Other Complexity Classes

Other Complexity Classes

For an arbitrary function $f : \mathbb{N} \to \mathbb{R}_+$ we can define the following classes:

- DTIME$(f(n))$ — the class of all decision problems that can be solved by a **deterministic** algorithm with **time** complexity in $O(f(n))$
- NTIME$(f(n))$ — the class of all decision problems that can be solved by a **nondeterministic** algorithm with **time** complexity in $O(f(n))$
- DSPACE$(f(n))$ — the class of all decision problems that can be solved by a **deterministic** algorithm with **space** complexity in $O(f(n))$
- NSPACE$(f(n))$ — the class of all decision problems that can be solved by a **nondeterministic** algorithm with **space** complexity in $O(f(n))$

Remark: These classes depend on a used particular model of computation (one-tape Turing machine, multitape Turing machine, RAM, . . .)

Classes PTIME and NPTIME

Classes PTIME and NPTIME, introduced before, can be defined as follows:

$$\mathsf{PTIME} = \bigcup_{k \in \mathbb{N}} \mathsf{DTIME}(n^k)$$

$$\mathsf{NPTIME} = \bigcup_{k \in \mathbb{N}} \mathsf{NTIME}(n^k)$$

- PTIME — the class of all decision problems that can be solved by a deterministic algorithm with a polynomial time complexity
- NPTIME — the class of all decision problems that can solved by a nondeterministic algorithm with a polynomial time complexity

Remark: Classes PTIME and NPTIME are (more or less) independent of a particular model of computation — different models are able to simulate each other with at most polynomial increase in a running time.

Classes PSPACE and NPSPACE

Similarly, classes PSPACE and NPSPACE can be defined for a space complexity:

$$\mathrm{PSPACE} = \bigcup_{k\in\mathbb{N}} \mathrm{DSPACE}(n^k)$$

$$\mathrm{NPSPACE} = \bigcup_{k\in\mathbb{N}} \mathrm{NSPACE}(n^k)$$

- PSPACE — the class of all decision problems that can be solved by a deterministic algorithm with a polynomial space complexity
- NPSPACE — the class of all decision problems that can be solved by a nondeterministic algorithm with a polynomial space complexity

Classes PSPACE and NPSPACE

Simple observation:

- An arbitrary nondeterministic algorithm with a time complexity in $O(f(n))$ can be simulated by a **deterministic** algorithm with space complexity $O(f(n))$ — it can systematically try all possible computations of the given nondeterministic algorithm.

It follows from this observation that NPTIME $\subseteq$ PSPACE.

Classes PSPACE and NPSPACE

Theorem (Savitch, 1970)

It holds for each function $f \in \Omega(\log(n))$ that

$$\mathrm{NSPACE}(f(n)) \subseteq \mathrm{DSPACE}((f(n))^2).$$

Corollary: $\mathrm{PSPACE} = \mathrm{NPSPACE}$

Classes EXPTIME and NEXPTIME

Similarly as the classes PTIME and NPTIME, the classes EXPTIME and NEXPTIME can be introduced:

$$\mathrm{EXPTIME} = \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(2^{n^k})$$

$$\mathrm{NEXPTIME} = \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(2^{n^k})$$

- EXPTIME — the class of all decision problems that can be solved by a deterministic algorithm with an exponential time complexity
- NEXPTIME — the class of all decision problems that can be solved by a nondeterministic algorithm with an exponential time complexity

Relations between Classes

Simple observation:

- An arbitrary deterministic algorithm with a space complexity in $O(f(n))$ has a time complexity at most $O(c^{f(n)})$ where c is a constant — no configuration can repeat during a computation and the number of possible configurations is at most $O(c^k)$.

It follows from this observation that PSPACE $\subseteq$ EXPTIME.

PSPACE-Complete Problems

- A problem P is PSPACE-**hard** if for every problem P' in PSPACE thare exists a polynomial time reduction from P' to P.
- A problem P is PSPACE-**complete** if it is PSPACE-hard and also belongs to PSPACE.

