拉格朗日插值

一、引入

给出这样的问题：已知平面上 x 坐标互不相同的 $n+1$ 个点，求穿过这 $n+1$ 个点的多项式曲线；或者说已知次数不超过 n 的多项式函数的 $n+1$ 个函数值，求这个多项式函数。

这类问题显然可以使用高斯消元求解。具体地，设多项式函数为 $f(x)=a_nx^n+\cdots+a_1x+a_0$ 。多项式有 $n+1$ 个系数，我们将多项式的系数看作未知数，将每个已知点 (x_i,y_i) 代入 $y=f(x)$ 可以写出一个 $n+1$ 元一次方程。$n+1$ 个线性方程构成一个线性方程组，可用高斯消元求得 $n+1$ 个未知数（也就是多项式的系数）。时间复杂度 $O(n^3)$.

二、简介

现在我们介绍一种复杂度比高斯消元更优的算法——拉格朗日插值法。

我们可以运用拉格朗日插值法求得多项式，设 x 是自变量，(x_i,y_i) 是已知的点，那么函数表达式为：

$$f(x)=\sum_{i=0}^{n}y_i\prod_{i\neq j}\frac{x-x_j}{x_i-x_j}$$

拉格朗日插值有两种：**朴素法** 和 **重心拉格朗日插值法。**

时间复杂度

	求解多项式系数	计算指定自变量的函数值
高斯消元	$O(n^3)$	$O(n^3)$
拉格朗日插值的朴素法	$O(n^3)$	$O(n^2)$
重心拉格朗日插值法	$O(n^2)$	$O(n^2)$

由此可见拉格朗日插值法是比高斯消元在时间复杂度上更优越的算法。

三、拉格朗日插值法

1. 初探

我们从二次多项式入手，展开对拉格朗日插值法的探索。

已知该曲线经过下面三个点$(x_1,y_1),(x_2,y_2),(x_3,y_3)$，试找到一条穿过它们的多项式曲线。

设这是一个二次曲线 $y=a_0+a_1x+a_2x^2$ ，然后可以解方程。

$$\begin{cases}y_1=a_0+a_1x_1^1+a_2x_1^2\\y_2=a_0+a_1x_2^1+a_2x_2^2\\y_3=a_0+a_1x_3^1+a_2x_3^2\end{cases}$$

然而，如果不解方程呢？我们发现，可以构造三个二次函数，然后相加得到我们要的函数。具体方法如下：

构造二次函数 $f_1(x)$，使其在 $x=x_1$ 处函数值为 1, 在 $x=x_2,x_3$ 处函数值都为 0。

构造二次函数 $f_2(x)$，使其在 $x=x_2$ 处函数值为 1，其余两处值为 0。

构造二次函数 $f_3(x)$，使其在 $x=x_3$ 处函数值为 1，其余两处值为 0。

那么 $f(x)=y_1f_1(x)+y_2f_2(x)+y_3f_3(x)$ 满足在 x_1,x_2,x_3 处的取值分别是 y_1,y_2,y_3。 因为它过$(x_1,y_1),(x_2,y_2),(x_3,y_3)$，三点确定唯一的抛物线，所以这个函数就是唯一满足条件的答案。

2. 朴素法

假如有 $n+1$ 个点，形如 $(x_0,y_0),(x_1,y_1),\ldots,(x_n,y_n)$ ，假设这 $n+1$ 个点横坐标互不相同，从数学角度容易得出：过这 $n+1$ 个点的次数不超过 n 的多项式是唯一的。我们考虑用上文的方法合成目标多项式：

设

$$f_i(x)=\prod_{j=0,j\neq i}^{n}\frac{x-x_j}{x_i-x_j}$$

$f_i(x)$ 满足在 $x=x_i$ 处函数值为 1, 在 $x=x_j(i\neq j)$ 处值为 0。

那么多项式

$$f(x)=\sum_{i=0}^{n}y_if_i(x)=\sum_{i=0}^{n}y_i\prod_{j=0,j\neq i}^{n}\frac{x-x_j}{x_i-x_j}$$

是过点 $(x_0,y_0),(x_1,y_1),\ldots\ldots,(x_n,y_n)$ 的曲线，由唯一性可知，上式即为目标多项式。

使用朴素法可以解决关于多项式的问题，例如：

Ⅰ.计算指定自变量的函数值

我们直接把要求的 x 带入上面的式子中。时间复杂度 $O(n^2)$。

代码实现

```
//拉格朗日插值朴素法，求过n+1个点的一元n次方程，f(X)的值
ll Lagrange(ll X){
    ll ans = 0;
    for (int i = 0; i <= n; i++){
        ll s1 = 1, s2 = 1;
        for (int j = 0; j <= n; j++)
            if (i != j){
                s1 =  s1 * (X - x[j]) % mod;
                s2 =  s2 * (x[i] - x[j]) % mod;
            }
        ans = (ans + y[i] * s1 % mod * QuickPow(s2, mod - 2) % mod) % mod;
    }
    ans = (ans + mod) % mod;
    return ans;
}
```

