3.10pt

CS711008Z Algorithm Design and Analysis

Lecture 3. Problem hardness and polynomial-time reduction

Dongbo Bu

Institute of Computing Technology
Chinese Academy of Sciences, Beijing, China

Outline

- Problem intrinsic property: hardness
- Reduction: to identify the relationship between different problems;

NP-Completeness Cartoon: Bandersnatch problem

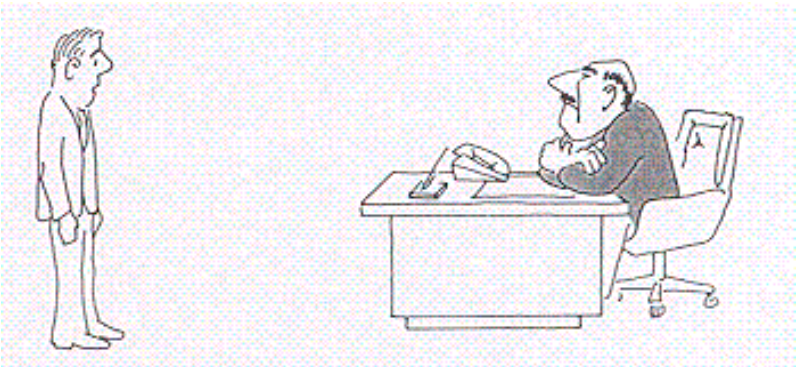
- One day your boss calls you into his office and confides that the company is about to enter the highly competitive "bandersnatch" market.
- A good method is needed for determining whether or not any given set of specifications for a new bandersnatch component can be met and, if so, for constructing a design that meets them.
- Since you are the company's chief algorithm official, your charge is to find an efficient algorithm for doing this.

1

[1] Excerpted from Computer and Intractability (by Garey and Johnson)

Trial 1: attempt to solve this problem

- Some weeks later, you have not been able to come up with any algorithm substantially better than searching through all possible designs. This would involve years of computation time for just one set of specifications.
- You simply return to your boss's office and report: "I can't find an efficient algorithm, I guess I'm just too dumb. "



- But perhaps this is unfair to you: the problem might be intrinsically hard.

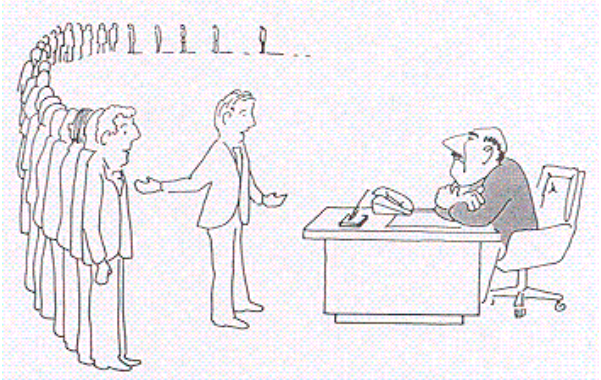
Trial 2: try to prove the hardness directly

- So it would be much better if you could prove that the bandersnatch problem is inherently intractable, that no algorithm could possibly solve it quickly.
- Then you could stride confidently into the boss's office and proclaim: " I can't find an efficient algorithm, because no such algorithm is possible. "

- Unfortunately, proving inherent intractability can be just as hard as finding efficient algorithms.

Trial 3: to show the relative hardness with other hard problems

- For such a "grey area" problem, an alternative way is to prove that the problem is "just as hard as" a large number of other problems that are widely recognize as being difficult and that have been confounding the experts for years.
- "I can't find an efficient algorithm, but neither can all these famous people. "



Trial 3: to show the relative hardness with other hard problems. cont'd

- Two advantages:
  1. At the very least, this would inform your boss that it would do no good to fire you and hire another expert on algorithms.
  2. More importantly, you can spend your time looking for efficient algorithms that solve various special cases of the general problem.

Problem and its hardness

Hardness or complexity: an intrinsic property of problem

- Problems can be:
  - Easy (existing polynomial-time algorithm): StableMatching problem;
  - NP-hard: Satisfiability problem;
  - Truly hard (provably non-polynomial, exponential): Given a Turing machine, does it halt in at most k steps?
  - Impossible: HALT problem

Abstracting problem

- Problem: consisting of `INPUT` and `OUTPUT` parts.
  Formally, a problem can be described as a *relation* $P \subseteq I \times S$, where I denotes the **problem input**, and S denotes the set of problem **solutions**.
- Instance: a particular `INPUT`.