**Computational Complexity Theory, Fall 2010** August 25

# Lecture 1: Time Complexity

Lecturer: Peter Bro Miltersen Scribe: Søren Valentin Haagerup

## 1 Introduction to the course

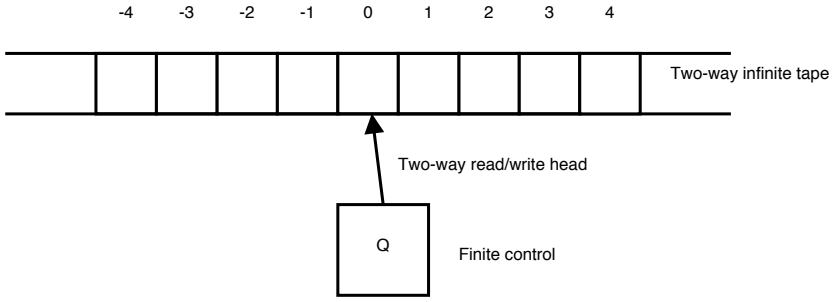
The field of Computational Complexity strives to solve big problems like P $\stackrel{?}{=}$ NP, PSPACE $\stackrel{?}{=}$ NP, NP $\stackrel{?}{=}$? coNP. Part of the course will cover which approaches and tools has been developed so far in the search of proofs to these statements.

It is also interesting to the field what consequences it has to assume statements like P $\neq$ NP, PSPACE $\neq$ NP or NP $\neq$ coNP as axioms.

## 2 First computational model: The 1-tape sequential Turing Machine

The 1-tape sequential Turing Machine (TM) consists of:

- Two-way infinite tape. The cells of the tape are numbered $\ldots -2, -1, 0, 1, 2 \ldots$ as a convenience when proving properties of a TM. These indexes are not known to the TM.
- Two-way read/write head
- Finite control (defined by the set of states $Q$ and the transition function $\delta$)




It is defined by:

- $Q$: finite set of states
- $\Sigma$ : Input alphabet
- $\Gamma$: Tape alphabet ( $\Sigma \cup \{\triangle\} \subset \Gamma$, $\triangle$ : blank symbol). A common trick when programming a TM is to extend the tape alphabet with marked versions of the input-alphabet.
- $\delta$: Transition function: $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, L\}$
- $s \in Q$: Start state

- "yes" $\in Q$: Accepted state
- "no" $\in Q$: Reject state

Initially:

- Input $x_1, \ldots x_n$ placed on cells $1, \ldots, n$.
- All other cells contain blank symbol $\triangle$. Head starts in position 0.

### 2.1 Time complexity

**Definition 1.** $time_M(x) :=$ # steps $M$ takes to halt given input $x$

**Definition 2.** $M$ is $T(n)$ time-bounded if $time_M(x) \leq T(|x|)$ for all $x \in \Sigma^*$

**Definition 3.** The language accepted by $M$: $L(M) = \{x \in \Sigma^* | M \text{ accepts on input } x\}$

*Example* 1. The language of palindromes

$$PAL = \{x_1, x_2, \ldots, x_n \in \Sigma^* | x_1 x_2 \ldots x_n = x_n x_{n-1} \ldots x_1\}$$

can be accepted in $O(n^2)$ steps by a 1-tape Turing machine (i.e. it is $n^2$ time bounded). Informally it can be done as follows:

- Read symbol of the string, and let the finite control remember it (by state), write the blank symbol, and move to the right until a blank symbol is reached (if a blank symbol is immediately reached, accept).
- Move 1 step left, read the symbol, and compare it with the remembered symbol (and reject when they are not equal).
- Write a blank symbol, and move left until reaching a blank symbol (if a blank symbol is immediately reached, accept). Move 1 step right.
- Repeat.

We will prove that PAL can **not** be accepted by any 1-tape Turing machine taking $o(n^2)$ steps ($f = o(g) \Leftrightarrow \frac{f(n)}{g(n)} \to 0$ as $n \to \infty$)

### 2.2 Crossing Sequences

This section essentially covers the content in Section 10.4 (p. 144-146) in Hopcroft & Ullman, but with a slightly different example.

