Modular Exponentiation (Power in Modular Arithmetic)

Difficulty Level : Medium • Last Updated : 22 Apr, 2021

Given three numbers x, y and p, compute $(x^y) \% p$.

Examples :

```
Input:  x = 2, y = 3, p = 5
Output: 3
Explanation: 2^3 % 5 = 8 % 5 = 3.

Input:  x = 2, y = 5, p = 13
Output: 6
Explanation: 2^5 % 13 = 32 % 13 = 6.
```

Recommended: Please solve it on "***PRACTICE***" first, before moving on to the solution.

We have discussed recursive and iterative solutions for power.

Below is discussed iterative solution.

C++

```
/* Iterative Function to calculate (x^y) in O(log y) */
int power(int x, int y)
{

    // Initialize answer
    int res = 1;

    // Check till the number becomes zero
    while (y)
```

```
    {

        // If y is odd, multiply x with result
        if (y % 2 == 1)
            res = (res * x);

        // y = y/2
        y = y >> 1;

        // Change x to x^2
        x = (x * x);
    }
    return res;
}

// This code is contributed by yaswanth0412
```

C

```
/* Iterative Function to calculate (x^y) in O(log y) */
int power(int x, unsigned int y)
{
    int res = 1;     // Initialize result

    while (y > 0)
    {
        // If y is odd, multiply x with result
        if (y & 1)
            res = res*x;

        // y must be even now
        y = y>>1; // y = y/2
        x = x*x;  // Change x to x^2
    }
    return res;
}
```

Java

```
/* Iterative Function to calculate (x^y) in O(log y) */
static int power(int x, int y)
{
    int res = 1;     // Initialize result

    while (y > 0)
    {
```

```
        // If y is odd, multiply x with result
        if ((y & 1) != 0)
            res = res * x;

        // y must be even now
        y = y >> 1; // y = y/2
        x = x * x;  // Change x to x^2
    }
    return res;
}

// This code is contributed by Dharanendra L V.
```

Python3

```
# Iterative Function to calculate (x^y) in O(log y)
def power(x, y):

    # Initialize result
    res = 1

    while (y > 0):

        # If y is odd, multiply x with result
        if ((y & 1) != 0):
            res = res * x

        # y must be even now
        y = y >> 1 # y = y/2
        x = x * x  # Change x to x^2

    return res

# This code is contributed by Khushboogoyal499
```

C#

```
/* Iterative Function to calculate (x^y) in O(log y) */
static int power(int x, int y)
{
    int res = 1;     // Initialize result

    while (y > 0)
    {
        // If y is odd, multiply x with result
        if ((y & 1) != 0)
            res = res * x;
```

```
        // y must be even now
        y = y >> 1; // y = y/2
        x = x * x;  // Change x to x^2
    }
    return res;
}

// This code is contributed by Dharanendra L V.
```

Javascript

```
<script>
/* Iterative Function to calculate (x^y) in O(log y) */
function power(x, y)
{
    let res = 1;     // Initialize result

    while (y > 0)
    {
        // If y is odd, multiply x with result
        if (y & 1)
            res = res*x;

        // y must be even now
        y = y>>1; // y = y/2
        x = x*x; // Change x to x^2
    }
    return res;
}

// This code is contributed by _saurabh_jaiswal
</script>
```

Efficient Approach:

The problem with above solutions is, overflow may occur for large value of n or x. Therefore, power is generally evaluated under modulo of a large number.

Below is the fundamental modular property that is used for efficiently computing power under modular arithmetic.

```
(ab) mod p = ( (a mod p) (b mod p) ) mod p

For example a = 50,  b = 100, p = 13
50  mod 13  = 11
100 mod 13  = 9

(50 * 100) mod 13 = ( (50 mod 13) * (100 mod 13) ) mod 13
or (5000) mod 13 = ( 11 * 9 ) mod 13
or 8 = 8
```

Below is the implementation based on above property.

C++14

```
// Iterative C++ program to compute modular power
#include <iostream>
using namespace std;

/* Iterative Function to calculate (x^y)%p in O(log y) */
int power(long long x, unsigned int y, int p)
{
    int res = 1;     // Initialize result

    x = x % p; // Update x if it is more than or
```

```
                // equal to p

    if (x == 0) return 0; // In case x is divisible by p;

    while (y > 0)
    {
        // If y is odd, multiply x with result
        if (y & 1)
            res = (res*x) % p;

        // y must be even now
        y = y>>1; // y = y/2
        x = (x*x) % p;
    }
    return res;
}

// Driver code
int main()
{
    int x = 2;
    int y = 5;
    int p = 13;
    cout << "Power is " << power(x, y, p);
    return 0;
}

// This code is contributed by shubhamsingh10
```

