ACM-ICPC 模板整理

HDU_Coach

2019 年 9 月 17 日

目录

1 动态规划

1.1 基于位运算的最长公共子序列

输入两个串 S 和 T，长度分别为 n_1 和 n_2，压 B 位，$ap[i][j]$ 表示字符 i 在 S 串的第 j 位是否存在，$\sum_{k=0}^{j} row[i][k]$ 表示 T 串前 i 位与 S 串前 j 位的 LCS。时间复杂度 $O(\frac{n_1 n_2}{B})$。

```
#include<cstdio>
typedef long long ll;
const int BIT=808,E=62;
int n1,n2,m,h,i,j,p,ans;char s[50000],t[50000];
struct Num{
  ll x[BIT];
  Num(){for(int i=0;i<BIT;i++)x[i]=0;}
  void set(int p){x[p/E]|=1LL<<(p%E);}
  Num operator&(Num b){
    Num c;
    for(int i=0;i<=m;i++)c.x[i]=x[i]&b.x[i];
    return c;
  }
  Num operator|(Num b){
    Num c;
    for(int i=0;i<=m;i++)c.x[i]=x[i]|b.x[i];
    return c;
  }
  Num operator^(Num b){
    Num c;
    for(int i=0;i<=m;i++)c.x[i]=x[i]^b.x[i];
    return c;
  }
  Num operator-(Num b){
    Num c;
    for(int i=0;i<=m;i++)c.x[i]=x[i]-b.x[i];
    for(int i=0;i<m;i++)if(c.x[i]<0)c.x[i]+=(1LL<<E),c.x[i+1]--;
    return c;
  }
  void shl(){
    ll o=1,p;
    for(int i=0;i<=m;o=p,i++){
      p=x[i]&(1LL<<h),(x[i]<<=1)&=~(1LL<<(h+1));
      if(o)x[i]|=1;
    }
  }
}ap[4],x,row[2];
int hash(int x){
  if(x==’A’)return 0;
  if(x==’C’)return 1;
  if(x==’G’)return 2;
  return 3;
}
int main(){
  scanf(”%d%d%s%s”,&n1,&n2,s,t);i=0;
  for(m=(n1-1)/E,h=(m?E:n1)-1;i<n1;i++)ap[hash(s[i])].set(i);
  for(i=0;i<n2;i++){
    p^=1;
    x=ap[hash(t[i])]|row[p^1];
```

```
    row[p^1].shl();
    row[p]=x&((x-row[p^1])^x);
  }
  for(i=m;~i;i--)for(j=h;~j;j--)if(row[p].x[i]&(1LL<<j))ans++;
  printf("%d",ans);
}
```

1.2 决策单调且不要求在线时的糙快猛优化方法

$[l,r]$ 表示要 DP 的区间，$[dl,dr]$ 表示可用的最优决策取值区间，时间复杂度 $O(n\log n)$。

```
void dp(int l,int r,int dl,int dr){
  if(l>r)return;
  int m=(l+r)>>1,i,dm;
  for(i=dl;i<=dr;i++)if(i更新f[m]更优)dm=i;
  用dm更新f[m];
  dp(l,m-1,dl,dm),dp(m+1,r,dm,dr);
}
```

1.3 悬线法

输入 $n\times m$ 的 01 矩阵，求面积最大的全为 1 的子矩阵，时间复杂度 $O(nm)$。

```
#include<cstdio>
#define N 1001
int n,m,i,j,ans,l[N],r[N],h[N],lmax,rmax,a[N][N];
int main(){
  for(scanf("%d%d",&n,&m),i=1;i<=n;i++)for(j=1;j<=m;j++)scanf("%d",&a[i][j]);
  for(i=1;i<=m;i++)l[i]=1,r[i]=m;
  for(i=1;i<=n;i++){
    for(lmax=j=1;j<=m;j++)if(a[i][j]){
      h[j]++;
      if(lmax>l[j])l[j]=lmax;
    }else h[j]=0,l[j]=1,r[j]=m,lmax=j+1;
    for(rmax=j=m;j;j--)if(a[i][j]){
      if(rmax<r[j])r[j]=rmax;
      if((r[j]-l[j]+1)*h[j]>ans)ans=(r[j]-l[j]+1)*h[j];
    }else rmax=j-1;
  }
  printf("%d",ans);
}
```

1.4 插头 DP

以三进制表示轮廓线上插头的状态，0 表示无插头，1 表示左括号，2 表示右括号。时间复杂度 $O(nm3^m)$。

代码中点表示这个块可以通过，横表示这个块只可以左右通过，竖表示这个块只可以上下通过，井表示这个块不能通过，最后求出的是把所有可以通过的块都经过且只经过一次并回到原地的方案数。

```
#include<cstdio>
#define now f[j]
#define pre f[j-1]
typedef long long ll;
int n,m,x,y,i,j,k,h,S,e,pow[14],q[41836],p[14][41840],st[14],can;
int lasti,lastj,firsti,firstj,hash[1594324];
ll ans,f[14][41836];
char ch[14][14];
int bit(int x,int i){return x/pow[i]%3;}
void up(ll&a,ll b){a+=b;}
int main(){
  scanf("%d%d",&n,&m);
  for(i=1;i<=n;i++)for(scanf("%s",ch[i]+1),j=1;j<=m;j++)if(ch[i][j]!='#'){
    lasti=i,lastj=j;
    if(!firsti)firsti=i,firstj=j;
  }
  for(pow[0]=i=1;i<=m+1;i++)pow[i]=pow[i-1]*3;
  S=pow[m+1];
  for(i=0;i<S;i++){
    can=1;st[0]=0;
    for(j=0;j<=m;j++){
      k=bit(i,j);
      if(k==2){
        if(!st[0]){can=0;break;}
        p[st[st[0]]][q[0]+1]=j;p[j][q[0]+1]=st[st[0]];
        --st[0];
      }
      if(k==1)st[++st[0]]=j;
    }
    if(can&&!st[0])q[hash[i]=++q[0]]=i;
  }
  for(i=1;i<=n;i++){
    for(k=0;k<=q[0];k++)f[0][k]=0;
    for(k=1;k<=q[0];k++)
      if(f[m][k]&&!bit(q[k],m))
        f[0][hash[(q[k]-bit(q[k],m)*pow[m])*3]]=f[m][k];
    for(j=1;j<=m;j++)for(k=0;k<=q[0];k++)f[j][k]=0;
    for(j=1;j<=m;j++){
      if(ch[i][j]=='.'&&i==firsti&&j==firstj)up(now[hash[pow[j-1]+pow[j]*2]],1);
      for(h=1;h<=q[0];h++){
        k=q[h];
        if(!pre[h])continue;
        x=bit(k,j-1),y=bit(k,j),e=k-x*pow[j-1]-y*pow[j];
        if(!x&&!y){
          if(ch[i][j]!='-'&&ch[i][j]!='|'){
            if(ch[i][j]=='.')up(now[hash[e+pow[j-1]+2*pow[j]]],pre[h]);
            if(ch[i][j]=='#')up(now[h],pre[h]);
          }
        }else if(!x){
          if(ch[i][j]!='#'&&ch[i][j]!='-'){
            up(now[hash[e+y*pow[j-1]]],pre[h]);
            if(ch[i][j]=='.')up(now[hash[e+y*pow[j]]],pre[h]);
          }
        }else if(!y){
          if(ch[i][j]!='#'&&ch[i][j]!='|'){
            if(ch[i][j]=='.')up(now[hash[e+x*pow[j-1]]],pre[h]);
```

```
          up(now[hash[e+x*pow[j]]],pre[h]);
        }
      }else if(ch[i][j]=='.'){
        if(x==1&&y==2&&!e&&i==lasti&&j==lastj)up(ans,pre[h]);
        else if(x==2&&y==1)up(now[hash[e]],pre[h]);
        else if(x==y){
          int t=e-bit(e,p[j-1][h])*pow[p[j-1][h]]
                -bit(e,p[j][h])*pow[p[j][h]]
                +pow[p[j][h]]+2*pow[p[j-1][h]];
          up(now[hash[t]],pre[h]);
        }
      }
    }
  }
  }
  printf("%lld",ans);
}
```

1.5 整数划分

$f[i][j]$ 表示选了 i 种不同的数字，总和为 j 的方案数。

$f[i][j]=f[i-1][j-1]+f[i][j-i]$，此式子的意义为要么新选一个 1，要么之前选的都增加 1。若每种数字最多选一个，那么有 $f[i][j]=f[i-1][j-i]+f[i][j-i]$。

对于求把 n 划分成若干整数的和的方案数，设 $g[i]$ 表示 $n=i$ 时的答案，代码如下，时间复杂度 $O(n\sqrt{n})$。

```
#include<cstdio>
const int N=731,P=1000000007;
int n,m,i,j,f[N+1],g[200001];
int main(){
  for(f[1]=1,f[2]=2,f[3]=5,f[4]=7,i=5;i<N;i++)f[i]=3+2*f[i-2]-f[i-4];
  for(scanf("%d",&n),g[0]=i=1;i<=n;i++)
    for(j=1;f[j]<=i;j++)
      if((j+1)>>1&1)g[i]=(g[i]+g[i-f[j]])%P;
      else g[i]=(g[i]-g[i-f[j]])%P;
}
```

2 莫队算法

2.1 普通莫队算法

```
#include<cstdio>
#include<algorithm>
#define N 50010
using namespace std;
int n,m,lim,i,l,r,ans[N];
struct Q{int l,r,p;}q[N];
bool cmp(const Q&a,const Q&b){return pos[a.l]==pos[b.l]?a.r<b.r:pos[a.l]<pos[b.l];}
int main(){
  scanf("%d%d",&n,&m);
  while(lim*lim<n)lim++;
  for(i=1;i<=n;i++)pos[i]=(i-1)/lim+1;
  for(i=1;i<=m;i++)read(q[i].l),read(q[i].r),q[i].p=i;
  sort(q+1,q+m+1,cmp);
  for(i=l=1,r=0;i<=m;i++){
    int L=q[i].l,R=q[i].r;
    if(r<R){for(r++;r<=R;r++)add(r);r--;}
    if(r>R)for(;r>R;r--)del(r);
    if(l<L)for(;l<L;l++)del(l);
    if(l>L){for(l--;l>=L;l--)add(l);l++;}
    ans[q[i].p]=now;
  }
  for(i=1;i<=m;i++)printf("%d\n",ans[i]);
}
```

2.2 树上莫队算法

按 DFS 序分块，查询的时候需要额外考虑 lca。

```
#include<cstdio>
#include<algorithm>
#include<cmath>
#define N 100010
#define K 17
using namespace std;
struct P{int l,r,z,id;}Q[N];
int lim,pos[N<<1],l,r,c[N],g[N],v[N<<1],nxt[N<<1],ed;
int n,m,i,j,x,y,z,loc[N<<1],dfn,st[N],en[N],d[N],f[N][18];
int ans[N],cnt[N],sum;bool vis[N];
bool cmp(const P&a,const P&b){return pos[a.l]==pos[b.l]?a.r<b.r:pos[a.l]<pos[b.l];}
void add(int x,int y){v[++ed]=y;nxt[ed]=g[x];g[x]=ed;}
void dfs(int x){
  for(int i=vis[loc[st[x]=++dfn]=x]=1;i<=K;i++)f[x][i]=f[f[x][i-1]][i-1];
  for(int i=g[x];i;i=nxt[i])if(!vis[v[i]])d[v[i]]=d[f[v[i]][0]=x]+1,dfs(v[i]);
  loc[en[x]=++dfn]=x;
}
int lca(int x,int y){
  if(x==y)return x;
  if(d[x]<d[y])swap(x,y);
  for(int i=K;~i;i--)if(d[f[x][i]]>=d[y])x=f[x][i];
  if(x==y)return x;
```

```
  for(int i=K;~i;i--)if(f[x][i]!=f[y][i])x=f[x][i],y=f[y][i];
  return f[x][0];
}
void deal(int x){
  if(!vis[x]){if(!(--cnt[c[x]]))sum--;}else if(!(cnt[c[x]]++))sum++;
  vis[x]^=1;
}
int main(){
  for(read(n),read(m),i=1;i<=n;i++)read(c[i]);
  for(i=1;i<=n;i++)read(x),read(y),add(x,y),add(y,x);
  dfs(d[1]=1),lim=(int)sqrt(n*2+0.5);
  for(i=1;i<=dfn;i++)pos[i]=(i-1)/lim+1;
  for(i=1;i<=m;i++){
    read(x),read(y);Q[i].id=i;
    if(st[x]>st[y])swap(x,y);
    z=lca(x,y);
    if(z==x)Q[i].l=st[x],Q[i].r=st[y];
    else Q[i].l=en[x],Q[i].r=st[y],Q[i].z=z;
  }
  for(sort(Q+1,Q+m+1,cmp),i=1,l=1,r=0;i<=m;i++){
    if(r<Q[i].r){for(r++;r<=Q[i].r;r++)deal(loc[r]);r--;}
    if(r>Q[i].r)for(;r>Q[i].r;r--)deal(loc[r]);
    if(l<Q[i].l)for(;l<Q[i].l;l++)deal(loc[l]);
    if(l>Q[i].l){for(l--;l>=Q[i].l;l--)deal(loc[l]);l++;}
    if(Q[i].z)deal(Q[i].z);
    ans[Q[i].id]=now;
    if(Q[i].z)deal(Q[i].z);
  }
  for(i=1;i<=m;i++)printf("%d\n",ans[i]);
}
```

