**Computational Complexity**

**TM representation**

using a state diagram

\[
\begin{array}{c}
M \text{ accepts palindromes over } \{a,b\} \\
\text{upper path for processing } a, \text{ lower for } b \\
\text{check } aR a \text{ repeat, } (aR a)^n \text{ case in } q_9, \text{ odd accepted in } q_9, \text{ even length in } q, \text{ based on } \lambda.
\end{array}
\]

**Computation of } M**

<table>
<thead>
<tr>
<th>Length 0</th>
<th>Length 1</th>
<th>Length 2</th>
<th>Length 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8, B, B</td>
<td>8, B, B</td>
<td>8, B, B</td>
<td>8, B, B</td>
</tr>
<tr>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
</tr>
<tr>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
</tr>
<tr>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
</tr>
<tr>
<td>8, B, B</td>
<td>8, B, B</td>
<td>8, B, B</td>
<td>8, B, B</td>
</tr>
<tr>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
</tr>
<tr>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
</tr>
<tr>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
</tr>
<tr>
<td>8, B, B</td>
<td>8, B, B</td>
<td>8, B, B</td>
<td>8, B, B</td>
</tr>
<tr>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
</tr>
<tr>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
</tr>
<tr>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
</tr>
<tr>
<td>8, B, B</td>
<td>8, B, B</td>
<td>8, B, B</td>
<td>8, B, B</td>
</tr>
<tr>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
</tr>
<tr>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
</tr>
<tr>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
<td>+ B, B</td>
</tr>
</tbody>
</table>

\[\delta(q_8, x) = [q_8, y, L]\]
Time complexity (TC) of a TM $M$ is a function $t_{TM} : N \rightarrow N$ s.t. $t_{TM}(n)$ is the maximum number of transitions processed by a computation of $M$ when initiated with an input of length $n$.

Assumption: Computation terminates for every input string.

Example 1: $t_{TM}(0) = 1$, $t_{TM}(1) = 2$, $t_{TM}(2) = 6$, $t_{TM}(3) = 10$


t_{TM}(n) = \begin{cases} \frac{(n+2)(n+1)}{2} & n \geq 1 \\ 1 & n = 0 \end{cases}

Example 2: 2-tape $TM^2$:

$$
M' : \xrightarrow{1} \overbrace{[Bla(1), Bla(2)]}^{2} \xrightarrow{62} \overbrace{[Bla(3), Bla(4)]}^{94}
$$

$$
t_{TM}(n) = 3(n+1)+1
$$

Definition: A language $L$ is accepted in deterministic time $f(n)$ if $\exists$ a deterministic $TM$ of any kind with $t_{TM}(n) \leq f(n)$ + $n \in N$.

E.g. for palindromes, acceptance time is $(n^2 + 3n + 3)/2$.

Linear Speedup: A machine accepting a language $L$ can be "speed up" to a linear acceptance time by an arbitrary multiplicative factor.

Thus let $M$ be a 1-tape TM, $k \geq 1$, that accept $L$ with $t_{TM}(n) = f(n)$.

Then let $N$ be a $k$-tape machine $M$ that accepts $L$ with $t_{TM}(n) \leq [f(n)] + 2n$ for any constant $c < n_0$.

Rate of Growth: *exact computability of a language is futile.

We need to represent "order" of complexity.

*rate of growth: intuitively, the most significant contributor e.g. $n^2 \sim n^2 + 5$.
Definition: Function \( f = O(g) \), if \( f \) is a positive constant \( c \) and a natural \( n_0 \) so that
\[
\exists c, n_0 \in \mathbb{R}^+ : \forall n > n_0 \quad f(n) \leq c \cdot g(n)
\]
\( f = \text{big-O of } g \)

\( f = O(g) \) if
\[ f(n) \leq c \cdot g(n) \quad \forall n > n_1, \text{ and} \]
\[ g(n) \leq c_2 \cdot f(n) \quad \forall n > n_2 \]

\( \Rightarrow \) \[ \frac{f(n)}{c} \leq g(n) \leq c_2 \cdot f(n) \quad \text{and} \]
\[ g(n) / c_2 \leq f(n) \leq c \cdot g(n) \]

\( \exists \) \[ f(n) = n^2 + 2n + 5 \quad \text{and} \quad g(n) = n^2 \]

We see that \( n^2 \leq n^2 + 2n + 5 \), hence \( c = 1 \), \( n_0 = 0 \)

\[ 2n \leq 2n^2 \quad \text{and} \quad 5 \leq 5n^2 \quad \forall n > 1 \]

Then \( f(n) \leq n^2 + 2n + 5 \)
\[ \leq n^2 + 2n^2 + 5n^2 \]
\[ = 8n^2 \]
\[ = 8 \cdot g(n) \quad \text{for } n > 1 \]

\( \exists \) \[ f(n) = n^2, \quad g(n) = n^3, \quad f = O(g) \], \( g \neq O(f) \)

Clearly \( n^2 = O(n^3) \), i.e. \( n^2 \leq n^3 \).

For \( \text{and } g \neq O(f) \)

Assume \( n^3 = O(n^2) \), then \( \exists c, n > n_0 \)

\[ n^3 \leq c \cdot n^2 \quad \forall n > n_0 \]

Choose \( n_1 = \max \{ n_0 + 1, c + 1 \} \), then

\[ n_1^2 = n_1 \cdot n_1 > c \cdot n_1^2 \quad \text{and} \quad n_1 > n_0 \]

i.e. a contradiction.

