Divisor Summatory Function

forthright48 on July 20, 2015

Problem

Given an integer N, find the Sum of Number of Divisors from $1 \rightarrow N$. That is, find $\sum_{i=1}^{N} NOD(i)$.

For example, find the Sum of Number of Divisors, $SNOD()$, when $N = 5$.
$SNOD(5) = NOD(1) + NOD(2) + NOD(3) + NOD(4) + NOD(5) = 1 + 2 + 2 + 3 + 2 = 10$.

Divisor Summatory Function

> **"In number theory, the divisor summatory function is a function that is a sum over the divisor function." – Wiki**

Divisor Summatory Function is defined as $D()$ in Wiki, but we will use $SNOD()$ in this article. Divisor Summatory Function finds Sum of Number of Divisors from $1 \rightarrow N$.

Brute Force Solution – $O(N^2)$ Approach

A simple solution is to run two nested loop which counts all divisors from $1 \rightarrow N$. It is simple but inefficient.

```
int SNOD( int n ) {
    int res = 0;
    for ( int i = 1; i <= n; i++ ) { // For each number from 1 - N
        for ( int j = 1; j <= i; j++ ) { // Find possible divisors of "i"
            if ( i % j == 0 ) res++;
        }
    }
    return res;
}
```

Using $NOD()$ - $O(N \times \frac{\sqrt{N}}{ln(\sqrt{N})})$ Approach

We can use the code for $NOD()$ to get the number of divisors for each integer from 1 to N instead of using a $O(N)$ loop. This will make it faster than the above solution.

```
int SNOD( int n ) {
    int res = 0;
    for ( int i = 1; i <= n; i++ ) {
        res += NOD(i);
    }
    return res;
}
```

Since $NOD()$ has same complexity as `factorize()`, the complexity for this code is $O(N \times \frac{\sqrt{N}}{ln(\sqrt{N})})$.

Using Divisor List - $O(N)$ Approach

Suppose we are trying to find $SNOD(10)$. Let us write down the divisors for each integer between 1 and 10.

$NOD(1) = 1, : 1$
$NOD(2) = 2, : 1, 2$
$NOD(3) = 2, : 1, 3$
$NOD(4) = 3, : 1, 2, 4$
$NOD(5) = 2, : 1, 5$
$NOD(6) = 4, : 1, 2, 3, 6$
$NOD(7) = 2, : 1, 7$
$NOD(8) = 4, : 1, 2, 4, 8$
$NOD(9) = 3, : 1, 3, 9$
$NOD(10) = 4, : 1, 2, 5, 10$

So $SNOD(10) = 1 + 2 + 2 + 3 + 2 + 4 + 2 + 4 + 3 + 4 = 27$.

Now, look at the divisor list for each of integer between 1 and N. Each divisor in the divisor lists contributes 1 to the result. There are 27 divisors in total.

How many times do we find 1 in the divisor lists? It will appear once for every integer it can divide from 1 to N. 1 can divide $\frac{N}{1}$ numbers. So it appears $\frac{10}{1} = 10$ times.

Repeat for $2 \rightarrow N$. How many times do we find 2? 2 can divide $\frac{10}{2} = 5$ numbers, namely $2, 4, 6, 8, 10$. So it will appear 5 times.

By repeating this from 1 to N we can find $SNOD(N)$.

```
1 int SNOD( int n ) {
2     int res = 0;
3     for ( int i = 1; i <= n; i++ ) {
4         res += n / i;
5     }
6     return res;
7 }
```

We removed $NOD()$ from the code in line 4 and replaced it with $\frac{n}{i}$.

Using Divisor Pairs - $O(\sqrt{N})$ Approach

Let us create another list. Instead of making divisor list, we will make "Ordered Divisor Pair" list for each value between 1 to N. A divisor pair of value X is a pair (A, B) such that $A \times B = X$. Ordered pair means if $A \neq B$, then (A, B) is a different pair than (B, A).

$NOD(1) = 1, : (1, 1)$
$NOD(2) = 2, : (1, 2)(2, 1)$
$NOD(3) = 2, : (1, 3)(3, 1)$
$NOD(4) = 3, : (1, 4)(2, 2)(4, 1)$
$NOD(5) = 2, : (1, 5)(5, 1)$
$NOD(6) = 4, : (1, 6)(2, 3)(3, 2)(6, 1)$
$NOD(7) = 2, : (1, 7)(7, 1)$
$NOD(8) = 4, : (1, 8)(2, 4)(4, 2)(8, 1)$
$NOD(9) = 3, : (1, 9)(3, 3)(9, 1)$
$NOD(10) = 4, : (1, 10)(2, 5)(5, 2)(10, 1)$

The "Ordered Divisor Pair" list is almost same as "Divisor List". The only difference is that each divisor A is now paired with $B = \frac{X}{A}$.

