拉格朗日插值法及应用 - CSDN 博客

拉格朗日插值法

- 快速根据点值逼近函数
- 在取点大于n的情况下解出n次多项式是唯一解。

例如:
$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
只需取三个点$(1,1),(2,3),(3,6)$即可确定唯一方程。

同理$\sum_{i=1}^{n} i^2$只需由四个点确定，但可以取三个点来逼近。

作用?
当 n 很大时，直接带入求出的函数求解。

一般方法

对于$n+1$个点对$(x_0,y_0),(x_1,y_1)\dots(x_n,y_n)$，求一个函数$f_i$，使得该函数在$x_i$处取得对应$y_i$值，而在其他$x_j$取得$0$，最后把这几个函数线性结合即可。可得:

$$f_i(x)=\frac{\prod_{j\neq i}(x-x_j)}{\prod_{j\neq i}(x_i-x_j)}*y_i$$

$$g(x)=\sum_{i=0}^{n}f_i(x)$$

求原函数的复杂度为$O(n^2)$，单次求值的复杂度为$O(n^2)$。

重心拉格朗日插值法

考虑得到的函数:

$$f_i(x)=\frac{\prod_{j\neq i}(x-x_j)}{\prod_{j\neq i}(x_i-x_j)}*y_i$$

$$g(x)=\sum_{i=0}^{n}f_i(x)$$

不难发现每个f_i计算了重复部分。

设

$$l(x)=\prod_{i=1}^{n}(x-x_i)$$

对于每一个f_i
设:

$$\omega_i=\frac{y_i}{\prod_{j\neq i}(x_i-x_j)}$$

则:

$$g(x)=l(x)\sum_{i=1}^{n}\frac{\omega_i}{(x-x_i)}$$

求原函数复杂度为$O(n^2)$, 单次求值复杂度$O(n^2)$。

动态加点?
每次增加一个点，一般求法会重新计算一次达到$O(n^2)$的复

杂度，而重心拉格朗日插值法只需$O(n)$计算w_i即可。

横坐标连续?
不需多次计算w_i，单次求值$O(n)$，预处理逆元的话求原函数也只会$O(n)$。

应用

bzoj4559: 成绩比较

求

$$\prod_x \sum_{i=1}^{U_x} i^{R_x} (U_x - i)^{n-R_x-1} (U_x \le 10^9, R_x \le 10^2, n \le 10^2)$$

很显然是一个n次多项式，对于每一个x计算$n+1$个点值确定函数，然后带入U_x计算即可，因为x_i连续，所以时间复杂度$O(n^2)$。

bzoj2655: calc

听说结果是次数为$2n$的多项式，那么先 DP 出小的点，直接上拉格朗日就行了。

bzoj3453:XLkxc

求

$$f(n) = \sum_{i=0}^{n} \sum_{j=1}^{a+ib} \sum_{x=1}^{j} x^k (k \le 123, n, a, b \le 10^9)$$

虽然看上去很复杂，其实拆开并不复杂。
首先:

$$\sum_{x=1}^{j} x^k$$

是一个$k+1$次多项式（二项式定理展开后差分）。

其次:

$$g(m)=\sum_{j=1}^{m}\sum_{x=1}^{j}x^k$$

是一个$k+2$次多项式（同理）。

可以对$g(m)$进行插值求大的m数。

最后:

$$f(n)=\sum_{i=0}^{n}g(a+ib)$$

是一个$k+3$次多项式（同理）。
再对$f(n)$进行插值就好了。

而且得出了两个结论:
1. 每个求和号一般会增加多项式次数 1.
2. 插值可以嵌套。
特别是第二点，基本上可以在$O(k^2)$解决所有跟多项式有关的题。

Code for bzoj3453:

```
#include<bits/stdc++.h>
using namespace std;
struct IO
{
    streambuf *ib,*ob;
    int buf[50];
    inline void init()
    {
        ios::sync_with_stdio(false);
        cin.tie(NULL);cout.tie(NULL);
        ib=cin.rdbuf();ob=cout.rdbuf();
    }
    inline int read()
    {
        char ch=ib->sbumpc();int i=0,f=1;
        while(!isdigit(ch)){if(ch=='-')f=-1;ch=ib->sbum
        while(isdigit(ch)){i=(i<<1)+(i<<3)+ch-'0';ch=ib
        return i*f;
    }
    inline void W(int x)
```

```
    {
        if(!x){ob->sputc('0');return;}
        if(x<0){ob->sputc('-');x=-x;}
        while(x){buf[++buf[0]]=x%10,x/=10;}
        while(buf[0])ob->sputc(buf[buf[0]--]+'0');
    }
}io;
const int mod=1234567891,lim=130;
int k,a,n,d;
int fg[150],fy[150],wf[150],fac[150],inv[150];
inline int power(int a,int b)
{
    int res=1;
    for(;b;b>>=1)
    {
        if(b&1)res=1ll*res*a%mod;
        a=1ll*a*a%mod;
    }
    return res;
}
int DIV;
inline int calc(int *w,int u)
{
    if(u<=lim)return w[u];
    static int l,res,Div;
    l=1,res=0,Div=DIV;
    for(int i=1;i<=lim;i++)l=1ll*l*((1ll*u-i+mod)%mod)
    for(int i=1;i<=lim;i++)
    {
        res=(1ll*res+1ll*Div*power(((1ll*u-i+mod)%mod
        Div=1ll*Div*(i-lim+mod)%mod*inv[i]%mod;
    }
    return 1ll*res*l%mod;
}
int main()
{
    io.init();int T=io.read();
    fac[0]=fac[1]=inv[0]=inv[1]=1;DIV=1;
    for(int i=2;i<=lim;i++)inv[i]=(-1ll*(mod/i)*inv[m
    for(int i=1;i<=lim;i++)fac[i]=1ll*fac[i-1]*i%mod;
    for(int i=2;i<=lim;i++)DIV=1ll*DIV*power(1-i+mod,m
    while(T--)
    {
        k=io.read(),a=io.read(),n=io.read(),d=io.rea
        for(int i=1;i<=lim;i++){fg[i]=power(i,k);}
        for(int i=1;i<=lim;i++){fg[i]=(1ll*fg[i]+fg[i
        for(int i=1;i<=lim;i++){fg[i]=(1ll*fg[i]+fg[i
```

```
        fy[0]=calc(fg,a);
        for(int i=1;i<=lim;i++)
        {
            fy[i]=(1ll*fy[i-1]+calc(fg,(1ll*a+1ll*i*d
        }
        printf("%d\n",calc(fy,n));
    }
}
```

全文完

本文由 简悦 SimpRead 优化，用以提升阅读体验。