Algorithm Templates

BiteTheDust

November 1, 2019

Contents

1 动态规划

1.1 数位 dp

```cpp
#include <bits/stdc++.h>
using namespace std;
long long i,n,m,dp[20][300],a[20],T;
long long dfs(int len,bool maxi,int sum)
{
    if(dp[len][sum]!=-1&&maxi==0)return dp[len][sum];
    long long cnt=0;
    if(!len)return sum%10==0;
    int maxn=maxi?a[len]:9;
    for(int i=0;i<=maxn;i++)cnt+=dfs(len-1,maxi&&i==a[len],sum+i);
    return maxi?cnt:dp[len][sum]=cnt;
}
long long div(long long tmp)
{
    memset(a,0,sizeof(a));
    int p=0;
    while(tmp)a[++p]=tmp%10,tmp/=10;
    return dfs(p,1,0);
}
int main()
{
    memset(dp,-1,sizeof(dp));
    scanf("%lld",&T);
    for(i=1;i<=T;i++)
    {
        scanf("%lld %lld",&n,&m);
        printf("Case #%lld: ",i);
        if(n)printf("%lld\n",div(m)-div(n-1));
        else printf("%lld\n",div(m));
    }
    return 0;
}
```

1.2 SOSdp

```cpp
for(int mask = 0;mask < (1<<N); ++mask){
    for(int i = 0;i < (1<<N); ++i){
        if((mask&i) == i){
            F[mask] += A[i];
        }
    }
}
4^N
// iterate over all the masks
for (int mask = 0; mask < (1<<n); mask++){
    F[mask] = A[0];
    // iterate over all the subsets of the mask
    for(int i = mask; i > 0; i = (i-1) & mask){
        F[mask] += A[i];
    }
}
3^N
//iterative version
for(int mask = 0; mask < (1<<N); ++mask){
    dp[mask][-1] = A[mask]; //handle base case separately (leaf states)
    for(int i = 0;i < N; ++i){
        if(mask & (1<<i))
            dp[mask][i] = dp[mask][i-1] + dp[mask^(1<<i)][i-1];
        else
            dp[mask][i] = dp[mask][i-1];
    }
    F[mask] = dp[mask][N-1];
}

//memory optimized, super easy to code.
for(int i = 0; i<(1<<N); ++i)
    F[i] = A[i];
for(int i = 0;i < N; ++i) for(int mask = 0; mask < (1<<N); ++mask){
    if(mask & (1<<i))
        F[mask] += F[mask^(1<<i)];
}
N*2^N
```

1.3 斜率优化 dp

```
#include<bits/stdc++.h>
using namespace std;
int i,i0,i1,n,k,ans;
long long dp[2005][5005];
struct node
{
    long long w,h;
}a[5005];
struct line
{
    long long k,b;
};
long long cal(line a,long long pos){return a.k*pos+a.b;}
double cross(line a,line b)
{
    return ((double)a.b-b.b)/(b.k-a.k);
}
bool cmp(node a,node b)
{
    return a.h<b.h;
}
deque<line>q;
map<long long,long long>mp;
int main()
{
    scanf("%d %d",&n,&k);
    long long sum=0;
    for(i=1;i<=n;i++)scanf("%lld %lld",&a[i].w,&a[i].h),mp[a[i].h]+=a[i].w,sum+=a[i].h*a[i].w;
    i=1;
    for(auto it:mp)
    {
        a[i].h=it.first;
        a[i].w=it.second;
        i++;
    }
    n=mp.size();
    sort(a+1,a+1+n,cmp);
    for(i=1;i<=k;i++)
    {
        long long sumw=0;
        while(!q.empty())q.pop_back();
        for(i0=1;i0<=n;i0++)
        {
            sumw+=a[i0].w;
            while(q.size()>1)
            {
                line x=q.back();
                q.pop_back();
                line y=q.back();
                if(cross(x,{a[i0].h,dp[i-1][i0-1]-(sumw-a[i0].w)*a[i0].h})<=cross(y,x))
                {
                    continue;
                }
                else
                {
                    q.push_back(x);
                    break;
                }
            }
            if(!(i==1&&i0!=1))q.push_back({a[i0].h,dp[i-1][i0-1]-(sumw-a[i0].w)*a[i0].h});
            while(q.size()>1)
            {
                line x=q.front();
                q.pop_front();
                line y=q.front();
                if(cal(x,sumw)<=cal(y,sumw))continue;
                else
                {
                    q.push_front(x);
                    break;
                }
            }
            dp[i][i0]=cal(q.front(),sumw);
        }
    }
    printf("%lld\n",sum-dp[k][n]);
    return 0;
}
```

1.4 可逆背包

```
#include<bits/stdc++.h>
using namespace std;
const int mod=1e9+7;
int i,i0,n,m,dp[1005];
void add_dp(int x,int cnt,int n)
{
    for(int i=n;i>=(cnt+1)*x;i--)dp[i]-=dp[i-(cnt+1)*x],dp[i]+=mod,dp[i]%=mod;
    for(int i=x;i<=n;i++)dp[i]+=dp[i-x],dp[i]%=mod;
}
void del_dp(int x,int cnt,int n)
{
    for(int i=n;i>=x;i--)dp[i]-=dp[i-x],dp[i]+=mod,dp[i]%=mod;
    for(int i=(cnt+1)*x;i<=n;i++)dp[i]+=dp[i-(cnt+1)*x],dp[i]%=mod;
}
int main()
{
    scanf("%d",&n);
    dp[0]=1,m=0;
    while(n--)
    {
        int op,x,y;
        scanf("%d %d %d",&op,&x,&y);//添加价值为x的y个物品 在1000范围内
        if(op==1)add_dp(x,y,1000);
        if(op==2)del_dp(x,y,1000);
        for(i=0;i<=1000;i++)m^=dp[i];
    }
    printf("%d\n",m);
    return 0;
}
```

1.5 树形 dp

```
#include<bits/stdc++.h>
using namespace std;
int i,i0,n,m,dp[1005][2],ans;
vector<int>mp[1005];
void dfs(int now,int root)
{
    for(int i:mp[now])
    {
        if(i==root)continue;
        dfs(i,now);
        if(dp[i][0]+1>dp[now][0])dp[now][1]=dp[now][0],dp[now][0]=dp[i][0]+1;
        else if(dp[i][0]+1>dp[now][1])dp[now][1]=dp[i][0]+1;
    }
}
```

```
void solve(int now,int root);
void change_root(int nowa,int nowb)
{
    int tmpa[2]={dp[nowa][0],dp[nowa][1]},tmpb[2]={dp[nowb][0],dp[nowb][1]};
    int son=dp[nowa][dp[nowb][0]+1==dp[nowa][0]]+1;
    if(son>dp[nowb][0])dp[nowb][1]=dp[nowb][0],dp[nowb][0]=son;
    else if(son>dp[nowb][1])dp[nowb][1]=son;
    solve(nowb,nowa);
    dp[nowb][0]=tmpb[0],dp[nowb][1]=tmpb[1];
}
void solve(int now,int root)
{
    ans=max(ans,dp[now][0]);
    for(auto i:mp[now])
    {
        if(i==root)continue;
        change_root(now,i);
    }
}
int main()
{
    int T;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d",&n);
        memset(dp,0,sizeof(dp));
        for(i=1;i<=n;i++)mp[i].clear();
        ans=0;
        for(i=1;i<n;i++)
        {
            int a,b;
            scanf("%d %d",&a,&b);
            mp[a].push_back(b);
            mp[b].push_back(a);
        }
        dfs(1,-1);
        solve(1,-1);
        printf("%d\n",n*2-ans-2);
    }
    return 0;
}
```

1.6 UpDowndp

```
#include<bits/stdc++.h>
using namespace std;
int i,i0,n,m,v[100005],dp[100005],ans[100005];
struct node
{
    int to,v;
};
```

```
vector<node>mp[100005];
int dfs(int now,int root)
{
    if(dp[now])return dp[now];
    int cnt=v[now];
    for(int i=0;i<mp[now].size();i++)
    {
        if(mp[now][i].to==root)continue;
        cnt=max(cnt,dfs(mp[now][i].to,now)+v[now]-mp[now][i].v);
    }
    return dp[now]=cnt;
}
void dfs0(int now,int root,int maxn1,int maxn2,int maxi1,int maxi2)
{
    if(maxn1<0)maxn1=0;
    if(maxn2<0)maxn2=0;
    maxn1+=v[now],maxn2+=v[now];
    if(maxi1==now)maxi1=maxi2,maxn1=maxn2;
    ans[now]=max(dp[now],maxn1);
    for(int i=0;i<mp[now].size();i++)
    {
        if(mp[now][i].to==root)continue;
        if(dp[mp[now][i].to]+v[now]-mp[now][i].v>maxn1)maxn2=maxn1,maxn1=dp[mp[now][i
            ].to]+v[now]-mp[now][i].v,maxi1=mp[now][i].to;
        else if(dp[mp[now][i].to]+v[now]-mp[now][i].v>maxn2)maxn2=dp[mp[now][i].to]+v
            [now]-mp[now][i].v,maxi2=mp[now][i].to;
    }
    for(int i=0;i<mp[now].size();i++)
    {
        if(mp[now][i].to==root)continue;
        dfs0(mp[now][i].to,now,maxn1-mp[now][i].v,maxn2-mp[now][i].v,maxi1,maxi2);
    }
    return;
}
int main()
{
    while(scanf("%d",&n)!=EOF)
    {
        for(i=1;i<=n;i++)mp[i].clear(),dp[i]=0;
        for(i=1;i<=n;i++)scanf("%d",&v[i]);
        for(i=1;i<n;i++)
        {
            int a,b,c;
            scanf("%d %d %d",&a,&b,&c);
            mp[a].push_back({b,c});
            mp[b].push_back({a,c});
        }
        dfs(1,-1);
        dfs0(1,-1,0,0,0,0);
        for(i=1;i<=n;i++)
        {
            printf("%d",ans[i]);
            if(i==n)printf("\n");
            else printf(" ");
        }
    }
    return 0;
}
```

1.7 换根 dp

```
#include<bits/stdc++.h>
using namespace std;
int i,i0,n,m,v[100005],dp[100005],ans[100005];
struct node{int to,d;};
vector<node>mp[100005];
void dfs(int now,int root)
{
    dp[now]=v[now];
    for(auto it:mp[now])
    {
        if(it.to==root)continue;
        dfs(it.to,now);
        dp[now]=max(dp[now],dp[it.to]+v[now]-it.d);
    }
}
void solve(int now,int root);
void change_root(int nowa,int nowb,int d)
{
    int tmpa=dp[nowa],tmpb=dp[nowb];
    dp[nowa]=v[nowa];
    for(auto it:mp[nowa])
    {
        if(it.to==nowb)continue;
        dp[nowa]=max(dp[nowa],dp[it.to]+v[nowa]-it.d);
    }
    dp[nowb]=max(dp[nowb],dp[nowa]+v[nowb]-d);
    solve(nowb,nowa);
    dp[nowa]=tmpa,dp[nowb]=tmpb;
}
void solve(int now,int root)
{
    ans[now]=dp[now];
    for(int i=0;i<mp[now].size();i++)
    {
        if(mp[now][i].to==root)continue;
        change_root(now,mp[now][i].to,mp[now][i].d);
    }
}
int main()
{
    while(scanf("%d",&n)!=EOF)
    {
        for(i=1;i<=n;i++)mp[i].clear(),dp[i]=0;
        for(i=1;i<=n;i++)scanf("%d",&v[i]);
        for(i=1;i<n;i++)
        {
            int a,b,c;
            scanf("%d %d %d",&a,&b,&c);
            mp[a].push_back({b,c});
            mp[b].push_back({a,c});
        }
        dfs(1,-1);
        solve(1,-1);
        for(i=1;i<=n;i++)
        {
            printf("%d",ans[i]);
            if(i==n)printf("\n");
            else printf(" ");
        }
    }
    return 0;
}
```

1.8 基环树 dp

```
#include<bits/stdc++.h>
using namespace std;
long long i,i0,n,m,v[1000005],pre[1000005],dp[1000005][2][2],ans;
vector<int>mp[1000005];
vector<pair<int,int> >q;
int fin(int x)
{
    int tmp=x,tmp2;
    while(pre[tmp]!=tmp)tmp=pre[tmp];
    while(x!=tmp)tmp2=pre[x],pre[x]=tmp,x=tmp2;
    return tmp;
}
void uni(int x,int y){if(fin(x)!=fin(y))pre[fin(y)]=fin(x);}
void dfs(int now,int ex,int root)
{
    dp[now][0][1]=dp[now][1][1]=v[now];
    for(auto i:mp[now])
    {
        dfs(i,ex,root);
        for(int i0=0;i0<2;i0++)
        {
            dp[now][i0][0]+=max(dp[i][i0][0],dp[i][i0][1]);
            dp[now][i0][1]+=dp[i][i0][0];
        }
    }
    if(ex==now)dp[now][1][0]=dp[now][0][1]=0;
}
int main()
{
    scanf("%lld",&n);
    ans=0;
    for(i=1;i<=n;i++)pre[i]=i;
    for(i=1;i<=n;i++)
    {
        scanf("%lld %lld",&v[i],&m);
        if(fin(m)==fin(i))q.push_back({i,m});
        else uni(m,i),mp[m].push_back(i);
    }
    for(i=0;i<q.size();i++)
    {
        dfs(q[i].first,q[i].second,q[i].first);
        ans+=max(dp[q[i].first][1][0],max(dp[q[i].first][0][0],dp[q[i].first][0][1]))
            ;
    }
    printf("%lld\n",ans);
    return 0;
}
```

2 字符串

2.1 Manacher

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=100005;
char ma[maxn*2],s[maxn];
int mp[maxn*2];
void Manacher(char *s,int len)
{
    int l=0;
    ma[l++]='$',ma[l++]='#';
    for(int i=0;i<len;i++)ma[l++]=s[i],ma[l++]='#';
    ma[l]=0;
    int mx=0,id=0;
    for(int i=0;i<l;i++)
    {
        mp[i]=mx>i?min(mp[id*2-i],mx-i):1;
        while(ma[i+mp[i]]==ma[i-mp[i]])mp[i]++;
        if(i+mp[i]>mx)mx=i+mp[i],id=i;
    }
}
int main()
{
    while(scanf("%s",s)!=EOF)
    {
        int len=strlen(s);
        Manacher(s,len);
        int ans=0;
        for(int i=0;i<2*len+2;i++)ans=max(ans,mp[i]-1);
        printf("%d\n",ans);
    }
}
```

2.2 KMP

```
#include<bits/stdc++.h>
using namespace std;
int a[1000005],b[10005],nex[10005],i,i0,n,m,T,ans,k;
void cal_nex(int *str,int len)
{
    nex[0]=-1;
    for (int q=1,k=-1;q<len;q++)
    {
        while(k!=-1&&str[k+1]!=str[q])k=nex[k];
        if (str[k+1]==str[q])k++;
        nex[q]=k;
    }
}
int KMP(int *str,int slen,int *ptr,int plen)
{
    cal_nex(ptr,plen);
    for(int i=0,k=-1;i<slen;i++)
    {
        while(k>-1&&ptr[k+1]!=str[i])k=nex[k];
        if(ptr[k+1]==str[i])k++;
        if(k==plen-1)return i-plen+2;
    }
    return -1;
}
int main()
{
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d %d",&n,&m);
        for(i=0;i<n;i++)scanf("%d",&a[i]);
        for(i=0;i<m;i++)scanf("%d",&b[i]);
        printf("%d\n",KMP(a,n,b,m));
    }
    return 0;
}
```

2.3 EXKMP

```
#include<bits/stdc++.h>
using namespace std;
int n,T,nex[200005],ex[200005];
char s[200005];
void cal_nex(char *str,int len)
{
    nex[0]=len,nex[1]=0;
    while(nex[1]+1<len&&str[nex[1]]==str[nex[1]+1])nex[1]++;
    for(int i=2,i0=1;i<len;i++)
    {
        if(i+nex[i-i0]<nex[i0]+i0)nex[i]=nex[i-i0];
```

```cpp
        else
        {
            nex[i]=max(0,nex[i0]+i0-i);
            while(i+nex[i]<len&&str[nex[i]]==str[i+nex[i]])nex[i]++;
            i0=i;
        }
    }
}
void EX_KMP(char *str,int slen,char *ptr,int plen)
{
    cal_nex(ptr,plen);
    ex[0]=0;
    while(ex[0]<slen&&ex[0]<plen&&str[ex[0]]==ptr[ex[0]])ex[0]++;
    for(int i=1,i0=0;i<slen;i++)
    {
        if(i+nex[i-i0]<ex[i0]+i0)ex[i]=nex[i-i0];
        else
        {
            ex[i]=max(0,ex[i0]+i0-i);
            while(i+ex[i]<slen&&ex[i]<plen&&str[i+ex[i]]==ptr[ex[i]])ex[i]++;
            i0=i;
        }
    }
}
int main()
{
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d %s",&n,s);
        EX_KMP(s,n,s,n);
        int ans=0;
        for(int i=0;i<n;i++)ans+=ex[i],ans%=10007;
        printf("%d\n",ans);
    }
    return 0;
}
```

2.4 字符串 hash

```cpp
#include<bits/stdc++.h>
using namespace std;
int i,i0,n,k;
char s[100005];
const int pr=769;
unsigned long long dic[100005],pre[100005];
unsigned long long get(int l,int r){return pre[r]-pre[l-1]*dic[r-l+1];}
map<unsigned long long,int>mp;
int main()
{
    scanf("%d %d %s",&n,&k,s+1);
    dic[0]=1;
    for(i=1;i<=n;i++)dic[i]=dic[i-1]*pr;
    for(i=1;i<=n;i++)pre[i]=pre[i-1]*pr+(s[i]-'a');
    for(i=1;i+k-1<=n;i++)mp[get(i,i+k-1)]++;
    int ans=0;
    for(auto i:mp)ans=max(ans,i.second);
    printf("%d\n",ans);
    return 0;
}
```

2.5 后缀自动机

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=1000005,ALP=26;
struct SuffixAutomaton
{
    int next[maxn*2][ALP],p,cnt[maxn*2];
    int fail[maxn*2];//表示节点i的后缀链接
    int len[maxn*2];//表示节点i表示的等价集合的最长子串长度
    int last;//指向新添加一个字母后所形成的最长子串所指向的节点
    void init()
    {
        last=p=0,fail[0]=-1;
        newnode(0);
    }
    int newnode(int l)
    {
        for(int i=0;i<ALP;i++)next[p][i]=-1;
        len[p]=l,cnt[p]=0;
        return p++;
    }
    void add(int c)
    {
        int cur=newnode(len[last]+1),t=last;
        cnt[cur]++;
        for(;t!=-1&&next[t][c]==-1;t=fail[t])next[t][c]=cur;
        if(t==-1)fail[cur]=0;
        else
        {
            int q=next[t][c];
            if(len[t]+1==len[q])fail[cur]=q;
            else
            {
                int clone=newnode(len[t]+1);
                fail[clone]=fail[q];
                for(int i=0;i<26;i++)next[clone][i]=next[q][i];
                for(;t!=-1&&next[t][c]==q;t=fail[t])next[t][c]=clone;
                fail[q]=fail[cur]=clone;
            }

        }
        last=cur;
    }
    int lcs(char* str,int slen)
    {
        int now=0,l=0,ans=0,sum=0;
        for(int i=0;i<slen;i++)
        {
            int c=str[i]-'a';
            while(now&&next[now][c]==-1)now=fail[now],l=len[now];
            if(next[now][c]!=-1)now=next[now][c],l++;
            ans=max(ans,l);
            sum+=ans;
        }
        return sum;
    }
    int rk[maxn*2];
    void sort()
    {
        for(int i=0;i<p;i++)rk[i]=i;
        std::sort(rk,rk+p,[&](int x,int y){return len[x]>len[y];});
        for(int i=0;i<p;i++)if(fail[rk[i]]!=-1)cnt[fail[rk[i]]]+=cnt[rk[i]];
    }
}sam;
char s[maxn];
int main()
{
    scanf("%s",s);
    sam.init();
    for(int i=0;s[i]!='\0';i++)sam.add(s[i]-'a');
    int ans=0;
    for(int i=0;i<sam.p;i++)if(sam.cnt[i]>1)ans=max(ans,sam.cnt[i]*sam.len[i]);
    printf("%d\n",ans);
    return 0;
}
```

2.6 维护 endpos

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=100005,ALP=26,lm=20;
int n,m;
namespace SegmentTree
{
    #define mid (l+r)/2
    int cnt=0;
    struct node
    {
        int lson,rson,sum;
    }tree[maxn*lm*2];
    int new_node()
    {
        int p=++cnt;
        tree[p].lson=tree[p].rson=0;
        tree[p].sum=0;
        return p;
    }
    void c_tree(int l,int r,int p,int x)
    {
        tree[p].sum++;
        if(l!=r)
        {
            if(x<=mid)
            {
                tree[p].lson=new_node();
                c_tree(l,mid,tree[p].lson,x);
            }
            else
            {
                tree[p].rson=new_node();
                c_tree(mid+1,r,tree[p].rson,x);
            }
        }
    }
    int m_tree(int l,int r,int pa,int pb)
    {
        if(pa&&pb)
        {
            int t=new_node();
            tree[t].lson=m_tree(l,mid,tree[pa].lson,tree[pb].lson);
            tree[t].rson=m_tree(mid+1,r,tree[pa].rson,tree[pb].rson);
            tree[t].sum=tree[pa].sum+tree[pb].sum;
            return t;
        }
        else return pa|pb;
    }
    int q_tree(int l,int r,int p,int k)
    {
        if(tree[p].sum<k)return -1;
        if(l==r)return l;
        else
        {
            if(tree[tree[p].lson].sum>=k)return q_tree(l,mid,tree[p].lson,k);
            else return q_tree(mid+1,r,tree[p].rson,k-tree[tree[p].lson].sum);
        }
    }
}
struct SuffixAutomaton
{
    int next[maxn*2][ALP],p;
    int root[maxn*2];
    int fail[maxn*2];//表示节点i的后缀链接
    int len[maxn*2];//表示节点i表示的等价集合的最长子串长度
    int last;//指向新添加一个字母后所形成的最长子串所指向的节点
    void init()
    {
        last=p=0,fail[0]=-1;
        SegmentTree::cnt=0;
        newnode(0);
    }
    int newnode(int l)
    {
        for(int i=0;i<ALP;i++)next[p][i]=-1;
        len[p]=l;
        root[p]=SegmentTree::new_node();
        return p++;
    }
    int d[maxn];
    void add(int c,int pos)
    {
        int cur=newnode(len[last]+1),t=last;
        d[pos]=cur;
        SegmentTree::c_tree(1,n,root[cur],pos);
        for(;t!=-1&&next[t][c]==-1;t=fail[t])next[t][c]=cur;
        if(t==-1)fail[cur]=0;
        else
        {
            int q=next[t][c];
            if(len[t]+1==len[q])fail[cur]=q;
            else
            {
                int clone=newnode(len[t]+1);
                fail[clone]=fail[q];
                for(int i=0;i<26;i++)next[clone][i]=next[q][i];
                for(;t!=-1&&next[t][c]==q;t=fail[t])next[t][c]=clone;
                fail[q]=fail[cur]=clone;
            }
        }
        last=cur;
    }
    int rk[maxn*2];
    int Fa[maxn*2][lm+1];
    void count()
    {
        for(int i=0;i<p;i++)rk[i]=i;
        sort(rk,rk+p,[&](int x,int y){return len[x]>len[y];});
        for(int i=0;i<p-1;i++)root[fail[rk[i]]]=SegmentTree::m_tree(1,n,root[fail[rk[
            i]]],root[rk[i]]);
        for(int i=0;i<p;i++)Fa[i][0]=fail[i];
        for(int k=1;k<=lm;k++)for(int i=0;i<p;i++)Fa[i][k]=Fa[Fa[i][k-1]][k-1];
    }
    int find(int l,int r)
    {
        int u=d[r],length=r-l+1;
        if(len[fail[u]]+1>length)
        {
            for(int k=lm;k>=0;k--)if(len[fail[Fa[u][k]]]+1>length)u=Fa[u][k];
            u=fail[u];
        }
        return root[u];
    }
}sam;
char s[maxn];
int main()
{
    int T;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d %d %s",&n,&m,s);
        sam.init();
        for(int i=0;i<n;i++)sam.add(s[i]-'a',i+1);
        sam.count();
        while(m--)
        {
            int l,r,k;
            scanf("%d %d %d",&l,&r,&k);
            int z=SegmentTree::q_tree(1,n,sam.find(l,r),k);
            if(z!=-1)z+=l-r;
            printf("%d\n",z);
        }
    }
    return 0;
}
```

2.7 回文自动机

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=300005,ALP=26;
string t;
struct PalindromeAutomaton
{
    int next[maxn][ALP],fail[maxn],s[maxn],n,p;
    int cnt[maxn];//表示节点i表示的本质不同的串的个数（建树时求出的不是完全的，最后count()
        函数跑一遍以后才是正确的）
    int num[maxn];//表示以节点i表示的最长回文串的最右端点为回文串结尾的回文串个数
    int len[maxn];//len[i]表示节点i表示的回文串的长度（一个节点表示一个回文串），当前有p个
        节点
    int last;//指向新添加一个字母后所形成的最长回文串表示的节点
    //num[last]为最后添加的一个字母所增加的回文子串个数。
    int newnode(int l)
    {
        for(int i=0;i<ALP;i++)next[p][i]=0;
        cnt[p]=num[p]=0,len[p]=l;
        return p++;
    }
    void init()
    {
        p=last=n=0,s[n]=-1,fail[0]=1;
        newnode(0),newnode(-1);
    }
    int get_fail(int x)
    {
        while(s[n-len[x]-1]!=s[n])x=fail[x];
        return x;
    }
    void add(int c)
    {
        s[++n]=c;
        int cur=get_fail(last);
        if(!next[cur][c])
        {
            int now=newnode(len[cur]+2);
            fail[now]=next[get_fail(fail[cur])][c];
            next[cur][c]=now,num[now]=num[fail[now]] + 1;
        }
        last=next[cur][c];
        cnt[last]++;
    }
    void count()
    {
        for(int i=p-1;i>=0;i--)cnt[fail[i]]+=cnt[i];
    }
    void dfs(int now)
    {
        for(int i=0;i<ALP;i++)
        {
            if(next[now][i])
            {
                t+='a'+i;
                cout<<t<<endl;
                dfs(next[now][i]);
                t=t.substr(0,t.size()-1);
            }
        }
    }
}pam;
char s[300005];
int main()
{
    while(scanf("%s",s)!=EOF)
    {
        pam.init();
        int len=strlen(s);
        for(int i=0;i<len;i++)pam.add(s[i]-'a');
        pam.dfs(0);
    }
    return 0;
}
```

2.8 AC 自动机

```
#include<bits/stdc++.h>
```

```
using namespace std;
const int MAXNODE=500005,MAXCHAR=26,MAXN=1000005,MAXM=10005;
int ans,T,n,success[MAXM];
char s[MAXN],t[MAXN];
struct AC_automaton
{
    int ch[MAXNODE][MAXCHAR];
    int sz;
    int val[MAXNODE];
    int fail[MAXNODE];
    int last[MAXNODE];
    int q[MAXNODE];
    int head,tail;
    void clear()
    {
        for(int i=0;i<MAXCHAR;++i)
        {
            ch[0][i]=0;
        }
        sz=head=tail=0;
    }
    int new_node()
    {
        sz++;
        for(int i=0;i<MAXCHAR;++i)
        {
            ch[sz][i]=0;
        }
        fail[sz]=last[sz]=val[sz]=0;
        return sz;
    }
    void insert(char t[])
    {
        int root=0;
        for(int i=0; t[i]!='\0'; ++i)
        {
            if(!ch[root][t[i]-'a'])
            {
                ch[root][t[i]-'a']=new_node();
            }
            root=ch[root][t[i]-'a'];
        }
        ++val[root];
    }
    void getfail()
    {
        for(int i=0; i<MAXCHAR; i++)
        {
            if(ch[0][i])
            {
                q[++tail]=ch[0][i];
            }
        }
        while(head!=tail)
        {
            int root=q[++head];
            for(int i=0; i<MAXCHAR; i++)
            {
                if(!ch[root][i])
                {
                    ch[root][i]=ch[fail[root]][i];
                    continue;
                }
                int child=ch[root][i];
                fail[child]=ch[fail[root]][i];
                last[child]=val[fail[child]]?fail[child]:last[fail[child]];
                q[++tail]=child;
            }
        }
    }
    void addnum(int root)
    {
        if(root)
        {
            ans+=val[root];
            addnum(last[root]);
        }
    }
    void find(char s[])
    {
        int root=0;
        for(int i=0; s[i]!='\0'; i++)
        {
            root=ch[root][s[i]-'a'];
            if(val[root]) addnum(root);
            else if(val[last[root]]) addnum(last[root]);
        }
    }
}AC;

int main()
{
    scanf("%d",&T);
    while(T--)
    {
        AC.clear();
        ans=0;
        scanf("%d",&n);
        for(int i=1;i<=n;++i)
        {
            scanf("%s",&t);
            AC.insert(t);
            success[i]=0;
        }
        AC.getfail();
        scanf("%s",&s);
        AC.find(s);
        printf("%d\n",ans);
    }
    return 0;
}
```

2.9 序列自动机

```
int nex[MAX][26];
void work(char *s,int len)
{
    mem(nex[len],0);
    for(int i=len;i;i--)
    {
        for(int j=0;j<26;j++)
        {
            nex[i-1][j]=nex[i][j];
        }
        nex[i-1][s[i]-'a']=i;
    }
}
```

2.10 最小表示法

```
int getmin(char *s){
    int n=strlen(s);
    int i=0,j=1,k=0,t;
    while(i<n && j<n && k<n){
        t=s[(i+k)%n]-s[(j+k)%n];
        if (!t) k++;
        else{
            if (t>0) i+=k+1;
            else j+=k+1;
            if (i==j) j++;
            k=0;
        }
    }
    return i<j?i:j;
}
```

