ACM 模板

无敌爆哥

2023 年 9 月 13 日

目录

1 数学

1.1 康托展开

可以 O(nlogn) 复杂度求一个排列对于排列中的排名

```
const int N=1e6+10,mod=998244353;
long long fac[N];
int tr[N];
int n;
int a[N];
int lowbit(int x)
{
    return x&-x;
 }
void add(int x)
{
    for(int i=x;i<=n;i+=lowbit(i)) tr[i]+=1;
}
int qr(int x)
{
    int res=0;
    for(int i=x;i;i-=lowbit(i)) res+=tr[i];
    return res;
}
long long cantor(){
    fac[0]=1;
    for(int i=1;i<=n;i++) {
        fac[i]=fac[i-1]*i%mod;
    }
    long long ans=0;
    for(int i=1;i<=n;i++){
        ans=(ans+(a[i]-1-qr(a[i]))*fac[n-i]%mod)%mod;
        add(a[i]);
    }
    return (ans+1)%mod;
}
```

1.2 逆康托展开

已经知道排名得到原排列，逆向推出原序列

```cpp
vector<int> incantor(int x,int n) {
    x--;// 得到以0开始的排名
    vector<int> res(n);// 保存数列答案
    int cnt;
    bool st[10];// 标记数组
    memset(st,0,sizeof st);
    for(int i = 0;i < n; ++i) {
        cnt = x/fact[n - i - 1];// 比a[i]小且没有出现过的数的个数
        x %= fact[n - i - 1];// 更新 x
        for(int j = 1;j <= n; ++j) {// 找到a[i]，从1开始向后找
            if(st[j]) continue;// 如果被标记过，就跳过
            if(!cnt) {// 如果cnt == 0说明当前数是a[i]
                st[j] = 1;//标记
                res[i] = j;// 第i位是j
                break;
            }
            cnt --;// 如果当前不是0，就继续往后找
        }
    }
    return res;// 返回答案
}
```

1.3 线性筛求因子数量

```
int fac[N],q[N],top;
bool vis[N];
int cnt[N];
void init(int n=1e6)
{
    vis[1]=true;
    fac[1]=1;
    for(int i=2;i<=n;i++)
    {
        if(vis[i]==false)
        {
            q[top++]=i;
            fac[i]=2;
            cnt[i]=1;
        }
        for(int j=0;1ll*q[j]*i<=n;j++)
        {
            int nu=i*q[j];
            vis[nu]=true;
            if(i%q[j]==0)
            {
                fac[nu]=fac[i]/(cnt[i]+1)*(cnt[i]+2);
                cnt[nu]=cnt[i]+1;
                break;
            }
            fac[nu]=fac[i]*fac[q[j]];
            cnt[nu]=1;
        }
    }
}
```

1.4 二进制 GCD

快!

```
 1 int gcd(int a, int b) {
 2     int az = __builtin_ctz(a);
 3     int bz = __builtin_ctz(b);
 4     int z = min(az, bz);
 5     b >>= bz;
 6     while (a) {
 7         a >>= az;
 8         int diff = a - b;
 9         az = __builtin_ctz(diff);
10         b = min(a, b), a = abs(diff);
11     }
12     return b << z;
13 }
```

1.5 快速 gcd

预处理值域，常数访问 gcd (T 取 $\sqrt{max}$)

```
const int M=1e6+10,T=1e3+10;//值域
bool vis[M];
int q[M],cnt;
int fac[M][3];
int gc[T][T];
int gcd(int a,int b)
{
    return b?gcd(b,a%b):a;
}
void init(int n=M-10)
{
    vis[1]=true;
    vis[0]=true;
    fac[1][0]=fac[1][1]=fac[1][2]=1;
    for(int i=2;i<=n;i++)
    {
        if(!vis[i])
        {
            q[cnt++]=i;
            fac[i][0]=fac[i][1]=1;
            fac[i][2]=i;
        }
        for(int j=0;1ll*q[j]*i<=n;j++)
        {
            int nu=q[j]*i;
            vis[nu]=true;
            fac[nu][0]=fac[i][0]*q[j];
            fac[nu][1]=fac[i][1];
            fac[nu][2]=fac[i][2];
            if(fac[nu][0]>fac[nu][1]) swap(fac[nu][0],fac[
               nu][1]);
            if(fac[nu][1]>fac[nu][2]) swap(fac[nu][1],fac[
               nu][2]);
            if(i%q[j]==0) break;
```

```
33          }
34      }
35      for(int i=0;i<=T-10;i++)
36      {
37          for(int j=0;j<=T-10;j++)
38          {
39              if(!i&&!j) continue;
40              gc[i][j]=gcd(i,j);
41          }
42      }
43  }
44  int qgcd(int x,int y)
45  {
46      int res=1;
47      for(int i=0,t=1;i<3;i++)
48      {
49          if(fac[x][i]>T-10)
50          {
51              if(y%fac[x][i]) t=1;
52              else t=fac[x][i];
53          }
54          else t=gc[fac[x][i]][y%fac[x][i]];
55          res=res*t;y/=t;
56      }
57      return res;
58  }
```

1.6 求矩阵行列式值

原理是通过高斯消元将行列式下三角清 0，那么答案就是主对角线的累乘，这里由于模值不一定是质数，不能保障每一个元素都在 P 取模情况下有逆元，所以采用了特别妙的方法，通过主元行与其他行进来辗转相除来进行清 0 过程。原理是行列式中一行减去另外一行的乘积，行列式答案不变。

```
int mod=998244353;
const int N=610;
long long mp[N][N];
int n;
long long gauss()//求矩阵行列式的值
{
    long long res=1;
    for(int i=0;i<n;i++)
    {
        for(int j=i+1;j<n;j++)
        {
            while(mp[i][i])
            {
                long long div=mp[j][i]/mp[i][i];
                for(int k=i;k<n&&div;k++) {
                    mp[j][k]=((mp[j][k]-div*mp[i][k])%mod+
                       mod)%mod;
                }
                for(int k=i;k<n;k++) swap(mp[i][k],mp[j][k
                   ]);res=-res;
            }
            for(int k=i;k<n;k++) swap(mp[i][k],mp[j][k]);
               res=-res;
        }
    }
    res=(res+mod)%mod;
    for(int i=0;i<n;i++) res=res*mp[i][i]%mod;
    return res;
}
```

1.7 线性基

```
struct Vec{
    vector<long long>d,g;
    int n;
    int le;
    Vec(){};
    Vec(int _n){n=_n;d.resize(n+1);g.resize(n+1);}
    bool insert(long long x){
        for(int i=n;i>=0;i--){
            if((x>>i)&1){
                if(d[i]==0){
                    d[i]=x;
                    return true;
                }else{
                    x^=d[i];
                }
            }
        }
        return false;
    }
    void re(){
        for(int i=0;i<=n;i++){
            for(int j=i-1;j>=0;j--){
                if((d[i]>>j)&1){
                    d[i]^=d[j];
                }
            }
        }
        le=0;
        for(int i=0;i<=n;i++){
            if(d[i]){
                g[le++]=d[i];
            }
        }
    }
    long long qrk(long long x){
```

```
36          if(x>(1ll<<le)) return -1;
37          x--;
38          long long res=0;
39          for(int i=0;i<le;i++){
40              if((x>>i)&1){
41                  res^=g[i];
42              }
43          }
44          return res;
45      }
46  };
```

1.8 拓展中国剩余定理（EXCRT）

CRT 求的问题是有一个数余上不同的模值 M 之后，求余数 R 值
通过公式分析可以推出最终的模值为 M=LCM(m1,m2,m3..) ,R=LCM*k+q.m1+r1
有时候如果可以用中国剩余定理搞的题目，需要质数分别当模数，如果 wa 了试一下不用 5 什么的

```
long long gcd(long long a,long long b)
{
    return b?gcd(b,a%b):a;
}
long long exgcd(long long a,long long b,long long &x,long
   long &y)
{
    if(b==0)
    {
        x=1;
        y=0;
        return a;
    }
    long long x1,y1,d;
    d=exgcd(b,a%b,x1,y1);
    x=y1;
    y=x1-a/b*y1;
    return d;
}
long long lcm(long long a,long long b)
{
    return a/gcd(a,b)*b;
}
long long m[20],r[20];
int tp;
long long CRT()
{
    for(int i=1;i<tp;i++)
    {
        long long d,x,y;
        d=exgcd(m[i-1],m[i],x,y);
```

```
31          if((r[i]-r[i-1])%d) return -1;
32          x=x*(r[i]-r[i-1])/d;
33          x=(x%(m[i]/d)+m[i]/d)%(m[i]/d);
34          r[i]=m[i-1]*x+r[i-1];
35          m[i]=lcm(m[i-1],m[i]);
36          r[i]=(r[i]%m[i]+m[i])%m[i];
37      }
38      return r[tp-1]%m[tp-1];
39  }
```

解 at[i]*x%m[i]=r[i]//带有系数的 excrt

```
 1  long long CRT()
 2  {
 3      tp=n;
 4      long long ans=0,M=1,A,B,C,x,y;
 5      for(int i=1;i<=tp;i++)
 6      {
 7          A=(__int128)at[i]*M%m[i];
 8          B=m[i];
 9          C=(r[i]-at[i]*ans%m[i]+m[i])%m[i];
10          long long d=exgcd(A,B,x,y);
11          if(C%d) return -1;
12          x=(x%B+B)%B;
13          ans+=(__int128)(C/d)*x%(B/d)*M%(M*=B/d);
14          ans%=M;
15      }
16      return ans;
17  }
```

1.9 BSGS

$$a^x = b(mod p)$$

```
long long gcd(long long a,long long b)
{
    return b?gcd(b,a%b):a;
}
long long BSGS(long long a,long long b,long long p)
{
    a%=p,b%=p;
    if(b==1||p==1){
        return 0;
    }
    int k=0;
    long long A=1;
    while(1)
    {
        long long d=gcd(a,p);
        if(b%d!=0) {
            return -1;
        }
        if(d==1) break;
        k++;b/=d;p/=d;A=A*(a/d)%p;b%=p;
        if(A==b) {
            return k;
        }
    }
    long long m=ceil(sqrt(p));
    map<long long,int>mp;
    long long t=b;
    for(int i=0;i<m;i++)
    {
        mp[t]=i;
        t=t*a%p;
    }
    t=1;
    for(int i=1;i<=m;i++)
```

```
35      {
36          t=t*a%p;
37      }
38      long long tt=A;
39      for(int i=1;i<=m;i++)
40      {
41          tt=tt*t%p;
42          if(mp.count(tt)&&i*m-mp[tt]>=0)
43          {
44              return i*m-mp[tt]+k;
45          }
46
47      }
48      return -1;
49  }
```

1.10 卢卡斯求组合数

满足 mod 为质数

```
long long lucas(long long n,long long m)
{
    if(m==0) return 1;
    return lucas(n/mod,m/mod)*C(n%mod,m%mod)%mod;
}
```

1.11 错排

n 个不相同盒子和 n 个不相同球，对应球不能放到下标相同盒子中

```
f[0]=1;
f[1]=0;
f[2]=1;
f[i]=(f[i-1]+f[i-2])*(i-1);
```

