多项式全家桶.

这是(af)oier spinach退役前的笔记与解题记录,不能保证内容的严谨性,仅供参考

如果您发现文章存在可能的错误,请立刻联系我.

作者正在学文化课,不能保证及时修正,抱歉.

poly newton method

$$G(F(x)) = 0$$
$$G(F_0(x)) \equiv 0 \pmod{x^t} \quad G(F(x)) \equiv 0 \pmod{x^{2t}}$$
$$0 \equiv G(F(x)) \equiv G(F_0(x)) + G'(F_0(x))(F(x) - F_0(x)) \pmod{x^{2t}}$$
$$F(x) \equiv F_0(x) - \frac{G(F_0(x))}{G'(F_0(x))} \pmod{x^{2t}}$$

poly derivative&integral

$$A(x) = \sum_{i=0}^{n} a_i x^i$$
$$A'(x) = \sum_{i=0}^{n-1} (i+1) a_{i+1} x^i$$
$$\int A(x)dx = C + \sum_{i=1}^{n+1} \frac{a_{i-1}}{i} x^i$$

poly pow

$$F(x) = A(x)^k \quad ln\, F(x) = kln\, A(x) \quad F(x) = exp(kln\, A(x))$$

poly ln

$$F(x) = ln\, A(x) \quad F'(x) = \frac{A'(x)}{A(x)} \quad F(x) = \int \frac{A'(x)}{A(x)}$$

poly inv

$$A(x)F(x) \equiv 1 \pmod{x^{2t}} \Rightarrow A(x)F(x) \equiv 1 \pmod{x^t}$$
$$A(x)B(x) \equiv 1 \pmod{x^t}$$
$$A(x)(F(x) - B(x)) \equiv 0 \pmod{x^t} \quad F(x) - B(x) \equiv 0 \pmod{x^t}$$
$$F^2(x) - 2F(x)B(x) + B^2(x) \equiv 0 \pmod{x^{2t}}$$
$$A(x)(F^2(x) - 2F(x)B(x) + B^2(x)) \equiv 0 \pmod{x^{2t}}$$
$$F(x) - 2B(x) + A(x)B^2(x) \equiv 0 \pmod{x^{2t}}$$
$$F(x) \equiv B(x)(2 - A(x)B(x)) \pmod{x^{2t}}$$

poly exp

$$F(x) = exp(A(x)) \quad G(F(x)) = ln\, F(x) - A(x) = 0$$
$$G'(F(x)) = F^{-1}(x)$$
$$G(F(x)) = F_0(x) - \frac{G(F_0(x))}{G'(F_0(x))} = F_0(x) - F_0(x)(ln\, F_0(x) - A(x)) = F_0(x)(1 - ln\, F_0(x) + A(x))$$

poly sqrt

$$G(F(x)) = F^2(x) - A(x) = 0 \quad G'(F(x)) = 2F(x)$$
$$F(x) = F_0(x) - \frac{F_0^2(x) - A(x)}{2F_0(x)}$$
$$= \frac{F_0^2(x) + A(x)}{2F_0(x)} = \frac{1}{2}(F_0(x) + \frac{A(x)}{F_0(x)})$$

poly div

$$A(x)=\sum_{i=0}^{n}a_ix^i \quad B(x)=\sum_{i=0}^{m}b_ix^i$$
$$\Rightarrow Q(x) \quad R(x)$$
$$A(x)=B(x)Q(x)+R(x)$$
$$deg(A(x))=n \quad deg(B(x))=m$$
$$deg(Q(x))=n-m, deg(R(x))<m$$

$$F(x)=\sum_{i=0}^{n}a_ix^i, F^R(x)=\sum_{i=0}^{n}a_{n-i}x^i \quad F^R(x)=x^nF(\frac{1}{x}) \quad \text{就是反转系数向量}$$
$$A(x)=B(x)Q(x)+R(x)\Rightarrow A(\frac{1}{x})=B(\frac{1}{x})Q(\frac{1}{x})+R(\frac{1}{x})$$
$$x^nA(\frac{1}{x})=x^nB(\frac{1}{x})Q(\frac{1}{x})+x^nR(\frac{1}{x})$$
$$A^R(x)=(x^mB(\frac{1}{x})\cdot x^{n-m}Q(\frac{1}{x}))+x^{n-m+1}(x^{m-1}R(\frac{1}{x}))$$
$$A^R(x)=B^R(x)Q^R(x)+x^{n-m+1}R^R(x)$$

