

Def: For $U = \mathbf{N}$ or Σ_0^* , let $f : U \rightarrow U$ be a total or partial function. We say that f is a **partial, recursive function** iff \exists TM $M(f = M(\cdot))$, i.e., $\forall w \in U (f(w) = M(w))$.

Def: For $S \subseteq U$, S is a **recursive set** iff the function χ_S is a (total) recursive function,

$$\chi_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{otherwise} \end{cases} \quad \text{the characteristic function of } S$$

S is a **recursively enumerable set** (S is **r.e.**) iff the function p_S is a (partial) recursive function,

$$p_S(x) = \begin{cases} 1 & \text{if } x \in S \\ \nearrow & \text{otherwise} \end{cases} \quad \text{the partial characteristic function of } S$$

Def: The **set accepted by Turing machine M** is defined as follows:

$$\mathcal{L}(M) = \{w \in U \mid M(w) = 1\} .$$

Prop: A set is r.e. iff it is accepted by some Turing machine

Prop: If S is recursive then S is r.e.

Def: $\text{co-r.e.} = \{S \mid \bar{S} \in \text{r.e.}\},$

i.e., $S \in \text{co-r.e.}$ iff $\bar{S} \in \text{r.e.}.$

Th: $\text{Recursive} = \text{r.e.} \cap \text{co-r.e.}$

Definition 5.1 A string $w \in \Sigma^*$ is a *palindrome* iff it is the same as its reversal, i.e., $w = w^R$. □

Examples of palindromes:

- 101
- 1101001011
- ABLE WAS IERE I SAW ELBA
- A MAN A PLAN A CANAL PANAMA

Proposition 5.2 *The set of PALINDROMES (over a fixed alphabet, Σ) is a recursive set.*

Proof: [Verbal sketch:]



□

Fact 5.3 *Time $O(n^2)$ is necessary and sufficient for a one-tape Turing machine to accept the set, PALINDROMES.*

Proof: Sufficiency is obvious. To prove necessity do problems 2.8.4, 2.8.5 from [P]. □

Definition 5.4 A k -tape Turing machine, $M = (Q, \Sigma, \delta, s)$

Q : finite set of states; $s \in Q$

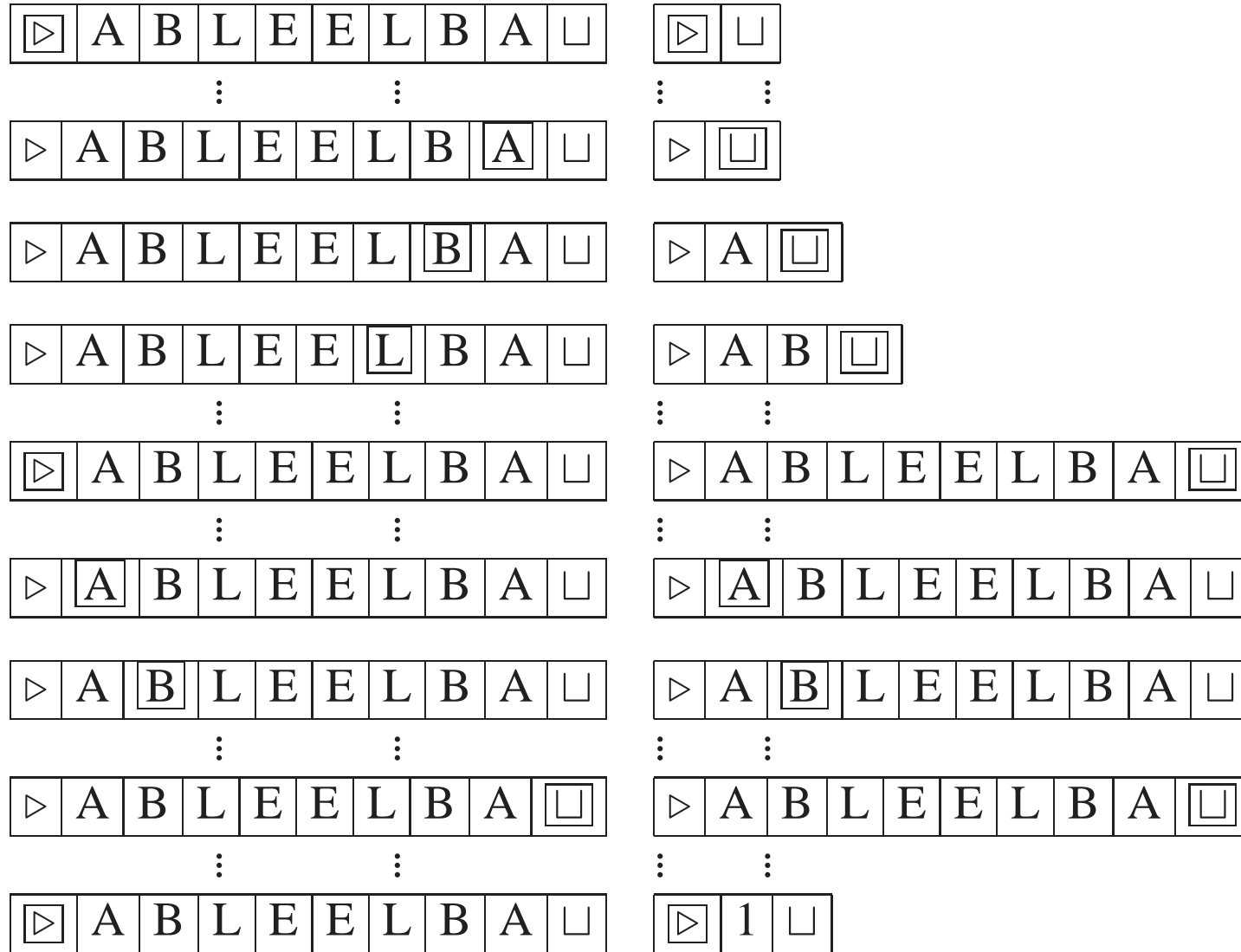
Σ : finite tape alphabet;