2.3 树上带修改莫队算法

将 DFS 序分成 $n^{\frac{1}{3}}$ 块，枚举两块，然后按时间处理操作，时间复杂度 $O(n^{\frac{5}{3}})$。

```
#include<cstdio>
#include<algorithm>
#define N 50010
#define K 15
using namespace std;
struct Que{int l,r,t,z;}ask[N];
int ans[N];
int n,m,q,i,j,k,x,y,z,f[N][16],d[N],B,nl,nr,l,r,vis[N],C[N],c[N];
int T,mx[N],my[N],op[N];
int st[N],en[N],dfn[N<<1],pos[N<<1],post;
int g[N],v[N<<1],nxt[N<<1],ed,que[50][50],fin[50][50];
int ap[N],h[N],full[N],now[N];
void add(int x,int y){v[++ed]=y;nxt[ed]=g[x];g[x]=ed;}
void dfs(int x,int pre){
  dfn[st[x]=++post]=x;
  int i=1;
  for(f[x][0]=pre;i<=K;i++)f[x][i]=f[f[x][i-1]][i-1];
  for(i=g[x];i;i=nxt[i])if(v[i]!=pre)d[v[i]]=d[x]+1,dfs(v[i],x);
  dfn[en[x]=++post]=x;
}
```

```
int lca(int x,int y){
  if(x==y)return x;
  if(d[x]<d[y])swap(x,y);
  for(int i=K;~i;i--)if(d[f[x][i]]>=d[y])x=f[x][i];
  if(x==y)return x;
  for(int i=K;~i;i--)if(f[x][i]!=f[y][i])x=f[x][i],y=f[y][i];
  return f[x][0];
}
inline void addq(int x,int y,int z){
  v[++ed]=z;nxt[ed]=0;
  if(!que[x][y])que[x][y]=ed;else nxt[fin[x][y]]=ed;
  fin[x][y]=ed;
}
void deal(int x){
  if(c[x]<=n){
    if(vis[x]){
      ap[c[x]]--;
      if(!ap[c[x]])now[pos[c[x]]]--;
    }else{
      if(!ap[c[x]])now[pos[c[x]]]++;
      ap[c[x]]++;
    }
  }
  vis[x]^=1;
}
int askmex(){
  for(int i=0;;i++)if(full[i]>now[i])
    for(int j=h[i];;j++)if(!ap[j])return j;
}
int main(){
  read(n);read(q);
  for(i=1;i<=n;i++)read(C[i]);
  for(i=1;i<n;i++)read(x),read(y),add(x,y),add(y,x);
  dfs(d[1]=1,ed=0);
  while(B*B*B<post)B++;B*=B;
  for(i=1;i<=post;i++)pos[i]=(i-1)/B+1;m=pos[post];
  for(i=1;i<=q;i++){
    read(op[i]);read(x);read(y);
    if(!op[i])mx[++T]=x,my[T]=y;
    else{
      if(st[x]>st[y])swap(x,y);
      z=lca(x,y);
      if(z==x)nl=st[x],nr=st[y];else nl=en[x],nr=st[y];
      ask[i].t=T;
      ask[i].l=nl;ask[i].r=nr;
      if(z!=x)ask[i].z=z;
      addq(pos[nl],pos[nr],i);
    }
  }
  for(B=1;B*B<=n;B++);
  for(i=1;i<=n;i++)pos[i]=(i-1)/B+1;
  for(i=0;i<=n;i++)full[pos[i]]++;
  for(i=n;i;i--)h[pos[i]]=i;
  for(i=1;i<=m;i++)for(j=i;j<=m;j++)if(que[i][j]){
    for(k=0;k<=n;k++)c[k]=C[k],vis[k]=ap[k]=0;
    for(k=0;k<=pos[n];k++)now[k]=0;
    T=1;l=(i-1)*B+1;r=l-1;
```

```
    for(k=que[i][j];k;k=nxt[k]){
      if(r<ask[v[k]].r){for(r++;r<=ask[v[k]].r;r++)deal(dfn[r]);r--;}
      if(r>ask[v[k]].r)for(;r>ask[v[k]].r;r--)deal(dfn[r]);
      if(l<ask[v[k]].l)for(;l<ask[v[k]].l;l++)deal(dfn[l]);
      if(l>ask[v[k]].l){for(l--;l>=ask[v[k]].l;l--)deal(dfn[l]);l++;}
      while(T<=ask[v[k]].t){
        bool flag=(ask[v[k]].l<=st[mx[T]]&&st[mx[T]]<=ask[v[k]].r)
                  ^(ask[v[k]].l<=en[mx[T]]&&en[mx[T]]<=ask[v[k]].r);
        if(flag)deal(mx[T]);
        c[mx[T]]=my[T];
        if(flag)deal(mx[T]);
        T++;
      }
      if(ask[v[k]].z)deal(ask[v[k]].z);
      ans[v[k]]=askmex();
      if(ask[v[k]].z)deal(ask[v[k]].z);
    }
  }
  for(i=1;i<=q;i++)if(op[i])printf("%d\n",ans[i]);
}
```

2.4 二维莫队算法

二维莫队算法，将矩阵横着分成 $\sqrt{n}$ 份，竖着分成 $\sqrt{m}$ 份，一共得到 $\sqrt{nm}$ 块，从左往右，从上到下编号。对于询问，以左上角所在块为第一关键字，右下角所在块为第二关键字排序，然后暴力转移。时间复杂度 $O(qn^{\frac{3}{2}})$。

```
#include<cstdio>
#include<algorithm>
using namespace std;
const int N=210,M=100010;
int n,m,Q,sn,sm,i,j,a[N][N],pos[N][N];
int X0,X1,Y0,Y1,l0,r0,l1,r1,now,ans[M];
struct P{int a,b,c,d,p;}q[M];
bool cmp(const P&a,const P&b){
  return pos[a.a][a.c]==pos[b.a][b.c]?
         pos[a.b][a.d]<pos[b.b][b.d]:pos[a.a][a.c]<pos[b.a][b.c];
}
int main(){
  scanf("%d%d",&n,&m);
  for(i=1;i<=n;i++)for(j=1;j<=m;j++)scanf("%d",&a[i][j]);
  while(sn*sn<n)sn++;
  while(sm*sm<m)sm++;
  for(i=1;i<=n;i++)for(j=1;j<=m;j++)pos[i][j]=i/sn*n+j/sm;
  for(scanf("%d",&Q),i=1;i<=Q;i++){
    scanf("%d%d%d%d",&X0,&Y0,&X1,&Y1);
    if(X0>X1)swap(X0,X1);
    if(Y0>Y1)swap(Y0,Y1);
    q[i].a=X0,q[i].b=X1,q[i].c=Y0,q[i].d=Y1,q[i].p=i;
  }
  for(sort(q+1,q+Q+1,cmp),i=l0=l1=1,r0=r1=0;i<=Q;i++){
    int L0=q[i].a,R0=q[i].b,L1=q[i].c,R1=q[i].d;
    if(r0<R0){for(r0++;r0<=R0;r0++)for(j=l1;j<=r1;j++)add(a[r0][j]);r0--;}
    if(r0>R0)for(;r0>R0;r0--)for(j=l1;j<=r1;j++)del(a[r0][j]);
```

```
    if(l0<L0)for(;l0<L0;l0++)for(j=l1;j<=r1;j++)del(a[l0][j]);
    if(l0>L0){for(l0--;l0>=L0;l0--)for(j=l1;j<=r1;j++)add(a[l0][j]);l0++;}
    if(r1<R1){for(r1++;r1<=R1;r1++)for(j=l0;j<=r0;j++)add(a[j][r1]);r1--;}
    if(r1>R1)for(;r1>R1;r1--)for(j=l0;j<=r0;j++)del(a[j][r1]);
    if(l1<L1)for(;l1<L1;l1++)for(j=l0;j<=r0;j++)del(a[j][l1]);
    if(l1>L1){for(l1--;l1>=L1;l1--)for(j=l0;j<=r0;j++)add(a[j][l1]);l1++;}
    ans[q[i].p]=now;
  }
  for(i=1;i<=Q;i++)printf(”%d\n”,ans[i]);
}
```

3 数据结构

3.1 Hash

3.1.1 Hash 表

```
const int M=262143;
struct E{int v;E*nxt;}*g[M+1],pool[N],*cur=pool,*p;int vis[M+1];
void ins(int v){
  int u=v&M;
  if(vis[u]<T)vis[u]=T,g[u]=NULL;
  for(p=g[u];p;p=p->nxt)if(p->v==v)return;
  p=cur++;p->v=v;p->nxt=g[u];g[u]=p;
}
int ask(int v){
  int u=v&M;
  if(vis[u]<T)return 0;
  for(p=g[u];p;p=p->nxt)if(p->v==v)return 1;
  return 0;
}
void init(){T++,cur=pool;}
```

3.1.2 字符串 Hash

```
const int N=20010,P=31,D=1000173169,M=262143;
int n,i,pow[N],f[N];char a[N];
int hash(int l,int r){return(ll)(f[r]-(ll)f[l-1]*pow[r-l+1]%D+D)%D;}
int main(){
  scanf("%d%s",&n,a+1);
  for(pow[0]=i=1;i<=n;i++)pow[i]=(ll)pow[i-1]*P%D;
  for(i=1;i<=n;i++)f[i]=(ll)((ll)f[i-1]*P+a[i])%D;
}
```

3.1.3 矩阵 Hash

找出某个 $x \times y$ 的矩阵在某个 $n \times m$ 的矩阵中的所有出现位置。首先对于每个位置，求出它开始长度为 y 的横行的 hash 值，然后对于 hash 值再求一次竖列的 hash 值即可。

```
#include<cstdio>
#include<algorithm>
#define N 1010
typedef unsigned long long ll;
const ll D1=197,D2=131;
int n,m,x,y,i,j,ans,t,cnt;
char a[N][N];
ll pow1[N],pow2[N],h[N][N],tmp;
struct P{
  int x,y;ll z;
  P(){}
  P(int _x,int _y,ll _z){x=_x,y=_y,z=_z;}
}q[N*N],fin[N];
bool cmp(const P&a,const P&b){return a.z<b.z;}
```

```
bool cmp2(const P&a,const P&b){return a.x==b.x?a.y<b.y:a.x<b.x;}
int main(){
  scanf("%d%d",&n,&m);gets(a[0]);
  for(i=1;i<=n;i++)gets(a[i]+1);
  scanf("%d%d",&x,&y);
  for(pow1[0]=pow2[0]=i=1;i<=n||i<=m;i++){pow1[i]=pow1[i-1]*D1,pow2[i]=pow2[i-1]*D2;
  for(i=1;i<=n;i++){
    for(tmp=0,j=1;j<y;j++)tmp=tmp*D1+a[i][j],h[i][j]=0;
    for(j=y;j<=m;j++)h[i][j]=tmp=tmp*D1-pow1[y]*a[i][j-y]+a[i][j];
  }
  for(t=0,i=y;i<=m;i++){
    for(tmp=0,j=1;j<x;j++)tmp=tmp*D2+h[j][i];
    for(j=x;j<=n;j++)q[++t]=P(j-x+1,i-y+1,tmp=tmp*D2-pow2[x]*h[j-x][i]+h[j][i]);
  }
  for(std::sort(q+1,q+t+1,cmp),j=1,i=2;i<=t;i++)if(q[i-1].z!=q[i].z){
    if(i-j>ans)ans=i-j,tmp=q[j].z;
    j=i;
  }
  if(t-j+1>ans)ans=t-j+1,tmp=q[t].z;
  printf("%d %d\n",x,y);
  for(i=1;i<=t;i++)if(q[i].z==tmp)fin[++cnt]=P(q[i].x,q[i].y,0);
  std::sort(fin+1,fin+cnt+1,cmp2);
  for(i=0;i<x;puts(""),i++)for(j=0;j<y;j++)putchar(a[fin[1].x+i][fin[1].y+j]);
  for(printf("%d\n",cnt),i=1;i<=cnt;i++)printf("%d %d\n",fin[i].x,fin[i].y);
}
```

3.2 树状数组区间修改区间查询

维护一个序列 $b[i]$，一开始都是 0，支持以下操作：

1. 把区间 $[x,y]$ 内的 $b[i]$ 加上 $a[i]$。

2. 查询区间 $[x,y]$ 内 $b[i]$ 的和。

代码中 s 为 a 的前缀和。

```
int n,m,i,op,x,y;
struct BIT{
  int n,s[N],a[N];ll b[N];
  void init(int x){n=x;for(int i=1;i<=n;i++)a[i]=b[i]=s[i]=0;}
  void modify(int x,int p){for(int i=x;i<=n;i+=i&-i)a[i]+=p,b[i]+=p*s[x-1];}
  ll ask(int x){
    int t0=0;ll t1=0;
    for(int i=x;i;i-=i&-i)t0+=a[i],t1+=b[i];
    return 1LL*s[x]*t0-t1;
  }
}T;
int main(){
  scanf("%d",&n);
  for(i=1;i<=n;i++)scanf("%d",&T.s[i]),T.s[i]+=T.s[i-1];
  scanf("%d",&m);
  while(m--){
    scanf("%d%d%d",&op,&x,&y);
    if(op==1)T.modify(x,1),T.modify(y+1,-1);
    else printf("%lld\n",T.ask(y)-T.ask(x-1));
  }
}
```

3.3 K-D Tree

```
#include<cstdio>
#include<algorithm>
#define N 200010
int n,i,id[N],root,cmp_d;
struct node{int d[2],l,r,Max[2],Min[2],val,sum,f;}t[N];
bool cmp(const node&a,const node&b){return a.d[cmp_d]<b.d[cmp_d];}
void umax(int&a,int b){if(a<b)a=b;}
void umin(int&a,int b){if(a>b)a=b;}
void up(int x){
  if(t[x].l){
    umax(t[x].Max[0],t[t[x].l].Max[0]);
    umin(t[x].Min[0],t[t[x].l].Min[0]);
    umax(t[x].Max[1],t[t[x].l].Max[1]);
    umin(t[x].Min[1],t[t[x].l].Min[1]);
  }
  if(t[x].r){
    umax(t[x].Max[0],t[t[x].r].Max[0]);
    umin(t[x].Min[0],t[t[x].r].Min[0]);
    umax(t[x].Max[1],t[t[x].r].Max[1]);
    umin(t[x].Min[1],t[t[x].r].Min[1]);
  }
}
int build(int l,int r,int D,int f){
  int mid=(l+r)>>1;
  cmp_d=D,std::nth_element(t+l+1,t+mid+1,t+r+1,cmp);
  id[t[mid].f]=mid;
  t[mid].f=f;
  t[mid].Max[0]=t[mid].Min[0]=t[mid].d[0];
  t[mid].Max[1]=t[mid].Min[1]=t[mid].d[1];
  t[mid].val=t[mid].sum=0;
  if(l!=mid)t[mid].l=build(l,mid-1,!D,mid);else t[mid].l=0;
  if(r!=mid)t[mid].r=build(mid+1,r,!D,mid);else t[mid].r=0;
  return up(mid),mid;
}
//输入的第x个点的权值增加p
void change(int x,int p){for(t[x=id[x]].val+=p;x;x=t[x].f)t[x].sum+=p;}
/*
估价函数：
欧几里得距离下界：
sqr(max(max(X-x.Max[0],x.Min[0]-X),0))+sqr(max(max(Y-x.Max[1],x.Min[1]-Y),0))
曼哈顿距离下界：
max(x.Min[0]-X,0)+max(X-x.Max[0],0)+max(x.Min[1]-Y,0)+max(Y-x.Max[1],0)
欧几里得距离上界：
max(sqr(X-x.Min[0]),sqr(X-x.Max[0]))+max(sqr(Y-x.Min[1]),sqr(Y-x.Max[1])
曼哈顿距离上界：
max(abs(X-x.Max[0]),abs(x.Min[0]-X))+max(abs(Y-x.Max[1]),abs(x.Min[1]-Y))
*/
//查询矩形范围内所有点的权值和
void ask(int x){
  if(t[x].Min[0]>X2||t[x].Max[0]<X1||t[x].Min[1]>Y2||t[x].Max[1]<Y1)return;
  if(t[x].Min[0]>=X1&&t[x].Max[0]<=X2&&t[x].Min[1]>=Y1&&t[x].Max[1]<=Y2){
    k+=t[x].sum;
    return;
  }
```

```
  if(t[x].d[0]>=X1&&t[x].d[0]<=X2&&t[x].d[1]>=Y1&&t[x].d[1]<=Y2)k+=t[x].val;
  if(t[x].l)ask(t[x].l);
  if(t[x].r)ask(t[x].r);
}
int main(){
  while(~scanf("%d",&n)){
    for(i=1;i<=n;i++){
      scanf("%d%d",&x,&y);
      t[i].d[0]=x,t[i].d[1]=y,t[i].f=i;
    }
    root=build(1,n,0,0);
  }
  return 0;
}
```

3.4 Link-Cut Tree

```
int f[N],son[N][2],val[N],sum[N],tmp[N];bool rev[N];
bool isroot(int x){return !f[x]||son[f[x]][0]!=x&&son[f[x]][1]!=x;}
void rev1(int x){if(!x)return;swap(son[x][0],son[x][1]);rev[x]^=1;}
void pb(int x){if(rev[x])rev1(son[x][0]),rev1(son[x][1]),rev[x]=0;}
void up(int x){
  sum[x]=val[x];
  if(son[x][0])sum[x]+=sum[son[x][0]];
  if(son[x][1])sum[x]+=sum[son[x][1]];
}
void rotate(int x){
  int y=f[x],w=son[y][1]==x;
  son[y][w]=son[x][w^1];
  if(son[x][w^1])f[son[x][w^1]]=y;
  if(f[y]){
    int z=f[y];
    if(son[z][0]==y)son[z][0]=x;else if(son[z][1]==y)son[z][1]=x;
  }
  f[x]=f[y];f[y]=x;son[x][w^1]=y;up(y);
}
void splay(int x){
  int s=1,i=x,y;tmp[1]=i;
  while(!isroot(i))tmp[++s]=i=f[i];
  while(s)pb(tmp[s--]);
  while(!isroot(x)){
    y=f[x];
    if(!isroot(y)){if((son[f[y]][0]==y)^(son[y][0]==x))rotate(x);else rotate(y);}
    rotate(x);
  }
  up(x);
}
void access(int x){for(int y=0;x;y=x,x=f[x])splay(x),son[x][1]=y,up(x);}
int root(int x){access(x);splay(x);while(son[x][0])x=son[x][0];return x;}
void makeroot(int x){access(x);splay(x);rev1(x);}
void link(int x,int y){makeroot(x);f[x]=y;access(x);}
void cutf(int x){access(x);splay(x);f[son[x][0]]=0;son[x][0]=0;up(x);}
void cut(int x,int y){makeroot(x);cutf(y);}
int ask(int x,int y){makeroot(x);access(y);splay(y);return sum[y];}
```

