Overview

This week we will focus on reviewing the core concepts involved with NP, polynomial-time reducibility, and NP-Completeness.

1 Concept Review

1.1 Nondeterministic Polynomial Time

A verifier for a language \( L \) is an algorithm \( V \) such that \( L = \{ x : V \text{ accepts } \langle x, y \rangle \text{ for some string } y \} \). A polynomial time verifier is the one that runs in time polynomial in \(|x|\) on input \( \langle x, y \rangle \).

Definition 1.1 \( \text{NP} \) is the class of languages with polynomial time verifiers.

Usually there are two equivalent describes of \( \text{NP} \), however we do always use the one above.

1.2 Polynomial-time reducibility and NP-completeness

Definition 1.2 We say a language \( L_1 \subseteq \Sigma_1^* \) is polynomial-time reducible to another language \( L_2 \subseteq \Sigma_2^* \) (i.e., \( L_1 \leq_p L_2 \)) if there is a polynomial-time computable function \( f : \Sigma_1^* \rightarrow \Sigma_2^* \) such that for every \( x \in \Sigma_1^* \), \( x \in L_1 \iff f(x) \in L_2 \).

In this definition, it requires the reductions to be polynomial time, which is different with the previous definition of computable reductions.

Definition 1.3 A language \( L \) is \( \text{NP} \)-complete if and only if,

1. \( L \in \text{NP} \),

2. Every language in \( \text{NP} \) is polynomial-time reducible to \( L \).

Intuitively, \( \text{NP} \)-complete languages are the hardest \( \text{NP} \) languages. If there is an \( \text{NP} \)-complete problem which can be resolved in polynomial time, then \( \text{P} = \text{NP} \).

1.3 Cook-Levin Theorem

Theorem 1.1 SAT (Boolean satisfiability) is \( \text{NP} \)-complete.

To prove this theorem, we have to show that every \( \text{NP} \) problem is polynomial reducible to SAT. Let \( L \) be an \( \text{NP} \) problem which decided by a nondeterministic TM \( M \), then the main idea of the reduction is to describe computations of \( M \) by boolean variables.
2 Exercises

Exercise 2.1 Determine, with proof, whether $NP$ is closed under Kleene star.

Exercise 2.2 We define Hitting Set to be the problem of determining, given a family of sets $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ and an integer $B$, whether there is a set $H$ with $B$ or fewer elements such that for each $S_i \in \mathcal{F}$ we have $S_i \cap H \neq \emptyset$. Prove that Hitting Set is NP-complete. 

Hint: Reduce from Vertex Cover.

Exercise 2.3 We have shown in class that the languages $\text{CLIQUE} = \{(G, k) : \text{The graph } G \text{ contains a clique of size } k\}$ and $\text{TSP} = \{(m, D, B) : \text{There exists a tour of the } m \text{ cities with distance function } D \text{ of length } \leq B\}$ are in $NP$. Now show that if $P = NP$, then

(a) Given a graph $G$, we can determine in polynomial time the size of the largest clique in $G$

(b) Given $m$ cities with distance function $D$, we can determine in polynomial time the length of the shortest tour of all the cities.

Exercise 2.4 Given a list of currencies $1, \ldots, n$, and a matrix $M$ with positive rational entries, where $M_{i,j}$ is the exchange rate between currencies $i$ and $j$, we say there is an arbitrage of value at least $v$ if there exists a sequence of currency exchanges $(a_1, \ldots, a_k, a_{k+1} = a_1)$, $k \geq 2$, $a_1, \ldots, a_k$ being distinct currencies, such that

$$\prod_{i=1}^{k} M_{a_i, a_{i+1}} \geq v.$$

In the Arbitrage problem, you are given some matrix $M$ and a positive rational number $v$, and are asked to determine whether there is an arbitrage of value $v$. In the real world, if there exists an arbitrage value greater than 1, then some sequence of currency exchanges allows one to make essentially risk-free money.

(a) Show that Arbitrage can be solved in polynomial-time if $M_{i,j} \leq 1$ for all currencies $i, j$.

(b) Show that Arbitrage is $NP$-complete. (Hint: Reduce from HamPath, which is shown to be $NP$-complete in Sipser.)