SPCS CRYPTOGRAPHY
HOMEWORK 6

Please try all the unmarked problems. #, ⋆ problems are both optional, with ⋆ problems being harder. You are strongly encouraged to work in groups, but you have to write up the solution on your own.

Reference for today's lecture: Chapter 7, Chapter 9.4, 9.5, Chapter 10.1 - 10.3

1. Suppose that in Diffie-Hellman key exchange, instead of choosing a primitive root $g \bmod p$, we just pick a random element $g \not\equiv 0 \bmod p$, and proceed as before. What properties should g have so that the system is more secure?

2. Suppose that Eve has an oracle to solve the discrete logarithm problem, i.e., if she tells it g and m, then the oracle tells her an x so that

$$g^x \equiv m \bmod p$$

How can she use this oracle to break ElGamal encryption quickly?

3. Determine if the following statements are right, and explain why. You can assume that all functions involved are positive.
   (a) $n^2 = O(n^3)$.
   (b) $(\log n)^2 = O(n)$.
   (c) If $f(n) = O(g(n))$, then $g(n) = O(f(n))$.
   (d) If $f(n) = O(g(n))$, then $e^{f(n)} = O(e^{g(n)})$.
   (e) If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$, then $(f_1 \cdot f_2)(n) = O((g_1 \cdot g_2)(n))$.

4. Assume that each of the expressions below gives the processing time $T(n)$ spent by an algorithm for solving a problem of size n. Select the dominant term(s) in n and specify the lowest Big O complexity of each algorithm.
   (a) $5 + 4n + 6n^3$.
   (b) $6 + n^{1.5} + 8n^3$.
   (c) $n \log\log n + n \log n$.
   (d) $n(\log n)^2 + n \log n$.

# 5. (For those with programming background) The insertion sort works as follows,

```
for $j \leftarrow 2$ to length[A] do
    key $\leftarrow A[j]$
    $i \leftarrow j - 1$
    while $i > 0$ and $A[i] > key$ do
        $A[i+1] \leftarrow A[i]$
        $i \leftarrow i - 1$
    end while$A[i+1] \leftarrow$ key
end for
```

Explain in words how insertion sort works. Analyze the number of comparisons you need for insertion sort, in terms of the input (length of array A). You can use big O notation.

6. Suppose we have a list of n sorted numbers. Devise an algorithm to look for a particular number on the list, with $O(\log n)$ comparisons.

7. There are n points on the plane with given coordinates. We want to find the closest pair of points.
   (a) The naive algorithm computes the distance between any pair of points, and find the minimum. What is the running time complexity, if calculation of distance between one pair of points takes one unit of time?
   * (b) Can you do better?

8. Here is an algorithm to compute $g^k \bmod n$.
   - Set $a = g$, and compute $b = g^2 \bmod n$.
   - Write k in its base 2 expansion $(x_l \cdots x_0)_2$ with $x_l = 1$, and run through $x_0, \cdots, x_{l-1}$.
     - If $x_i = 0$, set $a = a^2 \bmod n$, $b = ab \bmod n$.
     - If $x_i = 1$, set $a = a \cdot b \bmod n$, $b = b^2 \bmod n$.
   - Output a.

   (a) Explain why this works, i.e. the final output is indeed $g^k \bmod n$.
   (b) What is the difference between this algorithm and the repeated squaring algorithm we discussed in class? More specifically, what is the computational cost of each step involved about x_i?