Java

```
// Iterative Java program to compute modular power
import java.io.*;
class GFG
{

  /* Iterative Function to calculate (x^y) in O(log y) */
  static int power(int x, int y, int p)
  {
    int res = 1; // Initialize result

    x = x % p; // Update x if it is more than or
    // equal to p

    if (x == 0)
      return 0; // In case x is divisible by p;

    while (y > 0)
    {
```

```
      // If y is odd, multiply x with result
      if ((y & 1) != 0)
        res = (res * x) % p;

      // y must be even now
      y = y >> 1; // y = y/2
      x = (x * x) % p;
    }
    return res;
  }

  // Driver Code
  public static void main(String[] args)
  {
    int x = 2;
    int y = 5;
    int p = 13;
    System.out.print("Power is " + power(x, y, p));
  }
}

// This code is contributed by Dharanendra L V.
```

Python3

```
# Iterative Python3 program
# to compute modular power

# Iterative Function to calculate
# (x^y)%p in O(log y)
def power(x, y, p) :
    res = 1      # Initialize result

    # Update x if it is more
    # than or equal to p
    x = x % p

    if (x == 0) :
        return 0

    while (y > 0) :

        # If y is odd, multiply
        # x with result
        if ((y & 1) == 1) :
            res = (res * x) % p

        # y must be even now
        y = y >> 1       # y = y/2
```

```
        x = (x * x) % p

    return res


# Driver Code

x = 2; y = 5; p = 13
print("Power is ", power(x, y, p))


# This code is contributed by Nikita Tiwari.
```

C#

```
using System;
public class GFG
{

  /* Iterative Function to calculate (x^y) in O(log y) */
  static int power(int x, int y, int p)
  {
    int res = 1; // Initialize result

    x = x % p; // Update x if it is more than or
    // equal to p

    if (x == 0)
      return 0; // In case x is divisible by p;

    while (y > 0)
    {

      // If y is odd, multiply x with result
      if ((y & 1) != 0)
        res = (res * x) % p;

      // y must be even now
      y = y >> 1; // y = y/2
      x = (x * x) % p;
    }
    return res;
  }

  // Driver Code
  static public void Main ()
  {
    int x = 2;
    int y = 5;
    int p = 13;
```

```
    Console.Write("Power is " + power(x, y, p));
  }
}

// This code is contributed by Dharanendra L V.
```

PHP

```
<?php
// Iterative PHP program to
// compute modular power

// Iterative Function to
// calculate (x^y)%p in O(log y)
function power($x, $y, $p)
{
    // Initialize result
    $res = 1;

    // Update x if it is more
    // than or equal to p
    $x = $x % $p;

    if ($x == 0)
        return 0;

    while ($y > 0)
    {
        // If y is odd, multiply
        // x with result
        if ($y & 1)
            $res = ($res * $x) % $p;

        // y must be even now

        // y = $y/2
        $y = $y >> 1;
        $x = ($x * $x) % $p;
    }
    return $res;
}

// Driver Code
$x = 2;
$y = 5;
$p = 13;
echo "Power is ", power($x, $y, $p);

// This code is contributed by aj_36
```

```
?>
```

Javascript

```
// Iterative Javascript program to
// compute modular power

// Iterative Function to
// calculate (x^y)%p in O(log y)
function power(x, y, p)
{
    // Initialize result
    let res = 1;

    // Update x if it is more
    // than or equal to p
    x = x % p;

    if (x == 0)
        return 0;

    while (y > 0)
    {
        // If y is odd, multiply
        // x with result
        if (y & 1)
            res = (res * x) % p;

        // y must be even now

        // y = $y/2
        y = y >> 1;
        x = (x * x) % p;
    }
    return res;
}

// Driver Code
let x = 2;
let y = 5;
let p = 13;
document.write("Power is " + power(x, y, p));

// This code is contributed by _saurabh_jaiswal
```

Output

```
Power is 6
```

Time Complexity of above solution is O(Log y).

Modular exponentiation (Recursive)

This article is contributed by **Shivam Agrawal**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Previous

Write a program to calculate pow(x,n)

Next

Modular exponentiation (Recursive)

RECOMMENDED ARTICLES

Page : 1 2 3

01 **Modular exponentiation (Recursive)**
31, Aug 18

02 **Modular Exponentiation of Complex Numbers**
23, Jul 19

03 **Modular Arithmetic**
04, May 20

04 **Find Nth term (A matrix exponentiation example)**
02, Oct 18

05 **Expected number of moves to reach the end of a board | Matrix Exponentiation**
27, Dec 19

06 **Matrix Exponentiation**
12, Mar 16

07 **Check if given number is a power of d where d is a power of 2**
05, Jun 18

08 **Compute power of power k times % m**
28, Jun 18

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy | Normal | Medium | Hard | Expert

Improved By : jit_t, rd10, Mithun Kumar, gp6, SHUBHAMSINGH10, ankit_tiwari_, suryalinkin, dharanendralv23, _saurabh_jaiswal, khushboogoyal499, yaswanth0412

Article Tags : Google, large-numbers, Modular Arithmetic, Divide and Conquer, Mathematical

Practice Tags : Google, Mathematical, Divide and Conquer, Modular Arithmetic

Improve Article | Report Issue

Company-wise
Topic-wise
How to begin?

Write Interview Experience
Internships
Videos