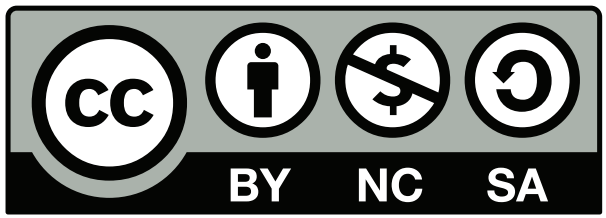
Complexity Classes

Dimitrios Dekas

Aristotle University of Thessaloniki



Theory of Computation

The field of theoretical computer science that examines the ability of a computational model to efficiently solve a particular problem using a certain algorithm.

Introduction

Theory of Computation

The field of theoretical computer science that examines the ability of a computational model to efficiently solve a particular problem using a certain algorithm.

Can be divided into two main sub-fields:

Theory of Computation

The field of theoretical computer science that examines the ability of a computational model to efficiently solve a particular problem using a certain algorithm.

Can be divided into two main sub-fields:

- Computability Theory

Introduction

Theory of Computation

The field of theoretical computer science that examines the ability of a computational model to efficiently solve a particular problem using a certain algorithm.

Can be divided into two main sub-fields:

- Computability Theory
- Computational Complexity Theory

Introduction

Computational Complexity Theory

It deals with the calculation of the resources required for the algorithmic solution of a problem. Classifies problems into complexity classes based on these costs.

Computational Complexity Theory

It deals with the calculation of the resources required for the algorithmic solution of a problem. Classifies problems into complexity classes based on these costs.

Important computational resources:

Introduction

Computational Complexity Theory

It deals with the calculation of the resources required for the algorithmic solution of a problem. Classifies problems into complexity classes based on these costs.

Important computational resources:

- Time

Introduction

Computational Complexity Theory

It deals with the calculation of the resources required for the algorithmic solution of a problem. Classifies problems into complexity classes based on these costs.

Important computational resources:

- Time
- Space

Complexity Classes

Equivalence classes that group problems of the same complexity with regard to the resource in question.

Complexity Classes

Equivalence classes that group problems of the same complexity with regard to the resource in question.

Each complexity class falls under the following definition:

Definition

A set of problems that are solvable by an automaton M, using the resource R, with complexity equal to $O(f(n))$, where n is the size of the input.

P (Polynomial Time)

Definition

The class containing every language L that is decidable by a deterministic Turing machine in polynomial time.

P (Polynomial Time)

Definition

The class containing every language L that is decidable by a deterministic Turing machine in polynomial time.

Problems included in this class

P (Polynomial Time)

Definition

The class containing every language L that is decidable by a deterministic Turing machine in polynomial time.

Problems included in this class

- AKS Primality Test

P (Polynomial Time)

Definition

The class containing every language L that is decidable by a deterministic Turing machine in polynomial time.

Problems included in this class

- AKS Primality Test
- Greatest Common Divisor

P (Polynomial Time)

Definition

The class containing every language L that is decidable by a deterministic Turing machine in polynomial time.

Problems included in this class

- AKS Primality Test
- Greatest Common Divisor
- ST-Connectivity

P (Polynomial Time)

Definition

The class containing every language L that is decidable by a deterministic Turing machine in polynomial time.

Problems included in this class

- AKS Primality Test
- Greatest Common Divisor
- ST-Connectivity
- Linear Programming

NP (Non-Deterministic Polynomial Time)

Definition

The class containing every language L that is decidable by a **non**-deterministic Turing machine in polynomial time.

NP (Non-Deterministic Polynomial Time)

Definition

The class containing every language L that is decidable by a **non**-deterministic Turing machine in polynomial time.

Problems included in this class

NP (Non-Deterministic Polynomial Time)

Definition

The class containing every language L that is decidable by a **non**-deterministic Turing machine in polynomial time.

Problems included in this class

- Discrete Log Problem

NP (Non-Deterministic Polynomial Time)

Definition

The class containing every language L that is decidable by a **non**-deterministic Turing machine in polynomial time.

Problems included in this class

- Discrete Log Problem
- Integer Factorization

NP (Non-Deterministic Polynomial Time)

Definition

The class containing every language L that is decidable by a **non**-deterministic Turing machine in polynomial time.

Problems included in this class

- Discrete Log Problem
- Integer Factorization
- Graph isomorphism

NP (Non-Deterministic Polynomial Time)

Definition

The class containing every language L that is decidable by a **non**-deterministic Turing machine in polynomial time.

Problems included in this class

- Discrete Log Problem
- Integer Factorization
- Graph isomorphism
- Minimum Circuit Size Problem

Cook-Levin Theorem

The Cook-Levin theorem proved that the boolean satisfiability problem belongs to the NP-Complete class. Succeeding this theorem, a plethora of different problems were also added to the NP-Complete class.

