Video 43: Understanding Complexity

- Complexity Classes
- Tractable Problems
- Untractable Problems

Complexity Classes

TABLE 1 Commonly Used Terminology for the Complexity of Algorithms.

Complexity	*Terminology*
$\Theta(1)$	Constant complexity
$\Theta(\log n)$	Logarithmic complexity
$\Theta(n)$	Linear complexity
$\Theta(n \log n)$	Linearithmic complexity
$\Theta(n^b)$	Polynomial complexity
$\Theta(b^n)$, where $b > 1$	Exponential complexity
$\Theta(n!)$	Factorial complexity

Effect of Complexity

TABLE 2 **The Computer Time Used by Algorithms.**

Problem Size	*Bit Operations Used*					
n	$\log n$	n	$n \log n$	n^2	2^n	$n!$
10	3×10^{-11} s	10^{-10} s	3×10^{-10} s	10^{-9} s	10^{-8} s	3×10^{-7} s
10^2	7×10^{-11} s	10^{-9} s	7×10^{-9} s	10^{-7} s	4×10^{11} yr	*
10^3	1.0×10^{-10} s	10^{-8} s	1×10^{-7} s	10^{-5} s	*	*
10^4	1.3×10^{-10} s	10^{-7} s	1×10^{-6} s	10^{-3} s	*	*
10^5	1.7×10^{-10} s	10^{-6} s	2×10^{-5} s	0.1 s	*	*
10^6	2×10^{-10} s	10^{-5} s	2×10^{-4} s	0.17 min	*	*

A bit operation is assumed to take 10^{-11} seconds, i.e., in one second we perform 100 billion bit operations. Times of more than 10^{100} years are indicated with an *.

Tractable Problems

Commonly, a problem is considered **tractable**, if there exists an algorithm and some d such that the algorithm can for input of size n produce the result with $O(n^d)$ operations.

This is the class **P** of **polynomial complexity.**

If such an algorithm does not exist, the problem is considered **intractable**.

- For large d and large inputs, it is often not so clear that the problem is really tractable in a practical sense
- **Example**: multiplying two very large matrices

The Class NP

- The class **NP** consists of those problems for which the correctness of a proposed solution can be verified in polynomial time
- **NP** stands for **nondeterministic polynomial time**
  - Every problem in **NP** can be solved in exponential time
  - Open problem (since 50 years): **P** = **NP**?
  - General assumption: **P** ≠ **NP**
  - **The Clay Mathematics Institute has offered a prize of $1,000,000 for a solution.**

Example of an NP Problem

Knapsack Decision Problem: Given a set S of n items, each item with a value and a weight, find the subset of items that exceeds a minimal value while fitting a maximal weight.

- No polynomial algorithm is known
- Checking correctness of solution is easy
- Exponential algorithm: try out all possible subsets

NP-Complete Problems

NP-complete problems are problems in NP, such that if a polynomial algorithm is found for the problem, all other problems in NP can also be solved in polynomial time.

Examples

- Knapsack Decision Problem
- 3-SAT

3-SAT

Satisfiability of Propositional Statements: NP-complete

3-SAT is a more special problem:

- Deciding satisfiability for formulas in conjunctive normal form, where each clause has at most 3 variables is NP-complete

Example: $(p \vee p \vee q) \wedge (\neg p \vee \neg q \vee \neg q) \wedge (\neg p \vee q \vee q)$ is a formula from the set of 3-SAT formulas

Summary

Tractable Problem: There exists a polynomial time algorithm to solve this problem. These problems are said to belong to the **Class P**.

Intractable Problem: There does not exist a polynomial time algorithm to solve this problem.

Unsolvable Problem: There does not exist any algorithm to solve the problem, e.g., halting problem.

Class NP: Solution can be checked in polynomial time. But no polynomial time algorithm has been found for finding a solution to problems in this class.

NP Complete Class: If you find a polynomial time algorithm for one member of the class, it can be used to solve all the problems in the class.