Homework 1

Asymptotics and Number Theoretic Algorithms

Part A (20 points)

Problem 1: In each of the following situations indicate whether $f = O(g)$ or $f = \Omega(g)$ or $f = \Theta(g)$:

1. $f(n) = \sqrt{2^{7n}},\ g(n) = \lg(7^{2n})$

   $f(n) = \sqrt{2^{7n}} = 2^{7n/2}$ and $g(n) = 2nlg(7)$. $f(n)$ is an exponential function, and $g(n)$ is a linear function. Because an exponential grows faster than a linear function, $f(n) = \Omega(g)$.

2. $f(n) = 2^{nln(n)},\ g(n) = n!$

   Since $g(n) = n!$, we can use Stirling's approximation: $ln(n!) = nln(n) - n + O(lnn)$. To get it in the same form as Stirling's we take ln of $f(n)$ and ln of $g(n)$ Stirling's approximation states that $ln(n!) = nln(n) - n + O(lnn)$. Taking the natural logarithm of $f(n)$, we have

$$\begin{aligned} ln(f(n)) &= ln(2^{nln(n)}) \\ &= ln(2)nln(n) \\ &= O(nlogn) \end{aligned}$$

   Thus, $f(n) = \Theta(g(n))$.

3. $f(n) = lg(lg^* n),\ g(n) = lg^*(lgn)$

4. $f(n) = \frac{lgn^2}{n},\ g(n) = lg^* n$

5. $f(n) = 2^n,\ g(n) = n^{lgn}$

6. $f(n) = 2^{\sqrt{lgn}},\ g(n) = n(lgn)^3$

7. $f(n) = e^{cos(n)},\ g(n) = lgn$

8. $f(n) = lgn^2,\ g(n) = (lgn)^2$

9. $f(n) = \sqrt{4n^2 - 12n + 9},\ g(n) = n^{\frac{3}{2}}$

10. $f(n) = \sum_{k=1}^{n} k, g(n) = (n+2)^2$

*N*ote: The $lg^* n$ function is the number of times the logarithm function must be iteratively applied before the result is less than or equal to 1, i.e., $lg^* n = 0$ if $n \leq 1$ and $lg^* n = 1 + lg^(lgn)$ if $n > 1$. For instance, $lg^* 4 = 2$, $lg^* 16 = 3$, $lg^* 65536 = 4$, etc.

Problem 2: Compute the asymptotic running time of the above algorithm as a function of its input parameter, given:

- The running times of integer arithmetic operations (e.g., multiplication of two large n-bit numbers is $O(n^2)$).
- Assume that sampling a number N is an operation linear to the number of bits needed to represent this number.

Do not just present the final result. For each line of pseudo-code indicate the best running time for the corresponding operation given current knowledge from lectures and recitations and then show how the overall running time emerges.

Part B (30 points)

Algorithm 1: `Number_Theoretic_Algorithm` (integer n)

$N \leftarrow Random\_Sample(0, 2^n - 1)$;
if *N is even* **then**
    $N \leftarrow N + 1$;
$m \leftarrow N \mod n$;
for $j \leftarrow 0$ *to* m **do**
    if `Greatest_Common_Divisor`$(j, N) \neq 1$ **then**
        return FALSE;
    Compute x, z so that $N - 1 = 2^z \cdot x$ and x is odd;
    $y_0 \leftarrow (N - 1 - j)^x \mod N$;
    for $i \leftarrow 1$ *to* m **do**
        $y_i \leftarrow y_{i-1}^2 \mod N$;
        $y_i \leftarrow y_i + y_{i-1} \mod N$;
    if `Low_Error_Primality_Test`(y_m) == FALSE **then**
        return FALSE;
return TRUE;

Problem 3:

- Consider that we have a tree data structure T_m^N, where every node can have at most m children and the tree has at most N nodes total. Compute a lower bound for the height of the tree.

- Consider two such trees T_m^N and $T_{m'}^N$ that are "perfect", i.e., every node has exactly m and m' children correspondingly. Now, consider the functions $h_m(N)$ and $h_{m'}(N)$ that express the heights of these perfect trees for different values of N. What is the asymptotic behavior of h_m relative to $h_{m'}$ and under what conditions?

- Consider the following rule for modular exponentiation, where x is in the order of 2^m and y is in the order of 2^n. What is the running time of computing the result according to this rule?

$$x^y = \begin{cases} (x^{\lfloor \frac{y}{2} \rfloor})^2, & \text{if y is even} \\ x \cdot (x^{\lfloor \frac{y}{2} \rfloor})^2, & \text{if y is odd} \end{cases}$$

Problem 4:

- Compute the following: 2^{902} mod 7.

- Find the modulo multiplicative inverse of 11 mod 120, 13 mod 45, 35 mod 77, 9 mod 11, 11 mod 1111.

