AtCoder Beginner Contest 172

Problema D: *Sum of Divisors*

Prof. Edson Alves - UnB/FGA

AtCoder Beginner Contest 172D – Sum of Divisors

For a positive integer X, let $f(X)$ be the number of positive divisors of X.

Given a positive integer N, find

$$\sum_{K=1}^{N} K \times f(K)$$

Entrada e saída

Constraints

- $1 \leq N \leq 10^7$

Input

Input is given from Standard Input in the following format:

$$N$$

Output

Print the value $\sum_{K=1}^{N} K \times f(K)$.

Exemplos de entradas e saídas

Entrada	Saída
4	23
100	26879
10000000	838627288460105

Solução com complexidade $O(N \log N)$

- Assim como fizemos no caso da função φ de Euler, é preciso computar o valor de τ para todos inteiros no intervalo $[1, N]$ de forma eficiente
- Uma vez que τ é uma função multiplicativa, isto pode ser feito por meio de uma variante do crivo de Erastótenes
- O crivo permite a identificação de um dos fatores primos de n, para $n \in [2, N]$
- No código da solução será mantido, para cada n, o maior primo p que o divide
- Uma vez identificados estes fatores primos, o valor de $\tau(n)$ é computado em ordem crescente

Solução com complexidade $O(N \log N)$

- Lembre que $\tau(1) = 1$, de modo que esta computação inicia em $n = 2$
- Para cada n, utilizamos o fator p para escrever $n = p^k \times m$, com $(p^k, m) = 1$
- Daí
$$\tau(n) = \tau(p^k)\tau(m) = (k+1)\tau(m)$$
- Como $m < n$, pois p é primo, quando $\tau(n)$ estiver sendo computado o valor de $\tau(m)$ já estará disponível
- Como a fatoração parcial de n tem complexidade $O(\log n)$, a solução terá complexidade $O(N \log N)$, a mesma do crivo modificado

Solução com complexidade $O(N \log N)$

```cpp
vector<ll> factors(ll N)
{
    bitset<MAX> sieve;
    vector<ll> fs(N + 1, 1);

    sieve.set();

    for (ll i = 2; i <= N; i++)
        if (sieve[i])
            for (ll j = i; j <= N; j += i)
            {
                sieve[j] = false;
                fs[j] = max(fs[j], i);
            }

    return fs;
}
```

Solução com complexidade $O(N \log N)$

```
vector<ll> divisors(ll N, const vector<ll>& fs)
{
    vector<ll> tau(N + 1, 1);

    for (ll n = 2; n <= N; ++n)
    {
        ll k = 0, m = n;

        for (auto p = fs[i]; m % p == 0; ++k, m /= p);

        tau[n] = (k + 1)*tau[m];
    }

    return tau;
}
```

Solução com complexidade $O(N \log N)$

```
ll solve(ll N)
{
    auto fs = factors(N);
    auto tau = divisors(N, fs);
    ll ans = 0;

    for (ll K = 1; K <= N; ++K)
        ans += (K * tau[K]);

    return ans;
}
```