COL351: Analysis and Design of Algorithms

Ragesh Jaiswal, CSE, IITD

Computational Intractability: NP and NP-complete

- We said that the problems INDEPENDENT-SET, VERTEX-COVER, SAT *seem* hard.

- We said that the problems INDEPENDENT-SET, VERTEX-COVER, SAT seem hard.
- Polynomial-time reductions just give pair-wise relationships between problems.
- Is there a common theme that binds all these problems in one computational class?

- We said that the problems INDEPENDENT-SET, VERTEX-COVER, SAT seem hard.
- Polynomial-time reductions just give pair-wise relationships between problems.
- Is there a common theme that binds all these problems in one computational class?
- Let us try to extract a theme that is common to some of the problems we saw:
  - INDEPENDENT-SET: Given (G, k), determine if G has an independent set of size at least k.
  - VERTEX-COVER: Given (G, k), determine if G has a vertex cover of size at most k.
  - SAT: Given a Boolean formula Ω in CNF, determine if the formula is satisfiable.

- Let us try to extract a theme that is common to some of the problems we saw:
  - INDEPENDENT-SET: Given (G, k), determine if G has an independent set of size at least k.
    - Suppose there is an independent set of size at least k and someone gives such a subset as a certificate. Then we can verify this certificate quickly.
  - VERTEX-COVER: Given (G, k), determine if G has a vertex cover of size at most k.
    - Suppose there is a vertex cover of size at most k and someone gives such a subset as a certificate. Then we can verify this certificate quickly.
  - SAT: Given a Boolean formula Ω in CNF, determine if the formula is satisfiable.
    - Suppose the formula Ω is satisfiable and someone gives such a satisfying assignment as a certificate. Then we can verify this certificate quickly.

- Problem encoding and algorithm:
  - An *instance* of a problem can be encoded using a finite string s.
  - A *decision* problem X can be thought of as a set of strings on which the answer is true (or 1).
  - We say that an algorithm A solves a problem X if for all strings s, $A(s) = 1$ if and only if s is in X.
  - We say that an algorithm A has a polynomial running time if there is a polynomial p such that for every string s, A terminates on input s in at most $O(p(|s|))$ steps.

- Efficient Certification:
  - We say that algorithm B is an efficient certifier for a problem X, iff the following holds:
    - B is a polynomial time algorithm that takes two input string s and t.
    - There is a polynomial p such that for every string s, we have $s \in X$ if and only if there exists a string t such that $|t| \leq p(|s|)$ and $B(s,t) = 1$.

- Efficient Certification:
  - We say that algorithm B is an efficient certifier for a problem X, iff the following holds:
    - B is a polynomial time algorithm that takes two input string s and t.
    - There is a polynomial p such that for every string s, we have $s \in X$ if and only if there exists a string t such that $|t| \leq p(|s|)$ and $B(s, t) = 1$.
- Note that B does not solve the problem but only verifies a proposed solution.
- Can we use B to solve the problem?

- Efficient Certification:
  - We say that algorithm B is an efficient certifier for a problem X, if the following holds:
    - B is a polynomial time algorithm that takes two input string s and t.
    - There is a polynomial p such that for every string s, we have $s \in X$ if and only if there exists a string t such that $|t| \leq p(|s|)$ and $B(s,t) = 1$.
- Note that B does not solve the problem but only verifies a proposed solution.
- Can we use B to solve the problem? Yes
- Can we use B to solve the problem efficiently?

- Efficient Certification:
  - We say that algorithm B is an efficient certifier for a problem X, if the following holds:
    - B is a polynomial time algorithm that takes two input string s and t.
    - There is a polynomial p such that for every string s, we have $s \in X$ if and only if there exists a string t such that $|t| \leq p(|s|)$ and $B(s, t) = 1$.
- Note that B does not solve the problem but only verifies a proposed solution.
- Can we use B to solve the problem? Yes
- Can we use B to solve the problem efficiently?

Definition (NP)

A problem is said to be in NP iff there exists an efficient certification algorithm for the problem.

- Efficient Certification:
  - We say that algorithm B is an efficient certifier for a problem X, if the following holds:
    - B is a polynomial time algorithm that takes two input string s and t.
    - There is a polynomial p such that for every string s, we have $s \in X$ if and only if there exists a string t such that $|t| \leq p(|s|)$ and $B(s, t) = 1$.

Definition (NP)

A problem is said to be in NP iff there exists an efficient certification algorithm for the problem.

- NP stands for Non-deterministic Polynomial time.
  - Non-deterministic algorithms are allowed to make non-deterministic choices (guesswork). Such algorithms can guess the certificate t for an instance s.

Definition (NP)

A problem is said to be in NP iff there exists an efficient certification algorithm for the problem.

Definition (P)

A problem is said to be in P iff there exists an efficient algorithm that solves the problem.

- Theorem: P $\subseteq$ NP.

Definition (NP)

A problem is said to be in NP iff there exists an efficient certification algorithm for the problem.

Definition (P)

A problem is said to be in P iff there exists an efficient algorithm that solves the problem.

- Theorem: P $\subseteq$ NP.
- Claim 1: INDEPENDENT-SET $\in$ NP
  - Proof sketch: The certificate is an independent set of size at least k. The certifier checks if the given set if indeed an independent set of size at least k.

Definition (NP)

A problem is said to be in NP iff there exists an efficient certification algorithm for the problem.

Definition (P)

A problem is said to be in P iff there exists an efficient algorithm that solves the problem.