Polynomial 1/ integral coefficients (of degree)

\[ f(n) = c_0 \cdot n^0 + c_1 \cdot n^1 + \ldots + c_m \cdot n^m \]

Rate of growth of \( f(n) \) is defined to be \( \max \{ m, \frac{n^m}{k} \} \).

May be a polynomial of degree \( R \), i.e.,

\( f = O(n^R) \)

\( \exists \) \[ m_1 = O(0) \quad \text{and} \quad f = O(n^R) \quad \text{and} \quad k > n \]

\( \exists \) \[ f \neq O(n^k) \quad \forall k < n \]
"Number theoretic" logarithmic \( f(n) = \lfloor \log_a(n) \rfloor \)

\[ \log_a(n) = \log_b(n) \log_b(l) \]

\( a, b \) cont.

\[ \text{independent of base.} \]

A big O hierarchy

\[ \begin{align*}
O(1) & \quad \text{constant} \\
O(\log_a(n)) & \quad \text{logarithmic} \\
O(n) & \quad \text{linear} \\
O(n \log_a(n)) & \quad \text{logarithmic} \\
O(n^2) & \quad \text{quadratic} \\
O(n^3) & \quad \text{cubic} \\
O(n^k) & \quad \text{polynomial} k > 0 \\
O(b^n) & \quad \text{exponential} b > 1 \\
O(n!) & \quad \text{factorial}
\end{align*} \]

Non-deterministic Complexity

Design: same as deterministic (using any choice of transition)

\[ \begin{align*}
& \text{2-type non-d deterministic accepting TM} \\
& \text{Ex: } \begin{cases}
[\text{a}\text{R,BL}\text{R}] & [\text{b}\text{R,BL}\text{R}] \\
[\text{a}\text{R,BL}\text{L}] & [\text{b}\text{R,BL}\text{R}] \\
[\text{B}\text{B}\text{R,BL}\text{R}] & [\text{B}\text{B}\text{R,BL}\text{L}] \\
[\text{B}\text{B}\text{L,BL}\text{L}] & [\text{B}\text{B}\text{R,BL}\text{R}] \\
[\text{B}\text{B}\text{L,BL}\text{L}] & [\text{B}\text{B}\text{L,BL}\text{L}] \\
[\text{B}\text{B}\text{R,BL}\text{R}] & [\text{B}\text{B}\text{L,BL}\text{L}] \\
[\text{B}\text{B}\text{L,BL}\text{L}] & [\text{B}\text{B}\text{L,BL}\text{L}] \\
[\text{B}\text{B}\text{R,BL}\text{R}] & [\text{B}\text{B}\text{L,BL}\text{L}] \\
\text{t}_{\text{tm}}(n) = n+1
\end{cases}
\end{align*} \]

Space Complexity: let \( M \) be a \( k \)-tape \( TM \). Space complexity of \( M \) is a \( \text{by } n \) \( \text{SCM} : n \rightarrow n \) at. \( \text{scm}(n) \) is the maximum number of squares read on tape 1...k by a computation of \( M \) when initiated with an input string of length \( n \).

Theorem: \( \text{\# of work tape} \)

\[ \begin{align*}
& \text{tape 1} \\
& \text{tape 2} \\
& \text{tape 3}
\end{align*} \]

Theorem: \( \text{\# of work tape} \)
Theorem: Let $M$ be a $k$-tape TM with $t_{TM}(n) = O(n^k)$. Then $S_{TM}(n) \leq (k-1) \cdot f(n)$.

If $S_{TM}(n) = O(n^k)$, then $t_{TM}(n) \leq C \cdot n^k$, where $C$ depends upon $M$.

Let $M$ be a TM with $t_{TM}(n) \leq C \cdot n^k$, where $C$ depends upon $M$. Then $S_{TM}(n) \leq C \cdot n^k$.

Tractable problems: $\exists$ a TM with polynomially bounded computational complexity.

Definition: A language $L$ is decidable in polynomial time if $L$ is a TM that accepts $L$ with $t_{TM} = O(n^k)$, where $C$ is independent of $n$.

Theorem: The class $NP$: non-deterministic polynomial time.

Definition: $NP$ is the family of all these $L$ as $P$.

Definition: $NP$ is the family of all these $L$ as $P$.

Definition: $NP$ is the family of all these $L$ as $P$.

Definition: $NP$ is the family of all these $L$ as $P$.

Definition: $NP$ is the family of all these $L$ as $P$.

Definition: $NP$ is the family of all these $L$ as $P$.

Definition: $NP$ is the family of all these $L$ as $P$.

Definition: $NP$ is the family of all these $L$ as $P$.

Definition: $NP$ is the family of all these $L$ as $P$.

Theorem: $P \subseteq NP$ (i.e., every $NP$ language is also in $P$).

Theorem: It is believed that $P \neq NP$.

Theorem: Let $Q$ and $L$ be languages over $\Sigma_1$ and $\Sigma_2$, respectively. $Q$ is reducible to $L$ in polynomial time if $Q$ is a poly-time computable function $f$ that maps elements of $\Sigma_1^*$ to elements of $\Sigma_2^*$ such that $f(q) \in L$.

Theorem: Let $Q$ and $L$ be languages over $\Sigma_1$ and $\Sigma_2$, respectively. If $Q \in NP$, then $Q \subseteq L$.

Definition: A language $L$ is called $NP$-hard if for every $Q \in NP$, $Q$ is reducible to $L$ in polynomial time. An $NP$-hard language that is also in $NP$ is called $NP$-complete.

Theorem: If there is an $NP$-complete language that is also in $P$,

Then $P = NP$. 