

# 多项式 1

## 拉格朗日插值法

### 算法简介

给定  $n$  个坐标  $(x_i, y_i)$  求解唯一确定的最高次不超过  $n-1$  次的多项式  $f(x)$  满足  $f(x_i) = y_i$

### 算法模板

[洛谷p4781](#)

构造  $g_i(x) = y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$  易知

$$g_i(x_j) = \begin{cases} y_i, & j = i \\ 0, & j \neq i \end{cases}$$

于是有

$$f(x) = \sum_{i=1}^n g_i(x) = \sum_{i=1}^n y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

根据上式可以  $O(n^2)$  计算出  $f(k)$

```
int x[MAXN], y[MAXN];
int Lagrange(int n, int k){
    int ans=0, a, b;
    _rep(i, 1, n){
        a=y[i], b=1;
        _rep(j, 1, n){
            if(j==i) continue;
            a=1LL*a*(k-x[j])%Mod;
            b=1LL*b*(x[i]-x[j])%Mod;
        }
        ans=(ans+1LL*a*inv(b)%Mod)%Mod;
    }
    return (ans%Mod+Mod)%Mod;
}
```

### 算法拓展

#### 线性优化

事实上，如果  $x_i$  取值连续，上述算法复杂度可以优化为  $O(n)$

以  $x_i = i$  为例，则 (1) 式化为

$$f(x) = \sum_{i=1}^n y_i \prod_{j \neq i} \frac{x - j}{i - j} = \sum_{i=1}^n (-1)^{n-i} y_i \frac{1}{i}$$

$$\{\prod_{j=1}^{i-1}(x-j)\prod_{j=i+1}^n(x-j)\} \{(i-1)!(n-i)!\}^{-1}$$

分子部分考虑  $O(n)$  预处理序列  $\{x_i\}$  的前缀积和后缀积，分母部分考虑  $O(n)$  预处理阶乘的逆。

```
int inv_frac[MAXN], y[MAXN], pre[MAXN], suf[MAXN];
int Lagrange(int x, int n) {
    pre[0] = suf[n+1] = 1;
    _rep(i, 1, n) pre[i] = 1LL * pre[i-1] * (x-i) % Mod;
    for(int i=n; i; i--) suf[i] = 1LL * suf[i+1] * (x-i) % Mod;
    int ans=0, sign;
    _rep(i, 1, n) {
        sign = ((n-i)&1) ? -1 : 1;
        ans = (ans + 1LL * sign * y[i] * pre[i-1] % Mod * suf[i+1] % Mod * inv_frac[i-1] % Mod * inv_frac[n-i]) % Mod;
    }
    return ans;
}
```

### 重心拉格朗日插值法

考虑  $O(n)$  时间动态插入每个点  $(x_i, y_i)$

令  $g(x) = \prod_{i=1}^n (x-x_i)$ ,  $w_i = \prod_{j \neq i} (x_i-x_j)$  于是  $(1)$  式化为

$$f(x) = \sum_{i=1}^n y_i \prod_{j \neq i} \frac{x-x_j}{x_i-x_j} = g(x) \sum_{i=1}^n \frac{y_i}{x_i} \frac{1}{w_i} \tag{2}$$

于是每次插入后动态更新每个  $w_i$  即可。

```
int x[MAXN], y[MAXN], w[MAXN], z;
void Insert(int tx, int ty) {
    w[++z] = 1;
    x[z] = tx, y[z] = ty;
    _for(i, 1, z) {
        w[i] = 1LL * w[i] * (x[i] - x[z]) % Mod;
        w[z] = 1LL * w[z] * (x[z] - x[i]) % Mod;
    }
}
int Lagrange(int k) {
    int g=1, s=0;
    _rep(i, 1, z) {
        g = 1LL * g * (k - x[i]) % Mod;
        s = (s + 1LL * y[i] * inv(1LL * (k - x[i]) * w[i] % Mod)) % Mod;
    }
    return (1LL * g * s % Mod + Mod) % Mod;
}
```