Ⅱ.求解多项式系数

将上述式子展开，对于每个点(x_i,y_i)，统计对第 k 项 x^k 系数的贡献。

显然，这个式子计算连乘部分直接展开是 $O(n^2)$ 的，总时间复杂度 $O(n^3)$。

代码实现

```
//拉格朗日插值朴素法，求过n+1个点，一元n次方程，f(X)的各项系数，ans[i]表示x^i的系数。
void Lagrange(){
    for (int i = 0; i <= n; i++){
```

```
        ll s = 1;
        for (int j = 0; j <= n; j++) f[j] = 0;//初始化
        f[0] = 1;
        for (int j = 0; j <= n; j++)
            if (i != j){
                for (int k = j + (j < i); k >= 1; k--){
                    f[k] = (k == 0 ? 0 : f[k - 1]) - f[k] * x[j];//简单转移
                    f[k] = (f[k] % mod + mod) % mod;
                }
                s =  (s * (x[i] - x[j]) % mod + mod) % mod;
            }
        for (int j = 0; j <= n; j++){
            ans[j] = ans[j] + f[j] * y[i] % mod * Quick_Pow(s, mod - 2) % mod;
            ans[j] = (ans[j] % mod + mod) % mod;
        }
    }
}
```

3. 在 x 取值连续时的做法

在绝大多数题目中已知的 x_i 都是连续的（例如从 0 取到 n ），这样的话我们可以把上面的算法进一步优化。

首先把 x_i 换成 i，新的式子为

$$f(x)=\sum_{i=0}^{n}y_i\prod_{j=0,j\neq i}^{n}\frac{x-j}{i-j}$$

考虑如何快速计算 $\prod_{j=0,j\neq i}^{n}\frac{x-j}{i-j}$。对于分子来说，我们维护出关于 x 的**前缀积**和**后缀积**，也就是 $pre_i=\prod_{j=0}^{i}(x-j)$ ， $suf_i=\prod_{j=i}^{n}(x-j)$ 。 对于分母来说，可以发现这其实就是阶乘的形式。

那么式子就变成了

$$f(x)=\sum_{i=0}^{n}y_i\cdot\frac{pre_{i-1}\cdot suf_{i+1}}{i!(n-i)!(-1)^{n-i}}$$

重新分析复杂度：

1. 计算指定自变量的函数值：时间复杂度$O(n)$。
2. 求解多项式系数：该条件不足以优化时间复杂度，仍是 $O(n^3)$ 。

代码实现

```
//拉格朗日插值，求n+1个点且x为连续整数时，一元n次方程f(X)的值。
ll Lagrange(ll X){
    ll ans = 0;
    pre[0] = (X - x[0]) % mod;
    suf[n] = (X - x[n]) % mod;
    for (int i = 1; i <= n; i++)
        pre[i] = pre[i - 1] * (X - x[i]) % mod;
    for (int i = n - 1; i >= 0; i--)
        suf[i] = suf[i + 1] * (X - x[i]) % mod;
    ifac[0] = ifac[1] = 1;
```

```
    for (int i = 2; i <= n; i++)
        ifac[i] = (-mod / i * ifac[mod % i] % mod + mod) % mod; //线性求逆元
    for (int i = 2; i <= n; i++)
        ifac[i] = ifac[i] * ifac[i - 1] % mod; //阶乘
    for (int i = 0; i <= n; i++){
        ans = ans + y[i]
            * (i == 0 ? 1 : pre[i - 1]) % mod
            * (i == n ? 1 : suf[i + 1]) % mod
            * ifac[i] % mod * ifac[n - i] % mod
            * (((n - i) & 1) ? -1 : 1);
        ans = (ans % mod + mod) % mod;
    }
    return ans;
}
```

4. 重心拉格朗日插值法

拉格朗日插值法的公式结构整齐紧凑，在理论分析中十分方便，然而在计算中，当插值点增加或减少一个时，所对应的基本多项式就需要全部重新计算，于是整个公式都会变化，非常繁琐；并且求解多项式的时间复杂度始终遭遇瓶颈。为了进一步优化时间复杂度，引入重心拉格朗日插值法。

观察

$$\prod_{j=0,j\neq i}^{n}\frac{x-x_j}{x_i-x_j},$$

发现每一个连乘的分子乘上一个二项式 $x-x_i$ 能变成一个统一的多项式 $\prod_{j=0}^{n}(x-x_j)$。所以我们事先算出多项式 $g(x)=\prod_{j=0}^{n}(x-x_j)$ 的每项系数，处理的时候用 $x-x_i$ 去除 $g(x)$ 即可。

具体地，设

$$g(x)=\prod_{j=0}^{n}(x-x_j),\ \ t_i=\frac{1}{\prod_{j=0,j\neq i}^{n}(x_i-x_j)}$$

则：

$$f(x)=\sum_{i=0}^{n}y_it_i\cdot\frac{g(x)}{x-x_i}$$

求解关于 x 的多项式 $f(x)$ 的系数时，枚举每个 i，用 $x-x_i$ 去除整个多项式 $g(x)$，因为除式的项只有两项，可以 $O(n)$ 求出 $\frac{g(x)}{x-x_i}$，总时间复杂度 $O(n^2)$。

当插值点的个数增加一个时，将每个 t_i 都除以 x_i-x_{n+1}，将 $g(x)$ 乘以 $x-x_{n+1}$，最后增加新增点的贡献 $y_{n+1}t_{n+1}\cdot\frac{g(x)}{x-x_{n+1}}$ 即可，删除则反向操作即可。

单次修改的时间复杂度为 $O(n)$。求解 n 次多项式系数的时间复杂度为 $O(n^2)$。

代码实现