Turing Machines: \[ M = (Q, \Sigma, \delta, s) \]

\[ \delta : Q \times \Sigma \rightarrow (Q \cup \{h\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\} \]

**Def:** Function \( f \) is *recursive* iff it is computed by a TM. \( f \) may be total or partial.

**Def:** A set \( S \) is *recursive* iff its characteristic function \( \chi_S \) is a recursive function.

*Recursive* is the set of recursive sets.

A set \( S \) is *recursively enumerable (r.e.)* iff its partial characteristic function \( p_S \) is a recursive function.

*r.e.* is the set of r.e. sets.

**Th:** 

*Recursive* = r.e. \( \cap \) 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 I ERE I SAW ELBA
- AMANAPLANACANALPANAMA

Fact 5.2 The set of PALINDROMES (over a fixed alphabet, $\Sigma$ is context-free but not regular.)
Proposition 5.3 The set of PALINDROMES (over a fixed alphabet, $\Sigma$) is a recursive set.

Proof:

```
A B L E E L B A □
```

Fact 5.4 Time $O(n^2)$ is necessary and sufficient for a one-tape Turing machine to accept the set, PALINDROMES.

Proof: Time $O(n^2)$ suffices. One way to see this is to do problems 2.8.4, 2.8.5 from [P].
Definition 5.5 A $k$-tape Turing machine, $M = (Q, \Sigma, \delta, s)$

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

$\Sigma$: finite set of symbols;

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

Proposition 5.6 Palindromes can be accepted in $\text{DTIME}[n]$ on a 2-tape TM.
Proof: (that PALINDROMES $\in$ DTIME$[n]$)

\[
\begin{array}{c}
\square A \quad B \quad L \quad E \quad E \quad L \quad B \quad A \quad \square \\
\vdots \quad \vdots \\
\square A \quad B \quad L \quad E \quad E \quad B \quad A \quad \square \\
\vdots \quad \vdots \\
\square A \quad B \quad L \quad E \quad E \quad L \quad B \quad A \quad \square \\
\vdots \quad \vdots \\
\square A \quad B \quad L \quad E \quad L \quad E \quad B \quad A \quad \square \\
\vdots \quad \vdots \\
\square A \quad B \quad L \quad E \quad L \quad E \quad L \quad B \quad A \quad \square \\
\vdots \quad \vdots \\
\square A \quad B \quad L \quad E \quad L \quad E \quad L \quad B \quad A \quad \square \\
\vdots \quad \vdots \\
\square A \quad B \quad L \quad E \quad L \quad E \quad L \quad B \quad A \quad \square \\
\vdots \quad \vdots \\
\square A \quad B \quad L \quad E \quad L \quad E \quad L \quad B \quad A \quad \square \\
\vdots \quad \vdots \\
\square 1 \quad \square \\
\end{array}
\]
Definition 5.7 A set $A \subseteq \Sigma^*$ is in $\text{DTIME}[t(n)]$ iff there exists a deterministic, multi-tape TM, $M$, and a constant $c$, such that,

1. $A = \mathcal{L}(M) \equiv \{w \in \Sigma^* \mid M(w) = 1\}$, and

2. $\forall w \in \Sigma^*, M(w)$ halts within $c(1 + t(|w|))$ steps.
Definition 5.8 A set $A \subseteq \Sigma^*$ is in $\text{DSPACE}[s(n)]$ iff there exists a deterministic, multi-tape TM, $M$, and a constant $c$, such that,

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

(Note: The input tape is read-only and not counted as space used. Otherwise space bounds below $n$ would rarely be useful. But in the real world we often want to limit space and work with read-only input.)

Example: $\text{PALINDROMES} \in \text{DTIME}[n], \text{DSPACE}[n]$. In fact, $\text{PALINDROMES} \in \text{DSPACE}[\log n]$. 
Definition 5.9  \( f : \Sigma^* \to \Sigma^* \) is in \( F(\text{DTIME}[t(n)]) \) iff there exists a deterministic, multi-tape TM, \( M \), and a constant \( c \), such that,

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

Definition 5.10  \( f : \Sigma^* \to \Sigma^* \) is in \( F(\text{DSPACE}[s(n)]) \) iff there exists a deterministic, multi-tape TM, \( M \), and a constant \( c \), such that,

1. \( f = M(\cdot) \);
2. \( \forall w \in \Sigma^*, M(w) \) uses at most \( c(1 + s(|w|)) \) 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:  \( \text{Plus} \in F(\text{DTIME}[n]) \), \( \text{Times} \in F(\text{DTIME}[n^2]) \)
L \equiv \text{DSPACE}[\log n]

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

\text{PSPACE} \equiv \text{DSPACE}[n^{O(1)}] \equiv \bigcup_{i=1}^{\infty} \text{DSPACE}[n^i]
Theorem 5.11 For any functions \( t(n) \geq n, s(n) \geq \log n \), we have

\[
\text{DTIME}[t(n)] \subseteq \text{DSPACE}[t(n)] \\
\text{DSPACE}[s(n)] \subseteq \text{DTIME}[2^{O(s(n))}]
\]

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

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

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

possible configurations.

