Introduction

Complexity Theory

Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic

Ondřej Lengál, Adam Rogalewicz

Computational Complexity

Computational complexity theory

- classifies the inherent complexity of problems into classes based on the amount of resources (time, space, . . .) they need,
  - *The problem of checking whether a Boolean formula is satisfiable is solvable in nondeterministic polynomial time (*SAT $\in$ **NP***).*
- relates these classes to each other.
  - *All problems solvable in nondeterministic polynomial time are also solvable in deterministic polynomial space (***NP** $\subseteq$ **PSPACE***).*

Relation to Decidability

Decidability

- Is it possible to solve a given problem at all?
  - *Given a Turing machine M and an input x, does M terminate on x?*

Complexity

- Is it possible to solve a given problem with limited resources, i.e. is there an algorithm that solves the problem using only the given resources? (upper bound)
  - *For a directed graph G and nodes u and v, can we decide whether there exists no path from u to v using only nondeterministic logarithmic space?*
- What resources are necessary to solve a problem, i.e. there is no algorithm that can use less resources to solve the problem? (lower bound)
  - *Is it possible to evaluate whether a formula in Presburger arithmetic is satisfiable with less than deterministic exponential time?*

Types of Problems

Types of problems:

- decision problems,
  - *Given a directed graph G and a pair of nodes u, v, is there a path from u to v in G?*
- search problems,
  - *Given a directed graph G and a pair of nodes u, v, find a path from u to v if it exists.*
- optimisation problems,
  - *Given a directed graph G and a pair of nodes u, v, find a path from u to v of minimum length if it exists.*
- counting problems.
  - *Given a directed graph G and a pair of nodes u, v, how many paths from u to v are in G?*

Kolmogorov Complexity

Kolmogorov complexity (descriptive complexity):

- is concerned about the length of an algorithm that solves the given problem,
    - *Can the problem be solved by a Turing machine with 4 states?*
- it often holds that fast algorithms are long, and slow algorithms are short their size.

Models of Computation

A model of computation:

- defines the operations that can be used in a computation and their costs,
- examples:
  - a Turing machine,
  - a random access machine (RAM),
  - a parallel RAM (PRAM),
  - a probabilistic Turing machine,
  - circuits,
  - a quantum computer, . . .

Cobham's Thesis

Cobham's Thesis

A problem can be feasibly computed on some computational device only if it can be computed in the time polynomial to the length of the input $\Rightarrow$ the class **P**.

- Existence of an algorithm does not imply an efficient solution to the problem.
- Cobham's thesis delimits the class of efficiently solvable problems.
- Indeed, for problems not in **P**, practical algorithms often use heuristics or find only an approximate solution.
- There are many objections to Cobham's thesis though, as it asserts that all problems in **P** are easy and all problems not in **P** are too hard, with neglecting the coefficients and other terms.

Turing Machine

Definition

A Turing Machine (TM) is a sextuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$ where

- Q is a finite non-empty set of states,
- Σ is the (finite non-empty) input alphabet,
- Γ is the (finite non-empty) tape alphabet, $\Sigma \subset \Gamma$, $\Delta \in \Gamma \setminus \Sigma$, (blank symbol: Δ)
- δ: $(Q \setminus \{q_F\}) \times \Gamma \to Q \times (\Gamma \uplus \{L, R\})$ is a partial transition function,
- $q_0 \in Q$ is the initial state,
- $q_F \in Q$ is the final state.

Turing Machine

Definition

A configuration C of M is given by the current state of M, state of the tape, and the position of tape head:

$$C \in Q \times (\gamma\Delta^\omega \mid \gamma \in \Gamma^*) \times \mathbb{N}$$

Example: $C = (q_1, aabbcc\Delta^\omega, 3)$.

Definition

The transition relation $\vdash_M$ of M is the smallest binary relation on configurations of M defined such that

$$\begin{array}{lll} (q_1, \gamma, n) & \vdash_M (q_2, \gamma, n+1) & \text{if } \delta(q_1, \gamma_n) = (q_2, R), \\ (q_1, \gamma, n) & \vdash_M (q_2, \gamma, n-1) & \text{if } \delta(q_1, \gamma_n) = (q_2, L) \text{ and } n > 0, \\ (q_1, \alpha x \beta, n) & \vdash_M (q_2, \alpha y \beta, n) & \text{if } \delta(q_1, x) = (q_2, y) \text{ where} \\ & & x, y \in \Gamma, \alpha \in \Gamma^n, \beta \in \Gamma^*\{\Delta^\omega\}. \end{array}$$

Example: $(q_1, aabbcc\Delta^\omega, 3) \vdash_M (q_2, aabdcc\Delta^\omega, 3)$ if $\delta(q_1, b) = (q_2, d)$.