NP-Complete

Definition

The class containing every language L so that:

1. L belongs to the NP class.
2. Every language in NP is reducible to L in polynomial time.

NP-Complete

Definition

The class containing every language L so that:

1. L belongs to the NP class.
2. Every language in NP is reducible to L in polynomial time.

Problems included in this class

NP-Complete

Definition

The class containing every language L so that:

1. L belongs to the NP class.
2. Every language in NP is reducible to L in polynomial time.

Problems included in this class

- Boolean satisfiability (Circuit-SAT, SAT, 3-SAT)

NP-Complete

Definition

The class containing every language L so that:

1. L belongs to the NP class.
2. Every language in NP is reducible to L in polynomial time.

Problems included in this class

- Boolean satisfiability (Circuit-SAT, SAT, 3-SAT)
- Hamiltonian Path

NP-Complete

Definition

The class containing every language L so that:

1. L belongs to the NP class.
2. Every language in NP is reducible to L in polynomial time.

Problems included in this class

- Boolean satisfiability (Circuit-SAT, SAT, 3-SAT)
- Hamiltonian Path
- Graph coloring

NP-Complete

Definition

The class containing every language L so that:

1. L belongs to the NP class.
2. Every language in NP is reducible to L in polynomial time.

Problems included in this class

- Boolean satisfiability (Circuit-SAT, SAT, 3-SAT)
- Hamiltonian Path
- Graph coloring
- Sudoku

P vs NP

Question

Is it possible to discover an algorithm for each problem belonging to the NP class, so that it can be solved in polynomial time and therefore it ultimately belongs to the class P?

P vs NP

Question

Is it possible to discover an algorithm for each problem belonging to the NP class, so that it can be solved in polynomial time and therefore it ultimately belongs to the class P?

The above question can be reformulated by using the NP-Complete class as follows:

P vs NP

Question

Is it possible to discover an algorithm for each problem belonging to the NP class, so that it can be solved in polynomial time and therefore it ultimately belongs to the class P?

The above question can be reformulated by using the NP-Complete class as follows:

Alternative Formulation

Is it possible to discover an algorithm for a single problem belonging to the NP-Complete class, so that it can be solved in polynomial time and therefore it ultimately belongs to the class P?

What would happen if P = NP

- Complete understanding of protein folding.

What would happen if P = NP

- Complete understanding of protein folding.
- Rapid progress in the production process.

What would happen if P = NP

- Complete understanding of protein folding.
- Rapid progress in the production process.
- A significant portion of cryptographic systems will no longer be usable.

Co-NP

Definition

The class containing every language L whose negative snapshots are decidable by a non-deterministic Turing machine in polynomial time.

Co-NP-Complete

Definition

The class containing every language L so that:

1. L belongs to the Co-NP class.
2. Every language in Co-NP is reducible to L in polynomial time.

EXPTIME (DEXPTIME)

Definition

The class containing every language L that is decidable by a deterministic Turing machine in exponential time.

EXPTIME (DEXPTIME)

Definition

The class containing every language L that is decidable by a deterministic Turing machine in exponential time.

Problems included in this class

EXPTIME (DEXPTIME)

Definition

The class containing every language L that is decidable by a deterministic Turing machine in exponential time.

Problems included in this class

- Winning game strategies

EXPTIME (DEXPTIME)

Definition

The class containing every language L that is decidable by a deterministic Turing machine in exponential time.

Problems included in this class

- Winning game strategies
- T-step Turing maching acceptance.

NEXPTIME

Definition

The class containing every language L that is decidable by a **non**-deterministic Turing machine in exponential time.

NEXPTIME

Definition

The class containing every language L that is decidable by a **non**-deterministic Turing machine in exponential time.

Problems included in this class

NEXPTIME

Definition

The class containing every language L that is decidable by a **non**-deterministic Turing machine in exponential time.

Problems included in this class

- SUCCINCT NP-Complete problems

RP (Randomized Polynomial Time)

Definition

The class containing every language L for which there is a probabilistic Turing machine with the following properties:

1. It always runs in polynomial time.
2. If the correct answer to the question of accepting a string is NO, then it **always** returns NO.
3. If the correct answer to the question of accepting a string is YES, then it returns YES with a probability of at least $\frac{1}{2}$, otherwise it returns NO.

Co-RP

Definition

The class containing every language L for which there is a probabilistic Turing machine with the following properties:

1. It always runs in polynomial time.
2. If the correct answer to the question of accepting a string is YES, then it **always** returns YES.
3. If the correct answer to the question of accepting a string is NO, then it returns NO with a probability of at least $\frac{1}{2}$, otherwise it returns YES.