$\delta: Q \times \Sigma^k \rightarrow (Q \cup \{h\}) \times (\Sigma \times \{\leftarrow, \rightarrow, -\})^k$

□

Proposition 5.5 *PALINDROMES can be accepted in $\mathbf{DTIME}[n]$ on a 2-tape TM.*

Proof: (that PALINDROMES \in **DTIME** $[n]$)



□

Time and Space functions: $t, s : \mathbf{N} \rightarrow \mathbf{N}^+$

Definition 5.6 A set $A \subseteq U$ is in **DTIME** $[t(n)]$ iff there exists a deterministic, multi-tape TM, M , and a constant c , such that,

1. $A = \mathcal{L}(M) \equiv \{w \in U \mid M(w) = 1\}$, and
2. $\forall w \in U$, $M(w)$ halts within $c \cdot t(|w|)$ steps. □

Definition 5.7 A set $A \subseteq U$ is in **DSPACE** $[s(n)]$ iff there exists a deterministic, multi-tape TM, M , and a constant c , such that,

1. $A = \mathcal{L}(M)$, and
2. $\forall w \in U$, $M(w)$ uses at most $c \cdot s(|w|)$ work-tape cells.

(Input tape is “read-only” and not counted as space used.) □

Example: PALINDROMES \in **DTIME** $[n]$, **DSPACE** $[n]$.

In fact, as we will see later, PALINDROMES \in **DSPACE** $[\log n]$.

Definition 5.8 $f : U \rightarrow U$ is in $F(\mathbf{DTIME}[t(n)])$ iff there exists a deterministic, multi-tape TM, M , and a constant c , such that,

1. $f = M(\cdot)$;
2. $\forall w \in U$, $M(w)$ halts within $c \cdot t(|w|)$ steps;
3. $|f(w)| \leq |w|^{O(1)}$, i.e., f is polynomially bounded.

□

Definition 5.9 $f : U \rightarrow U$ is in $F(\mathbf{DSPACE}[s(n)])$ iff there exists a deterministic, multi-tape TM, M , and a constant c , such that,

1. $f = M(\cdot)$;
2. $\forall w \in U$, $M(w)$ uses at most $c \cdot s(|w|)$ work-tape cells;
3. $|f(w)| \leq |w|^{O(1)}$, i.e., f is polynomially bounded.

(Input tape is “read-only”; Output tape is “write-only”. Neither is counted as space used.) □

Example: $\text{Plus} \in F(\mathbf{DTIME}[n])$, $\text{Mult} \in F(\mathbf{DTIME}[n^2])$

$$\mathbf{L} \equiv \mathbf{DSPACE}[\log n]$$

$$\mathbf{P} \equiv \mathbf{DTIME}[n^{O(1)}] \equiv \bigcup_{i=1}^{\infty} \mathbf{DTIME}[n^i]$$

$$\mathbf{PSPACE} \equiv \mathbf{DSPACE}[n^{O(1)}] \equiv \bigcup_{i=1}^{\infty} \mathbf{DSPACE}[n^i]$$

These classes will become your good friends during the last third of this course.