$$notice: \ deg(Q(x))=n-m\Rightarrow Q(x)=(Q(x) \mod x^{n-m+1})$$
$$A^R(x)=B^R(x)Q^R(x)+x^{n-m+1}R^R(x)$$
$$A^R(x)\equiv B^R(x)Q^R(x) \pmod{x^{n-m+1}}$$
$$Q^R(x)\equiv A^R(x)\frac{1}{B^R(x)} \pmod{x^{n-m+1}} \quad R(x)=A(x)-B(x)Q(x)$$

```
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cctype>
typedef long long Int;
const Int mod=998244353LL;
const Int G=3LL;
const int N=(1<<21);
Int qpow(Int a,Int p){
    Int r=1; a%=mod;
    while(p){
        if(p&1) r=r*a%mod;
        a=a*a%mod;
        p>>=1;
    }
```

```
    return r;
}
inline Int inv(Int x){ return qpow(x,mod-2); }
const int CutOff=20;
Int fftbuf[CutOff];
void fft(Int *A,int n,int f){
    Int base=qpow(G,(mod-1)/n),t=0,w=1;
    if(f<0) base=inv(base);
    if(n<CutOff){
        for(int i=0;i<n;i++){
            for(int j=n-1;j>=0;j--) t=(t*w+A[j])%mod;
            fftbuf[i]=t; t=0; w=w*base%mod;
        }
        for(int i=0;i<n;i++) A[i]=fftbuf[i];
        return ;
    }
    int m=n>>1,p=0; Int *A0=new Int[m],*A1=new Int[m];
    for(int i=0;i<m;i++){ A0[i]=A[p++]; A1[i]=A[p++]; }
    fft(A0,m,f); fft(A1,m,f);
    for(int i=0;i<m;i++){
        t=w*A1[i]%mod;
        A[i]=(A0[i]+t)%mod;
        A[i+m]=(A0[i]-t+mod)%mod;
        w=w*base%mod;
    }
    delete[] A0; delete[] A1;
}
inline void trans(Int *A,int n,int f){
    fft(A,n,f);
    if(f<0){
        Int x=inv(n);
        for(int i=0;i<n;i++) A[i]=A[i]*x%mod;
    }
}
inline void derivative(Int *A,int n,Int *B){
    for(int i=1;i<n;i++) B[i-1]=A[i]*i%mod;
    B[n]=0;
}
inline void integral(Int *A,int n,Int *B){
    for(int i=0;i<n;i++) B[i+1]=A[i]*inv(i+1)%mod;
    B[0]=0;
}
inline void clear(Int *A,int k){ for(int i=0;i<k;i++) A[i]=0; }
Int invbuf[N];
void polyinv(Int *A,int n,Int *B){
    if(n==1){ B[0]=inv(A[0]); return ; }
    polyinv(A,n>>1,B); int k=n<<1;
    for(int i=0;i<n;i++) invbuf[i]=A[i];
    trans(invbuf,k,1); trans(B,k,1);
    for(int i=0;i<k;i++){
        B[i]=B[i]*(2-invbuf[i]*B[i]%mod)%mod;
        B[i]=(B[i]+mod)%mod;
    } trans(B,k,-1);
```

```
    for(int i=n;i<k;i++) B[i]=0;
    clear(invbuf,k);
}
Int lnbuf[N];
void polyln(Int *A,int n,Int *B){
    int k=n<<1;
    derivative(A,n,lnbuf); polyinv(A,n,B);
    trans(B,k,1); trans(lnbuf,k,1);
    for(int i=0;i<k;i++) lnbuf[i]=B[i]*lnbuf[i]%mod;
    trans(lnbuf,k,-1); integral(lnbuf,n,B);
    for(int i=n;i<k;i++) B[i]=0;
    clear(lnbuf,k);
}
Int expbuf[N];
void polyexp(Int *A,int n,Int *B){
    if(n==1){ B[0]=1; return ; }
    int m=n>>1,k=n<<1;
    polyexp(A,m,B); polyln(B,n,expbuf);
    for(int i=0;i<n;i++)expbuf[i]=(A[i]-expbuf[i]+mod)%mod;
    expbuf[0]++;
    trans(B,k,1); trans(expbuf,k,1);
    for(int i=0;i<k;i++) B[i]=B[i]*expbuf[i]%mod;
    trans(B,k,-1);
    for(int i=n;i<k;i++) B[i]=0;
    clear(expbuf,k);
}

Int sqrtcpy[N],sqrtbuf[N];
void polysqrt(Int *A,int n,Int *B){
    if(n==1){ B[0]=1; return ; }
    int k=n<<1; polysqrt(A,n>>1,B);
    for(int i=0;i<n;i++) sqrtcpy[i]=A[i];
    polyinv(B,n,sqrtbuf);
    trans(B,k,1); trans(sqrtbuf,k,1); trans(sqrtcpy,k,1);
    for(int i=0,x=inv(2);i<k;i++)
        B[i]=(B[i]+sqrtcpy[i]*sqrtbuf[i]%mod)%mod*x%mod;
    trans(B,k,-1); for(int i=n;i<k;i++) B[i]=0;
    clear(sqrtcpy,k); clear(sqrtbuf,k);
}

namespace polydiv{
    Int ra[N],rb[N],buf[N];
    void polydiv(Int *A,int n,Int *B,int m,Int *Q,Int *R){
        for(int i=0;i<=n;i++) ra[i]=A[n-i];
        for(int i=0;i<=m;i++) rb[i]=B[m-i];
        int k=1; while(k<n-m+1) k<<=1;
        polyinv(rb,k,buf);
        for(int i=n-m+1;i<k;i++) buf[i]=0;
        while(k<n*2) k<<=1;
        trans(buf,k,1); trans(ra,k,1);
        for(int i=0;i<k;i++) buf[i]=buf[i]*ra[i]%mod;
        trans(buf,k,-1);
        for(int i=0;i<=n-m;i++) Q[i]=buf[n-m-i];
```

```
        for(int i=0;i<=m;i++) buf[i]=B[i];
        for(int i=m+1;i<k;i++) buf[i]=0;
        trans(Q,k,1); trans(buf,k,1);
        for(int i=0;i<k;i++) buf[i]=buf[i]*Q[i]%mod;
        trans(Q,k,-1); trans(buf,k,-1);
        for(int i=0;i<m;i++) R[i]=(A[i]-buf[i]+mod)%mod;
        clear(ra,k); clear(rb,k); clear(buf,k);
    }
}




int read(){
    int x=0;char c;
    do{c=getchar();}while(!isdigit(c));
    do{ x=x*10+c-'0'; c=getchar(); }while(isdigit(c));
    return x;
}
Int readmod(const Int mod){
    Int x=0;char c;
    do{c=getchar();}while(!isdigit(c));
    do{
        x=x*10+c-'0';x=(x%mod+mod)%mod;
        c=getchar();
    }while(isdigit(c));
    return x;
}
int main(){ return 0; }

