1. Show that there exists an efficient many-one reductions between VERTEX-COVER and SETCOVER problems. Estimate the corresponding efficiency guarantees. Formal definitions of both problems are following.

- **VERTEX-COVER Problem.** Given a graph, find a minimal set of vertices $W$ that covers all edges, i.e., at least one endpoint of every edge belongs to $W$.
- **SETCOVER Problem.** Given a universe of sets $U = \{S_1, \ldots, S_n\}$, find a minimal set of sets $\mathcal{C} \subseteq U$ that $\mathcal{C}$ contains all elements of $U$, i.e., for any element $u \in \bigcup_{i=1}^{n} S_i$ there exists $S_i \in \mathcal{C}$ that contains $u$.

**Hint:** How to encode Boolean variables and NAND gates?

2. Describe a natural but inefficient black-box reduction between HAMILTONIAN-CYCLE and TRAVELLING-SALESMAN problem. Establish the bounds on reduction efficiency. Formal definitions of both problems are following.

- **HAMILTONIAN-CYCLE Problem.** Given a graph, output 1 iff the graph contains a cycle that goes though each vertex exactly once.
- **TRAVELLING-SALESMAN Problem.** Given a graph with weighted edges, find and Hamiltonian cycle with the smallest summary weight.

**Hint:** How can you enumerate all Hamiltonian cycles?

(*) Provide an efficient black-box reduction or show that no such black-box reductions can exist.

3. Let $G$ be a finite group such that all elements $y \in G$ can be expressed as powers of $g \in G$. Then the discrete logarithm problem is following. Given $y \in G$, find a smallest integer $x$ such that $g^x = y$ in finite group $G$. Discrete logarithm problem is known to be hard in general, i.e., all universal algorithms for computing logarithm run in time $\Omega(\sqrt{|G|})$.

(a) Show that for a fixed group $G$, there exists a Turing machine that finds the discrete logarithm for every $y \in G$ in $O(\log^2 |G|)$ steps.
(b) Show that for a fixed group $G$, there exists an analogous Random Access Machine that achieves the same efficiency.
(c) Generalise the previous construction and show that for every fixed function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ there exists a Turing machine and a Random Access Machine such that they compute $f(x)$ for every input $x \in \{0,1\}^n$ in $O(n + m)$ steps.
(d) Are these constructions also valid in practise? Explain why these inconsistencies disappear when we formalise algorithms through universal computing devices.
Hint: What is the time-complexity of binary search algorithms?

4. Let $G$ be a finite group such that all elements $y \in G$ can be expressed as powers of $g \in G$. Then the computational Diffie-Hellman (CDH) problem is following. Given $x = g^a$ and $y = g^b$, find a group element $z = g^{ab}$.

(a) Show that computational Diffie-Hellman problem is random self-reducible, i.e., for any algorithm $B$ that achieves advantage $\text{Adv}^{\text{cdh}}_G(B) = \Pr \left[ x, y \leftarrow U_G : B(x, y) = g^{\log_g x \log_g y} \right]$ there exists an oracle algorithm $A^B$ that for any input $x, y \in G$ outputs the correct answer with the probability $\text{Adv}^{\text{cdh}}_G(B)$ and has roughly the same running time.

(b) Given that the CDH problem is random self-reducible, show that the difficulty of CDH instances cannot vary a lot. Namely, let $B$ be a $t$-time algorithm that achieves maximal advantage $\text{Adv}^{\text{cdh}}_G(B)$. What can we say about worst-case advantage

$$\min_{x,y} \Pr \left[ A(x,y) = g^{\log_g x \log_g y} \right] ?$$

Can there be a large number of pairs $(x,y)$ for which the CDH problem is easy?

(c) Provide a black-box reduction between DL and CDH problem.

(d) Show that if there exists an efficient procedure that can always compute the highest bit of $\log_g y$ then the DL problem is easy.

5. Password checks in TENEX system was implemented as follows

```c
int is_correct_password(char passwd[8])
{
    int i;
    for(i=0;i<8;++i)
    {
        if(passwd[i]!=true_passwd[i]) return 0;
    }
    return 1;
}
```

This implementation vulnerable against side-channel attacks. In brief, users could detect the last value of $i$ before function returned value by allocating a right amount of memory before login attempts and by observing page faults. Let $B(x)$ be the corresponding attack strategy, which returns the last value of $i$. Construct an efficient black-box reduction $A^B$ that recovers the correct password.
6. Consider a classical Turing machine without internal working tapes, i.e., the Turing machine has a single one-sided (input) tape that initially contains inputs and must contain the desired output after the execution.

(a) Show that all sorting algorithms take at least $\Omega(n^2)$ steps where $n$ is the total length of inputs $x_1, \ldots, x_k$. What is the time-complexity of best sorting algorithms? Explain this contradiction.

(b) Does the minimal time-complexity change if the Turing machine has internal working tapes?

(c) Sketch how one can simulate execution of Random Access Machines on a Turing machine. What is the corresponding overhead?

(*) Construct a set of tasks that can be implemented significantly more efficiently on Turing machines with $\ell+1$ working tapes than on Turing machines with $\ell$ tapes.

**Hint:** It is well-known fact that reversing $n$-bit string takes $\Omega(n^2)$ steps on a Turing machine without working tapes.