3 Algorithm and Turing Machines

3.1 Turing machines

An algorithm is a mechanical process to be followed in calculations or other problem-solving operation.

***e.g.* 3.1.**

1. *addition,substraction,multiplication,division*
2. ***Euclidean algorithm** for gcd:* $gcd(210, 25) = gcd(45, 30) = gcd(30, 15) = gcd(15, 0)$
3. *selection quicksort*
4. *Dijkstra's algorithm for shortest path*

In 1936 for the first time, Alan Turing rigorously defined algorithms

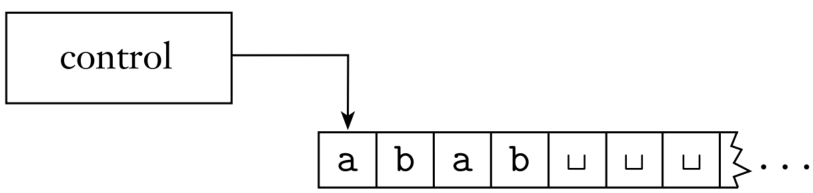


Figure 1: *Schematic of a Turing machine*

1. *It uses an infinite tape as its unlimited memory*
2. *It has a tape head that can read and write symbols and move around on the tape*

***def* 3.1.** *(Turing Machine)*

*A **Turing machine** is a 7-tuple,* $(Q, \Sigma, \Gamma, \delta, q_0, \underbrace{q_{accept}, q_{reject}}_{q_{end}})$*, where:*

1. Q *is a set of states*
2. Σ *is the input alphabet not containing the blank symbol* $\sqcup$
3. Γ *is the tape alphabet, where* $\sqcup \in \Gamma$ *and* $\Sigma \subseteq \Gamma$
4. $\delta : Q \times \Gamma \to Q \times \{L, R\}$ *is the transive 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, where* $q_{accept} \neq q_{reject}$

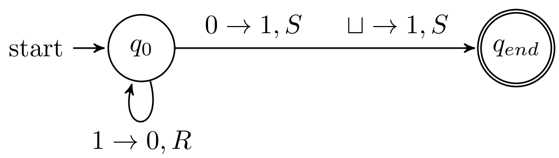
***e.g.* 3.2.**

1. *Input a binary number n (least significant bit first), output n+1*

   Input: $101 \sqcup \sqcup$

   Output: $011 \sqcup \sqcup$

   remark. *this is a computation problem(6-tuple)*



2. *decide if* $w \in \{0,1\}^*$ *is a palindrome, i.e.* $w = w^R$

3. *decide* $L = \{0^{2^n}, n \geq 0\}$*, where* $\Sigma = \{0\}$

   IDEA

   (a) *If there is a single 0, accept it*

   (b) *Sweep left to right across the tape, crossing off every other 0.*

   (c) *If the tape contained more than a single 0 and the number of 0s was odd, reject*

   (d) *Return the head to the left-hand end of the tape*

   (e) *go to step1*

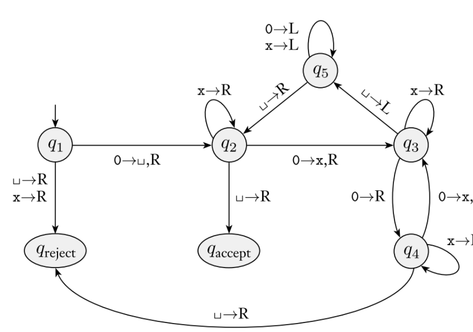


Figure 2: *State diagram for Turing machine M2*

4. $L = \{w\#w, w \in \{0,1\}^*\}$

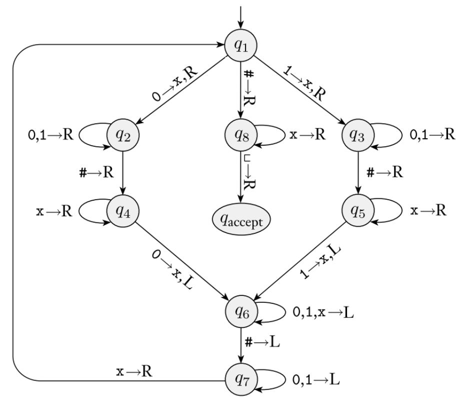


Figure 3: *State diagram for Turing machine M1*

***exercises* 3.1.**

1. *Binary comparison.* $L = \{x, y | x, y \in \{0,1\}^* x \geq y$, *most significantbit first, no leading 0 unless the number is 0*$\}$