```

技巧:自卷积递推式的分治+FFT

$$
\begin{gathered}
given\, g, f_0 \\
g[1..n) \to f[0..n) \quad just\, let\, g[0]=0 \\
f_m = \sum_{i=1}^{m} g_i f_{m-i} \quad f_m + = \sum_{i=0}^{m} g_i f_{m-i} \\
[l, mid) \Rightarrow [mid+1, r) \\
f_{mid+k} + = \sum_{i=l}^{mid-1} f_i g_{mid+k-i} = \sum_{i=0}^{mid-l-1} f_{l+i} g_{mid+k-(l+i)} \\
A[x] = f[l+x][l, mid), B[\dots] = g[0, r-l)
\end{gathered}
$$

> (最好先扩展到2^k长度)

> solve[l,r)
>
> ```
> solve[l,mid)
>
>     左边对于右边的贡献是卷积形式.
>
>     FFT算出来.加上去.
>
> solve[mid,r]计算.
> ```

也可以推式子求逆.

$$F(x)=\sum_{0\le n} f_n x^n \quad G(n)=\sum_{0\le n} g_n x^n$$

$$g_0=0, given\ f_0=A$$

$$F(x)G(x)=\sum_{0\le n} x^n\sum_{i=0}^{n} f_i g_{n-i}=(f_0g_0)x^0+\sum_{1\le n} x^n\sum_{i=0}^{n} f_{n-i}g_i$$

$$0+\sum_{1\le n} x^n(f_ng_0+\sum_{i=1}^{n} f_{n-i}g_i)=\sum_{1\le n} f_nx^n=F(x)-f_0$$

$$F(x)G(x)=F(x)-f_0$$

$$f_0=F(x)(1-G(x))$$

$$F(x)=\frac{f_0}{1-G(x)}=\frac{A}{1-G(x)}$$

习题选讲

集合划分问题.

无序集合划分 与 有序集合划分(也许应该叫集合有序划分/无序划分...)

参考资料表

- https://www.cnblogs.com/joyouth/p/5600541.html 第二类斯特林数与贝尔数以及有序划分数

- https://github.com/KingSann/useless-papers 多项式基础
- http://www.cnblogs.com/onioncyc/p/8722262.html 一个讲解详细的blog
- https://rqy.moe/Algorithms/generating-function/ rqy blog,有讲解bell数egf的推导

1. 前者(正常的划分数)为bell数B_n,枚举$1 \in T$的$|T|$,可以得到

$$B_n = \sum_{i=1}^{n} \binom{n-1}{i-1} B_{n-i}$$

枚举了包含1的集合,剩余的部分按照相对大小关系编号为[1..n-i]使用bell数递归下去划分.

显然两部分不相交,于是得到了把n元素集合,划分成一个集合簇的方案数即为bell数.

考虑计算B_n,这是一个类似于自卷积的式子,我们考虑使用分治来处理

$$B_n = \sum_{i=1}^{n} \binom{n-1}{i-1} B_{n-i} = \sum_{i=0}^{n-1} \binom{n-1}{i} B_{n-i-1}$$

$$B_n = (n-1)! \sum_{i=0}^{n-1} \frac{1}{i!} \frac{B_{n-i-1}}{(n-i-1)!}$$

我们进行这个过程,分治,考虑左侧对右侧的影响,递归处理.

$$\begin{aligned}
&solve[l,r) \\
&\quad solve[l,mid) \\
&\quad\quad B_{mid+k} += (mid+k-1)! \sum_{l \le i < mid} \frac{B_i}{i!} \frac{1}{(mid+k-i-1)!} \\
&\quad\quad F = B[l,mid), G = \frac{1}{i!}[0, r-l) \\
&\quad solve[mid,r)
\end{aligned}$$

推式子找求逆?不好意思这次失败了,和无向连通图计数的问题相比较,发现那个问题中,有两个多项式都是完全已知的,而求bell数并没有这个利好.

$$B_n = \sum_{i=0}^{n-1} \binom{n-1}{i} B_{n-i-1}$$

$$\frac{B_n}{(n-1)!} = \sum_{i=0}^{n-1} \frac{1}{i!} \frac{B_{n-i-1}}{(n-i-1)!}$$

$$F(x) = \sum_{1 \le n} \frac{B_n}{(n-1)!} x^n \quad G(x) = \sum_{0 \le n} \frac{x^n}{n!} \quad H(x) = \sum_{0 \le n} \frac{B_n}{n!} x^n$$

$$G(x)H(x) = \sum_{0 \le n} x^n \sum_{i=0}^{n} \frac{1}{i!} \frac{B_{n-i}}{(n-i)!} = \sum_{0 \le n} x^n \frac{B_{n+1}}{n!} = \sum_{1 \le n} x^n \frac{B_n}{(n-1)!} = F(x)$$

但是利用多项式exp这个好东西,我们可以这样玩.但是我目前对于EGF的套路还不熟悉,暂且不做深入.

