3. Turing Machine

Turing (1936) was the first to give a rigorous definition.

Common algorithms:

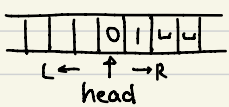
1. Addition, subtraction, multiplication, division
2. Euclidean algorithm for the greatest common divisor
3. Quicksort
4. Dijkstra's algorithm for shortest path

gcd(210, 45) = gcd(45, 30)
210 ÷ 45 = 4 ····· 30
45 ÷ 30 = 1 ····· 15
= gcd(30, 15) = 15

One dictionary gives this definition: an algorithm is a mechanical process to be followed in calculations or other problem-solving operations

Turing 1936 paper: On Computable Numbers, with an Application to Entscheidungs problem

Entscheidungs problem (Hilbert): Is there a mechanical process (algorithm) that can determine, in a finite number of steps, whether any given statement in first-order logic is true? X



$\in \Sigma$ $\quad \delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ (r/w)

Def 3.1 (Turing Machine) A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where ($q_{end}$)

1. Q is a finite set of states
2. Σ is the input alphabet not containing the blank symbol $\sqcup$
3. Γ is the tape alphabet, where $\Sigma \subseteq \Gamma$ and $\sqcup \in \Gamma$

可以用到的字符, 比 Σ 多

4. $\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ is the transition function.

5. $q_0 \in Q$ is the start state

6. $q_{accept} \in Q$ is the accept state
7. $q_{reject} \in Q$ is the reject state

} q_{end} is the end state

Example. 1) Binary add one, least significant bit first.

Input: 1011 ␣

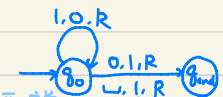
Output: 0111 ␣

Input: 11 ␣ ␣

Output: 001 ␣

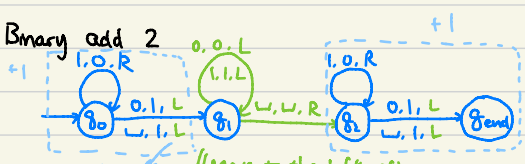
Input: 0 ␣

Output: 1 ␣



q_0: 要进着+1的状态

2) Binary add 2

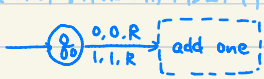


// Move to the leftmost

因为q_2要不停左移指针，遇到␣停

所以+1时要保证目前不在␣，+1完之后左移结束即可.

或:



最低位+2即不变，只要 +1即可

高 低

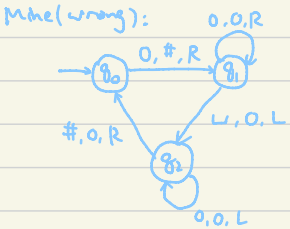
Exercise.

1) Unary add one

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \sqcup, \#, \cdots\}$

Mine (wrong):



00

⊔ ⊔

↑ ↑

⊔ ⊔

Input: 0^k

Output: 0^{k+1}

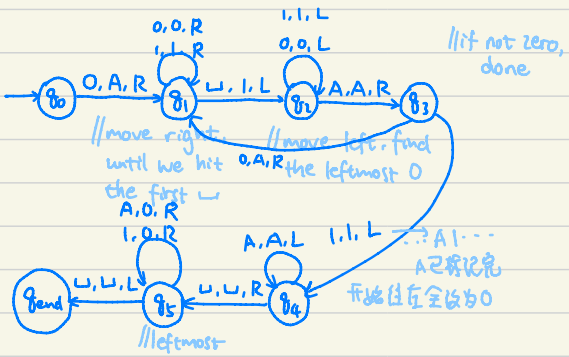
2) Duplicate zeros

Input: 0^k

Output: 0^{2k}

Input: 00⊔

Output: 0000⊔

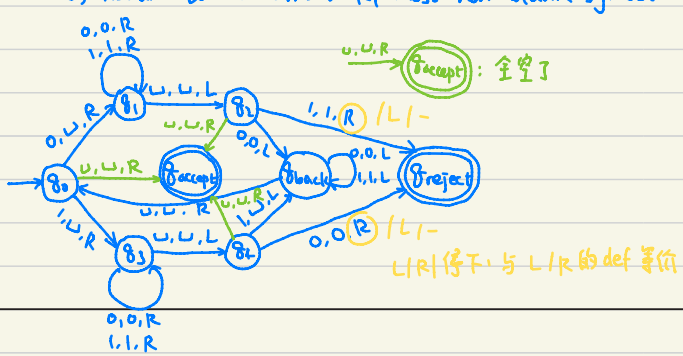


Example. Decide if the input string is a palindrome, i.e. $w = w^R$

Idea: 1) Read the leftmost symbol, memorize it, and erase it.