2. *Binary add 1*

3. $L = \{0^n 1^n\}$

***def* 3.2.** *Let* $L \subset \{0,1\}^*$, *M be a Turing Machine. Say M decides L in time T(n) if for* $\forall x \in \{0,1\}^*$*:*

1. *M halts in T(n) steps*

2. *If* $x \in L$, *then M accepts* x

3. *If* $x \notin L$, *then M rejects* x

***def* 3.3.** *Let* $L \subseteq \{0,1\}^*$. *Call L(Turing Machine) decidable if there is a Turing Machine decides it*

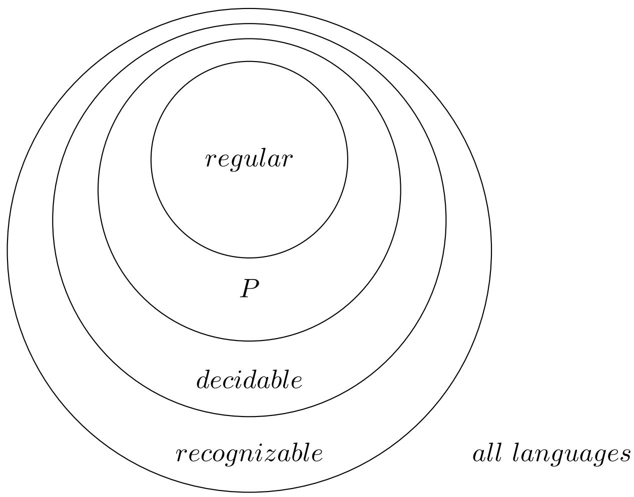
remark. *On an input x, a TM may accept, reject, or loop forever*

*In **definition 3.2**, the machine should never loop forever*

***def* 3.4.** *Let M be a TM, the set of strings that M accepts is the language recognized by M, denoted by L(M)*

***def* 3.5.** *Let* $L \subseteq \{0,1\}^*$. *Call L(Turing) recognizable if there is some TM recognizes it*

remark. *Obviously, every decidable language is recognizable. While the converse is not true, e.g.* $L = \{< M, x >, M \text{ halts on } x\}$



***def* 3.6.** *Let* $f : \{0,1\}^* \to \{0,1\}^* \cup \{undefined\}$. *Say TM M computes f in time T(n) if for* $\forall x \in \{0,1\}^*$, *with* $f(x) \neq undefined$, *M halts with f(x) on its tape in at most* $T(|x|)$ *steps*

What is algorithm? An algorithm is A Turing Machine. Despite its simplicity, it is capable of implementing any computer algorithm.
"Everything should be made as simple as possible, but no simpler."

3.2 Variants of Turing Machines

***lemma* 3.7.** *If language* $L \subseteq \{0,1\}^*$ *is decidable in time T(n) by a Turing Machine on alphabet* Γ, *then it is decidable in time* $O(log|\Gamma|T(n)) = O_\Gamma(T(n))$ *by a Turing Machine on alphabet* $\Gamma = \{0, 1, \sqcup, \triangleright\}$

Proof. *Encode any symbol in* Γ *using* $k = \lceil log_2|\Gamma|\rceil = O(log|\Gamma|)$ *bits. To simulate one step of M, the new Turing Machine will:*

1. *use k steps to read a symbol* $a \in \Gamma$
2. *transit to next step q', and get the new symbol b(to overwrite a)*
3. *overwrite a by b*
4. *go left or right for k steps, or stay*

In total, the simulation (for one step) takes less than k+1+k+k = O(k) $\square$

***def* 3.8.** *A k-tape Turing Machine M is a 7-tuple* $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$*, where:*

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

Usually, the first tape is the input tape, the last tape is the output tape, and the remaining tapes are work tapes.