> 我们考虑每个集合是由若干个元素组成的且集合非空，集合中元素无序 则可以得到集合的生成函数为g(x)=e^x-1 而贝尔数分解成若干个集合的划分方案,则我们有f=e^g 我们对e^x-1做泰勒展开，之后多项式求exp即可

> > The exponential generating function of the Bell numbers is
>
> $$B(x)=\sum_{0\le n}\frac{B_n}{n!}x^n=e^{e^x-1}$$
> $$f(x)=e^x-1 \quad B(x)=exp(f)$$

---

2. 后者(集合有序划分计数)我没有查询到,我们称为Q_n,枚举最终方案中,排在最前面的集合,可以得到.

$$Q_n=\sum_{i=1}^{n}\binom{n}{i}Q_{n-i}$$

虽然两部分其实是有相交的,

但是显然一个划分集合的排列,是不会被多次枚举到的.

于是我们得到了有序集合划分数即Q_n

考虑快速计算,首先肯定是分治.然后我们考虑更好的做法. **注意下标起始位置!**

如果你问我1-H常数项是0模意义下没有逆元怎么求逆的话请看粗体/黑题字的说明.

$$Q_n=\sum_{i=1}^{n}\binom{n}{i}Q_{n-i}$$
$$\frac{Q_n}{n!}=\sum_{i=1}^{n}\frac{1}{i!}\frac{Q_{n-i}}{(n-i)!}$$
$$F(x)=\sum_{0\le n}\frac{Q_n}{n!}x^n \quad H(x)=\sum_{1\le i}\frac{x^i}{i!}$$
$$F(x)H(x)=\sum_{1\le n}x^n\sum_{i=1}^{n}\frac{1}{i!}\frac{Q_{n-i}}{(n-i)!}=\sum_{1\le n}x^n\frac{Q_n}{n!}=F(x)-\frac{Q_0}{0!}=F(x)-1$$
$$F(x)(1-H(x))=1 \quad F(x)=(1-H(x))^{-1}$$

就一个求逆过去没了吧.

然后考虑几个例题

[luogu P5162]

[HE/TJ OI2016 求和]

第一个题.考虑直接按照期望是平均值来计算.

f_n,g_n分别表示有序集合划分数,以及所有有序划分方案中集合的数量之和.

第一个是刚刚讲完的集合有序划分数,考虑第二个,枚举个集合被计算的次数.然后得到.

$$g_n=f_n+\sum_{i=1}^{n}\binom{n}{i}g_{n-i}=\sum_{i=1}^{n}\binom{n}{i}(f_{n-i}+g_{n-i})$$
$$g_0=0,g_1=1$$

组合意义:第一个集合,有f_n个方案中有第一个集合,贡献为f_n,枚举第一个集合大小为i剩余的部分划分方案中涉及了g_{n-i}个集合,每一个都可以和前面第一个集合进行拼接操作,故贡献为$\binom{n}{i}g_{n-i}$,后面那个式子是我们计算用的.

考虑计算g,首先直接一个分治$O(nlog^2n)$打过去就可以AC了,然后考虑能不能更优秀一点.(就其实是我比较懒而且分治比较容易写挂23333)

注意一下下标的问题,下标起始位置非常重要.

$$\frac{g_n}{n!}=\sum_{i=1}^{n}\frac{1}{i!}(\frac{g_{n-i}}{(n-i)!}+\frac{f_{n-i}}{(n-i)!})$$
$$F(x)=EGF(f)\quad G(x)=EGF(g)\quad H(x)=\sum_{1\le n}\frac{x^n}{n!}$$
$$(F(x)+G(x))H(x)=\sum_{0\le n}x^n\sum_{i=1}^{n}\frac{1}{i!}\frac{f_{n-i}+g_{n-i}}{(n-1)!}=G(x)$$
$$FH+HG=G\quad G(1-H)=FH$$
$$F=(1-H)^{-1}$$
$$G=\frac{FH}{1-H}=F^2H$$
$$F\frac{H}{1-H}=F(\frac{1}{1-H}-1)=F(F-1)$$

然后就写一个求逆就没了....感觉很可做吧.

然后是可爱的斯特林数(这里一般只玩第二类斯特林数).

注意 第二类斯特林数计算中,由于组合意义(空集划分方案是唯一的),我们必须定义$0^0=1$,有$S_0^0=1$

$S_n^k=\left\{\begin{matrix}n\\m\end{matrix}\right\}$表示n元素集合划分为k个无序不相交集合的方案数.

$S_n^n=1,S_n^0=[n=0],S_0^1=S_1^0=0$

$B_n=\sum_{i=1}^n S_n^i$

$S_n^k=S_{n-1}^{k-1}+kS_{n-1}^k$

计算的话我们考虑计算一行,容斥一下可以得到(这里一下都是抄的还没有理解)

$$\left\{\begin{matrix}n\\m\end{matrix}\right\}=\frac{1}{m!}\sum_{i=0}^{m}(-1)^i*\binom{m}{i}*(m-i)^n$$

你看这个卷积式,美滋滋,我们一个FFT过去就可以求出一行啦(n元素,划分为[1..n]的方案)

(第二类)斯特林数在OI中的应用主要是进行下降幂的转换,有如下公式.

$$m^n=\sum_{i=1}^{n}\left\{\begin{matrix}n\\i\end{matrix}\right\}*i!*\binom{m}{i}$$
$$m^n=\sum_{i=1}^{n}\left\{\begin{matrix}n\\i\end{matrix}\right\}*m^{\underline{i}}$$