**Definition 4.** The crossing sequence $C_i(x) = (q_{i,1}, q_{i,2}, .., q_{i,k})$ is the sequence of states $M$ is in **immediately after** crossing the line between cells $i$ and $i+1$. Note that this sequence alternates right/left crossing.

**Lemma 5.** *Let $x_1, x_2, y_1, y_2$ be strings. Assume: $M$ accepts $x = x_1 x_2$ and $y = y_1 y_2$ and $C_i(x) = C_j(y)$, where $i = |x_1|$ $j = |y_1|$. Then $M$ accepts $x_1 y_2$ as well.*

*Proof.* Since $|C_i(x)| = |C_j(y)|$, $M$ either accepts in left parts for both $x$ and $y$ or in right parts. Now observe

$$C_i(x) = C_i(x_1x_2) = C_i(x_1y_2) = C_i(y_1y_2) = C_j(y)$$

since computation on the right side only depends on the state when crossing to the right, and vice versa for left. Hence $M$ also accepts $x_1y_2$. □

**Definition 6.** Given $n$, let $k = \lfloor \frac{n}{4} \rfloor$ Let

$$PAL_n := \{x\#^{n-2k}\text{rev}(x) \mid x \in \{0,1\}^k\}$$

Note: $PAL_n \subset PAL$.

**Definition 7.** Let $|x| = n$.

$$C(x) := \{C_i(x) | k \leq i \leq n-k\}$$

**Lemma 8.** *Let $x, y \in PAL_n$ and $x \neq y$. Then $C(x) \cap C(y) = \emptyset$.*

*Proof.* Assume

- $x = \hat{x}\#^{n-2k}\text{rev}(\hat{x})$
- $y = \hat{y}\#^{n-2k}\text{rev}(\hat{y})$

Note $x \neq y \Leftrightarrow \hat{x} \neq \hat{y}$. Also $|\hat{x}| = |\hat{y}|$.

Assume $C(x) \cap C(y) \neq \emptyset$.

Pick $C_i(x) \in C(x), C_j(y) \in C(y)$ s.t. $C_i(x) = C_j(y)$.

By previous lemma $z := \hat{x}\#^{n-2k+i-j}\text{rev}(\hat{y})$ is accepted. But if $z \in PAL$ we would have that $\hat{x} = \hat{y}$.

Remark: Perhaps $|z| < n$ or $|z| > n$ □

**Theorem 9.** *Suppose $M$ is $T(n)$ time-bounded and accepts $PAL$ (not just $PAL_n$). Then $T(n) = \Omega(n^2)$*

*Proof.* Let $x \in PAL_n$. Note

$$T(n) \geq time_M(x) \geq \sum_{i=k}^{n-k} |C_i(x)| \tag{1}$$

since each time a boundary between $k$ and $n-k$ is crossed a move is made. Let

$$m_x = \min_{k \leq i \leq n-k} |C_i(x)|$$

be the length of the smallest crossing sequence on input $x$.

Let $m = \max_{x \in PAL_n}\{m_x\}$ be the largest minimum obtained for an $x \in PAL_n$.

- "# crossing sequences of length $\leq m$" $= \sum_{j=0}^{m} |Q|^k = \frac{|Q|^{m+1}-1}{|Q|-1}$. Note that this is just an upper bound counting all possible sequences of states.
- "# elements of $PAL_n$" $= 2^k$. ($\hat{x}$ in $\hat{x}\#^{n-2k}\text{rev}(\hat{x}) \in PAL_n$ is of length $k$ and in the 2-character alphabet $\{0,1\}$)

We know that there are (several) unique crossing sequences for each element in $PAL_n$ (by the previous lemma). In other words:

$$\text{“\# crossing sequences of length } \leq m\text{”} \;\geq\; \text{“\# elements of } PAL_n\text{”}$$

We use that $k = \lfloor \frac{n}{4} \rfloor$.