3.5 Top Tree

```
const int N=100010*2,inf=~0U>>1;
struct tag{
int a,b;//ax+b
tag(){a=1,b=0;}
tag(int x,int y){a=x,b=y;}
bool ex(){return a!=1||b;}
tag operator+(const tag&x){return tag(a*x.a,b*x.a+x.b);}
};
int atag(int x,tag y){return x*y.a+y.b;}
struct data{
int sum,minv,maxv,size;
data(){sum=size=0,minv=inf,maxv=-inf;}
data(int x){sum=minv=maxv=x,size=1;}
data(int a,int b,int c,int d){sum=a,minv=b,maxv=c,size=d;}
data operator+(const data&x){
  return data(sum+x.sum,min(minv,x.minv),max(maxv,x.maxv),size+x.size);
}
};
data operator+(const data&a,const tag&b){
  return a.size?data(a.sum*b.a+a.size*b.b,atag(a.minv,b),atag(a.maxv,b),a.size):a;
}
//son:0-1：重链儿子，2-3：AAA 树儿子
int f[N],son[N][4],a[N],tot,rt,rub,ru[N];bool rev[N],in[N];
int val[N];
data csum[N],tsum[N],asum[N];
tag ctag[N],ttag[N];
bool isroot(int x,int t){
  if(t)return !f[x]||!in[f[x]]||!in[x];
  return !f[x]||(son[f[x]][0]!=x&&son[f[x]][1]!=x)||in[f[x]]||in[x];
}
void rev1(int x){
  if(!x)return;
  swap(son[x][0],son[x][1]);rev[x]^=1;
}
void tagchain(int x,tag p){
  if(!x)return;
  csum[x]=csum[x]+p;
  asum[x]=csum[x]+tsum[x];
  val[x]=atag(val[x],p);
  ctag[x]=ctag[x]+p;
}
void tagtree(int x,tag p,bool t){
  if(!x)return;
  tsum[x]=tsum[x]+p;
  ttag[x]=ttag[x]+p;
  if(!in[x]&&t)tagchain(x,p);else asum[x]=csum[x]+tsum[x];
}
void pb(int x){
  if(!x)return;
  if(rev[x])rev1(son[x][0]),rev1(son[x][1]),rev[x]=0;
  if(!in[x]&&ctag[x].ex()){
    tagchain(son[x][0],ctag[x]);
    tagchain(son[x][1],ctag[x]);
    ctag[x]=tag();
```

```
  }
  if(ttag[x].ex()){
    tagtree(son[x][0],ttag[x],0),tagtree(son[x][1],ttag[x],0);
    tagtree(son[x][2],ttag[x],1),tagtree(son[x][3],ttag[x],1);
    ttag[x]=tag();
  }
}
void up(int x){
  tsum[x]=data();
  for(int i=0;i<2;i++)if(son[x][i])tsum[x]=tsum[x]+tsum[son[x][i]];
  for(int i=2;i<4;i++)if(son[x][i])tsum[x]=tsum[x]+asum[son[x][i]];
  if(in[x]){
    csum[x]=data();
    asum[x]=tsum[x];
  }else{
    csum[x]=data(val[x]);
    for(int i=0;i<2;i++)if(son[x][i])csum[x]=csum[x]+csum[son[x][i]];
    asum[x]=csum[x]+tsum[x];
  }
}
int child(int x,int t){pb(son[x][t]);return son[x][t];}
void rotate(int x,int t){
  int y=f[x],w=(son[y][t+1]==x)+t;
  son[y][w]=son[x][w^1];
  if(son[x][w^1])f[son[x][w^1]]=y;
  if(f[y])for(int z=f[y],i=0;i<4;i++)if(son[z][i]==y)son[z][i]=x;
  f[x]=f[y];f[y]=x;son[x][w^1]=y;up(y);
}
void splay(int x,int t=0){
  int s=1,i=x,y;a[1]=i;
  while(!isroot(i,t))a[++s]=i=f[i];
  while(s)pb(a[s--]);
  while(!isroot(x,t)){
    y=f[x];
    if(!isroot(y,t)){if((son[f[y]][t]==y)^(son[y][t]==x))rotate(x,t);else rotate(y,t);}
    rotate(x,t);
  }
  up(x);
}
int newnode(){
  int x=rub?ru[rub--]:++tot;
  son[x][2]=son[x][3]=0;in[x]=1;
  return x;
}
void setson(int x,int t,int y){son[x][t]=y;f[y]=x;}
int pos(int x){for(int i=0;i<4;i++)if(son[f[x]][i]==x)return i;return 4;}
void add(int x,int y){//从x连出一条虚边到y
  if(!y)return;
  pb(x);
  for(int i=2;i<4;i++)if(!son[x][i]){
    setson(x,i,y);
    return;
  }
  while(son[x][2]&&in[son[x][2]])x=child(x,2);
  int z=newnode();
  setson(z,2,son[x][2]);
  setson(z,3,y);
```

```
  setson(x,2,z);
  splay(z,2);
}
void del(int x){//将x与其虚边上的父亲断开
  if(!x)return;
  splay(x);
  if(!f[x])return;
  int y=f[x];
  if(in[y]){
    int s=1,i=y,z=f[y];a[1]=i;
    while(!isroot(i,2))a[++s]=i=f[i];
    while(s)pb(a[s--]);
    if(z){
      setson(z,pos(y),child(y,pos(x)^1));
      splay(z,2);
    }
    ru[++rub]=y;
  }else{
    son[y][pos(x)]=0;
    splay(y);
  }
  f[x]=0;
}
int fa(int x){//x通过虚边的父亲
  splay(x);
  if(!f[x])return 0;
  if(!in[f[x]])return f[x];
  int t=f[x];
  splay(t,2);
  return f[t];
}
int access(int x){
  int y=0;
  for(;x;y=x,x=fa(x)){
    splay(x);
    del(y);
    add(x,son[x][1]);
    setson(x,1,y);
    up(x);
  }
  return y;
}
int lca(int x,int y){
  access(x);
  return access(y);
}
int root(int x){
  access(x);
  splay(x);
  while(son[x][0])x=son[x][0];
  return x;
}
void makeroot(int x){
  access(x);
  splay(x);
  rev1(x);
}
```

```
void link(int x,int y){
  makeroot(x);
  add(y,x);
  access(x);
}
void cut(int x){
  access(x);
  splay(x);
  f[son[x][0]]=0;
  son[x][0]=0;
  up(x);
}
void changechain(int x,int y,tag p){
  makeroot(x);
  access(y);
  splay(y);
  tagchain(y,p);
}
data askchain(int x,int y){
  makeroot(x);
  access(y);
  splay(y);
  return csum[y];
}
void changetree(int x,tag p){
  access(x);
  splay(x);
  val[x]=atag(val[x],p);
  for(int i=2;i<4;i++)if(son[x][i])tagtree(son[x][i],p,1);
  up(x);
  splay(x);
}
data asktree(int x){
  access(x);
  splay(x);
  data t=data(val[x]);
  for(int i=2;i<4;i++)if(son[x][i])t=t+asum[son[x][i]];
  return t;
}
int n,m,x,y,z,k,i,ed[N][2];
int main(){
  read(n);read(m);
  tot=n;
  for(i=1;i<n;i++)read(ed[i][0]),read(ed[i][1]);
  for(i=1;i<=n;i++)read(val[i]),up(i);
  for(i=1;i<n;i++)link(ed[i][0],ed[i][1]);
  read(rt);
  makeroot(rt);
  while(m--){
    read(k);
    if(k==1){//换根
      read(rt);
      makeroot(rt);
    }
    if(k==9){//x的父亲变成y
      read(x),read(y);
      if(lca(x,y)==x)continue;
```

```
      cut(x);
      link(y,x);
      makeroot(rt);
    }
    if(k==0){//子树赋值
      read(x),read(y);
      changetree(x,tag(0,y));
    }
    if(k==5){//子树加
      read(x),read(y);
      changetree(x,tag(1,y));
    }
    if(k==3){//子树最小值
      read(x);
      printf("%d\n",asktree(x).minv);
    }
    if(k==4){//子树最大值
      read(x);
      printf("%d\n",asktree(x).maxv);
    }
    if(k==11){//子树和
      read(x);
      printf("%d\n",asktree(x).sum);
    }
    if(k==2){//链赋值
      read(x),read(y),read(z);
      changechain(x,y,tag(0,z));
      makeroot(rt);
    }
    if(k==6){//链加
      read(x),read(y),read(z);
      changechain(x,y,tag(1,z));
      makeroot(rt);
    }
    if(k==7){//链最小值
      read(x),read(y);
      printf("%d\n",askchain(x,y).minv);
      makeroot(rt);
    }
    if(k==8){//链最大值
      read(x),read(y);
      printf("%d\n",askchain(x,y).maxv);
      makeroot(rt);
    }
    if(k==10){//链和
      read(x),read(y);
      printf("%d\n",askchain(x,y).sum);
      makeroot(rt);
    }
  }
}
```

3.6 Splay

3.6.1 普通 Splay

1.ADD x y D：区间 $[x, y]$ 加 D。

2.REVERSE x y：将区间 $[x, y]$ 翻转。

3.REVOLVE x y T：将区间 $[x, y]$ 向右旋转 T 个单位。

4.INSERT x P：在第 x 个数后插入 P。

5.DELETE x：删去第 x 个数。

6.MIN x y：查询区间 $[x, y]$ 内的最小值。

```
int n,m,a[N],val[N],mn[N],tag[N],size[N],son[N][2],f[N],tot,root;bool rev[N];
void rev1(int x){if(!x)return;swap(son[x][0],son[x][1]);rev[x]^=1;}
void add1(int x,int p){if(!x)return;val[x]+=p;mn[x]+=p;tag[x]+=p;}
void pb(int x){
  if(rev[x]){
    rev1(son[x][0]);
    rev1(son[x][1]);
    rev[x]=0;
  }
  if(tag[x]){
    add1(son[x][0],tag[x]);
    add1(son[x][1],tag[x]);
    tag[x]=0;
  }
}
void up(int x){
  size[x]=1,mn[x]=val[x];
  if(son[x][0]){
    size[x]+=size[son[x][0]];
    if(mn[x]>mn[son[x][0]])mn[x]=mn[son[x][0]];
  }
  if(son[x][1]){
    size[x]+=size[son[x][1]];
    if(mn[x]>mn[son[x][1]])mn[x]=mn[son[x][1]];
  }
}
void rotate(int x){
  int y=f[x],w=son[y][1]==x;
  son[y][w]=son[x][w^1];
  if(son[x][w^1])f[son[x][w^1]]=y;
  if(f[y]){
    int z=f[y];
    if(son[z][0]==y)son[z][0]=x;
    if(son[z][1]==y)son[z][1]=x;
  }
  f[x]=f[y];son[x][w^1]=y;f[y]=x;up(y);
}
void splay(int x,int w){
  int s=1,i=x,y;a[1]=x;
  while(f[i])a[++s]=i=f[i];
  while(s)pb(a[s--]);
  while(f[x]!=w){
    y=f[x];
```

```
    if(f[y]!=w){if((son[f[y]][0]==y)^(son[y][0]==x))rotate(x);else rotate(y);}
    rotate(x);
  }
  if(!w)root=x;
  up(x);
}
int build(int l,int r,int fa){
  int x=++tot,mid=(l+r)>>1;
  f[x]=fa;val[x]=a[mid];
  if(l<mid)son[x][0]=build(l,mid-1,x);
  if(r>mid)son[x][1]=build(mid+1,r,x);
  up(x);
  return x;
}
int kth(int k){
  int x=root,tmp;
  while(1){
    pb(x);
    tmp=size[son[x][0]]+1;
    if(k==tmp)return x;
    if(k<tmp)x=son[x][0];else k-=tmp,x=son[x][1];
  }
}
int main(){
  scanf("%d",&n);
  for(int i=1;i<=n;i++)scanf("%d",&a[i]);
  root=build(0,n+1,0);
  scanf("%d",&m);
  while(m--){
    char op[9];int x,y,z;
    scanf("%s%d",op,&x);
    if(op[0]=='A'){
      scanf("%d%d",&y,&z);
      x=kth(x),y=kth(y+2);
      splay(x,0),splay(y,x),add1(son[y][0],z);
    }
    if(op[0]=='R'&&op[3]=='E'){
      scanf("%d",&y);
      x=kth(x),y=kth(y+2);
      splay(x,0),splay(y,x),rev1(son[y][0]);
    }
    if(op[0]=='R'&&op[3]=='O'){
      scanf("%d%d",&y,&z),z%=y-x+1;
      if(z){
        int u=x,t;
        x=kth(y-z+1),y=kth(y+2);
        splay(x,0),splay(y,x),t=son[y][0];
        son[y][0]=0,up(y),up(x);
        x=kth(u),y=kth(u+1);
        splay(x,0),splay(y,x),son[y][0]=t,f[t]=y;
        up(y),up(x);
      }
    }
    if(op[0]=='I'){
      scanf("%d",&y);
      x=kth(x+1);
      splay(x,0);
```

```
      f[++tot]=x,val[tot]=y;
      son[tot][1]=son[x][1],f[son[x][1]]=tot,son[x][1]=tot;
      up(tot),up(x);
    }
    if(op[0]=='D'){
      y=x;
      x=kth(x),y=kth(y+2);
      splay(x,0),splay(y,x),son[y][0]=0;
      up(y),up(x);
    }
    if(op[0]=='M'){
      scanf("%d",&y);
      x=kth(x),y=kth(y+2);
      splay(x,0),splay(y,x),printf("%d\n",mn[son[y][0]]);
    }
  }
}
```