ZPP (Zero-Error Probabilistic Polynomial Time)

Definition

The class containing every language L for which there is a probabilistic Turing machine with the following properties:

1. It always runs in polynomial time.
2. It returns an answer equal to YES, NO or DO NOT KNOW.
3. The answer is always either DO NOT KNOW or the correct answer.
4. It returns DO NOT KNOW with probability $\leq \frac{1}{2}$.

Theorem

It is known that $ZPP = RP \cap Co - RP$.

BPP (Bounded-Error Probabilistic Polynomial Time)

Definition

The class containing every language L for which there is a probabilistic Turing machine with the following properties:

1. It always runs in polynomial time.
2. On any given run of the algorithm, there is a probability $\leq \frac{1}{3}$ of giving the wrong answer, whether the answer is YES or NO.

Quantum Computing Calculations

There is a similar complexity class, which deals with decision problems solved by quantum computers in polynomial time and is named Bounded-Error Quantum Polynomial Time.

PP (Probabilistic Polynomial Time)

Definition

The class containing every language L for which there is a probabilistic Turing machine with the following properties:

1. It always runs in polynomial time.
2. If the correct answer to the question of accepting a string is YES, then it returns YES with a probability $> \frac{1}{2}$.
3. If the correct answer to the question of accepting a string is NO, then it returns YES with a probability $\leq \frac{1}{2}$.

PSPACE

Definition

The class containing every language L that is decidable by a deterministic Turing machine that uses polynomial amount of space.

Problems included in this class

PSPACE

Definition

The class containing every language L that is decidable by a deterministic Turing machine that uses polynomial amount of space.

Problems included in this class

- First-order Theory Problems

PSPACE

Definition

The class containing every language L that is decidable by a deterministic Turing machine that uses polynomial amount of space.

Problems included in this class

- First-order Theory Problems
- QSAT

PSPACE

Definition

The class containing every language L that is decidable by a deterministic Turing machine that uses polynomial amount of space.

Problems included in this class

- First-order Theory Problems
- QSAT
- Succinct versions of many graph problems.

NPSPACE

Definition

The class containing every language L that is decidable by a **non**-deterministic Turing machine that uses polynomial amount of space.

EXPSPACE

Definition

The class containing every language L that is decidable by a deterministic Turing machine that uses exponential amount of space.

Problems included in this class

EXPSPACE

Definition

The class containing every language L that is decidable by a deterministic Turing machine that uses exponential amount of space.

Problems included in this class

- Ideal Membership Problem

EXPSPACE

Definition

The class containing every language L that is decidable by a deterministic Turing machine that uses exponential amount of space.

Problems included in this class

- Ideal Membership Problem
- Problems related to Vector Addition Systems

EXPSPACE

Definition

The class containing every language L that is decidable by a deterministic Turing machine that uses exponential amount of space.

Problems included in this class

- Ideal Membership Problem
- Problems related to Vector Addition Systems
- Temporal Planning with Concurrent Actions

NEXPSPACE

Definition

The class containing every language L that is decidable by a **non**-deterministic Turing machine that uses exponential amount of space.

Savitch's Theorem

Proved by Walter Savitch in 1970, it gives a foundational relationship between deterministic and non-deterministic space complexity. The theorem states that for any function $f \in \Omega\left(log\left(n()\right)\right.$, it holds that:

$$NSPACE\left(f\left(n\right)\right) \subseteq DSPACE\left(f\left(n\right)\right)^2$$

Savitch's Theorem

Proved by Walter Savitch in 1970, it gives a foundational relationship between deterministic and non-deterministic space complexity. The theorem states that for any function $f \in \Omega\left(log\left(n()\right)\right.$, it holds that:

$$NSPACE\left(f\left(n\right)\right) \subseteq DSPACE\left(f\left(n\right)\right)^2$$

Two useful results arise from the above theorem:

Savitch's Theorem

Proved by Walter Savitch in 1970, it gives a foundational relationship between deterministic and non-deterministic space complexity. The theorem states that for any function $f \in \Omega\left(log\left(n()\right)\right.$, it holds that:

$$NSPACE\left(f\left(n\right)\right) \subseteq DSPACE\left(f\left(n\right)\right)^2$$

Two useful results arise from the above theorem:

1. $PSPACE = NPSPACE$

Savitch's Theorem

Proved by Walter Savitch in 1970, it gives a foundational relationship between deterministic and non-deterministic space complexity. The theorem states that for any function $f \in \Omega\left(log\left(n()\right)\right.$, it holds that:

$$NSPACE\left(f\left(n\right)\right) \subseteq DSPACE\left(f\left(n\right)\right)^2$$

Two useful results arise from the above theorem:

1. $PSPACE = NPSPACE$
2. $EXPSPACE = NEXPSPACE$

Acknowledgements

Thank you for your attention.
Please feel free to make any question.

Special thanks to Thanasis Vranis for his valuable assistance.