分治乘法

sjzez czy

2019.1.23

目录

1 原理

设 $A(x)=\sum_{i=0}^{n-1}a_ix^i, B(x)=\sum_{i=0}^{n-1}b_ix^i$，且 $n=2^k$
将 $A(x),B(x)$ 分别拆成前后部分，即：

$$\begin{cases} A(x)=A_0(x)+x^{\frac{n}{2}}A_1(x) \\ B(x)=B_0(x)+x^{\frac{n}{2}}B_1(x) \end{cases}$$

则有：

$$\begin{aligned} A(x)B(x)=&\left(A_0(x)+x^{\frac{n}{2}}A_1(x)\right)\left(B_0(x)+x^{\frac{n}{2}}B_1(x)\right) \\ &A_0B_1+x^nA_1B_1+x^{\frac{n}{2}}\left(A_0B_1+A_1B_0\right) \\ &A_0B_1+x^nA_1B_1+x^{\frac{n}{2}}\left(\left(A_0-A_1\right)\left(B_1-B_0\right)+A_1B_1+A_0B_0\right) \end{aligned}$$

于是原先的四次乘法就变成了三次乘法

2 时间复杂度

$$T(n)=O(n)+3T(\frac{n}{2})=O(n^{\log_2 3})=O(n^{1.59})$$

3 空间复杂度

$$O(n)$$

4 代码实现

```
ll a[N], b[N], c[N], mem[N], top;
inline void mns(ll *a, ll *b, ll *res, int n) {
    for(int i = 0 ; i < n ; ++i) res[i] = (a[i] - b[i]) % mod;
}
inline void pls(ll *a, ll *b, ll *c, ll *res, int n) {
    for(int i = 0 ; i < n ; ++i) res[i] = (a[i] + b[i] + c[i]) % mod;
}
ll *_new(int x) {
    ll *res = mem + top;
    top += x;
```

```
    assert(top < N);
    return res;
}
void _delete(int x) {
    top -= x;
}
void sol(ll *a, ll *b, ll *res, int n) {
    if(n == 1) {
        res[0] = a[0] * b[0] % mod;
        res[1] = 0;
    } else if(n > 1) {
        ll *a0b0 = _new(n), *a1b1 = _new(n);
        ll *a0a1 = _new(n), *b1b0 = _new(n);
        ll *a0a1_b1b0 = _new(n);
        sol(a, b, a0b0, n / 2);
        sol(a + n / 2, b + n / 2, a1b1, n / 2);
        mns(a, a + n / 2, a0a1, n / 2);
        mns(b + n / 2, b, b1b0, n / 2);
        sol(a0a1, b1b0, a0a1_b1b0, n / 2);
        for(int i = 0 ; i < n ; ++i) {
            res[i] = a0b0[i];
            res[i + n] = a1b1[i];
        }
        pls(a0a1_b1b0, a1b1, a0b0, a0a1_b1b0, n);
        for(int i = 0 ; i < n ; ++i) {
            (res[i + n / 2] += a0a1_b1b0[i]) %= mod;
        }
        _delete(n * 5);
    }
}
```