# 1 Complexity Theory Review

## 1.1 Complexity Classes

The class **P** (Polynomial) is the class of decision problems that can be decided in polynomial time. The running-time must be $O(n^c)$ for some constant c.

**Super-polynomial** time describes functions that grow at a rate faster than any polynomial, but are not necessarily exponential. For example $O(n^{\log n})$ is a super-polynomial running time.

## 1.2 NP Completeness

The class **NP** (Nondeterministic Polynomial) is the class of all decision problems that:

- Given a witness (certificate) $\omega$, can verify, that $w \in L$ in polynomial time.
- Can be decided by a *Non-deterministic turing machine* in polynomial time

These two characterstics are equivalent.

## 1.3 Decision vs Solution

Given a decider for any problem in **NP**, we can determine a satisfying solution to the problem. For example, for the question *3-color*, which asks if there is a valid 3-coloring for a graph $G$, you can produce a valid 3-coloring for the graph with polynomially many calls to the decider.

The strategy for doing this is as follows. For any problem in NP, you can reduce the problem to circuit satisfiability, meaning that any problem can be reduced to the problem of asking if there is an input to a specific circuit that causes the circuit to output 1. If there exists a solution, for each input bit $x_1, ..., x_n$, you hardcode $x_i$ to 0 from 1 to $n$. If the decider outputs yes, there exists a valid input with $x_i$ as 0, else, a valid input with $x_i$ as 1 exists. Replace $x_i$ with the correct input value and repeat for all $i$.

### 1.4 Reductions

Many cryptographic protocols rely on assumptions about the difficulty of specific problems as the backbone of the security of the protocol. They are designed in a manner such that any adversary that could break the security of the protocol could break the *difficulty* of the problem. The **Diffie-Hellman** public key exchange assumes that the discrete logarithm problem is impossible to solve efficiently, and uses that to create a protocol that is secure under the *assumption* that this is the case.

The protocol is as follows:

1. Two parties $A$ and $B$ agree a prime order group $\mathbb{Z}_p$ and generator $g$. (Important to note that for prime order groups, any non-identity member can act as a generator for the group)

2. $A$ and $B$ select exponents $(x,\, y) \xleftarrow{\$} \mathbb{Z}_p$ secretly, meaning only $A$ knows $x$ and only $B$ knows $y$.

3. $A$ sends $g^x$ to $B$ and $B$ sends $g^y$ to $A$. They both produce secret key $k$ by calculateing $(g^x)^y = (g^y)^x$.

It is important to note that everything other than $x$ and $y$ are public to any potential adversary, including the sent messages.

## 2 Interactive Proofs

### 2.1 Graph Isomorphism

Proofs allowing for interaction between two parties that include a verifier (V) and a prover (P) provide a much stronger notion of verification than classical static proofs.

The problem of determining if two graphs are isomorphic is in NP. Given $\pi$ a permutation that labels nodes in $G_0$ to nodes of $G_1$, a verifier can easily determine if the two graphs are in fact isomorphic. On the otherhand, determining whether or now two graphs are not isomorphic is more difficult. The decision problem for this can be described as given two graphs $(G_0, G_1)$, determine if they are not isomorphic.

The protocol for this is as follows:

1. V (verifier) flips a coin and selects a bit $b = 0$ or 1 depending on the outcome. The verifier chooses $G_b$ and creates a random permutation $\pi$ that relabels the nodes of $G_b$.

2. The verifier sends $H = \pi(G_b)$ to the prover ($P$).

3. $P$ has to determine $b$ based on $H$ and sends its guess $b'$ back to $V$. (Note: We assume $P$ has a decider that is able to determine if two graphs are isomorphic. With this, it can test $H$ against both $G_0$ and $G_1$ and determine $b$ if the two graphs are non-isomorphic).

4. This process is repeated until the verifier is confident that the two graphs are non-isomorphic.

## 2.2 Properties of Interactive Proofs

There are two key properties of all interactive proofs:

- Completeness: $\forall x \in L, P[$ P convinces V $] > 2/3$. That is, for any $x$ in the language, the probability that $P$ convinces the verifier that $x$ is a part of the language must be greater that $2/3$.

- Soundness: $\forall x \in L, P[\ P(x)$ convinces $V\ ] < 1/3$. Or the prover can not convince the verifier less than a third of the time

It is important to note that both parties must agree on the protocol beforehand. The actual protocol must prove that the verifier will accept the majority of instances *only* if $x \in L$. Because of this, the bounds are arbitrary since the protocol can be repeated many times, and the verifier can decide whether to accept or not based on the majority.