3.6.2 缩点 Splay

0 p a b：在 p 位置和 $p+1$ 位置之间插入整数 $a, a+1, a+2, ..., b-1, b$。

1 a b：删除 $a, a+1, a+2, ..., b-1, b$ 位置的元素。

2 p：查询 p 位置的元素。

```
int n,m,i,k,x,y,z,tmp[N],tot,root,f[N],son[N][2],l[N],r[N],sum[N];
void build(int fa,int a,int b){
  int mid=(a+b)>>1,x=++tot;
  f[x]=fa,l[x]=r[x]=tmp[mid],sum[x]=b-a+1;
  if(a==b)return;
  if(mid>a)son[x][0]=tot+1,build(x,a,mid-1);
  if(mid<b)son[x][1]=tot+1,build(x,mid+1,b);
}
void up(int x){sum[x]=sum[son[x][0]]+sum[son[x][1]]+r[x]-l[x]+1;}
void setson(int x,int w,int y){son[x][w]=y;if(y)f[y]=x;}
void rotate(int x){
  int y=f[x],w=son[y][1]==x;
  son[y][w]=son[x][!w];
  if(son[x][!w])f[son[x][!w]]=y;
  if(f[y]){
    int z=f[y];
    if(son[z][0]==y)son[z][0]=x;
    if(son[z][1]==y)son[z][1]=x;
  }
  f[x]=f[y];f[y]=x;son[x][!w]=y;up(y);
}
void splay(int x,int w=0){
  while(f[x]!=w){
    int y=f[x];
    if(f[y]!=w){if((son[f[y]][0]==y)^(son[y][0]==x))rotate(x);else rotate(y);}
    rotate(x);
  }
  up(x);
  if(!w)root=x;
}
```

```
int kth(int k,int id=0){
  int x=root,nl,nr;
  while(1){
    nl=sum[son[x][0]]+1;nr=nl+r[x]-l[x];
    if(nl<=k&&k<=nr)return id?x:k-nl+l[x];
    if(k<nl)x=son[x][0];
    else k-=nr,x=son[x][1];
  }
}
int takeout(int k){
  int x=kth(k,1),val=kth(k);
  splay(x);
  int tl=l[x],tr=r[x],sl=son[x][0],sr=son[x][1];
  l[x]=r[x]=val;
  if(val!=tl){
    int y=++tot;
    l[y]=tl;r[y]=val-1;
    setson(y,0,sl);up(y);setson(x,0,y);
  }else setson(x,0,sl);
  if(val!=tr){
    int y=++tot;
    l[y]=val+1;r[y]=tr;
    setson(y,1,sr);up(y);setson(x,1,y);
  }else setson(x,1,sr);
  up(x);
  return x;
}
void ins(int k,int a,int b){
  int y=++tot;
  takeout(k+1);
  l[y]=a,r[y]=b;
  setson(y,1,son[root][1]);up(y);setson(root,1,y);up(root);
}
void del(int a,int b){
  int x=takeout(b+2);
  takeout(a);
  splay(x,root);setson(x,0,0);up(x);up(root);
}
int main(){
  scanf("%d%d",&n,&m);
  for(i=root=1;i<=n;i++)scanf("%d",tmp+i);
  build(0,0,n+1);
  while(m--){
    scanf("%d%d",&k,&x);
    if(k==0)scanf("%d%d",&y,&z),ins(x,y,z);
    if(k==1)scanf("%d",&y),del(x,y);
    if(k==2)printf("%d\n",kth(x+1));
  }
}
```

3.7 Treap

```
struct node{
  int val,cnt,sum,p;node *l,*r;
  node(){val=cnt=sum=p=0;l=r=NULL;}
```

```
  void up(){sum=cnt+l->sum+r->sum;}
}*blank=new(node),*root;
void Init(){
  blank->l=blank->r=blank;
  root=blank;
}
void Rotatel(node*&x){node*y=x->r;x->r=y->l;x->up();y->l=x;y->up();x=y;}
void Rotater(node*&x){node*y=x->l;x->l=y->r;x->up();y->r=x;y->up();x=y;}
//插入一个p
void Insert(node*&x,int p){
  if(x==blank){
    x=new(node);x->val=p;x->l=x->r=blank;x->cnt=x->sum=1;x->p=rand();
    return;
  }
  x->sum++;
  if(p==x->val){x->cnt++;return;}
  if(p<x->val){
    Insert(x->l,p);
    if(x->l->p>x->p)Rotater(x);
  }else{
    Insert(x->r,p);
    if(x->r->p>x->p)Rotatel(x);
  }
}
//删除一个p
void Delete(node*&x,int p){
  x->sum--;
  if(p==x->val){x->cnt--;return;}
  if(p<x->val)Delete(x->l,p);else Delete(x->r,p);
}
//查询大于p的数字的个数
int Ask(node*&x,int p){
  if(x==blank)return 0;
  if(p==x->val)return x->r->sum;
  if(p<x->val)return x->cnt+x->r->sum+Ask(x->l,p);
  return Ask(x->r,p);
}
//查询在[c,d]范围内的数字的个数
int Ask(node*&x,int a,int b,int c,int d){
  if(x==blank)return 0;
  if(c<=a&&b<=d)return x->sum;
  int t=c<=x->val&&x->val<=d?x->cnt:0;
  if(c<x->val)t+=Ask(x->l,a,x->val-1,c,d);
  if(d>x->val)t+=Ask(x->r,x->val+1,b,c,d);
}
```

3.8 替罪羊树实现动态标号

维护一个序列，一开始为空，支持以下操作：

1.Insert x：在序列中插入一个数，且插入后它位于从左往右第 x 个。

2.Ask x y：询问第 x 插入的数和第 y 个插入的数中哪一个在左边。

用替罪羊树支持动态标号，对于查询 x,y，等价于比较 $tm[x]$ 与 $tm[y]$ 哪个更小。插入 $O(\log n)$，查询 $O(1)$。

```
#include<cstdio>
#include<cmath>
#define N 400010
using namespace std;
typedef unsigned long long ll;
const ll inf=1ULL<<63;
const double A=0.8;
ll tl[N],tr[N],tm[N];
int size[N],son[N][2],f[N],tot,root;
int id[N],cnt;
int ins(int x,int b){
  size[x]++;
  if(!son[x][b]){
    son[x][b]=++tot;f[tot]=x;size[tot]=1;
    if(!b)tl[tot]=tl[x],tr[tot]=tm[x];else tl[tot]=tm[x],tr[tot]=tr[x];
    tm[tot]=(tl[tot]+tr[tot])>>1;
    return tot;
  }else return ins(son[x][b],b);
}
void dfs(int x){
  if(son[x][0])dfs(son[x][0]);
  id[++cnt]=x;
  if(son[x][1])dfs(son[x][1]);
}
int build(int fa,int l,int r,ll a,ll b){
  int mid=(l+r)>>1,x=id[mid];
  f[x]=fa;son[x][0]=son[x][1]=0;size[x]=1;tl[x]=a;tr[x]=b;tm[x]=(a+b)>>1;
  if(l==r)return x;
  if(l<mid)size[x]+=size[son[x][0]=build(x,l,mid-1,a,tm[x])];
  if(r>mid)size[x]+=size[son[x][1]=build(x,mid+1,r,tm[x],b)];
  return x;
}
int rebuild(int x){
  cnt=0;dfs(x);return build(f[x],1,cnt,tl[x],tr[x]);
}
int kth(int k){
  int x=root,rank;
  while(1){
    size[x]++;
    rank=size[son[x][0]]+1;
    if(k==rank)return x;
    if(k<rank)x=son[x][0];else k-=rank,x=son[x][1];
  }
}
void kthins(int k){
  if(!root){root=tot=size[1]=1;tr[1]=inf,tm[1]=inf>>1;return;}
  int x;
  if(k==1)x=ins(root,0);
  else if(k>size[root])x=ins(root,1);
  else{
    x=kth(k);
    if(son[x][0])x=ins(son[x][0],1);else{
      son[x][0]=++tot;f[tot]=x;size[tot]=1;
      tl[tot]=tl[x],tr[tot]=tm[x];
      tm[tot]=(tl[tot]+tr[tot])>>1;
      x=tot;
```

```
    }
  }
  int deep=1;int z=x;while(f[z])z=f[z],deep++;
  if(deep<log(tot)/log(1/A))return;
  while((double)size[son[x][0]]<A*size[x]&&(double)size[son[x][1]]<A*size[x])x=f[x];
  if(!x)return;
  if(x==root){root=rebuild(x);return;}
  int y=f[x],b=son[y][1]==x,now=rebuild(x);
  son[y][b]=now;
}
```

3.9 权值线段树中位数查询

离散化后一共有 m 个元素，支持以下操作：

1.ins c d e：插入一个离散化后是 c 的元素，对 cnt 的贡献为 d，对 sum 的贡献为 e。

2.ask：查询所有数字与中位数的差值的绝对值的和。

```
struct T{
int v[N];ll sum[N];
void ins(int c,int d,int e){
  int a=1,b=m,x=1,mid;
  while(1){
    v[x]+=d,sum[x]+=e;
    if(a==b)return;
    x<<=1;
    if(c<=(mid=(a+b)>>1))b=mid;else x|=1,a=mid+1;
  }
}
ll ask(){
  if(v[1]<=1)return 0;
  int a=1,b=m,mid,t,k=(v[1]+1)/2,x=1,cnt=0;ll ans=0;
  while(a<b){
    mid=(a+b)>>1,t=v[x<<=1];
    if(k<=t)cnt+=v[x|1],ans+=sum[x|1],b=mid;
    else cnt-=t,ans-=sum[x],k-=t,a=mid+1,x|=1;
  }
  return ans-sum[x]/v[x]*cnt;
}
};
```

3.10 线段树合并

合并根为 x, y 的两棵线段树，区间范围为 $[a, b]$。

```
int merge(int x,int y,int a,int b){
  if(!x)return y;
  if(!y)return x;
  int z=++tot;
  if(a==b){
    v[z]=v[x]+v[y];
    return z;
  }
  int mid=(a+b)>>1;
```

```
  l[z]=merge(l[x],l[y],a,mid);
  r[z]=merge(r[x],r[y],mid+1,b);
  v[z]=v[l[z]]+v[r[z]];
  return z;
}
```

3.11 树链剖分

```
void dfs(int x){
  size[x]=1;
  for(int i=g[x];i;i=nxt[i])if(v[i]!=f[x]){
    f[v[i]]=x,d[v[i]]=d[x]+1;
    dfs(v[i]),size[x]+=size[v[i]];
    if(size[v[i]]>size[son[x]])son[x]=v[i];
  }
}
void dfs2(int x,int y){
  st[x]=++dfn;top[x]=y;
  if(son[x])dfs2(son[x],y);
  for(int i=g[x];i;i=nxt[i])if(v[i]!=son[x]&&v[i]!=f[x])dfs2(v[i],v[i]);
  en[x]=dfn;
}
//查询x,y两点的lca
int lca(int x,int y){
  for(;top[x]!=top[y];x=f[top[x]])if(d[top[x]]<d[top[y]]){int z=x;x=y;y=z;}
  return d[x]<d[y]?x:y;
}
//x是y的祖先，查询x到y方向的第一个点
int lca2(int x,int y){
  int t;
  while(top[x]!=top[y])t=top[y],y=f[top[y]];
  return x==y?t:son[x];
}
//以root为根对x的子树操作
void subtree(int x){
  if(x==root){change(1,n);return;}
  if(st[x]>st[root]||en[x]<en[root]){change(st[x],en[x]);return;}
  int y=lca2(x,root);
  change(1,st[y]-1),change(en[y]+1,n);
}
//对x到y路径上的点进行操作
void chain(int x,int y){
  for(;top[x]!=top[y];x=f[top[x]]){
    if(d[top[x]]<d[top[y]]){int z=x;x=y;y=z;}
    change(st[top[x]],st[x]);
  }
  if(d[x]<d[y]){int z=x;x=y;y=z;}
  change(st[y],st[x]);
}
```

3.12 李超线段树

支持插入一条线段，查询横坐标为某个值时最上面的线段。插入 $O(\log^2 n)$，查询 $O(\log n)$。

```
#include<cstdio>
#include<cmath>
#include<algorithm>
#define N 39989
using namespace std;
struct Seg{
  double k,b;
  Seg(){}
  Seg(int x0,int y0,int x1,int y1){
    if(x0==x1)k=0,b=max(y0,y1);
    else k=1.0*(y0-y1)/(x0-x1),b=-k*x0+y0;
  }
  double gety(int x){return k*x+b;}
}s[100010];
int m,op,cnt,X0,Y0,X1,Y1,ans,v[131000];
inline int sig(double x){return fabs(x)<1e-8?0:(x>0?1:-1);}
void ins(int x,int a,int b,int c,int d,int p){
  if(c<=a&&b<=d){
    if(sig(s[p].gety(a)-s[v[x]].gety(a))>0
       &&sig(s[p].gety(b)-s[v[x]].gety(b))>0){v[x]=p;return;}
    if(sig(s[p].gety(a)-s[v[x]].gety(a))<=0
       &&sig(s[p].gety(b)-s[v[x]].gety(b))<=0)return;
    if(a==b)return;
  }
  int mid=(a+b)>>1;
  if(c<=mid)ins(x<<1,a,mid,c,d,p);
  if(d>mid)ins(x<<1|1,mid+1,b,c,d,p);
}
void ask(int x,int a,int b,int c){
  if(sig(s[ans].gety(c)-s[v[x]].gety(c))<0)ans=v[x];
  else if(!sig(s[ans].gety(c)-s[v[x]].gety(c))&&ans>v[x])ans=v[x];
  if(a==b)return;
  int mid=(a+b)>>1;
  c<=mid?ask(x<<1,a,mid,c):ask(x<<1|1,mid+1,b,c);
}
int main(){
  s[0].b=-1;
  read(m);
  while(m--){
    read(op);
    if(!op){
      read(X0);
      ans=0,ask(1,1,N,X0);
      printf("%d\n",ans);
    }else{
      read(X0),read(Y0),read(X1),read(Y1);
      s[++cnt]=Seg(X0,Y0,X1,Y1);
      ins(1,1,N,X0,X1,cnt);
    }
  }
}
```

3.13 ST 表

```
int Log[N],f[17][N];
int ask(int x,int y){
  int k=log[y-x+1];
  return max(f[k][x],f[k][y-(1<<k)+1]);
}
int main(){
  for(i=2;i<=n;i++)Log[i]=Log[i>>1]+1;
  for(j=1;j<K;j++)for(i=1;i+(1<<j-1)<=n;i++)f[j][i]=max(f[j-1][i],f[j-1][i+(1<<j-1)]);
}
```

3.14 左偏树

顶部为最小元素。

```
typedef pair<int,int>P;
struct Node{
  int l,r,d;P v;
  Node(){}
  Node(int _l,int _r,int _d,P _v){l=_l,r=_r,d=_d,v=_v;}
}T[N];
int merge(int a,int b){
  if(!a)return b;
  if(!b)return a;
  if(T[a].v>T[b].v)swap(a,b);
  T[a].r=merge(T[a].r,b);
  if(T[T[a].l].d<T[T[a].r].d)swap(T[a].l,T[a].r);
  T[a].d=T[a].r?T[T[a].r].d+1:0;
  return a;
}
int pop(int a){
  int l=T[a].l,r=T[a].r;
  T[a].l=T[a].r=T[a].d=0;
  return merge(l,r);
}
```