1.12 高精度

```
struct BigInt{
    int w=6;
    int tw=1e6;
    vector<int>v;
    int len=0;
    BigInt(){len=0;}
    BigInt(int _len){len=_len;v.resize(len);}
    BigInt operator +(const BigInt &w)const{
        BigInt res;
        res.v=v;
        res.len=len;
        long long tp=0;
        for(int i=0;i<(int)v.size();i++){
            if(i<(int)w.v.size()){
                tp+=w.v[i];
            }
            tp+=v[i];
            res.v[i]=(tp%tw);
            tp/=tw;
        }
        while(tp) res.v.push_back(tp%tw),tp/=tw,res.len++;
        return res;
    }
    BigInt operator +(const int &x)const{
        int tp=x;
        BigInt res;
        res.v=v;
        res.len=len;
        for(int i=0;i<(int)v.size();i++){
            tp+=v[i];
            res.v[i]=(tp%tw);
            tp/=tw;
            if(tp==0) break;
        }
        while(tp) res.v.push_back(tp%tw),tp/=tw,res.len++;
```

```
36          return res;
37      }
38      //sub只有a>=b
39      BigInt operator -(const BigInt &w)const{
40          BigInt res;
41          res.v=v;
42          res.len=len;
43          for(int i=0;i<len;i++){
44              if(i<(int)w.len){
45                  res.v[i]-=w.v[i];
46              }
47              if(res.v[i]<0) res.v[i]+=tw,res.v[i+1]--;
48          }
49          while(res.len>1&&res.v.back()==0){
50              res.v.pop_back();
51              res.len--;
52          }
53          return res;
54      }
55      BigInt operator - (const int b)const{
56          BigInt res;
57          res.v=v;
58          res.len=len;
59          res.v[0]-=b;
60          for(int i=0;i<(int)v.size();i++){
61              while(res.v[i]<0) res.v[i]+=tw,res.v[i]--;
62          }
63          while(res.len>1&&res.v.back()==0){
64              res.v.pop_back();res.len--;
65          }
66          return res;
67      }
68      BigInt operator *(const BigInt & w)const{
69          BigInt res(len+w.len+2);
70          for(int i=0;i<len;i++){
71              for(int j=0;j<w.len;j++){
72                  long long tp=1ll*v[i]*w.v[j];
```

```
73                  res.v[i+j+1]+=tp/tw;
74                  res.v[i+j]+=tp%tw;
75                  res.v[i+j+1]+=res.v[i+j]/tw;
76                  res.v[i+j]%=tw;
77              }
78          }
79          while(res.len>1&&res.v.back()==0){
80              res.v.pop_back();res.len--;
81          }
82          return res;
83      }
84      BigInt operator *(const int  w)const{
85          BigInt res;
86          res.v=v;
87          res.len=len;
88          long long tp=0;
89          for(int i=0;i<len;i++){
90              tp+=1ll*w*res.v[i];
91              res.v[i]=tp%tw;
92              tp/=tw;
93          }
94          while(tp){
95              res.v.push_back(tp%tw);res.len++;
96              tp/=tw;
97          }
98          return res;
99      }
100     BigInt operator /(const int w)const{
101         BigInt res;
102         res.v=v;
103         res.len=len;
104         long long tp=0;
105         for(int i=len-1;i>=0;i--){
106             tp=tp*tw+v[i];
107             res.v[i]=tp/w;
108             tp%=w;
109         }
```

```
110         while(res.len>1&&res.v.back()==0){
111             res.v.pop_back();res.len--;
112         }
113         return res;
114     }
115     int operator %(const int w)const{
116         long long res=0;
117         for(int i=len-1;i>=0;i--){
118             res=res*tw+v[i];
119             res%=w;
120         }
121         return res;
122     }
123     void read(string &a){
124         for(int i=(int)a.size();i-1>=0;i-=w){
125             int res=0;
126             for(int j=max(0,i-w);j<=i-1;j++){
127                 res=res*10+(a[j]-'0');
128             }
129             v.push_back(res);len++;
130         }
131     }
132     /*
133     1 a>b
134     -1 a<b
135     0 a=b
136     */
137     int equal(const BigInt &w){
138         if(len>w.len) return 1;
139         else if(len<w.len) return -1;
140         for(int i=len-1;i>=0;i--){
141             if(v[i]>w.v[i]) return 1;
142             else if(v[i]<w.v[i]) return -1;
143         }
144         return 0;
145     }
146     void pr(){
```

```
147         if(len==1){
148             cout<<v[0];
149         }
150         else{
151             int f=0;
152             for(int i=len-1;i>=0;i--){
153                 int x=v[i];
154                 int tww=tw;
155                 for(int j=0;j<w;j++){
156                     tww/=10;
157                     int y=x/tww;
158                     if(y) f=1;
159                     if(f) cout<<y;
160                     x%=tww;
161                 }
162             }
163         }
164         cout<<'\n';
165     }
166 };
```

1.13 高精度 GCD

```
BigInt gcd(BigInt a,BigInt b){
    int tp=0;
    while(1){
        int t=a.equal(b);
        if(t==-1) swap(a,b);
        if(t==0||b.len==1&&b.v[0]==0) break;
        if(!(a.v[0]&1)&&!(b.v[0]&1)){
            tp++;
            a=a/2;
            b=b/2;
        }
        else if(!(a.v[0]&1)){
            a=a/2;
        }
        else if(!(b.v[0]&1)){
            b=b/2;
        }else{
            a=a-b;
        }
    }
    while(tp--){
        a=a*2;
    }
    return a;
}
```

1.14 拉格朗日插值

$$f(x)=\sum_{i=1}^{n}y_i\prod_{j=1}^{n}\frac{x-x_j}{x_i-x_j}(x_i\neq x_j)$$

```
const int N=2e5+10,mod=998244353;
long long ksm(long long a,long long b){
    long long res=1;
    a%=mod;
    while(b){
        if(b&1) res=res*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return res;
}
struct pp{
    long long x,y;
}p[N];
int n;
long long lagrange(long long k){
    long long ans=0;
    for(int i=1;i<=n;i++){
        long long res=1;
        long long res1=1;
        for(int j=1;j<=n;j++){
            if(i==j) continue;
            res=res*(p[i].x-p[j].x+mod)%mod;
            res1=res1*(k-p[j].x+mod)%mod;
        }
        ans=(ans+p[i].y*res1%mod*ksm(res,mod-2)%mod)%mod;
    }
    return ans;
}
```

1.15 FFT(快速傅里叶变换)

系数表示法—> 点值表示法—> 系数表示法

采用单位根，运用负数表达，系数需要小数表达可以选择 fft

```
const int N=3e6+10;
int r[N];
int n,m;
const double PI=acos(-1);
struct complex_{
    double a,b;
    complex_(){a=0;b=0;}
    complex_(double _a,double _b){a=_a;b=_b;}
    complex_ operator +(const complex_ &w)const{
        return complex_(a+w.a,b+w.b);
    }
    complex_ operator -(const complex_ &w)const{
        return complex_(a-w.a,b-w.b);
    }
    complex_ operator *(const complex_ &w)const{
        return complex_(a*w.a-b*w.b,a*w.b+b*w.a);
    }
};
struct FFT{
    vector<complex_>v;
    int len;
    FFT(){}
    FFT(int _len){v.resize(_len);len=_len;}
    void init(int _len){
        v.resize(_len);
        len=_len;
    }
    void fft(int type){
        for(int i=0;i<len;i++){
            if(i<r[i]) swap(v[i],v[r[i]]);
        }
        for(int mid=1;mid<len;mid<<=1){
            complex_ wn(cos(PI/mid),type*sin(PI/mid));
```

```
34              for(int R=mid<<1,j=0;j<len;j+=R){
35                  complex_ w(1,0);
36                  for(int k=0;k<mid;k++,w=w*wn){
37                      complex_ x=v[j+k],y=w*v[j+k+mid];
38                      v[j+k]=x+y;
39                      v[j+k+mid]=x-y;
40                  }
41              }
42          }
43      }
44      void mul(const FFT &w){
45          for(int i=0;i<len;i++){
46              v[i]=v[i]*w.v[i];
47          }
48      }
49  };
50  int limit,top;
51  void init(){
52      limit=1;top=0;
53      while(limit<=n+m) limit<<=1,top++;
54      for(int i=0;i<limit;i++){
55          r[i]=(r[i>>1]>>1)|((i&1)<<(top-1));
56      }
57  }
```

1.16 原根

若 $gcd(a,m)==1$ 同时 $a^x \equiv 1(modm)$ 的最小阶为 $\phi(m)$ 则表示 a 是 m 的原根
求解通常是暴力算，已经被证明原根渐近 $m^{\frac{1}{4}}$ 所以时间还是比较可观
且只有 2,4,p^a ,$2p^a$（p 为奇质数）才有原根

```
int n,d;
    cin>>n>>d;
    if(!vvis[n]){ //通过线性筛预处理出 2 4 p^a 2p^a
        cout<<"0\n";
        cout<<'\n';
    }else{
        int t=getol(n);//获得欧拉函数
        vector<int>pr=getpr(t);//获得质因子
        int fas=0;
        for(int g=1;;g++){
            int f=0;
            if(__gcd(g,n)!=1) continue;
            //因为阶是n的欧拉函数，所以因子都不能满足
            for(int v:pr){
                if(ksm(g,t/v,n)==1){
                    f=1;break;
                }
            }
            if(f==0){
                fas=g;break;//最小原根
            }
        }
        vector<int>ans;
        int num=fas;
        //所有原根
        for(int i=1;i<=t;i++){
            if(__gcd(i,t)==1){
                ans.push_back(num);
            }
            num=num*fas%n;
        }
    }
```

1.17 NTT（快速数论变换）

采用原根来表达，都为整数运算，可以相对 fft 减少精度的影响。

```
/*
65537
998244353
1004535809
4179340454199820289
因为ntt受限制模数需要p=r*2^l+1 （其中l需要比给定运算的多项式的log(len)要大）
所有以上是常见的 且g=3 如果遇到一个其余模数，可以先预处理看是否符合ntt要求在选择ntt求解
*/
```

```
const int N=3e6+10;
int r[N];
int limit,L;
void init(int n){
    for(int i=0;i<n;i++){
        r[i]=(r[i>>1]>>1)|((i&1)<<(L-1));
    }
}
struct NTT{
    int len;
    int g=3,gi=332748118;
    long long mod=998244353;
    vector<long long>v;
    NTT(){g=3;gi=332748118;mod=998244353;}
    NTT(int _len){len=_len;v.resize(len);g=3;gi=332748118;mod=998244353;}
    void init(int _len){len=_len;v.resize(len);g=3;gi=332748118;mod=998244353;}
    void ntt(int type){
        for(int i=0;i<len;i++){
            if(i<r[i]) swap(v[i],v[r[i]]);
        }
        for(int mid=1;mid<len;mid<<=1){
```

```
22              long long wn=ksm(type==1?g:gi,(mod-1)/(mid<<1)
                    ,mod);
23              for(int j=0,R=mid<<1;j<limit;j+=R){
24                  long long w=1;
25                  for(int k=0;k<mid;k++,w=(w*wn)%mod){
26                      long long x=v[j+k],y=v[j+mid+k]*w%mod;
27                      v[j+k]=(x+y)%mod;
28                      v[j+k+mid]=(x-y+mod)%mod;
29                  }
30              }
31          }
32      }
33      void mul(const NTT &w){
34          for(int i=0;i<len;i++){
35              v[i]=(v[i]*w.v[i])%mod;
36          }
37      }
38  };
```

1.18 多项式求逆（ntt）

```
#include<bits/stdc++.h>
using namespace std;
int read() {
    int q=0;char ch='␣';
    while(ch<'0'||ch>'9') ch=getchar();
    while(ch>='0'&&ch<='9') q=q*10+ch-'0',ch=getchar();
    return q;
}
const int mod=998244353,G=3,N=2100000;

long long ksm(long long a,long long b)
{
    long long res=1;
    a%=mod;
    while(b)
    {
        if(b&1) res=1ll*res*a%mod;
        a=1ll*a*a%mod;
        b>>=1;
    }
    return res;
}
long long a[N],b[N],c[N];
int r[N];
void NTT1(long long *a,int n,int x) {
    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=ksm(G,(mod-1)/(i<<1));
        for(int j=0;j<n;j+=(i<<1)) {
            int t1,t2,g=1;
            for(int k=0;k<i;++k,g=1LL*g*gn%mod) {
                t1=a[j+k],t2=1LL*g*a[j+k+i]%mod;
                a[j+k]=(t1+t2)%mod,a[j+k+i]=(t1-t2+mod)%
                    mod;
            }
```

```
35          }
36      }
37      if(x==1) return;
38      long long ny=ksm(n,mod-2); reverse(a+1,a+n);
39      for(int i=0;i<n;++i) a[i]=1LL*a[i]*ny%mod;
40  }
41  void work(int n,long long *a,long long *b) {
42      if(n==1) {b[0]=ksm(a[0],mod-2);return;}
43      work((n+1)>>1,a,b);
44      int l=0,limit=1;
45      while(limit<2*n) limit<<=1,++l;
46      for(int i=1;i<limit;++i) r[i]=(r[i>>1]>>1)|((i&1)<<(l
            -1));
47      for(int i=0;i<n;++i) c[i]=a[i];
48      for(int i=n;i<limit;++i) c[i]=0;
49      NTT1(c,limit,1);
50      NTT1(b,limit,1);
51      for(int i=0;i<limit;++i)
52          b[i]=1LL*(2-1LL*c[i]*b[i]%mod+mod)%mod*b[i]%mod;
53      NTT1(b,limit,-1);
54      for(int i=n;i<limit;++i) b[i]=0;
55  }
56  int main()
57  {
58      int n;
59      n=read();
60      for(int i=0;i<n;++i) a[i]=read();
61      work(n,a,b);
62      for(int i=0;i<n;++i) printf("%lld␣",b[i]);
63      return 0;
64  }
```