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

\( \spadesuit \)

**Corollary 5.12** \( \text{L} \subseteq \text{P} \subseteq \text{PSPACE} \)
Using $O(\log n)$ workspace, we can keep track of and manipulate two pointers into the input.
RAM = Random Access Machine

Memory: \[ \kappa | r_0 | r_1 | r_2 | r_3 | r_4 | \cdots | r_i | \cdots \]

\( \kappa = \) program counter; \( r_0 = \) accumulator

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operand</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ j ( \uparrow j ) ( = j )</td>
<td>( r_0 := (r_j</td>
<td>r_{r_j}</td>
</tr>
<tr>
<td>STORE ( j ) ( \uparrow j )</td>
<td>( (r_j</td>
<td>r_{r_j}) := r_0 )</td>
</tr>
<tr>
<td>ADD ( j ) ( \uparrow j ) ( = j )</td>
<td>( r_0 := r_0 + (r_j</td>
<td>r_{r_j}</td>
</tr>
<tr>
<td>SUB ( j ) ( \uparrow j ) ( = j )</td>
<td>( r_0 := r_0 - (r_j</td>
<td>r_{r_j}</td>
</tr>
<tr>
<td>HALF</td>
<td></td>
<td>( r_0 := \lfloor r_0/2 \rfloor )</td>
</tr>
<tr>
<td>JUMP ( j )</td>
<td></td>
<td>( \kappa := j )</td>
</tr>
<tr>
<td>JPOS ( j )</td>
<td></td>
<td>if ( r_0 &gt; 0 ) then ( \kappa := j )</td>
</tr>
<tr>
<td>JZERO ( j )</td>
<td></td>
<td>if ( r_0 = 0 ) then ( \kappa := j )</td>
</tr>
<tr>
<td>HALT</td>
<td></td>
<td>( \kappa := 0 )</td>
</tr>
</tbody>
</table>
Theorem 5.13

\( \text{DTIME}[t(n)] \subseteq \text{RAM-TIME}[t(n)] \subseteq \text{DTIME}[(t(n))^3] \)

**Proof:** Memorize program in finite control.

Store all registers on one tape:

\[
\begin{array}{c|c|c|c}
\kappa & r_0 & r_5 & r_{11} \\
\hline
1 & 1 & 0 & 1 \\ 
0 & 1 & 0 & 1 \\ 
1 & 0 & 1 & 1 \\
\end{array}
\]

Store workspace for calculations on second tape:

\[
\begin{array}{c|c}
\kappa' & A \\
\hline
1 & 0 & 0 \\ 
1 & 0 & 1 & 1 \\
\end{array}
\]

Use the third tape for moving over sections of the first tape.

\[
\begin{array}{c|c|c|c}
r_0 & r_5 & r_{11} \\
\hline
0 & 1 & 0 & 1 \\ 
1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 \\
\end{array}
\]

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

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.

<table>
<thead>
<tr>
<th>guess.tm</th>
<th>s</th>
<th>g</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>g, □, ←</td>
<td>q, □, ←</td>
<td>s, 0, →</td>
</tr>
<tr>
<td>△</td>
<td>s, △, →</td>
<td></td>
<td></td>
</tr>
<tr>
<td>comment</td>
<td>g or q</td>
<td>guess 0 or 1</td>
<td>the rest</td>
</tr>
</tbody>
</table>

- 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.
<table>
<thead>
<tr>
<th>guess.tm</th>
<th>$s$</th>
<th>$g$</th>
<th>$q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\square$</td>
<td>$g, \square, -$</td>
<td>$q, \square, -$</td>
<td>$s, 0, \rightarrow$</td>
</tr>
<tr>
<td>$\triangleright$</td>
<td>$s, \triangleright, \rightarrow$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Comment:**

- $g$ or $q$
- Guess 0 or 1
- The rest

**Diagram:**

```
s
 g
 s
 0
 1
 g
 s
 0
 1
 g
 s
 0 1
 g
 0 1
 s
 0 1 1
 g
 0 1 1

... ...

s
 0 1 1 0 ...
 1
 q
 0 1 1 0 ...
 1
```
**Definition 5.14** The set accepted by a NTM, $N : \mathcal{L}(N) \equiv \{w \in \Sigma^* \mid \text{some run of } N(w) \text{ halts with output "1"} \}$

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. ♦
NTIME[t(n)] \equiv \text{probs. accepted by NTMs in time } O(t(n))

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

\textbf{Theorem 5.15} For any function \( t(n) \),

\text{DTIME}[t(n)] \subseteq \text{NTIME}[t(n)] \subseteq \text{DSPACE}[t(n)]

Recall: \( \text{DSPACE}[t(n)] \subseteq \text{DTIME}[2^{O(t(n))}] \)

\textbf{Corollary 5.16}

\( L \subseteq P \subseteq \text{NP} \subseteq \text{PSPACE} \)

\textbf{Corollary 5.17} The definition of Recursive and r.e. are unchanged if we use nondeterministic instead of deterministic Turing machines.