看一个直接套通项公式的水题.HEOI/TJOI 2016求和.

$$S_n^m = \frac{1}{m!}\sum_{i=0}^{m}(-1)^i\binom{m}{i}(m-i)^n$$

$$= \frac{1}{m!}\sum_{i=0}^{m}(-1)^i\frac{m!}{(m-i)!i!}(m-i)^n = \sum_{i=0}^{m}\frac{(-1)^i}{i!}\frac{(m-i)^n}{(m-i)!}$$

$$A[x] = \frac{(-1)^x}{x!} \quad B[x] = \frac{x^n}{x!} \quad S_n^m = (A*B)[m]$$

$$ans = \sum_{i=0}^{n}\sum_{j=0}^{i}S_i^j 2^j j! = \sum_{i=0}^{n}\sum_{j=0}^{n}S_i^j 2^j j! = \sum_{j=0}^{n}2^j j!\sum_{i=0}^{n}S_i^j$$

$$\sum_{j=0}^{n}2^j j!\sum_{i=0}^{n}\sum_{k=0}^{j}\frac{(-1)^k}{k!}\frac{(j-k)^i}{(j-k)!}$$

$$\sum_{j=0}^{n}2^j j!\sum_{k=0}^{j}\sum_{i=0}^{n}\frac{(-1)^k}{k!}\frac{(j-k)^i}{(j-k)!}$$

$$\sum_{j=0}^{n}2^j j!\sum_{k=0}^{j}\frac{(-1)^k}{k!}\frac{1}{(j-k)!}\sum_{i=0}^{n}(j-k)^i$$

$$f[x] = \frac{(-1)^x}{x!} \quad g[x] = \frac{\sum_{i=0}^{n}x^i}{x!} \quad Ans = \sum_{j=0}^{n}2^j j! \times (f*g)[j]$$

$$\sum_{i=0}^{n-1}q^i = \frac{q^n-1}{q-1}$$

注意 (第二类)斯特林数 的计算中,由于组合意义的存在,我们需要定义$0^0=1=S(0,0)$

```

#include <bits/stdc++.h>
using namespace std;
typedef long long Int;
const Int mod=998244353LL;
const Int G=3LL;
const int N=(1<<20);

Int qpow(Int a,Int b){
    Int r=1; a=(a%mod+mod)%mod;
    while(b){
        if(b&1) r=r*a%mod;
        a=a*a%mod; b>>=1;
    }
```

```
    return r;
}
inline Int inv(Int x){ return qpow(x,mod-2); }
const int CutOff=20;
Int buf[CutOff];
void fft(Int *A,int n,int f){
    Int base=qpow(G,(mod-1)/n),t=0,w=1;
    if(f<0) base=inv(base);
    if(n<CutOff){
        for(int i=0;i<n;i++){
            for(int j=n-1;j>=0;j--) t=(t*w+A[j])%mod;
            buf[i]=t; t=0; w=w*base%mod;
        }
        for(int i=0;i<n;i++) A[i]=buf[i];
        return ;
    }
    int m=n>>1,p=0;Int *A0=new Int[m],*A1=new Int[m];
    for(int i=0;i<m;i++){ A0[i]=A[p++]; A1[i]=A[p++]; }
    fft(A0,m,f); fft(A1,m,f);
    for(int i=0;i<m;i++){
        t=A1[i]*w%mod;
        A[i]=(A0[i]+t)%mod;
        A[i+m]=(A0[i]-t+mod)%mod;
        w=w*base%mod;
    }
    delete[] A0; delete[] A1;
}
void trans(Int *A,int n,int f){
    fft(A,n,f);
    if(f<0) for(int i=0,x=inv(n);i<n;i++)
        A[i]=A[i]*x%mod;
}
int n,k=1;
Int f[N],g[N],h[N],fac[N],ifac[N],iv[N];
inline Int qsum(int q,int n){
    if(q==0) return 1;
    if(q==1) return n+1;
    return (qpow(q,n+1)-1)*iv[q-1]%mod;
}
int main(){
    cin>>n; iv[1]=1;
    for(int i=2;i<N;i++)
        iv[i]=(mod-(mod/i)*iv[mod%i]%mod)%mod;
    fac[0]=ifac[0]=1;
    for(int i=1;i<=n;i++){
        fac[i]=fac[i-1]*i%mod;
        ifac[i]=ifac[i-1]*iv[i]%mod;
    }

    for(int i=0;i<=n;i++){
        f[i]=qpow(-1,i)*ifac[i]%mod;
        g[i]=qsum(i,n)*ifac[i]%mod;
    }
```

```
    
    while(k<2*n) k<<=1;
    trans(f,k,1); trans(g,k,1);
    for(int i=0;i<k;i++) h[i]=f[i]*g[i]%mod;
    trans(h,k,-1);
    Int ans=0,tmp=0;
    for(Int i=0,pw=1;i<=n;i++,pw=pw*2%mod){
        tmp=pw*fac[i]%mod*h[i]%mod;
        ans=(ans+tmp)%mod;
    }
    cout<<ans<<endl;
    return 0;
}
```

一些快速求和问题.

背包计数.