1.19 区间维护等差数列

当有修改操作等需要给某一段区间加上等差数列，可以通过如下操作来进行维护，最后多用于求某单点值

```
#include<bits/stdc++.h>
using namespace std;
/*
这是无离散化版本
离散化只需要将有下标的地方改成对应 id[下标]即可
*/
const int N=1e5+10;
int n,q;
int a[N];
struct pp
{
    long long a,d;
}tr[N<<2];

void build(int k,int l,int r)
{
    tr[k].a=tr[k].d=0;
    if(l==r)
    {
        tr[k].a=a[l];
        return;
    }
    int mi=(l+r)>>1;
    build(k<<1,l,mi);
    build(k<<1|1,mi+1,r);
}

void down(int k,int l,int r)
{
    int mi=(l+r)>>1;
    tr[k<<1].d+=tr[k].d;
    tr[k<<1|1].d+=tr[k].d;
    tr[k<<1].a+=tr[k].a;
```

```
34      tr[k<<1|1].a+=tr[k].a+(mi+1-l)*tr[k].d;
35      tr[k].a=tr[k].d=0;
36  }
37
38  void mif(int k,int l,int r,int x,int y,int tp,long long a,
        long long d)
39  {
40      if(l==x&&r==y)
41      {
42          tr[k].d+=d;
43          tr[k].a+=a+(l-tp)*d;
44          return;
45      }
46      int mi=(l+r)>>1;
47      down(k,l,r);
48      if(y<=mi) mif(k<<1,l,mi,x,y,tp,a,d);
49      else if(mi<x) mif(k<<1|1,mi+1,r,x,y,tp,a,d);
50      else mif(k<<1,l,mi,x,mi,tp,a,d),mif(k<<1|1,mi+1,r,mi
            +1,y,tp,a,d);
51  }
52
53  long long qr(int k,int l,int r,int x)
54  {
55      if(l==x&&r==x)
56      {
57          return tr[k].a;
58      }
59      int mi=(l+r)>>1;
60      down(k,l,r);
61      if(x<=mi) return qr(k<<1,l,mi,x);
62      else return qr(k<<1|1,mi+1,r,x);
63  }
64
65  int main()
66  {
67      ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
68      cin>>n>>q;
```

```
69      for(int i=1;i<=n;i++)
70      {
71          cin>>a[i];
72      }
73      build(1,1,n);
74      while(q--)
75      {
76          int op;
77          cin>>op;
78          if(op==1)
79          {
80              int l,r;
81              long long a,d;
82              cin>>l>>r>>a>>d;
83              mif(1,1,n,l,r,l,a,d);
84          }
85          else{
86              int x;
87              cin>>x;
88              cout<<qr(1,1,n,x)<<'\n';
89          }
90      }
91      return 0;
92  }
```

1.20 随机化

1.20.1 随机数据生成器

```
1 mt19937 rng(chrono::steady_clock::now().time_since_epoch()
     .count());   //uint
2 mt19937_64 rng(chrono::steady_clock::now().
     time_since_epoch().count());//ull  大概率生成不相同数据
3 int getRand(int l,int r){
4     uniform_int_distribution < int > uid(l,r);
5     return uid(rng);
6 }
```

1.20.2 随机排列数组

```
1 random_shuffle(a,a+n);
```

1.21 二项式反演

g_n 表示至多有 n 个/种方案数量，f_n 恰好 n 个/种方案数量

$$g_n = \sum_{i=0}^{n} \binom{n}{i} f_i \iff f_n = \sum_{i=0}^{n} (-1)^{n-i} \binom{n}{i} g_i$$

g_k 表示至少有 n 个/种方案数量，f_k 恰好 n 个/种方案数量

$$g_k = \sum_{i=k}^{n} \binom{i}{k} f_i \iff f_k = \sum_{i=k}^{n} (-1)^{i-k} \binom{i}{k} g_i$$

1.22 Pollard-Rho

```
#define LL long long
LL mul(LL a, LL b, LL p) {
    return (a * b - (LL)(a / (long double)p * b + 1e-3) *
       p + p) % p;
}
LL power(LL a, LL r, LL p) {
    LL res = 1;
    for (; r; a = mul(a, a, p), r >>= 1)
        if (r & 1)
            res = mul(res, a, p);
    return res;
};
bool miller_rabin(LL n) {
    static LL p[9] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
    if (n == 1)
        return false;
    LL d = n - 1, r = 0;
    for (; not(d & 1); d >>= 1)
        r += 1;
    bool res = true;
    for (int i = 0; i < 9 and p[i] < n and res; i += 1) {
        LL x = power(p[i], d, n);
        if (x == 1 or x == n - 1)
            continue;
        for (int j = 1; j < r; j += 1) {
            x = mul(x, x, n);
            if (x == n - 1)
                break;
        }
        if (x != n - 1)
            res = false;
    }
    return res;
};
vector<LL> pollard_rho(LL n) {
```

```
35      vector<LL> res;
36      function<void(LL)> rho = [&](LL n) {
37          if (n == 1)
38              return;
39          if (miller_rabin(n))
40              return res.push_back(n), void();
41          LL d = n;
42          while (d == n) {
43              d = 1;
44              for (LL k = 1, y = 0, x = 0, s = 1, c = rand()
                    % n; d == 1;
45                   k <<= 1, y = x, s = 1) {
46                  for (int i = 1; i <= k; i += 1) {
47                      x = (mul(x, x, n) + c) % n;
48                      s = mul(s, abs(x - y), n);
49                      if (not(i % 127) or i == k) {
50                          d = __gcd(s, n);
51                          if (d != 1)
52                              break;
53                      }
54                  }
55              }
56          }
57          rho(d);
58          rho(n / d);
59      };
60      rho(n);
61      return res;
62  }
```

1.23 杜教筛

能够在 $O(n^{\frac{2}{3}})$ 的时间负责度里求前 n 项积性函数的和

$g(1)s(n)=\sum_{i=1}^{n}(f*g)(i)-\sum_{i=2}^{n}g(i)s(\lfloor\frac{n}{i}\rfloor)$

前 n 项目欧拉函数和莫比乌斯函数和

```
const int N=3e6+10;
int q[N],cnt;
long long u[N],o[N];
bool vis[N];
long long sumol[N],sumu[N];
void init(int n=N-10)
{
    vis[1]=true;
    o[1]=1;
    u[1]=1;
    for(int i=2;i<=n;i++)
    {
        if(vis[i]==false)
        {
            q[cnt++]=i;
            u[i]=-1;
            o[i]=i-1;
        }
        for(int j=0;1ll*q[j]*i<=n;j++)
        {
            int nu=q[j]*i;
            vis[nu]=true;
            if(i%q[j]==0)
            {
                u[nu]=0;
                o[nu]=o[i]*q[j];
                break;
            }
            u[nu]=-u[i];
            o[nu]=o[i]*(q[j]-1);
        }
    }
```

```
33      for(int i=1;i<=n;i++)
34      {
35          sumol[i]=sumol[i-1]+o[i];
36          sumu[i]=sumu[i-1]+u[i];
37      }
38  }
39  map<long long,long long>mpo,mpu;
40  long long get_o(long long n)//前n项欧拉函数的和
41  {
42      if(n<=N-10) return sumol[n];
43      if(mpo.count(n)) return mpo[n];
44      long long ans=1ll*n*(n+1)/2;
45      for(long long int l=2,r;l<=n;l=r+1)
46      {
47          r=n/(n/l);
48          ans-=get_o(n/l)*(r-l+1);
49      }
50      return mpo[n]=ans;
51  }
52
53  long long get_u(long long n)//前n项莫比乌斯函数的和
54  {
55      if(n<=N-10) return sumu[n];
56      if(mpu.count(n)) return mpu[n];
57      long long ans=1;
58      for(long long int l=2,r;l<=n;l=r+1)
59      {
60          r=n/(n/l);
61          ans-=get_u(n/l)*(r-l+1);
62      }
63      return mpu[n]=ans;
64  }
```

1.24 伯特兰-切比雪夫定理

若整数 n>3，则至少存在一个质数 P，符合 n<P<2*n-2 ，一个稍弱说法，一个大于 1 的整数，至少存在一个质数 P，符合 n<P<2*n

1.25 基础博弈论

```
一．巴什博奕（Bash Game）：

A和B一块报数，每人每次报最少1个，最多报4个，看谁先报到30。这应该是最古老的关于巴什博奕的游戏了吧。

其实如果知道原理，这游戏一点运气成分都没有，只和先手后手有关，比如第一次报数，A报k个数，那么B报5-k个数，那么B报数之后问题就变为，A和B一块报数，看谁先报到25了，进而变为20,15,10,5，当到5的时候，不管A怎么报数，最后一个数肯定是B报的，可以看出，作为后手的B在个游戏中是不会输的。

那么如果我们要报n个数，每次最少报一个，最多报m个，我们可以找到这么一个整数k和r，使n=k*（m+1）+r，代入上面的例子我们就可以知道，如果r=0，那么先手必败；否则，先手必胜。

二．威佐夫博弈（Wythoff Game）：

有两堆各若干的物品，两人轮流从其中一堆取至少一件物品，至多不限，或从两堆中同时取相同件物品，规定最后取完者胜利。

直接说结论了，若两堆物品的初始值为（x，y），且x<y，则另z=y-x；

记w=（int）[（（sqrt（5）+1）/2）*z ]；

若w=x，则先手必败，否则先手必胜。

三．尼姆博弈（Nimm Game）：

尼姆博弈指的是这样一个博弈游戏：有任意堆物品，每堆物品的个
```

数是任意的，双方轮流从中取物品，每一次只能从一堆物品中取部分或全部物品，最少取一件，取到最后一件物品的人获胜。

结论就是：把每堆物品数全部异或起来，如果得到的值为0，那么先手必败，否则先手必胜。

四．斐波那契博弈：

有一堆物品，两人轮流取物品，先手最少取一个，至多无上限，但不能把物品取完，之后每次取的物品数不能超过上一次取的物品数的二倍且至少为一件，取走最后一件物品的人获胜。