Turing Machine

Definition

The language $L(M)$ of M is the set of words over the input alphabet for which there is a computation of M from the initial to the final state:

$$L(M) = \{w \in \Sigma^* \mid (q_0, \Delta w \Delta^\omega, 0) \vdash_M^* (q_F, \gamma, n)\}$$

where $\gamma \in \Gamma^*\{\Delta^\omega\}$ and $\vdash_M^*$ is the reflexive transitive closure of $\vdash_M$.

Definition

The function $f_M : \Sigma^* \to \Sigma^*$ is computed by M iff

$$(f_M(w) = w') \iff (q_0, \Delta w \Delta^\omega, 0) \vdash_M^* (q_F, \Delta w' \Delta^\omega, n)$$

for all $w, w' \in \Sigma^*$ and some $n \in \mathbb{N}$.

Time Complexity

Definition

The time complexity of the computation of the Turing Machine M on the input w is the function $t_M : \Sigma^* \to \mathbb{N} \cup \{\infty\}$ defined as

$t_M(w) = n \in \mathbb{N}$ iff the computation of M on w halts in n steps,
$t_M(w) = \infty$ iff the computation of M on w does not halt.

Definition

The time complexity of the Turing Machine M is the function $T_M : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ defined as

$$T_M(n) = \max\{t_M(w) \mid w \in \Sigma^n\}.$$

Definition

Given a function $f : \mathbb{N} \to \mathbb{N}$ we define the computational resource

$$\mathbf{DTIME}(f(n)) = \{L \subseteq \Sigma^* \mid \text{there is a TM } M \text{ s.t. } T_M(n) \le f(n)\}.$$

Space Complexity

Definition

i.e. α does not end with $\Delta \cdots \Delta$

Let $C = (q, \alpha\Delta^\omega, n), \alpha \in \Gamma^* \setminus (\Gamma^*\{\Delta\}), n \in \mathbb{N}$, be a configuration of the Turing Machine M. The space complexity $s(C)$ of the configuration C is defined as $s(C) = \max\{|\alpha|, n\}$.

Definition

The space complexity of the computation of the Turing Machine M on the input w is the function $s_M : \Sigma^* \to \mathbb{N} \cup \{\infty\}$ defined as

$$s_M(w) = \max\{s_M(C) \mid (q_0, \Delta w \Delta^\omega, 0) \vdash_M^* C\}.$$

where the maximum of an infinite set is ∞.

Space Complexity

Definition

The space complexity of the Turing Machine M is the function $S_M : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ defined as

$$S_M(n) = \max\{s_M(w) \mid w \in \Sigma^n\}.$$

Definition

Given a function $f : \mathbb{N} \to \mathbb{N}$ we define the computational resource

$$\mathbf{DSPACE}(f(n)) = \{L \subseteq \Sigma^* \mid \text{there is a TM } M \text{ s.t. } S_M(n) \le f(n)\}.$$

Non-deterministic Turing Machine

Definition

A Non-deterministic Turing Machine (NTM) is a sextuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$ where Q, Σ, Γ, q_0, and q_F are defined as for Turing Machines and

$$\delta : (Q \setminus \{q_F\}) \times \Gamma \to 2^{Q \times (\Gamma \uplus \{L,R\})}.$$

The configuration C, transition relation $\vdash_M$ and language $L(M)$ of M are defined as for Turing Machines. Note that for $w \in L(M)$ there may be multiple computations of M on w, some of them may be rejecting or not halting.

Time Complexity of NTMs

Definition

The time complexity of the Non-deterministic Turing Machine M is the function $T_M : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ defined as

$$T_M(n) = \max\{t_M(w) \mid w \in \Sigma^n\}$$

where t_M is the maximum number of steps of a computation of M on w (or ∞ if the computation of M loops on w).