3.15 带修改区间第 k 小

维护一个序列 a，支持以下操作：

1 x y：把 $a[x]$ 修改为 y。

2 x y k：查询 $[x,y]$ 内第 k 小值。

时间复杂度 $O(n\log^2 n)$，空间复杂度 $O(n\log n)$。

```
#include<cstdio>
#include<cstdlib>
#include<algorithm>
using namespace std;
const int N=200010,M=524289;
int n,m,cnt,i,a[N],op[N][4],b[N];
int lower(int x){
  int l=1,r=cnt,t,mid;
  while(l<=r)if(b[mid=(l+r)>>1]<=x)l=(t=mid)+1;else r=mid-1;
  return t;
}
```

```
struct node{
  int val,cnt,sum,p;node*l,*r;
  node(){val=sum=cnt=p=0;l=r=NULL;}
  inline void up(){sum=l->sum+r->sum+cnt;}
}*blank=new(node),*T[M],pool[2000000],*cur;
void Rotatel(node*&x){node*y=x->r;x->r=y->l;x->up();y->l=x;y->up();x=y;}
void Rotater(node*&x){node*y=x->l;x->l=y->r;x->up();y->r=x;y->up();x=y;}
void Ins(node*&x,int y,int p){
  if(x==blank){x=cur++;x->val=y;x->l=x->r=blank;x->sum=x->cnt=1;x->p=std::rand();return;}
  x->sum+=p;
  if(y==x->val){x->cnt+=p;return;}
  if(y<x->val){
    Ins(x->l,y,p);
    if(x->l->p>x->p)Rotater(x);
  }else{
    Ins(x->r,y,p);
    if(x->r->p>x->p)Rotatel(x);
  }
}
int Ask(node*x,int y){//ask how many <= y
  int t=0;
  while(x!=blank)if(y<x->val)x=x->l;else t+=x->l->sum+x->cnt,x=x->r;
  return t;
}
void add(int v,int i,int p){
  int a=1,b=cnt,mid,f=1,x=1;
  while(a<b){
    if(f)Ins(T[x],i,p);
    mid=(a+b)>>1;x<<=1;
    if(f=v<=mid)b=mid;else a=mid+1,x|=1;
  }
  Ins(T[x],i,p);
}
int kth(int l,int r,int k){
  int x=1,a=1,b=cnt,mid;
  while(a<b){
    mid=(a+b)>>1;x<<=1;
    int t=Ask(T[x],r)-Ask(T[x],l-1);
    if(k<=t)b=mid;else k-=t,a=mid+1,x|=1;
  }
  return a;
}
void build(int x,int a,int b){
  T[x]=blank;
  if(a==b)return;
  int mid=(a+b)>>1;
  build(x<<1,a,mid),build(x<<1|1,mid+1,b);
}
int main(){
  blank->l=blank->r=blank;
  while(~scanf("%d",&n)){
    cur=pool;
    for(i=1;i<=n;i++)read(a[i]),b[i]=a[i];
    cnt=n;
    read(m);
    for(i=1;i<=m;i++){
      read(op[i][0]),read(op[i][1]),read(op[i][2]);
```

```
      if(op[i][0]==1)b[++cnt]=op[i][2];else read(op[i][3]);
    }
    sort(b+1,b+cnt+1);
    for(i=1;i<=n;i++)a[i]=lower(a[i]);
    for(i=1;i<=m;i++)if(op[i][0]==1)op[i][2]=lower(op[i][2]);
    build(1,1,cnt);
    for(i=1;i<=n;i++)add(a[i],i,1);
    for(i=1;i<=m;i++){
      if(op[i][0]==1)add(a[op[i][1]],op[i][1],-1),add(a[op[i][1]]=op[i][2],op[i][1],1);
      else printf("%d\n",b[kth(op[i][1],op[i][2],op[i][3])]);
    }
  }
}
```

3.16 Segment Beats!

维护一个序列，支持下面几种操作：

1. 给一个区间 $[L,R]$ 加上一个数 x。
2. 把一个区间 $[L,R]$ 里小于 x 的数变成 x。
3. 把一个区间 $[L,R]$ 里大于 x 的数变成 x。
4. 求区间 $[L,R]$ 的和。
5. 求区间 $[L,R]$ 的最大值。
6. 求区间 $[L,R]$ 的最小值。

时间复杂度 $O(n\log^2 n)$。

```
const int N=1050000,inf=~0U>>1;
int n,m,i,a[500010],op,c,d,p,ans;
int len[N],cma[N],cmi[N],ma[N],ma2[N],mi[N],mi2[N],tma[N],tmi[N],ta[N];
long long sum[N],ret;
void tagma(int x,int p){
  tma[x]+=p;
  if(ma[x]==mi[x]){
    ma[x]+=p;
    mi[x]=ma[x];
    sum[x]=1LL*ma[x]*len[x];
    return;
  }
  ma[x]+=p;
  if(cma[x]+cmi[x]==len[x])mi2[x]+=p;
  sum[x]+=1LL*p*cma[x];
}
void tagmi(int x,int p){
  tmi[x]+=p;
  if(ma[x]==mi[x]){
    ma[x]+=p;
    mi[x]=ma[x];
    sum[x]=1LL*ma[x]*len[x];
    return;
  }
  mi[x]+=p;
  if(cma[x]+cmi[x]==len[x])ma2[x]+=p;
  sum[x]+=1LL*p*cmi[x];
```

```
}
void taga(int x,int p){
  ta[x]+=p;
  ma[x]+=p;
  mi[x]+=p;
  if(ma2[x]!=-inf)ma2[x]+=p;
  if(mi2[x]!=inf)mi2[x]+=p;
  sum[x]+=1LL*p*len[x];
}
void pb(int x){
  if(tma[x]){
    if(ma[x<<1]>ma[x<<1|1])tagma(x<<1,tma[x]);
    else if(ma[x<<1]<ma[x<<1|1])tagma(x<<1|1,tma[x]);
    else{
      tagma(x<<1,tma[x]);
      tagma(x<<1|1,tma[x]);
    }
    tma[x]=0;
  }
  if(tmi[x]){
    if(mi[x<<1]<mi[x<<1|1])tagmi(x<<1,tmi[x]);
    else if(mi[x<<1]>mi[x<<1|1])tagmi(x<<1|1,tmi[x]);
    else{
      tagmi(x<<1,tmi[x]);
      tagmi(x<<1|1,tmi[x]);
    }
    tmi[x]=0;
  }
  if(ta[x]){
    taga(x<<1,ta[x]);
    taga(x<<1|1,ta[x]);
    ta[x]=0;
  }
}
void up(int x){
  sum[x]=sum[x<<1]+sum[x<<1|1];
  if(ma[x<<1]>ma[x<<1|1]){
    ma[x]=ma[x<<1];
    ma2[x]=max(ma2[x<<1],ma[x<<1|1]);
    cma[x]=cma[x<<1];
  }else if(ma[x<<1]<ma[x<<1|1]){
    ma[x]=ma[x<<1|1];
    ma2[x]=max(ma[x<<1],ma2[x<<1|1]);
    cma[x]=cma[x<<1|1];
  }else{
    ma[x]=ma[x<<1];
    ma2[x]=max(ma2[x<<1],ma2[x<<1|1]);
    cma[x]=cma[x<<1]+cma[x<<1|1];
  }
  if(mi[x<<1]<mi[x<<1|1]){
    mi[x]=mi[x<<1];
    mi2[x]=min(mi2[x<<1],mi[x<<1|1]);
    cmi[x]=cmi[x<<1];
  }else if(mi[x<<1]>mi[x<<1|1]){
    mi[x]=mi[x<<1|1];
    mi2[x]=min(mi[x<<1],mi2[x<<1|1]);
    cmi[x]=cmi[x<<1|1];
```

```
  }else{
    mi[x]=mi[x<<1];
    mi2[x]=min(mi2[x<<1],mi2[x<<1|1]);
    cmi[x]=cmi[x<<1]+cmi[x<<1|1];
  }
}
void build(int x,int a,int b){
  len[x]=b-a+1;
  if(a==b){
    ma[x]=mi[x]=sum[x]=::a[a],ma2[x]=-inf,mi2[x]=inf;
    cma[x]=cmi[x]=1;
    return;
  }
  int mid=(a+b)>>1;
  build(x<<1,a,mid),build(x<<1|1,mid+1,b);
  up(x);
}
void change(int x,int a,int b){
  if(c<=a&&b<=d){
    taga(x,p);
    return;
  }
  pb(x);
  int mid=(a+b)>>1;
  if(c<=mid)change(x<<1,a,mid);
  if(d>mid)change(x<<1|1,mid+1,b);
  up(x);
}
void cmax(int x,int a,int b){
  if(c<=a&&b<=d){
    if(mi[x]>=p)return;
    if(mi2[x]>p){
      tagmi(x,p-mi[x]);
      return;
    }
  }
  pb(x);
  int mid=(a+b)>>1;
  if(c<=mid)cmax(x<<1,a,mid);
  if(d>mid)cmax(x<<1|1,mid+1,b);
  up(x);
}
void cmin(int x,int a,int b){
  if(c<=a&&b<=d){
    if(ma[x]<=p)return;
    if(ma2[x]<p){
      tagma(x,p-ma[x]);
      return;
    }
  }
  pb(x);
  int mid=(a+b)>>1;
  if(c<=mid)cmin(x<<1,a,mid);
  if(d>mid)cmin(x<<1|1,mid+1,b);
  up(x);
}
void qsum(int x,int a,int b){
```

```
  if(c<=a&&b<=d){ret+=sum[x];return;}
  pb(x);
  int mid=(a+b)>>1;
  if(c<=mid)qsum(x<<1,a,mid);
  if(d>mid)qsum(x<<1|1,mid+1,b);
}
void qmax(int x,int a,int b){
  if(c<=a&&b<=d){ans=max(ans,ma[x]);return;}
  pb(x);
  int mid=(a+b)>>1;
  if(c<=mid)qmax(x<<1,a,mid);
  if(d>mid)qmax(x<<1|1,mid+1,b);
}
void qmin(int x,int a,int b){
  if(c<=a&&b<=d){ans=min(ans,mi[x]);return;}
  pb(x);
  int mid=(a+b)>>1;
  if(c<=mid)qmin(x<<1,a,mid);
  if(d>mid)qmin(x<<1|1,mid+1,b);
}
int main(){
  for(read(n),i=1;i<=n;i++)read(a[i]);
  build(1,1,n);read(m);
  while(m--){
    read(op),read(c),read(d);
    if(op==1)read(p),change(1,1,n);
    if(op==2)read(p),cmax(1,1,n);
    if(op==3)read(p),cmin(1,1,n);
    if(op==4)ret=0,qsum(1,1,n),printf("%lld\n",ret);
    if(op==5)ans=-inf,qmax(1,1,n),printf("%d\n",ans);
    if(op==6)ans=inf,qmin(1,1,n),printf("%d\n",ans);
  }
}
```

3.17 二维树状数组矩阵修改矩阵求和

```
int n,m,j,x1,y1,x2,y2,p;char op[9];
struct BIT{
int a[N][N];
void add(int x,int y,int p){for(;x<=n;x+=x&-x)for(j=y;j<=m;j+=j&-j)a[x][j]+=p;}
int sum(int x,int y){int t=0;for(;x;x-=x&-x)for(j=y;j;j-=j&-j)t+=a[x][j];return t;}
}T1,T2,T3,T4;
void up(int x,int y,int p){
  if(!x||!y)return;
  T1.add(x,y,x*y*p);
  T2.add(x,1,x*p),T2.add(x,y,-x*p);
  T3.add(1,y,y*p),T3.add(x,y,-y*p);
  T4.add(1,1,p),T4.add(x,y,p);
  T4.add(x,1,-p),T4.add(1,y,-p);
}
int ask(int x,int y){
  return x&&y?T1.sum(x,y)+T2.sum(x,y)*y+T3.sum(x,y)*x+T4.sum(x,y)*x*y:0;
}
int main(){
  scanf("%d%d",&n,&m);
```

```
  while(~scanf(”%s%d%d%d%d”,op,&x1,&y1,&x2,&y2)){
    if(op[0]==’L’){
      scanf(”%d”,&p);
      up(x2,y2,p),up(x1−1,y1−1,p),up(x2,y1−1,−p),up(x1−1,y2,−p);
    }else printf(”%d\n”,ask(x2,y2)+ask(x1−1,y1−1)−ask(x2,y1−1)−ask(x1−1,y2));
  }
}
```

3.18 并查集按秩合并

```
int F(int x){return f[x]==x?x:F(f[x]);}
void merge(int x,int y){
  x=F(x),y=F(y);
  if(x==y)return;
  if(d[x]==d[y])d[x]++;
  if(d[x]<d[y])swap(x,y);
  f[y]=x;
}
```

4 树

4.1 动态维护树的带权重心

支持单点修改，查询带权重心到所有点的带权距离和。修改 $O(\log^2 n)$，查询 $O(\log n)$。

```
#include<cstdio>
typedef long long ll;
const int N=100010,M=2000000,T=262145;
int n,m,i,x,y,z;
int g[N],nxt[N<<1],v[N<<1],w[N<<1],ok[N<<1],ed,son[N],f[N],all,now,cnt,value[N];
int size[N],heavy[N],top[N],loc[N],seq[N],dfn;
int G[N],NXT[M],V[2][M],W[M],ED,tag[T];
ll val[T],sw[N],sdw[N],sew[N],sedw[N];
void add(int x,int y,int z){v[++ed]=y,w[ed]=z,nxt[ed]=g[x],ok[ed]=1,g[x]=ed;}
void ADD(int x,int y,int z,int w){V[0][++ED]=y;V[1][ED]=z;W[ED]=w;NXT[ED]=G[x];G[x]=ED;}
void findroot(int x,int pre){
  son[x]=1;f[x]=0;
  for(int i=g[x];i;i=nxt[i])if(ok[i]&&v[i]!=pre){
    findroot(v[i],x);
    son[x]+=son[v[i]];
    if(son[v[i]]>f[x])f[x]=son[v[i]];
  }
  if(all-son[x]>f[x])f[x]=all-son[x];
  if(f[x]<f[now])now=x;
}
void dfs(int x,int pre,int dis){
  ADD(x,now,cnt,dis);
  for(int i=g[x];i;i=nxt[i])if(ok[i]&&v[i]!=pre)dfs(v[i],x,dis+w[i]);
}
void solve(int x){
  int i;
  for(i=g[x];i;i=nxt[i])if(ok[i])++cnt,dfs(v[i],x,w[i]);
  for(i=g[x];i;i=nxt[i])if(ok[i]){
    ok[i^1]=0;
    f[0]=all=son[v[i]];
    findroot(v[i],now=0);
    solve(now);
  }
}
void dfs1(int x,int y){
  size[x]=1;f[x]=y;
  for(int i=g[x];i;i=nxt[i])if(v[i]!=y){
    dfs1(v[i],x);size[x]+=size[v[i]];
    if(size[v[i]]>size[heavy[x]])heavy[x]=v[i];
  }
}
void dfs2(int x,int y){
  top[x]=y;seq[loc[x]=++dfn]=x;
  if(heavy[x])dfs2(heavy[x],y);
  for(int i=g[x];i;i=nxt[i])if(v[i]!=heavy[x]&&v[i]!=f[x])dfs2(v[i],v[i]);
}
void add1(int x,int p){val[x]+=p,tag[x]+=p;}
void pb(int x){if(tag[x])add1(x<<1,tag[x]),add1(x<<1|1,tag[x]),tag[x]=0;}
void change(int x,int a,int b,int c,int d,int p){
  if(c<=a&&b<=d){add1(x,p);return;}
```

```
  pb(x);
  int mid=(a+b)>>1;
  if(c<=mid)change(x<<1,a,mid,c,d,p);
  if(d>mid)change(x<<1|1,mid+1,b,c,d,p);
  val[x]=val[x<<1]>val[x<<1|1]?val[x<<1]:val[x<<1|1];
}
int getroot(){
  int x=1,a=1,b=n,mid;
  while(a<b){
    pb(x),mid=(a+b)>>1;
    if(val[x<<1|1]*2>=val[1])a=mid+1,x=x<<1|1;else b=mid,x<<=1;
  }
  return seq[a];
}
void modify(int x,int y){
  value[x]+=y;
  for(int i=G[x];i;i=NXT[i]){
    sw[V[0][i]]+=y,sdw[V[0][i]]+=(ll)W[i]*y;
    sew[V[1][i]]+=y,sedw[V[1][i]]+=(ll)W[i]*y;
  }
  while(top[x]!=1)change(1,1,n,loc[top[x]],loc[x],y),x=f[top[x]];
  change(1,1,n,1,loc[x],y);
}
ll query(int x){
  ll t=sdw[x];
  for(int i=G[x];i;i=NXT[i])
    t+=(sw[V[0][i]]-sew[V[1][i]]+value[V[0][i]])*W[i]+sdw[V[0][i]]-sedw[V[1][i]];
  return t;
}
int main(){
  read(n),read(m);
  for(ed=i=1;i<n;i++)read(x),read(y),read(z),add(x,y,z),add(y,x,z);
  f[0]=all=n;findroot(1,now=0);solve(now);
  dfs1(1,0),dfs2(1,1);
  while(m--)read(x),read(y),modify(x,y),printf("%lld\n",query(getroot()));
}
```