PSPACE-Complete Problems

A typical example of a PSPACE-complete problem is the problem of **quantified boolean formular** — **QBF**:

QBF

Input: A quantified boolean formula of the form

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \cdots \exists x_{n-1} \forall x_n : \varphi,$$

where φ is a (standard) boolean formuala containing variables $x_1, x_2, \ldots, x_n$.

Question: Is the given formula true?

PSPACE-Complete Problems

EqNFA

Input: Nondeterministic finite automata $\mathcal{A}_1$ and $\mathcal{A}_2$.

Question: Is $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$?

Universality of NFA

Input: A nondeterministic finite automaton $\mathcal{A}$.

Question: Is $\mathcal{L}(\mathcal{A}) = \Sigma^*$?

PSPACE-Complete Problems

EqRE

Input: Regular expressions α_1 and α_2.

Question: Is $\mathcal{L}(\alpha_1) = \mathcal{L}(\alpha_2)$?

Universality of RE

Input: A regular expression α.

Question: Is $\mathcal{L}(\alpha) = \Sigma^*$?

PSPACE-Complete Problems

Consider the following game played by two players on a directed graph G:

- Players alternate in moving one pebble on the nodes of the graph G.
- During moves they mark nodes that were already visited with the pebble.
- A play starts in a specified node v_0.
- Let us say that the pebble is currently on a node v. The player whose turn it is chooses a node v' such that there is an edge from v to v' and such that v' has not been visited yet.
- A player that cannot move the pebble loses and his/her opponent wins.

Generalized Geografy

Input: A directed graph G with a specified initial node v_0.

Question: Does the player that plays first a winning strategy in the game on the graph G if the play starts in the node v_0?

Class EXPSPACE

The class EXPSPACE consists of those decision problems that can be solved in exponential space:

$$\mathrm{EXPSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{DSPACE}(2^{n^k})$$

The notions EXPSPACE-hard problem and EXPSPACE-complete problem are defined in a similar manner as the notions PSPACE-hard and PSPACE-complete problem.

EXPSPACE-Complete Problem

Regular expressions with squaring are defined in a similar manner as standard regular expressions but in addition to operators $+$, $\cdot$, and * they can contain a unary operator ^{2} with the following meaning:

- α^2 is a shorthand for $\alpha \cdot \alpha$.

The following two problems are EXPSPACE-complete:

Input:	Regular expressions with squaring α_1 and α_2.
Question:	Is $\mathcal{L}(\alpha_1) = \mathcal{L}(\alpha_2)$?

Input:	A regular expression with squaring α.
Question:	Is $\mathcal{L}(\alpha) = \Sigma^*$?

Complexity Classes

For definition of LOGSPACE class we specify more exacly what we consider as a space complexity of an algorithm.

For example, let us consider a Turing machine with three tapes:

- An **input tape** on which the input is written at the beginning.
- A **working tape** which is empty at the start of the computation. It is possible to read from this tape and to write on it.
- An **output tape** which is also empty at the start of the computation. It is only possible to write on it.

The amount of used space is then defined as the number of cells used on the working tape.

Complexity Classes

Other examples of complexity classes:

2-EXPTIME – the set of all problems for which there exists an algorithm with time complexity $2^{2^{O(n^k)}}$ where k is a constant

2-EXPSPACE – the set of all problems for which there exists an algorithm with space complexity $2^{2^{O(n^k)}}$ where k is a constant

ELEMENTARY – the set of all problems for which there exists an algorithm with time (or space) complexity

$$2^{2^{2^{\cdot^{\cdot^{\cdot^{2^{2^{O(n^k)}}}}}}}}$$

where k is a constant and the number of exponents is bounded by a constant.

Presburger Arithmetic

An example of a problem that is decidable but only with very high computational complexity:

Problem

Input: A closed formula of the first order predicate logic where the only predicate symbols are $=$ and $<$, the only function symbol is $+$, and the only constant symbols are 0 and 1.

Question: Is the given formula true in the domain of natural numbers (using the natural interpretation of all function and predicate symbols)?

For this problem, a (deterministic) algorithm is known with time complexity $2^{2^{2^{O(n)}}}$, and it is also known that every nondeterministic algorithm solving this problem must have time complexity at least $2^{2^{\Omega(n)}}$.