结论是：先手胜当且仅当n不是斐波那契数（n为物品总数）

2 几何

能不用浮点就不用

2.1 几何通用

```
//#define double long double//wa了就试一下
const double eps=1e-12;
const double pi=acos(-1);
int cmpb(double x)
{
    if(fabs(x)<eps) return 0;
    if(x>0) return 1;
    return -1;
}
struct point{
    double x,y;
    point(){x=0;y=0;}
    point(double _x,double _y){x=_x;y=_y;};
    double operator |(const point &w) const//点乘
    {
        return x*w.x+y*w.y;
    }
    double operator ^(const point &w) const//叉乘
    {
        return x*w.y-y*w.x;
    }
    point operator -(const point &w) const
    {
        point res;
        res.x=x-w.x;res.y=y-w.y;
        return res;
    }
    point operator +(const point &w) const
    {
        point res;
        res.x=x+w.x;res.y=y+w.y;
```

```
32          return res;
33      }
34      point operator /(double w) const
35      {
36          point res(x/w,y/w);
37          return res;
38      }
39      point operator *(double w) const
40      {
41          point res(x*w,y*w);
42          return res;
43      }
44      double db(double x)   //平方
45      {
46          return x*x;
47      }
48      double dict(point &to)//点距
49      {
50          return sqrt(db(x-to.x)+db(y-to.y));
51      }
52      double dz()
53      {
54          return sqrt(db(x)+db(y));
55      }
56      void rotate(double _o) //旋转都是逆时针
57      {
58          double _x=x,_y=y;
59          x=_x*cos(_o)-_y*sin(_o);
60          y=_x*sin(_o)+_y*cos(_o);
61      }
62      double arg()//绕正x轴旋转多少度到达
63      {
64          int xb=cmpb(x),yb=cmpb(y);
65          if(xb==0&&yb==0) return 0.0;
66          if(yb==0)
67          {
68              if(xb==1) return 0.0;
```

```
69              else return pi;
70          }
71          if(xb==0)
72          {
73              if(yb==1) return pi/2.0;
74              else return 1.5*pi;
75          }
76          double ta=fabs(atan(y/x));
77          if(xb==1&&yb==-1) ta+=1.5*pi;
78          else if(xb==-1&&yb==-1) ta+=pi;
79          else if(xb==-1&&yb==1) ta+=0.5*pi;
80          return ta;
81      }
82  };
83  struct Line{
84      double a,b,c;      //直线一般式
85      Line(){a=0;b=0;c=0;}
86      Line(double _a,double _b,double _c)
87      {
88          a=_a;
89          b=_b;
90          c=_c;
91      }
92      Line(point &x,point &y)//俩点一线
93      {
94          a=y.y-x.y;
95          b=x.x-y.x;
96          c=x.y*y.x-x.x*y.y;
97      }
98      point cross(Line &x)   //线交点
99      {
100         point res;
101         res.x=(x.c*b-c*x.b)/(a*x.b-x.a*b);
102         res.y=(x.c*a-c*x.a)/(b*x.a-x.b*a);
103         return res;
104     }
105      bool check(point &x)//判断点是否在线上
```

```
106     {
107         return cmpb(x.x*a+x.y*b+c)==0;
108     }
109     Line getL(point &x){//获得得某点到这个条线的垂直线
110         double a1,b1,c1;
111         if(a==0) {
112             a1=1.0;
113             b1=0;
114             c1=-x.x;
115         }
116         else if(b==0)
117         {
118             a1=0;
119             b1=1.0;
120             c1=-x.y;
121         }
122         else{
123             a1=b;
124             b1=-a;
125             c1=a*x.y-b*x.x;
126         }
127         Line re(a1,b1,c1);
128         return re;
129     }
130     bool Px(Line &w)//判断平行
131     {
132         return cmpb(a*w.b-b*w.a)==0;
133     }
134     bool Li(Line &w)//判断垂直
135     {
136         return cmpb(a*w.b+b*w.a)==0;
137     }
138 };
139 struct Line //起点 终点版
140 {
141     point s,e;
142     Line(){}
```

```
143     Line(point _s,point _e){s=_s;e=_e;}
144     point cross(Line &W) {
145         point u=s-W.s,v=e-s,w=W.e-W.s;
146         double t=(u^w)/(w^v);
147         return s+v*t;
148     }
149     bool right(Line b,Line c)//判断b,c交点是否在当前线段右
            边
150     {
151         point p=b.cross(c);
152         return ((e-s)^(p-s))<0;
153     }
154     double arg() //极角 (-pi,pi]
155     {
156         return atan2(e.y-s.y,e.x-s.x);
157     }
158 };
159 double cro(point a,point b,point c)
160 {
161     return (b-a)^(c-a);
162 }
163
164 double dot(point a,point b,point c)
165 {
166     return (b-a)|(c-a);
167 }
168
169 struct point3{//三维
170     double x,y,z;
171     point3()
172     {
173         x=0.0;y=0.0;z=0.0;
174     }
175     point3(double _x,double _y,double _z)
176     {
177         x=_x;y=_y;z=_z;
178     }
```

```
179     point3 operator -(const point3 w)
180     {
181         return (point3){x-w.x,y-w.y,z-w.z};
182     }
183     point3 operator +(const point3 w)
184     {
185         return (point3){x+w.x,y+w.y,z+w.z};
186     }
187     point3 operator ^(const point3 w)
188     {
189         return (point3){(y*w.z-z*w.y),(z*w.x-x*w.z),(x*w.y
              -y*w.x)};
190     }
191     double operator |(const point3 w)
192     {
193         return x*w.x+y*w.y+z*w.z;
194     }
195     double rand01()
196     {
197         return (double)rand()/(double)RAND_MAX;
198     }
199     double randeps()
200     {
201         return (double)(rand01()-0.5)*eps;
202     }
203     void noise()//三维点需要抖动一下避免有一些影响 十分重
          要!
204     {
205         x+=randeps();y+=randeps(),z+=randeps();
206     }
207     double db(double x)
208     {
209         return x*x;
210     }
211     double dict(const point3 w)
212     {
213         return sqrt(db(x-w.x)+db(y-w.y)+db(z-w.z));
```

```
214     }
215     double mod()
216     {
217         return sqrt(x*x+y*y+z*z);
218     }
219 };
```

2.2 极角序

```
//排序cmp
struct argcmp{//极角排序
    int quad(const point&x) //象限排序 根据题目定义
        {
            if(x.y<-eps) return 1;
            else if(x.y>eps) return 3;
            else if(x.x>eps) return 4;
            else if(x.x<-eps) return 5;
            return 2;
        };
    bool operator()(const point &x,const point &y)
    {

        int xb=quad(x);
        int yb=quad(y);
        if(xb!=yb)
        {
            return xb<yb;
        }
        return (x^y)>eps; //右手定则
    }
};
```

2.3 凸包

2.3.1 二维凸包

```
const int N=4e5+10;
bool cmp(point x,point y)
{
    return (x.x==y.x?x.y<y.y:x.x<y.x);
}
point st[N],p[N];
int top,tp;
double Andrew()
{
    top=0;//多组清空
    sort(p+1,p+tp+1,cmp);
    for(int i=1;i<=tp;i++)
    {
        while(top>1&&((st[top]-st[top-1])^(p[i]-st[top-1]))<=0.0) top--;
        st[++top]=p[i];
    }
    int t=top;
    for(int i=tp-1;i>=1;i--)
    {
        while(top>t&&((st[top]-st[top-1])^(p[i]-st[top-1]))<=0.0) top--;
        st[++top]=p[i];
    }
    double ans=0.0;
    for(int i=1;i<top;i++)
    {
        ans=ans+st[i].dict(st[i+1]);
    }
    return ans;
}
```

2.4 三维凸包

2.4.1 暴力

```
const int N=110;
point3 p[N];
int n;
bool check(int id,int id1,int id2)
{
    double now=0;
    int c1=0,c2=0;
    point3 vr=(p[id1]-p[id])^(p[id2]-p[id]);
    for(int i=1;i<=n;i++)
    {
        if(i==id||i==id1||i==id2) continue;
        now=((p[i]-p[id])|vr);
        if(now>0)
        {
            c1++;
        }
        else if(now<0){
            c2++;
        }
        if(c1&&c2) return false;

    }
    return true;
}
void solve()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%Lf%Lf%Lf",&p[i].x,&p[i].y,&p[i].z);
        p[i].noise();
    }
    double ans=0;
```

```
34      for(int i=1;i<=n;i++)
35      {
36          for(int j=i+1;j<=n;j++)
37          {
38              for(int k=j+1;k<=n;k++)
39              {
40                  if(check(i,j,k))
41                  {
42                      ans+=((p[j]-p[i])^(p[k]-p[i])).mod()
                           /2.0; //通过向量积的模长一半求得出
43                  }
44              }
45          }
46      }
47      printf(”%Lf”,ans);
48  }
```

2.5 增量法

```
const int N=2010;
point3 p[N];
struct face{
    int v[3];
    face(){v[0]=v[1]=v[2]=0;}
    face(int a,int b,int c){v[0]=a;v[1]=b;v[2]=c;}
    point3 normal(){return ((p[v[1]]-p[v[0]])^(p[v[2]]-p[v[0]]));}
    double area(){return normal().mod()/2.0;}
};
int cansee(face A,point3 b) {return ((b-p[A.v[0]])|A.normal())>0;}
face cv[N],h[N];
int n,th,cnt,vis[N][N];
void convex3() {
    cnt=th=0;
    cv[++cnt]=face(1,2,3);
    cv[++cnt]=face(3,2,1);
    for(int i=4;i<=n;++i) {
        for(int j=1,v;j<=cnt;++j) {
            if(!(v=cansee(cv[j],p[i])))h[++th]=cv[j];
            for(int k=0;k<3;++k)
                vis[cv[j].v[k]][cv[j].v[k>1?0:k+1]]=v;
        }
        for(int j=1;j<=cnt;++j)
            for(int k=0;k<3;++k) {
                int x=cv[j].v[k],y=cv[j].v[k>1?0:k+1];
                if(vis[x][y]&&!vis[y][x])h[++th]=face(x,y,i);
            }
        for(int j=1;j<=th;++j)cv[j]=h[j];
        cnt=th,th=0;
    }
}
double calc(){
```

```
33      double res=0;
34      for(int i=1;i<=cnt;i++)
35      {
36          res+=cv[i].area();
37      }
38      return res;
39  }
40  double vol6(point3 a,point3 b,point3 c,point3 d) {
41      return ((b-a)^(c-a))|(d-a);
42  } //计算体积的6倍。理解：先算出底的有向面积的2倍，向量的方向变为高，然后点乘就是另一条边在高方向的投影，由体积公式(V=1/3*Sh)，这个值就是体积的6倍
43  double calcV() {//体积
44      double res=0;
45      for(int i=1;i<=cnt;++i)
46          res+=fabs(vol6(p[1],p[cv[i].v[0]],p[cv[i].v[1]],p[
                cv[i].v[2]]));
47      res/=6.0;
48      return res;
49  }
```

2.6 线段到线段最小距离

```
1  double cro(point a,point b,point c)
2  {
3      return ((b-a)^(c-a));
4  }
5
6  double dot(point p0,point p1,point p2){  //点积 p0为角点
7      return (p1-p0)|(p2-p0);
8  }
9
10 double getDis(point p0,point p1,point p2){
11      if(cmpb(p0.dict(p1))==0) return p0.dict(p2);//p0 p1共
           同点
12      if(cmpb(dot(p0,p1,p2))==-1) return p2.dict(p0);//钝角
13      if(cmpb(dot(p1,p0,p2))==-1) return p2.dict(p1);//钝角
14      return fabs(cro(p0,p1,p2)/p0.dict(p1));  //点到线的距
           离
15 }
16
17 double minDis(point p1,point p2,point p3,point p4){//线段
      间最短距离
18      return min(min(getDis(p1,p2,p3),getDis(p1,p2,p4)),min
           (getDis(p3,p4,p1),getDis(p3,p4,p2)));
19 }
```

2.7 旋转卡壳

2.7.1 凸包直径

```
double rotating_calingpers()
{
    top--;
    double ans=0;
    for(int i=0,j=1;i<top;i++)
    {
        while(((st[(i+1)%top]-st[i])^(st[j]-st[i]))<
              ((st[(i+1)%top]-st[i])^(st[(j+1)%top]-st[i])
                 )){
            j=(j+1)%top;
        }
        ans=max(ans,st[i].dict(st[j]));
        ans=max(ans,st[(i+1)%top].dict(st[j]));
    }
    return ans;
}
```