$$\frac{|Q|^{m+1} - 1}{|Q| - 1} \geq 2^{\lfloor \frac{n}{4} \rfloor}$$

$$\Rightarrow \; \log(|Q|^{m+1} - 1) - \log(|Q| - 1) \geq \lfloor \frac{n}{4} \rfloor \tag{2}$$

For all turing machines $|Q| \geq 3$ (start state, accept state, reject state), and $\log(x) \geq 0$ for $x \geq 1$. Using this, and the fact that log increases monotonously:

$$\log(|Q|^{m+1} - 1) - \log(|Q| - 1) \leq \log(|Q|^{m+1} - 1) \leq \log(|Q|^{m+1}) = (m+1)\log|Q| \tag{3}$$

Combining (2) and (3) to

$$(m+1)\log|Q| \geq \lfloor \frac{n}{4} \rfloor$$

we get $m = \Omega(n)$.

Using this and (1)

$$\sum_{i=k}^{n-k} |C_i(x)| \geq (n-2k)m = \Omega(n^2) \Rightarrow T(n) = \Omega(n^2)$$

□

**Open problem 10.** Can a 1-tape turing machine decide $SAT$ in time $O(n^2)$?

# 3 Standard computational model: The $k$-tape sequential Turing machine

- $k$ two-way infinite tapes and $k$ heads
- tape 1 acts as input tape
- $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{R, L\}^k$

*Example* 2. PAL can be accepted in $O(n)$ steps on a 2-tape Turing machine.

Informally it can be done as follows: Copy all the content from tape 1 to tape 2. Move the head of tape 1 to the right, until a blank symbol is reached, and move 1 step left. Compare the symbols (if they are not equal, reject). Let tape 1 move left and let tape 2 move right. Repeat until both tapes reach a blank symbol and accept.

### 3.1 Why not use the Random Access Machines (RAM) as a model?

- it is messy to define (instruction sets, indirect addressing etc.)
- it is unrealistic for big amounts of data (no random access in constant time when working with PBs of data)
- the $k$-tape model is good enough when trying to distinct exponential and polynomial time

## 4 Universal Turing Machines

**Proposition 11.** *For any $k$, there exist a $k+1$ tape Turing machine $U_k$, that given as input a description of a $k$-tape Turing machine $M$ as well as input $x$ simulates $M$ on input $x$ in $O(time_M(x))$ steps, where the constant in the $O(\cdot)$ notation depends on $M$.*

*Proof.* Simple simulation. Keep description and state on tape $k+1$. Other tapes work just like on $M$ (except an encoding of the tape alphabet is used).

□

*Observation* 12. It is enough to use $k$ tapes, by shifting the contents of tape $k+1$ along on one of the tapes.

**Exercise 13.** Prove that the constant in the $O(\cdot)$ notation depends on $M$.

### 4.1 Tape reduction ($k$-tape → 1-tape)

Assume $T(n) \geq n$. The following proof is given with more details in Thm. 10.4 (p. 139) in Hopcroft & Ullman.

**Proposition 14.** *Suppose $L$ is accepted by a $k$-tape $T(n)$ time-bounded TM $M$. Then $L$ is also accepted by a $O(T(n)^2)$ time-bounded 1-tape TM $M'$.*

*Proof.* Enlarging the tape alphabet we maintain $k$ "tracks" on the tape of $M'$. Positions of heads are maintained as "markers" on the tracks. A simulation of a single step of $M$ is done by sweeping the tape, recording symbols under the head markers. Then the move is determined and we can update tracks and markers by antoher sweep. Thus every step can be done in time ($O(T(n))$. □

*Observation* 15. The example of $PAL$ shows that this reduction is optimal.

### 4.2 Tape reduction ($k$-tape → 2-tape)

**Proposition 16.** *Suppose $L$ is accepted by a $k$-tape, $T(n)$ time-bounded TM $M$. Then $L$ is also accepted by a 2-tape $O(T(n)\log T(n))$ time bounded TM $M'$.*

*Proof.* To be shown friday. □

**Open problem 17.** Can $L$ also be accepted by a 2-tape $O(T(n))$ time bounded TM ?