2) Move to the rightmost end and read the symbol. If it does not match the leftmost symbol, reject

3) Return to the current leftmost non-blank symbol and repeat.



Turing Machine 与 C语言(eg.)程序能实现的一样，复杂度差一个多项式

P/NP定义不改变，仍高效

Example. Decide $L=\{0^{2^n} : n\geq 0\}$

An algorithm = A TM

Idea:

1) Sweep from left to right, crossing out every second 0

2) If exactly one 0 remains on the tape, accept.

3) If more than one 0 remains and their count is odd, reject.

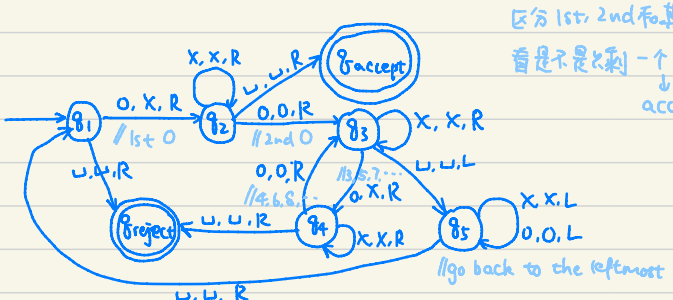
4) Move the head back to the left of the tape.

5) Repeat from Step 1).

区分 1st, 2nd和其他是为了看是不是比到一个0了.

↓

accept

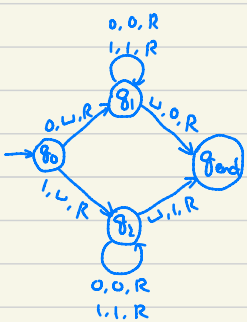


Exercise.

1) Cyclic shift

Input: ␣ $X_1 X_2 \cdots X_n$ ␣

Output: ␣ $X_2 X_3 \cdots X_n X_1$ ␣



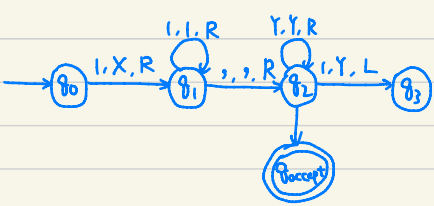
2) Unary comparison

Input: 1^x, 1^y ␣

Accept iff $x \geq y$

111, 11 ␣

XX···1, YY···1 ␣



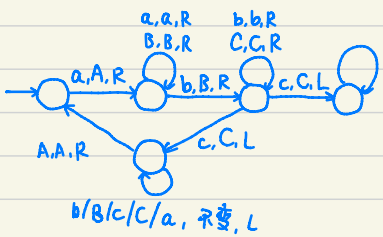
Input: 111, 11␣

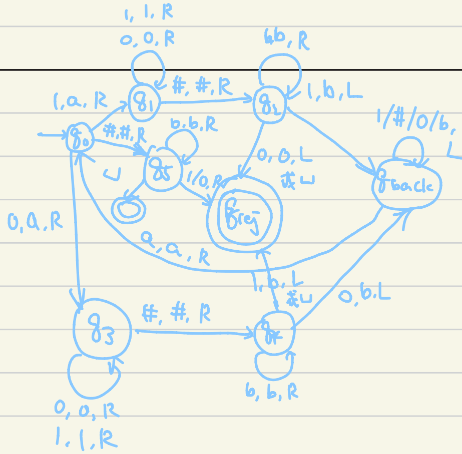
Accept

3) $L = \{0^n 1^n : n \geq 1\}$

4) $L = \{a^n b^n c^n : n \geq 1\}$

AAaBBbCCc




1, 1, R
0, 0, R
b,b, R
1,a, R
#, #, R
1, b, L
#,#, R
b,b,R
1/#/0/b, L
q0
q1
q2
q5
1/0, R
0, 0, L
或 ⊔
qback
qrej
⊔
0, a, R
a, a, R
1, b, L
或 ⊔
0, b, L
#, #, R
q3
0, 0, R
1, 1, R
b, b, R

Def 3.2 Let $L \subseteq \{0,1\}^*$. Let M be a TM. Say TM M decides L in time $T(n)$. If for every $x \in \{0,1\}^*$,
(1) M halts in $T(|x|)$ steps
(2) If $x \in L$, then M accepts x.
(3) If $x \notin L$, then M rejects x.
We call a language decidable if there is a TM that decides it.

