CS5800: Algorithms SEC 05 Summer Full 2024 (Seattle)
Homework 2

Payal Chavan

05/22/2024

Question 1: Find the inverse of: 20 mod 79, 3 mod 62, 21 mod 91, 5 mod 23.

To find the inverse of a number a modulo n (i.e., a mod n), we need to identify a number x such that $a \times x$ is congruent to 1 modulo n (i.e., $a \times x \equiv 1 \mod n$), this can be accomplished by using the Extended Euclidean Algorithm.

This algorithm allows us to find the modular multiplicative inverse efficiently. The Extended Euclidean Algorithm is particularly useful when a and b are co-prime (i.e., their gcd is 1). Under that condition, x is defined as the modular multiplicative inverse of a modulo b.

Part (a): 20 mod 79

Solution:
Let us consider 'x' as the multiplicative inverse of mod 79 of 20.

$$\begin{aligned} & 20 \times x \equiv 1(\mod 79) \\ \implies & 20 \times x = 79N + 1 && \text{(For some integer N)} \\ \implies & 20 \times x = 79(1) + 1 && \text{(Let } N = 1\text{)} \\ \implies & 20 \times x = 80 \\ \implies & x = 4 \end{aligned}$$

Therefore, inverse of 20 mod 79 is 4.

Part (b): 3 mod 62

Solution:
Let us consider 'x' as the multiplicative inverse of mod 62 of 3.

$$\begin{aligned} & 3 \times x \equiv 1(\mod 62) \\ \implies & 3 \times x = 62N + 1 && \text{(For some integer N)} \\ \implies & 3 \times x = 62(1) + 1 && \text{(Let } N = 1\text{)} \\ \implies & 3 \times x = 63 \\ \implies & x = 21 \end{aligned}$$

Therefore, inverse of 3 mod 62 is 21.

Part (c): 21 mod 91

Solution:
$gcd(21, 91) = 7 \neq 1$

So, in this case we do not have any multiplicative inverse for 21 mod 91.

Part (d): 5 mod 23

Solution:
Let us consider 'x' as the multiplicative inverse of mod 23 of 5.

$$5 \times x \equiv 1(\mod 23)$$

$\implies 5 \times x = 23N + 1$ (For some integer N)
$\implies 5 \times x = 23(3) + 1$ (Let $N = 3$)
$\implies 5 \times x = 70$
$\implies x = 14$
Therefore, inverse of 5 mod 23 is 14.

Question 2: If p is prime, how many elements of $0, 1, \ldots, p^n - 1$ have an inverse modulo p^n?

Solution: When p is a prime number, the number of elements in the set $0, 1, \ldots, p^n - 1$ that have an inverse modulo can be determined as follows:

1. For a number(a) to have an inverse modulo $(m = p^n)$, it must satisfy the condition that the number(a) and $(m = p^n)$ are relatively prime (i.e., their greatest common divisor is 1). In other words, if $(\gcd(a, p^n) = 1)$, then (a) has an inverse modulo (p^n).

2. Given that p is a prime number, the only divisors of p^n are powers of p itself. In other words, the powers of p are: $1, p, p^2, p^3, \ldots, p^n$. Now, if we want to find the modulo inverse of an integer a modulo p, (denoted as $a^{-1} \mod p$), we need to ensure that a is not divisible by p. Otherwise the modular division does not exist. Therefore, for a to have an inverse modulo p^n, a must not be divisible by p.

3. There are a total of p^n elements in the set $0, 1, p, p^2, p^3, \ldots, p^n$.

4. Out of the p^n elements in the set, the numbers that are divisible by p are p^{n-1} (as every p^{th} number is divisible by p).

5. Thus, the numbers in the set that are not divisible by p are: $p^n - p^{n-1} = p^{n-1}(p - 1)$. (and therefore have an inverse modulo p^n)

6. Hence, we have got $p^{n-1}(p - 1)$ elements in the set $0, 1, 2, \ldots, p^n$, which has an inverse modulo p^n. (where p is the prime number)

Question 3: Calculate $2^{125} \mod 127$ using any method you choose.

Solution:
Fermat's Little Theorem states that for prime p, we have $a^{p-1} \equiv 1 \mod p$.
Using Fermat's Little Theorem, we can rewrite,

$2^{125} = 2^{7*17+6} = (2^7)^{17} \times 2^6$

$2^{125} \mod 127 = (2^7)^{17} \times 2^6 \ (\mod 127)$

Using the modular arithmetic multiplication property, we have,
Since congruence modulo is an equivalence relation for (mod C).