2.11 Lyndon 分解

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

#define Abigail inline void
typedef long long LL;

const int N=1<<20;

```

```
char s[N+9];
int n,x[N+9],ts;

void Lyndon_word(){
  int j,k;
  for (int i=1;i<=n;){
    for (j=i,k=i+1;k<=n&&s[k]>=s[j];++k) //一直循环直到串结束或情况3出现
      s[k]>s[j]?j=i:++j; //情况1与情况2
    for (;i<=j;i+=k-j)
      x[++ts]=i+k-j-1; //记录右端点
  }
}

Abigail into(){
  scanf("%s",s+1);
  n=strlen(s+1);
}

Abigail work(){
  Lyndon_word();
}

Abigail outo(){
  for (int i=1;i<=ts;++i)
    printf("%d ",x[i]);
}

int main(){
  into();
  work();
  outo();
  return 0;
}
```

3 图论

3.1 dijkstra

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=200005;
int i,i0,n,m,r;
long long d[maxn];
bool vis[maxn];
struct edge
{
    int to;
    long long v;
    bool operator<(edge a)const{return a.v<v;}
};
vector<edge>mp[maxn];
priority_queue<edge>q;
```

```
int main()
{
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        int a,b,c,d;
        scanf("%d %d %d %d",&a,&b,&c,&d);
        if(i==n)r=a;
        mp[i].push_back({d,a});
        for(int i0=1;i0*i0<=c;i0++)
        {
            if(c%i0==0)
            {
                if(i+i0<=n)mp[i].push_back({i+i0,a+b});
                if(i+c/i0<=n)mp[i].push_back({i+c/i0,a+b});
            }
        }
    }
    int s=1,t=n;
    for(i=1;i<=n;i++)d[i]=(long long)INT_MAX*(n+5);
    d[s]=0;
    q.push({s,d[s]});
    while(!q.empty())
    {
        int to=q.top().to;
        long long v=q.top().v;
        q.pop();
        if(!vis[to])
        {
            vis[to]=1;
            for(auto i:mp[to])
            {
                if(d[i.to]>v+i.v)
                {
                    d[i.to]=v+i.v;
                    q.push({i.to,d[i.to]});
                }
            }
        }
    }
    d[t]+=r;
    if(d[t]>=(long long)INT_MAX*(n+5))d[t]=-1;
    printf("%lld\n",d[t]);
    return 0;
}
```

3.2 dinic

```
#include<cstdio>
#include<cstring>
#include<queue>

```

```cpp
using namespace std;

const int VM=2520;
const int EM=500010;
const int INF=0x3f3f3f3f;

struct Edge{
    int u,v,nxt;
    int flow;
}edge[EM<<1];

int n,m,cnt,head[VM];
int src,des,dep[VM];

void addedge(int cu,int cv,int cf){
    edge[cnt].u=cu; edge[cnt].v=cv; edge[cnt].flow=cf;
    edge[cnt].nxt=head[cu]; head[cu]=cnt++;

    edge[cnt].u=cv; edge[cnt].v=cu; edge[cnt].flow=0;
    edge[cnt].nxt=head[cv]; head[cv]=cnt++;
}

int dir[4][2]={{-1,0},{1,0},{0,-1},{0,1}};

int legal(int i,int j,int k){
    int x=i+dir[k][0];
    int y=j+dir[k][1];
    return x>=1 && x<=n && y>=1 && y<=m;
}

int BFS(){
    queue<int> q;
    while(!q.empty())
        q.pop();
    memset(dep,-1,sizeof(dep));
    dep[src]=0;
    q.push(src);
    while(!q.empty()){
        int u=q.front();
        q.pop();
        for(int i=head[u];i!=-1;i=edge[i].nxt){
            int v=edge[i].v;
            if(edge[i].flow>0 && dep[v]==-1){
                dep[v]=dep[u]+1;
                q.push(v);
            }
        }
    }
    return dep[des]!=-1;
}
int DFS(int u,int minx){
    int ans=0;
    if(u==des)
```

```
        return minx;
    for(int i=head[u];i!=-1 && ans<minx;i=edge[i].nxt){
        int v=edge[i].v;
        if(edge[i].flow>0 && dep[v]==dep[u]+1){
            int tmp=min(edge[i].flow,minx-ans);
            tmp=DFS(v,tmp);
            ans+=tmp;
            edge[i].flow-=tmp;
            edge[i^1].flow+=tmp;
        }
    }
    if(!ans)
        dep[u]=-2;
    return ans;
}

int Dinic(){
    int ans=0,tmp;
    while(BFS()){
        while(1){
            tmp=DFS(src,INF);
            if(tmp==0)
                break;
            ans+=tmp;
        }
    }
    return ans;
}

int main()
{
    while(~scanf("%d%d",&n,&m))
    {
        cnt=0;
        memset(head,-1,sizeof(head));
        int x,sum=0;
        src=0; des=n*m+1;
        for(int i=1;i<=n;i++)
            for(int j=1;j<=m;j++){
                scanf("%d",&x);
                sum+=x;
                if((i+j)%2==0){
                    addedge(src,(i-1)*m+j,x);
                    for(int k=0;k<4;k++){
                        if(legal(i,j,k))
                            addedge((i-1)*m+j,(i+dir[k][0]-1)*m+(j+dir[k][1]),INF);
                    }
                }else{
                    addedge((i-1)*m+j,des,x);
                    for(int k=0;k<4;k++){
                        if(legal(i,j,k))
                            addedge((i+dir[k][0]-1)*m+(j+dir[k][1]),(i-1)*m+j,INF);
                    }
            }
        }
        int maxflow=Dinic();
        printf("%d\n",sum-maxflow);
    }
    return 0;
}
```

3.3 dij 费用流

```
#include<bits/stdc++.h>
using namespace std;
typedef pair<int,int>P;
const int MAX_V=5005;
const int INF=0x3f3f3f3f;
struct edge {
    int to,cap,cost,rev;
};
int V;
vector<edge>G[MAX_V];
int h[MAX_V];
int dist[MAX_V];
int prevv[MAX_V],preve[MAX_V];
void add_edge(int from,int to,int cap,int cost) {
    G[from].push_back((edge) {
        to,cap,cost,(int)G[to].size()
    });
    G[to].push_back((edge) {
        from,0,-cost,(int)G[from].size()-1
    });
}
int min_cost_flow(int s,int t,int f) {
    int res=0;
    fill(h,h+V+1,0);
    while(f>0) {
        priority_queue<P,vector<P>,greater<P>>que;
        fill(dist,dist+V+1,INF);
        dist[s]=0;
        que.push(P(0,s));
        while(!que.empty()) {
            P p=que.top();
            que.pop();
            int v=p.second;
            if(dist[v]<p.first)continue;
            for(int i=0; i < G[v].size(); i++) {
                edge &e=G[v][i];
                if(e.cap>0&&dist[e.to]>dist[v]+e.cost+h[v]-h[e.to]) {
                    dist[e.to]=dist[v]+e.cost+h[v]-h[e.to];
                    prevv[e.to]=v;
                    preve[e.to]=i;
                    que.push(P(dist[e.to],e.to));
                }
            }
        }
        if(dist[t]==INF) {
            return -1;
        }
        for(int v=1; v<=V; v++)h[v]+=dist[v];
        int d=f;
        for(int v=t; v!=s; v=prevv[v]) {
            d=min(d,G[prevv[v]][preve[v]].cap);
        }
        f-=d;
        res+=d*h[t];
        for(int v=t; v!=s; v=prevv[v]) {
            edge &e=G[prevv[v]][preve[v]];
            e.cap-=d;
            G[v][e.rev].cap+=d;
        }
    }
    return res;
}
int a[2005];
int main() {
    int T;
    scanf("%d",&T);
    while(T--)
    {
        int n,k;
        scanf("%d %d",&n,&k);
        V=n*2+3;
        for(int i=1;i<=V;i++)G[i].clear();
        int s=2*n+1,s0=s+1,t=s0+1;
        int x,y,z,zz;
        x=s,y=s0,z=k,zz=0;
        add_edge(x,y,z,zz);
        for(int i=1;i<=n;i++)
        {
            scanf("%d",&a[i]);
            x=s0,y=i,z=1,zz=-a[i];
            add_edge(x,y,z,zz);
            x=i,y=i+n,z=1,zz=0;
            add_edge(x,y,z,zz);
            x=n+i,y=t,z=1,zz=0;
            add_edge(x,y,z,zz);
            for(int i0=1;i0<i;i0++)
            {
                if(a[i0]>a[i])continue;
                x=i0+n,y=i,z=1,zz=-a[i];
                add_edge(x,y,z,zz);
            }
        }
        printf("%d\n",-min_cost_flow(s,t,k));
    }
    return 0;
```

```
}
```

3.4 zkw 费用流

```
#include<bits/stdc++.h>
using namespace std;
bool vis[200001];
int n,m,s,t,ans,ans0;
int nedge=-1,p[200001],c[200001],cc[200001],nex[200001],head[200001],dist[200001];
void addedge(int x,int y,int z,int zz){p[++nedge]=y;c[nedge]=z;cc[nedge]=zz;nex[nedge]=head[x];head[x]=nedge;}
bool spfa(int s,int t)
{
    memset(vis,0,sizeof vis);
    for(int i=0;i<=n;i++)dist[i]=1e9;
    dist[t]=0,vis[t]=1;
    deque<int>q;
    q.push_back(t);
    while(!q.empty())
    {
        int now=q.front();
        q.pop_front();
        for(int k=head[now];k>-1;k=nex[k])if(c[k^1]&&dist[p[k]]>dist[now]-cc[k])
        {
            dist[p[k]]=dist[now]-cc[k];
            if(!vis[p[k]])
            {
                vis[p[k]]=1;
                if(!q.empty()&&dist[p[k]]<dist[q.front()])q.push_front(p[k]);
                else q.push_back(p[k]);
            }
        }
        vis[now]=0;
    }
    return dist[s]<1e9;//不要求流量最大则设置为<0
}
int dfs(int x,int low)
{
    if(x==t)
    {
        vis[t]=1;
        return low;
    }
    int used=0,a;
    vis[x]=1;
    for(int k=head[x];k>-1;k=nex[k])
    {
        if(!vis[p[k]]&&c[k]&&dist[x]-cc[k]==dist[p[k]])
        {
            a=dfs(p[k],min(c[k],low-used));
            if(a)ans+=a*cc[k],c[k]-=a,c[k^1]+=a,used+=a;
            if(used==low)break;
        }
    }
    return used;
}
int costflow()
{
    int flow=0;
    while(spfa(s,t))
    {
        vis[t]=1;
        while(vis[t])
        {
            memset(vis,0,sizeof vis);
            flow+=dfs(s,1e9);
        }
    }
    return flow;
}
int main()
{
    memset(nex,-1,sizeof nex),memset(head,-1,sizeof head);
    scanf("%d%d%d%d",&n,&m,&s,&t);
    for(int i=1;i<=m;i++)
    {
        int x,y,z,zz;
        scanf("%d%d%d%d",&x,&y,&z,&zz);
        addedge(x,y,z,zz),addedge(y,x,0,-zz);
    }
    ans=0;
    ans0=costflow();
    printf("%d %d\n",ans0,ans);
    return 0;
}
```

3.5 KM 算法

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

const int maxn=305;
const int inf=1e9;

int n,minz;
int wx[maxn],wy[maxn],match[maxn];
int mp[maxn][maxn],slack[maxn],pre[maxn];
bool viy[maxn];

void Bfs(int k)
{
    int py=0,px,yy=0,delta;
    match[py]=k;
    for(int i=0;i<=n;i++)pre[i]=0,slack[i]=inf;
```

```cpp
    do
    {
        px=match[py],delta=inf,viy[py]=1;
        for(int i=1; i<=n; ++i)
        {
            if(!viy[i])
            {
                if(wx[px]+wy[i]-mp[px][i]<slack[i])slack[i]=wx[px]+wy[i]-mp[px][i],pre[
                    i]=py;
                if(slack[i]<delta)delta=slack[i],yy=i;
            }
        }
        for(int i=0; i<=n; ++i)
        {
            if(viy[i])wx[match[i]]-=delta,wy[i]+=delta;
            else slack[i]-=delta;
        }
        py=yy;
    }while(match[py]!=0);
    while(py)match[py]=match[pre[py]],py=pre[py];
}

int Km()
{
    for(int i=1; i<=n; ++i)
    {
        wx[i]=0,wy[i]=0,match[i]=0;
        for(int j=1; j<=n; ++j)wx[i]=max(wx[i],mp[i][j]);
    }
    for(int i=1;i<=n;++i)memset(viy,0,sizeof(viy)),Bfs(i);
    int Ans=0;
    for(int i=1;i<=n;++i)Ans+=wx[match[i]]+wy[i];
    return Ans;
}

int main()
{
    while(~scanf("%d",&n))
    {
        for(int i=1;i<=n;++i)for(int j=1;j<=n;++j)scanf("%d",&mp[i][j]);
        printf("%d\n",Km());
    }
    return 0;
}
```

3.6 带花树

```cpp
#include<bits/stdc++.h>
#define T 1205
using namespace std;
int head[T],lst[T*T*2],nxt[T*T];
int tot,n,m;
```

```
void ins(int x,int y)
{
    lst[++tot]=y;
    nxt[tot]=head[x];
    head[x]=tot;
}
int ma[T],st[T],pr[T],fa[T],q[T],v[T];
int ans,TI,u,t;
int lca(int x,int y)
{
    for(TI++;;swap(x,y))if(x)
    {
        if(v[x]==TI)return x;
        v[x]=TI;
        x=fa[pr[ma[x]]];
    }
}
void up(int x,int y,int f)
{
    while(fa[x]!=f)
    {
        pr[x]=y;
        if(st[ma[x]]>0)st[q[++t]=ma[x]]=0;
        if(fa[x]==x)fa[x]=f;
        if(fa[ma[x]]==ma[x])fa[ma[x]]=f;
        x=pr[y=ma[x]];
    }
}
int match(int x)
{
    for(int i=1;i<=n;i++)fa[i]=i,st[i]=-1;
    st[q[t=1]=x]=0;
    for(int l=1;l<=t;l++)for(int i=head[q[l]];i;i=nxt[i])if(st[lst[i]]<0)
    {
        st[lst[i]]=1;
        pr[lst[i]]=q[l];
        if(!ma[lst[i]])
        {
            for(int j=q[l],k=lst[i];j;j=pr[k=u])
            {
                u=ma[j];
                ma[j]=k;
                ma[k]=j;
            }
            return 1;
        }
        st[q[++t]=ma[lst[i]]]=0;
    }
    else if(fa[lst[i]]!=fa[q[l]]&&!st[lst[i]])
    {
        int f=lca(lst[i],q[l]);
        up(q[l],lst[i],f);
        up(lst[i],q[l],f);
        for(int j=1;j<=n;j++)fa[j]=fa[fa[j]];
    }
    return 0;
}
int main()
{
    int cas;
    scanf("%d",&cas);
    while(cas--)
    {
        TI=tot=0;
        memset(head,0,sizeof(head));
        memset(ma,0,sizeof(ma));
        memset(v,0,sizeof(v));
        scanf("%d %d",&n,&m);
        ans=-n;
        for(int i=1;i<=n;i++)
        {
            ins(i,i+n),ins(i+n,i);
            int num,tmp;
            scanf("%d",&num);
            while(num--)
            {
                scanf("%d",&tmp);
                tmp+=2*n;
                ins(tmp,i),ins(i,tmp);
                ins(tmp,i+n),ins(i+n,tmp);
            }
        }
        n=2*n+m;
        for(int i=1;i<=n;i++)ans+=!ma[i]&&match(i);
        printf("%d\n",ans);
    }
    return 0;
}
```

3.7　并查集

```
int fin(int x)
{
    int tmp=x,tmp2;
    while(pre[tmp]!=tmp)tmp=pre[tmp];
    while(x!=tmp)tmp2=pre[x],pre[x]=tmp,x=tmp2;
    return tmp;
}

int fin(int x){return (pre[x]==x)?x:pre[x]=fin(pre[x]);}
void uni(int x,int y){if(fin(x)!=fin(y))pre[fin(y)]=fin(x);}

struct DisjointSetUnion
{
    int fa[maxn*2],rank[maxn*2];
    stack<pair<int*,int>>stk;
    void init(){for(int i=1;i<=maxn;i++)fa[i]=i,rank[i]=0;}
    int find(int x){return (x^fa[x])?find(fa[x]):x;}
    void join(int x,int y)
    {
        x=find(x),y=find(y);
        if(rank[x]<=rank[y])
        {
            stk.push({fa+x,fa[x]}),fa[x]=y;
            if(rank[x]==rank[y])stk.push({rank+y,rank[y]}),rank[y]++;
        }
        else stk.push({fa+y,fa[y]}),fa[y]=x;
    }
    void undo(){*stk.top().first=stk.top().second,stk.pop();}
}DSU;
```

3.8 割点

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=100005;
using namespace std;
int n,m,len=0,id=0,ans=0;
int last[maxn],low[maxn],dfn[maxn];
bool bz[maxn];
vector<int>mp[maxn];
void dfs(int x,int root)
{
    int tot=0;
    low[x]=dfn[x]=++id;
    for(int y:mp[x])
    {
        if(!dfn[y])
        {
            dfs(y,root);
            low[x]=min(low[x],low[y]);
            if(low[y]>=dfn[x]&&x!=root) bz[x]=true;
            if(x==root) tot++;
        }
        low[x]=min(low[x],dfn[y]);
    }
    if(x==root&&tot>=2) bz[root]=true;
}
int main()
{
    int x,y,t;
    scanf("%d %d",&n,&m);
    for(int i=1;i<=m;i++)
    {
        scanf("%d %d",&x,&y);
        mp[x].push_back(y);
        mp[y].push_back(x);
    }
    for(int i=1;i<=n;i++)if(!dfn[i])dfs(i,i);
    for(int i=1;i<=n;i++)if(bz[i])printf("%d\n",i);
}
```

3.9 桥

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=5005;
int i,i0,n,m,T,dfn[maxn],low[maxn],deep,cnt[maxn],sum,pre[maxn];
vector<int>mp[maxn];
int fin(int x){return (pre[x]==x)?x:pre[x]=fin(pre[x]);}
void uni(int x,int y){if(fin(x)!=fin(y))pre[fin(y)]=fin(x);}
void dfs(int now,int root)
{
    dfn[now]=low[now]=++deep;
    bool f=0;
    for(int i:mp[now])
    {
        if(i==root&&!f)f=1;
        else
        {
            if(!dfn[i])dfs(i,now);
            low[now]=min(low[now],low[i]);
            if(low[now]==low[i])uni(now,i);
        }
    }
}
int main()
{
    while(scanf("%d %d",&n,&m)!=EOF)
    {
        sum=deep=0;
        for(i=1;i<=n;i++)mp[i].clear(),dfn[i]=0,cnt[i]=0,pre[i]=i;
        while(m--)
        {
            int a,b;
            scanf("%d %d",&a,&b);
            mp[a].push_back(b),mp[b].push_back(a);
        }
        for(i=1;i<=n;i++)if(!dfn[i])dfs(i,-1);
        for(i=1;i<=n;i++)
        {
            for(int i0:mp[i])
            {
                if(fin(i)!=fin(i0))printf("%d %d\n",fin(i),fin(i0));
            }
        }
    }
    return 0;
}
```

3.10 强连通分量

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn=1005;
int i,i0,n,m,T,dfn[maxn],low[maxn],deep,col[maxn],sum,cnt[maxn],out[maxn];
bool vis[maxn];
vector<int>mp[maxn];
stack<int>stk;
void dfs(int now)
{
    dfn[now]=low[now]=++deep;
    stk.push(now);
    vis[now]=1;
    for(auto i:mp[now])
    {
        if(!dfn[i])
        {
            dfs(i);
            low[now]=min(low[now],low[i]);
        }
        else if(vis[i])low[now]=min(low[now],low[i]);
    }
    if(dfn[now]==low[now])
    {
        sum++;
        while(vis[now])
        {
            col[stk.top()]=sum;
            vis[stk.top()]=0;
            cnt[sum]++;
            stk.pop();
        }
    }
}
int main()
{
    while(scanf("%d %d",&n,&m)!=EOF)
    {
        sum=deep=0;
        for(i=1;i<=n;i++)mp[i].clear(),dfn[i]=0,cnt[i]=0,out[i]=0;
        while(m--)
        {
            int a,b;
            scanf("%d %d",&a,&b);
            mp[a].push_back(b);
        }
        for(i=1;i<=n;i++)if(!dfn[i])dfs(i);
        int ans=0;
        for(i=1;i<=n;i++)for(auto i0:mp[i])out[col[i]]+=(col[i]!=col[i0]);
        for(i=1;i<=sum;i++)if(!out[i]&&cnt[i]>1)ans++;
        printf("%d\n",ans);
    }
    return 0;
}
```

3.11 树链剖分

```
#include<bits/stdc++.h>
using namespace std;
#define mid (l+r)/2
const int maxn=100005;
int i,i0,n,m,dep[maxn],siz[maxn],fa[maxn],son[maxn],idx[maxn],cnt,top[maxn];
vector<int>mp[maxn];
void dfs(int now,int root,int deep)
{
    dep[now]=deep,fa[now]=root,siz[now]=1,son[now]=-1;
    for(auto i:mp[now])
    {
        if(dep[i])continue;
        dfs(i,now,deep+1);
        siz[now]+=siz[i];
        if(son[now]==-1||siz[son[now]]<siz[i])son[now]=i;
    }
}
void dfs0(int now,int topf)
{
    idx[now]=++cnt,top[now]=topf;
    if(son[now]!=-1)dfs0(son[now],topf);
    for(auto i:mp[now])if(!idx[i])dfs0(i,i);
}
int tree[maxn*4],lazy[maxn*4];
void p_tree(int l,int r,int p)
{
    if(lazy[p]==-1)return;
    lazy[p*2]=lazy[p],lazy[p*2+1]=lazy[p];
    tree[p*2]=(mid-l+1)*lazy[p],tree[p*2+1]=(r-mid)*lazy[p];
    lazy[p]=-1;
}
void c_tree(int l,int r,int p,int a,int b,int v)
{
    if(l==a&&r==b)lazy[p]=v,tree[p]=(r-l+1)*v;
    else
    {
        p_tree(l,r,p);
        if(b<=mid) c_tree(l,mid,p*2,a,b,v);
        else if(a>=mid+1)c_tree(mid+1,r,p*2+1,a,b,v);
        else c_tree(l,mid,p*2,a,mid,v),c_tree(mid+1,r,p*2+1,mid+1,b,v);
        tree[p]=tree[p*2]+tree[p*2+1];
    }
}
int q_tree(int l,int r,int p,int a,int b)
{
    if(l==a&&r==b)return tree[p];
    else
    {
        p_tree(l,r,p);
        if(b<=mid)return q_tree(l,mid,p*2,a,b);
        else if(a>=mid+1)return q_tree(mid+1,r,p*2+1,a,b);
        else return q_tree(l,mid,p*2,a,mid)+q_tree(mid+1,r,p*2+1,mid+1,b);
    }
}
int q_range(int x,int y)
{
    int ans=0;
    while(top[x]!=top[y])
    {
        if(dep[top[x]]<dep[top[y]])swap(x,y);
        ans+=q_tree(1,n,1,idx[top[x]],idx[x]);
        x=fa[top[x]];
    }
    if(dep[x]>dep[y])swap(x,y);
    ans+=q_tree(1,n,1,idx[x],idx[y]);
    return ans;
}
void c_range(int x,int y,int k)
{
    while(top[x]!=top[y])
    {
        if(dep[top[x]]<dep[top[y]])swap(x,y);
        c_tree(1,n,1,idx[top[x]],idx[x],k);
        x=fa[top[x]];
    }
    if(dep[x]>dep[y])swap(x,y);
    c_tree(1,n,1,idx[x],idx[y],k);
}
int q_son(int x){return q_tree(1,n,1,idx[x],idx[x]+siz[x]-1);}
void c_son(int x,int k){c_tree(1,n,1,idx[x],idx[x]+siz[x]-1,k);}
void init()
{
    memset(dep,0,sizeof(dep));
    memset(idx,0,sizeof(idx));
    memset(lazy,-1,sizeof(lazy));
    cnt=0;
}
int main()
{
    int r=1;
    scanf("%d",&n);
    for(i=2;i<=n;i++)
    {
        int x;
        scanf("%d",&x),x++;
        mp[i].push_back(x);
        mp[x].push_back(i);
    }
    init();
    dfs(r,-1,1);
    dfs0(r,r);
    scanf("%d",&m);
    int now=0;
    while(m--)
    {
        char op[10];
        int x;
        scanf("%s %d",op,&x),x++;
        if(op[0]=='i')c_range(x,r,1);
        if(op[0]=='u')c_son(x,0);
        int tmp=q_son(r);
        printf("%d\n",abs(now-tmp));
        now=tmp;
    }
    return 0;
}
```

3.12 LinkCutTree

```
#include<cstdio>
#include<iostream>
#include<algorithm>
#define inf 1000000000
#define ll long long
using namespace std;
int read()
{
    int x=0,f=1;char ch=getchar();
    while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}
    while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}
    return x*f;
}
int top,q[200005];
int n,m,ans=0;
int mx[200005],val[200005];
int p[200005],fa[200005],c[200005][2];
bool rev[200005];
struct Data{
    int u,v,a,b;
}e[100005];
int find(int x)
{
    return x==p[x]?x:p[x]=find(p[x]);
}
bool operator<(Data a,Data b)
{
    return a.a<b.a;
}
bool isroot(int x)
{
    return c[fa[x]][0]!=x&&c[fa[x]][1]!=x;
}
```

```
void update(int x)
{
    int l=c[x][0],r=c[x][1];
    mx[x]=x;
    if(val[mx[l]]<val[mx[x]])mx[x]=mx[l];
    if(val[mx[r]]<val[mx[x]])mx[x]=mx[r];
}
void pushdown(int x)
{
    int l=c[x][0],r=c[x][1];
    if(rev[x])
    {
        rev[x]^=1;rev[l]^=1;rev[r]^=1;
        swap(c[x][0],c[x][1]);
    }
}
void rotate(int &x)
{
    int y=fa[x],z=fa[y],l,r;
    if(c[y][0]==x)l=0;else l=1;r=l^1;
    if(!isroot(y))
    {
        if(c[z][0]==y)c[z][0]=x;
        else c[z][1]=x;
    }
    fa[x]=z;fa[y]=x;fa[c[x][r]]=y;
    c[y][l]=c[x][r];c[x][r]=y;
    update(y);update(x);
}
void splay(int &x)
{
    top=0;q[++top]=x;
    for(int i=x;!isroot(i);i=fa[i])q[++top]=fa[i];
    while(top)pushdown(q[top--]);
    while(!isroot(x))
    {
        int y=fa[x],z=fa[y];
        if(!isroot(y))
        {
            if(c[y][0]==x^c[z][0]==y)rotate(x);
            else rotate(y);
        }
        rotate(x);
    }
}
void access(int x)
{
    for(int t=0;x;t=x,x=fa[x])
        splay(x),c[x][1]=t,update(x);
}
void makeroot(int x)
{
    access(x);splay(x);rev[x]^=1;
```

```
}
void link(int x,int y)
{
   makeroot(x);fa[x]=y;
}
void cut(int x,int y)
{
   makeroot(x);access(y);splay(y);
   c[y][0]=fa[x]=0;update(y);
}
int query(int x,int y)
{
   makeroot(x);access(y);splay(y);
   return mx[y];
}
int main()
{
   n=read();m=read();
   for(int i=1;i<=n;i++)p[i]=i,val[i]=inf;
   for(int i=1;i<=m;i++)
   {
      e[i].u=read();e[i].v=read();e[i].a=read();e[ i].b=read();
   }
   sort(e+1,e+m+1);
   int tot=0;
   val[0]=inf;
   for(int i=1;i<=m;i++)
   {
      int u=e[i].u,v=e[i].v,a=e[i].a,b=e[i].b;
      if(find(u)==find(v))
      {
         int t=query(u,v);
         if(val[t]<e[i].b)
         {
            cut(t,e[t-n].u);
            cut(t,e[t-n].v);
         }
         else
         {
            if(find(1)==find(n))
            {
               //printf("%d %d\n",i,e[i].a,val[query(1,n)]);
               if(i==m)ans+=max(val[query(1,n)]-e[i].a+1,0);
               else ans+=max(min(val[query(1,n)],e[i+1].a-1)-e[i].a+1,0);
            }
            continue;
         }
      }
      else p[find(u)]=find(v);
      val[n+i]=e[i].b;mx[n+i]=n+i;
      link(u,n+i);link(v,n+i);
      if(find(1)==find(n))
      {
            int t=query(1,n);
            //printf("%d %d %d\n",i,e[i].a,val[t]);
            if(i==m)ans+=max(val[query(1,n)]-e[i].a+1,0);
            else ans+=max(min(val[query(1,n)],e[i+1].a-1)-e[i].a+1,0);
        }
    }
    printf("%d\n",ans);
    return 0;
}
```

3.13 任意根 lca

```
int r,x,y;
int rx=lca(r,x),ry=lca(r,y);
if(rx==ry)printf("%d\n",lca(x,y));
else printf("%d\n",d(rx,r)<d(ry,r)?rx:ry);
```

