1.FFT

```
//2e5要1s左右
#include<bits/stdc++.h>
using namespace std;
const int N=4000005;
const double pi=acos(-1);
typedef complex<double> E;
int R[N];
E A[N],B[N];
void fft(E *a,int n,int f){
    for(int i=0;i<n;i++)if(i<R[i])swap(a[i],a[R[i]]);
    for(int i=1;i<n;i<<=1){
        E wn(cos(pi/i),f*sin(pi/i));
        for(int p=i<<1,j=0;j<n;j+=p){
            E w(1,0);
            for(int k=0;k<i;k++,w*=wn){
                E x=a[j+k],y=w*a[j+k+i];
                a[j+k]=x+y;a[j+k+i]=x-y;
            }
        }
    }
}
void convolution(vector<int>&a,vector<int>&b){
    int n=a.size()-1,m=b.size()-1;
    for (int i=0;i<=n;i++)A[i]=a[i];
    for (int i=0;i<=m;i++)B[i]=b[i];
    for(int up=min(N-1,(n+m)*2),i=n+1;i<=up;++i)A[i]=0;
    for(int up=min(N-1,(n+m)*2),i=m+1;i<=up;++i)B[i]=0;
    m=n+m;int L=0;for(n=1;n<=m;n<<=1)L++;
    for (int i=0;i<n;i++)R[i]=(R[i>>1]>>1)|((i&1)<<(L-1));
    fft(A,n,1);fft(B,n,1);
    for (int i=0;i<n;i++)A[i]=A[i]*B[i];
    fft(A,n,-1);
    a.resize(m+1);
    for (int i=0;i<=m;i++){
        a[i]=(int)(A[i].real()/n+0.5);
    }
}
vector<int>a,b;
```

```
signed main(){
    int n,m;
    scanf("%d%d",&n,&m);
    a.resize(n+1);b.resize(m+1);
    for (int i=0;i<=n;i++)scanf("%d",&a[i]);
    for (int i=0;i<=m;i++)scanf("%d",&b[i]);
    convolution(a,b);
    for (int i=0;i<a.size();i++)printf("%d ",a[i]);
    return 0;
}
```

2.NTT

NTT需要满足 数组大小< (mod-1)的2的次数

```
#include<bits/stdc++.h>
using namespace std;
const int N=4000005;
const int G=3,mod=998244353;//(119<<23)+1
int A[N],B[N],R[N];
int qpow(int x,int y){
    int ans=1;
    while (y){
        if (y&1)ans=1ll*ans*x%mod;y>>=1;x=1ll*x*x%mod;
    }
    return ans;
}
void NTT(int* a,int n,int f){
    for (int i=0;i<n;i++)if(i<R[i])swap(a[i],a[R[i]]);
    for (int i=1;i<n;i<<=1){
        int gn=qpow(G,(mod-1)/(i<<1));
        for (int j=0;j<n;j+=(i<<1)){
            int g=1;
            for (int k=0;k<i;k++,g=1ll*g*gn%mod){
                int x=a[j+k],y=1ll*g*a[j+k+i]%mod;
                a[j+k]=(x+y)%mod;a[j+k+i]=(x-y+mod)%mod;
            }
        }
    }
    if (f==1)return;
    int nv=qpow(n,mod-2);reverse(a+1,a+n);
    for (int i=0;i<n;i++)a[i]=1ll*a[i]*nv%mod;
}
void ntt(int n,int m){
    m=n+m;int L=0;for(n=1;n<=m;n<<=1)L++;
    for (int i=0;i<n;i++)R[i]=(R[i>>1]>>1)|((i&1)<<(L-1));
    NTT(A,n,1);NTT(B,n,1);
    for (int i=0;i<n;i++)A[i]=1ll*A[i]*B[i]%mod;
    NTT(A,n,-1);
}
void convolution(vector<int>&a,vector<int>&b){
    int n=a.size()-1,m=b.size()-1;
    for (int i=0;i<=n;i++)A[i]=a[i];
    for (int i=0;i<=m;i++)B[i]=b[i];
    for(int up=min(N-1,(n+m)*2),i=n+1;i<=up;++i)A[i]=0;
    for(int up=min(N-1,(n+m)*2),i=m+1;i<=up;++i)B[i]=0;
    ntt(n,m);
```

```
    a.resize(n+m+1);
    for(int i=0;i<=n+m;++i)a[i]=A[i];
}
vector<int>a,b;
signed main(){
    int n,m;
    scanf("%d%d",&n,&m);
    a.resize(n+1);b.resize(m+1);
    for (int i=0;i<=n;i++)scanf("%d",&a[i]);
    for (int i=0;i<=m;i++)scanf("%d",&b[i]);
    convolution(a,b);
    for (int i=0;i<a.size();i++)printf("%d ",a[i]);
    return 0;
}
```

