forthright48

Learning Never Ends

Saturday, July 11, 2015

Number of Divisors of an Integer

Problem

Given an integer N, find its number of divisors.

For example, 12 has the following divisors $1, 2, 3, 4, 6$ and 12. So its number of divisors is 6.

Number Of Divisors Function: $NOD(N)$ or $\sigma_0(N)$

Number of Divisors of N is also called $\sigma_0(N)$. That's just a fancy way of asking for the number of divisors of N. You can read more on it from Wiki. I will refer to Number of Divisors of N as either $\sigma_0(N)$ or $NOD(N)$. They both mean same.

Brute Force Method O(N)

Anyways, what is the easiest way to find Number of Divisors (NOD)? We can try to divide N with all numbers from $1 - N$ and keep count of how many of those numbers divide N. This will definitely work, but obviously we need to do better.

Optimized to O($\sqrt{N}$)

In "Primality Test - Naive Methods", we established that if we can write $N = A \times B$, then one of $\{A, B\}$ must be $<= \sqrt{N}$.

So using that same idea we can find NOD by simply looping till $\sqrt{N}$ and check if that particular number divides N. If it doesn't then it is not a divisor and if it does then we found a $\{A, B\}$ pair of divisors. If $A \neq B$, then we add 2 to result, otherwise we add 1.

Examples

Suppose $N = 24$. We need to loop till $\lfloor \sqrt{24} \rfloor = 4$. We get the following pairs of divisors, $\{1, 24\}, \{2, 12\}, \{3, 8\}, \{4, 6\}$ So our answer is 8.

Let's try again with $N = 36$. We need to loop till $\lfloor \sqrt{36} \rfloor = 6$. We get the following pairs of divisors, $\{1, 36\}, \{2, 18\}, \{3, 12\}, \{4, 9\}, \{6, 6\}$ So our answer is 9. Notice that the last pair does not satisfy the condition $A \neq B$. So we add one for that pair.

This brings out an interesting observation. NOD of N is always even except for when N is a perfect square. Because whenever N is perfect square, $\sqrt{N}$ would be its divisor and it will form a pair with itself.

Code Using Loop till $\sqrt{N}$

```
int NOD ( int n ) {
    int res = 0;
    int sqrtn = sqrt ( n );

    for ( int i = 1; i < sqrtn; i++ ) {
        if ( n % i == 0 ) res += 2; //Found a divisor pair {A,B}
    }

    //Need to check if sqrtn divides n
    if ( n % sqrtn == 0 ) {
        if ( sqrtn * sqrtn == n ) res++; //If n is perfect square
        else res += 2; //Not perfect square
    }

    return res;
```

We loop from 1 to $\sqrt{N}-1$ at line 5. Then at line 10, we handle $\sqrt{N}$ seperately.

Using Prime Factorization

It is possible to find NOD from prime factorization of N.

Suppose $N = 12$. Its prime factorization is $2^2 \times 3$. Is it possible to divide 12 with 5? No. Cause the prime factorization of 12 does not contain 5. We can divide N with primes that appear in factorization of N.

Next, can we divide 12 with 2^3? No. The power of 2 in prime factorization of 12 is 2^2. So we cannot divide 12 with 2^3, since 2^2 is not divisible by 2^3.

So, if we are trying to divide N, that has prime factorization $N = p_1^{a_1} \times p_2^{a_2} \times \dots p_x^{a_x}$, then the divisors of N, D, will have prime factorization of form $D = p_1^{b_1} \times p_2^{b_2} \times \dots p_x^{b_x}$, where $b_i <= a_i$.

For example, divisor of $12 = 2^2 \times 3$ will have prime factorization, $D = 2^x \times 3^y$, where $x <= 2, y <= 1$. When $x = 1$ and $y = 1$, $D = 2^1 \times 3^1 = 6$ which is a divisor of 12. If we use a different combination of $[x, y]$, then we get a different divisor. So how many different combination can we use here? We know that, $0 <= x <= 2$ and $0 <= y <= 1$. So for x we can use $\{0, 1, 2\}$ and for y we can use $\{0, 1\}$. So we can use $3 \times 2 = 6$ combinations of $\{x, y\}$. And that is our NOD.