3.14 最大独立集

```
#include<cstdio>
#include<cstring>
#define N 50
/*
最大团 = 补图G的最大独立集数
———>最大独立集数 = 补图G'最大团
*/
//最大团模板
bool a[N][N];//a为图的邻接表(从1开始)
int ans, cnt[N], group[N], n, m, vis[N];//ans表示最大团，cnt[N]表示当前最大团的节点数，
    group[N]用以寻找一个最大团集合
bool dfs( int u, int pos )//u为当从前顶点开始深搜，pos为深搜深度（即当前深搜树所在第几层
    的位置）
{
    int i, j;
    for( i = u+1; i <= n; i++)//按递增顺序枚举顶点
    {
        if( cnt[i]+pos <= ans ) return 0;//剪枝
        if( a[u][i] )
        {
             // 与目前团中元素比较，取 Non-N(i)
            for( j = 0; j < pos; j++ ) if( !a[i][ vis[j] ] ) break;
            if( j == pos )
            { // 若为空，则皆与 i 相邻，则此时将i加入到 最大团中
                vis[pos] = i;//深搜层次也就是最大团的顶点数目，vis[pos] = i表示当前第pos小
                    的最大团元素为i（因为是按增顺序枚举顶点 ）
                if( dfs( i, pos+1 ) ) return 1;
            }
        }
    }
    if( pos > ans )
    {
            for( i = 0; i < pos; i++ )
                group[i] = vis[i]; // 更新最大团元素
            ans = pos;
            return 1;
    }
    return 0;
}
void maxclique()//求最大团
{
    ans=-1;
    for(int i=n;i>0;i--)
    {
        vis[0]=i;
        dfs(i,1);
        cnt[i]=ans;
    }
}
int main()
{
    int T;
    //scanf("%d",&T);
    while(~scanf("%d",&n))
    {
        if(n==0) break;
        //scanf("%d%d",&n,&m );
        int x, y;
        memset( a, 0, sizeof(a));
        /*for(int i = 0; i < m; i++)
        {
            scanf("%d%d",&x,&y);
            a[x][y] = a[y][x] = 1;
        }*/
        //相邻顶点间有边相连，模型转换成求 无向图 最大独立集。
        //要求原图的最大独立集，转化为求原图的补图的最大团(最大团顶点数量 = 补图的最大独立集
            )
        for(int i = 1; i <= n; i++)//求原图的补图
            for(int j = 1; j <= n; j++)
                scanf("%d",&a[i][j]);
        maxclique();//求最大团
        if( ans < 0 ) ans = 0;//ans表示最大团
        printf("%d\n", ans );
        /*for(int i = 0; i < ans; i++)
            printf( i == 0 ? "%d" : " %d", group[i]);//group[N]用以寻找一个最大团集合
        if( ans > 0 ) puts("");*/
    }
}
```

3.15 拓扑排序

```
#include<bits/stdc++.h>
using namespace std;
int i,i0,n,m,cnt[505];
```

```
vector<int>mp[505],ans;
priority_queue<int,vector<int>,greater<int>>q;
int main()
{
    while(scanf("%d %d",&n,&m)!=EOF)
    {
        ans.clear();
        for(i=1;i<=n;i++)mp[i].clear();
        while(m--)
        {
            int a,b;
            scanf("%d %d",&a,&b);
            mp[a].push_back(b);
            cnt[b]++;
        }
        for(i=1;i<=n;i++)if(!cnt[i])q.push(i);
        while(!q.empty())
        {
            m=q.top(),q.pop();
            ans.push_back(m);
            for(auto i:mp[m])if(!--cnt[i])q.push(i);
        }
        for(i=0;i<ans.size();i++)printf("%d%c",ans[i],i==ans.size()-1?'\n':' ');
    }
    return 0;
}
```

3.16 树分治

```
#include<bits/stdc++.h>
using namespace std;
int i,i0,n,k,msiz[100005],siz[100005],sum[100005],ans[100005];
bool vis[100005];
vector<int>mp[100005];
int getRoot(int now,int root,int Siz)
{
    int res=-1;
    msiz[now]=0,siz[now]=1;
    for(int i:mp[now])
    {
        if(vis[i]||i==root)continue;
        int x=getRoot(i,now,Siz);
        msiz[now]=max(msiz[now],siz[i]),siz[now]+=siz[i];
        if(res==-1||msiz[x]<msiz[res])res=x;
    }
    msiz[now]=max(msiz[now],Siz-siz[now]);
    if(res==-1||msiz[now]<msiz[res])res=now;
    return res;
}
void add(int p, int x){while(p<=n)sum[p]+=x,p+=p&-p;}
int ask(int p){int res=0;while(p)res+=sum[p],p-=p&-p;return res;}
int range_ask(int l, int r){return ask(r)-ask(l-1);}
```

```
void cal(int now,int root,int dep,int sign)
{
   add(dep+1,sign);
   if(k-dep>0)for(int i:mp[now])if(!vis[i]&&root!=i)cal(i,now,dep+1,sign);
}
void dfs(int now,int root,int dep)
{
   if(k-dep>=0)ans[now]+=range_ask(1,k-dep+1);
   if(k-dep>0)for(int i:mp[now])if(!vis[i]&&root!=i)dfs(i,now,dep+1);
}
int calSiz(int now,int root)
{
   int Siz=1;
   for(int i:mp[now])if(!vis[i]&&root!=i)Siz+=calSiz(i,now);
   return Siz;
}
void div(int now)
{
   now=getRoot(now,-1,calSiz(now,-1)),vis[now]=1;
   add(1,1);
   for(int i:mp[now])if(!vis[i])cal(i,now,1,1);
   ans[now]+=range_ask(1,k+1);
   for(int i:mp[now])
   {
      if(!vis[i])
      {
         cal(i,now,1,-1);
         dfs(i,now,1);
         cal(i,now,1,1);
      }
   }
   for(int i:mp[now])if(!vis[i])cal(i,now,1,-1);
   add(1,-1);
   for(int i:mp[now])if(!vis[i])div(i);
}
int main()
{
   scanf("%d %d",&n,&k);
   for(i=1;i<n;i++)
   {
      int x,y;
      scanf("%d %d",&x,&y);
      mp[x].push_back(y),mp[y].push_back(x);
   }
   div(1);
   for(i=1;i<=n;i++)printf("%d%c",ans[i],i==n?'\n':' ');
   return 0;
}
```

3.17 DsuOnTree

```
#include<bits/stdc++.h>
```

```cpp
using namespace std;
const int MAXN=1e5+5;
int n,col[MAXN],a[MAXN],sz[MAXN],son[MAXN],cnt[MAXN],bigson,max1;
long long ans[MAXN],sum;
vector<int>G[MAXN];
void init() {
    for(int i=1; i<=n; i++)G[i].clear(),cnt[i]=0,max1=0;
}
void initdfs(int now,int fa) {
    sz[now]=1;
    for(auto to:G[now]) {
        if(to==fa)continue;
        initdfs(to,now);
        sz[now]+=sz[to];
        if(sz[to]>sz[son[now]])son[now]=to;//big son or not
    }
}
void add(int now,int fa,int val) {
    //operation
    cnt[col[now]]+=val;
    if(cnt[col[now]]>max1)max1=cnt[col[now]],sum=col[now];
    else if(cnt[col[now]]==max1)sum+=col[now];


    for(auto &to:G[now]) {
        if(to==fa||to==bigson)continue;
        add(to,now,val);
    }
}
void dfs(int now,int fa,int op) {
    for(auto &to:G[now]) {
        if(to==fa)continue;
        if(to!=son[now])dfs(to,now,0);//if(not big son) clear the influence
    }
    if(son[now]) { //as to big son
        dfs(son[now],now,1);//t clear influence
        bigson=son[now];//get big son position
    }
    add(now,fa,1),bigson=0;//count all small son ans
    ans[now]=sum;//update
    if(op==0) { //delete influence from small son
        add(now,fa,-1);
        sum=0;
        max1=0;
    }
}
int main() {
    cin>>n;
    for(int i=1; i<=n; i++)cin>>col[i];
    for(int i=1; i<n; i++) {
        int x,y;
        cin>>x>>y;
        G[x].push_back(y);
        G[y].push_back(x);
    }
    initdfs(1,-1);
    dfs(1,-1,0);
    for(int i=1; i<=n; i++)printf("%lld%c",ans[i],i==n?'\n':' ');
    return 0;
}
```

3.18 虚树

```
#include<bits/stdc++.h>
using namespace std;
#define mid (l+r)/2
const int maxn=100005;
int i,i0,n,m,dep[maxn],siz[maxn],fa[maxn],son[maxn],idx[maxn],cnt,top[maxn];
int dp[maxn][2];
vector<int>mp[maxn],mp0[maxn];
bool v[maxn];
void dfs(int now,int root,int deep)
{
    dep[now]=deep,fa[now]=root,siz[now]=1,son[now]=-1;
    for(auto i:mp[now])
    {
        if(i==root)continue;
        dfs(i,now,deep+1);
        siz[now]+=siz[i];
        if(son[now]==-1||siz[son[now]]<siz[i])son[now]=i;
    }
}
void dfs0(int now,int topf)
{
    idx[now]=++cnt,top[now]=topf;
    if(son[now]!=-1)dfs0(son[now],topf);
    for(auto i:mp[now])
    {
        if(i==fa[now]||i==son[now])continue;
        dfs0(i,i);
    }
}
int lca(int x,int y)
{
    while(top[x]!=top[y])
    {
        if(dep[top[x]]<dep[top[y]])swap(x,y);
        x=fa[top[x]];
    }
    if(dep[x]>dep[y])swap(x,y);
    return x;
}
vector<int>q;
bool cmp(int x,int y){return idx[x]<idx[y];}
stack<int>stk;
```

```
void link(int x,int y){mp[x].push_back(y),mp[y].push_back(x);}
void link0(int x,int y){mp0[x].push_back(y),mp0[y].push_back(x);}
void dfs(int now,int root)
{
    dp[now][0]=dp[now][1]=0;
    for(int i:mp0[now])
    {
        if(i!=root)
        {
            dfs(i,now);
            dp[now][1]+=dp[i][1],dp[now][0]+=dp[i][0];
            if(v[now])
            {
                if(v[i]&&fa[i]==now)dp[now][0]+=1000000;
                if(dp[i][1])dp[now][0]++;
            }
            v[i]=0;
        }
    }
    if(v[now])dp[now][1]=1;
    else
    {
        if(dp[now][1]>1)dp[now][1]=0,dp[now][0]++;
    }
    mp0[now].clear();
}
int main()
{
    int r=1;
    scanf("%d",&n);
    for(i=1;i<n;i++)
    {
        int x,y;
        scanf("%d %d",&x,&y);
        link(x,y);
    }
    dfs(r,-1,1),cnt=0,dfs0(r,r);
    scanf("%d",&m);
    while(m--)
    {
        int k,x;
        scanf("%d",&k);
        q.clear(),v[r]=0;
        while(k--)
        {
            scanf("%d",&x);
            if(x!=r)q.push_back(x);
            v[x]=1;
        }
        sort(q.begin(),q.end(),cmp);
        stk.push(r);
        for(auto i:q)
        {
            if(stk.empty())
            {
                stk.push(i);
                continue;
            }
            int p=lca(i,stk.top()),tmp=stk.top();
            stk.pop();
            while(!stk.empty()&&dep[stk.top()]>=dep[p])
            {
                link0(tmp,stk.top());
                tmp=stk.top(),stk.pop();
            }
            if(p!=tmp)link0(p,tmp),stk.push(p);
            else stk.push(tmp);
            stk.push(i);
        }
        x=stk.top(),stk.pop();
        while(!stk.empty())link0(x,stk.top()),x=stk.top(),stk.pop();
        dfs(r,-1);
        if(dp[r][0]>=1000000)dp[r][0]=-1;
        printf("%d\n",dp[r][0]);
    }
    return 0;
}
```

4 数据结构

4.1 莫队分块

```
#include<bits/stdc++.h>
using namespace std;
int i,n,m,T,a[50005],ans0[200005],bsize,cnt[1000005],p,p0,pl,pr,tim,ans,timp;
#define bel(x) ((x - 1) / bsize + 1)
struct node{int num,pos,i;}timer[50005];
struct query
{
    int l,r,i,p;
    bool operator < (const query &b) const
    {
        if(bel(l) != bel(b.l)) return l < b.l;
        if(bel(r) != bel(b.r)) return r < b.r;
        return i<b.i;
    }
}q[200005];
void change_time(int ntim)
{
    if(tim>ntim)
    {
        timp--;
        while(timp>=0&&timer[timp].i>ntim)
        {
            if(timer[timp].pos<=pr&&timer[timp].pos>=pl)
            {
                cnt[a[timer[timp].pos]]--;
                if(!cnt[a[timer[timp].pos]])ans--;
                swap(a[timer[timp].pos],timer[timp].num);
                if(!cnt[a[timer[timp].pos]])ans++;
                cnt[a[timer[timp].pos]]++;
            }
            else swap(a[timer[timp].pos],timer[timp].num);
            timp--;
        }
        timp++;
    }
    if(tim<ntim)
    {
        while(timp<p0&&timer[timp].i<ntim)
        {
            if(timer[timp].pos<=pr&&timer[timp].pos>=pl)
            {
                cnt[a[timer[timp].pos]]--;
                if(!cnt[a[timer[timp].pos]])ans--;
                swap(a[timer[timp].pos],timer[timp].num);
                if(!cnt[a[timer[timp].pos]])ans++;
                cnt[a[timer[timp].pos]]++;
            }
            else swap(a[timer[timp].pos],timer[timp].num);
            timp++;
        }
    }
    tim=ntim;
}
int main()
{

    while(scanf("%d",&n)!=EOF)
    {
        memset(cnt,0,sizeof(cnt));
        bsize=max(1,(int)pow(n,2.0/3));
        for(i=1;i<=n;i++)scanf("%d",&a[i]);
        scanf("%d",&m);
        for(i=1,p0=p=0;i<=m;i++)
        {
            int op;
            scanf("%d",&op);
            if(op==1)
            {
                scanf("%d %d",&q[p].l,&q[p].r);
                q[p].i=i;
                q[p].p=p;
                p++;
            }
            if(op==2)
            {
                scanf("%d %d",&timer[p0].pos,&timer[p0].num);
                timer[p0].i=i;
                p0++;
            }
        }
        sort(q,q+p);
        pl=1,pr=1,ans=1,tim=1,timp=0;
        cnt[a[1]]++;
        for(i=0;i<p;i++)
        {
            change_time(q[i].i);
            while(pl<q[i].l)
            {
                cnt[a[pl]]--;
                if(!cnt[a[pl]])ans--;
                pl++;
            }
            while(pl>q[i].l)
            {
                pl--;
                if(!cnt[a[pl]])ans++;
                cnt[a[pl]]++;
            }
            while(pr<q[i].r)
            {
                pr++;
                if(!cnt[a[pr]])ans++;
                cnt[a[pr]]++;
            }
            while(pr>q[i].r)
            {
                cnt[a[pr]]--;
                if(!cnt[a[pr]])ans--;
                pr--;
            }
            ans0[q[i].p]=ans;
        }
        for(i=0;i<p;i++)printf("%d\n",ans0[i]);
    }
    return 0;
}
```

4.2 带修改莫队分块

```
#include<bits/stdc++.h>
using namespace std;
int i,n,m,T,a[50005],ans0[200005],bsize,cnt[1000005],p,p0,pl,pr,tim,ans,timp;
#define bel(x) ((x - 1) / bsize + 1)
struct node{int num,pos,i;}timer[50005];
struct query
{
    int l,r,i,p;
    bool operator < (const query &b) const
    {
        if(bel(l) != bel(b.l)) return l < b.l;
        if(bel(r) != bel(b.r)) return r < b.r;
        return i<b.i;
    }
}q[200005];
void change_time(int ntim)
{
    if(tim>ntim)
    {
        timp--;
        while(timp>=0&&timer[timp].i>ntim)
        {
            if(timer[timp].pos<=pr&&timer[timp].pos>=pl)
            {
                cnt[a[timer[timp].pos]]--;
                if(!cnt[a[timer[timp].pos]])ans--;
                swap(a[timer[timp].pos],timer[timp].num);
                if(!cnt[a[timer[timp].pos]])ans++;
                cnt[a[timer[timp].pos]]++;
            }
            else swap(a[timer[timp].pos],timer[timp].num);
            timp--;
        }
        timp++;
    }
    if(tim<ntim)
    {
        while(timp<p0&&timer[timp].i<ntim)
        {
            if(timer[timp].pos<=pr&&timer[timp].pos>=pl)
            {
                cnt[a[timer[timp].pos]]--;
                if(!cnt[a[timer[timp].pos]])ans--;
                swap(a[timer[timp].pos],timer[timp].num);
                if(!cnt[a[timer[timp].pos]])ans++;
                cnt[a[timer[timp].pos]]++;
            }
            else swap(a[timer[timp].pos],timer[timp].num);
            timp++;
        }
    }
    tim=ntim;
}
int main()
{

    while(scanf("%d",&n)!=EOF)
    {
        memset(cnt,0,sizeof(cnt));
        bsize=max(1,(int)pow(n,2.0/3));
        for(i=1;i<=n;i++)scanf("%d",&a[i]);
        scanf("%d",&m);
        for(i=1,p0=p=0;i<=m;i++)
        {
            int op;
            scanf("%d",&op);
            if(op==1)
            {
                scanf("%d %d",&q[p].l,&q[p].r);
                q[p].i=i;
                q[p].p=p;
                p++;
            }
            if(op==2)
            {
                scanf("%d %d",&timer[p0].pos,&timer[p0].num);
                timer[p0].i=i;
                p0++;
            }
        }
        sort(q,q+p);
        pl=1,pr=1,ans=1,tim=1,timp=0;
        cnt[a[1]]++;
        for(i=0;i<p;i++)
        {
            change_time(q[i].i);
            while(pl<q[i].l)
            {
                cnt[a[pl]]--;
                if(!cnt[a[pl]])ans--;
                pl++;
            }
            while(pl>q[i].l)
            {
                pl--;
                if(!cnt[a[pl]])ans++;
                cnt[a[pl]]++;
            }
            while(pr<q[i].r)
            {
                pr++;
                if(!cnt[a[pr]])ans++;
                cnt[a[pr]]++;
            }
            while(pr>q[i].r)
            {
                cnt[a[pr]]--;
                if(!cnt[a[pr]])ans--;
                pr--;
            }
            ans0[q[i].p]=ans;
        }
        for(i=0;i<p;i++)printf("%d\n",ans0[i]);
    }
    return 0;
```

```
}
```

4.3 树状数组

```
//仅可用区间修改区间查询
int sum1[n],sum2[n],n;
void add(int p,int x){for(int i=p;i<=n;i+=i&-i)sum1[i]+=x,sum2[i]+=x*p;}
int ask(int p)
{
    int res=0;
    for(int i=p;i;i-=i&-i)res+=(p+1)*sum1[i]-sum2[i];
    return res;
}
void range_add(int l,int r,int x){add(l,x),add(r+1,-x);}
int range_ask(int l,int r){return ask(r)-ask(l-1);}

//单点修改 区间查询
int sum[n],n;
void add(int p, int x){while(p<=n)sum[p]+=x,p+=p&-p;}
int ask(int p)
{
    int res=0;
    while(p)res+=sum[p],p-=p&-p;
    return res;
}
int range_ask(int l, int r){return ask(r)-ask(l-1);}

void add(int x, int y, int z){
    int memo_y = y;
    while(x <= n){
        y = memo_y;
        while(y <= m)
            tree[x][y] += z,tree[x][y]%=mod, y += y & -y;
        x += x & -x;
    }
}
//区间修改 单点查询
void range_add(int xa, int ya, int xb, int yb, int z){//左上点和右下点
    add(xa, ya, z);
    add(xa, yb + 1, -z);
    add(xb + 1, ya, -z);
    add(xb + 1, yb + 1, z);
}
long long ask(int x, int y){
    int res = 0, memo_y = y;
    while(x){
        y = memo_y;
        while(y)
            res += tree[x][y],res%=mod, y -= y & -y;
        x -= x & -x;
    }
    res+=mod,res%=mod;
    return res;
}
```

4.4 线段树 (重载 + 最大子列和)

```
#include<bits/stdc++.h>
using namespace std;
#define mid ((l+r)/2)
const int maxn=300005;
int i,i0,n,m,a[maxn];
struct node
{
    long long sum,lm,rm,mx;
    friend node operator+(node a,node b)
    {
        node c;
        c.sum=a.sum+b.sum;
        c.lm=max(a.lm,a.sum+b.lm),c.rm=max(b.rm,b.sum+a.rm);
        c.mx=max(a.mx,b.mx),c.mx=max(c.mx,a.rm+b.lm);
        return c;
    }
}tree[maxn*4];
void b_tree(int l,int r,int p)
{
    if(l==r)tree[p].lm=tree[p].rm=tree[p].mx=max(0,a[l]),tree[p].sum=a[l];
    else
    {
        b_tree(l,mid,p*2),b_tree(mid+1,r,p*2+1);
        tree[p]=tree[p*2]+tree[p*2+1];
    }
}
void c_tree(int l,int r,int p,int x,int y)
{
    if(l==r)tree[p].lm=tree[p].rm=tree[p].mx=max(0,y),tree[p].sum=y;
    else
    {
        if(x<=mid)c_tree(l,mid,p*2,x,y);
        else c_tree(mid+1,r,p*2+1,x,y);
        tree[p]=tree[p*2]+tree[p*2+1];
    }
}
node q_tree(int l,int r,int p,int a,int b)
{
    if(l==a&&r==b)return tree[p];
    else
    {
        if(b<=mid)return q_tree(l,mid,p*2,a,b);
        else if(a>=mid+1)return q_tree(mid+1,r,p*2+1,a,b);
        else return q_tree(l,mid,p*2,a,mid)+q_tree(mid+1,r,p*2+1,mid+1,b);
    }
}
int main()
```

```
{
    scanf("%d %d",&n,&m);
    for(i=1;i<=n;i++)scanf("%d",&a[i]);
    b_tree(1,n,1);
    while(m--)
    {
        int l,r;
        scanf("%d %d",&l,&r);
        printf("%lld\n",q_tree(1,n,1,l,r).mx);
    }
    return 0;
}
```

4.5 可持久化线段树

```
#include<bits/stdc++.h>
using namespace std;
int i,i0,n,m,cnt,root[100005],head[100005];
struct node
{
    int l,r,sum;
}tree[2100005];
#define mid (l+r)/2
void build_tree(int l,int r,int &p)
{
    p=cnt++;
    if(l!=r)build_tree(l,mid,tree[p].l),build_tree(mid+1,r,tree[p].r);
    tree[p].sum=0;
}
void c_tree(int l,int r,int &p,int old,int a,int v)
{
    p=cnt++,tree[p]=tree[old];
    if(l==r)
    {
        tree[p].sum+=v;
        return;
    }
    if(a<=mid)c_tree(l,mid,tree[p].l,tree[old].l,a,v);
    else c_tree(mid+1,r,tree[p].r,tree[old].r,a,v);
    tree[p].sum=tree[tree[p].l].sum+tree[tree[p].r].sum;
}
int q_tree(int l,int r,int p,int a,int b)
{
    if(l==a&&r==b)return tree[p].sum;
    if(a>=mid+1)return q_tree(mid+1,r,tree[p].r,a,b);
    else if(b<=mid) return q_tree(l,mid,tree[p].l,a,b);
    else return q_tree(mid+1,r,tree[p].r,mid+1,b)+q_tree(l,mid,tree[p].l,a,mid);
}
struct point
{
    int num,i;
}a[100005];
```

```
bool cmp(point a,point b){return a.num<b.num;}
vector<int>v;
int main()
{
   int T;
   scanf("%d",&T);
   for(int cas=1;cas<=T;cas++)
   {
      printf("Case %d:\n",cas);
      cnt=0,v.clear();
      scanf("%d %d",&n,&m);
      for(i=1;i<=n;i++)scanf("%d",&a[i].num),a[i].i=i,v.push_back(a[i].num);
      sort(v.begin(),v.end()),v.erase(unique(v.begin(),v.end()),v.end());
      sort(a+1,a+1+n,cmp);
      build_tree(1,n,root[0]);
      for(i=1;i<=n;i++)
      {
         a[i].num=lower_bound(v.begin(),v.end(),a[i].num)-v.begin()+1;
         c_tree(1,n,root[i],root[i-1],a[i].i,1);
         head[a[i].num]=i;
      }
      while(m--)
      {
         int l,r,num;
         scanf("%d %d %d",&l,&r,&num);
         num=upper_bound(v.begin(),v.end(),num)-v.begin();
         printf("%d\n",q_tree(1,n,root[head[num]],l+1,r+1));
      }
   }
   return 0;
}
```

4.6 动态开点线段树

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=5000000;
#define mid (l+r)/2
long long cnt=0;
struct node
{
   long long lson,rson,v,sum;
}tree[maxn+5];
long long new_node()
{
   long long p=++cnt;
   tree[p].lson=tree[p].rson=0;
   tree[p].v=0,tree[p].sum=0;
   return p;
}
void p_tree(long long p)
{
   if(tree[tree[p].lson].v==tree[tree[p].rson].v)tree[p].v=tree[tree[p].lson].v;
   else tree[p].v=-1;
   tree[p].sum=tree[tree[p].lson].sum+tree[tree[p].rson].sum;
}
void new_son(int l,int r,int p){tree[p].lson=new_node(),tree[p].rson=new_node();}
void c_tree(long long l,long long r,long long p,long long L,long long R,long long x)
{
   if(l==L&&r==R)tree[p].v=x,tree[p].sum=(R-L+1)*x;
   else
   {
       if(!tree[p].lson)new_son(l,r,p);
       if(L<=mid)c_tree(l,mid,tree[p].lson,L,min(R,mid),x);
       if(R>=mid+1)c_tree(mid+1,r,tree[p].rson,max(mid+1,L),R,x);
       p_tree(p);
   }
}
void c0_tree(long long l,long long r,long long p,long long L,long long R,long long x
   )
{
   if(tree[p].sum==0)return;
   if(l==L&&r==R)
   {
       if(tree[p].v!=-1)
       {
           tree[p].v/=x,tree[p].sum=(R-L+1)*tree[p].v;
       }
       else
       {
           c0_tree(l,mid,tree[p].lson,l,mid,x),c0_tree(mid+1,r,tree[p].rson,mid+1,r,x
               );
           p_tree(p);
       }
   }
   else
   {
       if(!tree[p].lson)new_son(l,r,p);
       if(tree[p].v!=-1)
       {
           tree[tree[p].lson].v=tree[p].v,tree[tree[p].rson].v=tree[p].v;
           tree[tree[p].lson].sum=(mid-l+1)*tree[p].v,tree[tree[p].rson].sum=(r-mid)*
               tree[p].v;
       }
       if(L<=mid)c0_tree(l,mid,tree[p].lson,L,min(R,mid),x);
       if(R>=mid+1)c0_tree(mid+1,r,tree[p].rson,max(mid+1,L),R,x);
       p_tree(p);
   }
}
long long q_tree(long long l,long long r,long long p,long long L,long long R)
{
   if(tree[p].sum==0)return 0;
   if(tree[p].v!=-1)return (R-L+1)*tree[p].v;
   if(l==L&&r==R)return tree[p].sum;
   else
   {
       long long res=0;
       if(L<=mid&&tree[p].lson)res+=q_tree(l,mid,tree[p].lson,L,min(R,mid));
       if(R>=mid+1&&tree[p].rson)res+=q_tree(mid+1,r,tree[p].rson,max(mid+1,L),R);
       return res;
   }
}
int main()
{
   long long root=new_node(),n,m;
   scanf("%lld %lld",&n,&m);
   for(long long l=1,r=1;l<=n;l=r+1,r=(l<=n)?n/(n/l):0)c_tree(1,n,root,l,r,n/l);
   while(m--)
   {
       long long op,l,r,x;
       scanf("%lld %lld %lld",&op,&l,&r);
       if(op==1)
       {
           scanf("%lld",&x);
           if(x!=1)c0_tree(1,n,root,l,r,x);
       }
       else printf("%lld\n",q_tree(1,n,root,l,r));
   }
   return 0;
}
```

4.7 李超树

```
#define mid (l+r)/2
struct line
{
    long long k,b;
    bool flag;
}tree[400005];
long long cal(line a,long long pos){return a.k*pos+a.b;}
long long cross(line a,line b){return (a.b-b.b)/(b.k-a.k);}
//max
void c_tree(long long l,long long r,long long p,line k)
{
    if(!tree[p].flag)
    {
        tree[p]=k;
        if(l!=r)tree[p*2].flag=tree[p*2+1].flag=0;
    }
    else if(cal(k,l)>cal(tree[p],l)&&cal(k,r)>cal(tree[p],r))tree[p]=k;
    else if(cal(k,l)>cal(tree[p],l)||cal(k,r)>cal(tree[p],r))
    {
        if((cross(k,tree[p])<=mid)==(k.k>=tree[p].k))swap(k,tree[p]);
        if(cross(k,tree[p])<=mid)c_tree(l,mid,p*2,k);
        else c_tree(mid+1,r,p*2+1,k);
    }
}
```

```
long long q_tree(long long l,long long r,long long p,long long x)
{
    if(!tree[p].flag)return INT64_MIN;
    if(l==r)return cal(tree[p],x);
    else
    {
        long long ans=cal(tree[p],x);
        if(x<=mid)return max(ans,q_tree(l,mid,p*2,x));
        else return max(ans,q_tree(mid+1,r,p*2+1,x));
    }
}
//min and segment
void c_tree(int l,int r,int p,line k)
{
    if(k.l<=l&&r<=k.r)
    {
        if(!tree[p].flag)
        {
            tree[p]=k;
            if(l!=r)tree[p*2].flag=tree[p*2+1].flag=0;
        }
        else if(cal(k,l)<=cal(tree[p],l)&&cal(k,r)<=cal(tree[p],r))tree[p]=k;
        else if((cal(k,l)<=cal(tree[p],l))!=(cal(k,r)<=cal(tree[p],r)))
        {
            if((cross(k,tree[p])<=mid)!=(k.k>=tree[p].k))swap(k,tree[p]);
            if(cross(k,tree[p])<=mid)c_tree(l,mid,p*2,k);
            else c_tree(mid+1,r,p*2+1,k);
        }
    }
    else
    {
        int mid=(l+r)/2;
        if(k.l<=mid)c_tree(l,mid,p*2,k);
        if(mid<k.r)c_tree(mid+1,r,p*2+1,k);
    }
}
int q_tree(int l,int r,int p,int x)
{
    if(!tree[p].flag)return INT_MAX/2;
    if(l==r)return cal(tree[p],x);
    else
    {
        int ans=cal(tree[p],x);
        if(x<=mid)return min(ans,q_tree(l,mid,p*2,x));
        else return min(ans,q_tree(mid+1,r,p*2+1,x));
    }
}
```

4.8 整体二分

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

