Reading: 8.4-8.5

Last time:
- tractability and intractability
- decision problems

Today:
- \( \mathcal{NP} \)-completeness
- “notorious problem” \( \mathcal{NP} \).
- reductions from 3-SAT.

Problem 1: Independent Set (INDEPENDENT-SET)

input: \( G = (V, E) \)
output: \( S \subseteq V \)
- satisfying \( \forall v \in S, (u, v) \notin E \)
- maximizing \( |S| \)

Problem 2: SAT

Problem 3: Traveling Salesman (TSP)

input:
- \( G = (V, E) \), complete graph.

Problem 4: 3-SAT

input: boolean formula \( f(z) \)
- in conjunctive normal form (CNF)
- three literals per or-clause
- or-clauses anded together.
output:
- “Yes” if assignment \( z \) with \( f(z) = T \) exists
- “No” otherwise.

Problem 5: Hamiltonian Cycle (HC)

input: \( G = (V, E) \) (directed)
output: cycle \( C \) to visit each vertex exactly once.

Note: since \( X_d =_p X \), we write “X” but we mean “\( X_d \)”
A notoriously hard problem

“one problem to solve them all”

Note: all example problem have short certificates that could easily verify “yes” instance.

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

Historical Note: \( \mathcal{NP} = \text{non-deterministic polynomial time} \)

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

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

E.g., unsatisfiability.

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

Lemma: \( \text{INDEP-SET} \in \mathcal{NP} \).

Lemma: \( \text{SAT} \in \mathcal{NP} \).

Lemma: \( \text{TSP} \in \mathcal{NP} \).

Goal: show \( \text{INDEP-SET}, \text{SAT}, \text{TSP} \) are \( \mathcal{NP} \)-complete.

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” otherwise.

Fact: \( \mathcal{NP} \) is \( \mathcal{NP} \)-complete.

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

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

Plan:
1. \( \mathcal{NP} \leq_{P} \cdots \leq_{P} 3\text{-SAT} \)
2. \( 3\text{-SAT} \leq_{P} \text{INDEP-SET} \)
3. \( 3\text{-SAT} \leq_{P} HC \leq_{P} \text{TSP} \)
Problem: Hamiltonian Cycle

input: \( G = (V, E) \) (directed)

output: cycle \( C \) to visit each vertex exactly once.

**Lemma:** hamiltonian cycle is \( \mathcal{NP} \)-hard

**Proof:** (reduction from 3-SAT)

**Step 1:** construction

- turn 3-SAT formula \( f \) in to graph \( G \) with hamiltonian cycle iff \( f \) is satisfiable.
- idea: variable = isolated path, right-to-left = true, left-to-right = false.
- idea: clause is node, which needs to be hit by at most one literal being true.
- construction:
  - left-right path per variable.
  - splice in clause nodes.

**Step 2:** runtime.

**Step 3:** correctness.

---

TSP

**Lemma 0.1** TSP is \( \mathcal{NP} \)-hard.

**Proof:** reduction from Hamiltonian Cycle

- encode edges with cost 1
- encode non-edges with cost \( n \).

\( \Rightarrow \) exists HC iff TSP cost \( \leq n \)