# Computational Complexity

Peter Mayr

Computability Theory, November 6, 2023

## Question

How to measure the hardness of solvable computational problems?

Based on

- Arora, Sanjeev; Barak, Boaz. Computational complexity: a modern approach. Cambridge University Press, 2007
- Hopcroft, John; Motwani, Rajeev; Ullman, Jeffrey. Introduction to automata theory, languages, and computation. Pearson, 3rd edition, 2006.
- Papadimitriou, Christos. Computational complexity. Addison-Wesley, 1994.
- Sipser, Michael. Introduction to the theory of computation. Thomson Course Technology, Boston, 2nd edition, 2006.

# Time complexity of a DTM

## Definition

Let M be a DTM (1 tape, one-sided infinite) on an input alphabet $\Sigma$ that halts on every input, let $t\colon \mathbb{N} \to \mathbb{N}$.
M **runs in time** $t(n)$ (has (worst case) **running time**, **time complexity** $t(n)$) if $t(n)$ is the maximum number of steps that $M$ uses on any input $x \in \Sigma^*$ with $|x| = n$.

## Note

- $t(n)$ is usually hard to determine exactly.
- It is more relevant how the complexity increases in the size of the input, i.e. the asymptotics of $t$.

# Big O notation

## Definition

For $f, g : \mathbb{N} \to \mathbb{R}_0^+$, we say $f = O(g)$ (read $f$ is big O of $g$) if

$$\exists c, N \in \mathbb{N}\ \forall n \geq N : \ f(n) \leq cg(n).$$

## Example

- $5n^3 - n + 1 = O(n^3)$
- $n = O(n^3)$
- Constant functions are $O(1)$.
- $\log_2(n), \log_{10}(n) = O(\log(n))$ (base does not matter)

## Lemma

For $f_1 = O(g_1), f_2 = O(g_2)$ and $c > 0$

- $f_1 \cdot f_2 = O(g_1 g_2)$,
- $f_1 + f_2 = O(\max(g_1, g_2))$,
- $cf_1 = O(cg_1) = O(g_1)$.

## Example

$L = \{0^k 1^k \ : \ k \in \mathbb{N}\}$ is computable by a DTM.

On input $w$ of length $n$:

1. Scan across $w$. Reject if 1 is followed by 0.
2. While the tape is not empty, replace the first 0 and last 1 by $\sqcup$.
3. If some 0 remains but no 1 (or conversely), reject; else accept.

Time used

in 1. $O(n)$

in 2. $\frac{n}{2} O(n) = O(n^2)$

in 3. $O(n)$

Overall running time is $O(n^2)$.

Space complexity is $O(n)$

(For now space is just measured in the number of cells used on the tape; proper definition later).

# Complexity depends on the model

## Note

All Turing complete computational models have the same expressive power but

- may encode input in different sizes (e.g. numbers in unary, binary ...)
- may have different time/space complexity for the same computation.

## Theorem

Any $k$-tape DTM running in time $t(n) \geq n$ (on inputs of length $n$) can be simulated by a single tape DTM running in time $O(t(n)^2)$.

## Proof sketch.

Recall: a configuration of a $k$-tape DTM $M$ consists of

- state $q$
- $k$ tape contents $\sqcup l_1 a_1 r_1 \sqcup, \ldots, \sqcup l_k a_k r_k \sqcup$
- with tape heads reading $a_1, \ldots, a_k$ (here $l_i, r_i$ are finite strings over the tape alphabet $\Gamma$).

Extending $\Gamma$ with a delimiter $*$ and $\Gamma' := \{a' \ : \ a \in \Gamma\}$, a single tape DTM $M'$ may represent the $k$ tape contents as

- $* l_1 a'_1 r_1 * l_2 a'_2 r_2 * \cdots * l_k a'_k r_k * \ldots$

(Symbols $a'_i$ record the tape head positions of $M$.)

To simulate one step of $M$, $M'$ scans its tape left to right and back

- reading $a'_1, \ldots, a'_k$,
- applying the transition function of $M$,
- replacing $a'_i$ (and their neighbors) accordingly.

