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

**Definition 5.1** A set $A \subseteq U$ is in $\text{DTIME}[t(n)]$ iff there exists a deterministic, multi-tape TM, $M$, and a constant $c$, such that,

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

**Definition 5.2** A set $A \subseteq U$ is in $\text{DSPACE}[s(n)]$ iff there exists a deterministic, multi-tape TM, $M$, and a constant $c$, such that,

1. $A = L(M)$, 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:** $\text{PALINDROMES} \in \text{DTIME}[n], \text{DSPACE}[n]$.

In fact, $\text{PALINDROMES} \in \text{DSPACE}[\log n]$. [Exercise]
**Definition 5.3** $f : U \to U$ is in $F(DTIME[t(n)])$ 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 5.4** $f : U \to U$ is in $F(DSPACE[s(n)])$ 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:** Plus $\in F(DTIME[n])$, Mult $\in F(DTIME[n^2])$
\[ L \equiv \text{DSPACE}[\log n] \]

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

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

These classes will become your good friends soon.
Theorem 5.5  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 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 $2^{k's(n)}$ possible configurations:

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

# of states \cdot # of head positions \cdot # of tape contents

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

Corollary 5.6  $L \subseteq P \subseteq \text{PSPACE}$
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</td>
<td>$j</td>
<td>\uparrow j</td>
</tr>
<tr>
<td>STORE</td>
<td>$j</td>
<td>\uparrow j$</td>
</tr>
<tr>
<td>ADD</td>
<td>$j</td>
<td>\uparrow j</td>
</tr>
<tr>
<td>SUB</td>
<td>$j</td>
<td>\uparrow j</td>
</tr>
<tr>
<td>HALF</td>
<td></td>
<td>$r_0 := \lfloor r_0/2 \rfloor$</td>
</tr>
<tr>
<td>JUMP</td>
<td>$j$</td>
<td>$\kappa := j$</td>
</tr>
<tr>
<td>JPOS</td>
<td>$j$</td>
<td>if $(r_0 &gt; 0)$ then $\kappa := j$</td>
</tr>
<tr>
<td>JZERO</td>
<td>$j$</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 5.7  \( \text{DTIME}[t(n)] \subseteq \text{RAM-TIME}[t(n)] \subseteq \text{DTIME}[(t(n))^3] \)

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

\[
\begin{array}{c}
\kappa \\
r_0 \\
r_5 \\
r_{11}
\end{array}
\begin{array}{c}
1 \\
1 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
h \\
1
\end{array}
\begin{array}{c}
0 \\
1 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
\square
\end{array}
\]

Store workspace for calculations on second tape:

\[
\begin{array}{c}
\kappa' \\
A
\end{array}
\begin{array}{c}
1 \\
0 \\
h \\
1
\end{array}
\begin{array}{c}
0 \\
1 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
\square
\end{array}
\]

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

\[
\begin{array}{c}
r_0 \\
r_5 \\
r_{11}
\end{array}
\begin{array}{c}
1 \\
1 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
h \\
1
\end{array}
\begin{array}{c}
0 \\
1 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
h \\
1
\end{array}
\begin{array}{c}
1 \\
0 \\
\square
\end{array}
\]

Each register contains at most \( n + t(n) \) bits. \([O(\log n)]\text{ 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. \( \square \)
Nondeterministic Turing Machines choose one of two possible moves each step.

<table>
<thead>
<tr>
<th>guess.tm</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>s, ⊥, →</td>
<td>g, ⊥, →</td>
<td>s, 0, →</td>
</tr>
<tr>
<td>q, ⊥, →</td>
<td>s, ⊥, →</td>
<td>s, 1, →</td>
</tr>
<tr>
<td>⊢</td>
<td>g or q</td>
<td>guess 0 or 1</td>
</tr>
</tbody>
</table>

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.
<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, , , \rightarrow )</td>
<td>( q, \square, , , \rightarrow )</td>
<td>( s, 0, \rightarrow )</td>
</tr>
<tr>
<td>( \rightarrow )</td>
<td>( s, \rightarrow )</td>
<td>( \rightarrow )</td>
<td>( \rightarrow )</td>
</tr>
</tbody>
</table>

comment \( g \) or \( q \)    guess 0 or 1  the rest
**Definition 5.8** 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.
\[ s \rightarrow t(n) \]

- \( b_1 \)
- \( b_2 \)
- \( b_3 \)
- \( \cdots \)
- \( b_{t(n)} \)
NTIME\[t(n)\] ≡ problems accepted by NTMs in time \(O(t(n))\)

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

**Theorem 5.9** For any function \(t(n)\),

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

**Corollary 5.10** \(L \subseteq P \subseteq \text{NP} \subseteq \text{PSPACE}\)

**Corollary 5.11** The definition of Recursive and r.e. are unchanged if we use nondeterministic instead of deterministic Turing machines.
NSPACE[\(s(n)\)] is the set of problems accepted by NTMs using at most \(O(s(n))\) space on each branch. [Can run in time \(t(n) \leq 2^{O(s(n))}\).]
Definition 5.12 \( \text{REACH} = \{G \mid s \rightarrow t\} \)

Prop: \( \text{REACH} \in \text{NL} = \text{NSPACE}[\log n] \)

1. \( b := s \)
2. \( \textbf{for } c := 1 \textbf{ to } n = |V| \textbf{ do } \{ \)
3. \( \textbf{if } b = t \textbf{ then accept } \)
4. \( a := b \)
5. \( \textbf{choose new } b \)
6. \( \textbf{if } (\neg E(a, b)) \textbf{ then reject } \} \)
7. \( \text{reject} \)
**Def:** Problem $T$ is **complete** for complexity class $C$ iff

1. $T \in C$, and
2. $\forall A \in C \ (A \leq T)$

Reductions now must be in $F(L)$. 