So, $if\ N = p_1^{a_1} \times p_2^{a_2} \times \dots p_x^{a_x},\ then\ D = p_1^{b_1} \times p_2^{b_2} \times \dots p_x^{b_x}$. We get new divisors for using different combination for $\{b_1, b_2 \dots b_x\}$. How many different combinations can we make? $(a_1 + 1) \times (a_2 + 1) \times \dots (a_x + 1)$. Therefore, $\sigma_0(N) = (a_1 + 1) \times (a_2 + 1) \times \dots (a_x + 1)$.

So basically, in order to find NOD, all we need to do is factorize N, then take each of its prime factors power, increase them by 1 and finally multiply all of them. Easy right?

Examples

$N = 12 = 2^2 \times 3.\ NOD(N) = (2+1) \times (1+1) = 6.$
$N = 15 = 3 \times 5.\ NOD(N) = (1+1) \times (1+1) = 4.$
$N = 252 = 2^2 \times 3^2 \times 7.\ NOD(N) = (2+1) \times (2+1) \times (1+1) = 18.$

Try out this yourself using other small values of N.

Code Using Prime Factorization

The code for NOD is not very different from the one we used for $factorize()$ in Prime Factorization of an Integer. We just need to modify it slightly to suit our needs.

```
int NOD ( int n ) {
    int sqrtn = sqrt ( n );
    int res = 1;
    for ( int i = 0; i < prime.size() && prime[i] <= sqrtn; i++ ) {
        if ( n % prime[i] == 0 ) {
            int p = 0; /*Counter for power of prime*/
            while ( n % prime[i] == 0 ) {
                n /= prime[i];
                p++;
            }
            sqrtn = sqrt ( n );
            p++;/*Increase it by one at end*/
            res *= p; /*Multiply with answer*/
        }
    }
    if ( n != 1 ) {
        res *= 2; /*Remaining prime has power p^1. So multiply with 2*/
    }
    return res;
}
```

I have highlighted the lines that are different from $factorize()$.

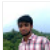
Reference

1. Divisor Function - https://en.wikipedia.org/wiki/Divisor_function

Posted by Mohammad Samiul Islam

Labels: Divisors, Factorization, Number Theory

1 comment:


Misbah Ahmad October 28, 2016 at 2:30 PM

Such a great Article.
Thanks for it.

Reply

Enter your comment...


Comment as: BIDDUT SARI ▼ Sign out

Publish Preview Notify me

Leave comments for Queries, Bugs and Hugs.

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Popular Posts

SPOJ LCMSUM - LCM Sum
Problem Problem Link - SPOJ LCMSUM Given n, calculate the sum $LCM(1,n) + LCM(2,n) + \ldots + LCM(n,n)$, where $LCM(i,n)$ denotes the ...

Euclidean Algorithm - Greatest Common Divisor
Problem Given two number A and B, find the greatest number that divides both A and B. What we are trying to find here is the Greatest Comm...

Extended Euclidean Algorithm
Extended Euclidean Algorithm is an extension of Euclidean Algorithm which finds two things for integer a and b: It finds the value of...

Chinese Remainder Theorem Part 1 - Coprime Moduli
Second part of the series can be found on: Chinese Remainder Theorem Part 2 - Non Coprime Moduli Wow. It has been two years since I pub...

Prufer Code: Linear Representation of a Labeled Tree
I guess this is going to be my first post (apart from the contest analysis') which is not about Number Theory! It's not about graph ...

Segmented Sieve of Eratosthenes
Problem Given two integers A and B, find number of primes inside the range of A and B inclusive. Here, $1 \leq A \leq B \leq 10^{...

Sieve of Eratosthenes - Generating Primes
Problem Given an integer N, generate all primes less than or equal to N. Sieve of Eratosthenes - Explanation Sieve of Eratosthenes ...

Number of Digits of Factorial
Problem Given an integer N, find number of digits in $N!$. For example, for $N = 3$, number of digits in $N! = 3! = 3\times 2\times 1...

Chinese Remainder Theorem Part 2 - Non Coprime Moduli
As promised on the last post, today we are going to discuss the "Strong Form" of Chinese Remainder Theorem, i.e, what do we do whe...

Euler Totient or Phi Function
I have been meaning to write a post on Euler Phi for a while now, but I have been struggling with its proof. I heard it required Chinese Rem...