2.7.2 俩个不相交凸包最小距离

```
double cro(point a,point b,point c)
{
    return ((b-a)^(c-a));
}

double dot(point p0,point p1,point p2){  //点积 p0为角点
    return (p1-p0)|(p2-p0);
}

double getDis(point p0,point p1,point p2){
     if(cmpb(p0.dict(p1))==0) return p0.dict(p2);//p0 p1共同点
     if(cmpb(dot(p0,p1,p2))==-1) return p2.dict(p0);//钝角
     if(cmpb(dot(p1,p0,p2))==-1) return p2.dict(p1);//钝角
     return fabs(cro(p0,p1,p2)/p0.dict(p1));  //点到线的距离
}

double minDis(point p1,point p2,point p3,point p4){//线段间最短距离
     return min(min(getDis(p1,p2,p3),getDis(p1,p2,p4)),min(getDis(p3,p4,p1),getDis(p3,p4,p2)));
}

double r_c(int f)
{
    int ymi=1,ymx=1;
    for(int i=1;i<=top[f];i++)
    {
        if(st[f][i].y<st[f][ymi].y) ymi=i;
    }
    for(int i=1;i<=top[f^1];i++)
    {
        if(st[f^1][i].y>st[f^1][ymx].y) ymx=i;
    }//分别最小和最大的开始找就保证的下面那些情况保证在凸
```

```
        多边形靠近的那附件来旋转
32      double t=0,ans=1e20;
33      for(int i=1;i<=top[f];i++){
34          //因为是逆时针 所以所以旋转到acb>acd时候 就相当于
              (| /) (\ /) (\ |) 这种情况
35          while(t=cro(st[f][ymi],st[f][ymi+1],st[f^1][ymx])-
              cro(st[f][ymi],st[f][ymi+1],st[f^1][ymx+1])<-
              eps)
36          {
37              ymx=ymx%top[f^1]+1;
38          }
39          if(t>eps) ans=min(ans,getDis(st[f][ymi],st[f][ymi
              +1],st[f^1][ymx]));
40          else ans=min(ans,minDis(st[f][ymi],st[f][ymi+1],st
              [f^1][ymx],st[f^1][ymx+1]));
41          ymi=ymi%top[f]+1;
42      }
43      return ans;
44  }
```

2.7.3 最小矩阵覆盖

```
double r_c()
{
    double ans=1e20;
    int a,b,c;
    a=b=c=2;
    int n=top-1;
    for(int i=2;i<=top;i++)
    {
        while(cro(st[i-1],st[i],st[a])<cro(st[i-1],st[i],
           st[a%n+1])) a=a%n+1;
        while(dot(st[i-1],st[i],st[b])<dot(st[i-1],st[i],
           st[b%n+1])) b=b%n+1;
        if(i==2) c=a;
        while(dot(st[i],st[i-1],st[c])<dot(st[i],st[i-1],
           st[c%n+1])) c=c%n+1;
        double H=cro(st[i-1],st[i],st[a]);
        double d=st[i].dict(st[i-1]);
        H/=d;
        double R=dot(st[i-1],st[i],st[b])/d,L=dot(st[i],st
           [i-1],st[c])/d;
        if(H*(R+L-d)<ans)
        {
            ans=H*(R+L-d);
            res[0]=st[i-1]+(st[i]-st[i-1])*R/d;
            res[3]=st[i]+(st[i-1]-st[i])*L/d;
            res[1]=res[3]-res[0];
            res[1].rotate(-pi/2.0);
            res[1]=res[1]*H/(L+R-d);
            res[1]=res[1]+res[0];
            res[2]=res[0]-res[3];
            res[2]=res[2]*H/(L+R-d);
            res[2].rotate(pi/2.0);
            res[2]=res[2]+res[3];
        }

```

```
32      }
33      return ans;
34  }
```

2.8 闵可夫斯基和

A,B 俩个凸包的闵可夫斯基和为 $C=a+b|a\in A,b\in B$

```
const int N=4e5+10;
bool cmp(point x,point y)
{
    return (x.x==y.x?x.y<y.y:x.x<y.x);
}
point st[3][N],p[3][N];
int top[3],tp[3];
void Andrew(int id)
{
    top[id]=0;//多组清空
    sort(p[id]+1,p[id]+tp[id]+1,cmp);
    for(int i=1;i<=tp[id];i++)
    {
        while(top[id]>1&&((st[id][top[id]]-st[id][top[id
          ]-1])^(p[id][i]-st[id][top[id]-1]))<=0.0) top[
          id]--;
        st[id][++top[id]]=p[id][i];
    }
    int t=top[id];
    for(int i=tp[id]-1;i>=1;i--)
    {
        while(top[id]>t&&((st[id][top[id]]-st[id][top[id
          ]-1])^(p[id][i]-st[id][top[id]-1]))<=0.0) top[
          id]--;
        st[id][++top[id]]=p[id][i];
    }
    top[id]--;
}
point s1[N],s2[N];
void Minkovski(int id,int id1,int id2)//a={b+c}
{
    for(int i=1;i<top[id1];i++) s1[i]=st[id1][i+1]-st[id1
      ][i];s1[top[id1]]=st[id1][1]-st[id1][top[id1]];//s1
        是id1凸包
```

```
29      for(int i=1;i<top[id2];i++) s2[i]=st[id2][i+1]-st[id2
          ][i];s2[top[id2]]=st[id2][1]-st[id2][top[id2]];//s2
          是id2凸包
30      int p1=1,p2=1;
31      tp[id]=1;
32      p[id][tp[id]]=st[id1][1]+st[id2][1];//归并加入一下，p[
          id]数组就是闵可夫斯基和
33      while(p1<=top[id1]&&p2<=top[id2]) tp[id]++,p[id][tp[id
          ]]=p[id][tp[id]-1]+((s1[p1]^s2[p2])>=0?s1[p1++]:s2[
          p2++]);//极角序从小到大并从p[id][top[id]-1]开始往后
          接
34      while(p1<=top[id1]) tp[id]++,p[id][tp[id]]=p[id][tp[id
          ]-1]+s1[p1++];
35      while(p2<=top[id2]) tp[id]++,p[id][tp[id]]=p[id][tp[id
          ]-1]+s2[p2++];
36  }
37  bool cmp2(point a,point b)//判断
38  {
39      return (a^b)>0||(a^b)==0&&a.dz()<b.dz();
40  }
41  int in(point a,int id) //通过极角序来判断点是否在凸包中
42  {
43      if((a^st[id][1])>0||(st[id][top[id]]^a)>0) return 0;
44      int to=lower_bound(st[id]+1,st[id]+top[id]+1,a,cmp2)-
          st[id]-1;
45      return ((a-st[id][to])^(st[id][to%top[id]+1]-st[id][to
          ]))<=0;
46  }
```

2.9 半平面交

```
const int N=4e5+10;
Line le[N],dq[N];
int pt;
bool cmphp(Line a,Line b)//极角+左侧
{
    double A=a.arg(),B=b.arg();
    return cmpb(A-B)!=0?A<B:((a.e-a.s)^(b.e-a.s))<0.0;
}
void half_plane()//半平面交
{
    sort(le+1,le+pt+1,cmphp);
    int h=1,t=1;dq[1]=le[1];
    for(int i=2;i<=pt;i++)
    {
        if(le[i].arg()-le[i-1].arg()<eps) continue;
        while(h<t&&le[i].right(dq[t],dq[t-1])) t--;//维护头尾
        while(h<t&&le[i].right(dq[h],dq[h+1])) h++;
        dq[++t]=le[i];
    }
    while(h<t&&dq[h].right(dq[t],dq[t-1])) t--; //割多余尾巴
//  dq[++t]=dq[h];//封口 看题目需要
}
```

2.10 最小圆覆盖

```
const int N=1e5+10;
int n;
point p[N];
point gauss(point x,point y,point z){
    double c1=x.db(x.dz())-y.db(y.dz());
    double c2=x.db(x.dz())-z.db(z.dz());
    point e=x-y,e1=x-z;
    point o;
    o.x=(c1*e1.y-c2*e.y)/(2.0*(e.x*e1.y-e1.x*e.y));
    o.y=(c1*e1.x-c2*e.x)/(2.0*(e.y*e1.x-e1.y*e.x));
    return o;
}
void getcircle(point &o,double &r){
    random_shuffle(p,p+n);
    o=p[0];
    for(int i=1;i<n;i++){
        if(cmpb(r-o.dict(p[i]))==-1){
            o=p[i];
            r=0.0;
            for(int j=0;j<i;j++){
                if(cmpb(r-o.dict(p[j]))==-1){
                    o=(p[i]+p[j])/2.0;
                    r=o.dict(p[j]);
                    for(int k=0;k<j;k++){
                        if(cmpb(r-o.dict(p[k]))==-1){
                            o=gauss(p[i],p[j],p[k]);
                            r=o.dict(p[k]);
                        }
                    }
                }
            }
        }
    }
}
```

2.11 二维向量绕 0 点旋转 n 度

```
1 x1=x*cos(n)-y*sin(n);
2 y1=x*sin(n)+y*cos(n);
```

2.12 椭圆

a 为长半轴 b 为短半轴周长为 L=2πb+4(a-b) 面积为 S=πab

2.13 皮克定理

格点三角形的面积 $S = n + m/2 - 1$ n 表示三角形内部点的格点数量 m 表示三角形边上的顶点数

2.14 扫描线

```
#include<bits/stdc++.h>
using namespace std;
const int N=2e5+10;
struct Line
{
    int x,y1,y2,f,id1,id2;
    bool operator<(const Line &le1)
    {
        return x<le1.x;
    }
}le[N];
vector<int>b;
int n;
int find(int x)
{
    return lower_bound(b.begin(),b.end(),x)-b.begin()+1;
}
struct pp
{
    long long le;
    int f;
}tr[N<<4];
void up(int k,int l,int r)
{
    if(!tr[k].f){
        tr[k].le=tr[k<<1].le+tr[k<<1|1].le;
    }
    else{
        tr[k].le=b[r]-b[l-1];
    }
}
void mif(int k,int l,int r,int x,int y,int f)
{
    if(l==x&&r==y)
    {
```

```
36          tr[k].f+=f;
37          up(k,l,r);
38          return;
39      }
40      int mi=(l+r)>>1;
41      if(y<=mi) mif(k<<1,l,mi,x,y,f);
42      else if(mi<x) mif(k<<1|1,mi+1,r,x,y,f);
43      else mif(k<<1,l,mi,x,mi,f),mif(k<<1|1,mi+1,r,mi+1,y,f)
            ;
44      up(k,l,r);
45  }
46  int main()
47  {
48      ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
49      cin>>n;
50      for(int i=1;i<=n;i++)
51      {
52          int x,y,x1,y1;
53          cin>>x>>y>>x1>>y1;
54          le[i]={x,y,y1,1};
55          le[n+i]={x1,y,y1,-1};
56          b.push_back(y);
57          b.push_back(y1);
58      }
59      sort(b.begin(),b.end());
60      b.erase(unique(b.begin(),b.end()),b.end());
61      n<<=1;
62      sort(le+1,le+n+1);
63      for(int i=1;i<=n;i++)
64      {
65          le[i].id1=find(le[i].y1);
66          le[i].id2=find(le[i].y2);
67      }
68      long long ans=0;
69      int m=(int)b.size();
70      for(int i=1;i<n;i++)
71      {
```

```
72          mif(1,1,m,le[i].id1,le[i].id2-1,le[i].f);
73          ans+=1ll*tr[1].le*(le[i+1].x-le[i].x);
74      }
75      cout<<ans;
76      return 0;
77  }
```

2.15 平面最近点对

```
const int N=1e6+10;
int n;
point p[N],st[N];
bool cmpx(point &x,point &y){
    return x.x<y.x;
}
bool cmpy(point &x,point &y){
    return x.y<y.y;
}
double find(int l,int r){
    if(l==r) return 1e18;
    if(r-l==1){
        return p[l].dict(p[r]);
    }
    int mi=(l+r)>>1;
    double d=min(find(l,mi),find(mi+1,r));
    int tp=0;
    for(int i=l;i<=r;i++){
        if(cmpb(d-fabs(p[i].x-p[mi].x))==1){
            st[++tp]=p[i];
        }
    }
    sort(st+1,st+tp+1,cmpy);
    for(int i=1;i<tp;i++){
        for(int j=i+1;j<=tp&&st[j].y-st[i].y<d;j++){
            d=min(d,st[i].dict(st[j]));
        }
    }
    return d;
}
```