```cpp
int i,i0,n,m,ans[100005],bit[100005],f,bac[100005];
struct node
{
    int num,i,l,r,tpe;
}tmp,tmp0;
int lowbit(int t){return t&(-t);}
int sum(int i)
{
    int s=0;
    while(i>0)s+=bit[i],i-=lowbit(i);
    return s;
}
void add(int i,int x){while(i<=n+1)bit[i]+=x,i+=lowbit(i);}
queue<node>q[35][2];
void all_binary(int l,int r,int dep,bool f)
{
    if(q[dep][f].empty())return;
    if(l==r)
    {
        while(!q[dep][f].empty())
        {
            tmp=q[dep][f].front(),q[dep][f].pop();
            if(tmp.tpe==1)ans[tmp.i]=max(ans[tmp.i],min(tmp.r-tmp.l+1-tmp.num,l));
        }
        return;
    }
    int mid=(l+r+1)/2,tcnt;
    while(!q[dep][f].empty())
    {
        tmp=q[dep][f].front(),q[dep][f].pop();
        if(tmp.tpe==1)
        {
            tcnt=sum(tmp.r)-sum(tmp.l-1);
            if(tmp.r-tmp.l+1-tcnt-tmp.num<mid)q[dep+1][0].push(tmp);
            else tmp.num+=tcnt,q[dep+1][1].push(tmp);
        }
        if(tmp.tpe==2)
        {
            if(tmp.num<mid)
            {
                add(tmp.i,1);
                q[dep+1][0].push(tmp);
            }
            else
            {
                q[dep+1][1].push(tmp);
            }
        }
        if(tmp.tpe==3)
        {
            if(tmp.num<mid)
            {
                add(tmp.i,-1);
                q[dep+1][0].push(tmp);
            }
            else
            {
                q[dep+1][1].push(tmp);
            }
        }
    }
    all_binary(l,mid-1,dep+1,0);
    all_binary(mid,r,dep+1,1);
}
int main()
{
    while(scanf("%d %d",&n,&m)!=EOF)
    {
        memset(ans,-1,sizeof(ans));
        tmp.tpe=2;
        for(i=1;i<=n;i++)
        {
            scanf("%d",&bac[i]);
            tmp.num=bac[i],tmp.i=i;
            q[0][0].push(tmp);
        }
        for(i=0;i<m;i++)
        {
            scanf("%d",&tmp.tpe);
            if(tmp.tpe==1)scanf("%d %d",&tmp.l,&tmp.r),tmp.num=0,tmp.i=i,q[0][0].push(
                 tmp);
            if(tmp.tpe==2)
            {
                scanf("%d %d",&tmp.i,&tmp.num);
                tmp0.tpe=3,tmp0.i=tmp.i,tmp0.num=bac[tmp.i];
                q[0][0].push(tmp0);
                q[0][0].push(tmp);
                bac[tmp.i]=tmp.num;
            }
        }
        tmp.tpe=3;
        for(i=1;i<=n;i++)
        {
            tmp.num=bac[i],tmp.i=i;
            q[0][0].push(tmp);
        }
        all_binary(1,n,0,0);
        for(i=0;i<m;i++)if(ans[i]!=-1)printf("%d\n",ans[i]);
    }
    return 0;
}
```

4.9 扩展整体二分

```
#include<bits/stdc++.h>
```

```cpp
using namespace std;
#define mid ((l+r)/2)
const int maxn=100000;
int ans[3*maxn+5],tree[maxn*4+5];
struct node
{
    int tpe,id;//tpe为1表示插入，2表示查询,3表示删除。id表示编号，来确保输出答案时有序。
    int x,y,num;//num表示权值，x、y表示坐标。
    int lx,rx,ly,ry;//表示查询范围。
};
void c_tree(int l,int r,int p,int x,int num)
{
    if(l==r)
    {
        tree[p]=max(tree[p],num);
        if(!num)tree[p]=0;
    }
    else
    {
        if(x<=mid)c_tree(l,mid,p*2,x,num);
        else c_tree(mid+1,r,p*2+1,x,num);
        tree[p]=max(tree[p*2],tree[p*2+1]);
    }
}
int q_tree(int l,int r,int p,int L,int R)
{
    if(l==L&&r==R)return tree[p];
    else
    {
        if(R<=mid)return q_tree(l,mid,p*2,L,R);
        else if(L>mid)return q_tree(mid+1,r,p*2+1,L,R);
        else return max(q_tree(l,mid,p*2,L,mid),q_tree(mid+1,r,p*2+1,mid+1,R));
    }
}
queue<node>q[25][2];
void all_binary(int l,int r,int dep,bool f)
{
    if(q[dep][f].empty())return;
    while(!q[dep][f].empty())
    {
        node tmp=q[dep][f].front();
        q[dep][f].pop();
        if(tmp.tpe==1)
        {
            c_tree(1,maxn,1,tmp.x,tmp.num);
            if(l!=r)
            {
                if(tmp.y<=mid)q[dep+1][0].push(tmp);
                else q[dep+1][1].push(tmp);
            }
        }
        if(tmp.tpe==2)
        {
            if(tmp.ly==l&&tmp.ry==r)ans[tmp.id]=max(ans[tmp.id],q_tree(1,maxn,1,tmp.lx
                ,tmp.rx));
            else
            {
                if(l!=r)
                {
                    if(tmp.ly<=mid)
                    {
                        node tmp0=tmp;
                        tmp0.ry=min(tmp0.ry,mid);
                        q[dep+1][0].push(tmp0);
                    }
                    if(tmp.ry>mid)
                    {
                        node tmp0=tmp;
                        tmp0.ly=max(tmp0.ly,mid+1);
                        q[dep+1][1].push(tmp0);
                    }
                }
            }
        }
        if(tmp.tpe==3)
        {
            c_tree(1,maxn,1,tmp.x,0);
            if(l!=r)
            {
                if(tmp.y<=mid)q[dep+1][0].push(tmp);
                else q[dep+1][1].push(tmp);
            }
        }
    }
    all_binary(l,mid,dep+1,0),all_binary(mid+1,r,dep+1,1);
}
vector<node>v;
int main()
{
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        node tmp;
        ans[i]=-1;
        tmp.id=i;
        scanf("%d",&tmp.tpe);
        if(tmp.tpe==1)scanf("%d %d %d",&tmp.x,&tmp.y,&tmp.num),v.push_back(tmp);
        if(tmp.tpe==2)scanf("%d %d %d %d",&tmp.lx,&tmp.rx,&tmp.ly,&tmp.ry);
        q[0][0].push(tmp);
    }
    for(node tmp:v)
    {
        tmp.tpe=3;
        q[0][0].push(tmp);
    }
    all_binary(1,maxn,0,0);
    for(int i=1;i<=n;i++)if(ans[i]!=-1)printf("%d\n",ans[i]);
    return 0;
}
```

4.10 字典树

```
#include<bits/stdc++.h>
using namespace std;
int i,i0,n,m,tree[1000005][26],num,sum[1000005],cnt;
char s[11];
void c_tree()
{
    int len=strlen(s),root=0;
    for(int i=0;i<len;i++)
    {
        if(!tree[root][s[i]-'a'])tree[root][s[i]-'a']=++cnt;
        sum[tree[root][s[i]-'a']]++;
        root=tree[root][s[i]-'a'];
    }
}
int q_tree()
{
    int len=strlen(s),root=0;
    for(int i=0;i<len;i++)
    {
        root=tree[root][s[i]-'a'];
        if(!root)return 0;
    }
    return sum[root];
}
int main()
{
    while(gets(s)&&s[0]!='\0')c_tree();
    while(gets(s)!=NULL) printf("%d\n",q_tree());
    return 0;
}
```

4.11 块状链表

```
#include <cstdio>
#include <cstring>
#include<algorithm>
using namespace std;
const int maxn = 1500;
int pos, Size[maxn], List[maxn], Next[maxn];
char s[2000000], c, data[maxn][maxn], cmd[20];
int New_Node(void)
{
    return List[pos++];
}
```

```

void Del_Node(int t)
{
    List[--pos] = t;
}

void Find(int& p, int& b)
{
    for (b = 0; b != -1 && p > Size[b]; b = Next[b])
    {
        p -= Size[b];
    }
}

void Fill_Block(int b, int n, char* str, int e)
{
    if (b == -1)
    {
        return;
    }
    Next[b] = e;
    Size[b] = n;
    memcpy(data[b], str, n);
}

void Split(int b, int p)
{
    if (b == -1 || p == Size[b])
    {
        return;
    }
    int t = New_Node();
    Fill_Block(t, Size[b]-p, data[b]+p, Next[b]);
    Next[b] = t;
    Size[b] = p;
}

void Maintain(int b)
{
    for (; b != -1; b = Next[b])
    {
        for (int t = Next[b]; t != -1 && Size[b]+Size[t] <= maxn; t = Next[b])
        {
            memcpy(data[b]+ Size[b], data[t], Size[t]);
            Size[b] += Size[t];
            Next[b] = Next[t];
            Del_Node(t);
        }
    }
}

void Insert(int p, int n, char* str)
{
    int b, t, i;
    Find(p, b);
    Split(b, p);
    for (i = 0; i+maxn <= n; i += maxn)
    {
        t = New_Node();
        Fill_Block(t, maxn, str+i, Next[b]);
        Next[b] = t;
        b = t;
    }
    if (n-i)
    {
        t = New_Node();
        Fill_Block(t, n-i, str+i, Next[b]);
        Next[b] = t;
    }
    Maintain(b);
}

void Erase(int p, int n)
{
    int b, e;
    Find(p, b);
    Split(b, p);
    for (e = Next[b]; e != -1 && n > Size[e]; e = Next[e])
    {
        n -= Size[e];
    }
    Split(e, n);
    e = Next[e];
    for (int t = Next[b]; t != e; t = Next[b])
    {
        Next[b] = Next[t];
        Del_Node(t);
    }
    Maintain(b);
}

void Copy(int p, int n, char* str)
{
    int b, t, i;
    Find(p, b);
    i = min(n, Size[b]-p);
    memcpy(str, data[b]+p, i);
    for (t = Next[b]; t != -1 && i+Size[t] <= n; i += Size[t], t = Next[t])
    {
        memcpy(str+i, data[t], Size[t]);
    }
    if (n-i && t != -1)
    {
        memcpy(str+i, data[t], n-i);
    }
}
```

```

void Init(void)
{
    for (int i = 1; i < maxn; ++i)
    {
        List[i] = i;
    }
    pos = 1;
    Next[0] = -1;
    Size[0] = 0;
}

int main()
{

    int t, cur = 0, n;
    Init();
    scanf("%d", &t);
    while (t--)
    {
        scanf("%s", cmd);
        if (cmd[0] == 'M')
        {
            scanf("%d", &cur);
        }
        else if (cmd[0] == 'I')
        {
            scanf("%d", &n);
            for (int i = 0; i < n;)
            {
                c = getchar();
                if (32 <= c && c <= 126)
                {
                    s[i++] = c;
                }
            }
            s[n] = '\0';
            Insert(cur, n, s);
        }
        else if (cmd[0] == 'D')
        {
            scanf("%d", &n);
            Erase(cur, n);
        }
        else if (cmd[0] == 'G')
        {
            scanf("%d", &n);
            Copy(cur, n, s);
            s[n] = '\0';
            printf("%s\n", s);
        }
        else if (cmd[0] == 'P')
        {
            --cur;
        }
        else
        {
            ++cur;
        }
    }
    return 0;
}
```

5 数论

5.1 GCD、LCM、EXGCD

```
long long lcm(long long a,long long b){return a/__gcd(a,b)*b;}
int exgcd(int a,int b,int &x,int &y)
{
    if(b==0)
    {
        x=1,y=0;
        return a;
    }
    int r=exgcd(b,a%b,x,y),t=x;
    x=y,y=t-a/b*y;
    return r;
}
```

5.2 乘法逆元

```
void extgcd(long long a,long long b,long long& d,long long& x,long long& y)
{
    if(!b){d=a;x=1;y=0;}
    else{extgcd(b,a%b,d,y,x);y-=x*(a/b);}
}
long long inv(long long a,long long n)//要搿成正数来求
{
    if(a<0)a+=mod;
    long long d,x,y;
    extgcd(a,n,d,x,y);
    return d==1?(x+n)%n:-1;
}


inv[1]=1;
for(i=2;i<=7000;i++)inv[i]=(MOD-MOD/i)*inv[MOD%i]%MOD;

static int mod_inv(int a, int m = MOD) {
    // https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm#Example
    int g = m, r = a, x = 0, y = 1;

    while (r != 0) {
        int q = g / r;
        g %= r; swap(g, r);
        x -= q * y; swap(x, y);
    }

    return x < 0 ? x + m : x;
}
```

5.3 MillerRobin、PollardRho

```
#include<bits/stdc++.h>
using namespace std;
const int bace[5]={2,3,7,61,24251};
long long qmul(long long x,long long y,long long mod){return (x*y-(long long)((long
    double)x/mod*y)*mod+mod)%mod;}
long long qpow(long long a,long long b,long long mod){long long r=1,t=a; while(b){if
    (b&1)r=qmul(r,t,mod);b>>=1;t=qmul(t,t,mod);}return r;}
bool millerrabin(long long x)
{
    if(x==46856248255981ll||x<2)return false;
    if(x==2||x==3||x==7||x==61||x==24251)return true;
    long long ba=x-1,r;
    int ti=0,j;
    while(!(ba&1))ba>>=1,++ti;
    for(int i=0;i<=1;i++)
    {
        r=qpow(bace[i],ba,x);
        if(r==1||r==x-1)continue;
        for(j=1;j<=ti;++j)
        {
            r=qmul(r,r,x);
            if(r==x-1)break;
        }
        if(j>ti)return false;
    }
    return true;
}
mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());
long long pollardpho(long long n,int c)
{
    long long x,y,d;
    int i=1,k=2;
    y=x=rng()%(n-1)+1;
    while(++i)
    {
        x=(qmul(x,x,n)+c)%n;
        d=__gcd(y-x,n);
        if(d>1&&d<n)return d;
        if(x==y)return n;
        if(i==k)y=x,k<<=1;
    }
}
```

```cpp
vector<long long>v0;
void fin(long long x,int cnt)
{
    if(x==1)return;
    if(millerrabin(x))
    {
        v0.push_back(x);
        return;
    }
    long long p=x;
    while(p==x)p=pollardpho(x,cnt--);
    fin(p,cnt),fin(x/p,cnt);
}
int main()
{
    long long n,T;
    scanf("%lld",&T);
    while(T--)
    {
        scanf("%lld",&n);
        v0.clear();
        if(n==1)v0.push_back(1);
        fin(n,120);
        sort(v0.begin(),v0.end());
        for(int i=0;i<v0.size();i++)printf("%d%c",v0[i],i==v0.size()-1?'\n':' ');
    }
    return 0;
}
```

5.4 快速幂乘

```cpp
long long qpow(long long a,long long b,long long mod){long long r=1,t=a; while(b){if
    (b&1)r=(r*t)%mod;b>>=1;t=(t*t)%mod;}return r;}


long long qmul(long long x,long long y,long long mod){return (x*y-(long long)((long
    double)x/mod*y)*mod+mod)%mod;}
long long qpow(long long a,long long b,long long mod){long long r=1,t=a; while(b){if
    (b&1)r=qmul(r,t,mod);b>>=1;t=qmul(t,t,mod);}return r;}
```

5.5 矩阵快速幂

```cpp
#include<bits/stdc++.h>
using namespace std;
const int MAXN=4,MOD=10007;
struct MAT
{
    int mat[MAXN][MAXN];
    MAT operator*(const MAT &a)const
    {
        MAT b;
        memset(b.mat,0,sizeof(b.mat));
        for(int i=0;i<MAXN;i++)
        {
            for(int j=0;j<MAXN;j++)
            {
                for(int k=0;k<MAXN;k++)b.mat[i][j]=(b.mat[i][j]+mat[i][k]*a.mat[k][j]);
                b.mat[i][j]%=MOD;
            }
        }
        return b;
    }
};
MAT Mqpow(MAT base,int b)
{
    MAT r;
    for(int i=0;i<MAXN;i++)for(int j=0;j<MAXN;j++)r.mat[i][j]=i==j;
    while(b)
    {
        if(b&1)r=base*r;
        base=base*base;
        b>>=1;
    }
    return r;
}
int main()
{
    int n;
    while(scanf("%d",&n)!=EOF)
    {
        MAT start,r;
        //转移矩阵 横着为f[i][0]=2*f[i-1][0]+f[i-1][1]+f[i-1][2]....
        start.mat[0][0]=2,start.mat[0][1]=1,start.mat[0][2]=1,start.mat[0][3]=0;
        start.mat[1][0]=1,start.mat[1][1]=2,start.mat[1][2]=0,start.mat[1][3]=1;
        start.mat[2][0]=1,start.mat[2][1]=0,start.mat[2][2]=2,start.mat[2][3]=1;
        start.mat[3][0]=0,start.mat[3][1]=1,start.mat[3][2]=1,start.mat[3][3]=2;
        //r第一列为初始值 f[0][3]=1...;
        r.mat[0][0]=0,r.mat[0][1]=0,r.mat[0][2]=0,r.mat[0][3]=0;
        r.mat[1][0]=0,r.mat[1][1]=0,r.mat[1][2]=0,r.mat[1][3]=0;
        r.mat[2][0]=0,r.mat[2][1]=0,r.mat[2][2]=0,r.mat[2][3]=0;
        r.mat[3][0]=1,r.mat[3][1]=0,r.mat[3][2]=0,r.mat[3][3]=0;

        printf("%d\n",(Mqpow(start,n)*r).mat[3][0]);
    }
    return 0;
}
```

5.6 快速阶乘

```
#include<bits/stdc++.h>
using namespace std;
#define R register
#define ll long long
```

```
#define fp(i,a,b) for(R int i=(a),I=(b)+1;i<I;++i)
#define fd(i,a,b) for(R int i=(a),I=(b)-1;i>I;--i)
#define go(u) for(int i=head[u],v=e[i].v;i;i=e[i].nx,v=e[i].v)
const int N=(1<<17)+5;int P;
inline int add(R int x,R int y){return 0ll+x+y>=P?0ll+x+y-P:x+y;}
inline int dec(R int x,R int y){return x-y<0?x-y+P:x-y;}
inline int mul(R int x,R int y){return 1ll*x*y-1ll*x*y/P*P;}
int ksm(R int x,R int y){
    R int res=1;
    for(;y;y>>=1,x=mul(x,x))(y&1)?res=mul(res,x):0;
    return res;
}
const double Pi=acos(-1.0);
struct cp{
    double x,y;
    inline cp(){}
    inline cp(R double xx,R double yy):x(xx),y(yy){}
    inline cp operator +(const cp &b)const{return cp(x+b.x,y+b.y);}
    inline cp operator -(const cp &b)const{return cp(x-b.x,y-b.y);}
    inline cp operator *(const cp &b)const{return cp(x*b.x-y*b.y,x*b.y+y*b.x);}
    inline cp operator *(const double &b)const{return cp(x*b,y*b);}
    inline cp operator ~()const{return cp(x,-y);}
}w[2][N];
int r[21][N],ifac[N],lg[N],inv[N];double iv[21];
void Pre(){
    iv[0]=1;
    fp(d,1,17){
        fp(i,0,(1<<d)-1)r[d][i]=(r[d][i>>1]>>1)|((i&1)<<(d-1));
        lg[1<<d]=d,iv[d]=iv[d-1]*0.5;
    }
    inv[0]=inv[1]=ifac[0]=ifac[1]=1;
    fp(i,2,131072)inv[i]=mul(P-P/i,inv[P%i]),ifac[i]=mul(ifac[i-1],inv[i]);
    for(R int i=1,d=0;i<131072;i<<=1,++d)fp(k,0,i-1)
        w[1][i+k]=cp(cos(Pi*k*iv[d]),sin(Pi*k*iv[d])),
        w[0][i+k]=cp(cos(Pi*k*iv[d]),-sin(Pi*k*iv[d]));
}
int lim,d;
void FFT(cp *A,int ty){
    fp(i,0,lim-1)if(i<r[d][i])swap(A[i],A[r[d][i]]);
    cp t;
    for(R int mid=1;mid<lim;mid<<=1)
        for(R int j=0;j<lim;j+=(mid<<1))
            fp(k,0,mid-1)
                A[j+k+mid]=A[j+k]-(t=w[ty][mid+k]*A[j+k+mid]),
                A[j+k]=A[j+k]+t;
    if(!ty)fp(i,0,lim-1)A[i]=A[i]*iv[d];
}
void MTT(int *a,int *b,int len,int *c){
    static cp f[N],g[N],p[N],q[N];
    lim=len,d=lg[lim];
    fp(i,0,len-1)f[i]=cp(a[i]>>16,a[i]&65535),g[i]=cp(b[i]>>16,b[i]&65535);
    fp(i,len,lim-1)f[i]=g[i]=cp(0,0);
    FFT(f,1),FFT(g,1);
    fp(i,0,lim-1){
        cp t,f0,f1,g0,g1;
        t=~f[i?lim-i:0],f0=(f[i]-t)*cp(0,-0.5),f1=(f[i]+t)*0.5;
        t=~g[i?lim-i:0],g0=(g[i]-t)*cp(0,-0.5),g1=(g[i]+t)*0.5;
        p[i]=f1*g1,q[i]=f1*g0+f0*g1+f0*g0*cp(0,1);
    }
    FFT(p,0),FFT(q,0);
    fp(i,0,lim-1)c[i]=((((ll)(p[i].x+0.5)%P<<16)%P<<16)+((ll)(q[i].x+0.5)<<16)+((ll)
        (q[i].y+0.5)))%P;
}
void calc(int *a,int *b,int n,int k){
    static int f[N],g[N],h[N],sum[N],isum[N];
    int len=1;while(len<=n+n)len<<=1;
    fp(i,0,n)f[i]=mul(a[i],mul(ifac[i],ifac[n-i]));
    for(R int i=n-1;i>=0;i-=2)f[i]=P-f[i];
    int t=dec(k,n);
    fp(i,0,n+n)g[i]=add(i,t);
    sum[0]=g[0];fp(i,1,n+n)sum[i]=mul(sum[i-1],g[i]);
    isum[n+n]=ksm(sum[n+n],P-2);
    fd(i,n+n,1)isum[i-1]=mul(isum[i],g[i]);
    fp(i,1,n+n)g[i]=mul(isum[i],sum[i-1]);g[0]=isum[0];
    fp(i,n+1,len-1)f[i]=0;fp(i,n+n+1,len-1)g[i]=0;

    MTT(f,g,len,h);
    int res=1,p1=k-n,p2=k;
    fp(i,p1,p2)res=1ll*res*i%P;
    res=add(res,0);

    fp(i,0,n)g[i]=(0ll+P+p1+i)%P;
    sum[0]=g[0];fp(i,1,n)sum[i]=mul(sum[i-1],g[i]);
    isum[n]=ksm(sum[n],P-2);
    fd(i,n,1)isum[i-1]=mul(isum[i],g[i]);
    fp(i,1,n)g[i]=mul(isum[i],sum[i-1]);g[0]=isum[0];

    for(R int i=0;i<=n;p2=add(p2,1),++i)
        b[i]=mul(h[i+n],res),res=mul(res,mul(g[i],p2+1));
}
int solve(int bl){
    static int a[N],b[N],c[N];
    int s=0;for(int p=bl;p;p>>=1)++s;a[0]=1,--s;
    int qwq=ksm(bl,P-2);
    for(int p=0;s>=0;--s){
        if(p){
            calc(a,b,p,p+1);
            fp(i,0,p)a[p+i+1]=b[i];a[p<<1|1]=0;
            calc(a,b,p<<1,mul(p,qwq));
            p<<=1;fp(i,0,p)a[i]=mul(a[i],b[i]);
        }
        if(bl>>s&1){
            fp(i,0,p)a[i]=mul(a[i],(1ll*bl*i+p+1)%P);
            p|=1,a[p]=1;
            fp(i,1,p)a[p]=mul(a[p],(1ll*bl*p+i)%P);
        }
    }
    int res=1;
    fp(i,0,bl-1)res=mul(res,a[i]);
    return res;
}
int GetFac(int n){
    int s=sqrt(n),res=solve(s);
    fp(i,s*s+1,n)res=mul(res,i);
    return res;
}
int Fac(int n){
    if(n>P-1-n){
        int res=ksm(GetFac(P-1-n),P-2);
        return n&1?res:P-res;
    }
    return GetFac(n);
}
int n;
int main(){
    scanf("%d%d",&n,&P),Pre();
    printf("%d\n",Fac(n));
    return 0;
}
```

5.7 欧拉函数

```
int euler(int n)
{
    int ret=n,t=(int)sqrt(n*1.0);
    for(int i=2;i<=t;i++)
    {
        if(n%i==0)
        {
            ret=ret/i*(i-1);
            while(n%i==0)n/=i;
        }
    }
    if(n>1) ret=ret/n*(n-1);
    return ret;
}

for(i=1;i<=3000000;i++)p[i]=i;
for(i=2;i<=3000000;i+=2)p[i]/=2;
for(i=3;i<=3000000;i+=2)
if(p[i]==i)
{
    for(j=i;j<=3000000;j+=i)
    p[j]=p[j]/i*(i-1);
}
```

5.8 欧拉降幂

```
//a^b mod p
long long Dphi(int a,int b,int p)
{
    int mod=phi(p);
    return qpow(a,b<mod?b:b%mod+mod,p);
}
```

5.9 线性基

```
#include <bits/stdc++.h>
using namespace std;
int n,m,i;
struct Linebasis
{
    typedef unsigned int bint;
    const static int sz=32;
    bint p[sz];
    void init(){memset(p, 0, sizeof(p));}
    bool insert(bint x,bool f=1)
    {
        for(int i=sz-1;i>=0;i--)
        {
            if((x>>i)&1)
            {
                if(!p[i])
                {
                    if(f)p[i]=x;
                    break;
                }
                x^=p[i];
            }
        }
        return x;
    }
    Linebasis operator |(const Linebasis &r)const
    {
        Linebasis res=r;
        for(int i=0;i<sz;i++)if(p[i])res.insert(p[i]);
        return res;
    }
    Linebasis operator &(const Linebasis &r)const
    {
        Linebasis All,C,D;
        All.init(),C.init(),D.init();
        for (int i =sz-1;i>=0;i--)All.p[i]=this->p[i],D.p[i]=1ll<<i;
        for (int i =sz-1;i>=0;i--)
        {
            if(r.p[i])
            {
                bint v=r.p[i],k=0;
                bool f=1;
                for (int j=sz-1;j>=0;j--)
                {
                    if(v&(1ll<<j))
                    {
                        if (All.p[j])v^=All.p[j],k^=D.p[j];
                        else
                        {
                            f=0,All.p[j]=v,D.p[j]=k;
                            break;
                        }
                    }
                }
                if(f)
                {
                    bint v=0;
                    for (int j=sz-1;j>=0;j--)if(k&(1ll<<j))v^=this->p[j];
                    C.insert(v);
                }
            }
        }
        return C;
    }
    bint get_max(bint x)
    {
        for(int i=sz-1;i>=0;i--)x=max(x,x^p[i]);
        return x;
    }
    bint get_min(bint x)
    {
        for(int i=sz-1;i>=0;i--)x=min(x,x^p[i]);
        return x;
    }
}tree[50005*4];
vector<unsigned int>v[50005];
#define mid (l+r)/2
void b_tree(int l,int r,int p)
{
    if(l==r)for(unsigned int x:v[l])tree[p].insert(x);
    else
    {
        b_tree(l,mid,p*2),b_tree(mid+1,r,p*2+1);
        tree[p]=tree[p*2]&tree[p*2+1];
    }
}
bool q_tree(int l,int r,int p,int a,int b,int x)
{
    if(l==a&&r==b)return !tree[p].insert(x,0);
    else
    {
        if(b<=mid)return q_tree(l,mid,p*2,a,b,x);
        else if(a>=mid+1)return q_tree(mid+1,r,p*2+1,a,b,x);
        else return q_tree(l,mid,p*2,a,mid,x)&&q_tree(mid+1,r,p*2+1,mid+1,b,x);
    }
```

```
}
int main()
{
    scanf("%d %d",&n,&m);
    for(i=1;i<=n;i++)
    {
        int k;
        scanf("%d",&k);
        while(k--)
        {
            unsigned int x;
            scanf("%u",&x);
            v[i].push_back(x);
        }
    }
    b_tree(1,n,1);
    while(m--)
    {
        int l,r;
        unsigned int x;
        scanf("%d %d %u",&l,&r,&x);
        if(q_tree(1,n,1,l,r,x))printf("YES\n");
        else printf("NO\n");
    }
    return 0;
}
```

5.10 线性筛

```
const int MAXN=10000000;
vector<int>prime;
int check[MAXN+5],phi[MAXN+5],mob[MAXN+5],fac[MAXN+5],tim[MAXN+5],sd[MAXN+5],sp[MAXN
    +5];
void sieve()
{
    phi[1]=1; //欧拉筛
    mob[1]=1; //莫比乌斯函数筛
    fac[1]=1; //约数个数筛
    tim[1]=0; //最小质因子次幂
    sd[1]=1; //约数和筛
    sp[1]=1; //最小质因子的等比数列和 (1+p+p^2+p^3+...+p^k)
    for(int i=2;i<=MAXN;i++)
    {
        if(check[i]==0)
        {
            prime.push_back(i);
            phi[i]=i-1;
            mob[i]=-1;
            fac[i]=2;
            tim[i]=1;
            sd[i]=i+1;
            sp[i]=i+1;
        }
        for(int j=0;j<prime.size()&&i*prime[j]<=MAXN;j++)
        {
            check[i*prime[j]]=1;
            if(i%prime[j]==0)
            {
                phi[i*prime[j]]=phi[i]*prime[j];
                mob[i*prime[j]]=0;
                fac[i*prime[j]]=fac[i]/(tim[i]+1)*(tim[i]+2);
                tim[i*prime[j]]=tim[i]+1;
                sd[i*prime[j]]=sd[i]/sp[i]*(sp[i]*prime[j]+1);
                sp[i*prime[j]]=sp[i]*prime[j]+1;
                break;
            }
            phi[i*prime[j]]=phi[i]*(prime[j]-1);
            mob[i*prime[j]]=-mob[i];
            fac[i*prime[j]]=fac[i]*2;
            tim[i*prime[j]]=1;
            sd[i*prime[j]]=sd[i]*(prime[j]+1);
            sp[i*prime[j]]=prime[j]+1;
        }
    }
}
```

5.11 杜教筛

