Counting Divisors of a Number

Devising the divisors isn't that easy!

Pre Requisites

Basic Maths , $O(\sqrt{N})$ Factorisation , Primality testing

Motivation Problem

There are T test cases. Each test case contains a number N. For each test case , output the number of factors of N.

$1 <= T <= 10$
$1 <= N <= 10^{18}$

Note : 1 and N are also treated as factors of the number N.

Solution

Naive Solution - $O(\sqrt{N})$ *Factorisation*

The most naive solution here is to do the $O(\sqrt{N})$ factorisation. But as we can see, it will surely receive the Time Limit Exceeded verdict. You may try to compute the prime numbers till required range and loop over them , but that too exceeds the usual limit of 10^8 operations. Some optimisations and heuristics may allow you to squeeze through your solution, but usually at the cost of a few TLE's.

Supposing you fit in the above description and cannot think of anything better than the naive solution, I would like to present to you a very simple algorithm which helps you count the number of divisors in $O(N^{\frac{1}{3}})$, which would be useful for this kind of questions.

Firstly, let's do a quick recap of how we obtain the $O(\sqrt{N})$ algorithm for any number N :

We write N as product of two numbers P and Q.
$P * Q = N, where\ P <= Q$
$Maximum\ value\ of\ P = \sqrt{N}$
Looping over all values of P gives us the count of factors of N.

Before you move forward to the next section, it would be useful if you try to come up with an algorithm that finds the factors in $O(N^{\frac{1}{3}})$. As a hint, the way of thinking is same as that of getting to the $O(\sqrt{N})$ algorithm.

$\mathbf{O(N^{\frac{1}{3}})}$ Factorisation

Let's start off with some maths to reduce our $O(\sqrt{N})$ factorisation to $O(N^{\frac{1}{3}})$:

We write N as product of three numbers P, Q and R.
$P * Q * R = N, where\ P <= Q <= R$
$Maximum\ value\ of\ P = N^{\frac{1}{3}}$

We can loop over all prime numbers in range $[2, N^{\frac{1}{3}}]$ and try to reduce N to it's prime factorisation, which would help us count the number of factors of N.
To know how to calculate divisors using prime factorisation, click here.

We will split our number N into two numbers X and Y such that $X * Y = N$. Further, X contains only prime factors in range $[2, N^{\frac{1}{3}}]$ and Y deals with higher prime factors ($> N^{\frac{1}{3}}$). Thus, gcd(X , Y) = 1. Let the count of divisors of a number N be denoted by the function $F(N)$. It is easy to prove that this function is multiplicative in nature, i.e., $F(m * n) = F(m) * F(n)$, if gcd(M,N) = 1. So, if we can find $F(X)$ and $F(Y)$, we can also find $F(X * Y)$ or $F(N)$ which is the required quantity.

For finding $F(X)$, we use the naive trial division to prime factorise X and calculate the number of factors. Once this is done, we have $Y = N/X$ remaining to be factorised. This may look tough, but we can see that there are only three cases which will cover all possibilities of Y :

1. **Y is a prime number** : $F(Y) = 2$.
2. **Y is square of a prime number** : $F(Y) = 3$.
3. **Y is product of two distinct prime numbers** : $F(Y) = 4$.

We have only these three cases since there can be at max two prime factors of Y. If it would have had more than two prime factors, one of them would surely have been $<= N^{\frac{1}{3}}$, and hence it would be included in X and not in Y.

So once we are done with finding $F(X)$ and $F(Y)$, we are also done with finding $F(X * Y)$ or $F(N)$.

Pseudo Code

$N = input()$
primes = array containing primes till 10^6
$ans = 1$
for all p in primes :
    $if\ p * p * p > N$:
        break
    $count = 1$
    while N divisible by p:
        $N = N/p$
        $count = count + 1$
    $ans = ans * count$
if N is prime:
    $ans = ans * 2$
else if N is square of a prime:
    $ans = ans * 3$
else if $N \neq 1$:
    $ans = ans * 4$

Checking for primality can be done quickly using Miller Rabin. Thus, the time complexity is $O(N^{\frac{1}{3}})$ for every test case and hence we can solve our problem efficiently.

At this point, you may think that in a similar way, we can reduce this to $O(N^{\frac{1}{4}})$ by handling some cases. I have not thought much on it, but the number of cases to be handled are high since after trial division N could be factorised into one, two or three primes. This is easy enough to code in contest environment, which is our prime objective.

This trick is not quite commonly known and people tend to make bugs in handling the three cases. A problem in regionals which uses this trick directly :

1. Problem F — Codeforces Gym

You can try also this technique on problems requiring$\sqrt{N}$ factorisation for practice purposes.

Hope you found this useful!

Happy Coding!