EECS 336: Lecture 11: Introduction to Algorithms Intractability

Deriving NP: NP, CIRCUIT-SAT

Reading: 8.3

Last Time:

- 3-SAT $\leq_P$ INDEP-SET
- 3-SAT $\leq_P$ HC
- 3-SAT $\leq_P$ 3D-MATCHING

Today:

- decision problems
- $\mathcal{NP}$ problems
- "Notorious Problem" NP
- NP $\leq_P$ CIRCUIT-SAT

Goal: a framework for showing problems are intractable.

Challenge: lower bounds for algorithms are very difficulty to prove.

Approach: reduce believed hard problem to new problem to show that new problem is probably also hard.

Challenge: problems look quite different, e.g., 3-SAT, HC, INDEP-SET

Approach: decision problems.

Decision Problems

"problems with yes/no answer"

Def: A **decision problem** asks "does a feasible solution exist?"

Example: is there satisfying assignment $\mathbf{z}$ for 3-SAT formula f?

Example: is there independent set S in INDEP-SET graph (V, E) with size at least θ?

Note: Can convert optimization problem to decision problem

Def: the decision problem X_d for optimization problem X has input (x, θ) = "does instance x of X have a feasible solution with value at most (or at least) θ?"

Fact: $X_d \leq_P X$

Proof: obvious.

Theorem: $X \leq_P X_d$

Proof:

- identify reasonable range for OPT, e.g., $[1, h]$
- binary search with X_d solver.

QED.

Note: This is not a one-call reduction.

A notoriously hard problem

"one problem to solve them all"

Note: all example problems have **short certificates** that could easily verify "yes" instances.

Def: $\mathcal{NP}$ is the class of problem that have short (polynomial sized) certificates that can easily (in polynomial time) verify "yes" instances.

Historical Note: $\mathcal{NP}$ = non-deterministic polynomial time

"a nondeterministic algorithm could guess the certificate and then verify it in polynomial time"

Defs:

- Problem X is in $\mathcal{NP}$ if exists short easily-verifiable certificate.
- Problem X is $\mathcal{NP}$-hard if $\forall Y \in \mathcal{NP}$, $Y \leq_{\mathcal{P}} X$.
- Problem X is in $\mathcal{NP}$ if $X \in \mathcal{NP}$ and X is $\mathcal{NP}$-hard.

Lemma: INDEP-SET $\in \mathcal{NP}$

Lemma: 3-SAT $\in \mathcal{NP}$

Lemma: HC $\in \mathcal{NP}$

Goal: show INDEP-SET, SAT, NP are $\mathcal{NP}$-complete.

Note: Not all problems are in $\mathcal{NP}$.

E.g., unsatisfiability, chess

Notorious Problem: NP

input:

- decision problem verifier program VP.
- polynomial $p(\cdot)$.
- decision problem instance: x

output:

- "Yes" if exists certificate c such that $VP(x, c)$ has "verified = true" at computational step $p(|x|)$.
- "No" othersiwe.

Fact: NP is $\mathcal{NP}$-complete.

Note: Unknown whether $\mathcal{P} = \mathcal{NP}$.

Note: $\leq_{\mathcal{P}}$ is transitive: if $Y \leq_{\mathcal{P}} X$ and $X \leq_{\mathcal{P}} Z$ then $Y \leq_{\mathcal{P}} Z$.

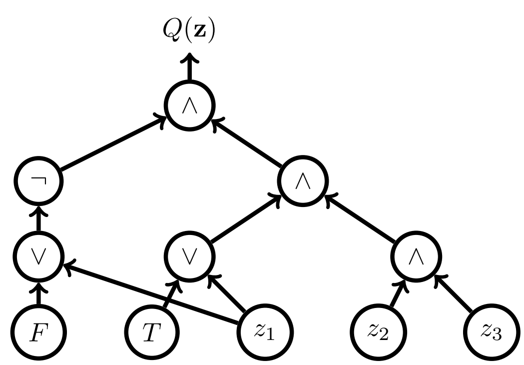
Plan:

1. NP $\leq_{\mathcal{P}}$ ... $\leq_{\mathcal{P}}$ 3-SAT
2. 3-SAT $\leq_{\mathcal{P}}$ INDEP-SET
3. 3-SAT $\leq_{\mathcal{P}}$ HC $\leq_{\mathcal{P}}$ TSP
4. 3-SAT $\leq_{\mathcal{P}}$ 3D-MATCHING

Agenda: Find a first simple $\mathcal{NP}$-complete problem.

Circuit Satisfiability

Example:



Problem: CIRCUIT-SAT

input: boolean circuit $Q(\mathbf{z})$

- directed acyclic graph $G = (V, E)$
- internal nodes labeled by logical gates:

  "and", "or", or "not"
- leaves labeled by variables or constants

  $T, F, z_1, ..., z_n$.
- root r is output of circuit

output:

- "Yes" if exists $\mathbf{z}$ with $Q(\mathbf{z}) = T$
- "No" otherwise.

Theorem: CIRCUIT-SAT is $\mathcal{NP}$-hard.

Part I: forward instance construction

convert NP instance (VP, p, x) to CIRCUIT-SAT instance Q.

- $\mathrm{VP}(\cdot, \cdot)$ polynomial time

  $\Rightarrow$ computer can run it in poly steps.
- each step of computer is circuit.
- output of one step is input of next step
- unroll $p(|x|)$ steps of computation

  $\Rightarrow \exists$ poly-size circuit $Q'(\mathbf{x}, \mathbf{c}) = \mathrm{VP}(x, c)$
- hardcode $\mathbf{x}$: $Q(\mathbf{c}) = Q'(\mathbf{x}, \mathbf{c})$

Part II-III: backward/forward certificate construction

- $\mathbf{c} \Leftrightarrow c$