Optimization Techniques | Set 1 (Modulus)

Modulus operator is costly.

The modulus operator (%) in various languages is costly operation. Ultimately every operator/operation must result in processor instructions. Some processors won't have modulus instruction at hardware level, in such case the compilers will insert stubs (predefined functions) to perform modulus. It impacts performance.

There is simple technique to extract remainder when a number is divided by another number (divisor) that is power of 2? A number that is an exact power of 2 will have only one bit set in it's binary representation. Consider the following powers of 2 and their binary representations

2 – 10

4 – 100

8 – 1000

16 – 10000

Note those zeros in red color, they contribute to remainder in division operation. We can get mask for those zeros by decrementing the divisor by 1.

Generalizing the above pattern, a number that can be written in 2^n form will have only one bit set followed by *n* zeros on the right side of 1. When a number (N) divided by (2^n), the bit positions corresponding to the above mentioned *zeros* will contribute to the remainder of division operation. An example can make it clear,

```
N = 87 (1010111 – binary form)

N%2 = N & (2-1) = 1010111 & 1 = 1 = 1

N%4 = N & (4-1) = 1010111 & 11 = 11 = 3

N%8 = N & (8-1) = 1010111 & 111 = 111 = 7
```

```
N%16 = N & (16-1) = 1010111 & 1111 = 111 = 7

N%32 = N & (32-1) = 1010111 & 11111 = 10111 = 23
```

Modulus operation over exact powers of 2 is simple and faster bitwise ANDing. This is the reason, programmers usually make buffer length as powers of 2.

Note that the technique will work only for divisors that are powers of 2.

An Example:

Implementation of circular queue (ring buffer) using an array. Omitting one position in the circular buffer implementation can make it easy to distinguish between *full* and *empty* conditions. When the buffer reaches SIZE-1, it needs to wrap back to initial position. The wrap back operation can be simple AND operation if the buffer size is power of 2. If we use any other size, we would need to use modulus operation.

Note:

Per experts comments, premature optimization is an evil. The optimization techniques provided are to fine tune your code after finalizing design strategy, algorithm, data structures and implementation. We recommend to avoid them at the start of code development. Code readability is key for maintenance.

Thanks to Venki for writing the above article. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Practice Tags : Bit Magic

Article Tags : Bit Magic

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content. Login to Improve this Article

Recommended Posts:

Add 1 to a given number

Next higher number with same number of set bits
Multiply a given Integer with 3.5
What are the differences between bitwise and logical AND operators in C/C++?
Compute the minimum or maximum of two integers without branching
Remove duplicates from a string in O(1) extra space
Calculate the total fine to be collected
Sum of bitwise OR of all subarrays
Value in a given range with maximum XOR
Check if a number can be expressed as 2^x + 2^y

(Login to Rate)

2.7 Average Difficulty : **2.7/5.0**
Based on **17** vote(s)

Basic | Easy | Medium | Hard | Expert

Add to TODO List
Mark as DONE

Feedback | Add Notes | Improve Article

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Share this post!

rexin冲 • 2 years ago

For any N bit binary numbers when we divide it by 2 ^x then last "x" bit gives remainder and remaining "n-x" bit give quotient !!

1 • Reply • Share ›

Vibhu Tiwari • 6 years ago

Thanks for the code its really a good logic

• Reply • Share ›

vinay polisetti • 7 years ago

Though the logic is silly but it is worth knowing it !!

```
/* Paste your code here (You may delete these lines if not writing code) */
```

1 • Reply • Share ›

Abhirup Ghosh • 8 years ago

This solution does not work for N being -ve numbers.

• Reply • Share ›

Venki ➚ Abhirup Ghosh • 8 years ago

@Abhirup, thanks for pointing. Signed numbers will have different bit pattern based on their representations. Most processors support 2's complement form of -ve numbers.

The above explanation holds for unsigned integers, and with little modification for signed magnitude of negative numbers.

• Reply • Share ›

A computer science portal for geeks

710-B, Advant Navis Business Park,
Sector-142, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org