I just wanted to introduce them briefly today.

Theorem 5.10 For any functions $t(n) \geq n$, $s(n) \geq \log n$, we have

$$\begin{aligned} \mathbf{DTIME}[t(n)] &\subseteq \mathbf{DSPACE}[t(n)] \\ \mathbf{DSPACE}[s(n)] &\subseteq \mathbf{DTIME}[2^{O(s(n))}] \end{aligned}$$

Proof: Let M be a $\mathbf{DSPACE}[s(n)]$ TM, let $w \in \Sigma_0^*$, let $n = |w|$

$M(w)$ has k tapes and uses at most $cs(n)$ work-tape cells.

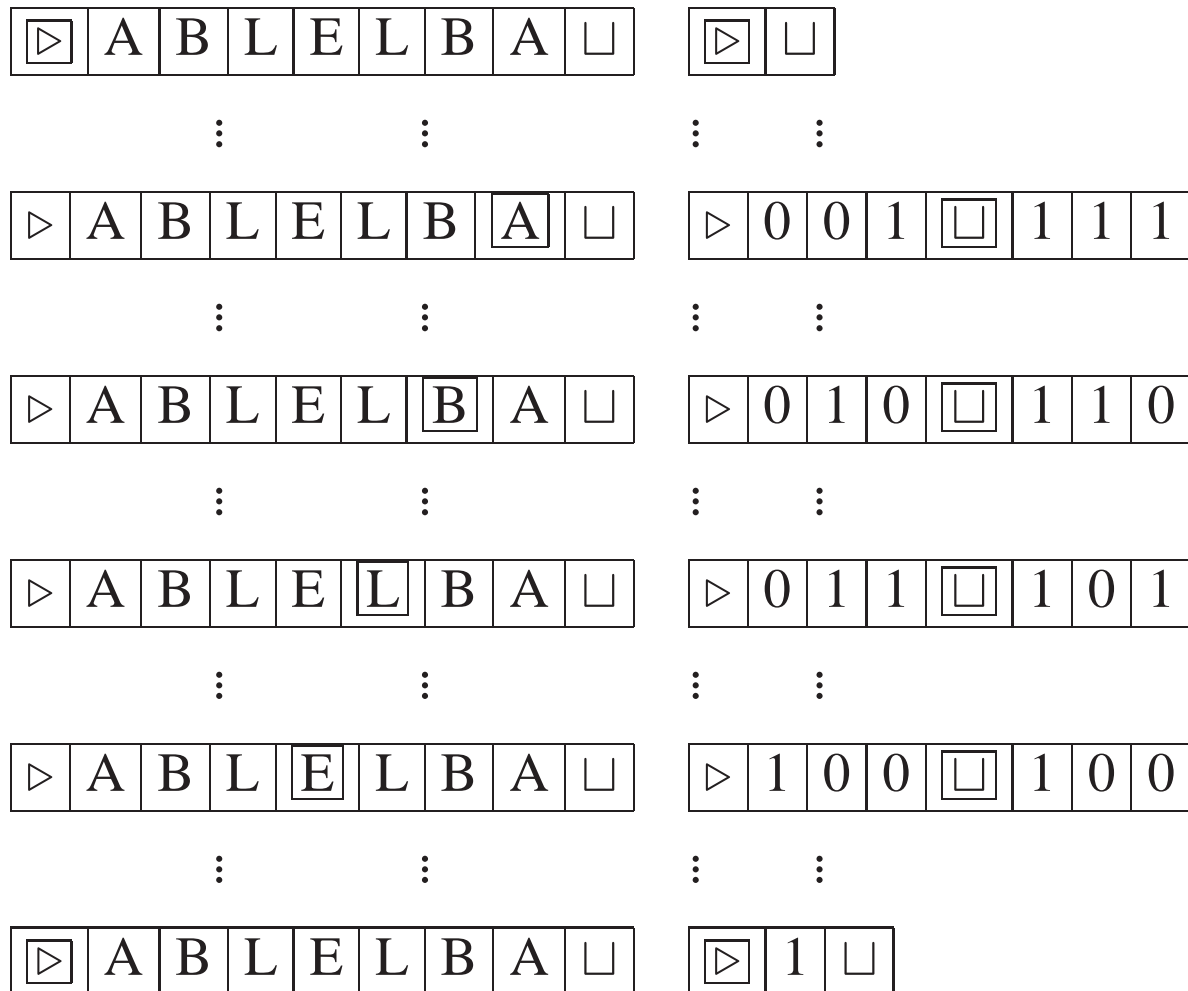
$M(w)$ has at most $2^{k's(n)}$ possible configurations:

$$|Q| \cdot (n + cs(n) + 2)^k \cdot |\Sigma|^{cs(n)} < 2^{k's(n)}$$

of states · # of head positions · # of tape contents

Thus, after $2^{k's(n)}$ steps, $M(w)$ must be in an infinite loop. □

Corollary 5.11 $\mathbf{L} \subseteq \mathbf{P} \subseteq \mathbf{PSPACE}$



Using $O(\log n)$ workspace, we can keep track of and manipulate two pointers into the input.

RAM = Random Access Machine

Memory:

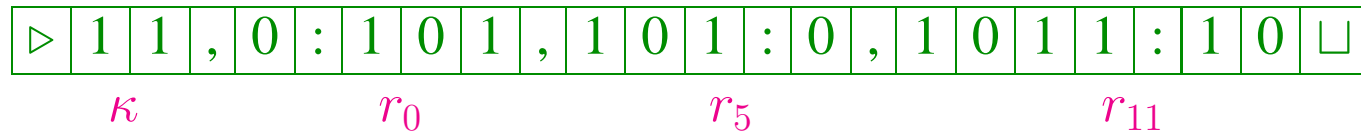
κ	r_0	r_1	r_2	r_3	r_4	\dots	r_i	\dots
----------	-------	-------	-------	-------	-------	---------	-------	---------

κ = program counter; r_0 = accumulator

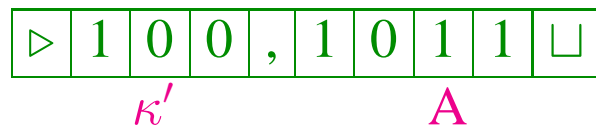
Instruction	Operand	Semantics
READ	$j \mid \uparrow j \mid = j$	$r_0 := (r_j \mid r_{r_j} \mid j)$
STORE	$j \mid \uparrow j$	$(r_j \mid r_{r_j}) := r_0$
ADD	$j \mid \uparrow j \mid = j$	$r_0 := r_0 + (r_j \mid r_{r_j} \mid j)$
SUB	$j \mid \uparrow j \mid = j$	$r_0 := r_0 - (r_j \mid r_{r_j} \mid j)$
HALF		$r_0 := \lfloor r_0/2 \rfloor$
JUMP	j	$\kappa := j$
JPOS	j	if $(r_0 > 0)$ then $\kappa := j$
JZERO	j	if $(r_0 = 0)$ then $\kappa := j$
HALT		$\kappa := 0$

Theorem 5.12 $\mathbf{DTIME}[t(n)] \subseteq \mathbf{RAM-TIME}[t(n)] \subseteq \mathbf{DTIME}[(t(n))^3]$

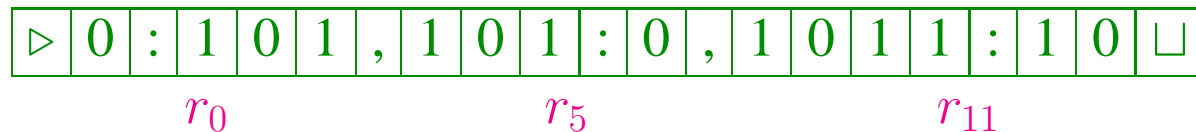
Proof: Memorize program in finite control. Store all registers on one tape:



Store workspace for calculations on second tape:



Use the third tape for copying and pasting sections of the first tape.



Each register contains at most $n+t(n)$ bits.

[$O(\log n)$ would be more realistic.]

The total number of tape cells used is at most $2t(n)(n + t(n)) = O((t(n))^2)$.

Each step takes at most $O((t(n))^2)$ steps to simulate.