If $A \equiv B(\mod C)$ $then, B \equiv A(\mod C)$

$\implies (2^7 \mod 127)^{17} \times 2^6 \ (\mod 127)$

$\implies (128 \mod 127)^{17} \times 2^6 \ (\mod 127)$

$$\implies (1)^{17} \times 2^6 \ \ (\mod 127)$$

$$\implies (2^6 \ \ (\mod 127)$$

$$\implies 64 \ \ (\mod 127)$$

Therefore, $2^{125} \mod 127 = 64$.

Question 4: Suppose you are choosing between the following three algorithms:

- **Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.**
- **Algorithm B solves problems of size n by recursively solving two subproblems of size n-1 and then combining the solutions in constant time.**
- **Algorithm C solves problems of size n by dividing them into nine subproblems of size n/3, recursively solving each subproblem, and then combining the solutions in $\mathcal{O}(n^2)$ time.**

What are the running times of each of these algorithms (in big-O notation), and which would you choose?

Solution: The Master Theorem is given as follows:
If T(n) = $aT([n/b]) + \mathcal{O}(n^d)$ for some constants $a > 0, b > 1$, and $d \geq 0$, then

$$\begin{cases} T(n) = \mathcal{O}(n^d) & \text{if } d > \log_b a \\ T(n) = \mathcal{O}(n^d \log n) & \text{if } d = \log_b a \\ T(n) = \mathcal{O}(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

1. Recurrence for Algorithm A: $\begin{cases} T(n) = 5T([n/2]) + n \\ T(1) = 1 \end{cases}$

   Here a = 5, b = 2, f(n) = $\Theta(n), d = \log_2 5, n^d = n^{\log_2 5} = n^{2.32}$

   We see that case 1 of the Master Theorem fits this problem as $d = 2.32 > 0$.

   Therefore, $T(n) = \Theta(n^{log_2 5})$.

2. Recurrence for Algorithm B: $\begin{cases} T(n) = 2T([n-1]) + c \\ T(1) = 1 \end{cases}$

   Note that, we cannot apply master theorem in this case because the size of the subproblems is not the fraction of the size of the input problem. In other terms, there is no constant (b) for which the size of the subproblems is (n/b). However, the subproblems doubles n times, and each subproblem takes constant time ($\mathcal{O}(1)$).

   Therefore, $T(n) = \Theta(2^n)$.

3. Recurrence for Algorithm C: $\begin{cases} T(n) = 9T([n/3]) + n^2 \\ T(1) = 1 \end{cases}$

   Here a = 9, b = 3, f(n) = $\Theta(n^2), d = \log_3 9, n^d = n^{\log_3 9} = n^2$

   The case 2 of the master theorem applies here because d = $\log_3 9 = 2$.

Therefore, $T(n) = \Theta(n^2 \log n)$.

Choice of Algorithm:

For Algorithm A, we have $T(n) = 5T(n/2) + n = \Theta(n^{\log_2 5})$

Algorithm A: $(\mathcal{O}(n))$ - breaks down the problem into five smaller subproblems, each half the size of the original problem. This approach leads to a logarithmic height in the recursion tree and allows for linear time when combining the solutions.

For Algorithm B, we have $T(n) = 2T(n-1) + 1 = \Theta(2^n)$

Algorithm B: $(\mathcal{O}(2^n))$ - recursively tackles two subproblems, each of size $n-1$. Unfortunately, this approach results in exponential time complexity due to the branching factor of 2.

For Algorithm C, we have $T(n) = 9T(n/3) + n^2 = \Theta(n^2 \log n)$

Algorithm C: $(\mathcal{O}(n^2))$ - By breaking down the problem into nine subproblems, each of size n/3, this approach results in a polynomial time complexity. The larger number of subproblems contributes to this complexity, along with the $\mathcal{O}(n^2)$ time required to combine their solutions.

Given that, Algorithm B has an exponential time complexity, it will be the slowest. Therefore, we will compare Algorithm A and Algorithm C.

$\mathcal{O}(n^{\log_2 5}) = \mathcal{O}(n^{2.32}) > \mathcal{O}(n^2 \log n)$
That's why I choose Algorithm C, because it grows asymptotically slow with a running time of $\mathcal{O}(n^2 \log n)$.

Question 5: Given the prime numbers p = 823 and q = 173, what are the two public keys (N, e) and the private key (d)? In addition, show the encrypted version of the message 42.

Solution: Given: $p = 823, q = 173, M = 42$

To Find: N, e, d

we have, $N = p \times q$

Therefore, $N = 823 \times 173 = 142379$

To compute d,
$\phi(N) = (p-1) \times (q-1)$

$\phi(N) = (823-1) \times (173-1) = 822 \times 172 = 141384$

Now, we need to choose a public key (e), such that it is relatively prime to $\phi(N)$, and also a smaller prime number.

Let's take $e = 5$

Now, to calculate private key (d), we can use the Extended Euclidean Algorithm.