4.2 支持加边的树的重心的维护

ans 表示每个连通块的重心到其它点的距离和的和，时间复杂度 $O(n\log^2 n)$。

```
int n,m,i,x,y,ans;char op[5];
int g[N],v[N<<1],nxt[N<<1],; ed
int f[N],son[N][2],val[N],tag[N],sum[N],ts[N],td[N],size[N],tmp[N];
bool isroot(int x){return !f[x]||son[f[x]][0]!=x&&son[f[x]][1]!=x;}
void add1(int x,int p){if(!x)return;val[x]+=p;tag[x]+=p;}
void add2(int x,int s,int d){if(!x)return;sum[x]+=s+size[son[x][1]]*d;ts[x]+=s;td[x]+=d;}
void pb(int x){
  if(tag[x]){
    add1(son[x][0],tag[x]);
    add1(son[x][1],tag[x]);
    tag[x]=0;
  }
  if(td[x]){
    add2(son[x][0],ts[x]+(size[son[x][1]]+1)*td[x],td[x]);
```

```
    add2(son[x][1],ts[x],td[x]);
    ts[x]=td[x]=0;
  }
}
void up(int x){size[x]=size[son[x][0]]+size[son[x][1]]+1;}
void rotate(int x){
  int y=f[x],w=son[y][1]==x;
  son[y][w]=son[x][w^1];
  if(son[x][w^1])f[son[x][w^1]]=y;
  if(f[y]){
    int z=f[y];
    if(son[z][0]==y)son[z][0]=x;else if(son[z][1]==y)son[z][1]=x;
  }
  f[x]=f[y];f[y]=x;son[x][w^1]=y;up(y);
}
void splay(int x){
  int s=1,i=x,y;tmp[1]=i;
  while(!isroot(i))tmp[++s]=i=f[i];
  while(s)pb(tmp[s--]);
  while(!isroot(x)){
    y=f[x];
    if(!isroot(y)){if((son[f[y]][0]==y)^(son[y][0]==x))rotate(x);else rotate(y);}
    rotate(x);
  }
  up(x);
}
void access(int x){for(int y=0;x;y=x,x=f[x])splay(x),son[x][1]=y,up(x);}
int root(int x){access(x);splay(x);while(son[x][0])x=son[x][0];return x;}
void addleaf(int x,int y){
  f[y]=x,son[y][0]=son[y][1]=val[y]=tag[y]=sum[y]=ts[y]=td[y]=0,size[y]=1;
  x=root(x),access(y),splay(x),add1(x,1),add2(x,0,1);
  for(y=son[x][1];son[y][0];y=son[y][0]);splay(y);
  int vx=val[x],vy=val[y];
  if(vy*2>vx){
    val[y]=vx,val[x]-=vy;
    sum[x]-=sum[y]+vy,sum[y]+=sum[x]+vx-vy;
    access(y),splay(x),son[x][0]=y,son[x][1]=0;
  }
}
void dfs(int x,int y){
  addleaf(y,x);
  for(int i=g[x];i;i=nxt[i])if(v[i]!=y)dfs(v[i],x);
}
void addedge(int x,int y){v[++ed]=y;nxt[ed]=g[x];g[x]=ed;}
void link(int x,int y){
  int X=root(x),Y=root(y);
  ans-=sum[X]+sum[Y];
  if(val[X]<val[Y])swap(x,y);
  dfs(y,x),addedge(x,y),addedge(y,x);
  ans+=sum[root(x)];
}
int main(){
  scanf(”%d%d”,&n,&m);
  for(i=1;i<=n;i++)val[i]=size[i]=1;
  while(m--){
    scanf(”%s”,op);
    if(op[0]==’A’)scanf(”%d%d”,&x,&y),link(x,y);
```

```
    if(op[0]=='Q')printf("%d\n",ans);
  }
}
```

4.3 虚树

除了 1 之外再给定 m 个点，构造它们的虚树，时间复杂度 $O(m\log n)$。

```
int m,i,a[N],q[N],t,tot;bool vip[N],vis[N];
bool cmp(int x,int y){return st[x]<st[y];}
int main(){
  while(~scanf("%d",&m)){
    vis[1]=vip[1]=a[1]=1;
    for(tot=++m,i=2;i<=m;i++)read(a[i]),vis[a[i]]=vip[a[i]]=1;
    sort(a+1,a+m+1,cmp);
    for(i=1;i<m;i++)if(!vis[x=lca(a[i],a[i+1])])vis[a[++tot]=x]=1;
    m=tot,sort(a+1,a+m+1,cmp);
    for(q[t=1]=1,i=2;i<=m;q[++t]=a[i++]){
      while(st[a[i]]<st[q[t]]||en[a[i]]>en[q[t]])t--;
      addedge(q[t],a[i]);
    }
    for(i=1;i<=m;i++)vis[a[i]]=vip[a[i]]=0;
  }
}
```

4.4 曼哈顿最小生成树

```
int n,m,i,j,w[N],c[N],bit[N],f[N];ll ans;
struct P{
  int x,y,p;
  P(){}
  P(int _x,int _y,int _p){x=_x,y=_y,p=_p;}
}a[N],b[N],e[N<<2];
bool cmp(const P&a,const P&b){return a.x==b.x?a.y<b.y:a.x<b.x;}
bool cmpe(const P&a,const P&b){return a.p<b.p;}
int lower(int x){
  int l=1,r=n,t,mid;
  while(l<=r)if(c[mid=(l+r)>>1]<=x)l=(t=mid)+1;else r=mid-1;
  return t;
}
int abs(int x){return x>0?x:-x;}
int dis(int x,int y){return abs(a[x].x-a[y].x)+abs(a[x].y-a[y].y);}
void ins(int x,int p){for(;x<=n;x+=x&-x)if(w[p]<=w[bit[x]])bit[x]=p;}
int ask(int x){int t=0;for(;x;x-=x&-x)if(w[bit[x]]<=w[t])t=bit[x];return t;}
int F(int x){return f[x]==x?x:f[x]=F(f[x]);}
void solve(){
  for(sort(b+1,b+n+1,cmp),sort(c+1,c+n+1),i=1;i<=n;i++){
    if(j=ask(lower(b[i].y)))e[++m]=P(b[i].p,j,dis(b[i].p,j));
    ins(lower(b[i].y),b[i].p);
  }
}
ll ManhattanMst(){
  for(w[0]=~0U>>1,m=ans=0,i=1;i<=n;i++)f[i]=i;
```

```
  for(i=1;i<=n;i++){
    b[i]=P(-a[i].x,a[i].x-a[i].y,i);
    c[i]=b[i].y;
    w[i]=a[i].x+a[i].y;
    bit[i]=0;
  }
  solve();
  for(i=1;i<=n;i++){
    b[i]=P(-a[i].y,a[i].y-a[i].x,i);
    c[i]=b[i].y;
    bit[i]=0;
  }
  solve();
  for(i=1;i<=n;i++){
    b[i]=P(a[i].y,-a[i].x-a[i].y,i);
    c[i]=b[i].y;
    w[i]=a[i].x-a[i].y;
    bit[i]=0;
  }
  solve();
  for(i=1;i<=n;i++){
    b[i]=P(-a[i].x,a[i].y+a[i].x,i);
    c[i]=b[i].y;
    bit[i]=0;
  }
  solve();
  sort(e+1,e+m+1,cmpe);
  for(ans=0,i=1;i<=m;i++)if(F(e[i].x)!=F(e[i].y)){
    f[f[e[i].x]]=f[e[i].y];
    ans+=e[i].p;
  }
  return ans;
}
int main(){
  scanf("%d",&n);
  for(i=1;i<=n;i++)scanf("%d%d",&a[i].x,&a[i].y);
  printf("%lld",ManhattanMst());
}
```

4.5 树链求交

```
bool have(int x,int y){return st[x]<=st[y]&&en[y]<=en[x];}
int deeper(int x,int y){return d[x]>d[y]?x:y;}
int lower(int x,int y){return d[x]<d[y]?x:y;}
/*
0:none
-1:nothing
*/
struct P{
  int x,y,z;
  P(){}
  P(int _x,int _y,int _z){x=_x,y=_y,z=_z;}
  bool check(int o){
    if(!z)return 1;
    if(z<0)return 0;
```

```
    return have(z,o)&&(have(o,x)||have(o,y));
  }
  void read(){
    scanf("%d%d",&x,&y);
    z=lca(x,y);
  }
  ll ans(){
    if(z<1)return 0;
    return dis[x]+dis[y]-2*dis[z];
  }
}tmp,val[M];
P merge(P a,P b){
  if(!a.z)return b;
  if(!b.z)return a;
  if(a.z<0||b.z<0)return P(0,0,-1);
  if(d[a.z]<d[b.z])swap(a,b);
  if(!b.check(a.z))return P(0,0,-1);
  if(a.z==b.z){
    return P(deeper(lca(a.x,b.x),lca(a.x,b.y)),
             deeper(lca(a.y,b.x),lca(a.y,b.y)),a.z);
  }
  int x=deeper(lca(a.x,b.y),lca(a.y,b.x)),
      y=deeper(lca(a.x,b.x),lca(a.y,b.y));
  return P(lower(x,y),deeper(x,y),lca(a.z,b.z));
}
```

5 图

5.1 欧拉回路

欧拉回路：

无向图：每个顶点的度数都是偶数，则存在欧拉回路。

有向图：每个顶点的入度 = 出度，则存在欧拉回路。

欧拉路径：

无向图：当且仅当该图所有顶点的度数为偶数，或者除了两个度数为奇数外其余的全是偶数。

有向图：当且仅当该图所有顶点出度 = 入度或者一个顶点出度 = 入度 +1，另一个顶点入度 = 出度 +1，其他顶点出度 = 入度。

下面 $O(n+m)$ 求欧拉回路的代码中，n 为点数，m 为边数，若有解则依次输出经过的边的编号，若是无向图，则正数表示 x 到 y，负数表示 y 到 x。

```
namespace UndirectedGraph{
int n,m,i,x,y,d[N],g[N],v[M<<1],w[M<<1],vis[M<<1],nxt[M<<1],ed;
int ans[M],cnt;
void add(int x,int y,int z){
  d[x]++;
  v[++ed]=y;w[ed]=z;nxt[ed]=g[x];g[x]=ed;
}
void dfs(int x){
  for(int&i=g[x];i;){
    if(vis[i]){i=nxt[i];continue;}
    vis[i]=vis[i^1]=1;
    int j=w[i];
    dfs(v[i]);
    ans[++cnt]=j;
  }
}
void solve(){
  scanf("%d%d",&n,&m);
  for(i=ed=1;i<=m;i++)scanf("%d%d",&x,&y),add(x,y,i),add(y,x,-i);
  for(i=1;i<=n;i++)if(d[i]&1){puts("NO");return;}
  for(i=1;i<=n;i++)if(g[i]){dfs(i);break;}
  for(i=1;i<=n;i++)if(g[i]){puts("NO");return;}
  puts("YES");
  for(i=m;i;i--)printf("%d ",ans[i]);
}
}
namespace DirectedGraph{
int n,m,i,x,y,d[N],g[N],v[M],vis[M],nxt[M],ed;
int ans[M],cnt;
void add(int x,int y){
  d[x]++;d[y]--;
  v[++ed]=y;nxt[ed]=g[x];g[x]=ed;
}
void dfs(int x){
  for(int&i=g[x];i;){
    if(vis[i]){i=nxt[i];continue;}
    vis[i]=1;
```

```
    int j=i;
    dfs(v[i]);
    ans[++cnt]=j;
  }
}
void solve(){
  scanf(”%d%d”,&n,&m);
  for(i=1;i<=m;i++)scanf(”%d%d”,&x,&y),add(x,y);
  for(i=1;i<=n;i++)if(d[i]){puts(”NO”);return;}
  for(i=1;i<=n;i++)if(g[i]){dfs(i);break;}
  for(i=1;i<=n;i++)if(g[i]){puts(”NO”);return;}
  puts(”YES”);
  for(i=m;i;i—)printf(”%d ”,ans[i]);
}
}
```

5.2 最短路

5.2.1 Dijkstra

```
typedef pair<int,int> P;
priority_queue<P,vector<P>,greater<P> >Q;
void dijkstra(int S){
  int i,x;
  for(i=1;i<=n;i++)d[i]=inf;Q.push(P(d[S]=0,S));
  while(!Q.empty()){
    P t=Q.top();Q.pop();
    if(d[x=t.second]<t.first)continue;
    for(i=g[x];i;i=nxt[i])if(d[x]+w[i]<d[v[i]])Q.push(P(d[v[i]]=d[x]+w[i],v[i]));
  }
}
```

5.2.2 SPFA

```
int q[66000];unsigned short h,t;
void add(int x,int y){
  if(y>=d[x])return;d[x]=y;
  if(!in[x]){
    in[x]=1;
    if(y<d[q[h]])q[—h]=x;else q[++t]=x;//SLF优化
  }
}
void spfa(int S){//S为源点
  int i,x;
  for(i=h=1;i<=n;i++)d[i]=inf,in[i]=0;add(S,t=0);
  while(h!=t+1)for(i=g[x=q[h++]],in[x]=0;i;i=nxt[i])add(v[i],d[x]+w[i]);
}
```

5.2.3 Astar 求 k 短路

求有向图中 S 到 T 的前 k 短路，其中 g 为反图，h 为正图。

```
typedef pair<int,int> P;
const int N=1010,M=10010,inf=1000000010;
int n,m,k,S,T,i,x,y,z;
int g[N],h[N],v[M<<1],w[M<<1],nxt[M<<1],ed,d[N],vis[N],ans[N];
priority_queue<P,vector<P>,greater<P> >Q;
void add(int x,int y,int z){
  v[++ed]=x;w[ed]=z;nxt[ed]=g[y];g[y]=ed;
  v[++ed]=y;w[ed]=z;nxt[ed]=h[x];h[x]=ed;
}
int main(){
  scanf("%d%d%d",&n,&m,&k);S=n,T=1;
  for(i=1;i<=k;i++)ans[i]=-1;
  while(m--)scanf("%d%d%d",&x,&y,&z),add(x,y,z);
  for(i=1;i<=n;i++)d[i]=inf;Q.push(P(d[T]=0,T));
  while(!Q.empty()){
    P t=Q.top();Q.pop();
    if(d[t.second]<t.first)continue;
    for(i=g[x=t.second];i;i=nxt[i])if(d[x]+w[i]<d[v[i]])
      Q.push(P(d[v[i]]=d[x]+w[i],v[i]));
  }
  if(d[S]<inf)Q.push(P(d[S],S));
  while(!Q.empty()){
    P t=Q.top();Q.pop();vis[x=t.second]++;
    if(x==T&&vis[T]<=k)ans[vis[T]]=t.first;
    if(vis[T]>k)break;
    if(vis[x]<=k)for(i=h[x];i;i=nxt[i])Q.push(P(t.first-d[x]+d[v[i]]+w[i],v[i]));
  }
  for(i=1;i<=k;i++)printf("%d\n",ans[i]);
}
```