□

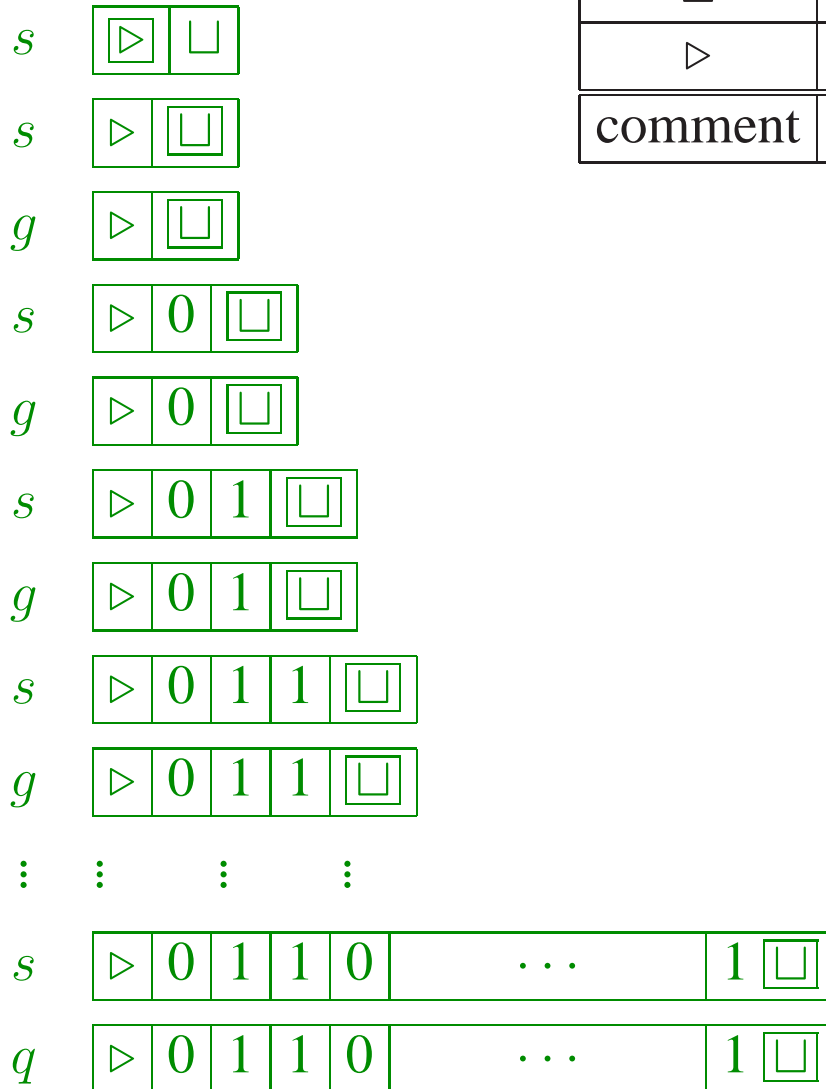
Nondeterministic Turing Machines choose one of two possible moves each step.

guess.tm	s	g	q
0			
1			
\sqcup	$g, \sqcup, - \mid q, \sqcup, -$	$s, 0, \rightarrow \mid s, 1, \rightarrow$	
\triangleright	$s, \triangleright, \rightarrow$		
comment	g or q	guess 0 or 1	the rest

Nondeterministic Guess Machine is a typical example:

- Write down an arbitrary string, $g \in \{0, 1\}^*$: the guess.
- Proceed with the rest of the computation, using g if desired.
- Accept iff there exists some guess that leads to acceptance.

guess.tm	s	g	q
0			
1			
\sqcup	$g, \sqcup, - \mid q, \sqcup, -$	$s, 0, \rightarrow \mid s, 1, \rightarrow$	
\triangleright	$s, \triangleright, \rightarrow$		
comment	g or q	guess 0 or 1	the rest



Definition 5.13 The set accepted by a NTM, N :

$$\mathcal{L}(N) \equiv \{w \in U \mid \text{some run of } N(w) \text{ halts with output "1"}\}$$

The **time** taken by N on $w \in \mathcal{L}(N)$ is the **number of steps** in the **shortest computation** of $N(w)$ that accepts. □

Unfortunately, this is a mathematical fiction.

As far as we know, you can't **really** build a nondeterministic Turing Machine.

NTIME $[t(n)] \equiv$ problems accepted by NTMs in time $O(t(n))$

$$\mathbf{NP} \equiv \mathbf{NTIME}[n^{O(1)}] \equiv \bigcup_{i=1}^{\infty} \mathbf{NTIME}[n^i]$$

Theorem 5.14 *For any function $t(n)$,*

$$\mathbf{DTIME}[t(n)] \subseteq \mathbf{NTIME}[t(n)] \subseteq \mathbf{DSPACE}[t(n)] \subseteq \mathbf{DTIME}[2^{O(t(n))}]$$

Corollary 5.15 $\mathbf{L} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$

Corollary 5.16 *The definition of **Recursive** and **r.e.** are unchanged if we use non-deterministic instead of deterministic Turing machines.*

