OJ 12043

Divisors

Prof. Edson Alves - UnB/FGA

OJ 12043 – Divisors

Let us define the functions $d(n)$ *and* $\sigma(n)$ *as*

$$d(n) = \text{number of divisors of } n$$
$$\sigma(n) = \text{summation of divisors of } n$$

Here divisors of n *include both 1 and* n*. For example divisors of 6 are 1, 2, 3 and 6. So* $d(6) = 4$ *and* $\sigma(n) = 12$*.*

Now let us define two more function $g(a, b, k)$ *and* $h(a, b, k)$ *as*

$$g(a, b, k) = \sum_i d(i)$$
$$h(a, b, k) = \sum_i \sigma(i)$$

where $a \leq i \leq b$ *and* i *is divisible by* k*.*

For example, $g(5, 12, 3) = d(6) + d(9) + d(12) = 4 + 3 + 6 = 13$ *and* $h(5, 12, 3) = \sigma(6) + \sigma(9) + \sigma(12) = 13 + 13 + 28 = 53$*. Given* a, b, k *you have to calculate* $g(a, b, k)$ *and* $h(a, b, k)$*.*

Entrada e saída

Input

The first line of the input file contains an integer T $(T \leq 75)$ *which denotes the total number of test cases. The description of each test case is given below:*

Three integers in a line. First integer is contains a*, second integer is* b *and third integer is* k*. You may assume* $0 < a \leq b \leq 10^5, 0 < k < 2000$*.*

Output

For each test case print one line containing $g(a, b, k)$ *and* $h(a, b, k)$ *separated by a space as defined above.*

Exemplo de entrada e saída

Entrada

```
2
5 12 3
1 100 3
```

Saída

```
13 53
217 3323
```

Solução em $O(B \log B)$

- Para resolver este problema dentro do limite de tempo estabelecido, é preciso computar, de forma eficiente, as funções $\tau(m)$ e $\sigma(m)$ para todos os inteiros m de 1 a N
- Isto pode ser feito por meio de uma variante do crivo de Erastótenes
- Lembre que 1 é o único positivo que tem apenas um divisor
- Para os demais inteiros positivos tem, no mínimo, dois divisores: 1 e o próprio número
- A ideia portanto é iniciar os valores $\tau(m) = 2$ e $\sigma(m) = m + 1$
- Após esta inicialização, para cada inteiro positivo d no intervalo de 2 a N, devemos identificar quais inteiros m são divisíveis por d

Solução em $O(B \log B)$

- Para cada um destes inteiros os valores de $\tau(m)$ e $\sigma(m)$ devem ser atualizados, de acordo com o valor de $k = m/d$
- Se $d \neq k$, então $\tau(m)$ deve ser acrescido em duas unidades, pois são dois novos divisores de m encontrados, e $\sigma(m)$ deve aumentar em $d + k$ unidades
- Nos casos em que $d = k$, $\tau(m)$ deve ser incrementado em uma única unidade, e o valor de $\sigma(m)$ deve ser acrescido em d unidades
- É preciso tomar cuidado para que nenhum divisor seja contabilizado mais de uma vez
- Assim, os múltiplos de d começarão a ser considerados a partir de d^2

Solução em $O(B \log B)$

- De acordo com o crivo de Erastótenes, ao proceder desta maneira os múltiplos de d menores que d^2 já terão sido processados anteriormente
- De posse dos valores pré-computados de $\tau(n)$ e de $\sigma(n)$, a soma pode ser feita de forma linear
- Para evitar iterar sobre valores que não são múltiplos de k, o laço deve iniciar no primeiro múltiplo m de k que é maior ou igual a a, e o incremento deve ser feito em passos de tamanho k
- Este múltiplo m pode ser obtido por meio da expressão $m = kt$, onde

$$t = \left\lceil \frac{a}{k} \right\rceil$$

Solução $O(B \log B)$

```cpp
pair<vector<ll>, vector<ll>> tau_and_sigma(ll N)
{
    vector<ll> tau(N + 1, 1), sigma(N + 1, 1);

    for (ll m = 2; m <= N; ++m)
    {
        tau[m] = 2;
        sigma[m] = m + 1;
    }

```

Solução $O(B \log B)$

```
    for (ll d = 2; d <= N; ++d)
    {
        for (ll m = d*d; m <= N; m += d)
        {
            ll k = m / d;

            tau[m] += (d == k ? 1 : 2);
            sigma[m] += (d == k ? d : d + k);
        }
    }

    return { tau, sigma };
}
```

Solução $O(B \log B)$

```
pair<ll, ll>
solve(int a, int b, int k, const vector<ll>& tau, const vector<ll>& sigma)
{
    int t = (a + k - 1)/k, m = k*t;
    ll x = 0, y = 0;

    for (int i = m; i <= b; i += k)
    {
        x += tau[i];
        y += sigma[i];
    }

    return { x, y };
}
```