Multiplicative order - GeeksforGeeks

Source: https://www.geeksforgeeks.org/multiplicative-order/

Courses Tutorials Practice Jobs DSA Tutorial Interview Questions Quizzes Must Do Advanced DSA System Design Aptitude Puzzles Interview Corner DSA Python Technical Scripter 2026 Explore DSA Fundamentals Logic Building Problems Analysis of Algorithms Data Structures Array Data Structure String in Data Structure Hashing in Data Structure Linked List Data Structure Stack Data Structure Queue Data Structure Tree Data Structure Graph Data Structure Trie Data Structure Algorithms Searching Algorithms Sorting Algorithms Introduction to Recursion Greedy Algorithms Tutorial Graph Algorithms Dynamic Programming or DP Bitwise Algorithms Advanced Segment Tree Binary Indexed Tree or Fenwick Tree Square Root (Sqrt) Decomposition Algorithm Binary Lifting Geometry Interview Preparation Interview Corner GfG160 Practice Problem GeeksforGeeks Practice - Leading Online Coding Platform Problem of The Day - Develop the Habit of Coding DSA Course 90% Refund Multiplicative order Last Updated : 14 Mar, 2023 In number theory, given an integer A and a positive integer N with gcd(A , N) = 1, the multiplicative order of a modulo N is the smallest positive integer k with A^k(mod N) = 1. (0 < K < N) Examples : Input : A = 4 , N = 7 Output : 3 explanation : GCD(4, 7) = 1 A^k(mod N) = 1 (smallest positive integer K) 4^1 = 4(mod 7) = 4 4^2 = 16(mod 7) = 2 4^3 = 64(mod 7) = 1 4^4 = 256(mod 7) = 4 4^5 = 1024(mod 7) = 2 4^6 = 4096(mod 7) = 1

smallest positive integer K = 3

Input : A = 3 , N = 1000 Output : 100 (3^100 (mod 1000) == 1)

Input : A = 4 , N = 11 Output : 5 If we take a close look then we observe that we do not need to calculate power every time. we can be obtaining next power by multiplying 'A' with the previous result of a module. Explanation : A = 4 , N = 11 initially result = 1 with normal with modular arithmetic (A * result) 4^1 = 4 (mod 11) = 4 || 4 * 1 = 4 (mod 11) = 4 [result = 4] 4^2 = 16(mod 11) = 5 || 4 * 4 = 16(mod 11) = 5 [result = 5] 4^3 = 64(mod 11) = 9 || 4 * 5 = 20(mod 11) = 9 [result = 9] 4^4 = 256(mod 11)= 3 || 4 * 9 = 36(mod 11) = 3 [result = 3] 4^5 = 1024(mod 5) = 1 || 4 * 3 = 12(mod 11) = 1 [result = 1]

smallest positive integer 5 Run a loop from 1 to N-1 and Return the smallest +ve power of A under modulo n which is equal to 1. Below is the implementation of above idea. C++ // C++ program to implement multiplicative order #include <bits/stdc++.h> using namespace std ; // function for GCD int GCD (int a , int b) { if (b == 0) return a ; return GCD (b , a % b) ; } // Function return smallest +ve integer that // holds condition A^k(mod N) = 1 int multiplicativeOrder (int A , int N) { if (GCD (A , N) != 1) return -1 ; // result store power of A that raised to // the power N-1 unsigned int result = 1 ; int K = 1 ; while (K < N) { // modular arithmetic result = (result * A) % N ; // return smallest +ve integer if (result == 1) return K ; // increment power K ++ ; } return -1 ; } //driver program to test above function int main () { int A = 4 , N = 7 ; cout << multiplicativeOrder (A , N); return 0 ; } Java // Java program to implement multiplicative order import java.io.* ; class GFG { // function for GCD static int GCD (int a , int b) { if (b == 0) return a ; return GCD (b , a % b); } // Function return smallest +ve integer that // holds condition A^k(mod N) = 1 static int multiplicativeOrder (int A , int N) { if (GCD (A , N) != 1) return - 1 ; // result store power of A that raised to // the power N-1 int result = 1 ; int K = 1 ; while (K < N) { // modular arithmetic result = (result * A) % N ; // return smallest +ve integer if (result == 1) return K ; // increment power K ++ ; } return - 1 ; } // driver program to test above function public static void main (String args []) { int A = 4 , N = 7 ; System . out . println (multiplicativeOrder (A , N)); } } /* This code is contributed by Nikita Tiwari.*/ Python3 # Python 3 program to implement # multiplicative order # function for GCD def GCD (a , b) : if (b == 0) : return a return GCD (b , a % b) # Function return smallest + ve # integer that holds condition # A ^ k(mod N) = 1 def multiplicativeOrder (A , N) : if (GCD (A , N) != 1) : return - 1 # result store power of A that raised # to the power N-1 result = 1 K = 1 while (K < N) : # modular arithmetic result = (result * A) % N # return smallest + ve integer if (result == 1) : return K # increment power K = K + 1 return - 1 # Driver program A = 4 N = 7 print (multiplicativeOrder (A , N)) # This code is contributed by Nikita Tiwari. C# // C# program to implement multiplicative order using System ; class GFG { // function for GCD static int GCD (int a , int b) { if (b

== 0) return a ; return GCD (b , a % b); } // Function return smallest +ve integer // that holds condition A^k(mod N) = 1 static int multiplicativeOrder (int A , int N) { if (GCD (A , N) != 1) return - 1 ; // result store power of A that // raised to the power N-1 int result = 1 ; int K = 1 ; while (K < N) { // modular arithmetic result = (result * A) % N ; // return smallest +ve integer if (result == 1) return K ; // increment power K ++ ; } return - 1 ; } // Driver Code public static void Main () { int A = 4 , N = 7 ; Console . Write (multiplicativeOrder (A , N)); } } // This code is contributed by Nitin Mittal. PHP <?php // PHP program to implement // multiplicative order // function for GCD function GCD ($a , $b) { if ($b == 0) return $a ; return GCD ($b , $a % $b) ; } // Function return smallest // +ve integer that holds // condition A^k(mod N) = 1 function multiplicativeOrder ($A , $N) { if (GCD ($A , $N) != 1) return - 1 ; // result store power of A // that raised to the power N-1 $result = 1 ; $K = 1 ; while ($K < $N) { // modular arithmetic $result = ($result * $A) % $N ; // return smallest +ve integer if ($result == 1) return $K ; // increment power $K ++ ; } return - 1 ; } // Driver Code $A = 4 ; $N = 7 ; echo (multiplicativeOrder ($A , $N)); // This code is contributed by Ajit. ?> JavaScript < script > // JavaScript program to implement // multiplicative order // function for GCD function GCD (a , b) { if (b == 0) return a ; return GCD (b , a % b); } // Function return smallest +ve integer that // holds condition A^k(mod N) = 1 function multiplicativeOrder (A , N) { if (GCD (A , N) != 1) return - 1 ; // result store power of A that raised to // the power N-1 let result = 1 ; let K = 1 ; while (K < N) { // modular arithmetic result = (result * A) % N ; // return smallest +ve integer if (result == 1) return K ; // increment power K ++ ; } return - 1 ; } // Driver Code let A = 4 , N = 7 ; document . write (multiplicativeOrder (A , N)); // This code is contributed by chinmoy1997pal. < /script> Output : 3 Time Complexity: O(N) Space Complexity: O(1) Reference: https://en.wikipedia.org/wiki/Multiplicative_order Comment Article Tags: Article Tags: Mathematical DSA Modular Arithmetic