3.FWT

$C_i = \sum_{i=j \bigoplus k} A_j B_k$

```
#include<bits/stdc++.h>
using namespace std;
const int N=(1<<17)+5;
typedef long long ll;
static int n,Len=1<<17;
const int mod=998244353;
inline int ad(int u,int v){return (u+=v)>=mod?u-mod:u;}
inline void FMT(int *a){//or
    for(register int z=1;z<Len;z<<=1)
        for (int j=0;j<Len;j++)if(z&j)a[j]=ad(a[j],a[j^z]);
}
inline void IFMT(int *a){
    for(register int z=1;z<Len;z<<=1)
        for (int j=0;j<Len;j++)if(z&j)a[j]=ad(a[j],mod-a[j^z]);
}

inline void FWTand(int *a){
    for(register int i=2,ii=1;i<=Len;i<<=1,ii<<=1)
        for(register int j=0;j<Len;j+=i)
            for(register int k=0;k<ii;++k)
                a[j+k]=ad(a[j+k],a[j+k+ii]);
}
inline void IFWTand(int *a){
    for(register int i=2,ii=1;i<=Len;i<<=1,ii<<=1)
        for(register int j=0;j<Len;j+=i)
            for(register int k=0;k<ii;++k)
                a[j+k]=ad(a[j+k],mod-a[j+k+ii]);
}

inline void FWTxor(int *a){
    static int t;
    for(register int i=2,ii=1;i<=Len;i<<=1,ii<<=1)
        for(register int j=0;j<Len;j+=i)
            for(register int k=0;k<ii;++k){
                t=a[j+k];
                a[j+k]=ad(t,a[j+k+ii]);
                a[j+k+ii]=ad(t,mod-a[j+k+ii]);
            }
```

```
}
inline int div2(int x){return x&1?(mod+x)/2:x/2;}
inline void IFWTxor(int *a){
    static int t;
    for(register int i=2,ii=1;i<=Len;i<<=1,ii<<=1)
        for(register int j=0;j<Len;j+=i)
            for(register int k=0;k<ii;++k){
                t=a[j+k];
                a[j+k]=div2(ad(t,a[j+k+ii]));
                a[j+k+ii]=div2(ad(t,mod-a[j+k+ii]));
            }
}
static int A[N],B[N],C[N];
inline void init(){
    cin>>n;Len=1<<n;
    for (int i=0;i<Len;i++)cin>>A[i];
    for (int i=0;i<Len;i++)cin>>B[i];
}
inline void solve(){
    FMT(A);FMT(B);
    for (int i=0;i<Len;i++)C[i]=(unsigned long long)A[i]*B[i]%mod;
    IFMT(A);IFMT(B);IFMT(C);
    for (int i=0;i<Len;i++)cout<<C[i]<<" ";putchar('\n');

    FWTand(A);FWTand(B);
    for (int i=0;i<Len;i++)C[i]=(unsigned long long)A[i]*B[i]%mod;
    IFWTand(A);IFWTand(B);IFWTand(C);
    for (int i=0;i<Len;i++)cout<<C[i]<<" ";putchar('\n');

    FWTxor(A);FWTxor(B);
    for (int i=0;i<Len;i++)C[i]=(unsigned long long)A[i]*B[i]%mod;
    IFWTxor(C);
    for (int i=0;i<Len;i++)cout<<C[i]<<" ";putchar('\n');
}
int main(){
    init();
    solve();
    return 0;
}
```

