Reference for Assignment 4

problem 4.1

Let $p(n) = \frac{1}{f(n)}$, where $f(n)$ is a non-polynomial function, and $\forall b > 0. f(n) = O(b^n)$, i.e., $f(n)$ is a function that grows between polynomial (exclusive) and exponential functions (exclusive). For example, taking $f(n) = 2^{\sqrt{n}}$ corresponds to $p(n) = 2^{-\sqrt{n}}$.

Some common incorrect answers include $p(n) = \frac{1}{f(n)}$ where $f(n)$ is a polynomial, which does not satisfy the negligibility condition; or $p(n) = \frac{1}{f(n)}$ where $f(n)$ is an exponential function, which does not satisfy $\frac{1}{b^n} = O(p(n))$.

problem 4.2

Let (P, V) be the interactive proof system described in the problem. We construct another proof system (P', V'): for the proposition S, repeat (P, V) for n times, and if V outputs "accept" at least once, V' outputs "accept"; otherwise, it outputs "reject". If S is true, the probability that V' outputs "accept" is at least $1 - (1 - \frac{2}{3})^n = 1 - (\frac{1}{3})^n$, where $(\frac{1}{3})^n$ is negligible. Therefore, if S is true, V' outputs "accept" with high probability, satisfying the definition of completeness in Lecture 4.2.

problem 4.3

Completeness: If P knows the homomorphic mapping θ, then the probability that V accepts is 1, which is high. Therefore, it has completeness.

Soundness: If P does not know the homomorphic mapping θ, then the probability that it answers correctly in each round is at most $\frac{1}{2}$, and after n rounds, the probability that V accepts is at most $(\frac{1}{2})^n$, which is negligible. Therefore, it has soundness.

PZK: Construct a simulator σ that generates a random bit $b' \xleftarrow{\$} \{0, 1\}$ and a random homomorphic mapping θ' in each round. If $b' = 0$, it sends $G'' = \theta(G)$ to V^*; otherwise, it sends $G'' = \theta^{-1}(G')$ to V^*. If $b' = 0$ and σ simulates V^* asking for the homomorphic mapping between G and G'', or if $b' = 1$ and σ simulates V^* asking for the homomorphic mapping between G' and G'', it returns θ'. Otherwise, the process is repeated. This simulator is probabilistic polynomial-time and can successfully simulate with high probability, producing an output with the same distribution as the View of V^*. Therefore, the interactive proof system is a PZK.

problem 4.4

Solution: (Problem 4.4) Let n be the length (*i.e.*, the number of bits) of the description of an isomorphism. We construct a pseudo-ZK proof system (P, V) for proving G is isomorphic to G':

- The verifier V picks x from n-bit strings uniformly at random, and sends x to the prover P.
- P computes $\theta' = \theta \oplus x$ to V, where θ is the isomorphism from G to G'.
- P sends θ' to V.
- V checks that $\theta' \oplus x$ is an isomorphism from G to G'. If that's the case, then V accepts; otherwise, V rejects.

Clearly, the proof is not PZK, because in fact, it completely reveals θ to V. However, it meets the requirement of pseudo-ZKness, because the only message received by V, θ', is uniformly distributed over n-bit strings. The simulator has no difficulty in generating a random bit string that follows exactly the same distribution. $\square$

problem 4.5

Construct a simulator σ that generates a random bit $b' \xleftarrow{\$} \{0, 1\}$ and an integer $f \xleftarrow{\$} \{1, 2, \dots, q-1\}$. If $b' = 0$, it sends $j = g^f\ (mod\ p)$ to V^*; otherwise, it sends $j = h^{-1}\backslash * g^f\ (mod\ p)$ to $V^{\backslash *}$. Then, if σ simulates V^*'s output b, if $b = b'$, σ outputs f; otherwise, the process is repeated. This simulator is probabilistic polynomial-time and can successfully simulate with high probability, producing an output that is computationally indistinguishable from the View of V^*. Therefore, the interactive proof system is a CZK.

problem 4.6

Solution: (Problem 4.6) Suppose the input is (G, G') such that $\theta(G) = G'$. At the very beginning, the prover P flips n coins, where n is the security parameter. If all coins are 1, then P sends θ to the verifier V. Otherwise, P starts the execution of an existing PZK proof for graph isomorphism.

(P, V) is SZK, because we can build a SZK simulator based on the simulator for the PZK proof, as described below. If the PZK simulator fails, our SZK simulator simply outputs a random bit string. Notice that the output our SZK simulator differs from the view of the possibly cheating verifier $V^\star$ only if the SZK proof fails, or the initial n coin flips are all 1s. The total probability of these two events is negligible. Therefore, (P, V) is SZK.

(P, V) is not PZK, because it leaks information about θ with probability 2^{-n}. (Some people might argue that you can construct a simulator to show it's PZK, because your simulator could fails exactly when θ is leaked. We emphasize that they are completely wrong, and such a simulator does not exist. The reason is that the simulator has no access to P's coin flips, and thus it does not know when it should fail.) $\square$

problem 4.7

Yes. We note the definition using "distinguisher" as Definition 1 and the one using "algorithm" as Definition 2.

If two random variables are computationally distinguishable by Definition 2, then there exists a algorithm A s.t. $\sum_v |Pr[A(X) = 1] - Pr[A(Y) = 1]|$ is not negligible.

Let $S = \{v : Pr[A(X) = v] > Pr[A(Y) = v]\}$. We construct a distinguisher D as follows:

D simulates A. Then outputs 1 if $A(x) \in S$ and outputs 0 otherwise. Obviously, $Pr[D(X) = 1] = Pr[A(X) \in S]$. Then by definition 2,

$$
\begin{aligned}
&\sum_v Pr[A(X) = v] - Pr[A(Y) = v] \\
=&\sum_{v \in S} Pr[A(X) = v] - Pr[A(Y) = v] + \sum_{v \notin S} Pr[A(X) = v] - Pr[A(Y) = v] \\
=&Pr[A(X) \in S] - Pr[A(Y) \in S] + Pr[A(X) \notin S] - Pr[A(Y) \notin S] \\
=&2(Pr[A(X) \in S] - Pr[A(Y) \in S]) \\
>&\frac{1}{p(n)}.
\end{aligned}
$$

So $|Pr[D(X) = 1] - Pr[D(Y) = 1]| > \frac{1}{p(n)}$, and X and Y are computationally distinguishable by Definition 1.

problem 4.8

(a)False,

Note that the distributions of x and u, and y and v are not necessarily independent. For example:

Distribution of x with respect to u:

	0_s	1_s
0_s	0.5	0
1_s	0	0.5

Distribution of y with respect to v:

	0_s	1_s
0_s	0	0.5
1_s	0.5	0

(b) True, proof omitted.