NP-Completeness

Arnav Gupta

March 30, 2024

Contents

1 NP Class

For a problem X, represent an instance of X as a binary string S.

A problem X is **in NP** if there is a polynomial time verification algorithm ALG_X such that the input S is a yes-instance if and only if there is a proof (certificate) t which is a binary string of length poly($|S|$) so that $ALG_X(S,t)$ returns yes.

Vertex Cover, Clique, IS, HC, HP, Subset-Sum, and 3-SAT are in NP.

Not all problems are in NP.

co-NP: the set of decison problems whose no-instances can be certified in polynomial time

Every polynomial time solvable problem is in NP, where **P** is the set of decision problems that can be solved in polynomial time.

$$P \subseteq NP$$

The most important open problme in theoretical computer science is $P = NP$?

The name NP comes from Non-deterministic Polynomial time, where a non-deterministic machine has the power to correctly guess a solution.

2 NP-Completeness

A problem is NP-complete if it is the hardest problem in NP.

A problem $X \in NP$ is **NP-complete** if $Y \leq_P X$ for all $Y \in NP$

$P = NP$ if an only if an NP-complete problem can be solved in poly-time

3-SAT is NP-Complete, proven by Cook and Levin

Consequences of this are

- for any NP problem X, if one can prove 3-SAT $\leq_P X$, then X is NP-complete
- to prove that a problem $X \in NP$ is NP-complete, just need to find an NP-complete problem Y and prove that $Y \leq_P X$

Some NP-complete problems are

- 3-SAT, SAT
- independent set, vertex cover, clique
- (directed) Hamiltonian cycle, Hamiltonian path
- travelling salesman
- subset sum
- 0/1 knapsack

2.1 Circuit Satisfiability

Circuit-SAT is defined by

- instance: a circuit is a DAG with labels on the vertices
- inputs labelled by and/or/not
- there is a marked vertex v for output
- **problem**: is there a choice of boolean x_i that makes v true

Plan for the Cook-Levin Theorem is

1. show that Circuit-SAT is NP-complete
2. show that Circuit-SAT $\leq_P$ 3-SAT

Sketch of proof that Circuit-SAT is NP-complete uses the following idea:

- given: instance S of $A \in NP$
- want: proof (certificate) t such that $ALG_A(S, t)$ is true
- verification algorithm $ALG_A(S, t)$ can be turned into a circuit with t as input
- call Circuit-SAT to find t