

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

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

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

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;
}
```

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=1596448187 

Last update: **2020/08/03 17:49**