The Complexity Class NP

1 Certificates

Every yes-instance of an NP problem has a short and easily checkable certificate, for example an assignment for satisfiability

Definition: Certificate

A potential solution, it may be correct or incorrect

2 Verifiers

Definition: Verifier

An acceptor machine (FSM with accepting states) V which halts on all inputs is called a verifier for a language $\mathcal{L}$ if

$$\mathcal{L} = \{w | V \text{accepts "w; c" for some string c}\}$$

- w;c just means a problem certificate pair
- The string c is called a certificate (or witness) for w
- A verifier is said to be polynomial-time if it is a polynomial-time TM, and there is a polynomial $p(x)$ such that, for any $w \in \mathcal{L}$, there is a certificate c with $|c| \leqslant p(|w|)$

3 The class NP

Definition: NP

The class of languages that have polynomial-time verifiers is called NP

Problem: Composite Number

Instance - A positive integer k
Question - Are there integers $u, v > 1$ such that $u \cdot v = k$

Problem: Subset Sum

Instance - A collection of positive integers $S = \{a_1, ..., a_k\}$ and a target integer t
Question - Is there a subset $T \subseteq S$ such that $\sum_{i \in T} a_i = t$

4 Problems (probably) not in NP

Problem: No Hamiltonian Cycle

Instance - A graph G
Question - Is it true that G has no Hamiltonian cycle?

Problem: Checkers

Instance - An integer n and a position in checkers on $n \times n$ board
Question - Is it a winning position for white?

5 Nondeterministic Machines

We can get an alternative definition of the class NP by considering non-deterministic machines.

Recall that if NT is a non-deterministic Turing Machine, then NT(x) denotes the tree of configurations which can be entered with input x, and NT accepts x if there is some accepting path in NT(x).

Definition: Time Complexity

The time complexity of a non-deterministic Turing Machine NT is the function $NTime_{NT}$ such that $NTime_{NT}(x)$ is the number of steps in the shortest accepting path $NT(x)$ is there is one, otherwise it is the number of steps in the shortest rejecting path

6 Non-Deterministic time complexity

Definition: Non-deterministic time complexity

For any function f, we say that the non-deterministic time complexity of a decidable language $\mathcal{L}$ is $O(f)$ is there exists a non-deterministic TM NT which decides $\mathcal{L}$, and constants n_0, and c such that for all inputs x with $|x| > n_0$

$$NTime_{NT}(x) \leqslant c \cdot f(|x|)$$

Definition: Non-deterministic time complexity class

The non-deterministic time complexity class NTIME[f] is defined to be the class of all problems (i.e. languages), for which there exists an algorithm with non-deterministic time complexity in $O(f)$.

7 Alternative NP Definition

$$\mathrm{NP} = \bigcup_{k \geq 0} NTME\left[n^k\right]$$

Proof:

- If $\mathcal{L} \in NTIME[n^k]$, then there is a non-deterministic machine NT such that $x \in \mathcal{L}$ iff there is an accepting computation path in $NT(x)$. Furthermore, the length of these paths is $O(|x|^k)$
- Using (some encoding of) these computation paths as the certificates, we can construct a polynomial time verifier for $\mathcal{L}$ which simply checks that each step of the computation path is valid
- Conversely, if $\mathcal{L}$ has a polynomial-time verifier V, then we can construct a non-deterministic machine that first "guesses" the value of the certificate, and then simulates V with that certificate
- Since the length of this certificate is polynomial in the length of the input, this machine is a non-deterministic-polynomial-time decision procedure for $\mathcal{L}$

8 Complete Problems

- Any complexity class can be partitioned into equivalence classes via polynomial time reduction - each class contains problems that are reducible to each other
- These equivalence classes are partially ordered by reduction
- Problems in the maximal class are called complete

9 NP-completeness

- To show that $\mathcal{L}$ is NP-complete we must show that every language in NP can be reduced to $\mathcal{L}$ in polynomial time
- However once we have one NP-complete language $\mathcal{L}_0$, we can show any other language $\mathcal{L}$ is NP complete by showing that $\mathcal{L}_0 \leqslant \mathcal{L}$

10 Ladner's theorem

Theorem 1 *If $P \neq NP$ then NP contains infinitely many (polynomial time) inequivalent problems*

- This implies that unless P=NP, the class NP contains (infinitely many) problems that are neither in P nor NP-complete. Such problems are called NP-intermediate

11 Linear Programming

Problem: Linear Programming

Instance: Integer vectors $V_i = (v_1^i, ..., v_n^i), 1 \leqslant i \leqslant m, D = (d_1, ..., d_n), C = (c_1, ..., c_n)$ and an integer B
Question: Is there a rational vector $X = (x_1, ..., x_n)$ such that $V_i \cdot X \leqslant d_i$ for all $1 \leqslant i \leqslant m$ and such that $C \cdot X \geqslant B$

- This is in P, but the same problems where X is required to be an integer vector is NP-complete

12 Primes/composite

Problem: Composite

Instance: Positive integer K
Question: Is K composite?

- Recently proven that it is in P

13 Graph Isomorphism

Problem: Graph Isomorphism

Instance: Two undirected graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$
Question: Are G and H isomorphic, i.e., is there a bijection $f : V_G \rightarrow V_H$ such that $(u, v) \in E_G$ iff $(f(u), f(g)) \in E_H$?

- Currently the main candidate for NP-intermediate