```
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
using namespace std;
int i,i0,n,m,T,ans;
const int MAXN=7000000;
vector<int>prime;
bool check[MAXN+5];
short mob[MAXN+5],premob[MAXN+5];
int phi[MAXN+5];
long long prephi[MAXN+5];
void sieve()
{
    prephi[1]=phi[1]=1; //欧拉筛
    premob[1]=mob[1]=1; //莫比乌斯函数筛
    for(int i=2;i<=MAXN;i++)
    {
        if(check[i]==0)
        {
            prime.push_back(i);
            phi[i]=i-1;
            mob[i]=-1;
        }
        for(int j=0;j<prime.size()&&i*prime[j]<=MAXN;j++)
        {
            check[i*prime[j]]=1;
            if(i%prime[j]==0)
            {
                phi[i*prime[j]]=phi[i]*prime[j];
                mob[i*prime[j]]=0;
                break;
            }
            phi[i*prime[j]]=phi[i]*(prime[j]-1);
            mob[i*prime[j]]=-mob[i];
        }
        prephi[i]=prephi[i-1]+phi[i];
        premob[i]=premob[i-1]+mob[i];
    }
}
namespace MU
{
    long long get_preFcovG(int n)
    {
        return 1;
    }
    long long get_preG(int n)
    {
        return n;
    }
    gp_hash_table<int,long long>F;
    long long get_preF(int n)
    {
        if(n<=MAXN)return premob[n];
        if(F.find(n)!=F.end())return F[n];
        long long ans=get_preFcovG(n);
        for(long long l=2,r;l<=n;l=r+1)
        {
            r=n/(n/l);
            ans-=(get_preG(r)-get_preG(l-1))*get_preF(n/l);
        }
        return F[n]=ans;
    }
}
namespace PHI
{
    unsigned long long get_preFcovG(int n)
    {
        return (1llu+n)*n/2;
    }
    long long get_preG(int n)
    {
        return n;
    }
    gp_hash_table<int,long long>F;
    long long get_preF(int n)
    {
        if(n<=MAXN)return prephi[n];
        if(F.find(n)!=F.end())return F[n];
        long long ans=get_preFcovG(n);
        for(long long l=2,r;l<=n;l=r+1)
        {
            r=n/(n/l);
            ans-=(get_preG(r)-get_preG(l-1))*get_preF(n/l);
        }
        return F[n]=ans;
    }
}
int main()
{
    sieve();
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d",&n);
        printf("%lld %lld\n",PHI::get_preF(n),MU::get_preF(n));
    }
    return 0;
}
```

5.12 组合数

```
long long C(int n,int m)
{
    if(m<n-m) m=n-m;
    long long ans=1;
    for(int i=m+1;i<=n;i++)ans*=i;
    for(int i=1;i<=n-m;i++)ans/=i;
    return ans;
}
//lucas定理 组合数求模
long long F[100010];
void init(long long p) //初始化
{
    F[0]=1;
    for(int i=1;i<=p;i++)F[i]=F[i-1]*i%(1000000007);
}
long long inv(long long a,long long m)
{
    if(a==1)return 1;
    return inv(m%a,m)*(m-m/a)%m;
}
long long Lucas(long long n,long long m,long long p) //n中取m个 m、p中有一个小于1e6
{
    long long ans=1;
    while(n&&m)
    {
        long long a=n%p,b=m%p;
        if(a<b)return 0;
        ans=ans*F[a]%p*inv(F[b]*F[a-b]%p,p)%p;
        n/=p;
        m/=p;
    }
    return ans;
}
//lucas2
long long qpow(long long x,long long y,long long MOD)
{
    long long ans=1;
    while(y)
    {
        if(y&1)ans=(x*ans)%MOD;
        x=(x*x)%MOD,y>>=1;
    }
    return ans;
}
long long C(long long n, long long m,long long p)
{
    if(m>n) return 0;
    long long ans=1;
    for(int i=1;i<=m;i++)
    {
        long long a=(n+i-m)%p,b=i%p;
        ans=ans*(a*qpow(b,p-2,p)%p)%p;
    }
    return ans;
}
long long Lucas(long long n,long long m,long long p)
{
    if(m==0)return 1;
    return C(n%p,m%p,p)*Lucas(n/p,m/p,p)%p;
}

//组合数逆元打表
//2
long long fac[maxn+5],inv[maxn+5];
long long qpow(long long a,long long b){long long r=1,t=a; while(b){if(b&1)r=(r*t)%
    mod;b>>=1;t=(t*t)%mod;}return r;}
long long C(long long n,long long m){return fac[n]*inv[m]%mod*inv[n-m]%mod;}
void init()
{
    fac[0]=fac[1]=1;
    for (int i=2;i<=maxn;i++)fac[i]=fac[i-1]*i%mod;
    inv[maxn]=qpow(fac[maxn],mod-2);
    for(int i=maxn-1;i>=0;i--)inv[i]=inv[i+1]*(i+1)%mod;
}
```

5.13 模系解码、分数间最小分子

```
#include<bits/stdc++.h>
using namespace std;
void euclid(long long pa,long long pb,long long qa,long long qb,long long &x,long
    long &y)//pa/pb<x/y<qa/qb min(x)
{
    long long z=(pa+pb-1)/pb;
    if(z<=qa/qb)
    {
        x=z,y=1;
        return;
    }
    pa-=(z-1)*pb,qa-=(z-1)*qb;
    euclid(qb,qa,pb,pa,y,x);
    x+=(z-1)*y;
}
void solve(long long x,long long p)
{
    long long a,b,y;
    euclid(p,x,p,x-1,b,y);
    a=b*x-p*y;
    printf("%lld/%lld\n",a,b);
}
int main()
{
    long long x,mod;
    while(scanf("%lld %lld",&x,&mod)!=EOF)solve(x,mod);
}
```

5.14 EXCRT

```
#include <cstdio>
const int MAXN = 100010;
typedef long long ll;
int n;
ll a[MAXN], b[MAXN], ans, M, x, y;
ll exgcd(ll a, ll b, ll &x, ll &y){
    if(!b){ x = 1; y = 0; return a; }
    ll d = exgcd(b, a % b, x, y);
    ll z = x; x = y; y = z - (a / b) * y;
    return d;
}
ll Slow_Mul(ll n, ll k, ll mod){
    ll ans = 0;
    while(k){
      if(k & 1) ans = (ans + n) % mod;
      k >>= 1;
      n = (n + n) % mod;
    }
    return ans;
}
//min x=a mod b
int main(){
    scanf("%d", &n);
    for(int i = 1; i <= n; ++i)
       scanf("%lld%lld", &b[i], &a[i]);
    ans = a[1];
    M = b[1];
    for(int i = 2; i <= n; ++i){
      ll B = ((a[i] - ans) % b[i] + b[i]) % b[i];
      ll GCD = exgcd(M, b[i], x, y);
      x = Slow_Mul(x, B / GCD, b[i]);
      ans += M * x;
      M *= b[i] / GCD;
      ans = (ans + M) % M;
    }
    printf("%lld\n", ans);
    return 0;
}
```

5.15 n 次同余

```
#include<bits/stdc++.h>
using namespace std;
long long quick_mod(long long a,long long b,long long mod)
{
    long long ans=1;
    while(b)
    {
        if(b&1)ans=ans*a%mod;
        b>>=1;
        a=a*a%mod;
    }
    return ans;
}
//快速幂
long long ex_gcd(long long a, long long b, long long &x, long long &y)
{
    if (b == 0)
    {
        x = 1, y = 0;
        return a;
    }
    else
    {
        long long r = ex_gcd(b, a % b, y, x);
        y -= x * (a / b);
        return r;
    }
}
//扩展欧几里得算法
vector<long long>v;
bool g_text(long long g,long long p)
{
    for(long long i=0;i<v.size();i++)if(quick_mod(g,(p-1)/v[i],p)==1)return 0;
    return 1;
}
long long primitive_root(long long p)
{
    long long tmp=p-1;
    for(long long i=2;i<=tmp/i;i++)
    {
        if(tmp%i==0)
        {
            v.push_back(i);
            while(tmp%i==0)tmp/=i;
        }
    }
    if(tmp!=1)v.push_back(tmp);
    long long g=1;
    while(1)
    {
        if(g_text(g,p))return g;
        ++g;
    }
}
//求解原根
struct sa
{
    long long x;
    int id;
    bool operator<(const sa &b)const
    {
        if (x == b.x) return id < b.id;
        return x<b.x;
    }
}rec[100500];
//用rec存离散对数
long long discerte_log(long long x,long long n,long long m)
{
    int s=(int)(sqrt((double)m+0.5));
    while((long long)s*s<=m)s++;
    long long cur=1;
    sa tmp;
    for(int i=0;i<s;i++)
    {
        tmp.x=cur,tmp.id=i;
        rec[i]=tmp;
        cur=cur*x%m;
    }
    sort(rec,rec+s);
    //这里不能用map查找比较慢, 采用排序二分就快了
    long long mul= quick_mod(cur, m - 2, m) % m;
    //这里有的方法是在下面的循环里求解快速幂, 但本题是不行的 要在循环外面弄, 保证时间
    cur=1;
    for(long long i=0;i<s;i++)
    {
        long long more=n*cur%m;
        tmp.x=more,tmp.id=-1;
        int j=lower_bound(rec,rec+s,tmp)-rec;
        if(rec[j].x==more)return i*s+rec[j].id;
        cur=cur*mul%m;
    }
    return -1;
}
//求解离散对数
vector<long long>residue(long long p,long long n,long long a)
{
    vector<long long>ret;
    if(a==0)
    {
        ret.push_back(0);
        return ret;
    }
    long long g=primitive_root(p),m=discerte_log(g,a,p);
    if(m==-1)return ret;
    long long A=n,B=p-1,C=m,x,y,G=ex_gcd(A,B,x,y);
    if(C%G!=0)return ret;
    x=x*(C/G)%B;
    long long delta=B/G;
    for(int i=0;i<G;i++)
    {
        x=((x+delta)%B+B)%B;
        ret.push_back(quick_mod(g,x,p));
    }
    sort(ret.begin(),ret.end());
    ret.erase(unique(ret.begin(),ret.end()),ret.end());
    return ret;
}
//求解n次剩余 X^A MOD P =B
int main()
{
    int t;
    scanf("%d",&t);
    while(t--){
        long long p,A,b;
        v.clear();
        scanf("%I64d%I64d%I64d",&p,&A,&b);
        vector<long long>ans;
        ans=residue(p,A,b);
        if(ans.empty()){
            puts("No Solution");
        }
        else {
            for(unsigned int i=0;i<ans.size();i++){
                printf("%I64d ",ans[i]);
            }
            puts("");
        }
    }
    return 0;
}
```

5.16 离散对数

```
//a^x=b mod p
#include<bits/stdc++.h>
using namespace std;
unordered_map<int,int>Hash;
#define mul(a,b,p) (1ll*(a)*(b)%p)
int gcd(int a,int b){return b?gcd(b,a%b):a;}
int exBSGS(int a,int b,int p)
{
    a%=p,b%=p;
    if(b==1) return 0;
    if(!b&&!a) return 1;
    if(!a) return -1;
    if(!b)
    {
        int ret=0,d;
        while((d=gcd(a,p))!=1)
        {
            ++ret,p/=d;
            if(p==1) return ret;
        }
        return -1;
    }
    int ret=0,A=a,B=b,P=p,C=1,d;
    while((d=gcd(A,P))!=1)
    {
        if(B%d) return -1;
        P/=d,B/=d;
        C=mul(C,A/d,P);
        ++ret;
        if(C==B) return ret;
    }
    Hash.clear();
    int f=1,t=sqrt(P)+1;
    for(int i=0;i<t;i++)
    {
        Hash[mul(f,B,P)]=i;
        f=mul(f,A,P);
    }
    int tf=f;
    f=mul(f,C,P);
    for(int i=1;i<=t;i++)
    {
        if(Hash.find(f)!=Hash.end()) return ret+i*t-Hash[f];
        f=mul(f,tf,P);
    }
    return -1;
}
int main()
{
    int a,p,b;
    while(scanf("%d %d %d",&a,&p,&b)&&(a||p||b))
    {
        int ans=exBSGS(a,b,p);
        if(~ans) printf("%d\n",ans);
        else puts("No Solution");
    }
    return 0;
}

//p-1的质因子只有2, 3
#include<bits/stdc++.h>
using namespace std;
int i,i0,n,m,T,ans,cnt2,cnt3;
long long qmul(long long x,long long y,long long mod){return (x*y-(long long)((long
    double)x/mod*y)*mod+mod)%mod;}
long long qpow(long long a,long long b,long long mod){long long r=1,t=a; while(b){if
    (b&1)r=qmul(r,t,mod);b>>=1;t=qmul(t,t,mod);}return r;}
bool check_root(int x,long long p)
{
    if(cnt2&&qpow(x,(p-1)/2,p)==1)return 0;
    if(cnt3&&qpow(x,(p-1)/3,p)==1)return 0;
    return 1;
}
int find_root(long long p){for(int i=2;1;i++)if(check_root(i,p))return i;}
long long cal(long long x,long long r,long long p)
{
    long long t=p-1,res=p-1,i,i0,i1;
    for(i=1,i0=1,i1=r;i<=cnt2;i++,i0=i0*2%p,i1=qmul(i1,i1,p))
    {
        t/=2;
        if(qpow(x,t,p)!=1)res=(res-i0)%p,x=qmul(x,i1,p);
    }
    for(i=1;i<=cnt3;i++,i0=i0*3%p,i1=qmul(i1,qmul(i1,i1,p),p))
    {
        t/=3;
        if(qpow(x,t,p)!=1)res=(res-i0)%p,x=qmul(x,i1,p);
        if(qpow(x,t,p)!=1)res=(res-i0)%p,x=qmul(x,i1,p);
    }
    return res;
}
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 r=exgcd(b,a%b,x,y),t=x;
    x=y,y=t-a/b*y;
    return r;
}
int main()
{
    scanf("%d",&T);
    while(T--)
    {
        long long a,b,p,r,A,B,x,y;
        scanf("%lld %lld %lld",&p,&a,&b);
        cnt2=cnt3=0;
        long long tmp=p-1;
        while(tmp%2==0)cnt2++,tmp/=2;
        while(tmp%3==0)cnt3++,tmp/=3;
        r=find_root(p);
        A=cal(a,r,p);
        B=cal(b,r,p);
        long long g=exgcd(A,p-1,x,y);
        if(B%g)printf("-1\n");
        else
        {
            y=(p-1)/g,x=qmul(x,B/g,y);
            printf("%lld\n",x);
        }
    }
    return 0;
}
```

5.17 BerlekampMassey、ReedsSloane

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

///BM: 解决递推式.请保证模数的平方不会爆long long!!!
///不用全抄,需要那一部分,就抄哪一部分.
using VI=vector<ll>;
class Linear_Seq{
public:
    static const int N = 50010;///多项式系数最大值
    ll res[N],c[N],md[N],COEF[N]/**COEF是多项式系数*/,Mod;
    vector<int> Md;
    inline static ll gcdEx(ll a, ll b, ll&x, ll& y)
    {
        if(!b) {x=1;y=0;return a;}
        ll d = gcdEx(b,a%b,y,x);
        y -= (a/b)*x;
        return d;
    }
    static ll Inv(ll a, ll Mod) {
        ll x, y;
        return gcdEx(a, Mod, x, y)==1?(x%Mod+Mod)%Mod:-1;
    };

    inline void mul(ll *a,ll *b,int k) {///下边的线性齐次递推用的.
        fill(c,c+2*k,0) ;
        for(int i(0);i<k;++i)if(a[i])for(int j(0);j<k;++j)
            c[i+j]=(c[i+j]+a[i]*b[j])%Mod;
        for (int i(2*k-1);i>=k;--i) if (c[i])for(size_t j(0);j<Md.size();++j)
            c[i-k+Md[j]]=(c[i-k+Md[j]]-c[i]*md[Md[j]])%Mod;
        copy(c,c+k,a) ;
    }

    int solve(ll n,VI A,VI B) { //线性齐次递推:A系数,B初值B[n]=A[0]*B[n-1]+...
        ///这里可以可以单独用,给出递推系数和前几项代替矩阵快速幂求递推式第n项.
        ll ans(0),cnt(0);
        int k(A.size());
        for(int i(0);i<k;++i) md[k-i-1]=-A[i];
        md[k]=1 ; Md.clear() ;
        for(int i(0);i<k;++i) {
            res[i] = 0 ;
            if (md[i]) Md.push_back(i);
        }
        res[0]=1;
        while ((1LL<<cnt)<=n) ++ cnt;
        for (int p(cnt);~p;-- p) {
            mul(res,res,k);
            if ((n>>p)&1) {
                copy(res,res+k,res+1) ; res[0]=0;
                for(size_t j(0);j<Md.size();++j)
                    res[Md[j]]=(res[Md[j]]-res[k]*md[Md[j]])%Mod;
            }
        }
        for(int i(0);i<k;++i) ans=(ans+res[i]*B[i])%Mod;
        return ans+(ans<0?Mod:0);
    }

    ///1-st***********模数是质数用这里******************/
    VI BM(VI s) {///BM算法求模数是质数的递推式子的通项公式,可以单独用
        VI C(1,1),B(1,1);
        int L(0),m(1),b(1);
        for(size_t n(0);n<s.size();++n) {
            ll d(0);
            for(int i(0);i<=L;++i) d=(d+(ll)C[i]*s[n-i])%Mod;
            if (!d) ++m;
            else {
                VI T(C);
                ll c(Mod-d*Inv(b,Mod)%Mod);
                while (C.size()<B.size()+m) C.push_back(0);
                for (size_t i(0);i<B.size();++i)
                    C[i+m]=(C[i+m]+c*B[i])%Mod;
                if (2*L<=(int)n) {L=n+1-L; B=T; b=d; m=1;}
                else ++m ;
            }
        }
        /** //下边这样写能够输出递推式的系数.
        printf("F[n] = ") ;
        for(size_t i(0);i<C.size();++i) {
            COEF[i+1] = min(C[i],Mod-C[i]) ;
            if(i>0) {
                if(i != 1) printf(" + ") ;
                printf("%lld*F[n-%d]",COEF[i+1],i+1) ;
                putchar(i+1==C.size()?'\n':' ') ;
            }
        }
        */
        return C;
    }
    ///1-ed************模数是质数用这里******************/





    ///2-st************模数非质数用这里******************/
    inline static void extand(VI &a, size_t d, ll value = 0) {
        if (d <= a.size()) return; a.resize(d, value);
    }
    static ll CRT(const VI &c, const VI &m) {///中国剩余定理合并
        int n(c.size());
        ll M(1), ans(0);
        for (int i = 0; i < n; ++i) M *= m[i];
        for (int i = 0; i < n; ++i) {
            ll x,y,tM(M / m[i]);
            gcdEx(tM, m[i], x, y);
            ans = (ans + tM * x * c[i] % M) % M;
        }
        return (ans + M) % M;
    }

    static VI ReedsSloane(const VI &s, ll Mod) {///求模数不是质数的递推式系数
        auto L = [](const VI &a, const VI &b) {
            int da = (a.size()>1||(a.size()== 1&&a[0]))?a.size()-1:-1000;
            int db = (b.size()>1||(b.size()== 1&&b[0]))?b.size()-1:-1000;
            return max(da, db + 1);
        };
        auto prime_power = [&](const VI &s, ll Mod, ll p, ll e) {
            vector<VI> a(e), b(e), an(e), bn(e), ao(e), bo(e);
            VI t(e), u(e), r(e), to(e, 1), uo(e), pw(e + 1);;
            pw[0] = 1;
            for (int i(pw[0] = 1); i <= e; ++i) pw[i] = pw[i - 1] * p;
            for (ll i(0); i < e; ++i) {
                a[i] = {pw[i]}; an[i] = {pw[i]};
                b[i] = {0}; bn[i] = {s[0] * pw[i] % Mod};
                t[i] = s[0] * pw[i] % Mod;
                if (!t[i]) {t[i] = 1; u[i] = e;}
                else for (u[i] = 0; t[i] % p == 0; t[i] /= p, ++u[i]);
            }
            for (size_t k(1);k < s.size(); ++k) {
                for (int g(0); g < e; ++g) {
                    if (L(an[g], bn[g]) > L(a[g], b[g])) {
                        int id (e-1-u[g]);
                        ao[g] = a[id]; bo[g] = b[id];
                        to[g] = t[id]; uo[g] = u[id];
                        r[g] = k - 1;
                    }
                }
                a = an; b = bn;
                for (int o(0); o < e; ++o) {
                   ll d(0);
                   for (size_t i(0); i < a[o].size() && i <= k; ++i)
                      d = (d + a[o][i] * s[k - i]) % Mod;
                   if (d == 0) {t[o] = 1;u[o] = e;}
                   else {
                      for (u[o]=0,t[o]=d;!(t[o]%p);t[o]/=p,++u[o]);
                      int g (e-1-u[o]);
                      if (!L(a[g], b[g])) {
                         extand(bn[o], k + 1);
                         bn[o][k] = (bn[o][k] + d) % Mod;
                      } else {
                         ll coef = t[o]*Inv(to[g],Mod)%Mod*pw[u[o]-uo[g]]%Mod;
                         int m(k-r[g]);
                         extand(an[o],ao[g].size()+m); extand(bn[o],bo[g].size()+m);
                         auto fun = [&](vector<VI> &vn,vector<VI> &vo,bool f) {
                            for (size_t i(0);i < vo[g].size(); ++i) {
                               vn[o][i+m] -= coef*vo[g][i]%Mod;
                               if (vn[o][i + m]<0) vn[o][i+m] += Mod*(f?1:-1);
                            }
                            while (vn[o].size() && !vn[o].back()) vn[o].pop_back();
                         } ;
                         fun(an,ao,1) ;fun(bn,bo,-1) ;
                      }
                   }
                }
             }
             return make_pair(an[0], bn[0]);
          };
          vector<tuple<ll, ll, int> > fac;
          for (ll i(2); i*i <= Mod; ++i)
             if (!(Mod % i)) {
                ll cnt(0),pw(1);
                while (!(Mod % i)) {Mod /= i; ++cnt; pw *= i;}
                fac.emplace_back(pw, i, cnt);
             }
          if (Mod > 1) fac.emplace_back(Mod, Mod, 1);
          vector<VI> as;
          size_t n = 0;
          for (auto &&x: fac) {
             ll Mod, p, e;
             VI a, b;
             std::tie(Mod, p, e) = x;
             auto ss = s;
             for (auto &&x: ss) x %= Mod;
             std::tie(a, b) = prime_power(ss, Mod, p, e);
             as.emplace_back(a);
             n = max(n, a.size());
          }
          VI a(n),c(as.size()),m(as.size());
          for (size_t i(0); i < n; ++i) {
            for (size_t j(0); j < as.size(); ++j) {
                m[j] = std::get<0>(fac[j]);
                c[j] = i < as[j].size() ? as[j][i] : 0;
            }
            a[i] = CRT(c, m);
        }
        return a;
    }
    ///2-ed***********模数非质数用这里******************/
    ll solve(VI a,ll n,ll Mod,bool prime=true) {
        VI c; this->Mod = Mod ;
        if(prime) c = BM(a);///如果已经知道系数了,直接输入到c就行了,不用调用BM().
        else c = ReedsSloane(a,Mod);
        c.erase(c.begin()) ;
        for(size_t i(0);i<c.size();++i) c[i] = (Mod-c[i])%Mod;
        return solve(n,c,VI(a.begin(),a.begin()+c.size()));
    }
}BMEX;
///BMEX.slove(初始值vector[从0开始],要得到的项数,模数,模数是不是质数)
///质数为1, 非质数为0
int f[2025],sum[2025];
ll quickpow(ll a,ll b,ll mod){
    ll ans=1;
    a%=mod;
    while(b){
        if(b&1)ans=ans*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return ans%mod;
}
int main(){
    ll mod=1e9,n,m;
    scanf("%lld%lld",&n,&m);
    VI G;
    f[0]=sum[0]=0;
    f[1]=sum[1]=1;
    for(int i=2;i<2020;i++){
        f[i]=(f[i-1]+f[i-2])%mod;
        sum[i]=(sum[i-1]+quickpow(f[i],m,mod))%mod;
    }
    for(int i=0;i<2020;i++)G.push_back(sum[i]);
    printf("%lld\n",BMEX.solve(G,n,mod,0));
    return 0;
}
```

5.18 拉格朗日插值法

```
#include<bits/stdc++.h>
using namespace std;
const int mod=1e9+7;
long long qpow(long long a,long long b){long long r=1,t=a; while(b){if(b&1)r=(r*t)%
```

```
    mod;b>>=1;t=(t*t)%mod;}return r;}
namespace polysum
{
    //先init前M项,然后计算
    const int D=1000005;
    long long a[D],f[D],g[D],p[D],p1[D],p2[D],b[D],h[D][2],C[D];
    long long calcn(int d,long long *a,long long n)
    {
        if (n<=d) return a[n];
        p1[0]=p2[0]=1;
        long long ans=0;
        for(int i=0;i<=d;i++)p1[i+1]=p1[i]*(n-i+mod)%mod,p2[i+1]=p2[i]*(n-d+i+mod)%
            mod;
        for(int i=0;i<=d;i++)
        {
            long long t=g[i]*g[d-i]%mod*p1[i]%mod*p2[d-i]%mod*a[i]%mod;
            if ((d-i)&1)ans=(ans-t+mod)%mod;
            else ans=(ans+t)%mod;
        }
        return ans;
    }
    void init(int M)
    {
        f[0]=f[1]=g[0]=g[1]=1;
        for(int i=2;i<=M+4;i++)f[i]=f[i-1]*i%mod;
        g[M+4]=qpow(f[M+4],mod-2);
        for(int i=M+3;i>=2;i--)g[i]=g[i+1]*(i+1)%mod;
    }
    long long polysum(long long n,long long *a,long long m)
    { // a[0].. a[m] \sum_{i=0}^{n-1} a[i]
        a[m+1]=calcn(m,a,m+1);
        for(int i=1;i<=m+1;i++)a[i]=(a[i-1]+a[i])%mod;
        return calcn(m+1,a,n-1);
    }
    long long qpolysum(long long R,long long n,long long *a,long long m)
    { // a[0].. a[m] \sum_{i=0}^{n-1} a[i]*R^i
        if(R==1)return polysum(n,a,m);
        a[m+1]=calcn(m,a,m+1);
        long long r=qpow(R,mod-2),p3=0,p4=0,c,ans;
        h[0][0]=0,h[0][1]=1;
        for(int i=1;i<=m+1;i++)h[i][0]=(h[i-1][0]+a[i-1])*r%mod,h[i][1]=h[i-1][1]*r%
            mod;
        for(int i=0;i<=m+1;i++)
        {
            long long t=g[i]*g[m+1-i]%mod;
            if (i&1) p3=((p3-h[i][0]*t)%mod+mod)%mod,p4=((p4-h[i][1]*t)%mod+mod)%mod;
            else p3=(p3+h[i][0]*t)%mod,p4=(p4+h[i][1]*t)%mod;
        }
        c=qpow(p4,mod-2)*(mod-p3)%mod;
        for(int i=0;i<=m+1;i++)h[i][0]=(h[i][0]+h[i][1]*c)%mod,C[i]=h[i][0];
        ans=(calcn(m,C,n)*qpow(R,n)-c)%mod;
        if(ans<0)ans+=mod;
        return ans;
    }
}
long long a[1000005];
int main()
{
    int n,k;
    scanf("%d %d",&n,&k);
    for(int i=0;i<=k;i++)a[i]=qpow(i,k);
    polysum::init(k);
    if(k==0)printf("%d\n",n);
    else printf("%lld\n",polysum::polysum(n+1,a,k));
    return 0;
}
```

5.19 高斯消元

```
bool gauss(int n)
{
    long long del;
    for(int i=1;i<=n;i++)
    {
        int k=i;
        for(int j=i+1;j<=n;j++)if(a[j][i])k=j;
        if((del=a[k][i])==0)return 0;
        long long invdel=inv(del);
        for(int j=i;j<=n+1;j++)swap(a[i][j],a[k][j]);
        for(int j=i;j<=n+1;j++)a[i][j]=a[i][j]*invdel%mod;
        for(k=1;k<=n;k++)if(k!=i)
        {
            del=a[k][i];
            for(int j=i;j<=n+1;j++)a[k][j]=(a[k][j]-a[i][j]*del%mod)%mod;
        }
    }
    return 1;
}

const int maxn = 100 + 5;
int n;
double a[maxn][maxn];//用二维数组存系数矩阵

inline void Gauss(){//高斯消元的主函数
    for(int i = 1;i <= n;i++){//选取一列作为消灭系数的对象
        for(int j = i;j <= n;j++)//挨个系数化一
            for(int k = n + 1;k >= i;k--)
                a[j][k] /= a[j][i];
        for(int j = i + 1;j <= n;j++)//挨个选取方程，减去系数（可能不太好理解，请自行手推
            弄懂高斯消元）
            for(int k = i;k <= n + 1;k++)
                a[j][k] -= a[i][k];
    }
    for(int i = n;i >= 1;i--)//直接在系数的那个位置乘上未知数，然后移到方程的等号右边
        for(int j = n;j >= i + 1;j--)
```

```cpp
            a[i][j] *= a[j][n + 1],a[i][n + 1] -= a[i][j];
}

inline bool judge(){//判断函数
    for(int i = 1;i <= n;i++)
        if(a[i][n + 1] != a[i][n + 1])//挨个寻找解，如果有某个解自己不等于自己，说明没有唯一解
            return false;
    return true;
}
```

5.20 Dirichlet

```cpp
void Dirichlet(long long *a,long long *b)//a*b
{
    memset(tmp,0,sizeof tmp);
    for(int i=1;i*i<=n;++i)
    {
        tmp[i*i]+=a[i]*b[i]%mod, Mod(tmp[i*i]);
        for(int j=i+1;i*j<=n;++j)//下边加上a[i]*b[j]和a[j]*b[i]，所以j从i+1开始即可
            (tmp[i*j]+=a[i]*b[j]%mod+a[j]*b[i]%mod)%=mod;//注意这加两个数不能一步用Mod取模。。
    }
    memcpy(a,tmp,sizeof tmp);
}
```

