x \in L_f$).

- Example: Given a graph: is it connected?
- Example: Given a number: is it prime?
- Example: Does a given TM M halt over input x in time t ?

Turing Machines as Their Own Input!

- A Turing Machine $M = (Q, \Gamma, \delta)$ can be described in bits.

over $\{0,1\}^*$

- Let $f : \{0,1\}^* \to \{0,1\}^*$ be defined as a ~~(partial)~~ function that takes (M, x) and outputs $M(x)$

- Important Theorem: There is a Turing Machine U that computes $f(\cdot)$

$U_f(M,x) = M(x)$

interpreter

- More Important Theorem:
  U's running time is not "much more" than $M(x)$

U_f

(M, x)

emulate$(M(x))$

Efficient Universal Turing Machine

Theorem 1.9 *(Efficient Universal Turing machine)*
There exists a TM $\mathcal{U}$ such that for every $x, \alpha \in \{0,1\}^*$, $\mathcal{U}(x, \alpha) = M_\alpha(x)$, where M_α denotes the TM represented by α.

Moreover, if M_α halts on input x within T steps then $\mathcal{U}(x, \alpha)$ halts within $CT \log T$ steps, where C is a number independent of $|x|$ and depending only on M_α's alphabet size, number of tapes, and number of states.

$L = \{ \langle M, x, t \rangle \mid M(x)$ halts in t steps$\}$

- Recall : Is there a TM that finds out: Does a given TM M halt over input x in time t ?

Theorem 1.9 *(Efficient Universal Turing machine)*
There exists a TM $\mathcal{U}$ such that for every $x, \alpha \in \{0,1\}^*$, $\mathcal{U}(x, \alpha) = M_\alpha(x)$, where M_α denotes the TM represented by α.

Moreover, if M_α halts on input x within T steps then $\mathcal{U}(x, \alpha)$ halts within $CT \log T$ steps, where C is a number independent of $|x|$ and depending only on M_α's alphabet size, number of tapes, and number of states.

- Answer to question above: YES

What can be solved in time $T(\cdot)$?

$DTime(n^{10})$

Definition 1.12 *(The class* **DTIME**.*)* Let $T : \mathbb{N} \to \mathbb{N}$ be some function. A language L is in $\mathbf{DTIME}(T(n))$ iff there is a Turing machine that runs in time $c \cdot T(n)$ for some constant $c > 0$ and decides L. ◊

Definition 1.13 *(The class* **P***)*
$\mathbf{P} = \cup_{c \geq 1} \mathbf{DTIME}(n^c)$

Examples of Languages in P

is $t = 2^{\ell}$ a polynomial over $|t| \simeq \log(t)$

- Given an input $x \in \{0,1\}^n$ decide if x is "acceptable" or not
  - Example: Given a graph: is it connected?
  - Example: Given a number: is it prime?

Primes is in P

  - Example: Does a given TM M halt over input x in time t ?

input (M, x, t) 1^t

Algorithm: Universal TM : run M over x and see if it halts in t steps.

A closer model to out own computers?

Define a *RAM Turing machine* to be a Turing machine that has *random access memory*. We formalize this as follows: the machine has an infinite array A that is initialized to all blanks. It accesses this array as follows. One of the machine's work tapes is designated as the *address tape*. Also the machine has two special alphabet symbols denoted by $\mathtt{R}$ and $\mathtt{W}$ and an additional state we denote by q_{access}. Whenever the machine enters q_{access}, if its address tape contains $\llcorner i \lrcorner\mathtt{R}$ (where $\llcorner i \lrcorner$ denotes the binary representation of i) then the value $A[i]$ is written in the cell next to the $\mathtt{R}$ symbol. If its tape contains $\llcorner i \lrcorner\mathtt{W}\sigma$ (where σ is some symbol in the machine's alphabet) then $A[i]$ is set to the value σ.

- Can be shown that RAM and TM are equivalent up to a polynomial time slow-down.

Next Time

- Nondeterministic Computation
- Class NP
- Reduction
- NP Completeness