- Assume that for a number x the following property is true: $\forall\ y \in [1, x-1] :\ gcd(x, y) = 1$. Compute the running time of an efficient algorithm for finding all the inverses modulo x^m from the set $\{0, 1, \ldots, x^m - 1\}$ that exist.

Problem 5:

- Assume two positive integers x < y. Then the pairs $(5x + 3y, 3x + 2y)$ and (x, y) have the same greater common divisor. True or False, explain.

- Consider the following sequence of numbers: $s_n = 1 + \Pi_{i=0}^{n-1} s_i$, where $s_0 = 2$. Prove that any two numbers in this sequence are relatively prime.

Part C (30 points)

Problem 6: Our goal is to assign $(2^{31} - 1)$ integers into 256 slots $\{0, 1, \ldots, 255\}$ and achieving the properties of universal hashing. Notice that 256 is not a prime number. Assume the following approach towards this objective.

Consider a 32-bit integer y and the following set $\mathcal{M}$ of hash functions, so that:

- each $m \in \mathcal{M}$ corresponds to a unique 8×32 matrix M having elements only 0 or 1.
- and $m(y) = M \cdot y \bmod 2$, i.e., multiply the matrix with the 32-bit vector corresponding to number y and then apply the modulo operation on each bit of the resulting vector.

Such functions map a 32-bit vector into an 8-bit vector, which can then be interpreted as an 8-bit number that indicates the original number's slot in the range $0 \sim 255$. Notice that there are $2^{8 \times 32} = 2^{256}$ different $\{0, 1\}$ matrices in the set $\mathcal{H}$.

Provide the following:

- A proof that the hash function family $\mathcal{M}$ is universal.
- A comparison to the universal hash function family described in DPV chapter 1.5.2. How many random bits are needed here?

Problem 7: Answer the following sequence of problems:

- Assume that number n is prime, then all numbers $1 \le x < n$ are invertible modulo n. Which of these numbers are their own inverse module n?
- Show that $(n-1)! \equiv -1 \mod n)$ for prime n. [Hint: Compute the value of $(n-1) \mod n$ by considering how it arises by pairing two numbers smaller than n that are multiplicative inverses modulo n.]
- Show that if n is not prime, then $(n-1)! \not\equiv -1 \mod n$. [Hint: What does it mean that n is not prime in terms of the numbers $1 \le x < n$?]
- The above process can be used as a primality test instead of Fermat's Little theorem as it is an if-and-only-if condition for primality. Why can't we immediately base a primality test on this rule? [Tip: Even if you are not able to answer the previous two questions, you should be able to argue about this question.]

Part D (30 points)

Problem 8:
A. Make a table with three columns. The first column is all numbers from 0 to 36. The second is the residues of these numbers modulo 5; the third column is the residues modulo 7.

B. Consider two different prime numbers x and y. Show that the following is true: for every pair of numbers m and n so that: $0 \le m < x$ and $0 \le n < y$, there is a unique integer q, where $0 \le q < xy$, so that:

$$q \equiv m \pmod{x}$$
$$q \equiv n \pmod{y}$$

[Hint: Think how many q's in the range $[0, xy]$ can have the same result modulo x and modulo y and count how many q's there are.]

C. The previous problem asks to go from q to (m, n). It is also possible to go the other way. In particular, show the following:

$$q = (m \cdot y \cdot (y^{-1} \bmod x) + n \cdot x \cdot (x^{-1} \bmod y)) \bmod xy$$

[Hint: Ensure that if the above is true then the expressions in section B are also true. Consider the values of the following terms: $c_x = y \cdot (y^{-1} \bmod x)$ and $c_y = x \cdot (x^{-1} \bmod y)$ and their values $\bmod x$ and $\bmod y$.]

D. What happens in the case of three primes x, y and z? Do the above properties still hold? If they do, how do they look like in this case?

Problem 9: There is an office, in which every member uses RSA to conduct secure communication with others. For example, when Alice wants to send Bob a message, she will first use Bob's public key to encrypt the message, then sends it to Bob, Bob then uses his private key to decrypt the message. In order to make things easy, the office maintains a directory listing every member's public key, everybody in the office has access to it. A public key looks like (N, e) (as defined in DPV-1.4.2).
One day, Alice sent a message (smaller than any N) to Bob, Charlie, and David via the RSA based secure communication, but the encrypted messages were intercepted by the webmaster Mallory. Mallory used some network tricks and found out who the receivers were, then he checked their public keys in the directory. This is what Mallory got:

Receiver	Encrypted message	Public key
Bob	153	(155, 3)
Charlie	196	(203, 3)
David	27	(117, 3)

Finally, Mallory recovered the original message. How did Mallory do this? What is the original message?

[Hint: Consider how the three transmitted messages are computed. The result of the previous problem 8 is useful here. Be careful with the details and under which conditions you apply certain expressions.]