5.21 类欧几里德

```cpp
constexpr int mod = 998244353;
constexpr int inv2 = 499122177;
constexpr int inv6 = 166374059;
//f:ai+b/c向下取整1-n求和
long long f(long long a, long long b, long long c, long long n) {
    if (a == 0)
        return (n + 1) * (b / c) % mod;
    if (n == 0) return (b / c);
    if (a >= c || b >= c)
        return (f(a % c, b % c, c, n) + (a / c) * n % mod * (n + 1) % mod * inv2 % mod + (b / c) * (n + 1) % mod) % mod;
    long long m = (a * n + b) / c;
    return (n * m % mod - f(c, c - b - 1, a, m - 1)) % mod;
}
//g:ai+b/c向下取整的平方1-n求和
long long g(long long a, long long b, long long c, long long n) {
    if (a == 0) return (b / c) * n % mod * (n + 1) % mod * inv2 % mod;
    if (n == 0) return 0;
    if (a >= c || b >= c) return (g(a % c, b % c, c, n) + (a / c) * n % mod * (n + 1) % mod * (2 * n + 1) % mod * inv6 % mod + (b / c) * n % mod * (n + 1) % mod * inv2 % mod) % mod;
    long long m = (a * n + b) / c;
    return (n * (n + 1) % mod * m % mod - f(c, c - b - 1, a, m - 1) - h(c, c - b -
        1, a, m - 1)) % mod * inv2 % mod;
}
//f:i*(ai+b/c向下取整)1-n求和
long long h(long long a, long long b, long long c, long long n) {
    if (a == 0) return (n + 1) * (b / c) % mod * (b / c) % mod;
    if (n == 0) return (b / c) * (b / c) % mod;
    if (a >= c || b >= c)
        return ((a / c) * (a / c) % mod * n % mod * (n + 1) % mod * (2 * n + 1) % mod
            * inv6 % mod +
             (b / c) * (b / c) % mod * (n + 1) % mod + (a / c) * (b / c) % mod * n %
                 mod * (n + 1) % mod +
             h(a % c, b % c, c, n) + 2 * (a / c) % mod * g(a % c, b % c, c, n) % mod
                 +
             2 * (b / c) % mod * f(a % c, b % c, c, n) % mod) % mod;
    long long m = (a * n + b) / c;
    return (n * m % mod * (m + 1) % mod - 2 * g(c, c - b - 1, a, m - 1) - 2 * f(c, c
        - b - 1, a, m - 1) - f(a, b, c, n)) % mod;
}
```

6 其他

6.1 01 分数规划

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
#define fi first
#define se second
#define mem(a, b) memset(a, b, sizeof(a))
#define INF 0X3f3f3f3f
const ll MAXN = 5000 + 7;
const ll MOD = 1e9 + 7;
//-------------------------------------------//
int n, m;
struct pop
{
    int a, b;
    double v;
    bool operator<(const pop &a) const
    {
        return a.v > v;//a.v越小越好，所以从大到小排序
    }
} num[MAXN];
double check(double ans)
{
    double va = 0, vb = 0;
    for (int i = 1; i <= n; i++)
    {
        num[i].v = num[i].a - num[i].b * ans;//num[i].v是偏移量。
    }
    sort(num + 1, num + 1 + n);
```

```cpp
    for (int i = m + 1; i <= n;i++)
    {
        va+= num[i].a;
        vb += num[i].b;
    }
    return va / vb;
}
int main()
{
    while (scanf("%d %d", &n, &m) != EOF && n && m)
    {
        for (int i = 1; i <= n; i++)
        {
            scanf("%d", &num[i].a);
        }
        for (int i = 1; i <= n; i++)
        {
            scanf("%d", &num[i].b);
        }
        double temp = 0, ans = 1;
        while (fabs(temp - ans) > 1e-4)//精度
        {
            ans = temp;
            temp = check(temp);
        }
        printf("%.0lf\n", ans*100);
    }
}
```

6.2 单纯形法

```cpp
#include <bits/stdc++.h>
using namespace std;
const int maxn = 500; // 变量数目上限
const int maxm = 1500; // 约束数目上限
const double INF = 1e10;
const double eps = 1e-6;
struct Simplex
{
    int n; // 变量个数
    int m; // 约束个数
    double a[maxm+5][maxn+5]; // 输入矩阵
    double x[maxn+5];//方案
    int B[maxm+5], N[maxn+5]; // 算法辅助变量
    bool fn[maxn+5],fm[maxm+5];
    void pivot(int r, int c)
    {
        swap(N[c], B[r]);
        a[r][c] = 1 / a[r][c];
        if(fm[r])
        {
            fm[r]=0;
        for (int j = 0; j <= n; j++)
        {
            if (j != c) a[r][j] *= a[r][c];
            if(abs(a[r][j])>eps)fm[r]=1;
        }
        if(fn[c])
        {
            fn[c]=0;
            for (int i = 0; i <= m; i++)
            {
                if (i != r&&abs(a[i][c])>eps)
                {
                    for (int j = 0; j <= n; j++)if (j != c) a[i][j] -= a[i][c] * a[r
                        ][j];
                    a[i][c] = -a[i][c] * a[r][c];
                }
                if(abs(a[i][c])>eps)fn[c]=1;
            }
        }
    }
}
bool feasible()
{
    while(1)
    {
        int r, c;
        double p = INF;
        for (int i = 0; i < m; i++)if (a[i][n] < p) p = a[r = i][n];
        if (p > -eps) return true;
        p = 0;
        for (int i = 0; i < n; i++)if (a[r][i] < p) p = a[r][c = i];
        if (p > -eps) return false;
        p = a[r][n] / a[r][c];
        for (int i = r + 1; i < m; i++)
        {
            if (a[i][c] > eps)
            {
                double v = a[i][n] / a[i][c];
                if (v < p) r = i, p = v;
            }
        }
        pivot(r, c);
    }
}
// 解有界返回1，无解返回0，无界返回-1。b[i]为x[i]的值，ret为目标函数的值
int simplex()
{
    for (int i = 0; i < n; i++) N[i] = i,fn[i]=1;
    for (int i = 0; i < m; i++) B[i] = n + i,fm[i]=1;
    if (!feasible()) return 0;
    while(1)
    {
        int r, c;
```

```
            double p = 0;
            for (int i = 0; i < n; i++)if (a[m][i] > p) p = a[m][c = i];
            if (p < eps)
            {
                //输出方案
                for (int i = 0; i < n; i++)if (N[i] < n) x[N[i]] = 0;
                for (int i = 0; i < m; i++)if (B[i] < n) x[B[i]] = a[i][n];
                //a[m][n]就是最大值 但是好像有精度误差
                return 1;
            }
            p = INF;
            for (int i = 0; i < m; i++)
                if (a[i][c] > eps)
                {
                    double v = a[i][n] / a[i][c];
                    if (v < p) r = i, p = v;
                }
            if (p == INF) return -1;
            pivot(r, c);
        }
    }
}sp;
double a[maxn];
int main()
{
    int T;
    scanf("%d",&T);
    while(T--)
    {
        int n;
        scanf("%d",&n);
        for(int i=0;i<n;i++)scanf("%lf",&a[i]);
        int k,L,R;
        scanf("%d %d %d",&k,&L,&R);
        sp.n=n,sp.m=0;
        for(int i=0;i<n;i++)
        {
            for(int i0=0;i0<n;i0++)sp.a[sp.m][i0]=(i0==i);
            sp.a[sp.m++][n]=1;
        }
        for(int i=0;i+k-1<n;i++)
        {
            int l=i,r=i+k-1;
            for(int i0=0;i0<n;i0++)sp.a[sp.m][i0]=(i0>=l&&i0<=r);
            sp.a[sp.m++][n]=R;
            for(int i0=0;i0<n;i0++)sp.a[sp.m][i0]=-(i0>=l&&i0<=r);
            sp.a[sp.m++][n]=-L;
        }
        for(int i=0;i<n;i++)sp.a[sp.m][i]=a[i];
        sp.simplex();
        double ans=0;
        for(int i=0;i<n;i++)if(sp.x[i]>eps)ans+=a[i];
        printf("%.0f\n",ans);
        for(int i=0;i<n;i++)printf("%d",(sp.x[i]>eps));
        printf("\n");
    }
    return 0;
}
```

6.3 java 大数

```
1、新建一个值为123的大整数对象
BigInteger a=new BigInteger( “123” ); //第一种，参数是字符串
BigInteger a=BigInteger.valueOf(123); //第二种，参数可以是int、long
2、大整数的四则运算
a. add(b); //a,b均为BigInteger类型，加法
a.subtract(b); //减 法
a.divide(b); //除法
a.multiply(b); //乘法

3、大整数比较大小
a.equals(b); //如果a、b相等返回true否则返回false
a.comareTo(); //a小于b返回-1，等于返回0，大于返回1

4、常用方法
a.mod(b); //求余
a.gcd(b); //求最大公约数
a.max(b); //求最大值
a.min(b); //求最小值

5、BigInteger中的常数
BigInteger.ZERO //大整数0
BigInteger.ONE //大整数1
BigInteger.TEN //大整数10

Scanner cin = new Scanner(System.in);//读入
while(cin.hasNext()) {//等同于!=EOF
    BigInteger a;
    a = cin.BigInteger(); //读入一个BigInteger;
    System.out.println(a); //输出a并换行
}


1、A == B ?//高精度小数运算
题目描述：输入两个非常大的数A和B，判断A是否等于B，如果相等输出YES，否则输出NO
分析：这个题在hdu上实际上并没有给出范围，WA了之后才知道这是道大数题，因为仅仅涉及到输入、比
    较和输出，所以非常适合用作大数的入门题
注意：这里只是说给出两个数A和B，并没有说是两个整数，所以应该采用BigDecimal
import java.math.BigDecimal;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) { // TODO Auto-generated method stub
        BigDecimal a, b;
        Scanner cin = new Scanner(System.in);
        while (cin.hasNext()) {
            a = cin.nextBigDecimal();
            b = cin.nextBigDecimal();
            if (a.compareTo(b) == 0) System.out.println("YES");
            else System.out.println("NO");
        }
    }
}

BigDecimal的常用方法：
加：add（BigDecima）
减：subtract（BigDecimal）
乘：multiply（BigDecimal）
除：divide（BigDecimal）
乘方：pow（int）
取绝对值：abs（）
取反：negate（）
对比：compareTo（BigDecimal）
设置小数点精确度：setScale（int）
设置保留小数点精确度并添加保留方式（直接加1或者四舍五入）：setScale（int， int）
BigDecimal a=new BigDecimal("0.1000");
System.out.println(a.stripTrailingZeros().toPlainString());
String a = "1";//去掉后面无用的0；
String b = "4.56";
BigDecimal aBD = new BigDecimal(a);
BigDecimal bBD = new BigDecimal(b);
BigDecimal resultBD = aBD.divide(bBD).setScale(3, java.math.BigDecimal.ROUND_HALF_UP
    );

例子：
import java.util.*;
import java.math.*;
import java.math.BigInteger;
public class Main {
    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);
        while (in.hasNext()) {
            int n=in.nextInt();
            BigInteger ans=new BigInteger("1");
            while(n-->0)
            {
                BigInteger a = in.nextBigInteger();
                ans=ans.multiply(a);
            }
            System.out.println(ans);
        }
        in.close();
    }
}

import java.math.BigInteger;
import java.util.Scanner;
import java.math.*;
```

```java
public class Main {
    public static void main(String[] args) {
        Scanner in =new Scanner (System.in);
        while(in.hasNext()) {
            int t=in.nextInt();
            while(t-->0) {
                BigInteger m=in.nextBigInteger();
                BigInteger n=in.nextBigInteger();
                BigInteger n2=new BigInteger("1");
                BigInteger ans=new BigInteger("1");
                for(BigInteger i=n2;; i=i.add(n2)) {
                    if(i.compareTo(n)==0)
                        break;
                    ans=ans.multiply(m);
                }
                BigInteger temp=ans.gcd(n);
                BigInteger up=n.divide(temp);
                BigInteger down=ans.divide(temp);
                System.out.println(up+"/"+down);
            }
        }
    }
}
```

6.4 mini 读入挂

```cpp
namespace FastIO {
    inline int read() {
        char ch = getchar(); int r = 0, w = 1;
        while(!isdigit(ch)) {if(ch == '-') w = -1; ch = getchar();}
        while(isdigit(ch)) {r = r * 10 + ch - '0', ch = getchar();}
        return r * w;
    }
    void _write(int x) {
        if(x < 0) putchar('-'), x = -x;
        if(x > 9) _write(x / 10);
        putchar(x % 10 + '0');
    }
    inline void write(int x) {
        _write(x);
        puts("");
    }
}
```

6.5 FFT 快速傅里叶变换

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N=1<<18;
struct E
{
    double a,b;
    E(double A=0,double B=0){a=A;b=B;}
    E operator + (E y){return E(a+y.a,b+y.b);}
    E operator - (E y){return E(a-y.a,b-y.b);}
    E operator * (E y){return E(a*y.a-b*y.b,a*y.b+b*y.a);}
}w[N],a[N];
void fft(E *a,int n,int tp)
{
    for(int i=1,j=0; i<n; i++)
    {
        for(int k=(n>>1);!((j^=k)&k); k>>=1);
        if(i<j)swap(a[i],a[j]);
    }
    for(int j=2;j<=n;j<<=1)
    {
        w[0]=1,w[1]=E(cos(2*acos(-1)/j),tp*sin(2*acos(-1)/j));
        int m=(j>>1)-1;
        for(int i=2;i<=m;i++)w[i]=w[i-1]*w[1];
        for(int i=0; i<n; i+=j)
        {
            for(int k=0;k<=m;k++)
            {
                E x=a[i+k+(j>>1)]*w[k];
                a[i+k+(j>>1)]=a[i+k]-x;
                a[i+k]=a[i+k]+x;
            }
        }
    }
}
int x,n,m,i;
int main()
{
    scanf("%d %d",&n,&m);
    for(i=0;i<=n;i++)scanf("%d",&x),a[i].a=x;
    for(i=0;i<=m;i++)scanf("%d",&x),a[i].b=x;
    int r=1;
    while(r<=n+m)r*=2;
    fft(a,r,1);
    for(i=0;i<=r;i++)a[i]=a[i]*a[i];
    fft(a,r,-1);
    for(i=0;i<=n+m;i++)printf("%d ",int(a[i].b/(r*2)+0.5));
    return 0;
}
```

6.6 FWT 快速沃尔什变换

```
const ll mod = 1e9+7;
const int maxn = 6e5+10;
int a[maxn],b[maxn];
int sum;
ll rev = 5e8+4;
void FWT(int a[],int n)
```

```cpp
{
    for(int d=1;d<n;d<<=1)
        for(int m=d<<1,i=0;i<n;i+=m)
            for(int j=0;j<d;j++)
            {
                int x=a[i+j],y=a[i+j+d];
                a[i+j]=(x+y)%mod,a[i+j+d]=(x-y+mod)%mod;
                //xor:a[i+j]=x+y,a[i+j+d]=x-y;
                //and:a[i+j]=x+y;
                //or:a[i+j+d]=x+y;
            }
}

void UFWT(int a[],int n)
{
    for(int d=1;d<n;d<<=1)
        for(int m=d<<1,i=0;i<n;i+=m)
            for(int j=0;j<d;j++)
            {
                int x=a[i+j],y=a[i+j+d];
                a[i+j]=1LL*(x+y)*rev%mod,a[i+j+d]=(1LL*(x-y)*rev%mod+mod)%mod;
                //xor:a[i+j]=(x+y)/2,a[i+j+d]=(x-y)/2;
                //and:a[i+j]=x-y;
                //or:a[i+j+d]=y-x;
            }
}
```

6.7 NTT 快速数论变换

```cpp
#include<bits/stdc++.h>
using namespace std;
set<int>s1,s2;
const int MOD = 998244353;
struct mod_int
{
    int val;

    mod_int(long long v = 0)
    {
        if (v < 0)
            v = v % MOD + MOD;

        if (v >= MOD)
            v %= MOD;

        val = v;
    }

    static int mod_inv(int a, int m = MOD)
    {
        // https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm#Example
        int g = m, r = a, x = 0, y = 1;

        while (r != 0)
        {
            int q = g / r;
            g %= r;
            swap(g, r);
            x -= q * y;
            swap(x, y);
        }

        return x < 0 ? x + m : x;
    }

    explicit operator int() const
    {
        return val;
    }

    mod_int &operator+=(const mod_int &other)
    {
        val += other.val;
        if (val >= MOD)
            val -= MOD;
        return *this;
    }

    mod_int &operator-=(const mod_int &other)
    {
        val -= other.val;
        if (val < 0)
            val += MOD;
        return *this;
    }

    static unsigned fast_mod(uint64_t x, unsigned m = MOD)
    {
        return x % m;
        // Optimized mod for Codeforces 32-bit machines.
        // x must be less than 2^32 * m for this to work, so that x / m fits in a 32-
            bit integer.
        unsigned x_high = x >> 32, x_low = (unsigned)x;
        unsigned quot, rem;
        asm("divl %4\n"
            : "=a"(quot), "=d"(rem)
            : "d"(x_high), "a"(x_low), "r"(m));
        return rem;
    }

    mod_int &operator*=(const mod_int &other)
    {
        val = fast_mod((uint64_t)val * other.val);
        return *this;
    }

    mod_int &operator/=(const mod_int &other)
    {
        return *this *= other.inv();
    }

    friend mod_int operator+(const mod_int &a, const mod_int &b) { return mod_int(a) += b; }
    friend mod_int operator-(const mod_int &a, const mod_int &b) { return mod_int(a) -= b; }
    friend mod_int operator*(const mod_int &a, const mod_int &b) { return mod_int(a) *= b; }
    friend mod_int operator/(const mod_int &a, const mod_int &b) { return mod_int(a) /= b; }

    mod_int &operator++()
    {
        val = val == MOD - 1 ? 0 : val + 1;
        return *this;
    }

    mod_int &operator--()
    {
        val = val == 0 ? MOD - 1 : val - 1;
        return *this;
    }

    mod_int operator++(int)
    {
        mod_int before = *this;
        ++*this;
        return before;
    }
    mod_int operator--(int)
    {
        mod_int before = *this;
        --*this;
        return before;
    }

    mod_int operator-() const
    {
        return val == 0 ? 0 : MOD - val;
    }

    bool operator==(const mod_int &other) const { return val == other.val; }
    bool operator!=(const mod_int &other) const { return val != other.val; }

    mod_int inv() const
    {
        return mod_inv(val);
    }

```

```cpp
    mod_int pow(long long p) const
    {
        assert(p >= 0);
        mod_int a = *this, result = 1;

        while (p > 0)
        {
            if (p & 1)
                result *= a;

            a *= a;
            p >>= 1;
        }

        return result;
    }

    friend ostream &operator<<(ostream &stream, const mod_int &m)
    {
        return stream << m.val;
    }
};

namespace NTT
{
    vector<mod_int> roots = {0, 1};
    vector<int> bit_reverse;
    int max_size = -1;
    mod_int root;

    bool is_power_of_two(int n)
    {
        return (n & (n - 1)) == 0;
    }

    int round_up_power_two(int n)
    {
        assert(n > 0);

        while (n & (n - 1))
            n = (n | (n - 1)) + 1;

        return n;
    }

    // Given n (a power of two), finds k such that n == 1 << k.
    int get_length(int n)
    {
        assert(is_power_of_two(n));
        return __builtin_ctz(n);
    }

    // Rearranges the indices to be sorted by lowest bit first, then second lowest,
      etc., rather than highest bit first.
    // This makes even-odd div-conquer much easier.
    void bit_reorder(int n, vector<mod_int> &values)
    {
        if ((int)bit_reverse.size() != n)
        {
            bit_reverse.assign(n, 0);
            int length = get_length(n);

            for (int i = 0; i < n; i++)
                bit_reverse[i] = (bit_reverse[i >> 1] >> 1) + ((i & 1) << (length - 1))
                    ;
        }

        for (int i = 0; i < n; i++)
            if (i < bit_reverse[i])
                swap(values[i], values[bit_reverse[i]]);
    }

    void find_root()
    {
        int order = MOD - 1;
        max_size = 1;

        while (order % 2 == 0)
        {
            order /= 2;
            max_size *= 2;
        }

        root = 2;

        // Find a max_size-th primitive root of MOD.
        while (!(root.pow(max_size) == 1 && root.pow(max_size / 2) != 1))
            root++;
    }

    void prepare_roots(int n)
    {
        if (max_size < 0)
            find_root();

        assert(n <= max_size);

        if ((int)roots.size() >= n)
            return;

        int length = get_length(roots.size());
        roots.resize(n);

        // The roots array is set up such that for a given power of two n >= 2, roots
            [n / 2] through roots[n - 1] are
        // the first half of the n-th primitive roots of MOD.
        while (1 << length < n)
        {
            // z is a 2^(length + 1)-th primitive root of MOD.
            mod_int z = root.pow(max_size >> (length + 1));

            for (int i = 1 << (length - 1); i < 1 << length; i++)
            {
                roots[2 * i] = roots[i];
                roots[2 * i + 1] = roots[i] * z;
            }

            length++;
        }
    }

    void fft_iterative(int N, vector<mod_int> &values)
    {
        assert(is_power_of_two(N));
        prepare_roots(N);
        bit_reorder(N, values);

        for (int n = 1; n < N; n *= 2)
            for (int start = 0; start < N; start += 2 * n)
                for (int i = 0; i < n; i++)
                {
                    mod_int even = values[start + i];
                    mod_int odd = values[start + n + i] * roots[n + i];
                    values[start + n + i] = even - odd;
                    values[start + i] = even + odd;
                }
    }

    const int FFT_CUTOFF = 150;

    vector<mod_int> mod_multiply(vector<mod_int> left, vector<mod_int> right)
    {
        int n = left.size();
        int m = right.size();

        // Brute force when either n or m is small enough.
        if (min(n, m) < FFT_CUTOFF)
        {
            const uint64_t ULL_BOUND = numeric_limits<uint64_t>::max() - (uint64_t)MOD
                 * MOD;
            vector<uint64_t> result(n + m - 1);

            for (int i = 0; i < n; i++)
                for (int j = 0; j < m; j++)
                {
                    result[i + j] += (uint64_t)((int)left[i]) * ((int)right[j]);

                    if (result[i + j] > ULL_BOUND)
                        result[i + j] %= MOD;
                }

            for (uint64_t &x : result)
                if (x >= MOD)
                    x %= MOD;

            return vector<mod_int>(result.begin(), result.end());
        }

        int N = round_up_power_two(n + m - 1);
        left.resize(N);
        right.resize(N);

        bool equal = left == right;
        fft_iterative(N, left);

        if (equal)
            right = left;
        else
            fft_iterative(N, right);

        mod_int inv_N = mod_int(N).inv();

        for (int i = 0; i < N; i++)
            left[i] *= right[i] * inv_N;

        reverse(left.begin() + 1, left.end());
        fft_iterative(N, left);
        left.resize(n + m - 1);
        return left;
    }

    vector<mod_int> mod_power(const vector<mod_int> &v, int exponent)
    {
        assert(exponent >= 0);
        vector<mod_int> result = {1};

        if (exponent == 0)
            return result;

        for (int k = 31 - __builtin_clz(exponent); k >= 0; k--)
        {
            result = mod_multiply(result, result);

            if (exponent >> k & 1)
                result = mod_multiply(result, v);
        }

        return result;
    }
    } // namespace NTT
    int n, m,a[300005];
    vector<mod_int>v;
```

```
map<int,long long>mp;
int main()
{
    int n,k;
    scanf("%d %d",&n,&k);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&a[i]);
    }
    while(k--)
    {
        int x;
        scanf("%d",&x);

        s1.clear(),s2.clear();
        for(int i=1;i<=n;i++)
        {
            if(a[i]<x)
            {
                if(s1.count(a[i]))s2.insert(a[i]);
                else s1.insert(a[i]);
            }
        }
        vector<mod_int>v=NTT::mod_multiply(NTT::mod_power({1,2},s1.size()-s2.size()),
            NTT::mod_power({1,2,1},s2.size()));
        for(int i=0;i<v.size();i++)
        {
            mp[(x+1+i)*2]+=v[i].val;
            mp[(x+1+i)*2]%=MOD;
        }

    }
    int q;
    scanf("%d",&q);
    while(q--)
    {
        int x;
        scanf("%d",&x);
        printf("%lld\n",mp[x]);
    }
    return 0;
}
```

6.8 MTT 快速数论变换（任意模数）

```
#include <algorithm>
#include <cstdio>
#include <cstring>
int mod;
namespace Math {
    inline int pw(int base, int p, const int mod) {
        static int res;
        for (res = 1; p; p >>= 1, base = static_cast<long long> (base) * base % mod)
            if (p & 1) res = static_cast<long long> (res) * base % mod;
        return res;
    }
    inline int inv(int x, const int mod) { return pw(x, mod - 2, mod); }
}

const int mod1 = 998244353, mod2 = 1004535809, mod3 = 469762049, G = 3;
const long long mod_1_2 = static_cast<long long> (mod1) * mod2;
const int inv_1 = Math::inv(mod1, mod2), inv_2 = Math::inv(mod_1_2 % mod3, mod3);
struct Int {
    int A, B, C;
    explicit inline Int() { }
    explicit inline Int(int __num) : A(__num), B(__num), C(__num) { }
    explicit inline Int(int __A, int __B, int __C) : A(__A), B(__B), C(__C) { }
    static inline Int reduce(const Int &x) {
        return Int(x.A + (x.A >> 31 & mod1), x.B + (x.B >> 31 & mod2), x.C + (x.C >>
            31 & mod3));
    }
    inline friend Int operator + (const Int &lhs, const Int &rhs) {
        return reduce(Int(lhs.A + rhs.A - mod1, lhs.B + rhs.B - mod2, lhs.C + rhs.C -
             mod3));
    }
    inline friend Int operator - (const Int &lhs, const Int &rhs) {
        return reduce(Int(lhs.A - rhs.A, lhs.B - rhs.B, lhs.C - rhs.C));
    }
    inline friend Int operator * (const Int &lhs, const Int &rhs) {
        return Int(static_cast<long long> (lhs.A) * rhs.A % mod1, static_cast<long
            long> (lhs.B) * rhs.B % mod2, static_cast<long long> (lhs.C) * rhs.C %
            mod3);
    }
    inline int get() {
        long long x = static_cast<long long> (B - A + mod2) % mod2 * inv_1 % mod2 *
            mod1 + A;
        return (static_cast<long long> (C - x % mod3 + mod3) % mod3 * inv_2 % mod3 *
            (mod_1_2 % mod) % mod + x) % mod;
    }
} ;

#define maxn 131072

namespace Poly {
#define N (maxn << 1)
    int lim, s, rev[N];
    Int Wn[N | 1];
    inline void init(int n) {
        s = -1, lim = 1; while (lim < n) lim <<= 1, ++s;
        for (register int i = 1; i < lim; ++i) rev[i] = rev[i >> 1] >> 1 | (i & 1) <<
             s;
        const Int t(Math::pw(G, (mod1 - 1) / lim, mod1), Math::pw(G, (mod2 - 1) / lim
            , mod2), Math::pw(G, (mod3 - 1) / lim, mod3));
        *Wn = Int(1); for (register Int *i = Wn; i != Wn + lim; ++i) *(i + 1) = *i *
            t;
    }
    inline void NTT(Int *A, const int op = 1) {
        for (register int i = 1; i < lim; ++i) if (i < rev[i]) std::swap(A[i], A[rev[
            i]]);
        for (register int mid = 1; mid < lim; mid <<= 1) {
            const int t = lim / mid >> 1;
            for (register int i = 0; i < lim; i += mid << 1) {
                for (register int j = 0; j < mid; ++j) {
                    const Int W = op ? Wn[t * j] : Wn[lim - t * j];
                    const Int X = A[i + j], Y = A[i + j + mid] * W;
                    A[i + j] = X + Y, A[i + j + mid] = X - Y;
                }
            }
        }
        if (!op) {
            const Int ilim(Math::inv(lim, mod1), Math::inv(lim, mod2), Math::inv(lim,
                mod3));
            for (register Int *i = A; i != A + lim; ++i) *i = (*i) * ilim;
        }
    }
#undef N
}

int n, m;
Int A[maxn << 1], B[maxn << 1];
int main() {
    scanf("%d%d%d", &n, &m, &mod); ++n, ++m;
    for (int i = 0, x; i < n; ++i) scanf("%d", &x), A[i] = Int(x % mod);
    for (int i = 0, x; i < m; ++i) scanf("%d", &x), B[i] = Int(x % mod);
    Poly::init(n + m);
    Poly::NTT(A), Poly::NTT(B);
    for (int i = 0; i < Poly::lim; ++i) A[i] = A[i]^a * B[i]^b;
    Poly::NTT(A, 0);
    for (int i = 0; i < n + m - 1; ++i) {
        printf("%d", A[i].get());
        putchar(i == n + m - 2 ? '\n' : ' ');
    }
    return 0;
}
```

6.9 SG 函数

```
const int MN= 10000;
int data[MN],SG[MN],mid[MN];
void SG(int n)
{
    int i,j;
    for(i=1;i<=n;i++)
    {
        memset(mid,0,sizeof(mid));
        for(j=1;data[j]<=i;j++) mid[SG[i-data[j]]]=1;
        for(j=0;j<=n;j++)
        if(mid[j]==0)
        {
            SG[i]=j;
            break;
        }
    }
}

SJ定理
(1)游戏的SG函数不为0且游戏中某个单一游戏的SG函数大于 1; (2)游戏的SG函数为0且游戏中没有单一
    游戏的SG函数大于1。
```