LOJ 6268

$$f_{n,v}=\sum_{i=0}^{\lfloor\frac{v}{a_n}\rfloor}f_{n-1,v-i*a_n}$$

$$F_k(x)=\sum f_{k,i}x^i\quad G_j(x)=\sum x^{ia_j}$$

$$F_k(x)=F_{k-1}(x)G_k(x)\quad F_n(x)=\prod_{i=1}^{n}G_i(x)$$

$$F(x)=\prod_{i=1}^{n}G_i(x)=exp(ln\prod_{i=1}^{n}G_i(x))=exp(\sum_{i=1}^{n}ln\,G_i(x))$$

$$A_k(x)=\sum_{i=0}^{\infty}x^{ki}=\frac{1}{1-x^k}$$

$$ln\,A_k(x)=\int\frac{A_k'(x)}{A_k(x)}dx=\int(1-x^k)\sum_{i=1}^{\infty}ik\,x^{ik-1}=\int\sum_{i=1}^{\infty}ik\,x^{ik-1}-\sum_{i=1}^{\infty}ik\,x^{(i+1)k-1}$$

$$=\int\sum_{i=1}^{\infty}ik\,x^{ik-1}-\sum_{i=2}^{\infty}(i-1)kx^{ik-1}=\int kx^{k-1}+\sum_{i=2}^{\infty}k\,x^{ik-1}=\int\sum_{i=1}^{\infty}kx^{ik-1}=\sum_{i=1}^{\infty}\frac{k}{ik}x^{ik}=\sum_{i=1}^{\infty}\frac{x^{ik}}{i}$$

如果k即a_i不重复,那么可以$\sum_{i=1}^{\infty}\frac{n}{i}=O(nlogn)$的时间内求出$\sum ln\,G_i(x)$,最后用一个$exp$在$O(nlogn)$内求出答案.

一个题.

~~我也不知道这个题在哪里可以提交,不过和下面一个题基本一致~~

$$\sum_{i=0}^{n}\sum_{j=0}^{n}a_i b_j\ j^i$$

多项式多点求值,$O(nlog^2n)$跑得龟速.

$$\sum_{j=0}^{n}b_j\sum_{i=0}^{n}a_i j^i=\sum_{j=0}^{n}b_jA(j)$$

$$A(x)=\sum_{i=0}^{n}a_ix^i$$

下面这个仍然龟速2333,不过至少比多点求值好很多.

$$\sum_{i=0}^{n}a_i\sum_{j=0}^{n}b_jj^i=\sum_{i=0}^{n}a_i\sum_{j=0}^{n}[x^i]B_j(x)$$

$$B_j(x)=\frac{b_j}{1-jx}=\sum_{i=0}^{\infty}b_jj^i\ x^i$$

$$\sum_{k=0}^{n}B_k(x)=\sum_{k=0}^{n}\frac{b_k}{1-kx}$$

$$\sum_{k=0}^{n}B_k(x)=\sum_{k=0}^{n}\frac{b_k\prod_{i\neq k}(1-ix)}{\prod_{i=0}^{n}(1-ix)}$$

$$F(x)=\prod_{i=0}^{n}(1-ix)$$

这里怎么做呢?首先分母不用看了.分治打过去加个求逆就好了.你也可以跟着递归过程计算.

我们考虑**分治逼近**类似的产物来解决分子.首先考虑叶子$[l,l+1)$这里分子为b_l.这是递归基.

考虑分别计算$[l,mid)\quad[mid,r)$之后,如何合并.$[l,mid)$需要卷积上$\prod_{i\in[mid,r)}(1-ix)$.而$[mid,r)$卷积上$\prod_{i\in[l,mid)}(1-ix)$.最后一个加法.

这样$O((r-l)\,log(r-l))$ 的时间完成了合并.发现合并复杂度仅仅和区间长度有关系.这个分治是有效的了.我们来计算时间复杂度.

$$T(n)=\sum_{i=0}^{log_2n}\frac{n}{2^i}\ i2^i=n\sum_{i=0}^{log_2n}=O(nlog^2n)$$

$$T(n)=2T(\frac{n}{2})+O(nlogn)=O(nlog^2n)$$

又一个经典题.

[luogu 4705]

[LOJ2409:thupc 2017 sum]

这里的问题是等幂和.

$$F(x)=\sum_{0\le k}x^k\sum_{i=1}^{n}a_i^k$$

我直接贴出我对于具体题目的推导.

$$\forall k\le t\qquad f_k=\sum_{i=1}^{n}\sum_{j=1}^{m}(a_i+b_j)^k$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{p=0}^{k}\binom{k}{p}a_i^pb_j^{k-p}=\sum_{p=0}^{k}\binom{k}{p}\sum_{i=1}^{n}a_i^p\sum_{j=1}^{m}b_j^{k-p}=\sum_{p=0}^{k}\binom{k}{p}\sum_{i=1}^{n}a_i^p\sum_{j=1}^{m}b_j^{k-p}=\sum_{p=0}^{k}\binom{k}{p}A(p)B(k-p)=k!\sum_{p=0}^{k}\frac{A(p)}{p!}\frac{B(k-p)}{(k-p)!}$$

拆分后发现是卷积形式.现在只要快速求出A,B即可得到答案.我们发现这是一个"等幂和". 阶乘先扔掉,可以最后处理,式子对称性很强,只考虑A,则B可以同理计算出来.