5.2.4 稳定 k 短路

```
#include<cstdio>
#include<algorithm>
#include<queue>
#include<vector>
using namespace std;
typedef pair<int,int>P;
const int N=1010,M=100010,inf=~0U>>1;
int n,m,i,S,T,K,x,y,z;
int g[N],v[M],u[M],w[M],nxt[M],d[N],f[N],h[N],tot;bool is[M],vis[N];
struct Node{
  int l,r,d;P v;
  Node(){}
  Node(int _l,int _r,int _d,P _v){l=_l,r=_r,d=_d,v=_v;}
}pool[2000010];
int build(P v){
  pool[++tot]=Node(0,0,0,v);
  return tot;
}
int merge(int a,int b){
  if(!a||!b)return a+b;
  if(pool[a].v>pool[b].v)swap(a,b);
  int x=++tot;
```

```
  pool[x]=pool[a];
  pool[x].r=merge(pool[a].r,b);
  if(pool[pool[x].l].d<pool[pool[x].r].d)swap(pool[x].l,pool[x].r);
  pool[x].d=pool[x].r?pool[pool[x].r].d+1:0;
  return x;
}
void getdis(){
  int i,x;
  priority_queue<P,vector<P>,greater<P> >q;
  for(i=1;i<=n;i++)d[i]=inf,f[i]=0;
  q.push(P(d[T]=0,T));
  while(!q.empty()){
    P t=q.top();q.pop();
    if(t.first>d[x=t.second])continue;
    for(i=g[x];i;i=nxt[i])if(d[v[i]]>d[x]+w[i]){
      f[v[i]]=i;
      q.push(P(d[v[i]]=d[x]+w[i],v[i]));
    }
  }
}
void dfs(int x){
  if(!f[x]||vis[x])return;
  vis[x]=1;
  dfs(u[f[x]]);
  h[x]=merge(h[x],h[u[f[x]]]);
}
void add(int x,int y,int z){v[++m]=x;u[m]=y;w[m]=z;nxt[m]=g[y];g[y]=m;}
int solve(){
  int mm=m;
  for(m=0,i=1;i<=n;i++)g[i]=0;
  while(mm--)scanf(”%d%d%d”,&x,&y,&z),add(x,y,z);
  scanf(”%d%d%d”,&S,&T,&K);
  if(S==T)K++;
  getdis();
  if(d[S]==inf)return -1;
  if(K==1)return d[S];
  K--;
  for(i=1;i<=m;i++)is[i]=0;
  for(tot=0,i=1;i<=n;i++)is[f[i]]=1,h[i]=vis[i]=0;
  for(i=1;i<=m;i++)if(!is[i]&&d[u[i]]<inf)
    h[v[i]]=merge(h[v[i]],build(P(w[i]-d[v[i]]+d[u[i]],u[i])));
  for(i=1;i<=n;i++)dfs(i);
  priority_queue<P,vector<P>,greater<P> >q;
  int ans,x,y;
  y=h[S];
  if(y)q.push(P(d[S]+pool[y].v.first,y));
  while(!q.empty()&&K){
    K--;
    P t=q.top();q.pop();
    ans=t.first;
    x=t.second,y=pool[x].l;
    if(y)q.push(P(ans-pool[x].v.first+pool[y].v.first,y));
    y=pool[x].r;
    if(y)q.push(P(ans-pool[x].v.first+pool[y].v.first,y));
    y=h[pool[x].v.second];
    if(y)q.push(P(ans+pool[y].v.first,y));
  }
```

```
  return K?-1:ans;
}
int main(){
  while(~scanf("%d%d",&n,&m))printf("%d\n",solve());
}
```

5.3 Tarjan

5.3.1 边双连通分量

$cut[i]$ 表示输入的第 i 条边是否是桥边，cnt 表示边双连通分量的个数，$from[i]$ 表示 i 点所属的边双连通分量。

```
int e[M][2],cut[M],g[N],v[M<<1],nxt[M<<1],ed=1;
int f[N],dfn[N],low[N],num,cnt,from[N];
void add(int x,int y){v[++ed]=y;nxt[ed]=g[x];g[x]=ed;}
void tarjan(int x){
  dfn[x]=low[x]=++num;
  for(int i=g[x];i;i=nxt[i])if(!dfn[v[i]]){
    f[v[i]]=i>>1,tarjan(v[i]);
    if(low[x]>low[v[i]])low[x]=low[v[i]];
  }else if(f[x]!=(i>>1)&&low[x]>dfn[v[i]])low[x]=dfn[v[i]];
  if(f[x]&&low[x]==dfn[x])cut[f[x]]=1;
}
void dfs(int x,int y){
  from[x]=y;
  for(int i=g[x];i;i=nxt[i])if(!from[v[i]]&&!cut[i>>1])dfs(v[i],y);
}
int main(){
  scanf("%d%d",&n,&m);
  for(ed=i=1;i<=m;i++){
    scanf("%d%d",&x,&y);
    e[i][0]=x,e[i][1]=y;
    add(x,y),add(y,x);
  }
  tarjan(1);
  for(i=1;i<=n;i++)if(!from[i])dfs(i,++cnt);
}
```

5.3.2 点双连通分量

$>n$ 的点表示点双，建立 Block Forest Data Structure。

```
void tarjan(int x,int f){
  dfn[x]=low[x]=++num,q[++t]=x;
  for(int i=g[x];i;i=nxt[i])if(!dfn[v[i]]){
    int y=v[i];
    tarjan(y,x);
    if(low[x]>low[y])low[x]=low[y];
    if(!f)sub++;
    if(dfn[x]<=low[y]&&f||!f&&sub>1){
      cut[x]=1;
      ADD(++all,x);
```

```
      while(1){
        int z=q[t--];
        ADD(all,z);
        if(z==y)break;
      }
    }
  }else if(low[x]>dfn[v[i]])low[x]=dfn[v[i]];
}
int main(){
  for(i=1;i<=n;i++)if(!dfn[i]){
    sub=0,tarjan(i,0);
    if(t){
      all++;
      while(t)ADD(all,q[t--]);
    }
  }
}
```

5.3.3 Dominator Tree

在保证 S 能到达所有点的情况下，求以 S 为源点的 Dominator Tree。

$dfn[x]$：x 的 DFS 序。

$id[x]$：DFS 序第 x 个是什么。

$gd[x]$：DFS 序第 x 个在 Dominator Tree 上的孩子列表。

$idom[x]$：DFS 序第 x 个在 Dominator Tree 上的父亲。

$sd[x]$：DFS 序第 x 个的半必经点。

$id[idom[dfn[x]]]$：x 的最近必经点。

```
#include<cstdio>
const int N=5010,M=200010;//点数，边数
int n,m,i,x,y,q[N],ans;
int g1[N],g2[N],gd[N],v[M*3+N],nxt[M*3+N],ed;
int cnt,dfn[N],id[N],fa[N],f[N],mn[N],sd[N],idom[N];
void add(int*g,int x,int y){v[++ed]=y;nxt[ed]=g[x];g[x]=ed;}
int F(int x){
  if(f[x]==x)return x;
  int y=F(f[x]);
  if(sd[mn[x]]>sd[mn[f[x]]])mn[x]=mn[f[x]];
  return f[x]=y;
}
void dfs(int x){
  id[dfn[x]=++cnt]=x;
  for(int i=g1[x];i;i=nxt[i])if(!dfn[v[i]])dfs(v[i]),fa[dfn[v[i]]]=dfn[x];
}
void tarjan(int S){
  int i,j,k,x;
  for(cnt=0,i=1;i<=n;i++)gd[i]=dfn[i]=id[i]=fa[i]=idom[i]=0,f[i]=sd[i]=mn[i]=i;
  dfs(S);
  for(i=n;i>1;i--){
    for(j=g2[id[i]];j;j=nxt[j])F(k=dfn[v[j]]),sd[i]=sd[i]<sd[mn[k]]?sd[i]:sd[mn[k]];
    add(gd,sd[i],i);
    for(j=gd[f[i]=x=fa[i]];j;j=nxt[j])F(k=v[j]),idom[k]=sd[mn[k]]<x?mn[k]:x;
    gd[x]=0;
```

```
  }
  for(i=2;i<=n;add(gd,idom[i],i),i++)if(idom[i]!=sd[i])idom[i]=idom[idom[i]];
}
int main(){
  while(~scanf(”%d%d”,&n,&m)){
    for(ed=0,i=1;i<=n;i++)g1[i]=g2[i]=0;
    while(m--)scanf(”%d%d”,&x,&y),add(g1,x,y),add(g2,y,x);
    tarjan(1);
  }
}
```

5.4 强连通分量

$G[0]$ 为正图，$G[1]$ 为反图，$G[2]$ 为缩点后的图，$f[i]$ 为 i 所在的 SCC。

```
int G[3][N],NXT[3][M<<1],V[3][M<<1],ed,f[N],q[N],t,vis[N];
void add(int x,int y){
  V[0][++ed]=y;NXT[0][ed]=G[0][x];G[0][x]=ed;
  V[1][ed]=x;NXT[1][ed]=G[1][y];G[1][y]=ed;
}
void ADD(int x,int y){V[2][++ed]=y;NXT[2][ed]=G[2][x];G[2][x]=ed;}
void dfs1(int x){
  vis[x]=1;
  for(int i=G[0][x];i;i=NXT[0][i])if(!vis[V[0][i]])dfs1(V[0][i]);
  q[++t]=x;
}
void dfs2(int x,int y){
  vis[x]=0,f[x]=y;
  for(int i=G[1][x];i;i=NXT[1][i])if(vis[V[1][i]])dfs2(V[1][i],y);
}
int main(){
  for(t=0,i=1;i<=n;i++)if(!vis[i])dfs1(i);
  for(i=n;i;i--)if(vis[q[i]])dfs2(q[i],q[i]);
  for(ed=0,i=1;i<=n;i++)for(j=G[0][i];j;j=NXT[0][j])
    if(f[i]!=f[V[0][j]])ADD(f[i],f[V[0][j]]);
}
```

5.5 无负权图的最小环

有向图：$d[i][i]=inf$，然后跑 floyd，$ans=\min(d[i][i])$。

求无向图中经过至少 3 个点的最小环代码如下：

```
int main(){
  for(i=1;i<=n;i++)for(j=1;j<=n;j++)g[i][j]=d[i][j]=inf;
  while(m--){
    scanf(”%d%d%d”,&x,&y,&z);
    if(z<g[x][y])g[x][y]=g[y][x]=d[x][y]=d[y][x]=z;
  }
  for(ans=inf,k=1;k<=n;k++){
    for(i=1;i<k;i++)for(j=i+1;j<k;j++)ans=min(ans,d[i][j]+g[i][k]+g[k][j]);
    for(i=1;i<=n;i++)for(j=1;j<=n;j++)d[i][j]=min(d[i][j],d[i][k]+d[k][j]);
  }
}
```

5.6 2-SAT

设一共有 n 个变量，对于一个变量 i，i 点表示它为 0，$i+n$ 点表示它为 1，$vis[i]$ 表示 i 点选不选。注意添加逆否命题。以下算法复杂度很高。

```
int q[N<<1],t;bool vis[N<<1];
bool dfs(int x){
  if(vis[x>n?x-n:x+n])return 0;
  if(vis[x])return 1;
  vis[q[++t]=x]=1;
  for(int i=g[x];i;i=nxt[i])if(!dfs(v[i]))return 0;
  return 1;
}
bool solve(){
  for(int i=1;i<=n;i++)if(!vis[i]&&!vis[i+n]){
    t=0;
    if(!dfs(i)){
      while(t)vis[q[t--]]=0;
      if(!dfs(i+n))return 0;
    }
  }
  return 1;
}
```

以下算法复杂度正确。

```
int q[N],t,f[N];bool vis[N];
struct E{int v;E*nxt;}*g[N],*h[N],pool[N*4],*cur;
inline void add(int x,int y){
  E*p=cur++;p->v=y;p->nxt=g[x];g[x]=p;
  p=cur++;p->v=x;p->nxt=h[y];h[y]=p;
}
void dfs1(int x){
  vis[x]=1;
  for(E*p=g[x];p;p=p->nxt)if(!vis[p->v])dfs1(p->v);
  q[++t]=x;
}
void dfs2(int x,int y){
  vis[x]=0,f[x]=y;
  for(E*p=h[x];p;p=p->nxt)if(vis[p->v])dfs2(p->v,y);
}
int main(){
  for(cur=pool,i=0;i<=n+n;i++)g[i]=h[i]=NULL,vis[i]=0;
  addedge;
  int o=0;
  for(t=0,i=1;i<=n+n;i++)if(!vis[i])dfs1(i);
  for(i=t;i;i--)if(vis[q[i]])dfs2(q[i],++o);
  for(i=1;i<=n;i++)if(f[i]==f[i+n])return;
  for(i=1;i<=n;i++)printf(”%d\n”,f[i]<f[i+n]?i+n:i);
}
```

5.7 完美消除序列

一个无向图称为弦图当图中任意长度大于 3 的环都至少有一个弦。

弦图的方法有着很多经典用途：例如用最少的颜色给每个点染色使得相邻的点染的颜色不同，通过完美消除序列从后往前依次给每个点染色，给每个点染上可以染的最小的颜色；最大独立集问题，选择最多的点使得任意两个点不相邻，通过完美消除序列从前往后能选就选。

给定一张 n 个点 m 条边的弦图，求把点最少分成多少组，使得每组点之间没有边。从后往前求完美消除序列。必要时请加上优先队列优化，时间复杂度 $O((n+m)\log n)$。

```
int i,j,k,col[N],ans;bool vis[N];
int main(){
  for(i=n;i;i—){
    for(k=0,j=1;j<=n;j++)if(!vis[j]&&col[j]>=col[k])k=j;
    for(vis[k]=1,j=g[k];j;j=nxt[j])if(!vis[v[j]])if(++col[v[j]]>ans)ans=col[v[j]];
  }
  printf(”%d”,ans+1);
}
```

5.8 最大团

5.8.1 搜索

```
int n,i,j,k,max[N],g[N][N],f[N][N],ans;
int dfs(int cur,int tot) {
  if(!cur){
    if(tot>ans)return ans=tot,1;
    return 0;
  }
  for(int i=0,j,u,nxt;i<cur;i++){
    if(cur−i+tot<=ans)return 0;
    u=f[tot][i],nxt=0;
    if(max[u]+tot<=ans)return 0;
    for(j=i+1;j<cur;j++)if(g[u][f[tot][j]])f[tot+1][nxt++]=f[tot][j];
    if(dfs(nxt,tot+1))return 1;
  }
  return 0;
}
int main(){
  scanf(”%d”,&n);
  while(scanf(”%d%d”,&i,&j)!=EOF)g[i−1][j−1]=g[j−1][i−1]=1;
  for(i=n−1;~i;dfs(k,1),max[i−−]=ans)for(k=0,j=i+1;j<n;j++)if(g[i][j])f[1][k++]=j;
  printf(”%d”,ans);
}
```

5.8.2 随机贪心

```
int T,n,m,i,j,k,g[N][N],a[N],del[N],ans,fin[N];
void solve(){
  for(i=0;i<n;i++)del[i]=0;
  for(k=i=0;i<n;i++)if(!del[i])for(k++,j=i+1;j<n;j++)if(!g[a[i]][a[j]])del[j]=1;
  if(k>ans)for(ans=k,i=j=0;i<n;i++)if(!del[i])fin[j++]=a[i];
}
int main(){
  scanf(”%d%d”,&n,&m);
```

```
  for(i=0;i<n;i++)a[i]=i;
  while(m--)scanf(”%d%d”,&i,&j),g[i][j]=g[j][i]=1;
  for(T=100;T--;solve())for(i=0;i<n;i++)swap(a[i],a[rand()%n]);
  for(printf(”%d\n”,ans),i=0;i<ans;i++)printf(”%d ”,fin[i]+1);
}
```

