Teoria dos Números

Teorema Fundamental da Aritmética

Prof. Edson Alves

Faculdade UnB Gama

Teorema Fundamental da Aritmética

Funções multiplicativas e números primos

- Uma função f é multiplicativa se $f(1) = 1$ e $f(ab) = f(a)f(b)$ se $(a, b) = 1$
- Uma consequência destas propriedades é que f pode ser determinada a partir dos valores que ela assume nas potências p^k de qualquer primo p
- Isto porque, se o inteiro positivo n é um produto de potências de primos, isto é

  $$n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k},$$

  então

  $$f(n) = f(p_1^{\alpha_1}) f(p_2^{\alpha_2}) \dots f(p_k^{\alpha_k})$$

- O Teorema Fundamental da Aritmética nos garante que, se $n > 1$, então n poderá ser escrito como este produto de primos

Teorema Fundamental da Arimética

Teorema Fundamental da Aritmética

Seja n um inteiro positivo maior do que 1. Então n pode ser escrito de forma única, exceto pela ordem dos fatores, como o produto de números primos.

Uma notação canônica para esta fatoração de n é a seguinte: se p_i é o i-ésimo número primo e $\alpha_i \geq 0$, para todo $i \in [1, k]$, então

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k}$$

Fatoração de inteiros

- O Teorema Fundamental da Aritmética apresenta a relação fundamental dos números primos com todos os números naturais
- O conhecimento da fatoração (ou decomposição) de um natural n como produto de primos permite o cálculo de várias funções importantes, como MDC, MMC ou qualquer função multiplicativa
- A fatoração serve como alternativa para a representação do número, principalmente quando o número é muito grande (maior do que a capacidade de um **`long long`**, por exemplo)

Implementação da fatoração de inteiros

- Uma maneira de se representar a fatoração de um inteiro positivo $n > 1$ é por meio de um dicionário, onde chave é um primo p divisor de n e o valor k é o maior expoente tal que p^k divide n
- É possível implementar a fatoração de duas formas
- Sem o conhecimento prévio da lista dos primos, a implementação terá complexidade $O(\sqrt{n})$
- O algoritmo será semelhante ao que lista todos os divisores de n, com duas diferenças

Implementação da fatoração de inteiros

- A primeira delas é que, ao encontrar um divisor p de n, o valor de n será atualizado para n/p^k
- A segunda é que, se após a verificação de todos os candidatos a divisores de n o valor de n permanecer maior do que 1, este valor remanescente será primo
- A fatoração pode ser implementada em $O(\pi(\sqrt{n}))$, se forem conhecida a lista dos números primos menores ou iguais a $\sqrt{n}$
- O algoritmo será o mesmo, com a diferença de que os candidatos a divisores serão todos primos

Implementação da fatoração em $O(\sqrt{n})$

```cpp
map<long long, long long> factorization(long long n) {
    map<long long, long long> fs;

    for (long long d = 2, k = 0; d * d <= n; ++d, k = 0) {
        while (n % d == 0) {
            n /= d;
            ++k;
        }

        if (k) fs[d] = k;
    }

    if (n > 1) fs[n] = 1;

    return fs;
}
```

Implementação da fatoração em $O(\pi(\sqrt{n}))$

```
map<long long, long long> factorization(long long n, vector<long long>& primes)
{
    map<long long, long long> fs;

    for (auto p : primes)
    {
        if (p * p > n)
            break;

        long long k = 0;

        while (n % p == 0) {
            n /= p;
            ++k;
        }

```

Implementação da fatoração em $O(\pi(\sqrt{n}))$

```
        if (k)
            fs[p] = k;
    }

    if (n > 1)
        fs[n] = 1;

    return fs;
}
```

MDC, MMC e fatoração

Sejam a e b dois inteiros positivos tais que $a = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$ e $b = p_1^{\beta_1} p_2^{\beta_2} \dots p_k^{\beta_k}$, com $\alpha_i, \beta_j \geq 0$ para todos $i, j \in [1, k]$. Então

$$(a, b) = p_1^{\min\{\alpha_1, \beta_1\}} p_2^{\min\{\alpha_2, \beta_2\}} \dots p_k^{\min\{\alpha_k, \beta_k\}}$$

e

$$[a, b] = p_1^{\max\{\alpha_1, \beta_1\}} p_2^{\max\{\alpha_2, \beta_2\}} \dots p_k^{\max\{\alpha_k, \beta_k\}}$$