6.10 第二类斯特林数

```
#include<bits/stdc++.h>
#define LL long long
using namespace std;
const int N=4e5+10;
const LL P=998244353,yg=3;
LL n,fac[N],inv[N],f[N],g[N],S2[N];
LL bin[N];

LL power(LL x,LL p) {
    LL ret=1;
    for (;p;p>>=1) {
        if (p&1) ret=(ret*x)%P;
        x=(x*x)%P;
    }
    return ret;
}

void NTT(LL *a,LL n,LL op) { //NTT:系数a数组, 长度为n, op=1求值op=-1插值
    for(LL i=0;i<n;i++) bin[i]=(bin[i>>1]>>1)|((i&1)*(n>>1));
    for(LL i=0;i<n;i++) if(i<bin[i]) swap(a[i],a[bin[i]]);
    for(LL i=1;i<n;i<<=1) {
        LL wn=power(yg,op==1?(P-1)/(2*i):(P-1)-(P-1)/(2*i)),w,t;
        for(LL j=0;j<n;j+=i<<1) {
            w=1;
            for(LL k=0;k<i;k++) {
                t=a[i+j+k]*w%P;w=w*wn%P;
                a[i+j+k]=(a[j+k]-t+P)%P;a[j+k]=(a[j+k]+t)%P;
            }
        }
    }
    if(op==-1) {
        LL Inv=power(n,P-2);
        for(LL i=0;i<n;i++) a[i]=a[i]*Inv%P;
    }
}
long long C(long long n,long long m){return fac[n]*inv[m]%P*inv[n-m]%P;}
int main()
{
    cin>>n;
    if(n==1)
    {
        printf("1\n");
        return 0;
    }
    if(n==2)
    {
        printf("2\n");
        return 0;
    }
    fac[0]=inv[0]=1;
    for (int i=1;i<=n;i++) fac[i]=fac[i-1]*i%P,inv[i]=power(fac[i],P-2);
    for (int i=0;i<=n;i++) f[i]=(power(-1,i)+P)%P*inv[i]%P;
    for (int i=0;i<=n;i++) g[i]=power(i,n-2)*inv[i]%P;

    LL N=n-1;
    LL len=1;while(len<(n+1)<<1) len<<=1;

    NTT(f,len,1); NTT(g,len,1);
    for (int i=0;i<len;i++) S2[i]=(f[i]*g[i])%P; //求f.g的卷积为S2
    NTT(S2,len,-1);

    long long ans=0;
    for(int i=1;i<=n-2;i++)
    {
        long long d=fac[i]*S2[i];
        d%=P;
        //printf("%d %lld %lld\n",n-i,d,S2[i]);
        ans+=d*C(n,i)%P*(n-i)%P;
        ans%=P;
    }
    ans%=P,ans+=P,ans%=P;
    printf("%lld\n",ans);
    return 0;
}
```

6.11 离散化

```
struct Discretization
{
    vector<int>v;
    void init(){v.clear();}
    void insert(int x){v.push_back(x);}
    void sortV(){sort(v.begin(),v.end()),v.erase(unique(v.begin(),v.end()),v.end())
        ;}
    int getPos(int x){return lower_bound(v.begin(),v.end(),x)-v.begin()+1;}
    int getVal(int pos){return v[pos-1];}
    int getSize(){return v.size();}
}D;
```

6.12 STL

```
int __builtin_ffs (unsigned int x)
返回x的最后一位1的是从后向前第几位，比如7368 (1110011001000) 返回4。
int __builtin_clz (unsigned int x)
返回前导的0的个数。
int __builtin_ctz (unsigned int x)
返回后面的0个个数，和__builtin_clz相对。
int __builtin_popcount (unsigned int x)
返回二进制表示中1的个数。
int __builtin_parity (unsigned int x)
返回x的奇偶校验位，也就是x的1的个数模2的结果。

此外，这些函数都有相应的usigned long和usigned long long版本，只需要在函数名后面加上l或ll
    就可以了，比如int __builtin_clzll。

mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());//随机数
shuffle(a, a + n, rng);

struct cmp
{
   bool operator()(int a,int b)
   {
      return dvv[a]<dvv[b];
   }
}
priority_queue<int,vector<int>,cmp>q;

multiset<int,greater<int>>sa;

ios::sync_with_stdio(false);

//读入一行
getline(cin,s);

#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/hash_policy.hpp>//用hash
using namespace __gnu_pbds;
cc_hash_table<long long,int>dp;
gp_hash_table<long long,int>dp;

struct pair_hash{inline size_t operator()(const pair<int,int> & p) const {return p.
    first*1007 + p.second;}};

#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
tree<int,null_type,less<int>,rb_tree_tag,tree_order_statistics_node_update>s;
s.order_of_key(n)

struct custom_hash {
   static uint64_t splitmix64(uint64_t x) {
      // http://xorshift.di.unimi.it/splitmix64.c
```

```
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }

    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().
            time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};

unordered_map<long long, int, custom_hash> safe_map;
gp_hash_table<long long, int, custom_hash> safe_hash_table;
```

6.13 星期

```
const int md
    [2][13]={{0,31,28,31,30,31,30,31,31,30,31,30,31},{0,31,29,31,30,31,30,31,31,30,31,30,31}};

struct Calendar
{
    int y,m,d;
    bool isLeap(){return (y%100!=0&&y%4==0||y%400==0);}
    bool isCalendar(){return !(y<1600||y>9999||m>12||m<1||d<1||d>md[isLeap()][m]);}
    int getWeek(){return (d+1+2*(m+(m<3)*12)+3*(m+(m<3)*12+1)/5+y-(m<3)+(y-(m<3))
        /4-(y-(m<3))/100+(y-(m<3))/400)%7;}//0为周日
    int getDist(Calendar x)
    {
        int res=(y-1)*365+(y-1)/4-(y-1)/100+y/400+d-((x.y-1)*365+(x.y-1)/4-(x.y-1)
            /100+(x.y-1)/400+x.d);
        for(int i=1;i<m;i++)res+=md[isLeap()][i];
        for(int i=1;i<x.m;i++)res-=md[x.isLeap()][i];
        return res;
    }
};
```

6.14 博弈

```
一、巴什博弈

    1、问题模型：只有一堆n个物品，两个人轮流从这堆物品中取物，规定每次至少取一个，最多取m
        个，最后取光者得胜。

    2、解决思路：当n=m+1时，由于一次最多只能取m个，所以无论先取者拿走多少个，后取者都能够一
        次拿走剩余的物品，后者取胜，所以当一方面对的局势是n%(m+1)=0时，其面临的是必败的局
        势。所以当n= (m+1)*r+s， (r为任意自然数，s≤m)时，如果先取者要拿走s个物品，如果后取
        者拿走x (≤m)个，那么先取者再拿走m+1-k个，结果剩下 (m+1) (r-1) 个，以后保持这样的取
        法，那么先取者肯定获胜。总之，要保持给对手留下 (m+1) 的倍数，就能最后获胜。

     3、变形：条件不变，改为最后取光的人输。

     结论：当（n-1）%（m+1）==0时后手胜利。

二、威佐夫博奕

     1、问题模型：有两堆各若干个物品，两个人轮流从某一堆或同时从两堆中取同样多的物品，规定每
        次至少取一个，多者不限，最后取光者得胜。

     2、解决思路：A：设（ai,bi）（ai ≤bi ,i=0，1，2，…,n)表示两堆物品的数量并称其为局势，
        如果甲面对（0，0），那么甲已经输了，这种局势我们称为奇异局势。前几个奇异局势是：（0
        ，0）、（1，2）、（3，5）、（4，7）、（6，10）、（8，13）、（9，15）、（11，18
        ）、（12，20）。任给一个局势（a，b），如下公式判断它是不是奇异局势： ak =[k（1+√5
        ）/2]，bk= ak + k （k=0，1，2，…,n 方括号表示取整函数）。（证明见百度百科）
三、Fibonacci博弈

     1、问题模型：

      有一堆个数为n的石子，游戏双方轮流取石子，满足：

      (1）先手不能在第一次把所有的石子取完；

      (2）之后每次可以取的石子数介于1到对手刚取的石子数的2倍之间（包含1和对手刚取的石子数的2
        倍）。 约定取走最后一个石子的人为赢家。

     2、解决思路：

       当n为Fibonacci数时，先手必败。即存在先手的必败态当且仅当石头个数为Fibonacci数。
四、尼姆博弈

     1、问题模型：有三堆各若干个物品，两个人轮流从某一堆取任意多的物品，规定每次至少取一个，
        多者不限，最后取光者得胜。

     2、解决思路：用（a，b，c）表示某种局势，显证（0，0，0）是第一种奇异局势，无论谁面对奇异
        局势，都必然失败。第二种奇异局势是（0，n，n），只要与对手拿走一样多的物品，最后都将
        导致（0，0，0）。

       搞定这个问题需要把必败态的规律找出：(a,b,c)是必败态等价于a^b^c=0(^表示异或运算）。
     3、推广一：如果我们面对的是一个非奇异局势（a，b，c），要如何变为奇异局势呢？假设 a < b<
         c,我们只要将 c 变为 a^b,即可,因为有如下的运算结果：a^b^(a^b)=(a^a)^(b^b)=0^0=0
        。要将c 变为a^b，只从 c中减去 c-（a^b）

     4、推广二：当石子堆数为n堆时，则推广为当对每堆的数目进行亦或之后值为零是必败态。
```

7 几何

7.1 二维几何

```cpp
// `计算几何模板`
const double eps = 1e-8;
const double inf = 1e20;
const double pi = acos(-1.0);
const int maxp = 1010;
//`判断小数和0是否相等`
int sgn(double x){
    if(fabs(x) < eps)return 0;
    if(x < 0)return -1;
    else return 1;
}
//浮点数的平方
inline double sqr(double x){return x*x;}
struct Point{
    double x,y;
    Point(){}
    Point(double _x,double _y){
        x = _x;
        y = _y;
    }
    void input(){
        scanf("%lf%lf",&x,&y);
    }
    void output(){
        printf("%.2f %.2f\n",x,y);
    }
    bool operator == (Point b)const{
        return sgn(x-b.x) == 0 && sgn(y-b.y) == 0;
    }
    bool operator < (Point b)const{
        return sgn(x-b.x)== 0?sgn(y-b.y)<0:x<b.x;
    }
    Point operator -(const Point &b)const{
        return Point(x-b.x,y-b.y);
    }
    //叉积
    double operator ^(const Point &b)const{
        return x*b.y - y*b.x;
    }
    //点积
    double operator *(const Point &b)const{
        return x*b.x + y*b.y;
    }
    //返回长度
    double len(){
        return hypot(x,y);//库函数
    }
    //返回长度的平方
    double len2(){
        return x*x + y*y;
    }
    //返回两点的距离
    double distance(Point p){
        return hypot(x-p.x,y-p.y);
    }
    Point operator +(const Point &b)const{
        return Point(x+b.x,y+b.y);
    }
    Point operator *(const double &k)const{
        return Point(x*k,y*k);
    }
    Point operator /(const double &k)const{
        return Point(x/k,y/k);
    }
    //`计算pa 和 pb 的夹角`
    //`就是求这个点看a,b 所成的夹角`
    //`测试 LightOJ1203`
    double rad(Point a,Point b){
        Point p = *this;
        return fabs(atan2( fabs((a-p)^(b-p)),(a-p)*(b-p) ));
    }
    //`化为长度为r的向量`
    Point trunc(double r){
        double l = len();
        if(!sgn(l))return *this;
        r /= l;
        return Point(x*r,y*r);
    }
    //`逆时针旋转90度`
    Point rotleft(){
        return Point(-y,x);
    }
    //`顺时针旋转90度`
    Point rotright(){
        return Point(y,-x);
    }
    //`绕着p点逆时针旋转angle`
    Point rotate(Point p,double angle){
        Point v = (*this) - p;
        double c = cos(angle), s = sin(angle);
        return Point(p.x + v.x*c - v.y*s,p.y + v.x*s + v.y*c);
    }
};
struct Line{
    Point s,e;
    Line(){}
    Line(Point _s,Point _e){
        s = _s;
        e = _e;
    }
    bool operator ==(Line v){
        return (s == v.s)&&(e == v.e);
    }
    //`根据一个点和倾斜角angle确定直线,0<=angle<pi`
    Line(Point p,double angle){
        s = p;
        if(sgn(angle-pi/2) == 0){
            e = (s + Point(0,1));
        }
        else{
            e = (s + Point(1,tan(angle)));
        }
    }
    //ax+by+c=0
    Line(double a,double b,double c){
        if(sgn(a) == 0){
            s = Point(0,-c/b);
            e = Point(1,-c/b);
        }
        else if(sgn(b) == 0){
            s = Point(-c/a,0);
            e = Point(-c/a,1);
        }
        else{
            s = Point(0,-c/b);
            e = Point(1,(-c-a)/b);
        }
    }
    void input(){
        s.input();
        e.input();
    }
    void adjust(){
        if(e < s)swap(s,e);
    }
    //求线段长度
    double length(){
        return s.distance(e);
    }
    //`返回直线倾斜角 0<=angle<pi`
    double angle(){
        double k = atan2(e.y-s.y,e.x-s.x);
        if(sgn(k) < 0)k += pi;
        if(sgn(k-pi) == 0)k -= pi;
        return k;
    }
    //`点和直线关系`
    //`1 在左侧`
    //`2 在右侧`
    //`3 在直线上`
    int relation(Point p){
        int c = sgn((p-s)^(e-s));
        if(c < 0)return 1;
        else if(c > 0)return 2;
        else return 3;
    }
    // 点在线段上的判断
    bool pointonseg(Point p){
        return sgn((p-s)^(e-s)) == 0 && sgn((p-s)*(p-e)) <= 0;
    }
    //`两向量平行(对应直线平行或重合)`
    bool parallel(Line v){
        return sgn((e-s)^(v.e-v.s)) == 0;
    }
    //`两线段相交判断`
    //`2 规范相交`
    //`1 非规范相交`
    //`0 不相交`
    int segcrossseg(Line v){
        int d1 = sgn((e-s)^(v.s-s));
        int d2 = sgn((e-s)^(v.e-s));
        int d3 = sgn((v.e-v.s)^(s-v.s));
        int d4 = sgn((v.e-v.s)^(e-v.s));
        if( (d1^d2)==-2 && (d3^d4)==-2 )return 2;
        return (d1==0 && sgn((v.s-s)*(v.s-e))<=0) ||
            (d2==0 && sgn((v.e-s)*(v.e-e))<=0) ||
            (d3==0 && sgn((s-v.s)*(s-v.e))<=0) ||
            (d4==0 && sgn((e-v.s)*(e-v.e))<=0);
    }
    //`直线和线段相交判断`
    //`-*this line -v seg`
    //`2 规范相交`
    //`1 非规范相交`
    //`0 不相交`
    int linecrossseg(Line v){
        int d1 = sgn((e-s)^(v.s-s));
        int d2 = sgn((e-s)^(v.e-s));
        if((d1^d2)==-2) return 2;
        return (d1==0||d2==0);
    }
    //`两直线关系`
    //`0 平行`
    //`1 重合`
    //`2 相交`
    int linecrossline(Line v){
        if((*this).parallel(v))
            return v.relation(s)==3;
        return 2;
    }
    //`求两直线的交点`
    //`要保证两直线不平行或重合`
    Point crosspoint(Line v){
        double a1 = (v.e-v.s)^(s-v.s);
        double a2 = (v.e-v.s)^(e-v.s);
        return Point((s.x*a2-e.x*a1)/(a2-a1),(s.y*a2-e.y*a1)/(a2-a1));
    }
    //点到直线的距离
    double dispointtoline(Point p){
        return fabs((p-s)^(e-s))/length();
    }
    //点到线段的距离
    double dispointtoseg(Point p){
        if(sgn((p-s)*(e-s))<0 || sgn((p-e)*(s-e))<0)
            return min(p.distance(s),p.distance(e));
        return dispointtoline(p);
    }
    //`返回线段到线段的距离`
    //`前提是两线段不相交，相交距离就是0了`
    double dissegtoseg(Line v){
        return min(min(dispointtoseg(v.s),dispointtoseg(v.e)),min(v.dispointtoseg(s),
            v.dispointtoseg(e)));
    }
    //`返回点p在直线上的投影`
    Point lineprog(Point p){
        return s + ( ((e-s)*((e-s)*(p-s)))/((e-s).len2()) );
    }
    //`返回点p关于直线的对称点`
    Point symmetrypoint(Point p){
        Point q = lineprog(p);
        return Point(2*q.x-p.x,2*q.y-p.y);
    }
};
//圆
struct circle{
    Point p;//圆心
    double r;//半径
    circle(){}
    circle(Point _p,double _r){
        p = _p;
        r = _r;
    }
    circle(double x,double y,double _r){
        p = Point(x,y);
        r = _r;
    }
    //`三角形的外接圆`
    //`需要Point的+ / rotate() 以及Line的crosspoint()`
    //`利用两条边的中垂线得到圆心`
    //`测试：UVA12304`
    circle(Point a,Point b,Point c){
        Line u = Line((a+b)/2,((a+b)/2)+((b-a).rotleft()));
        Line v = Line((b+c)/2,((b+c)/2)+((c-b).rotleft()));
        p = u.crosspoint(v);
        r = p.distance(a);
    }
    //`三角形的内切圆`
    //`参数bool t没有作用，只是为了和上面外接圆函数区别`
    //`测试：UVA12304`
    circle(Point a,Point b,Point c,bool t){
        Line u,v;
        double m = atan2(b.y-a.y,b.x-a.x), n = atan2(c.y-a.y,c.x-a.x);
        u.s = a;
        u.e = u.s + Point(cos((n+m)/2),sin((n+m)/2));
        v.s = b;
        m = atan2(a.y-b.y,a.x-b.x) , n = atan2(c.y-b.y,c.x-b.x);
        v.e = v.s + Point(cos((n+m)/2),sin((n+m)/2));
        p = u.crosspoint(v);
        r = Line(a,b).dispointtoseg(p);
    }
    //输入
    void input(){
        p.input();
        scanf("%lf",&r);
    }
    //输出
    void output(){
        printf("%.2lf %.2lf %.2lf\n",p.x,p.y,r);
    }
    bool operator == (circle v){
        return (p==v.p) && sgn(r-v.r)==0;
    }
    bool operator < (circle v)const{
        return ((p<v.p)||((p==v.p)&&sgn(r-v.r)<0));
    }
    //面积
    double area(){
        return pi*r*r;
    }
    //周长
    double circumference(){
        return 2*pi*r;
    }
    //`点和圆的关系`
    //`0 圆外`
    //`1 圆上`
    //`2 圆内`
    int relation(Point b){
        double dst = b.distance(p);
        if(sgn(dst-r) < 0)return 2;
        else if(sgn(dst-r)==0)return 1;
        return 0;
    }
    //`线段和圆的关系`
    //`比较的是圆心到线段的距离和半径的关系`
    int relationseg(Line v){
        double dst = v.dispointtoseg(p);
        if(sgn(dst-r) < 0)return 2;
        else if(sgn(dst-r) == 0)return 1;
        return 0;
    }
    //`直线和圆的关系`
    //`比较的是圆心到直线的距离和半径的关系`
    int relationline(Line v){
        double dst = v.dispointtoline(p);
        if(sgn(dst-r) < 0)return 2;
        else if(sgn(dst-r) == 0)return 1;
        return 0;
    }
    //`两圆的关系`
    //`5 相离`
    //`4 外切`
    //`3 相交`
    //`2 内切`
    //`1 内含`
    //`需要Point的distance`
    //`测试：UVA12304`
    int relationcircle(circle v){
        double d = p.distance(v.p);
        if(sgn(d-r-v.r) > 0)return 5;
        if(sgn(d-r-v.r) == 0)return 4;
        double l = fabs(r-v.r);
        if(sgn(d-r-v.r)<0 && sgn(d-l)>0)return 3;
        if(sgn(d-l)==0)return 2;
        if(sgn(d-l)<0)return 1;
    }
    //`求两个圆的交点，返回0表示没有交点，返回1是一个交点，2是两个交点`
    //`需要relationcircle`
    //`测试：UVA12304`
    int pointcrosscircle(circle v,Point &p1,Point &p2){
        int rel = relationcircle(v);
        if(rel == 1 || rel == 5)return 0;
        double d = p.distance(v.p);
        double l = (d*d+r*r-v.r*v.r)/(2*d);
        double h = sqrt(r*r-l*l);
        Point tmp = p + (v.p-p).trunc(l);
        p1 = tmp + ((v.p-p).rotleft().trunc(h));
        p2 = tmp + ((v.p-p).rotright().trunc(h));
        if(rel == 2 || rel == 4)
            return 1;
        return 2;
    }
    //`求直线和圆的交点，返回交点个数`
    int pointcrossline(Line v,Point &p1,Point &p2){
        if(!(*this).relationline(v))return 0;
        Point a = v.lineprog(p);
        double d = v.dispointtoline(p);
        d = sqrt(r*r-d*d);
        if(sgn(d) == 0){
            p1 = a;
            p2 = a;
            return 1;
        }
        p1 = a + (v.e-v.s).trunc(d);
        p2 = a - (v.e-v.s).trunc(d);
        return 2;
    }
    //`得到过a,b两点，半径为r1的两个圆`
    int gercircle(Point a,Point b,double r1,circle &c1,circle &c2){
        circle x(a,r1),y(b,r1);
        int t = x.pointcrosscircle(y,c1.p,c2.p);
        if(!t)return 0;
        c1.r = c2.r = r;
        return t;
    }
    //`得到与直线u相切，过点q,半径为r1的圆`
```

```
//`测试：UVA12304`
int getcircle(Line u,Point q,double r1,circle &c1,circle &c2){
   double dis = u.dispointtoline(q);
   if(sgn(dis-r1*2)>0)return 0;
   if(sgn(dis) == 0){
      c1.p = q + ((u.e-u.s).rotleft().trunc(r1));
      c2.p = q + ((u.e-u.s).rotright().trunc(r1));
      c1.r = c2.r = r1;
      return 2;
   }
   Line u1 = Line((u.s + (u.e-u.s).rotleft().trunc(r1)),(u.e + (u.e-u.s).rotleft
       ().trunc(r1)));
   Line u2 = Line((u.s + (u.e-u.s).rotright().trunc(r1)),(u.e + (u.e-u.s).
       rotright().trunc(r1)));
   circle cc = circle(q,r1);
   Point p1,p2;
   if(!cc.pointcrossline(u1,p1,p2))cc.pointcrossline(u2,p1,p2);
   c1 = circle(p1,r1);
   if(p1 == p2){
      c2 = c1;
      return 1;
   }
   c2 = circle(p2,r1);
   return 2;
}
//`同时与直线u,v相切，半径为r1的圆`
//`测试：UVA12304`
int getcircle(Line u,Line v,double r1,circle &c1,circle &c2,circle &c3,circle &
    c4){
   if(u.parallel(v))return 0;//两直线平行
   Line u1 = Line(u.s + (u.e-u.s).rotleft().trunc(r1),u.e + (u.e-u.s).rotleft().
       trunc(r1));
   Line u2 = Line(u.s + (u.e-u.s).rotright().trunc(r1),u.e + (u.e-u.s).rotright
       ().trunc(r1));
   Line v1 = Line(v.s + (v.e-v.s).rotleft().trunc(r1),v.e + (v.e-v.s).rotleft().
       trunc(r1));
   Line v2 = Line(v.s + (v.e-v.s).rotright().trunc(r1),v.e + (v.e-v.s).rotright
       ().trunc(r1));
   c1.r = c2.r = c3.r = c4.r = r1;
   c1.p = u1.crosspoint(v1);
   c2.p = u1.crosspoint(v2);
   c3.p = u2.crosspoint(v1);
   c4.p = u2.crosspoint(v2);
   return 4;
}
//`同时与不相交圆cx,cy相切，半径为r1的圆`
//`测试：UVA12304`
int getcircle(circle cx,circle cy,double r1,circle &c1,circle &c2){
   circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
   int t = x.pointcrosscircle(y,c1.p,c2.p);
   if(!t)return 0;
   c1.r = c2.r = r1;
   return t;
```

```
    }

    //`过一点作圆的切线(先判断点和圆的关系)`
    //`测试：UVA12304`
    int tangentline(Point q,Line &u,Line &v){
        int x = relation(q);
        if(x == 2)return 0;
        if(x == 1){
            u = Line(q,q + (q-p).rotleft());
            v = u;
            return 1;
        }
        double d = p.distance(q);
        double l = r*r/d;
        double h = sqrt(r*r-l*l);
        u = Line(q,p + ((q-p).trunc(l) + (q-p).rotleft().trunc(h)));
        v = Line(q,p + ((q-p).trunc(l) + (q-p).rotright().trunc(h)));
        return 2;
    }
    //`求两圆相交的面积`
    double areacircle(circle v){
        int rel = relationcircle(v);
        if(rel >= 4)return 0.0;
        if(rel <= 2)return min(area(),v.area());
        double d = p.distance(v.p);
        double hf = (r+v.r+d)/2.0;
        double ss = 2*sqrt(hf*(hf-r)*(hf-v.r)*(hf-d));
        double a1 = acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
        a1 = a1*r*r;
        double a2 = acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
        a2 = a2*v.r*v.r;
        return a1+a2-ss;
    }
    //`求圆和三角形pab的相交面积`
    //`测试：POJ3675 HDU3982 HDU2892`
    double areatriangle(Point a,Point b){
        if(sgn((p-a)^(p-b)) == 0)return 0.0;
        Point q[5];
        int len = 0;
        q[len++] = a;
        Line l(a,b);
        Point p1,p2;
        if(pointcrossline(l,q[1],q[2])==2){
            if(sgn((a-q[1])*(b-q[1]))<0)q[len++] = q[1];
            if(sgn((a-q[2])*(b-q[2]))<0)q[len++] = q[2];
        }
        q[len++] = b;
        if(len == 4 && sgn((q[0]-q[1])*(q[2]-q[1]))>0)swap(q[1],q[2]);
        double res = 0;
        for(int i = 0;i < len-1;i++){
            if(relation(q[i])==0||relation(q[i+1])==0){
                double arg = p.rad(q[i],q[i+1]);
                res += r*r*arg/2.0;
            }
            else{
                res += fabs((q[i]-p)^(q[i+1]-p))/2.0;
            }
        }
        return res;
    }
};

struct polygon{
    int n;
    Point p[maxp];
    Line l[maxp];
    void input(int _n){
        n = _n;
        for(int i = 0;i < n;i++)
            p[i].input();
    }
    void add(Point q){
        p[n++] = q;
    }
    void getline(){
        for(int i = 0;i < n;i++){
            l[i] = Line(p[i],p[(i+1)%n]);
        }
    }
    struct cmp{
        Point p;
        cmp(const Point &p0){p = p0;}
        bool operator()(const Point &aa,const Point &bb){
            Point a = aa, b = bb;
            int d = sgn((a-p)^(b-p));
            if(d == 0){
                return sgn(a.distance(p)-b.distance(p)) < 0;
            }
            return d > 0;
        }
    };
    //`进行极角排序`
    //`首先需要找到最左下角的点`
    //`需要重载号好Point的 < 操作符(min函数要用) `
    void norm(){
        Point mi = p[0];
        for(int i = 1;i < n;i++)mi = min(mi,p[i]);
        sort(p,p+n,cmp(mi));
    }
    //`得到凸包`
    //`得到的凸包里面的点编号是0$\sim$n-1的`
    //`两种凸包的方法`
    //`注意如果有影响，要特判下所有点共点，或者共线的特殊情况`
    //`测试 LightOJ1203 LightOJ1239`
    void getconvex(polygon &convex){
        sort(p,p+n);
        convex.n = n;
        for(int i = 0;i < min(n,2);i++){
            convex.p[i] = p[i];
        }
        if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;//特判
        if(n <= 2)return;
        int &top = convex.n;
        top = 1;
        for(int i = 2;i < n;i++){
            while(top && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i])) <= 0)
                top--;
            convex.p[++top] = p[i];
        }
        int temp = top;
        convex.p[++top] = p[n-2];
        for(int i = n-3;i >= 0;i--){
            while(top != temp && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i])) <=
                 0)
                top--;
            convex.p[++top] = p[i];
        }
        if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;//特判
        convex.norm();//`原来得到的是顺时针的点，排序后逆时针`
    }
    //`得到凸包的另外一种方法`
    //`测试 LightOJ1203 LightOJ1239`
    void Graham(polygon &convex){
        norm();
        int &top = convex.n;
        top = 0;
        if(n == 1){
            top = 1;
            convex.p[0] = p[0];
            return;
        }
        if(n == 2){
            top = 2;
            convex.p[0] = p[0];
            convex.p[1] = p[1];
            if(convex.p[0] == convex.p[1])top--;
            return;
        }
        convex.p[0] = p[0];
        convex.p[1] = p[1];
        top = 2;
        for(int i = 2;i < n;i++){
            while( top > 1 && sgn((convex.p[top-1]-convex.p[top-2])^(p[i]-convex.p[top
                 -2])) <= 0 )
                top--;
            convex.p[top++] = p[i];
        }
        if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;//特判
    }
    //`判断是不是凸的`
    bool isconvex(){
        bool s[3];
        memset(s,false,sizeof(s));
        for(int i = 0;i < n;i++){
            int j = (i+1)%n;
            int k = (j+1)%n;
            s[sgn((p[j]-p[i])^(p[k]-p[i]))+1] = true;
            if(s[0] && s[2])return false;
        }
        return true;
    }
    //`判断点和任意多边形的关系`
    //` 3 点上`
    //` 2 边上`
    //` 1 内部`
    //` 0 外部`
    int relationpoint(Point q){
        for(int i = 0;i < n;i++){
            if(p[i] == q)return 3;
        }
        getline();
        for(int i = 0;i < n;i++){
            if(l[i].pointonseg(q))return 2;
        }
        int cnt = 0;
        for(int i = 0;i < n;i++){
            int j = (i+1)%n;
            int k = sgn((q-p[j])^(p[i]-p[j]));
            int u = sgn(p[i].y-q.y);
            int v = sgn(p[j].y-q.y);
            if(k > 0 && u < 0 && v >= 0)cnt++;
            if(k < 0 && v < 0 && u >= 0)cnt--;
        }
        return cnt != 0;
    }
    //`直线u切割凸多边形左侧`
    //`注意直线方向`
    //`测试: HDU3982`
    void convexcut(Line u,polygon &po){
        int &top = po.n;//注意引用
        top = 0;
        for(int i = 0;i < n;i++){
            int d1 = sgn((u.e-u.s)^(p[i]-u.s));
            int d2 = sgn((u.e-u.s)^(p[(i+1)%n]-u.s));
            if(d1 >= 0)po.p[top++] = p[i];
            if(d1*d2 < 0)po.p[top++] = u.crosspoint(Line(p[i],p[(i+1)%n]));
        }
    }
    //`得到周长`
    //`测试 LightOJ1239`
    double getcircumference(){
        double sum = 0;
        for(int i = 0;i < n;i++){
            sum += p[i].distance(p[(i+1)%n]);
        }
        return sum;
    }
    //`得到面积`
    double getarea(){
        double sum = 0;
        for(int i = 0;i < n;i++){
            sum += (p[i]^p[(i+1)%n]);
        }
        return fabs(sum)/2;
    }
    //`得到方向`
    //` 1 表示逆时针，0表示顺时针`
    bool getdir(){
        double sum = 0;
        for(int i = 0;i < n;i++)
            sum += (p[i]^p[(i+1)%n]);
        if(sgn(sum) > 0)return 1;
        return 0;
    }
    //`得到重心`
    Point getbarycentre(){
        Point ret(0,0);
        double area = 0;
        for(int i = 1;i < n-1;i++){
            double tmp = (p[i]-p[0])^(p[i+1]-p[0]);
            if(sgn(tmp) == 0)continue;
            area += tmp;
            ret.x += (p[0].x+p[i].x+p[i+1].x)/3*tmp;
            ret.y += (p[0].y+p[i].y+p[i+1].y)/3*tmp;
        }
        if(sgn(area)) ret = ret/area;
        return ret;
    }
    //`多边形和圆交的面积`
    //`测试：POJ3675 HDU3982 HDU2892`
    double areacircle(circle c){
        double ans = 0;
        for(int i = 0;i < n;i++){
            int j = (i+1)%n;
            if(sgn( (p[j]-c.p)^(p[i]-c.p) ) >= 0)
                ans += c.areatriangle(p[i],p[j]);
            else ans -= c.areatriangle(p[i],p[j]);
        }
        return fabs(ans);
    }
    //`多边形和圆关系`
    //` 2 圆完全在多边形内`
    //` 1 圆在多边形里面，碰到了多边形边界`
    //` 0 其它`
    int relationcircle(circle c){
        getline();
        int x = 2;
        if(relationpoint(c.p) != 1)return 0;//圆心不在内部
        for(int i = 0;i < n;i++){
            if(c.relationseg(l[i])==2)return 0;
            if(c.relationseg(l[i])==1)x = 1;
        }
        return x;
    }
};
//`AB X AC`
double cross(Point A,Point B,Point C){
    return (B-A)^(C-A);
}
//`AB*AC`
double dot(Point A,Point B,Point C){
    return (B-A)*(C-A);
}
//`最小矩形面积覆盖`
//` A 必须是凸包(而且是逆时针顺序)`
//` 测试 UVA 10173`
double minRectangleCover(polygon A){
    //`要特判A.n < 3的情况`
    if(A.n < 3)return 0.0;
    A.p[A.n] = A.p[0];
    double ans = -1;
    int r = 1, p = 1, q;
    for(int i = 0;i < A.n;i++){
        //`卡出离边A.p[i] - A.p[i+1]最远的点`
        while( sgn( cross(A.p[i],A.p[i+1],A.p[r+1]) - cross(A.p[i],A.p[i+1],A.p[r]) )
             >= 0 )
            r = (r+1)%A.n;
        //`卡出A.p[i] - A.p[i+1]方向上正向n最远的点`
        while(sgn( dot(A.p[i],A.p[i+1],A.p[p+1]) - dot(A.p[i],A.p[i+1],A.p[p]) ) >= 0
             )
            p = (p+1)%A.n;
        if(i == 0)q = p;
        //`卡出A.p[i] - A.p[i+1]方向上负向最远的点`
        while(sgn(dot(A.p[i],A.p[i+1],A.p[q+1]) - dot(A.p[i],A.p[i+1],A.p[q])) <= 0)
            q = (q+1)%A.n;
        double d = (A.p[i] - A.p[i+1]).len2();
        double tmp = cross(A.p[i],A.p[i+1],A.p[r]) *
            (dot(A.p[i],A.p[i+1],A.p[p]) - dot(A.p[i],A.p[i+1],A.p[q]))/d;
        if(ans < 0 || ans > tmp)ans = tmp;
    }
    return ans;
}

//`直线切凸多边形`
//`多边形是逆时针的，在q1q2的左侧`
//`测试:HDU3982`
vector<Point> convexCut(const vector<Point> &ps,Point q1,Point q2){
    vector<Point>qs;
    int n = ps.size();
    for(int i = 0;i < n;i++){
        Point p1 = ps[i], p2 = ps[(i+1)%n];
        int d1 = sgn((q2-q1)^(p1-q1)), d2 = sgn((q2-q1)^(p2-q1));
        if(d1 >= 0)
            qs.push_back(p1);
        if(d1 * d2 < 0)
            qs.push_back(Line(p1,p2).crosspoint(Line(q1,q2)));
    }
    return qs;
}
//`半平面交`
//`测试 POJ3335 POJ1474 POJ1279`
//***************************
struct halfplane:public Line{
    double angle;
    halfplane(){}
    //`表示向量s->e逆时针(左侧)的半平面`
    halfplane(Point _s,Point _e){
        s = _s;
        e = _e;
    }
    halfplane(Line v){
        s = v.s;
        e = v.e;
    }
    void calcangle(){
        angle = atan2(e.y-s.y,e.x-s.x);
    }
    bool operator <(const halfplane &b)const{
        return angle < b.angle;
    }
};
struct halfplanes{
    int n;
    halfplane hp[maxp];
    Point p[maxp];
    int que[maxp];
    int st,ed;
    void push(halfplane tmp){
        hp[n++] = tmp;
    }
    //去重
    void unique(){
        int m = 1;
        for(int i = 1;i < n;i++){
            if(sgn(hp[i].angle-hp[i-1].angle) != 0)
                hp[m++] = hp[i];
            else if(sgn( (hp[m-1].e-hp[m-1].s)^(hp[i].s-hp[m-1].s) ) > 0)
                hp[m-1] = hp[i];
        }
        n = m;
    }
    bool halfplaneinsert(){
        for(int i = 0;i < n;i++)hp[i].calcangle();
        sort(hp,hp+n);
        unique();
        que[st=0] = 0;
        que[ed=1] = 1;
        p[1] = hp[0].crosspoint(hp[1]);
        for(int i = 2;i < n;i++){
            while(st<ed && sgn((hp[i].e-hp[i].s)^(p[ed]-hp[i].s))<0)ed--;
            while(st<ed && sgn((hp[i].e-hp[i].s)^(p[st+1]-hp[i].s))<0)st++;
            que[++ed] = i;
            if(hp[i].parallel(hp[que[ed-1]]))return false;
            p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
        }
        while(st<ed && sgn((hp[que[st]].e-hp[que[st]].s)^(p[ed]-hp[que[st]].s))<0)ed
            --;
        while(st<ed && sgn((hp[que[ed]].e-hp[que[ed]].s)^(p[st+1]-hp[que[ed]].s))<0)
            st++;
        if(st+1>=ed)return false;
        return true;
    }
    //`得到最后半平面交得到的凸多边形`
    //`需要先调用halfplaneinsert() 且返回true`
    void getconvex(polygon &con){
        p[st] = hp[que[st]].crosspoint(hp[que[ed]]);
        con.n = ed-st+1;
        for(int j = st,i = 0;j <= ed;i++,j++)
            con.p[i] = p[j];
    }
};
//**************************

const int maxn = 1010;
struct circles{
    circle c[maxn];
    double ans[maxn];//`ans[i]表示被覆盖了i次的面积`
    double pre[maxn];
    int n;
    circles(){}
    void add(circle cc){
        c[n++] = cc;
    }
    //`x包含在y中`
    bool inner(circle x,circle y){
        if(x.relationcircle(y) != 1)return 0;
        return sgn(x.r-y.r)<=0?1:0;
    }
    //圆的面积并去掉内含的圆
    void init_or(){
        bool mark[maxn] = {0};
        int i,j,k=0;
        for(i = 0;i < n;i++){
            for(j = 0;j < n;j++)
            if(i != j && !mark[j]){
               if( (c[i]==c[j])||inner(c[i],c[j]) )break;
            }
         if(j < n)mark[i] = 1;
      }
      for(i = 0;i < n;i++)
         if(!mark[i])
            c[k++] = c[i];
      n = k;
   }
   //`圆的面积交去掉内含的圆`
   void init_add(){
      int i,j,k;
      bool mark[maxn] = {0};
      for(i = 0;i < n;i++){
         for(j = 0;j < n;j++)
            if(i != j && !mark[j]){
               if( (c[i]==c[j])||inner(c[j],c[i]) )break;
            }
         if(j < n)mark[i] = 1;
      }
      for(i = 0;i < n;i++)
         if(!mark[i])
            c[k++] = c[i];
      n = k;
   }
   //`半径为r的圆，弧度为th对应的弓形的面积`
   double areaarc(double th,double r){
      return 0.5*r*r*(th-sin(th));
   }
   //`测试SPOJVCIRCLES SPOJCIRUT`
   //`SPOJVCIRCLES求n个圆并的面积，需要加上init\_or()去掉重复圆（否则WA）`
   //`SPOJCIRUT 是求被覆盖k次的面积，不能加init\_or()`
   //`对于求覆盖多少次面积的问题，不能解决相同圆，而且不能init\_or()`
   //`求多圆面积并，需要init\_or,其中一个目的就是去掉相同圆`
   void getarea(){
      memset(ans,0,sizeof(ans));
      vector<pair<double,int> >v;
      for(int i = 0;i < n;i++){
         v.clear();
         v.push_back(make_pair(-pi,1));
         v.push_back(make_pair(pi,-1));
         for(int j = 0;j < n;j++)
            if(i != j){
               Point q = (c[j].p - c[i].p);
               double ab = q.len(),ac = c[i].r, bc = c[j].r;
               if(sgn(ab+ac-bc)<=0){
                  v.push_back(make_pair(-pi,1));
                  v.push_back(make_pair(pi,-1));
                  continue;
               }
               if(sgn(ab+bc-ac)<=0)continue;
               if(sgn(ab-ac-bc)>0)continue;
```

```
                double th = atan2(q.y,q.x), fai = acos((ac*ac+ab*ab-bc*bc)/(2.0*ac*
                    ab));
                double a0 = th-fai;
                if(sgn(a0+pi)<0)a0+=2*pi;
                double a1 = th+fai;
                if(sgn(a1-pi)>0)a1-=2*pi;
                if(sgn(a0-a1)>0){
                    v.push_back(make_pair(a0,1));
                    v.push_back(make_pair(pi,-1));
                    v.push_back(make_pair(-pi,1));
                    v.push_back(make_pair(a1,-1));
                }
                else{
                    v.push_back(make_pair(a0,1));
                    v.push_back(make_pair(a1,-1));
                }
            }
            sort(v.begin(),v.end());
            int cur = 0;
            for(int j = 0;j < v.size();j++){
                if(cur && sgn(v[j].first-pre[cur])){
                    ans[cur] += areaarc(v[j].first-pre[cur],c[i].r);
                    ans[cur] += 0.5*(Point(c[i].p.x+c[i].r*cos(pre[cur]),c[i].p.y+c[i].
                        r*sin(pre[cur]))^Point(c[i].p.x+c[i].r*cos(v[j].first),c[i].p.y
                        +c[i].r*sin(v[j].first)));
                }
                cur += v[j].second;
                pre[cur] = v[j].first;
            }
        }
        for(int i = 1;i < n;i++)
            ans[i] -= ans[i+1];
    }
};
```