## 算法练习

### 习题一

洛谷p5437

### 题意

给定一个完全图，完全图中节点编号为  $1 \sim n$ ， $i, j$  节点间连一条边权为  $(i+j)^k$  的边。

随机选取  $n-1$  条边得到一棵生成树，问生成树的边权和的期望值。

### 题解

发现每条边地位都是等价的，于是每条边出现在生成树的概率为  $\frac{2(n-1)}{n(n-1)} = \frac{2}{n}$

于是不难得到答案为  $\frac{2}{n} \sum_{i=1}^{n-1} \sum_{j=i+1}^n (i+j)^k$  记

$f(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^n (i+j)^k$  考虑如何快速求出  $f(n)$

$$f(n) - f(n-1) = \sum_{i=1}^{n-1} \sum_{j=i+1}^n (i+j)^k - \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} (i+j)^k = (2n-1)^k + \sum_{i=1}^{n-2} (i+n) = \sum_{i=n+1}^{2n-1} i^k$$

于是有

$$f(n) = \begin{cases} 0, & n=1 \\ \sum_{i=2}^n \sum_{j=i+1}^{2i-1} j^k, & n \geq 2 \end{cases}$$

易知  $\sum_{j=i+1}^{2i-1} j^k$  是  $k+1$  次多项式，于是

$\sum_{i=2}^n \sum_{j=i+1}^{2i-1} j^k$  是  $k+2$  次多项式。

线性筛预处理  $1^k, 2^k, \dots, (k+3)^k$  于是可以线性时间求出  $f(1), f(2), \dots, f(k+3)$  最后考虑拉格朗日插值即可。

总时间复杂度  $O(k)$

```
const int MAXN=1e7+5,MAXV=2e7+10,Mod=998244353;
int quick_pow(int x,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*x%Mod;
        x=1LL*x*x%Mod;
        k>>=1;
    }
    return ans;
}
int prime[MAXV],kpow[MAXV],cnt;
void Prime(int k){
    kpow[1]=1;
    _for(i,2,MAXV){
```

```
    if(!kpow[i]) prime[cnt++]=i,kpow[i]=quick_pow(i,k);
    for(int j=0;j<cnt&& i*prime[j]<MAXV;j++){
        kpow[i*prime[j]]=1LL*kpow[i]*kpow[prime[j]]%Mod;
        if(i%prime[j]==0) break;
    }
}
int inv[MAXN];
void get_inv(){
    inv[1]=1;
    _for(i,2,MAXN)
        inv[i]=1LL*(Mod-Mod/i)*inv[Mod%i]%Mod;
}
int inv_frac[MAXN],y[MAXN],pre[MAXN],suf[MAXN],m;
int Lagrange(int x,int n){
    pre[0]=suf[n+1]=1;
    _rep(i,1,n)pre[i]=1LL*pre[i-1]*(x-i)%Mod;
    for(int i=n;i-->0)suf[i]=1LL*suf[i+1]*(x-i)%Mod;
    int ans=0,sign;
    _rep(i,1,n){
        sign=((n-i)&1)?-1:1;
        ans=(ans+1LL*sign*y[i]*pre[i-1]%Mod*suf[i+1]%Mod*inv_frac[i-1]%Mod*inv_frac[n-i])%Mod;
    }
    return ans;
}
int main()
{
    int n=read_int(),k=read_int();
    Prime(k);get_inv();
    inv_frac[0]=1,y[0]=0;
    _for(i,1,MAXN)inv_frac[i]=1LL*inv_frac[i-1]*inv[i]%Mod;
    _for(i,1,MAXV)kpow[i]=(kpow[i]+kpow[i-1])%Mod;
    _for(i,1,MAXN)y[i]=(y[i-1]+kpow[2*i-1]-kpow[i])%Mod;
    int ans=2LL*Lagrange(n,k+3)*quick_pow(n,Mod-2)%Mod;
    enter((ans+Mod)%Mod);
    return 0;
}
```

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

Permanent link: [https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:jxm2001:%E5%A4%9A%E9%A1%B9%E5%BC%8F\\_1&rev=1596465886](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E5%A4%9A%E9%A1%B9%E5%BC%8F_1&rev=1596465886)

Last update: 2020/08/03 22:44