Definition

Given a function $f : \mathbb{N} \to \mathbb{N}$ we define the computational resource

$$\mathbf{NTIME}(f(n)) = \{L \subseteq \Sigma^* \mid \text{there is a NTM } M \text{ s.t. } T_M(n) \leq f(n)\}\,.$$

Note: If there is a word $w \in \Sigma^*$ such that there is a computation of M on w that loops, then $T_M(|w|) = \infty$. However, if $L(M) \in \mathbf{NTIME}(f(n))$ then there exists a NTM M' s.t. each computation of M on w ends in at most $f(|w|)$ steps.

Time Complexity of NTMs

Lemma

For all $f : \mathbb{N} \to \mathbb{N}$*:*

$$\mathbf{DTIME}(f(n)) \subseteq \mathbf{NTIME}(f(n)).$$

Proof. TM is a special case of a NTM. □

Space Complexity of NTMs

Definition

The space complexity of the Non-deterministic Turing Machine M is the function $S_M : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ defined as

$$S_M(n) = \max\{s_M(w) \mid w \in \Sigma^n\}$$

where $s_M(w) = \max\{s(C) \mid (q_0, \Delta w \Delta^\omega, 0) \vdash_M^* C\}$.

Definition

Given a function $f : \mathbb{N} \to \mathbb{N}$ we define the computational resource

$$\textbf{NSPACE}(f(n)) = \{L \subseteq \Sigma^* \mid \text{there is a NTM } M \text{ s.t. } S_M(n) \leq f(n)\}.$$

Linear Speedup

Lemma

Let $L \in DTIME(f(n))$*. Then for each* $\epsilon > 0 : L \in DTIME(\epsilon * f(n) + n)$.

Proof. By construction of TM over working alphabet $\Gamma \cup \Gamma^k$ for a suitable constant k. Such a TM can perform k steps of the original TM within finite number of steps. □

Linear Space Compression

Lemma

Let $L \in DSPACE(g(n))$. Then for each $\epsilon > 0 : L \in DSPACE(\epsilon * g(n))$.

Proof. By construction of TM over working alphabet $\Gamma \cup \Gamma^k$ for a suitable constant k. Such a TM can perform k steps of the original TM within finite number of steps. □

Linear Speedup and Space Compression

- Linear speedup and space compression work also for nondeterministic complexity classes.
- For each constant $c > 0$, the following equalities hold:
  - $DTIME(f(n)) = DTIME(c * f(n) + n)$
  - $NTIME(f(n)) = NTIME(c * f(n) + n)$
  - $DSPACE(f(n)) = DSPACE(c * f(n))$
  - $NSPACE(f(n)) = NSPACE(c * f(n))$

Big $\mathcal{O}$ Notion

To avoid problems related to linear speedup and space compression, we define the functions O (big-O), Σ and Θ as follows:

Asymptotic upper bound

$O(f(n)) = \{g(n) \in \mathcal{F} \mid \exists c \in \mathbb{R}^+ \; \exists n_0 \in \mathbb{N} \; \forall n \in \mathbb{N} : n \geq n_0 \Rightarrow 0 \leq g(n) \leq c.f(n)\}.$

Asymptotic lower bound

$\Omega(f(n)) = \{g(n) \in \mathcal{F} \mid \exists c \in \mathbb{R}^+ \; \exists n_0 \in \mathbb{N} \; \forall n \in \mathbb{N} : n \geq n_0 \Rightarrow 0 \leq c.f(n) \leq g(n)\}.$

Asymptotic both-side bound

$\Theta(f(n)) = \{g(n) \in \mathcal{F} \mid \exists c_1, c_2 \in \mathbb{R}^+ \; \exists n_0 \in \mathbb{N} \; \forall n \in \mathbb{N} : n \geq n_0 \Rightarrow 0 \leq c_1.f(n) \leq g(n) \leq c_2.f(n)\}.$

Big $\mathcal{O}$ Notion – Conventions

The following conventions are often used in literature:

- $f(n) = O(g(n))$ iff $f(n) \in O(g(n))$
- $f(n) = \Omega(g(n))$ iff $f(n) \in \Omega(g(n))$
- $f(n) = \Theta(g(n))$ iff $f(n) \in \Theta(g(n))$

Equivalent definitions of DTIME, DSPACE, NTIME and NSPACE

The following definitions are equivalent due to the linear speedup and space compression.