2.16 自适应辛普森公式

分成很多小段把一小段当成二次函数，通过分治进行求解

```
const double eps=1e-12;
int cmpb(double x)
{
    if(fabs(x)<eps) return 0;
    if(x>0) return 1;
    return -1;
}
double f(double x)//对应题目函数
{
    double ans;
    return ans;
}
double simpson(double l,double r){//辛普森公式
    return (r-l)*(f(r)+f(l)+4.0*f((r+l)/2.0))/6.0;
}
double asr(double l,double r,double ans){//自适应
    double mi=(l+r)/2.0;
    double a=simpson(l,mi),b=(simpson(mi,r));
    if(cmpb(a+b-ans)==0) return ans;
    return asr(l,mi,a)+asr(mi,r,b);
}
```

2.17 浮点输出

```
1 cout<<fixed<<setprecision(2)<<ans;
```

3 其它

3.1 快读快写开 02

```
#pragma GCC optimize(2)
#pragma GCC optimize(3,"Ofast","inline")

inline int read()
{
    char c=getchar();int x=0,f=1;
    while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}
    while(c>='0'&&c<='9'){x=x*10+c-'0';c=getchar();}
    return x*f;
}
inline void write(int x)
{
    static int t[25];int tp=0;
    if(x==0) return(void)(putchar('0'));else if(x<0)
       putchar('-'),x=-x;
    while(x) t[tp++]=x%10,x/=10;
    while(tp--) putchar(t[tp]+48);
}
```

3.2 自定义开栈

```
#pragma comment(linker, "/STACK:1024000000,1024000000")

//自定义开栈
int size(512<<20); // 512M
 __asm__ ( "movq %0, %%rsp\n"::"r"((char*)malloc(size)+
    size)); // YOUR CODE
```

3.3 测时间

```
    clock_t start, finish;
```

```
2       start = clock();
3       /*******代码*******/
4       /****************/
5       finish = clock();
6       cout << "the time cost is" <<double(finish - start) /
            CLOCKS_PER_SEC;
```

3.4 bitset 处理多维偏序

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N=5010,M=510;
int n,m;
int mp[M][N];
long long p[N],dp[N];
bitset<N>tp[N];
vector<int>tr[N];
int in[N];
typedef pair<int,int> pii;
bool cmp(pii x,pii y)
{
    if(x.first==y.first)
    {
        return x.second<y.second;
    }
    return x.first<y.first;
}
int main()
{
    ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
    cin>>n>>m;
    for(int i=1;i<=m;i++)
    {
        cin>>p[i];
        dp[i]=p[i];
    }
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=m;j++)
        {
            cin>>mp[i][j];
        }
    }
```

```
36      for(int i=1;i<=m;i++)
37      {
38          for(int j=1;j<=m;j++){
39              tp[i][j]=1;
40          }
41      }
42      for(int i=1;i<=n;i++)
43      {
44          vector<pii>v;
45          for(int j=1;j<=m;j++)
46          {
47              v.push_back({mp[i][j],j});
48          }
49          sort(v.begin(),v.end(),cmp);
50          bitset<N>tpp;
51          int l=0;
52          for(int j=0;j<(int)v.size();j++)
53          {
54              if(v[l].first!=v[j].first)
55              {
56                  while(l<j){
57                      tpp[v[l].second]=1;
58                      l++;
59                  }
60              }
61              tp[v[j].second]&=tpp;
62          }
63      }
64      long long ans=0;
65      for(int i=1;i<=m;i++)
66      {
67          for(int j=1;j<=m;j++)
68          {
69              if(tp[i][j]){
70                  tr[j].push_back(i);
71                  in[i]++;
72              }
```

```
73          }
74      }
75      queue<int>q;
76      for(int i=1;i<=m;i++)
77      {
78          if(in[i]==0) q.push(i);
79      }
80      while(q.size())
81      {
82          int v=q.front();
83          q.pop();
84          ans=max(ans,dp[v]);
85          for(int u:tr[v])
86          {
87              dp[u]=max(dp[u],dp[v]+p[u]);
88              if(--in[u]==0) q.push(u);
89          }
90      }
91      cout<<ans;
92      return 0;
93  }
```

3.5 CDQ 分治

三维偏序（陌上开花）

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N=2e5+10;
int n,k;
struct pp
{
    int a,b,c,cnt,ans;
    bool operator ==(const pp &x)
    {
        return a==x.a&&b==x.b&&c==x.c;
    }
    bool operator <(const pp &x)
    {
        if(a==x.a)
        {
            if(b==x.b)
            {
                return c<x.c;
            }
            return b<x.b;
        }
        return a<x.a;
    }
}p[N],nw[N];
bool cmp(pp x,pp y)
{
    return x.b<y.b;
}
int top;
int tr[N];
int lowbit(int x)
{
    return x&-x;
```

```
35 }
36 void add(int x,int val)
37 {
38     for(int i=x;i<=k;i+=lowbit(i)) tr[i]+=val;
39 }
40 int qr(int x)
41 {
42     int res=0;
43     for(int i=x;i;i-=lowbit(i)) res+=tr[i];
44     return res;
45 }
46 void CDQ(int l,int r)
47 {
48     if(l==r) return;
49     int mi=(l+r)>>1;
50     CDQ(l,mi);
51     CDQ(mi+1,r);
52     sort(nw+l,nw+mi+1,cmp);
53     sort(nw+mi+1,nw+r+1,cmp);
54     int j=l;
55     for(int i=mi+1;i<=r;i++)
56     {
57         while(j<=mi&&nw[j].b<=nw[i].b){
58             add(nw[j].c,nw[j].cnt);
59             j++;
60         }
61         nw[i].ans+=qr(nw[i].c);
62     }
63     for(int i=l;i<j;i++) add(nw[i].c,-nw[i].cnt);
64 }
65 int ans[N];
66 int main()
67 {
68     ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
69     cin>>n>>k;
70     for(int i=1;i<=n;i++)
71     {
```

```
72          cin>>p[i].a>>p[i].b>>p[i].c;p[i].cnt=1;
73      }
74      sort(p+1,p+n+1);
75      for(int i=1,r=1;i<=n;i=r)
76      {
77          top++;
78          int re=1;
79          while(re+i<=n&&p[i]==p[re+i]){
80              re++;
81          }
82          nw[top]=p[i];nw[top].cnt=re;
83          r=i+re;
84      }
85      CDQ(1,top);
86      for(int i=1;i<=top;i++)
87      {
88          ans[nw[i].cnt+nw[i].ans-1]+=nw[i].cnt;
89      }
90      for(int i=0;i<n;i++) {
91          cout<<ans[i]<<'\n';
92      }
93      return 0;
94  }
```

3.6 Tanjan

3.6.1 边双联通分量

```
#include<bits/stdc++.h>
using namespace std;

const int N=5e6+10;
int n,m;
int head[N],ne[N],to[N],ct=1;
void add(int v,int u)
{
    ne[++ct]=head[v];
    head[v]=ct;
    to[ct]=u;
}
int dfn[N],low[N],p;
int st[N],tp;
int tot[N],cnt;
bool in[N];
void tanjan(int now,int old)
{
    if(dfn[now]==0)
    {
        dfn[now]=++p;
        low[now]=dfn[now];
        st[++tp]=now;
        in[now]=true;
    }
    for(int i=head[now];~i;i=ne[i])
    {
        int v=to[i];
        if(!dfn[v])
        {
            tanjan(v,i);
            low[now]=min(low[now],low[v]);
        }
```

```
34          else if(i!=(old^1)) {
35              low[now]=min(low[now],dfn[v]);
36          }
37      }
38      if(low[now]==dfn[now])
39      {
40          ++cnt;
41          int x;
42          do
43          {
44              x=st[tp--];
45              tot[x]=cnt;
46          }while(x!=now);
47      }
48  }
49
50  int main()
51  {
52      ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
53      cin>>n>>m;
54      for(int i=1;i<=n;i++)
55      {
56          head[i]=-1;
57      }
58      for(int i=1;i<=m;i++)
59      {
60          int v,u;
61          cin>>v>>u;
62          if(v==u) continue;
63          add(v,u);
64          add(u,v);
65      }
66      vector<vector<int>>ans(n+1);
67      for(int i=1;i<=n;i++)
68      {
69          if(!dfn[i])
70          {
```

```
71              tanjan(i,0);
72          }
73      }
74      for(int i=1;i<=n;i++) ans[tot[i]].push_back(i);
75      cout<<cnt<<'\n';
76      for(int i=1;i<=cnt;i++)
77      {
78          cout<<(int)ans[i].size()<<" ";
79          for(int j=0;j<(int)ans[i].size();j++)
80          {
81              cout<<ans[i][j]<<" ";
82          }
83          cout<<'\n';
84      }
85      return 0;
86  }
```

3.6.2 缩点

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N=2e5+10;
int head[N],ne[N],to[N],ct=1;
void add(int v,int u)
{
    ne[++ct]=head[v];
    head[v]=ct;
    to[ct]=u;
}
int n,m;
long long w[N],totw[N];
int tot[N];
int st[N],tp;
int dfn[N],low[N],cnt,p;
bool vis[N];
void tanjan(int now,int old)
{
    dfn[now]=++p;
    low[now]=p;
    st[++tp]=now;
    vis[now]=true;
    for(int i=head[now];~i;i=ne[i])
    {
        int v=to[i];
        if(!dfn[v])
        {
            tanjan(v,i);
            low[now]=min(low[now],low[v]);
        }
        else if(vis[v]){
            low[now]=min(low[now],dfn[v]);
        }
    }
```

```
36      if(dfn[now]==low[now])
37      {
38          ++cnt;
39          int x;
40          do{
41              x=st[tp--];
42              totw[cnt]+=w[x];
43              vis[x]=false;
44              tot[x]=cnt;
45          }while(x!=now);
46      }
47  }
48  vector<int>tr[N];
49  int in[N];
50  long long dp[N];
51  int main()
52  {
53      ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
54      cin>>n>>m;
55      for(int i=1;i<=n;i++)
56      {
57          cin>>w[i];
58          head[i]=-1;
59      }
60      for(int i=1;i<=m;i++)
61      {
62          int v,u;
63          cin>>v>>u;
64          if(v==u) continue;
65          add(v,u);
66      }
67      for(int i=1;i<=n;i++)
68      {
69          if(!dfn[i])
70          {
71              tanjan(i,0);
72          }
```

```
73      }
74      for(int i=1;i<=n;i++)
75      {
76          for(int j=head[i];~j;j=ne[j])
77          {
78              int v=to[j];
79              if(tot[i]!=tot[v])
80              {
81                  tr[tot[i]].push_back(tot[v]);
82              }
83          }
84      }
85      queue<int>q;
86      for(int i=1;i<=cnt;i++)
87      {
88          sort(tr[i].begin(),tr[i].end());
89          tr[i].erase(unique(tr[i].begin(),tr[i].end()),tr[i
               ].end());
90      }
91      for(int i=1;i<=cnt;i++)
92      {
93          for(int v:tr[i])
94          {
95              in[v]++;
96          }
97      }
98      for(int i=1;i<=cnt;i++) {
99          if(in[i]==0){
100           q.push(i);
101         }
102         dp[i]=totw[i];
103     }
104     long long ans=0;
105     while(q.size())
106     {
107         int u=q.front();
108         q.pop();
```

```
109         ans=max(ans,dp[u]);
110         for(int i=0;i<(int)tr[u].size();i++)
111         {
112             int v=tr[u][i];
113             dp[v]=max(dp[u]+totw[v],dp[v]);
114             if(--in[v]==0) q.push(v);
115         }
116     }
117     cout<<ans;
118     return 0;
119 }
```

3.7 匈牙利算法

```cpp
using pii=pair<int,int>;
const int maxn=1e6;
vector<int>v[maxn];
int mt[maxn];
int n,m,e;
bitset<maxn>vis;
bool dfs(int x)
{
    if(vis[x])return 0;
    vis[x]=1;
    for(auto y:v[x]){
        if(!mt[y]||dfs(mt[y])){
            mt[y]=x;
            return 1;
        }
    }
    return 0;
}
int main()
{
    cin>>n>>m>>e;
    for(int i=1;i<=e;i++){
        int x,y;
        cin>>x>>y;
        v[x].push_back(y);
        // v[y].push_back(x);
    }
    int ans=0;
    for(int i=1;i<=n;i++){
        vis.reset();
        if(dfs(i))ans++;
    }
    cout<<ans<<endl;
}
```

3.8 倍增优化建图

