

# 多项式求逆

准备工作

```
void calc_rev(int &n, int &lim, const int m) {
    n = 1, lim = 0;
    while(n < m) n <= 1, lim++;
    for(int i = 1; i < n; i++) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << lim - 1);
}
```

递归复杂度  $O(n \log n)$

```
void Inv(const int *a, int *b, const int len) { //多项式求逆
    static int c[N];
    if(len == 1) {
        b[0] = qpow(a[0], mod - 2);
        return;
    }
    Inv(a, b, len + 1 >> 1);
    int n, lim;
    calc_rev(n, lim, len << 1);
    for(int i = 0; i < n; i++) c[i] = a[i];
    for(int i = len; i < n; i++) c[i] = 0;
    NTT(c, n, 1), NTT(b, n, 1);
    for(int i = 0; i < n; i++) b[i] = (2 - (ll)c[i] * b[i] % mod + mod) * b[i] % mod;
    NTT(b, n, -1);
    for(int i = len; i < n; i++) b[i] = 0;
}
```

检查的方法:随机一个多项式求逆两次看看是不是自己。

From:  
<https://wiki.cvbbacm.com/> - CVBB ACM Team



Permanent link:  
[https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:%E7%8E%8B%E6%99%BA%E5%BD%AA.%E5%A4%9A%E9%A1%B9%E5%BC%8F%E6%B1%82%E9%80%86](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:%E7%8E%8B%E6%99%BA%E5%BD%AA.%E5%A4%9A%E9%A1%B9%E5%BC%8F%E6%B1%82%E9%80%86)

Last update: 2021/07/20 21:35