- Theorem: P $\subseteq$ NP.
- Claim 1: INDEPENDENT-SET $\in$ NP
  - Proof sketch: The certificate is an independent set of size at least k. The certifier checks if the given set if indeed an independent set of size at least k.
- Claim 2: SAT $\in$ NP
  - Proof sketch: The certificate is a satisfying assignment.The certifier checks if the assignment makes all clauses true.

Definition (NP)

A problem is said to be in NP iff there exists an efficient certification algorithm for the problem.

Definition (P)

A problem is said to be in P iff there exists an efficient algorithm that solves the problem.

- Theorem: $\mathsf{P} \subseteq \mathsf{NP}$.
- Is $\mathsf{P} = \mathsf{NP}$?
- What are the hardest problems in NP?

Definition (NP)

A problem is said to be in NP iff there exists an efficient certification algorithm for the problem.

Definition (P)

A problem is said to be in P iff there exists an efficient algorithm that solves the problem.

- Theorem: P $\subseteq$ NP.
- Is P = NP?
- What are the hardest problems in NP?
- A problem $X \in$ NP is the hardest problem in NP if for all problems $Y \in$ NP, $Y \leq_p X$.
- Such problems are called NP-complete problems.

Definition (NP)

A problem is said to be in NP iff there exists an efficient certification algorithm for the problem.

Definition (P)

A problem is said to be in P iff there exists an efficient algorithm that solves the problem.

Definition (NP-complete)

A problem X is said to be NP-complete iff the following two properties hold:

1. $X \in$ NP.
2. For all $Y \in$ NP, $Y \leq_p X$.

Definition (NP)

A problem is said to be in NP iff there exists an efficient certification algorithm for the problem.

Definition (P)

A problem is said to be in P iff there exists an efficient algorithm that solves the problem.

Definition (NP-complete)

A problem X is said to be NP-complete iff the following two properties hold:

1. $X \in$ NP.
2. For all $Y \in$ NP, $Y \leq_p X$.

- How do we show that there is a problem that is NP-complete?

Definition (NP)

A problem is said to be in NP iff there exists an efficient certification algorithm for the problem.

Definition (P)

A problem is said to be in P iff there exists an efficient algorithm that solves the problem.

Definition (NP-complete)

A problem X is said to be NP-complete iff the following two properties hold:

1. $X \in \text{NP}$.
2. For all $Y \in \text{NP}$, $Y \leq_p X$.

- How do we show that there is a problem that is NP-complete?
- Suppose by some magic we have shown that SAT is NP-complete, does that mean that there are more NP-complete problems?

Definition (NP)

A problem is said to be in NP iff there exists an efficient certification algorithm for the problem.

Definition (P)

A problem is said to be in P iff there exists an efficient algorithm that solves the problem.

Definition (NP-complete)

A problem X is said to be NP-complete iff the following two properties hold:

1. $X \in \text{NP}$.
2. For all $Y \in \text{NP}$, $Y \leq_p X$.

Theorem (Cook-Levin Theorem)

3-SAT is NP*-complete.*

Definition (NP-complete)

A problem X is said to be NP-complete iff the following two properties hold:

1. $X \in \mathrm{NP}$.
2. For all $Y \in \mathrm{NP}$, $Y \leq_p X$.

Theorem (Cook-Levin Theorem)

3-SAT is NP*-complete.*

Proof sketch

- Claim 1: CIRCUIT-SAT is NP-complete.
- Claim 2: CIRCUIT-SAT $\leq_p$ 3-SAT.

Theorem (Cook-Levin Theorem)

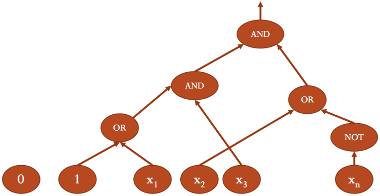
3-SAT is NP-*complete.*

Proof sketch

- Claim 1: CIRCUIT-SAT is NP-complete.
- Claim 2: CIRCUIT-SAT $\leq_p$ 3-SAT.

- Circuit: A directed acyclic graph where each node is either:
  - Constant nodes: Labeled 0/1
  - Input nodes: These denote the variables
  - Gates: AND, OR, and NOT

  There is a single output node.



Theorem (Cook-Levin Theorem)

3-SAT is NP*-complete.*

Proof sketch

- Claim 1: CIRCUIT-SAT is NP-complete.
- Claim 2: CIRCUIT-SAT $\leq_p$ 3-SAT.

- Circuit: A directed acyclic graph where each node is either:
  - Constant nodes: Labeled 0/1
  - Input nodes: These denote the variables
  - Gates: AND, OR, and NOT

  There is a single output node.

Problem

CIRCUIT-SAT: Given a circuit, determine if there is an input such that the output of the circuit is 1.

Theorem (Cook-Levin Theorem)

3-SAT is NP-complete.

Proof sketch

- Claim 1: CIRCUIT-SAT is NP-complete.
  - Fact: For every algorithm that runs in time polynomial in the input size n, there is an equivalent circuit of size polynomial in n.
  - Given an input instance s of any NP problem X, consider the equivalent circuit for the efficient certifier of X. The input gates of this circuit has s and t.
  - $s \in X$ if and only if this circuit is satisfiable.
- Claim 2: CIRCUIT-SAT $\leq_p$ 3-SAT.
  - For any circuit, we can write an equivalent 3-SAT formula.

Problem

CIRCUIT-SAT: Given a circuit, determine if there is an input such that the output of the circuit is 1.

End