7.2 三维几何

```
const double eps = 1e-8;
int sgn(double x){
    if(fabs(x) < eps)return 0;
    if(x < 0)return -1;
    else return 1;
}
struct Point3{
    double x,y,z;
    Point3(double _x = 0,double _y = 0,double _z = 0){
        x = _x;
        y = _y;
        z = _z;
    }
    void input(){
        scanf("%lf%lf%lf",&x,&y,&z);
    }
    void output(){
        printf("%.2lf %.2lf %.2lf\n",x,y,z);
    }
    bool operator ==(const Point3 &b)const{
        return sgn(x-b.x) == 0 && sgn(y-b.y) == 0 && sgn(z-b.z) == 0;
    }
    bool operator <(const Point3 &b)const{
        return sgn(x-b.x)==0?(sgn(y-b.y)==0?sgn(z-b.z)<0:y<b.y):x<b.x;
    }
    double len(){
        return sqrt(x*x+y*y+z*z);
    }
    double len2(){
        return x*x+y*y+z*z;
    }
    double distance(const Point3 &b)const{
        return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y)+(z-b.z)*(z-b.z));
    }
    Point3 operator -(const Point3 &b)const{
        return Point3(x-b.x,y-b.y,z-b.z);
    }
    Point3 operator +(const Point3 &b)const{
        return Point3(x+b.x,y+b.y,z+b.z);
    }
    Point3 operator *(const double &k)const{
        return Point3(x*k,y*k,z*k);
    }
    Point3 operator /(const double &k)const{
        return Point3(x/k,y/k,z/k);
    }
    //点乘
    double operator *(const Point3 &b)const{
        return x*b.x+y*b.y+z*b.z;
    }
    //叉乘
    Point3 operator ^(const Point3 &b)const{
        return Point3(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
    }
    double rad(Point3 a,Point3 b){
        Point3 p = (*this);
        return acos( ( (a-p)*(b-p) )/ (a.distance(p)*b.distance(p)) );
    }
    //变换长度
    Point3 trunc(double r){
        double l = len();
        if(!sgn(l))return *this;
        r /= l;
        return Point3(x*r,y*r,z*r);
    }
};
struct Line3
{
    Point3 s,e;
    Line3(){}
    Line3(Point3 _s,Point3 _e)
    {
        s = _s;
        e = _e;
    }
    bool operator ==(const Line3 v)
    {
        return (s==v.s)&&(e==v.e);
    }
    void input()
    {
        s.input();
        e.input();
    }
    double length()
    {
        return s.distance(e);
    }
    //点到直线距离
    double dispointtoline(Point3 p)
    {
        return ((e-s)^(p-s)).len()/s.distance(e);
    }
    //点到线段距离
    double dispointtoseg(Point3 p)
    {
        if(sgn((p-s)*(e-s)) < 0 || sgn((p-e)*(s-e)) < 0)
            return min(p.distance(s),e.distance(p));
        return dispointtoline(p);
    }
    //`返回点p在直线上的投影`
    Point3 lineprog(Point3 p)
    {
        return s + ( ((e-s)*((e-s)*(p-s)))/((e-s).len2()) );
    }
    //`p绕此向量逆时针arg角度`
    Point3 rotate(Point3 p,double ang)
    {
        if(sgn(((s-p)^(e-p)).len()) == 0)return p;
        Point3 f1 = (e-s)^(p-s);
        Point3 f2 = (e-s)^(f1);
        double len = ((s-p)^(e-p)).len()/s.distance(e);
        f1 = f1.trunc(len); f2 = f2.trunc(len);
        Point3 h = p+f2;
        Point3 pp = h+f1;
        return h + ((p-h)*cos(ang)) + ((pp-h)*sin(ang));
    }
    //`点在直线上`
    bool pointonseg(Point3 p)
    {
        return sgn( ((s-p)^(e-p)).len() ) == 0 && sgn((s-p)*(e-p)) == 0;
    }
};
struct Plane
{
    Point3 a,b,c,o;//`平面上的三个点，以及法向量`
    Plane(){}
    Plane(Point3 _a,Point3 _b,Point3 _c)
    {
        a = _a;
        b = _b;
        c = _c;
        o = pvec();
    }
    Point3 pvec()
    {
        return (b-a)^(c-a);
    }
    //`ax+by+cz+d = 0`
    Plane(double _a,double _b,double _c,double _d)
    {
        o = Point3(_a,_b,_c);
        if(sgn(_a) != 0)
            a = Point3((-_d-_c-_b)/_a,1,1);
        else if(sgn(_b) != 0)
            a = Point3(1,(-_d-_c-_a)/_b,1);
        else if(sgn(_c) != 0)
            a = Point3(1,1,(-_d-_a-_b)/_c);
    }
    //`点在平面上的判断`
    bool pointonplane(Point3 p)
    {
        return sgn((p-a)*o) == 0;
    }
    //`两平面夹角`
    double angleplane(Plane f)
    {
        return acos((o*f.o)/(o.len()*f.o.len()));
    }
    //`平面和直线的交点，返回值是交点个数`
    int crossline(Line3 u,Point3 &p)
    {
        double x = o*(u.e-a);
        double y = o*(u.s-a);
        double d = x-y;
        if(sgn(d) == 0)return 0;
        p = ((u.s*x)-(u.e*y))/d;
        return 1;
    }
    //`点到平面最近点(也就是投影)`
    Point3 pointtoplane(Point3 p)
    {
        Line3 u = Line3(p,p+o);
        crossline(u,p);
        return p;
    }
    //`平面和平面的交线`
    int crossplane(Plane f,Line3 &u)
    {
        Point3 oo = o^f.o;
        Point3 v = o^oo;
        double d = fabs(f.o*v);
        if(sgn(d) == 0)return 0;
        Point3 q = a + (v*(f.o*(f.a-a))/d);
        u = Line3(q,q+oo);
        return 1;
    }
};
```

7.3 三维凸包

```
const double eps = 1e-8;
const int MAXN = 550;
int sgn(double x){
    if(fabs(x) < eps)return 0;
    if(x < 0)return -1;
    else return 1;
}
struct Point3{
    double x,y,z;
    Point3(double _x = 0, double _y = 0, double _z = 0){
        x = _x;
        y = _y;
        z = _z;
    }
    void input(){
        scanf("%lf%lf%lf",&x,&y,&z);
    }
    bool operator ==(const Point3 &b)const{
        return sgn(x-b.x) == 0 && sgn(y-b.y) == 0 && sgn(z-b.z) == 0;
    }
    double len(){
        return sqrt(x*x+y*y+z*z);
    }
    double len2(){
        return x*x+y*y+z*z;
    }
    double distance(const Point3 &b)const{
        return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y)+(z-b.z)*(z-b.z));
    }
    Point3 operator -(const Point3 &b)const{
        return Point3(x-b.x,y-b.y,z-b.z);
    }
    Point3 operator +(const Point3 &b)const{
        return Point3(x+b.x,y+b.y,z+b.z);
    }
    Point3 operator *(const double &k)const{
        return Point3(x*k,y*k,z*k);
    }
    Point3 operator /(const double &k)const{
        return Point3(x/k,y/k,z/k);
    }
    //点乘
    double operator *(const Point3 &b)const{
        return x*b.x + y*b.y + z*b.z;
    }
    //叉乘
    Point3 operator ^(const Point3 &b)const{
        return Point3(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
    }
};
struct CH3D{
    struct face{
        //表示凸包一个面上的三个点的编号
        int a,b,c;
        //表示该面是否属于最终的凸包上的面
        bool ok;
    };
    //初始顶点数
    int n;
    Point3 P[MAXN];
    //凸包表面的三角形数
    int num;
    //凸包表面的三角形
    face F[8*MAXN];
    int g[MAXN][MAXN];
    //叉乘
    Point3 cross(const Point3 &a,const Point3 &b,const Point3 &c){
        return (b-a)^(c-a);
    }
    //`三角形面积*2`
    double area(Point3 a,Point3 b,Point3 c){
        return ((b-a)^(c-a)).len();
    }
    //`四面体有向面积*6`
    double volume(Point3 a,Point3 b,Point3 c,Point3 d){
        return ((b-a)^(c-a))*(d-a);
    }
    //`正：点在面同向`
    double dblcmp(Point3 &p,face &f){
        Point3 p1 = P[f.b] - P[f.a];
        Point3 p2 = P[f.c] - P[f.a];
        Point3 p3 = p - P[f.a];
        return (p1^p2)*p3;
    }
    void deal(int p,int a,int b){
        int f = g[a][b];
        face add;
        if(F[f].ok){
            if(dblcmp(P[p],F[f]) > eps)
                dfs(p,f);
            else {
                add.a = b;
                add.b = a;
                add.c = p;
                add.ok = true;
                g[p][b] = g[a][p] = g[b][a] = num;
                F[num++] = add;
            }
        }
    }
    //递归搜索所有应该从凸包内删除的面
    void dfs(int p,int now){
        F[now].ok = false;
        deal(p,F[now].b,F[now].a);
        deal(p,F[now].c,F[now].b);
        deal(p,F[now].a,F[now].c);
    }
    bool same(int s,int t){
        Point3 &a = P[F[s].a];
        Point3 &b = P[F[s].b];
        Point3 &c = P[F[s].c];
        return fabs(volume(a,b,c,P[F[t].a])) < eps &&
            fabs(volume(a,b,c,P[F[t].b])) < eps &&
            fabs(volume(a,b,c,P[F[t].c])) < eps;
    }
    //构建三维凸包
    void create(){
        num = 0;
        face add;

        //***********************************
        //此段是为了保证前四个点不共面
        bool flag = true;
        for(int i = 1;i < n;i++){
            if(!(P[0] == P[i])){
                swap(P[1],P[i]);
                flag = false;
                break;
            }
        }
        if(flag)return;
        flag = true;
        for(int i = 2;i < n;i++){
            if( ((P[1]-P[0])^(P[i]-P[0])).len() > eps ){
                swap(P[2],P[i]);
                flag = false;
                break;
            }
        }
        if(flag)return;
        flag = true;
        for(int i = 3;i < n;i++){
            if(fabs( ((P[1]-P[0])^(P[2]-P[0]))*(P[i]-P[0]) ) > eps){
                swap(P[3],P[i]);
                flag = false;
                break;
            }
        }
        if(flag)return;
        //**********************************

        for(int i = 0;i < 4;i++){
            add.a = (i+1)%4;
            add.b = (i+2)%4;
            add.c = (i+3)%4;
            add.ok = true;
            if(dblcmp(P[i],add) > 0)swap(add.b,add.c);
            g[add.a][add.b] = g[add.b][add.c] = g[add.c][add.a] = num;
            F[num++] = add;
        }
        for(int i = 4;i < n;i++)
            for(int j = 0;j < num;j++)
                if(F[j].ok && dblcmp(P[i],F[j]) > eps){
                    dfs(i,j);
                    break;
                }
        int tmp = num;
        num = 0;
        for(int i = 0;i < tmp;i++)
            if(F[i].ok)
                F[num++] = F[i];
    }
    //表面积
    //`测试：HDU3528`
    double area(){
        double res = 0;
        if(n == 3){
            Point3 p = cross(P[0],P[1],P[2]);
            return p.len()/2;
        }
        for(int i = 0;i < num;i++)
            res += area(P[F[i].a],P[F[i].b],P[F[i].c]);
        return res/2.0;
    }
    double volume(){
        double res = 0;
        Point3 tmp = Point3(0,0,0);
        for(int i = 0;i < num;i++)
            res += volume(tmp,P[F[i].a],P[F[i].b],P[F[i].c]);
        return fabs(res/6);
    }
    //表面三角形个数
    int triangle(){
        return num;
    }
    //表面多边形个数
    //`测试：HDU3662`
    int polygon(){
        int res = 0;
        for(int i = 0;i < num;i++){
            bool flag = true;
            for(int j = 0;j < i;j++)
                if(same(i,j)){
                    flag = 0;
                    break;
                }
            res += flag;
        }
        return res;
    }
    //重心
    //`测试：HDU4273`
    Point3 barycenter(){
        Point3 ans = Point3(0,0,0);
        Point3 o = Point3(0,0,0);
        double all = 0;
        for(int i = 0;i < num;i++){
            double vol = volume(o,P[F[i].a],P[F[i].b],P[F[i].c]);
            ans = ans + (((o+P[F[i].a]+P[F[i].b]+P[F[i].c])/4.0)*vol);
            all += vol;
        }
        ans = ans/all;
        return ans;
    }
    //点到面的距离
    //`测试：HDU4273`
    double ptoface(Point3 p,int i){
        double tmp1 = fabs(volume(P[F[i].a],P[F[i].b],P[F[i].c],p));
        double tmp2 = ((P[F[i].b]-P[F[i].a])^(P[F[i].c]-P[F[i].a])).len();
        return tmp1/tmp2;
    }
};
CH3D hull;
int main()
{
    while(scanf("%d",&hull.n) == 1){
        for(int i = 0;i < hull.n;i++)hull.P[i].input();
        hull.create();
        Point3 p = hull.barycenter();
        double ans = 1e20;
        for(int i = 0;i < hull.num;i++)
            ans = min(ans,hull.ptoface(p,i));
        printf("%.3lf\n",ans);
    }
    return 0;
}
```

7.4 多边形交

```
/*
   * 多边形的交，多边形的边一定是要按逆时针方向给出
   * 还要判断是凸包还是凹包，调用相应的函数
   * 面积并，只要和面积减去交即可
   */
#include <bits/stdc++.h>
using namespace std;
const int maxn = //300;
const double eps = 1e-8;
int dcmp(double x)
{
   if(x > eps) return 1;
   return x < -eps ? -1 : 0;
}
struct Point
{
   double x, y;
};
double cross(Point a,Point b,Point c) ///叉积
{
   return (a.x-c.x)*(b.y-c.y)-(b.x-c.x)*(a.y-c.y);
}
Point intersection(Point a,Point b,Point c,Point d)
{
   Point p = a;
   double t =((a.x-c.x)*(c.y-d.y)-(a.y-c.y)*(c.x-d.x))/((a.x-b.x)*(c.y-d.y)-(a.y-b.
       y)*(c.x-d.x));
   p.x +=(b.x-a.x)*t;
   p.y +=(b.y-a.y)*t;
   return p;
}
//计算多边形面积
double PolygonArea(Point p[], int n)
{
   if(n < 3) return 0.0;
   double s = p[0].y * (p[n - 1].x - p[1].x);
   p[n] = p[0];
   for(int i = 1; i < n; ++ i)
      s += p[i].y * (p[i - 1].x - p[i + 1].x);
   return fabs(s * 0.5);
}
double CPIA(Point a[], Point b[], int na, int nb)//ConvexPolygonIntersectArea
{
   Point p[20], tmp[20];
   int tn, sflag, eflag;
   a[na] = a[0], b[nb] = b[0];
   memcpy(p,b,sizeof(Point)*(nb + 1));
   for(int i = 0; i < na && nb > 2; i++)
   {
      sflag = dcmp(cross(a[i + 1], p[0],a[i]));
      for(int j = tn = 0; j < nb; j++, sflag = eflag)
```

```
        {
            if(sflag>=0) tmp[tn++] = p[j];
            eflag = dcmp(cross(a[i + 1], p[j + 1],a[i]));
            if((sflag ^ eflag) == -2)
                tmp[tn++] = intersection(a[i], a[i + 1], p[j], p[j + 1]); ///求交点
        }
        memcpy(p, tmp, sizeof(Point) * tn);
        nb = tn, p[nb] = p[0];
    }
    if(nb < 3) return 0.0;
    return PolygonArea(p, nb);
}
double SPIA(Point a[], Point b[], int na, int nb)///SimplePolygonIntersectArea 调用
    此函数
{
    int i, j;
    Point t1[4], t2[4];
    double res = 0, num1, num2;
    a[na] = t1[0] = a[0], b[nb] = t2[0] = b[0];
    for(i = 2; i < na; i++)
    {
        t1[1] = a[i-1], t1[2] = a[i];
        num1 = dcmp(cross(t1[1], t1[2],t1[0]));
        if(num1 < 0) swap(t1[1], t1[2]);
        for(j = 2; j < nb; j++)
        {
            t2[1] = b[j - 1], t2[2] = b[j];
            num2 = dcmp(cross(t2[1], t2[2],t2[0]));
            if(num2 < 0) swap(t2[1], t2[2]);
            res += CPIA(t1, t2, 3, 3) * num1 * num2;
        }
    }
    return res;
}
Point p1[maxn], p2[maxn];
int n1, n2;
int main()
{
    while(cin>>n1>>n2)
    {
        for(int i = 0; i < n1; i++) scanf("%lf%lf", &p1[i].x, &p1[i].y);
        for(int i = 0; i < n2; i++) scanf("%lf%lf", &p2[i].x, &p2[i].y);
        double Area = SPIA(p1, p2, n1, n2);
    }
    return 0;
}
```

8 工具

8.1 对拍

```cpp
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int s,t,s0,cnt=0;
    while(1)
    {
        system("cls");
        do
        {
            system("data.exe > data.txt"); //data是数据生成程序，自己写
            s=clock();
            system("TmpCode0.exe < data.txt > try0.out"); //a是要测试的程序
            t=clock();
            system("TmpCode1.exe < data.txt > try1.out"); //b是正确的程序
            s0=clock();
            printf("Test:%d\n",++cnt);
            if(system("fc try0.out try1.out > nul"))break;
            else printf("AC time0: %ldms time1:%ldms\n",t-s,s0-t); //并输出运行时间
        }while(1);
        printf("WA time0: %ldms time1:%ldms\n",t-s,s0-t); //运行时间 wa了会停止对拍，wa
            的那一组会在data.txt里，所以说为了方便查找一次不要生成太大量的数据
        system("fc try0.out try1.out");
        system("pause>nul");
    }
    return 0;
}
```

8.2 数据生成器

```cpp
#include<bits/stdc++.h>
using namespace std;
mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());//Ë??ú?ý
long long rd(long long l,long long r)
{
    unsigned long long ans=rng();
    return (ans%(r-l+1))+l;
}
int main()
{
    int T=1;
    //printf("%d\n",T);
    while(T--)
    {
        int n=rd(1,10000);
        printf("%d\n",n);
    }
    return 0;
}
```

8.3 cb 环境配置

```
settings-->environment-->generalsettings-->Terminal to launch console programs:
把方框里默认的终端改成 gnome-terminal -t $TITLE -x

cb闪退:setting-editor-symbolbrowser 关闭
```