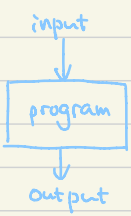
可判定：更强 一定√/× 都是
可接受：不一定全都√/×

Def 3.3 The set of strings that a TM M accepts is called the language recognized by M, denoted by L(M).

Def 3.4 Let $L \subseteq \{0,1\}^*$. We call L (Turing) recognizable if there is a TM such that the language recognized by M is L, i.e., $L(M) = L$.

Every decidable language is recognizable; the converse is not true.

停机问题
$L_{HALT} = \{\langle M \rangle : M \text{ halts on input } \varepsilon\}$. $\langle M \rangle$: encoding of M
L_{HALT} is recognizable, but not decidable



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

$\{0,1\}^* \to \{0,1\}^*$ 是不可数的，和R一样多
但程序和Z一样多

可以被计算的函数

Def 3.5 (computable function) Let $T: \mathbb{N} \to \mathbb{N}$.
Let $f: \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ be a partial function, where $\perp$ denotes "undefined". Say that a TM M computes f in time $T(n)$ if, for every $x \in \{0,1\}^*$, whenever $f(x) \neq \perp$, M halts with $f(x)$ on its tape after at most $T(n)$ steps.

3.2 Variants of Turing Machine

The original definition and its reasonable variants have the same computational power — they compute/decide/recognize the same class of languages. Moreover, their runtimes differ only by a polynomial factor.

如何知道已经读过k个：设置k个状态，之间转移

Lemma 3.6 (change of alphabet) If $L \subseteq \{0,1\}^*$ is decidable in time $T(n)$ by a TM on alphabet Γ, then L is decidable in time $O(\log|\Gamma| \cdot T(n))$ by a TM on alphabet $\Gamma' = \{0, 1, \sqcup\}$.

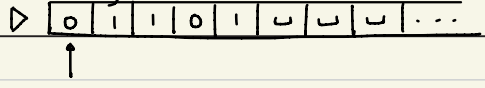
$= O_\Gamma(T(n))$

Proof. Encode each symbol in Γ using $k = \lceil \log_2 |\Gamma| \rceil$ bits. To simulate one step of M, we do the following:

1) Spend k steps to read a symbol $a \in \Gamma$, and determine the symbol b to be written.
2) Overwrite a with b
3) Move the head to the next symbol by k steps
4) Transition to the next state

Simulating one step takes $O(k)$ steps. Total running time is $O(k) \cdot T(n) = O(kT(n)) = O(\underbrace{\lceil \log_2 |\Gamma| \rceil}_{= O(\log|\Gamma|)} \cdot T(n)) = O(\log|\Gamma| \cdot T(n))$ □

One-way TM:



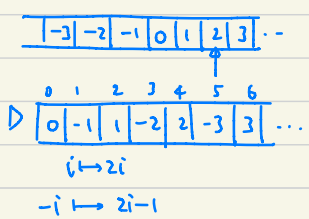
require: $\delta(q, \triangleright) = (q', \triangleright, R)$

Def 3.7 (One-way TM) An one-way TM is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$.

1. Q is the set of states
2. Σ is the alphabet
3. Γ is the tape alphabet, where $\sqcup, \triangleright \in \Gamma$ and $\Sigma \subseteq \Gamma$. Here $\sqcup$ is the blank symbol, and $\triangleright$ is the left-end marker symbol.
4. δ $\quad \delta(q, \triangleright) = (q', \triangleright, R)$
5. q_0
6. q_{accept}
7. q_{reject}

Lem 3.8 Let $L \subseteq \{0,1\}^*$. If L is decidable by a (two-way) TM in time $T(n)$, then L is decidable by a one-way TM in time $O(T(n))$ over the same alphabet.