## Running time of the single tape $M'$ on input $x$

- Initially $M'$ formats its tape as $*x*\sqcup*\cdots*$ in $O(|x|)$.
- Since $M$ halts after $\leq t(|x|)$ steps on $x$, it never uses more than $t(|x|)$ space on any of its tapes.
- So $M'$ needs $O(kt(|x|))$ space.
- Simulating a single step of $M$ takes $O(kt(|x|))$ steps for $M'$.
- Hence $M'$ has running time $O(|x|) + O(kt(|x|)^2) = O(t(|x|)^2)$. □

## Question

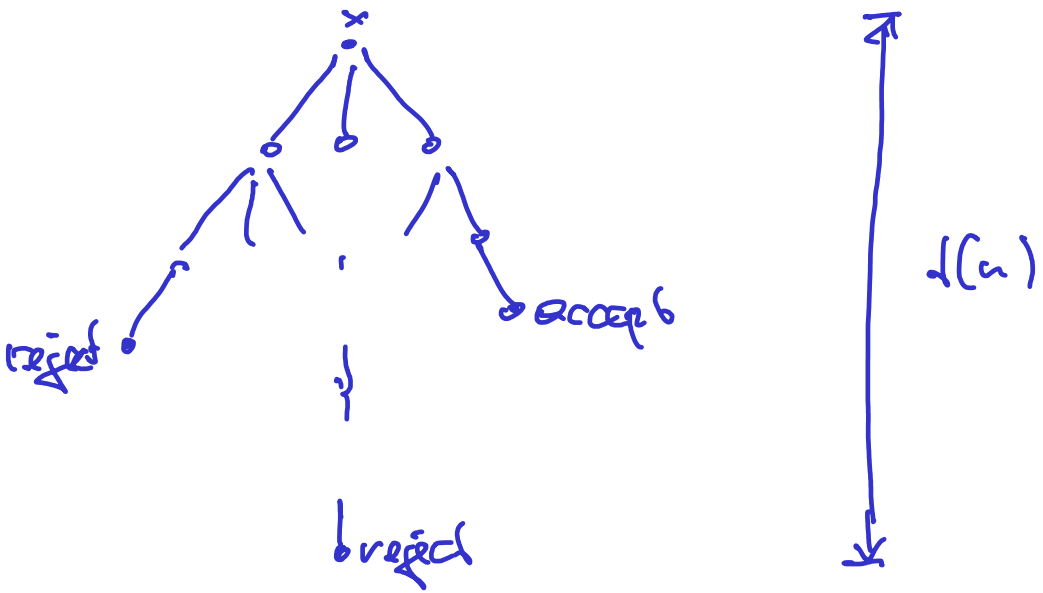
What is the running time of a DTM with a single one-sided infinite tape simulating a DTM with a single bi-infinite tape?

# Time complexity of a non-deterministic TM

## Definition

Let $N$ be a non-deterministic TM (1 tape, one-sided infinite) on an input alphabet $\Sigma$ such that all computation branches halt on all inputs, let $t : \mathbb{N} \to \mathbb{N}$.

$N$ **runs in time** $t(n)$ if $t(n)$ is the maximum number of steps that $N$ uses on any computation branch on any input $x \in \Sigma^*$ with $|x| = n$.



## Note

Why the maximum length over all branches and not the length of the shortest accepting branch if that exists?

- The given measure is easier to analyze and
- will yield the same class NP.

# Changing from non-deterministic to deterministic

## Theorem

Any non-deterministic TM N running in time $t(n) \geq \log(n)$ (on inputs of length $n$) can be simulated by a DTM M running in time $2^{O(t(n))}$.

## Proof sketch.

Recall: Construct a DTM M that tries all branches of N's computation tree breadth first.

Let $b := \max\{|\Delta(q, a)| \ : \ q \in Q, a \in \Gamma\}$, label all transitions in $\Delta(q, a)$ by $1, \ldots, b$.

For $s = 1, 2, \ldots$

- Enumerate paths $(a_1, \ldots, a_s)$ with $a_i \leq b$.
- Simulate the computation branch of $N$ labelled $(a_1, \ldots, a_s)$.
- If N accepts, then so does M; else M uses the next path.
- If at some stage $s$ all computations of N halt without accepting, then M rejects.

## Running time of M

- There are $\leq \sum_{s=1}^{t(n)} b^s = O(b^{t(n)})$ paths.
- $O(t(n))$ time to generate and simulate a single path.
- Hence M has running time $O(t(n)b^{t(n)}) = 2^{O(t(n))}$.

□

Q: Can one do better?