The complexity class NP [1]

Camilo Rocha & Miguel Romero

October 15, 2019

Pontificia Universidad Javeriana de Cali

Análisis y Diseño de Algoritmos

Class P

The **class P** consists of those **problems** that are solvable in **polynomial time**. More specifically, they are problems that can be solved in time $O(n^k)$ for some constant k, where n is the size of the input to the problem.

Class P

Formally defined, the **complexity class P** is the set of concrete decision problems that are polynomial-time solvable.

Class P

$$\mathsf{P} = \{L \subseteq \{0,1\}^* : \text{there exists an algorithm } A \text{ that } \textbf{decides } L \text{ in polynomial time}\}.$$

In fact, P is also the class of languages that can be **accepted** in polynomial time.

Theorem 1
P $= \{L : L$ *is* **accepted** *by a polynomial-time algorithm*$\}$.

Class P

Almost all the algorithms we have studied thus far have been **polynomial-time algorithms**: on inputs of size n, their worst-case running time is $O(n^k)$ for some constant k.

Class NP

There are also problems that can be solved, but not in time $O(n^k)$ for any constant k. Generally, we think of problems that are solvable by polynomial-time algorithms as being **tractable**, or easy, and problems that require superpolynomial time as being **intractable**, or hard.

Polynomial-time verification

We define a **verification algorithm** as being a two-argument algorithm A, where one argument is an ordinary input string x and the other is a binary string y called a **certificate**. A two-argument algorithm A **verifies** an input string x if there exists a certificate y such that $A(x, y) = 1$.

Polynomial-time verification

The language verified by a verification algorithm A is

$\mathsf{L} = \{x \in \{0,1\}^* :$ there exists $y \in \{0,1\}^*$ such that $A(x,y) = 1\}$.

Intuitively, an algorithm A verifies a language L if for any string $x \in L$, there exists a certificate y that A can use to prove that $x \in L$. Moreover, for any string $x \notin L$, there must be no certificate proving that $x \in L$.

The complexity class NP

The **complexity class NP** is the class of languages that can be **verified by a polynomial-time algorithm**. More precisely, a language L belongs to NP if and only if there exist a two-input polynomial-time algorithm A and a constant c such that

$$\mathsf{L} = \{x \in \{0,1\}^* : \text{there exists a certificate } y \text{ with } |y| = O(|x|^c) \text{ such that } A(x,y) = 1\}.$$

We say that algorithm A **verifies** language L in **polynomial time**.

NP-complete

The **NP-complete** problems, is an interesting class of problems whose status is unknown.

No polynomial-time algorithm has yet been discovered for an **NP-complete problem**, nor has anyone yet been able to prove that no polynomial-time algorithm can exist for any one of them.

This so-called P$\neq$NP question has been one of the deepest, most perplexing open research problems in theoretical computer science since it was first posed in 1971.

Reducibility

Thus, our understanding of the precise relationship between P and NP is woefully incomplete. Nevertheless, even though we might not be able to prove that a particular problem is intractable, if we can prove that it is NP-complete, then we have gained valuable information about it.

Questions?

References

T. Cormen, C. Leiserson, R. Rivest, and C. Stein.
Introduction to Algorithms.
Computer science. MIT Press, 2009.

Thanks!