The resource $DTIME(f(n))$ can be defined as:

$\mathbf{DTIME}(f(n)) = \{L \subseteq \Sigma^* \mid \text{there is a TM } M \text{ s.t. } T_M(n) \leq f(n)\}$
as well as
$\mathbf{DTIME}(f(n)) = \{L \subseteq \Sigma^* \mid \text{there is a TM } M \text{ s.t. } T_M(n) \in O(f(n))\}$

The resource $DSPACE(g(n))$ can be defined as:

$\mathbf{DSPACE}(g(n)) = \{L \subseteq \Sigma^* \mid \text{there is a TM } M \text{ s.t. } S_M(n) \leq g(n)\}$
as well as
$\mathbf{DSPACE}(g(n)) = \{L \subseteq \Sigma^* \mid \text{there is a TM } M \text{ s.t. } S_M(n) \in O(g(n))\}$

The equal principle can be used also in definitions of $NTIME(f(n))$ and $NSPACE(g(n))$.

Multi-tape Turing Machine

Basic idea

Instead of a single infinite tape, a Multi-tape Turing Machine M uses several of them (together with a tape head for each tape).

- In each step, M performs a write/move on all tapes at once.
- The time complexity is, as for single-tape Turing Machines, the number of steps.
- The space complexity is extended by taking the sum of space complexities of configurations of all the tapes.

Multi-tape Turing Machine (2)

Lemma

Let M be a multi-tape TM recognizing a language $L = L(M)$ with time complexity $f(n)$. Then there exists a (single-tape) TM M' such that $L = L(M) = L(M')$ and M' recognizes L with time complexity $f(n)^2$.

Lemma

Let M be a multi-tape TM recognizing a language $L = L(M)$ with space complexity $g(n)$. Then there exists a (single-tape) TM M' such that $L = L(M) = L(M')$ and M' recognizes L with space complexity $g(n)$.

Proof. By construction of TM over working alphabet $\Gamma \cup \Gamma^2$. □

Multi-tape Turing Machine (3)

Lemma

Let M be a multi-tape TM recognizing a language $L = L(M)$ with time complexity $f(n)$. Then there exists a 2-tape TM M' such that $L = L(M) = L(M')$ and M' recognizes L with time complexity $O(f(n) * log(f(n)))$.

Turing Machine with Input and Output Tape:

- a variant of a Multi-tape Turing Machine:
  - the input tape is read-only,
  - the output tape is write-only,
  - there are also read/write work tapes,
  - the time complexity is the number of steps,
  - the space complexity is the sum of space complexities of configurations of all the tapes except the input and output.

Constructible Functions

- For a language $L \in \mathbf{DTIME}(f(n))$ (or $\mathbf{NTIME}(f(n))$), we would like all computations of a TM M accepting L halt in the order of $f(n)$ steps (i.e. in $k \cdot f(n)$ steps for some $k \in \mathbb{N}$).
- This can be done by computing $f(|w|)$ (where w is the input) first and then simulating the computation of M, in each step checking that the simulated computation has not exceeded $f(|w|)$ steps.
- For this we need to be able to compute $f(|w|)$ in the available time!
- And similarly for **DSPACE** (**NSPACE**) and used memory cells.

$\Rightarrow$ constructible functions

Constructible Functions

Definition

Let f be a function $f : \mathbb{N} \to \mathbb{N}$. f is time constructible iff there is a Turing Machine M_f that for every input of length n outputs the binary representation of $f(n)$ in at most $n + k \cdot f(n)$ steps for some $k \in \mathbb{N}$.

Definition

Let f be a function $f : \mathbb{N} \to \mathbb{N}$. f is space constructible iff there is a Turing Machine M_f with input and output tape that for an input of length n outputs the binary representation of $f(n)$ while using at most $k \cdot f(n)$ cells on its work tapes.

Example

- $f(n) = c, f(n) = n, f(n) = \log(n)$ are time and space constructible.
- a function that is neither time nor space constructible:

$$f(n) = \begin{cases} n^2 & \text{if } 1^n \text{ is an encoding of a TM that halts on all inputs,} \\ n^3 & \text{otherwise.} \end{cases}$$