4.MTT

```
#define ld long double
#define db double
#define ll long long
const ld pi=acos(-1);
struct cd{
    ld x,y;
    cd(ld _x=0.0,ld _y=0.0):x(_x),y(_y){}
    cd operator +(const cd &b)const{
        return cd(x+b.x,y+b.y);
    }
    cd operator -(const cd &b)const{
        return cd(x-b.x,y-b.y);
    }
    cd operator *(const cd &b)const{
        return cd(x*b.x - y*b.y,x*b.y + y*b.x);
```

```
    }
    cd operator /(const db &b)const{
        return cd(x/b,y/b);
    }
}a[N],b[N],c[N],d[N];
int rev[N],A[N],B[N],C[N];
void getrev(int bit){
    for(int i=0;i<(1<<bit);i++)rev[i]=(rev[i>>1]>>1)|((i&1)<<(bit-1));
}
void fft(cd *a,int n,int dft){
    for(int i=0;i<n;i++)if(i<rev[i])swap(a[i],a[rev[i]]);
    for(int step=1;step<n;step<<=1){
        cd wn(cos(dft*pi/step),sin(dft*pi/step));
        for(int j=0;j<n;j+=step<<1){
            cd wnk(1,0);
            for(int k=j;k<j+step;k++){
                cd x=a[k];
                cd y=wnk*a[k+step];
                a[k]=x+y;a[k+step]=x-y;
                wnk=wnk*wn;
            }
        }
    }
    if(dft==-1)for(int i=0;i<n;i++)a[i]=a[i]/n;
}
void mtt(int n,int m,int p) {
    int sz=0;
    while((1<<sz)<=n+m)sz++;getrev(sz);
    int len=1<<sz;
    for(int i=0;i<len;i++){
        int t1=A[i]%p,t2=B[i]%p;
        a[i]=cd(t1>>15,0),b[i]=cd(t1&0x7fff,0);
        c[i]=cd(t2>>15,0),d[i]=cd(t2&0x7fff,0);
    }
    fft(a,len,1);fft(b,len,1);fft(c,len,1);fft(d,len,1);
    for(int i=0;i<len;i++){
        cd aa=a[i],bb=b[i],cc=c[i],dd=d[i];
        a[i]=aa*cc;b[i]=bb*cc;c[i]=aa*dd;d[i]=bb*dd;
    }
    fft(a,len,-1);fft(b,len,-1);fft(c,len,-1);fft(d,len,-1);
    for(int i=0;i<len;i++){
        ll aa=(ll)(a[i].x+0.5),bb=(ll)(b[i].x+0.5);
        ll cc=(ll)(c[i].x+0.5),dd=(ll)(d[i].x+0.5);
        aa=(aa%p+p)%p;bb=(bb%p+p)%p;cc=(cc%p+p)%p;dd=(dd%p+p)%p;
        C[i]=((((aa<<15)%p)<<15)%p+(bb<<15)%p+(cc<<15)%p+dd)%p;
    }
}
void convolution(vector<int> &a,vector<int> &b,int mod){
    int n=a.size()-1,m=b.size()-1;
    for(int i=0;i<=n;++i)A[i]=a[i];
    for(int i=0;i<=m;++i)B[i]=b[i];
    for(int up=min(N-1,(n+m)*2),i=n+1;i<=up;++i)A[i]=0;
    for(int up=min(N-1,(n+m)*2),i=m+1;i<=up;++i)B[i]=0;
    for(int up=min(N-1,(n+m)*2),i=0;i<=up;++i)C[i]=0;
    mtt(n,m,mod);
    a.resize(n+m+1);
    for(int i=0;i<=n+m;++i)a[i]=C[i];
}
```

5.拉格朗日插值法

$y=y_1\frac{(x-x_2)(x-x_3)\ldots(x-x_n)}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_n)}+y_2\frac{(x-x_1)(x-x_3)\ldots(x-x_n)}{(x_2-x_1)(x_2-x_3)\ldots(x_2-x_n)}+\ldots+y_n\frac{(x-x_1)(x-x_2)\ldots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\ldots(x_n-x_{n-1})}$,经过n个点的曲线，但是会产生龙格现象

```
//O(klogk) 求1-n的k次方和
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int mod=1e9+7;
const int N=1e6+10;
ll n,k,fac[N],y[N];
ll qp(ll a,ll b) {
    ll s=1;
    while(b) {
        if(b&1)s=s*a%mod;
        a=a*a%mod,b>>=1;
    }
    return s;
}
ll sv(int n,int k) {
    fac[0]=fac[1]=1,y[1]=1;
    for(int i=2;i<=k+2;i++) fac[i]=fac[i-1]*i%mod;///预处理阶乘
    for(int i=2;i<=k+2;i++) y[i]=(y[i-1]+qp(i,k))%mod;///预处理求出每一项的结果
    if(n<=k+2)return y[n];
    ll sum=1,sig;
    for (ll i=n-k-2;i<=n-1;i++) sum=sum*i%mod;
    ll ans=0;
    for(ll i=1;i<=k+2;i++) {///k+1项的多项式有k+2项
        ll fz=sum*qp(n-i,mod-2)%mod;///分子
        ll fm=qp(fac[i-1]*fac[k+2-i]%mod,mod-2);///分母
        if((k+2-i)%2==0) sig=1;///正负号
        else sig=-1;
        ans=(ans+mod+sig*y[i]*fz%mod*fm%mod+mod)%mod;
    }
    return ans;
}
int main() {
    cin>>n>>k;
    cout<<sv(n,k)<<endl;
    return 0;
}
```