Implemetação do MDC usando a fatoração de n

```cpp
int gcd(int a, int b, const vector<int>& primes)
{
    auto ps = factorization(a, primes);
    auto qs = factorization(b, primes);

    int res = 1;

    for (auto p : ps) {
        int k = min(ps.count(p) ? ps[p] : 0, qs.count(p) ? qs[p] : 0);

        while (k--)
            res *= p;
    }

    return res;
}
```

Implemetação do MMC usando a fatoração de n

```cpp
int lcm(int a, int b, const vector<int>& primes)
{
    auto ps = factorization(a, primes);
    auto qs = factorization(b, primes);

    int res = 1;

    for (auto p : ps) {
        int k = max(ps.count(p) ? ps[p] : 0, qs.count(p) ? qs[p] : 0);

        while (k--)
            res *= p;
    }

    return res;
}
```

Fatoriais

Fatoração de Fatoriais

- Uma aplicação importante do Teorema Fundamental da Aritmética é a fatoração de fatoriais
- Os fatoriais crescem rapidamente, e mesmo para valores relativamente pequenos de n, o número $n!$ pode ser computacionalmente intratável
- A fatoração de $n!$ permite trabalhar com tais números e realizar algumas operações com os mesmos (multiplicação, divisão, MMC, MDC, etc)
- A função $E(n, p)$ retorna o inteiro k tal que p^k é a maior potência do primo p que divide $n!$

Exemplo de cálculo de $E(n, p)$ para $n = 12$ e $p = 2$

Para ilustrar o cálculo de $E(n, p)$ considere $n = 12$ e $p = 2$. A expansão de $12!$ é

$$1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 \times 9 \times 10 \times 11 \times 12$$

É fácil observar que todos os múltiplos de 2 contribuem com um fator 2. Cancelando estes fatores obtém-se

$$1 \times 1 \times 3 \times 2 \times 5 \times 3 \times 7 \times 4 \times 9 \times 5 \times 11 \times 6$$

Exemplo de cálculo de $E(n,p)$ para $n = 12$ e $p = 2$

Ainda restam ainda fatores 2 no produto, onde haviam originalmente os números $4, 8$ e 12. Isto acontece por, além de serem múltiplos de 2, os números $4, 8$ e 12 também são múltiplos de 2^2.

Eliminando os fatores 2 associados a 2^2 resulta em

$$1 \times 1 \times 3 \times {\color{red}1} \times 5 \times 3 \times 7 \times {\color{red}2} \times 9 \times 5 \times 11 \times {\color{red}3}$$

Mais 3 fatores foram eliminados, e sobrou ainda um fator, onde estava o 8. Isto acontece também porque 8 é múltiplo de 2^3. Eliminando este último fator, foi removido um total de $6 + 3 + 1 = 10$ fatores do produto. Portanto $E(12, 2) = 10$.

Cálculo de $E(n, p)$

O exemplo anterior permite a deducação da expressão para o cálculo de $E(n, p)$:

$$E(n,p) = \left\lfloor \frac{n}{p} \right\rfloor + \left\lfloor \frac{n}{p^2} \right\rfloor + \ldots + \left\lfloor \frac{n}{p^r} \right\rfloor,$$

onde $\left\lfloor \frac{a}{b} \right\rfloor$ é a divisão inteira de a por b e p^r é a maior potência de p que é menor ou igual a n.

Implementação de $E(n, p)$ em C++ em $O(\log n)$

```cpp
int E(int n, int p)
{
    int k = 0, base = p;

    while (base <= n)
    {
        k += n / base;
        base *= p;
    }

    return k;
}
```

Implementação da fatoração de $n!$ em $O(\pi(n) \log n)$

```cpp
map<int, int> factorial_factorization(int n, const vector<int>& primes)
{
    map<int, int> fs;

    for (const auto& p : primes)
    {
        if (p > n)
            break;

        fs[p] = E(n, p);
    }

    return fs;
}
```

Referências

1. **HALIM**, Felix; **HALIM**, Steve. *Competitive Programming 3*, 2010.
2. **HEFEZ**, Abramo. Arimética, Coleção PROFMAT, SBM, 2016.
3. **LAAKSONEN**, Antti. *Competitive Programmer's Handbook*, 2018.
4. **SKIENA**, Steven S.; **REVILLA**, Miguel A. *Programming Challenges*, 2003.