Now, $NOD(X)$ represents number of ordered divisor pairs (A, B) such that $A \times B = X$. So, $SNOD(N)$ is same as number of ordered divisor pairs such that $A \times B \leq N$.

How to find Number of Ordered Divisor Pairs $\leq N$?

We can find this by following three steps.

1. Find number of divisor pairs (A, B) where $A < B$

What are the possible values for A? Can it ever be $> \sqrt{N}$? No. If $A > \sqrt{N}$, then with $B > A$, B will be $> \sqrt{N}$ and $A \times B$ will be $> N$. So A can only be between 1 and $\lfloor \sqrt{N} \rfloor$.

Let $u = \lfloor \sqrt{N} \rfloor$. Then for each value of A between 1 and u, what are the possible values of B?

For any value of A, there are $\frac{N}{A}$ possible values of B for which $A \times B$ does not exceed N. For $N = 10$ and $A = 2$, there are $\frac{10}{2} = 5$ possible values for B, and they are $1, 2, 3, 4, 5$. Multiplying $A = 2$ with any of these values produces value $\leq N$. But we need $B > A$. So we omit $1, 2$ from the list. We can only use $3, 4, 5$ as B.

So, for any value of A, there will be $\lfloor \frac{N}{A} \rfloor$ possible values of B. The values will be from 1 to $\lfloor \frac{N}{A} \rfloor$. Out of those, we have to omit values that are $\leq A$. So we can use $\lfloor \frac{N}{A} \rfloor - A$ values for B.

For example, $N = 10$. Then $u = \lfloor \sqrt{10} \rfloor = 3$. When $A = 1$, number of legal values we can use is $\lfloor \frac{10}{1} \rfloor - 1 = 9$. Namely, B can be one of $2, 3, 4, 5, 6, 7, 8, 9, 10$.

When $A = 2$, legal values are $\lfloor \frac{10}{2} \rfloor - 2 = 3$. B can be $3, 4, 5$.
When $A = 3$, legal values are $\lfloor \frac{10}{3} \rfloor - 3 = 0$.

A cannot be bigger than u. So number of (A, B) pairs such that $A \times B \leq 10$ and $A < B$ is 12.

2. Multiply the result with 2

Once we find the number of pairs (A, B) such that $A < B$ and $A \times B \leq N$, we multiply that number with 2. That is because we want to find the number of ordered divisor pairs. When $A \neq B$, each pair (A, B) will occur in "Ordered Divisor Pair" list as (B, A) again.

Once we double the number, our result contains the number of ordered pair (A, B) such that $A \neq B$ and $A \times B \leq N$.

3. Add the number of pairs (A, B) where $A = B$

We still did not count pairs where $A = B$ and $A \times B \leq N$. How many pairs can there be? Since A must be between 1 and u, there cannot be more than u such pairs. The pairs will be $(1, 1), (2, 2) \ldots (u, u)$.

Code for $O(\sqrt{N})$ Approach

```
int SNOD( int n ) {
    int res = 0;
    int u = sqrt(n);
    for ( int i = 1; i <= u; i++ ) {
        res += ( n / i ) - i; //Step 1
    }
    res *= 2; //Step 2
    res += u; //Step 3
    return res;
}
```

Reference

1. forthright48 - Number of Divisors of an Integer - https://forthright48.com/2015/07/number-of-divisors-of-integer.html
2. Wiki - Divisor Summatory Function - https://en.wikipedia.org/wiki/Divisor_summatory_function

Category: CPPS, Number Theory

Previous:
Sum of Divisors of an Integer

Next:
Introduction to Modular Arithmetic

Archives

May 2019 (1)
April 2019 (1)
March 2019 (1)
December 2018 (2)
November 2018 (4)
September 2018 (2)
February 2018 (1)
January 2018 (1)
November 2017 (2)
September 2015 (7)
August 2015 (13)
July 2015 (15)

Categories

CPPS (45)
- Combinatorics (4)
- Data Structure (1)
- Number Theory (36)

Meta (1)
Misc (4)

Recent Comments

My Shopee Interview – Shadman Protik on My Interview Experience with Shopee / Garena / Sea Group
Istiad Hossain Akib on SPOJ LCMSUM – LCM Sum
Rifat Chowdhury on MyStory#02 – Deciding Where to Study CS
Salman Farsi on Leading Digits of Factorial
Learning Notes on Multiplicative Functions, Dirichlet Convolution, Mobius Inversion, etc – RobeZH's Blog on SPOJ LCMSUM – LCM Sum