6.群论

2*2的格子放2(m)种颜色，旋转相同的算同一种，问有几种方案。

burnside引理

|G|为轨迹数（旋转方法数，0，90，180，270）

则方案=1/|G|（不动点总数）=1/|G|∑(c1,...cg)(c1为轨迹一的不动点数)

1.转360度（不动）： a1=(1)(2)...(16)
2.逆时针转90度： a2=(1)(2)(3 4 5 6)(7 8 9 10) (11 12)(13 14 15 16)
3.顺时针转90度： a3=(1)(2)(6 5 4 3)(10 9 8 7)(11 12)(16 15 14 13)
4.转180度：a4=(1)(2)(3 5)(4 6)(7 9)(8 10)(11)(12)(13 15)(14 16)
每个为一种情况的旋转，c1=16 c2=2 c3=2 c4=4
总方案数为1/4*(16+2+2+4)=6

polya定理

m种颜色，n个元素构成的环

L=1/|G|∗∑m^c(i) c(i)为第i种轨迹 循环节(循环)个数
1.转360度（不动）： a1=(1)(2)(3)(4)
2.旋转90度： a2=(1 2 3 4)
3.旋转180度： a3=(1 3)(2 4)
4.旋转270度：a4=(1 4 3 2)
每个为一个格子的情况，c1=4,c2=1,c3=2,c4=1
总方案数为1/4*(2^{4+2}1+2^{2+2}1)=6
在颜色元素无要求下，可用polya定理，其他情况用burnside定理，例相邻不能同色

01通路矩阵，1代表通，0代表不通，^k可以算出，若行走k次后,i,j为1代表一开始从i出发走k次可以走到j

m种颜色，n个元素构成的环，旋转置换一共有n种，旋转$i*2\pi/n$可看作旋转$2\pi/n$的i次幂，可拆分成gcd(n,i)个轮换，总方案数为$\sum_{i=1}^{n} m^{gcd(n,i)} = \sum_{d|n} m^d * \varphi(n/d)$

```
//burnside定理使用
#include<bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
const int N=1e6+10,mod=998244353;
int n,k;
int ans[N];
ll fac[N],inv[N];//2
ll qpow(ll a,ll n){
    ll ret=1;
    for (;n;n>>=1,a=a*a%mod)if(n&1)ret=ret*a%mod;
    return ret;
}
void init(){
    fac[0]=1;
    for(int i=1;i<N;i++){
        fac[i]=fac[i-1]*i%mod;
    }
}
ll C(ll x, ll y){
    if (x<y)return 0;
    return fac[x]*qpow(fac[y]*fac[x-y]%mod,mod-2)%mod;
}
int vis[N],prim[N],phi[N];
void get_phi(int n){
    int num=0;phi[1]=1;
    for (int i=2;i<=n;i++){
        if (vis[i]==0){prim[++num]=i;phi[i]=i-1;}
        for (int j=1;j<=num&&prim[j]*i<=n;j++){
            vis[i*prim[j]]=1;
            if (i%prim[j]==0){
```

```
                phi[i*prim[j]]=phi[i]*prim[j];
                break;
            }
            else phi[i*prim[j]]=phi[i]*phi[prim[j]];
        }
    }
}
int gcd(int x,int y){
    return x%y?gcd(y,x%y):y;
}
signed main(){
    int T;
    init();get_phi(N-1);
    cin>>T;
    while(T--){
        cin>>n>>k;
        for (int i=1;i<=n;i++){//区间个数
            if (n%i)continue;
            int kk=k/i;//绿色的上限
            int len=n/i;//区块长度
            int now=0;
            for (int j=0;j<=kk;j++){
                if (j==0){
                    if (len%2==0)now+=2;
                    continue;
                }
                (now+=qpow(2,j)*C(len-j-1,j))%=mod;
                (now+=qpow(2,j+1)*C(len-j-1,j-1))%=mod;
            }
            ans[i]=now;
        }
        int sum=0;
        for (int i=1;i<=n;i++){
            sum+=ans[i]*phi[i];//n%i==0时,gcd(n,j)==n/i的个数为phi[i]，此时
(j=n/i*k,k与i互质)
            //sum+=ans[n/gcd(n,i)];
            sum%=mod;
        }
        cout<<sum*qpow(n,mod-2)%mod<<endl;
    }
    return 0;
}
```