```
int find(int x,int y){
    if(id[x][y])return id[x][y];
    id[x][y]=++Flow::tot;
    if(!y){
        Flow::Add(id[x][y],Flow::t,val[x]);
    }
    else {
        Flow::Add(id[x][y],find(x,y-1),INF);
        Flow::Add(id[x][y],find(f[x][y-1],y-1),INF);
    }
    return id[x][y];
}

// 与倍增求lca相同的循环
int lca=LCA(x,y);
for(int i=14;i>=0;i--)
    if((dep[x]-dep[lca])>>i&1)
    {
        Flow::Add(now,find(x,i),INF);
        x=f[x][i];
    }
for(int i=14;i>=0;i--)
    if((dep[y]-dep[lca])>>i&1)
    {
        Flow::Add(now,find(y,i),INF);
        y=f[y][i];
    }
```

3.9 字符串哈希

```
namespace EEE{
    class StringHash{
    public:
        const ull mod=21237044013013795 7ll;
        const ull prime=233317;
        const ull base=131;
        vector<ull>h;
        vector<ull>bas;
        StringHash():h(1),bas(1,1){}
        void push_back(char ch){
            h.push_back((h.back()*base+ch)%mod + prime);
            bas.push_back(bas.back()*base%mod);
        }
        ull get(int l,int r){
            return (h[r] + __int128(h[l])*(mod-bas[r-l]))%
               mod;
        }
    };
    void A()
    {
        string str;
        StringHash hs,rhs;
        int N=int(str.size());
        for(int i=0;i<N;i++)
            hs.push_back(str[i]);
        for(int i=N-1;i>=0;i--)
            rhs.push_back(str[i]);
    }
};
```

3.10 精确 DLX

```
const int maxn=1e6+100;
const int MaxM=450010;
const int MaxP=510;
struct DLX{
    int U[MaxM],D[MaxM],L[MaxM],R[MaxM];
    int Row[MaxM],Col[MaxM];
    int First[MaxP];
    int Size[MaxM];
    int tot=0;
    int ans;
    void build(int n,int num){
        for(int i=0;i<=num;i++){
            L[i]=i-1;R[i]=i+1;
            U[i]=i;D[i]=i;
            Size[i]=0;
        }
        L[0]=num;
        R[num]=0;
        tot=num;
        for(int i=1;i<=n;i++)
            First[i]=-1;
    }
    void insert(int r,int c){
        Row[++tot]=r; Col[tot]=c;
        D[tot]=D[c]; U[D[c]]=tot;
        U[tot]=c; D[c]=tot;
        if(First[r]<0){
            L[tot]=tot;
            R[tot]=tot;
            First[r]=tot;
        }
        else{
            R[tot]=R[First[r]];
            L[R[First[r]]]=tot;
            L[tot]=First[r];
```

```
36              R[First[r]]=tot;
37          }
38          Size[c]++;
39      }
40      void remove(int c){
41          L[R[c]]=L[c];
42          R[L[c]]=R[c];
43          for(int i=D[c];i!=c;i=D[i]){
44              for(int j=R[i];j!=i;j=R[j]){
45                  U[D[j]]=U[j];
46                  D[U[j]]=D[j];
47                  Size[Col[j]]--;
48              }
49          }
50      }
51      void resume(int c){
52          for(int i=U[c];i!=c;i=U[i]){
53              for(int j=L[i];j!=i;j=L[j]){
54                  D[U[j]]=j;
55                  U[D[j]]=j;
56                  Size[Col[j]]++;
57              }
58          }
59          R[L[c]]=c;
60          L[R[c]]=c;
61      }
62      void dance(int depth){
63          if(depth>=ans)
64              return ;
65          if(R[0]==0){
66              ans=depth;
67              return ;
68          }
69          int c=R[0];
70          for(int i=R[0];i!=0;i=R[i])
71              if(Size[i]<Size[c])
72                  c=i;
```

```
73          remove(c);
74          for(int i=D[c];i!=c;i=D[i]){
75              for(int j=R[i];j!=i;j=R[j])
76                  remove(Col[j]);
77              dance(depth+1);
78              for(int j=L[i];j!=i;j=L[j])
79                  resume(Col[j]);
80          }
81          resume(c);
82      }
83  }dlx;
84  int n,m,p;
85  void solve()
86  {
87      cin>>n>>m>>p;
88      dlx.build(p,n*m);
89      for(int i=1;i<=p;i++)
90      {
91          int x,y,_x,_y;
92          cin>>x>>y>>_x>>_y;
93          for(int I=x+1;I<=_x;I++)
94              for(int J=y+1;J<=_y;J++)
95                  dlx.insert(i,(I-1)*m+J);
96      }
97      dlx.ans=inf;
98      dlx.dance(0);
99      if(dlx.ans==inf)cout<<-1<<endl;
100     else cout<<dlx.ans<<endl;
101 }
```

3.11 最小树形图

```
#include<bits/stdc++.h>
using namespace std;
#define endl ”\n”
const int INF=2e9+7;
const int maxn=1e6+100;
struct node{
    int x,y,w;
}e[maxn];
int vis[maxn];//标记是否走过
int pre[maxn];//前驱节点
int inv[maxn];//节点权值
int id[maxn];//新的标号
int n,m,root;
int Edmonds()
{
    int ans=0;
    while(1)
    {
        /**********1.找最短弧集合**********/
        //初始化
        for(int i=1;i<=n;i++)
            inv[i]=INF;
        for(int i=1;i<=m;i++)
        {
            int x=e[i].x;
            int y=e[i].y;
            //找到每个点的最小入边 以及前驱
            if(x!=y&&e[i].w<inv[y])
                inv[y]=e[i].w,pre[y]=x;
        }
        //如果一个点不是根 并且没有入边则不是树形图
        for(int i=1;i<=n;i++)
            if(i!=root&&inv[i]==INF)
                return -1;
        int cnt=0;
```

```
36          //初始化标记
37          for(int i=1;i<=n;i++)
38              vis[i]=id[i]=0;
39          /*******2.判断集合E中有没有有向环，如果有转步骤3，
                否则转4*********/
40          for(int i=1;i<=n;i++)
41          {
42              if(i==root)continue;
43              ans+=inv[i];
44              int v=i;
45              while(vis[v]!=i&&!id[v]&&v!=root)
46              {
47                  vis[v]=i;
48                  v=pre[v];
49              }
50              /**********3.收缩G中的有向环*********/
51              if(!id[v]&&v!=root)
52              {
53                  id[v]=++cnt;
54                  for (int u=pre[v];u!=v;u=pre[u])
55                      id[u]=cnt;
56              }
57          }
58          if(cnt==0)break;
59          for(int i=1;i<=n;i++)
60              if(!id[i])id[i]=++cnt;
61          for(int i=1;i<=m;i++)
62          {
63              int u=e[i].x;
64              int v=e[i].y;
65              e[i].x=id[u];
66              e[i].y=id[v];
67              if(id[u]!=id[v])
68                  e[i].w-=inv[v];
69          }
70          root=id[root];
71          n=cnt;
```

```
72      }
73      return ans;
74  }
75  void solve()
76  {
77      cin>>n>>m>>root;
78      for(int i=1;i<=m;i++)
79          cin>>e[i].x>>e[i].y>>e[i].w;
80      cout<<Edmonds()<<endl;
81  }
82  signed main(){
83          solve();
84      return 0;
85  }
```

3.12 二维数点

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6 + 10, MAX = 1e6;

struct query{
    int r, x, y, id;
    bool operator < (const query &A)const{
        return r < A.r;
    }
}q[N * 2];

int a[N], ans[N], tr[MAX + 10];

void update(int r){
    for(int i = r; i <= MAX; i += (i & -i)) tr[i] ++;
}

int get_sum(int r){
    int ans = 0;
    for(int i = r; i >= 1; i -= (i & -i)) ans += tr[i];
    return ans;
}

int main(){
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);

    int n, m;
    cin >> n >> m;
    for(int i = 1; i <= n; i ++) cin >> a[i];
    for(int i = 1; i <= m; i ++){
        int l, r, x, y;
        cin >> l >> r >> x >> y;
        q[i] = {l - 1, x, y, i};
        q[i + m] = {r, x, y, i};
```

```
36          /*
37              当以矩阵形式给出时
38              cin >> x1 >> y1 >> x2 >> y2;
39              x1 ++, y1 ++, x2 ++, y2 ++; // 离散至大于0
40              q[i] = {x1 - 1, y1, y2, i};
41              q[i + m] = {x2, y1, y2, i};
42          */
43      }
44      m <<= 1;
45      sort(q + 1, q + 1 + m);
46
47      for(int i = 1, c = 0; i <= m; i ++){
48          auto [r, x, y, id] = q[i];
49          while(c < r) update(a[++ c]);
50          ans[id] = get_sum(y) - get_sum(x - 1) - ans[id];
51          /*
52              当以二维点坐标给出时
53              auto [x, y1, y2, id] = q[i];
54              while(j < n && p[j + 1].x <= x) update(p[++ j
                    ].y);
55              ans[id] = get_sum(y2) - get_sum(y1 - 1) - ans[
                    id];
56          */
57      }
58
59      m >>= 1;
60      for(int i = 1; i <= m; i ++){
61          cout << ans[i] << "\n";
62      }
63      return 0;
64  }
```

3.13 莫队

3.13.1 带修莫队

莫队是离线算法，因此当题目加入修改操作后一般认为强制在线不可做，但有部分题目时间复杂度允许的情况下能用带修莫队解决。

关键点在于再加入一维时间 t，一次修改相当于时间变化一次。相比于普通莫队在四种变化基础上 $[l,r+1]$，$[l,r-1]$，$[l+1,r]$，$[l-1,r]$ 再加入时间 t，$[l,r,t-1]$，$[l,r,t+1]$。

操作上并无太多变化，主要难点还在如何计算分块大小确保时间复杂度上。

时间复杂度：

设数组大小与询问为同一量级 n，修改次数为 t，分块大小为 $\sqrt[4]{n^3t}$ 时最优，总体复杂度为 $O(n^{\frac{5}{3}})$。

```
1  //P1903 [国家集训队] 数颜色 / 维护队列 https://www.luogu.
       com.cn/problem/P1903
2  // 1.每次询问区间 [l, r] 数的种类 2.修改第 x 个数为 y
3  #include <bits/stdc++.h>
4  using namespace std;
5
6  const int N = 140000, S = 1e6 + 10;
7
8  int n, m, mq, mc, len; // mq:询问次数 mc 修改次数 len: 分
       块大小
9
10 struct query{
11     int l, r, t, id;
12 }q[N];
13
14 struct modify{
15     int p, c;
16 }c[N];
17
18 int get(int x){ return x / len; } // 分块
19 bool cmp(query &A, query &B){
20     int al = get(A.l), ar = get(A.r);
21     int bl = get(B.l), br = get(B.r);
22     if(al != bl) return al < bl;
23     if(ar != br) return ar < br;
```

```
24      return A.t < B.t;
25  }
26
27  int a[N], cnt[S], ans[N], moans;
28  void add(int x){
29      if(++ cnt[x] == 1) moans ++;
30  }
31  void sub(int x){
32      if(-- cnt[x] == 0) moans --;
33  }
34
35  int main(){
36      ios::sync_with_stdio(false);
37      cin.tie(0); cout.tie(0);
38
39      cin >> n >> m;
40      for(int i = 1; i <= n; i ++) cin >> a[i];
41
42      moans = mc = mq = 0;
43      for(int i = 1; i <= m; i ++){
44          string op; int x, y;
45          cin >> op >> x >> y;
46          if(op[0] == 'Q') q[++ mq] = {x, y, mc, mq};
47          else c[++ mc] = {x, y};
48      }
49
50      len = cbrt((double)n * max(1, mc)) + 1; // cbrt:根号下
          3次方
51      sort(q + 1, q + 1 + mq, cmp);
52
53      for(int i = 1, L = 1, R = 0, t = 0; i <= mq; i ++){
54          auto [l, r, tm, id] = q[i];
55          while(R < r) add(a[++ R]);
56          while(R > r) sub(a[R --]);
57          while(L < l) sub(a[L ++]);
58          while(L > l) add(a[-- L]);
59          int w1 = t < tm, w2 = t > tm;
```

```
60          while(t != tm){ // 当前时间不是目标时间
61              t += w1;
62              if(L <= c[t].p && c[t].p <= R){ // 修改位置在
                    当前所求区间中
63                  sub(a[c[t].p]);
64                  add(c[t].c);
65              }
66              swap(a[c[t].p], c[t].c); // 交换这次修改的值,
                    下次反向移动时能修改回来
67              t -= w2;
68          }
69          ans[id] = moans;
70      }
71
72      for(int i = 1; i <= mq; i ++){
73          cout << ans[i] << "\n";
74      }
75      return 0;
76  }
```