$$A(p)=\frac{1}{p!}\sum_{i=1}^{n}a_i^p\quad B(p)=\frac{1}{p!}\sum_{i=1}^{m}b_i^p$$

$$A(x)=\sum_{0\le m}x^m\sum_{i=1}^{n}a_i^m=\sum_{i=1}^{n}F_i(x)$$

$$F_k(x)=\frac{1}{1-a_kx}=\sum_{0\le n}a_k^n\,x^n$$

$$G_k(x)=\frac{d\,ln(1-a_kx)}{d\,x}=\frac{-a_k}{1-a_kx}=-a_k\sum_{0\le n}a_k^nx^n=\sum_{0\le n}a_k^{n+1}x^n=\sum_{1\le n}a_k^nx^{n-1}$$

$$F_k(x)=-xG_k(x)+1=-x(ln(1-a_kx))'+1$$

$$A(x)=\sum_{i=1}^{n}F_i(x)=n-x\sum_{i=1}^{n}G_k(x)=n-x\sum_{i=1}^{n}(ln(1-a_ix))'=n-x(\sum_{i=1}^{n}ln(1-a_ix))'=n-x(ln\prod_{i=1}^{n}(1-a_ix))'$$

推到这里终于可以快速计算啦.

一个分治求出$\prod_{i=1}^{n}(1-a_ix)$之后多项式ln再补上系数前的阶乘逆元,即可得到A.最后一个卷积上去就没了.

详细推导如下。

$$nm\,Ans(k)=\sum_{x=1}^{n}\sum_{y=1}^{m}(a_x+b_y)^k=\sum_{x=1}^{n}\sum_{y=1}^{m}\sum_{i=0}^{k}\binom{k}{i}a_x^i b_y^{k-i}$$

$$\frac{nm}{k!}Ans(k)=\sum_{x=1}^{n}\sum_{y=1}^{m}\sum_{i=0}^{k}\frac{a_x^i}{i!}\frac{b_y^{k-i}}{(k-i)!}$$

$$\sum_{i=0}^{k}\frac{\sum_{x=1}^{n}a_x^i}{i!}\frac{\sum_{y=1}^{m}a_y^{k-i}}{(k-i)!}$$

$$A(x)=\sum_{0\le k}x^k\sum_{i=1}^{\infty}a_i^k=\sum_{i=1}^{n}\frac{1}{1-a_ix}$$

$$\frac{1}{1-a_ix}=\sum_{0\le k}x^k\,a_i^k$$

$$\frac{d\,ln(F(x))}{dx}=\frac{F'(x)}{F(x)}$$

$$\frac{d\,ln(1-a_ix)}{dx}=\frac{-a_i}{1-a_ix}=-a_i\sum_{0\le k}x^k\,a_i^k=-\sum_{0\le k}x^k\,a_i^{k+1}$$

$$\frac{1}{1-a_ix}=-\frac{d\,ln(1-a_ix)}{dx}x+1$$

$$A(x)=\sum_{0\le k}x^k\sum_{i=1}^{n}a_i^k=\sum_{i=1}^{n}\frac{1}{1-a_ix}=\sum_{i=1}^{n}(-\frac{d\,ln(1-a_ix)}{dx}x+1)=n-\frac{d(\sum_{i=1}^{n}ln(1-a_ix))}{dx}x$$