7.切比雪夫空间

曼哈顿距离转切比雪夫距离，菱形转正方形操作

曼哈顿空间上的点(x,y)转为切比雪夫上的点(x',y')

x'=x+y,y'=x-y

转换回来x=(x'+y')/2,y=(x-y)/2

数列

斐波那契数列

斐波那契数列$F_0=0,F_1=1,F_n=F_{n-1}+F_{n-2}$

通项公式：$\frac{1}{\sqrt{5}}[(\frac{1+\sqrt{5}}{2})^n-(\frac{1-\sqrt{5}}{2})^n]$

卢卡斯数列$L_0=2,L_1=1,L_n=L_{n-1}+L_{n-2}$

通项公式：$(\frac{1+\sqrt{5}}{2})^n+(\frac{1-\sqrt{5}}{2})^n$

$L_n^2-5F_n^2=-4$

求斐波那契第n项

1.矩阵快速幂

$\begin{bmatrix}F_{n-1} & F_n\end{bmatrix}=\begin{bmatrix}F_{n-2} & F_{n-1}\end{bmatrix}\begin{bmatrix}0 & 1\\1 & 1\end{bmatrix}$

令$P=\begin{bmatrix}0 & 1\\1 & 1\end{bmatrix}$，则$\begin{bmatrix}F_n & F_{n+1}\end{bmatrix}=\begin{bmatrix}F_0 & F_1\end{bmatrix}P^n$

2.快速倍增法

$F_{2k}=F_k(2F_{k+1}-F_k)$

$F_{2k+1}=F_{k+1}^2+F_k^2$

```
pair<int,int> fib(int n){//(F_n,F_n+1)
    if (n==0)return {0,1};
    auto p=fib(n>>1);
    int c=p.first*(2*p.second-p.first);
    int d=p.first*p.first+p.second*p.second;
    if (n&1)return {d,c+d};
    else return {c,d};
}
```

性质

$gcd(F_n,F_m)=F_{gcd(n,m)}$

自适应辛普森法(求积分)

求$\int_L^R\frac{cx+d}{ax+b}dx$

$a,b,c,d\in[-10,10],-100\le L\le R\le 100,R-L\ge 1$

法1：设$u=ax+b$换元，做积分

$(\frac{cx}{a}+(\frac{d}{a}-\frac{bc}{a^2})ln|ax+b|)|_L^R$

特判a=0

法2：自适应辛普森法

辛普森公式为用抛物线面积拟合函数面积

$f(x)$为原函数，$g(x)$为拟合的抛物线公式$g(x)=Ax^2+Bx+C$

$$
\begin{aligned}
\int_a^b f(x) \approx \int_a^b Ax^2+Bx+C &= (b^3-a^3)A/3+(b^2-a^2)B/2+(b-a)C \\
&= (b-a)/6(2A(b^2+ab+a^2)+3B(b+a)+6C) \\
&= \frac{b-a}{6}(f(a)+f(b)+4f(\frac{a+b}{2}))
\end{aligned}
$$

```
#include <bits/stdc++.h>
using namespace std;
double a,b,c,d,l,r;
inline double f(double x) {
    return (c*x+d)/(a*x+b);//原函数
}
inline double simpson(double l,double r){//Simpson公式
    double mid=(l+r)/2;
    return (f(l)+4*f(mid)+f(r))*(r-l)/6;
}
double asr(double l,double r,double eps,double ans) {
    double mid=(l+r)/2;
    double l_=simpson(l,mid),r_=simpson(mid,r);
    if(fabs(l_+r_-ans)<=15*eps)return l_+r_+(l_+r_-ans)/15;//确认精度
    return asr(l,mid,eps/2,l_)+asr(mid,r,eps/2,r_);//精度不够则递归调用
}
inline double asr(double l,double r,double eps) {
    return asr(l,r,eps,simpson(l,r));
}
signed main(){
    scanf("%lf%lf%lf%lf%lf%lf",&a,&b,&c,&d,&l,&r);
    printf("%.6lf",asr(l,r,1e-6));
    return 0;
}
```