3.13.2 回滚莫队/不删除莫队

当我们遇到新增操作简单，但删除操作比较困难时，例如当求区间最值时，新增可以 $O(1)$ 比较更新最值但删除时却不能确定最值是否改变。考虑回滚莫队/不删除莫队，既然维护删除比较困难我们就不维护了。

时间复杂度：$O(n\sqrt{n})$

证明：

分块大小 $len=\sqrt{n}$ 按左端点分块编号进行排序，再按右端点大小排序，每次询问将询问的左端点在同一块内的预处理出来 $[i,j]$，此时右端点保持升序，分情况处理。

1. 左右端点在同一块内

暴力求解，直接 $[l,r]$ 遍历一遍，因为在同一块内复杂度为块的大小 $\sqrt{n}$ 时间复杂为 $O(n\sqrt{n})$

2. 右端点和左端点不在同一块内

右端点 $r>right$ （块的右端点），右指针 R 不返回的从 $right$ 往右扫，执行 ‘add(++ R)‘ 操作，此时我们只处理好了右端点，将当前答案 res 备份 $backup=res$。对于左端点每次询问左指针 L 都从 $right+1$ 开始向左扫执行 ‘add(- L)‘，因为左端点肯定都是在一块内，处理完一次询问后将答案 res 回溯至备份 $backup$ 对于下一次询问重复这个过程。

因为右指针是不返回的往右扫时间复杂度 $O(n)$，左指针每次询问都是重新从 $right$ 出发，因为只在块内进行时间复杂度 $O(\sqrt{n})$ 相乘即为这部分时间复杂度。

综上 1,2 情况的复杂度相同，因此算法复杂度为 $O(n\sqrt{n})$。

```cpp
// https://www.luogu.com.cn/problem/AT_joisc2014_c
// 事件a_i的重要度为 a_i * Ta（a_i出现的次数）多次询问区间 [l, r] 重要度最大的事件的重要度
#include <bits/stdc++.h>
using namespace std;

#define ll long long
const int N = 1e5 + 10;

struct query{
    int l, r, id;
}q[N];

int n, m, len, a[N], t[N], cnt[N];

int get(int x){ return x / len; }
```

```
16 bool cmp(query& A, query& B){
17     int al = get(A.l), bl = get(B.l);
18     if(al != bl) return al < bl;
19     return A.r < B.r;
20 }
21
22 void add(int x, ll& res){
23     cnt[x] ++;
24     res = max(res, 1LL * cnt[x] * t[x]);
25 }
26
27 ll ans[N];
28 int main(){
29     ios::sync_with_stdio(false);
30     cin.tie(0); cout.tie(0);
31
32     int n, m;
33     cin >> n >> m;
34     for(int i = 1; i <= n; i ++){
35         cin >> a[i];
36         t[i] = a[i];
37     }
38     sort(t + 1, t + 1 + n);
39     int tn = unique(t + 1, t + 1 + n) - (t + 1);
40     for(int i = 1; i <= n; i ++){
41         a[i] = lower_bound(t + 1, t + 1 + tn, a[i]) - t;
42     }
43
44     for(int i = 1; i <= m; i ++){
45         int l, r;
46         cin >> l >> r;
47         q[i] = {l, r, i};
48     }
49
50     len = sqrt(n);
51     sort(q + 1, q + 1 + m, cmp);
52
```

```
53      for(int i = 1; i <= m; ){
54          int j = i;
55          while(j + 1 <= m && get(q[j + 1].l) == get(q[i].l)
               ) j ++; // 寻找在相同块内的
56
57          // 暴力求块内的询问
58          int right = get(q[i].l) * len + len - 1; // 确定块
               内的右边界
59          while(i <= j && q[i].r <= right){
60              ll res = 0;
61              auto [l, r, id] = q[i ++];
62              for(int k = l; k <= r; k ++) add(a[k], res);
                   // 直接从 [l, r] 全部计算
63              ans[id] = res;
64              for(int k = l; k <= r; k ++) cnt[a[k]] --;
65          }
66
67          // 处理r在块外的询问
68          ll res = 0;
69          int R = right, L = R + 1; // 左右指针
70          while(i <= j){
71              auto [l, r, id] = q[i ++];
72              while(R < r) add(a[++ R], res); // 不返回的右
                   移
73              ll backup = res; // 备份
74              while(L > l) add(a[-- L], res); // 每次都从
                   right + 1 开始右移
75              ans[id] = res;
76              while(L < right + 1) cnt[a[L ++]] --;
77              res = backup;    // 回溯
78          }
79          memset(cnt, 0, sizeof cnt);
80          // for(int k = right + 1; k <= q[i - 1].r; k ++)
               cnt[a[k]] = 0; 不同的清空方式
81      }
82
83      for(int i = 1; i <= m; i ++){
```

```
84          cout << ans[i] << "\n";
85      }
86      return 0;
87  }
```

3.13.3 树上莫队

对于树上路径 $[u,v]$ 的查询可以通过欧拉序/dfs 序转化为区间查询。

dfs 序就是根据 dfs 遍历的顺序写下每个点的编号，第一次遍历写一次，退出时再写一次。例如上图的一种可能的 dfs 序为：

‘1 2 2 3 5 5 6 6 7 7 3 4 8 8 4 1‘，每个点在 dfs 序中都出现两次我们将其称为第一次出现 $fist_u$ 和最后一次出现 $last_u$。

我们将询问分情况讨论，查询 u,v（默认 u 在 dfs 序中出现较早）

1. u 是 v 的祖先

找到 $first_u$ 和 $first_v$ 在此区间只出现一次的即为路径上的点，出现两次说明先访问了无关的子树之后又推出来。

2. u 和 v 的最近公共祖先为 lca 且 $lca \neq u, lca \neq v$

找到 $last_u$ 和 $first_v$ 在此区间只出现一次的即为路径上的点，因为 u,v 在同一棵子树此路径上不会包含最近公共祖先，单独加上即可。

```cpp
// https://www.acwing.com/problem/content/2536/ 2534. 树上计数2
// 多次询问树上两点 [u, v] 之间路径上不同点权的种类
#include <bits/stdc++.h>
using namespace std;

const int N = 5e4 + 10, M = 1e5 + 10;

int a[N], t[N];
vector<int> g[N];

int len, ans[M];
struct query{
    int l, r, fat, id;
}q[M];

int get(int x){ return x / len; }
bool cmp(query& A, query& B){
    int al = get(A.l), bl = get(B.l);
    if(al != bl) return al < bl;
    return A.r < B.r;
}

```

```
23 int dep[N], f[N][20], dfn[N * 2], fi[N], la[N], tot;
24 void dfs(int u, int fa){
25     dfn[++ tot] = u;
26     fi[u] = tot;
27 
28     f[u][0] = fa;
29     dep[u] = dep[fa] + 1;
30     for(int i = 1; (1 << i) <= dep[u]; i ++){
31         f[u][i] = f[f[u][i - 1]][i - 1];
32     }
33 
34     for(int v : g[u]){
35         if(v != fa) dfs(v, u);
36     }
37     dfn[++ tot] = u;
38     la[u] = tot;
39 }
40 
41 int LCA(int u, int v){
42     if(dep[u] < dep[v]) swap(u , v);
43     int d = dep[u] - dep[v]; // 记录深度差
44     for(int i = 0; i <= 20; i ++){
45         if((1 << i) & d) u = f[u][i];// 如果深度差包含2^i
              那么就向上跳2^i
46     }
47     if(u == v) return u; // 如果相同说明lca是u
48     for(int i = 19; i >= 0; i --){
49         if(f[u][i] != f[v][i]){
50             u = f[u][i];
51             v = f[v][i];
52         }
53     }
54     return f[u][0];
55 }
56 
57 int moans, cnt[N], st[N]; // st:出现次数，出现两次的抵消
     cnt:计数
```

```
58 void add(int x){
59     st[x] ^= 1;
60     if(st[x]) moans += (++ cnt[a[x]] == 1);
61     else moans -= (-- cnt[a[x]] == 0);
62 }
63 
64 int n, m;
65 int main(){
66     cin >> n >> m;
67     for(int i = 1; i <= n; i ++){
68         cin >> a[i];
69         t[i] = a[i];
70     }
71     sort(t + 1, t + 1 + n);
72     int tn = unique(t + 1, t + 1 + n) - (t + 1);
73     for(int i = 1; i <= n; i ++){
74         a[i] = lower_bound(t + 1, t + 1 + tn, a[i]) - t;
75     }
76 
77     for(int i = 1; i < n; i ++){
78         int u, v;
79         cin >> u >> v;
80         g[u].push_back(v);
81         g[v].push_back(u);
82     }
83 
84     dfs(1, 0);
85 
86     for(int i = 1; i <= m; i ++){
87         int u, v;
88         cin >> u >> v;
89         if(fi[u] > fi[v]) swap(u, v);
90         int lca = LCA(u, v);
91         if(u == lca) q[i] = {fi[u], fi[v], 0, i};
92         else q[i] = {la[u], fi[v], lca, i};
93     }
94 
```

```
95      len = sqrt(n);
96      sort(q + 1, q + 1 + m, cmp);
97
98      for(int i = 1, L = 1, R = 0; i <= m ;i ++){
99          auto [l, r, fat, id] = q[i];
100         while(L < l) add(dfn[L ++]);
101         while(L > l) add(dfn[-- L]);
102         while(R < r) add(dfn[++ R]);
103         while(R > r) add(dfn[R --]);
104         bool ad = (fat && !cnt[a[fat]]);
105         ans[q[i].id] = moans + ad;
106     }
107     for(int i = 1; i <= m; i ++) {
108         cout << ans[i] << "\n";
109     }
110     return 0;
111 }
```

3.14 可持久化 kmp

在原来朴素 kmp 的基础上提供了删除维护操作

```
int ne[N];
int pr[N];
char st[N];
int tp;
void add(char c)
{
    int j=tp;
    while(j&&st[ne[j]+1]!=c) j=pr[j];
    st[++tp]=c;
    j=ne[j]+1;
    if(tp==1) ne[1]=pr[1]=0;
    else if(st[j]==c)
    {
        ne[tp]=j;
        if(st[ne[j]+1]==st[j+1]) pr[tp]=pr[j];
        else pr[tp]=j;
    }
    else ne[tp]=pr[tp]=0;

}
```

3.15 map 排序

可以根据 map key-val 来进行排序具体操作

```
map<（key类型）,(val类型),decltype(&cmp)>mp{&cmp}; //cmp为定义的排序规则
 for(map<point,pair<double,double>,decltype(&cmp)>::iterator it=mp.begin();it!=mp.end();it++)//11都能用
{
    ;
}

class cmp{//换struct一样
    public:
    bool operator()(const point &x,const point &y) const//排序方法
    {
        return (x^y)>0;
    }
};
map<point,pair<double,double>,cmp>mp;
for(map<point,pair<double,double>,cmp>::iterator it=mp.begin();it!=mp.end();it++)//迭代器遍历
{
    ;
}
```