A multiple Turing Machine is an O(1)-tape Turing Machine.

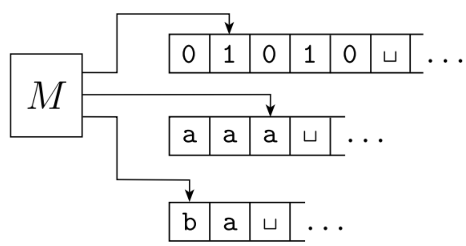


Figure 4: *3-tape Turing Machine*

***lemma* 3.9.** *Let* $L \subseteq \Sigma^*$*, If L is decidable by a k-tape Turing Machine in time T(n), then it is decidable by a single-tape Turing Machine in time* $O(kT(n)^2) \neq O(T(n)^2)$

Proof. *Use location* $i-1, k+i-1, 2k+i-1, ...$ *to store the contents of the* i^{th} *tape, where* $i = 1, 2, ..., k$. *For* $\forall a \in \Gamma$*, introduce* $a, \widehat{a} \in \Gamma$*, where* $\widehat{a}$ *denotes the location of the head.*

To simulate one step of M, single-tape Turing Machine M' will:

1. *sweep the tape from left to right to read k symbols marked by* $\widehat{}$
2. *apply M's transition function* δ *to determine the next state*
3. *sweep back from right to left to update k symbols, if needed, and move* $\widehat{}$ *if needed*

In total, 1.2.3. take T(n)+1+O(kT(n)) = O(kT(n)) $\square$

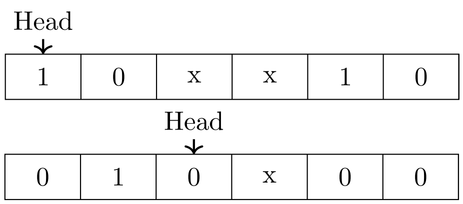


Figure 5: A two tape Turing Machine

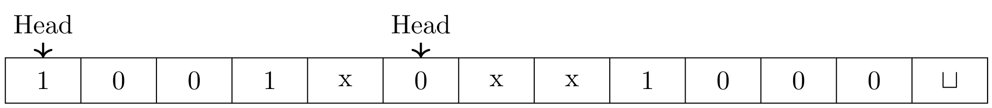


Figure 6: Representing two tapes with one

*A **Bidirectional-tape Turing Machine** is a Turing Machine whose tape is infinite in both directions*

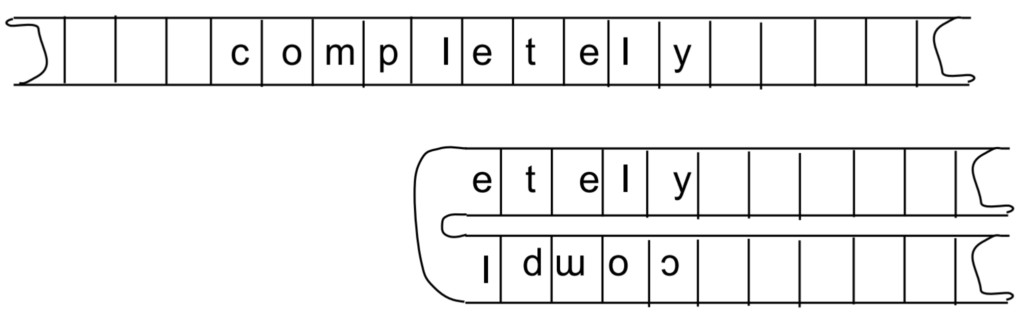


Figure 7: *A bidirectional-tape Turing Machine*

***lemma* 3.10.** *Let $L \subseteq \Sigma^*$. If L is decidable by a bidirectional-tape Turing Machine in $T(n)$, then it is decidable by a single-tape Turing Machine in $O(T(n))$ time.*