P≠NP based encryption

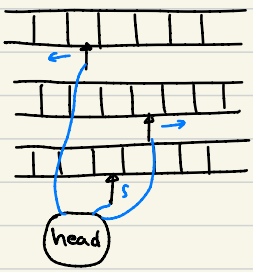


Proof. Index the one-way tape by $\mathbb{N}$.
Index the two-way tape by $\mathbb{Z}$.
Map position i to $2i, i \geq 0$, map position $-i$ to $2i-1, i \geq 1$
Let M denote the two-way TM. To simulate one step of M, the new one-way TM M' performs the following.

1) Read the current symbol.
2) Determine the next state
3) Write the new symbol
4) Move two cells (left or right)

5) Enters the next state

Each simulation takes $O(1)$ steps. The total simulation takes $O(1) \cdot T(n) = O(T(n))$. □



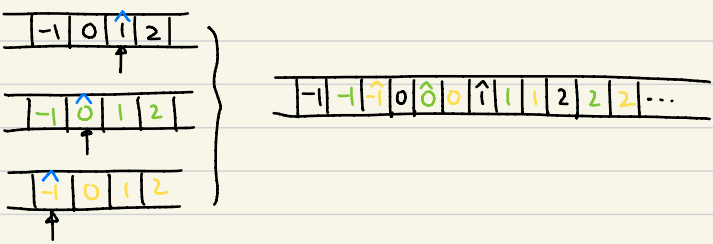
Def 3.9 (Multiple TM) A k-tape TM is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

where: $\delta: Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{L, R, S\}$

A multitape TM is a k-tape TM, where $k = O(1)$.

Lemma 3.10 (simulation of multitape TM by single-tape TM)

Let $L \subseteq \{0,1\}^*$. If L is decidable by a k-tape TM in time $T(n)$, then it is decidable in time $O(T(n)^2)$ by a single-tape TM. → where $T(n) \to \infty$



Each of the k simulated tapes is map to a set of cells on the single tape:

1) tape 1 uses: $0, \pm k, \pm 2k, \cdots$
2) tape 2 uses: $1, k+1, -k+1, 2k+1, -2k+1, \cdots$
3) In general, tape i ($k \geq i \leq k$) uses $i-1, k+i-1, -k+i-1, 2k+i-1, -2k+i-1, \cdots$

For every symbol $a \in \Gamma$, introduce two symbols in Γ' a and $\hat{a}$.
The hatted symbol $\hat{a}$ indicates the position of the head.
To simulate a single step, the new machine

有k!个排列组合(帽子), 但可用有限个状态描述.

??? → ?0?

第2个hat读到, 为0

3个不知道hat是谁

1) Read: M' sweeps the tape from left to right to read the k symbols (with a hat). 读k个带帽子的
2) Determine action by δ.
3) Update: Sweep from right to left to replace the symbols and move the hats.

(读hat; k个; 每个来回走一次)

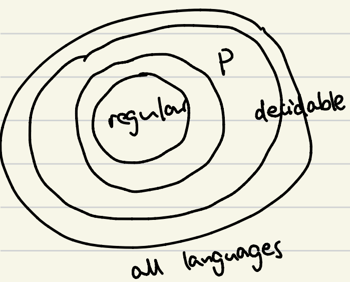
One step simulation takes $O(T(n)) + O(1) + O(T(n) + k \cdot 2k) = O(T(n))$

Total $O(T(n)) \cdot T(n) = O(T(n)^2)$ $T(n) \to \infty$ □.

复杂性类

Def 3.11 (DTIME) Let $T: \mathbb{N} \to \mathbb{N}$. Language $L \subseteq \{0,1\}^*$ is in $DTIME(T(n))$ if there is a multitape TM M that decides L in time $O(T(n))$.

Def 3.12 $P = \bigcup_{c \geq 1} DTIME(n^c)$. 多项式时间内可判定



Remark. The definition of P does not change if we replace multitape TM by single-tape TM, RAM TM etc.

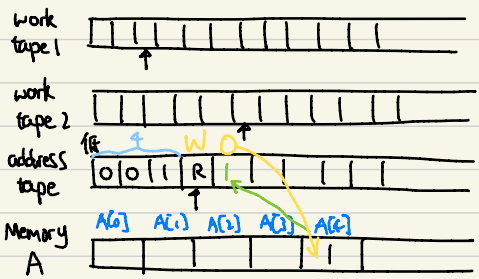
Def 3.13 (RAM TM) A random access memory TM (RAM TM) is a Turing Machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}, Q_{access} \subseteq Q)$, with random access memory, defined as follows:
1) M has an infinite memory tape A indexed by $\mathbb{N}$.
2) M has an address tape.
3) Γ contains two special symbols: R (read) and W (write).