$$=n-\frac{d\,ln(\prod_{i=1}^{n}(1-a_ix))}{dx}x$$

扔一个优化超多...用了yyb的NTT+xehoth的IO的代码.

```
#pragma GCC optimize(2)
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC optimize("inline")
#include <bits/stdc++.h>
using namespace std;
typedef long long Int;
const int N=(1<<20);
const Int mod=998244353LL;
const Int G=3LL;
Int invG=0,inv[N],fac[N],ifac[N];
inline Int fix(Int x){ return (x%mod+mod)%mod; }
Int qpow(Int a,Int p){
```

```
    Int r=1; a=(a%mod+mod)%mod;
    while(p){
        if(p&1) r=r*a%mod;
        a=a*a%mod; p>>=1;
    }
    return r;
}
namespace fft{
#define RG
    int r[N]; int Og[N];
    void NTT(Int *P,int N,int opt){
        for(RG int i=0;i<N;++i)if(i<r[i])swap(P[i],P[r[i]]);
        for(RG int i=1;i<N;i<<=1){
            RG int W=qpow(G,(mod-1)/(i<<1));Og[0]=1;
            for(RG int j=1;j<i;++j)Og[j]=1ll*Og[j-1]*W%mod;
            for(RG int p=i<<1,j=0;j<N;j+=p)
                for(RG int k=0;k<i;++k){
                    RG int X=P[j+k],Y=1ll*Og[k]*P[i+j+k]%mod;
                    P[j+k]=(X+Y)%mod;P[i+j+k]=(X+mod-Y)%mod;
                }
        }
        if(opt==-1){
            reverse(&P[1],&P[N]);
            for(RG int i=0,inv=qpow(N,mod-2);i<N;++i)P[i]=1ll*P[i]*inv%mod;
        }
    }
    inline void initrev(int N){
        int l=0; while((1<<l)!=N) l++;
        for(RG int i=0;i<N;++i)r[i]=(r[i>>1]>>1)|((i&1)<<(l-1));
    }
    inline void trans(Int *A,int n,int f){ NTT(A,n,f); }
}
using fft::trans;
using fft::initrev;


void clear(Int *A,int n){ for(int i=0;i<n;i++) A[i]=0; }
void cut(Int *A,int n,int k){ for(int i=n;i<k;i++) A[i]=0; }

void deri(Int *A,int n,Int *B){
    for(int i=1;i<n;i++) B[i-1]=A[i]*i%mod;
    B[n]=0;
}
void integral(Int *A,int n,Int *B){
    for(int i=0;i<n;i++) B[i+1]=A[i]*inv[i+1]%mod;
    B[0]=0;
}

namespace PolyInv{
    Int buf[N];
    void polyinv(Int *A,int n,Int *B){
        if(n==1){ B[0]=qpow(A[0],mod-2); return ; }
        polyinv(A,n>>1,B); for(int i=0;i<n;i++) buf[i]=A[i];
```

```
        int k=n<<1; initrev(k);
        trans(B,k,1); trans(buf,k,1);
        for(int i=0;i<k;i++) B[i]=fix(B[i]*(2-buf[i]*B[i]%mod+mod));
        trans(B,k,-1); clear(buf,k); cut(B,n,k);
    }
}
using PolyInv::polyinv;

namespace PolyLn{
    Int buf[N];
    void polyln(Int *A,int n,Int *B){
        polyinv(A,n,B); deri(A,n,buf);
        int k=n<<1; initrev(k);
        trans(B,k,1); trans(buf,k,1);
        for(int i=0;i<k;i++) buf[i]=buf[i]*B[i]%mod;
        trans(buf,k,-1); integral(buf,n,B);
        clear(buf,k); cut(B,n,k);
    }
}
using PolyLn::polyln;

int n,m,t,a[N],b[N];
Int A[N],B[N],tmp[N];
// B=\prod_{i \in [l,r) } (1-a_i x)
void solve(int *a,int l,int r,Int *B){
    if(l>=r) return ;
    if(r-l==1){ B[0]=1; B[1]=(mod-a[l])%mod; return ; }
    int mid=(l+r)>>1;
    int k=1; while(k<r-l) k<<=1; k<<=1;
    Int *L=new Int[k],*R=new Int[k]; clear(L,k); clear(R,k);
    solve(a,l,mid,L); solve(a,mid,r,R);
    initrev(k);
    trans(L,k,1); trans(R,k,1);
    for(int i=0;i<k;i++) B[i]=L[i]*R[i]%mod;
    trans(B,k,-1);
    delete[] L; delete[] R;
}

namespace IO{

    inline char read() {
        static const int IN_LEN = 1 << 20 | 1;
        static char buf[IN_LEN], *s, *t;
        return (s == t) && (t = (s = buf) + fread(buf, 1, IN_LEN, stdin)),
                        s == t ? -1 : *s++;
    }

    const int OUT_LEN = 1 << 21 | 1;

    char obuf[OUT_LEN], *oh = obuf;

    inline void print(char c) {
        (oh == obuf + OUT_LEN) && (fwrite(obuf, 1, OUT_LEN, stdout), oh = obuf);
```

```
            *oh++ = c;
        }

    template <typename T>
        inline void print(T x) {
            static int buf[21], cnt;
            if (x != 0) {
                for (cnt = 0; x; x /= 10) buf[++cnt] = x % 10 | 48;
                while (cnt) print((char)buf[cnt--]);
            } else {
                print('0');
            }
        }

    struct InputOutputStream {
        ~InputOutputStream() {
            fwrite(obuf, 1, oh - obuf, stdout);
        }

        template <typename T>
            inline InputOutputStream &operator>>(T &x) {
                static char c;
                for (c = read(); !isdigit(c); c = read()) {
                    if (c == -1) return *this;
                }
                for (x = 0; isdigit(c); c = read()) x = x * 10 + (c ^ '0');
                return *this;
            }

        template <typename T>
            inline InputOutputStream &operator<<(const T &x) {
                print(x);
                return *this;
            }
    } io;
}
using IO::io;

int main(){
    //freopen("in","r",stdin);
    invG=qpow(G,mod-2);
    fac[0]=ifac[0]=1; inv[1]=fac[1]=ifac[1]=1;
    for(int i=2;i<N;i++){
        inv[i]=fix(-(mod/i)*inv[mod%i]);
        fac[i]=fac[i-1]*i%mod;
        ifac[i]=ifac[i-1]*inv[i]%mod;
    }


#if 0
    n=read();m=read();
    for(int i=0;i<n;i++) a[i]=read();
    for(int i=0;i<m;i++) b[i]=read();
```

```
    t=read();
#endif
    io>>n>>m;
    for(int i=0;i<n;i++) io>>a[i];
    for(int i=0;i<m;i++) io>>b[i];
    io>>t;

    int k=1; while(k<=max(max(n,m),t)) k<<=1;

    solve(a,0,n,A); polyln(A,k,tmp);
    clear(A,n+1); deri(tmp,k,A);
    for(int i=t;i>0;i--) A[i]=(mod-A[i-1])%mod;
    A[0]=n;
    for(int i=0;i<=t;i++) A[i]=A[i]*ifac[i]%mod;
    cut(A,t+1,k);

    clear(tmp,k);

    solve(b,0,m,B); polyln(B,k,tmp);
    clear(B,m+1); deri(tmp,k,B);
    for(int i=t;i>0;i--) B[i]=(mod-B[i-1])%mod;
    B[0]=m;
    for(int i=0;i<=t;i++) B[i]=B[i]*ifac[i]%mod;
    cut(B,t+1,k);

    k<<=1; clear(tmp,k);
    initrev(k);
    trans(A,k,1); trans(B,k,1);
    for(int i=0;i<k;i++) tmp[i]=A[i]*B[i]%mod;
    trans(tmp,k,-1);

    for(int i=1;i<=t;i++) printf("%d\n",
            int(tmp[i]*inv[n]%mod*inv[m]%mod*fac[i]%mod));

    return 0;
}
```