Proof. *Index the biderectional tape by $\mathbb{Z}$, map location i to*

$$\begin{cases} 2i, i \geq 0 \\ -2i-1, i < 0 \end{cases}$$

For every step of M, M' will

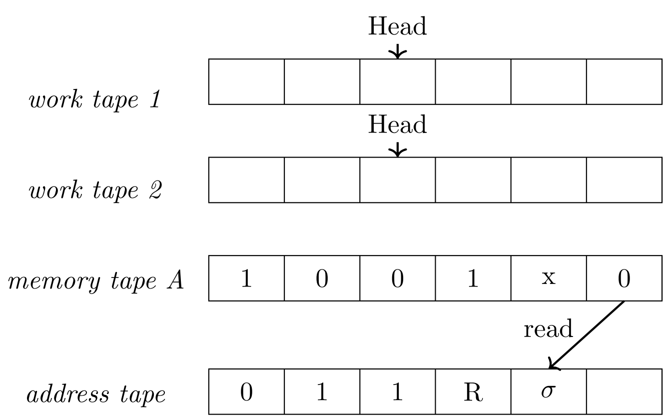
1. *read the symbol*
2. *transit to the next state*
3. *update the symbol*
4. *move left or right for two steps, if needed*

It takes O(1) to simulate one step. In total, the running time is O(T(n)) $\square$

***def* 3.11.** *Random access memory(RAM) Turing machine is a TM with random access memory:*

1. *M has an infinite memory tape A indexed by N*
2. *One of M's tapes is the address tape*
3. *Γ contains two speed symbols R(read) and W(write)*
4. *Q has some special states* $Q_{access} \subseteq Q$
   Whenever M gets into a state $q \in Q_{access}$
   (a) *If the address tape contains iR, the value* $A[i]$ *is written to the cell next to R*
   (b) *If the address tape contains* $iW\sigma$, *then set* $A[i]$ *to symbol* σ

Assume the TAM Turing Machine M has k work tapes and an address.
Then $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}, Q_{access})$, $\delta : Q \times \Gamma^{k+1} \to Q \times \Gamma^{k+1} \times \{L, R, S\}^{k+1}$



***lemma* 3.12.** *Let $L \subseteq \{0,1\}^*$. If L is decidable by an RAM Turing Machinein time T(n), then it is decidable by a multi-tape Turing Machine in time $O(T(n)^3)$. Moreover, if the length of the address is O(1), then L is decidable by a multi-tape Turing Machine in $O(T(n)^2)$*

Proof. *Use an extra work tape as memory that contains pairs $(i, A[i])$, where i is an integer in binary, $A[i] \in \Gamma$, for all memory addresses that have been referred to.*

To simulate one step of M, if M is in an access state, the new multi-tape Turing Machine M' will:

1. scans tape A to find an address that matches i in the address tape
2. if i does not exist, add a new pair $(i, A[i])$
3. read or write $A[i]$ accordingly

1,2,3 take $O(T(n)^2) + O(T(n)) + O(T(n)) = O(T(n)^2)$

In total, M' runs in time $T(n) * O(T(n)) = O(T(n)^3)$

If address length is $O(1)$, then $\#pairs \leq T(n)$, length of each pair is $O(1)$,1,2,3 take O(T(n)) steps to simulate □

***remark*.** *Ignoring polynomial factors, all Turing Machine variants are equivalent.*

$$\underbrace{C++}_{T(n)} \rightarrow \underbrace{AssemblyLanguage}_{O(T(n))} \rightarrow \underbrace{RAMTM}_{O(T(n))} \rightarrow \underbrace{multitapeTM}_{O(T(n^3))(O(T(n^2)))} \rightarrow \underbrace{single-tapeTM}_{O(T(n^6))(O(T(n^4)))}$$

***def* 3.13.** *Let $T : \mathbb{N} \rightarrow \mathbb{N}$, language $L \subseteq \{0,1\}^*$ is in **DTIME**(T(N)) iff there exists a multi-tape Turing Machine M that decides L in time O(T(n))*

***def* 3.14.** $P = \bigcup_{c \geq 1} DTIME(n^c)$