

# 拉格朗日插值

## 简介

在数值分析中，拉格朗日插值法是以法国18世纪数学家约瑟夫·拉格朗日命名的一种多项式插值方法。如果对实践中的某个物理量进行观测，在若干个不同的地方得到相应的观测值，拉格朗日插值法可以找到多项式，其恰好各个观测的点取到观测到的值。上面这样的多项式就称为拉格朗日（插值）多项式。

## 拉格朗日插值法

众所周知  $n + 1$  个  $x$  坐标不同的点可以确定唯一的最高为  $n$  次的多项式。在算法竞赛中，我们常常会碰到一类题目，题目中直接或间接的给出了  $n+1$  个点，让我们求由这些点构成的多项式在某一位的取值

一个最显然的思路就是直接高斯消元求出多项式的系数，但是这样做复杂度巨大  $(n^3)$  且根据算法实现不同往往会存在精度问题

而拉格朗日插值法可以在  $n^2$  的复杂度内完美解决上述问题

假设该多项式为  $f(x)$ ，第  $i$  个点的坐标为  $(x_i, y_i)$  我们需要找到该多项式在  $k$  点的取值

根据拉格朗日插值法

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

乍一看可能不是很好理解，我们来举个例子理解一下

假设给出的三个点为  $(1, 3)(2, 7)(3, 13)$

直接把  $f(k)$  展开

$$f(k) = 3 \frac{(k-2)(k-3)}{(1-2)(1-3)} + 7 \frac{(k-1)(k-2)}{(2-1)(2-3)} + 13 \frac{(k-1)(k-2)}{(3-1)(3-2)}$$

观察不难得到，如果我们把  $x_i$  带入的话，除第  $i$  项外的每一项的分子中都会有  $x_i - x_i$  这样其他的所有项就都被消去了

因此拉格朗日插值法的正确性是可以保证的

下面说一下拉格朗日插值法的拓展

在  $x$  取值连续时的做法

在绝大多数题目中我们需要用到的  $x_i$  的取值都是连续的，这样的话我们可以把上面的算法优化到  $O(n)$  复杂度

首先把  $x_i$  换成  $i$  新的式子为

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

考虑如何快速计算  $\prod_{i \neq j} \frac{k - j}{i - j}$

对于分子来说，我们维护出关于  $k$  的前缀积和后缀积，也就是

$$pre_i = \prod_{j=0}^i k - j$$

$$suf_i = \prod_{j=i}^n k - j$$

对于分母来说，观察发现这其实就是阶乘的形式，我们用  $fac[i]$  来表示  $i!$

那么式子就变成了

$$f(k) = \sum_{i=0}^n y_i \frac{pre_{i-1} * suf_{i+1}}{fac[i] * fac[N - i]}$$

**注意**:分母可能会出现符号问题，也就是说，当  $N - i$  为奇数时，分母应该取负号

### 重心拉格朗日插值法

再来看一下前面的式子

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

$$g = \prod_{i=0}^n k - x[i]$$

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

$$t_i = \frac{y_i}{\prod_{j \neq i} x_i - x_j}$$

$$f(k) = g \sum_{i=0}^n \frac{t_i}{(k - x[i])}$$

这样每次新加入一个点的时候只需要计算它的  $t_i$  即可

### 应用

首先讲一个经典应用：计算  $\sum_{i=1}^n i^k$  ( $n \leq 10^{15}$ ,  $k \leq 10^6$ )

老祖宗告诉我们，这个东西是个以  $n$  为自变量的  $k + 1$  次多项式，具体证明可以看第二份参考资料

然后直接带入  $k+1$  个点后用拉格朗日插值算即可，复杂度  $O(k)$

以上内容引自洛谷大佬博客：

<https://www.luogu.com.cn/blog/attack/solution-p4781>

下面是例题(洛谷P4781) <https://www.luogu.com.cn/problem/P4781>

## 题目背景

这是一道模板题

## 题目描述

由小学知识可知  $n$  个点  $(x_i, y_i)$  可以唯一地确定一个多项式  $y = f(x)$

现在，给定这  $n$  个点，请你确定这个多项式，并求出  $f(k) \bmod 998244353$  的值

## 输入格式

第一行两个整数  $n, k$

接下来  $n$  行，第  $i$  行两个整数  $x_i, y_i$

## 输出格式

一行一个整数，表示  $f(k) \bmod 998244353$  的值。

## 输入输出样例

输入 #1

```
3 100
1 4
2 9
3 16
```

输出 #1

```
10201
```

输入 #2

```
3 100
1 1
2 2
3 3
```

输出 #2

```
100
```

## 说明/提示

样例一中的多项式为  $f(x) = x^2 + 2x + 1$   $f(100) = 10201$

样例二中的多项式为  $f(x)=x^2 f(100) = 100^2$

$\sum_{1 \leq i < j \leq n} x_i x_j$  两两不同

## 题解

拉格朗日插值的公式大概是  $f(k) = \sum_{i=0}^n y_i \prod_{j \neq i} \frac{k - x_j}{x_i - x_j}$   $x_i, y_i$  是在  $x_i$  的取值。

```
#include <bits/stdc++.h>
#define int long long
using namespace std;
struct io {
    char buf[1 << 26 | 3], *s; int f;
    io() { f = 0, buf[fread(s = buf, 1, 1 << 26, stdin)] = '\n'; }
    io& operator >> (int&x) {
        for(x = f = 0; !isdigit(*s); ++s) f |= *s == '-';
        while(isdigit(*s)) x = x * 10 + (*s++ ^ 48);
        return x = f ? -x : x, *this;
    }
};

const int mod = 998244353;
int qpow(int x, int y) {
    int res = 1;
    for(; y; y >>= 1, x = x * x % mod)
        if(y & 1) res = res * x % mod;
    return res;
}
int inv(int x) { return qpow(x, mod - 2); }

int n, k;
const int maxn = 2e3 + 32;
int x[maxn], y[maxn];

#define out cout
signed main() {
#ifdef LOCAL
#define in cin
    ios :: sync_with_stdio(false), cin.tie(nullptr), cout.tie(nullptr);
    freopen("testdata.in", "r", stdin);
#else
    io in;
#endif
    in >> n >> k;
    for(int i = 1; i <= n; i++) in >> x[i] >> y[i];
    int ans = 0;
    for(int i = 1; i <= n; i++) {
```

```
int a, b; a = b = 1;
for(int j = 1 ; j <= n ; j ++ ) if(i ^ j) { a = a * (k - x[j]) % mod; }
for(int j = 1 ; j <= n ; j ++ ) if(i ^ j) { b = b * (x[i] - x[j]) % mod; }
}
a = (a + mod) % mod, b = (b + mod) % mod, b = inv(b);
ans = (ans + a * b % mod * y[i] % mod) % mod;
}
out << ans << '\n';
return 0;
}
```

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

Permanent link:  
<https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:manespace:%E6%8B%89%E6%A0%BC%E6%9C%97%E6%97%A5%E6%8F%92%E5%80%BC%E6%B3%95&rev=1589002995>

Last update: 2020/05/09 13:43