Monday, July 20, 2015

Sum of Divisors of an Integer

Problem

Given an integer N, find the sum of all its divisors.

For example, find Sum Of Divisors (SOD) of 12. 12 has the following divisors, $\{1,2,3,4,6,12\}$. So $SOD(12) = 1+2+3+4+6+12 = 28$.

Sum of Divisor Function: $SOD(N)$ or $\sigma_1(N)$

The Sum of Divisors of N is also called $\sigma_1(N)$. That's just a fancy way of saying Sum of Divisors of N. You can read more on it from Wiki. I will refer to Sum of Divisors of N as either $\sigma_1(N)$ or $SOD(N)$. They both mean same.

Inefficient Solutions

Let us first look into some simple, but inefficient, solutions. These solutions are similar to the naive solutions we came up for "Number of Divisors of an Integer".

Loop Till N

This is the simplest solution. We loop from $1\ to\ N$ and add all numbers that divide N.

Loop Till $\sqrt{N}$

If $N = A \times B$, where $A \leq B$, A must be $\leq \sqrt{N}$. Using this fact we can run a loop from $1\ to\ \sqrt{N}$ and add all numbers that divide N and their complements to result.

```
1  int SOD ( int n ) {
2      int sqrtn = sqrt ( n );
3      int res = 0;
4      for ( int i = 1; i < sqrtn; i++ ) {
5          if ( n % i == 0 ) {
6              res += i; //"i" is a divisor
7              res += n / i; //"n/i" is also a divisor
8          }
9      }
10     if ( n % sqrtn == 0 ) {
11         if ( sqrtn * sqrtn == n ) res += sqrtn; //Perfect Square.
12         else {
13             res += sqrtn; //Two different divisor
14             res += n / sqrtn;
15         }
16     }
17     return res;
18 }
```

At line 10, we handle $sqrtn$ separately cause when N is a perfect square we don't get different values for $\{A,B\}$ pair. For example, for 49 we have a divisor pair 7×7. We need to add 7 only once in our result.

Using Prime Factorization

It is possible to find $SOD(N)$ using the prime factorization of N. Let us see an example for it.

$$Let\ N = 12,$$
$$SOD(12) = 1+2+3+4+6+12,$$
$$SOD(12) = (2^0 \times 3^0) + (2^1 \times 3^0) + (2^0 \times 3^1) + (2^2 \times 3^0) + (2^1 \times 3^1) + (2^2 \times 3^1)$$
$$SOD(12) = 2^0(3^0+3^1) + 2^1(3^0+3^1) + 2^2(3^0+3^1),$$
$$SOD(12) = (2^0+2^1+2^2) \times (3^0+3^1)$$

This pattern emerges with any value of N. More generally, if $N = p_1^{a_1} \times p_2^{a_2} \times \ldots p_k^{a_k}$, then

$$SOD(N) = (p_1^0 + p_1^1 + p_1^2 \ldots p_1^{a_1}) \times (p_2^0 + p_2^1 + p_2^2 \ldots p_2^{a_2}) \times \ldots (p_k^0 + p_k^1 + p_k^2 \ldots p_k^{a_k})$$

Using this formula and our code for Prime Factorization, we can write the function $SOD(N)$. This code is similar to $factorize()$. We only need to modify it in few places.

```
1  int SOD( int n ) {
2      int res = 1;
3      int sqrtn = sqrt ( n );
4      for ( int i = 0; i < prime.size() && prime[i] <= sqrtn; i++ ) {
5          if ( n % prime[i] == 0 ) {
6              int tempSum = 1; //Contains value of (p^0+p^1+...p^a)
7              int p = 1;
8              while ( n % prime[i] == 0 ) {
9                  n /= prime[i];
10                 p *= prime[i];
11                 tempSum += p;
12             }
13             sqrtn = sqrt ( n );
14             res *= tempSum;
15         }
16     }
17     if ( n != 1 ) {
18         res *= ( n + 1 ); //Need to multiply (p^0+p^1)
19     }
20     return res;
21 }
```

For each prime we find in factorization, we need to find the corresponding value of $(p^0 + p^1 + \ldots p^a)$. For that, we use $tempSum$ in line 6. It contains p^0 initially. Then for each successful division at line 8, we increase the power of p in line 10 and add it to $tempSum$ in line 11. Eventually, we multiply the final sum to result at line 14.

In line 17, we check if we are left with a sole prime remainder. In this case, we know that n is a prime so we simply multiply $p^0 + p^1 = 1 + n$ to the result.

Reference

1. Wiki - Divisor Function - https://en.wikipedia.org/wiki/Divisor_function
2. forthright48 - Number of Divisors of an Integer - http://forthright48.blogspot.com/2015/07/number-of-divisors-of-integer.html
3. forthright48 - Prime Factorization - http://forthright48.blogspot.com/2015/07/prime-factorization-of-integer.html

Posted by Mohammad Samiul Islam G+

Labels: Divisors, Factorization, Number Theory

No comments:

Post a Comment

Leave comments for Queries, Bugs and Hugs.

Enter your comment...

Comment as: BIDDUT SARI ▼ Sign out

Publish Preview Notify me

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...