R: 把内存中东西读到 tape上
i: 地址
W: 写入内存 i:地址 σ: 要写的

4) Special access states $Q_{access} \subseteq Q$. Whenever M enters a state $q \in Q_{access}$:

a) If address tape contains i R, then the symbol $A[i]$ is written to the next cell of R.

b) If $\cdots iW\sigma$, then $A[i]$ is set to σ.

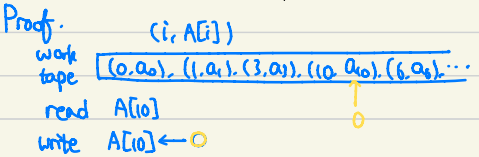


(Williams 25)

$DTIME(T(n)) \subseteq SPACE(\sqrt{T(n)}\log n)$

Lemma 3.14 Let $L \subseteq \{0,1\}^*$. If L is decidable by a RAM TM in time $T(n)$, then L is decidable by a multiple TM in time $O(T(n)^3)$. Moreover, if the address used is of length $O(1)$, then L is decidable by a multiple TM in time $O(T(n)^2)$.

Proof.



Use an additional tape as memory, consisting of pairs $(i, A[i])$. To simulate one step of M (in access state), the new machine M' scans the tape A to locate the address i, if no such entry exists, create

a new pair $(i, A[i])$. Then, read or write $A[i]$ accordingly.

Each step takes $O(\text{length of } A) = O(\#\text{address} \times \text{address length})$ ($O(T(n))$, $O(1)$)

$= O(T(n) \cdot T(n)) = O(T(n)^2)$

Total $O(T(n)^3)$. ($O(T(n)^2)$)

C++, Python $T(n)$

↓

RAM TM $\longrightarrow$ multitape $\longrightarrow$ singletape

$O(T(n))$ $O(T(n)^3)$ $O(T(n)^6)$

Variants of TM

Single-tape TM { one-way infinite
two-way infinite

multitape TM (complexity community)

RAM TM (algorithm community)

Ignoring polynomial factors, all variants are equivalent.

Simulate assembly language by a RAM TM.

Take x86-64 assembly language for example.

16-bit registers, plus various control, ⁉️栈指针, for system and vector operations. Put each register on a work tape.

RAM TM

Categories of x86-64 instructions:

1) Data movement instruction: MOV RAX, RBX // RAX = RBX

Constant time (64 bits)

MOV [RDI], AL // M[RDI] = AL

2) Arithmetic instructions

Constant time (64 bits)

ADD RAX, RBX // RAX = RAX + RBX

SUB RAX, 5 // RAX = RAX - 5

3) Logic and bitwise instructions

AND RAX, RBX // Bitwise AND, RAX = RAX & RBX.

XOR RAX, RAX // RAX = RAX ^ RAX = 0

4) Comparison and test instruction

CMP RAX, RBX // Set flags based RAX - RBX

eg. 1 if ... >= 0
0 if ... < 0

5) Control flow instruction

JMP label // jump conditionally 状态转移

JE equal_case // jump if ZF=1

6) Stack instructions

用内存模拟. PUSH, 写内存

PUSH RAX // Push RAX onto stack, RSP -= 8

POP RBX // Pop 8 bytes from the stack into RBX (64次读内存)

7) Floating-point instruction

Each instruction can be implemented by a RAM TM in time O(1).

C++ → Assembly instruction → RAM TM → multitape TM → single-tape TM

T(n) O(T(n)) O(T(n)) (log bit addr) $O(T(n)^2)$ $O(T(n)^4)$

3.3 Church-Turing thesis

"Every function that can be physically computed can be computed by a Turing machine." // 普遍认为是对的

The strong CT thesis: ..., with polynomial overhead. // 不太对

eg. 量子计算机.

Possible exception: quantum computers: integer factorization, discrete log can be solved by quantum computers in polynomial time

一旦做到这些, 一些加密算法不安全

private key: $k \le 2^{256}-1$ public key: G^k 在椭圆曲线上的乘法

$k \le p$ 安全性依赖于从 G^k 求 k 难, 比 P≠NP 更强