$d \equiv e^{-1} \mod \phi(N)$

$d \equiv 5^{-1} \mod 142379$

$5d \equiv 1 \mod 142379$

$d = 28277$

Therefore, the private key $(d) = 28277$.

To encrypt the message $M = 42$,

We have encryption formula as follows: $c \equiv M^e \mod N$

$\implies$ Encrypted message $= 42^5 \mod 142379 = 129689$

Therefore, the public keys are $(N, e) = (142379, 5)$, the private key is $(d = 28277)$.
And the Encrypted version of the message 42 is 129689 .

Question 6: Use the divide and conquer integer multiplication algorithm to multiply the two binary integers 1011 and 0110. (Figure 2.1 in Dasgupta et. al.)

Solution: According to Divide and Conquer Algorithm for integer multiplication,

Split the binary numbers:
Given: $(x = 1011_2 = 11_{10})$ and $(y = 0110_2 = 6_{10})$

Divide each number into left and right halves:
$(x = x_L \cdot 2^2 + x_R)$, where $(x_L = 10) and (x_R = 11)$.
$(y = y_L \cdot 2^2 + y_R)$, where $(y_L = 01) and (y_R = 10)$.

Compute intermediate products:
say $(a = x_L + x_R = 101)$ (in decimal, 5).
say $(b = y_L + y_R = 11)$ (in decimal, 3).
Compute $P1 = (x_L \times y_L = 10)$ (in decimal, 2).
Compute $P2 = (x_R \times y_R = 110)$ (in decimal, 6).
Compute $P3 = (a \times b = 1111)$ (in decimal, 15).

$P1 \times 2^n + (P3 - P1 - P2) \times 2^{n/2} + P2$
$\implies 10 \times 2^4 + (1111 - 10 - 110) \times 2^{4/2} + 110$
$\implies 10 \times 10000 + 111 \times 100 + 110$
$\implies 100000 + 11100 + 110$
$\implies 100000 + 100010$
$\implies 1000010$ (in decimal, 66).

Question 7: From the Goddard text, section A3 problem 3. What is the big-Oh complexity of your algorithm?

Suppose we have a list of n numbers. The list is guaranteed to have a number which appears more than n/2 times on it. Devise a good algorithm to find the Majority element.

Solution: To solve this problem, we can consider using divide and conquer approach.
Our main goal is to find out if there is an element that appears more than n/2 times in an array. This problem can be approached by:

1. We will recursively divide the array into smaller subproblems, until we reach the base case.
2. The key finding is that an element must be a majority element in at least one of its halves if it is a majority element across the array.
3. For Base Case: We will return majority element, if the array has only one element. Otherwise, if the array is empty, we will return "No Majority Element".
4. To divide: Compute the mid index by mid = (l + r)/2, where l = left index and r = right index.
5. Determine the majority element in the left half recursively, from (l to mid) and the right half starting from (mid + 1 to r).
6. To combine: That element is the overall majority element if the majority elements of the left and right halves are equal. If not, in order to identify the overall majority element, we must compare the counts of the majority items in the two halves.
7. The big-O complexity of the Divide-and-Conquer Algorithm for finding the majority element is $\mathcal{O}(n \log n)$, because the array is split in half with each recursive call. This results in logarithmic time complexity. And we have $\mathcal{O}(1)$ space complexity.

Reflection:
Through the exercise problems, I have grasped the concept of solving multiplicative inverse problems using the Extended Euclidean Algorithm. Additionally, I have gained insights into other algorithms, including Fermat's Little Theorem, the Master Theorem, and Divide-and-Conquer techniques. Furthermore, I have learned about encryption and decryption processes and how to analyze algorithm efficiency in terms of running time.

Acknowledgements:
I would like to express my sincere gratitude to the following individuals and resources for their invaluable contributions to this assignment:
1) Prof. Bruce Maxwell: Thank you, Professor Bruce Maxwell for your guidance in this assignment. Your expertise in algorithms greatly influenced my work.
2) Classmates and TA's: I appreciate the discussions of my classmates, and TA's who clarified my doubts.
3) DPV Algorithms Textbook: This textbook was a useful resource for me to understand the basics of algorithms.
4) https://www.geeksforgeeks.org/karatsuba-algorithm-for-fast-multiplication-using-divide-and-conquer-algorithm/: This website provided clear explanations of binary multiplication, enhancing my understanding.
5) ibraryguides.centennialcollege.ca/c.php?g=717548p=5121841: This website helped me to understand modular arithmetic properties.
6) https://www.geeksforgeeks.org/introduction-to-divide-and-conquer-algorithm-data-structure-and-algorithm-tutorials/: This website provided me with an understanding of various algorithms used to find the majority element in an array.
7) https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/equivalence-relations: To understand modular arithmetic properties in detail.