5.8.3 独立集最大团计数

```
typedef unsigned long long ULL;
int ctz(ULL s){return s?__builtin_ctzll(s):64;}
//枚举极大独立集，下标0开始，自环表示这个点必然不选
void BronKerbosch(const vector<ULL>&g,ULL cur,ULL allow,ULL forbid){
  if(allow==0&&forbid==0){cout<<cur<<endl;return;}
  if(allow==0)return false;
  int pivot=ctz(allow|forbid);
  ULL z=allow&~g[pivot];
  for(size_t u=ctz(z);u<g.size();u+=ctz(z>>(u+1))+1){
    BronKerbosch(g,cur|(1ULL<<u),allow&g[u],forbid&g[u]);
    allow^=1ULL<<u; forbid|=1ULL<<u;
  }
  return false;
}
void max_clique(){
  vector<ULL>g(n,0);
  for(int i=0;i<n;++i)g[i]=(1ULL<<n)-1-(1ULL<<i);
  for(int i=0;i<n;++i)for(auto &j:G[i])g[i]^=1ULL<<j;
  BronKerbosch(g,0,(1ULL<<n)-1,0);
}
//数组版本，极大团计数，下标1开始
int G[MAXN][MAXN],all[MAXN][MAXN],n,m;
int S,some[MAXN][MAXN],none[MAXN][MAXN];
void dfs(int d,int an,int sn,int nn){
  S+=sn==0&&nn==0;
  int u=sn>0?some[d][0]:none[d][0];
  for(int i=0;i<sn;++i){
    int v=some[d][i];if(G[u][v])continue;
    int tsn=0,tnn=0;
    for(int j=0;j<an;++j)all[d+1][j]=all[d][j];
    all[d+1][an]=v;
    for(int j=0;j<sn;++j)if(G[v][some[d][j]])some[d+1][tsn++]=some[d][j];
    for(int j=0;j<nn;++j)if(G[v][none[d][j]])none[d+1][tnn++]=none[d][j];
    dfs(d+1,an+1,tsn,tnn);
    some[d][i]=0;none[d][nn++]=v;
  }
}
void solve(){
  S=0;
  for(int i=0;i<n;++i)some[0][i]=i+1;
  dfs(0,0,n,0);
}
```

5.9 最大独立集的随机算法

```
int T,n,i,k,m,x,y,ans,q[N],t,loc[N],del[N],have;
int main(){
  for(T=1000;T;T--){
    for(have=0,t=n,i=1;i<=n;i++)q[i]=loc[i]=i,del[i]=0;
    while(t){
      y=q[x=std::rand()%t+1],loc[q[x]=q[t--]]=x,have++;
      for(p=g[y];p;p=p->nxt)if(!del[p->v])del[p->v]=1,x=loc[p->v],loc[q[x]=q[t--]]=x;
    }
    if(have>ans)ans=have;
  }
  printf("%d",ans);
}
```

5.10 差分约束系统

a 向 b 连一条权值为 c 的有向边表示 $b-a \le c$，用 SPFA 判断是否存在负环，存在即无解。

5.11 点覆盖、独立集、最大团、路径覆盖

二分图最小路径覆盖 = 最大独立集 = 总节点数 -最大匹配数，最小点覆盖 = 最大匹配数。任意图中，最大独立集 + 最小点覆盖集 =V，最大团 = 补图的最大独立集。

5.12 匈牙利算法

```
bool find(int x){
  for(int i=g[x];i;i=nxt[i])if(!b[v[i]]){
    b[v[i]]=1;
    if(!f[v[i]]||find(f[v[i]]))return f[v[i]]=x,1;
  }
  return 0;
}
int main(){
  for(j=1;j<=m;j++)f[j]=0;
  for(i=1;i<=n;i++){
    for(j=1;j<=m;j++)b[j]=0;
    if(find(i))ans++;
  }
}
```

5.12.1 二分图字典序最小匹配

先用匈牙利算法求出最大匹配，然后从小到大考虑每个点，尝试与更小的点匹配，且不破坏前面的匹配边。

```
bool find(int x){
  for(int i=1;i<=n;i++)if(!v[i]&&g[x][i]){
    v[i]=1;
    if(!f[i]||find(f[i]))return f[i]=x,1;
  }
  return 0;
```

```
}
bool find2(int x){
  for(int i=1;i<=n;i++)if(!v[i]&&g[x][i]){
    v[i]=1;
    if(f[i]==S||f[i]>S&&find2(f[i]))return f[i]=x,1;
  }
  return 0;
}
int main(){
  for(i=1;i<=n;i++){
    for(j=1;j<=n;j++)v[j]=0;
    if(!find(i))return puts("NIE"),0;
  }
  puts("TAK");
  for(S=1;S<=n;S++){
    for(j=1;j<=n;j++)v[j]=0;
    find2(S);
    for(j=1;j<=n;j++)if(f[j]==S)printf("%d\n",j);
  }
}
```

5.13 Hall 定理

二分图中的两部分顶点组成的集合分别为 X, Y，则有一组无公共点的边，一端恰好为组成 X 的点的充分必要条件是：X 中的任意 k 个点至少与 Y 中的 k 个点相邻。对于区间图只需要考虑极端情况，线段树维护。

5.14 网络流

5.14.1 ISAP 求最大流

```
const int N=410,inf=~0U>>2;
struct edge{int t,f;edge*nxt,*pair;}*g[N],*d[N],pool[M],*cur=pool;
int n,m,i,S,T,h[N],gap[N],maxflow;
void add(int s,int t,int f){
  edge*p=cur++;p->t=t;p->f=f;p->nxt=g[s];g[s]=p;
  p=cur++;p->t=s;p->f=0;p->nxt=g[t];g[t]=p;
  g[s]->pair=g[t];g[t]->pair=g[s];
}
int sap(int v,int flow){
  if(v==T)return flow;
  int rec=0;
  for(edge*p=d[v];p;p=p->nxt)if(h[v]==h[p->t]+1&&p->f){
    int ret=sap(p->t,min(flow-rec,p->f));
    p->f-=ret;p->pair->f+=ret;d[v]=p;
    if((rec+=ret)==flow)return flow;
  }
  if(!(--gap[h[v]]))h[S]=T;
  gap[++h[v]]++;d[v]=g[v];
  return rec;
}
int main(){
  S=n+1,T=S+1;
```

```
  for(cur=pool,i=1;i<=T;i++)g[i]=d[i]=NULL,h[i]=gap[i]=0;
  addedge;
  for(gap[maxflow=0]=T,i=1;i<=T;i++)d[i]=g[i];
  while(h[S]<T)maxflow+=sap(S,inf);
}
```

5.14.2 上下界有源汇网络流

T 向 S 连容量为正无穷的边，将有源汇转化为无源汇。

每条边容量减去下界，设 $in[i]$ 表示流入 i 的下界之和减去流出 i 的下界之和。

新建超级源汇 SS,TT，对于 $in[i]>0$ 的点，SS 向 i 连容量为 $in[i]$ 的边。对于 $in[i]<0$ 的点，i 向 TT 连容量为 $-in[i]$ 的边。

求出以 SS,TT 为源汇的最大流，如果等于 $\sum in[i](in[i]>0)$，则存在可行流。再求出以 S,T 为源汇的最大流即为最大流。

费用流：建完图后等价于求以 SS,TT 为源汇的的费用流。

上下界费用流，要先把下界的费用加入答案。

```
const int N=110,inf=~0U>>2;
int n,m,i,j,w,t,S,T,SS,TT,h[N],gap[N],maxflow,sum,in[N],id[N];
struct edge{int t,f;edge*nxt,*pair;}*g[N],*d[N];
void add(int s,int t,int f){
  edge *p=new(edge);p->t=t;p->f=f;p->nxt=g[s];g[s]=p;
  p=new(edge);p->t=s;p->f=0;p->nxt=g[t];
  g[t]=p;g[s]->pair=g[t];g[t]->pair=g[s];
}
int sap(int v,int flow,int S,int T){
  if(v==T)return flow;
  int rec=0;
  for(edge *p=d[v];p;p=p->nxt)if(h[v]==h[p->t]+1&&p->f){
    int ret=sap(p->t,min(flow-rec,p->f),S,T);
    p->f-=ret;p->pair->f+=ret;d[v]=p;
    if((rec+=ret)==flow)return flow;
  }
  d[v]=g[v];
  if(!(--gap[h[v]]))h[S]=TT;
  gap[++h[v]]++;
  return rec;
}
int main(){
  scanf("%d%d%d%d",&n,&m,&S,&T);
  for(i=1;i<=n;i++)id[i]=i;
  swap(id[S],id[n-1]),swap(id[T],id[n]);
  S=n-1,T=S+1,SS=T+1,TT=SS+1;add(T,S,inf);
  while(m--){
    scanf("%d%d%d%d",&i,&j,&w,&t);
    i=id[i],j=id[j];
    if(t)in[i]-=w,in[j]+=w;else add(i,j,w);
  }
  for(i=1;i<=TT;i++)if(in[i]>0)sum+=in[i],add(SS,i,in[i]);else add(i,TT,-in[i]);
  for(gap[i=0]=TT;i++<TT;)d[i]=g[i];
  while(h[SS]<TT)maxflow+=sap(SS,inf,SS,TT);
  if(sum!=maxflow)return puts("-1"),0;
```

```
  for(maxflow=i=0;i<=TT;i++)d[i]=g[i],h[i]=gap[i]=0;
  gap[0]=TT;
  while(h[S]<TT)maxflow+=sap(S,inf,S,T);
  printf("%d",maxflow);
}
```

5.14.3 费用流

最小费用流：若 $d[T]$ 为负则继续增广。

最小费用最大流：若 $d[T]$ 不为 inf 则继续增广。

```
const int inf=~0U>>2,N=210,M=20000;
int n,m,i,tmp,ans;
int u[M],v[M],c[M],co[M],nxt[M],t=1,S,T,l,r,q[M],g[N],f[N],d[N];bool in[N];
void add(int x,int y,int z,int zo){
  u[++t]=x;v[t]=y;c[t]=z;co[t]=zo;nxt[t]=g[x];g[x]=t;
  u[++t]=y;v[t]=x;c[t]=0;co[t]=-zo;nxt[t]=g[y];g[y]=t;
}
bool spfa(){
  int x,i;
  for(i=1;i<=T;i++)d[i]=inf,in[i]=0;
  d[S]=0;in[S]=1;l=r=M>>1;q[l]=S;
  while(l<=r){
    int x=q[l++];
    if(x==T)continue;
    for(i=g[x];i;i=nxt[i])if(c[i]&&co[i]+d[x]<d[v[i]]){
      d[v[i]]=co[i]+d[x];f[v[i]]=i;
      if(!in[v[i]]){
        in[v[i]]=1;
        if(d[v[i]]<d[q[l]])q[--l]=v[i];else q[++r]=v[i];
      }
    }
    in[x]=0;
  }
  return d[T]<inf;
}
int main(){
  S=0,T=n+1;
  while(spfa()){
    for(tmp=inf,i=T;i!=S;i=u[f[i]])if(tmp>c[f[i]])tmp=c[f[i]];
    for(ans+=d[i=T]*tmp;i!=S;i=u[f[i]])c[f[i]]-=tmp,c[f[i]^1]+=tmp;
  }
  printf("%d",ans);
}
```

5.14.4 混合图欧拉回路判定

首先给无向边随便定一个方向，设 $deg[x]$ 为 x 连出去的边数 $-$ 连入 x 的边数。

若存在 $deg[x]$ 为奇数，或者图不连通，则无解。否则建立源点 S，汇点 T。

对于一个点 x，若 $deg[x]>0$，则 S 向 x 连边，容量 $\frac{deg[x]}{2}$；若 $deg[x]<0$，则 x 向 T 连边，容量 $-\frac{deg[x]}{2}$。

对于一条定了向的无向边 x->y，x 向 y 连边，容量 1，求出最大流，若与 S 和 T 连的每条边都满流，则有解。

5.14.5 线性规划转费用流

首先添加松弛变量，将不等号都变为等号。分别用下一个式子减去上一个式子，如果每个变量只出现了两次且符号一正一负，那么可以转化为费用流。对于每个式子建立一个点，那么每个变量对应一条边，从一个点流出，向另一个点流入。这样，对于等式右边的常数 C，如果是正的，对应从源点向该点连一条流量 C，费用 0 的边；如果是负的对应从该点向汇点连一条流量 $-C$，费用 0 的边。对于每个变量，从它系数为正的式子向系数为负的式子连一条容量为 inf，费用为它在目标函数里系数的边。这样网络流模型就构造完毕了。

5.15 最小树形图

```
const int N=10050,M=50050,inf=0x7fffffff;
struct DMST{
  int n,size,pre[N],id[N],vis[N],in[N];
  struct EDGE{
    int u,v,cost;
    EDGE(){}
    EDGE(int a,int b,int c):u(a),v(b),cost(c){}
  }E[M];
  void init(int _n){n=_n,size=0;}
  void add(int u,int v,int w){E[size++]=EDGE(u,v,w);}
  int dmst(int root){
    int u,v,cnt,ret=0;
    while(1){
      for(int i=0;i<n;i++)in[i]=inf;
      for(int i=0;i<size;i++){
        u=E[i].u,v=E[i].v;
        if(E[i].cost<in[v]&&u!=v){
          pre[v]=u,in[v]=E[i].cost;
          if(u==root)ROOT=i;
        }
      }
      for(int i=0;i<n;i++)if(i!=root&&in[i]==inf)return -1;
      cnt=in[root]=0;
      for(i=0;i<n;i++)id[i]=vis[i]=-1;
      for(int i=0;i<n;i++){
        ret+=in[i],v=i;
        while(vis[v]!=i&&id[v]==-1&&v!=root)vis[v]=i,v=pre[v];
        if(v!=root&&id[v]==-1){
          for(u=pre[v];u!=v;u=pre[u])id[u]=cnt;
          id[v]=cnt++;
        }
      }
      if(!cnt)break;
      for(int i=0;i<n;i++)if(id[i]==-1)id[i]=cnt++;
      for(int i=0;v=E[i].v,i<size;i++){
        E[i].u=id[E[i].u],E[i].v=id[E[i].v];
        if(E[i].u!=E[i].v)E[i].cost-=in[v];
      }
```

```cpp
      n=cnt,root=id[root];
    }
    return ret;
  }
};
void variable(int &cost,int &root){//Variable Root
  for(int i=0;i<n;i++)G.add(st,i,tot);//st=n,tot=sum of Edge Wight+1
  int ans=G.dmst(st);
  if(ans==-1||ans-tot>tot)return;//No solution
  cost=ans-tot,root=ROOT-m;
}
```

5.15.1 输出方案

多源最小树形图, Edmonds 算法, 邻接阵形式, 复杂度 $O(n^3)$。

返回最小树形图的边权和, 构造失败返回负值。

传入图的大小 n 和邻接阵 $G[][]$, 不相邻点边权 inf, 下标 0 到 $n-1$。

可更改边权的类型, $pre[]$ 返回树的构造, 用父结点表示。

传入时 $pre[]$ 数组清零, 用 -1 标出可能的源点。

```cpp
const int N=1010,inf=1e9;
int edmonds(int n,int G[][N],int pre[]){
  static int vis[N][N],l[N],p[N];
  int m=n,cnt,ret=0,i,j,k;
  for(i=0;i<n;++i)l[i]=i;
  do{
    memset(vis,0,sizeof(vis));
    memset(p,0xff,sizeof(p));
    cnt=m;
    for(i=0;i<m;++i)vis[i][i]=1;
    for(i=0;i<cnt;++i)if(l[i]==i&&pre[i]!=-1){
      for(j=0;j<m;++j)
        if(l[j]==j&&i!=j&&G[j][i]<inf&&(p[i]==-1||G[j][i]<G[p[i]][i]))p[i]=j;
      if((pre[i]=p[i])==-1)return -1;//no solution
      if(vis[i][p[i]]){
        for(j=0;j<=m;++j)G[j][m]=G[m][j]=inf;
        for(k=i;l[k]!=m;l[k]=m,k=p[k])for(j=0;j<m;++j)if(l[j]==j){
          if(G[j][k]-G[p[k]][k]<G[j][m])G[j][m]=G[j][k]-G[p[k]][k];
          if(G[k][j]<G[m][j])G[m][j]=G[k][j];
        }
        vis[m][m]=1;l[m]=m;m++;
      }
      for(j=0;j<m;++j)if(vis[i][j])for(k=p[i];k!=-1&&l[k]==k; k=p[k])vis[k][j]=1;
    }
  }while(cnt<m);
  for(;m-->n;pre[k]=pre[m])for(i=0;i<m;++i)if(l[i]==m){
    for(j=0;j<m;++j)if(pre[j]==m&&G[i][j]==G[m][j])pre[j]=i;
    if(G[pre[m]][m]==G[pre[m]][i]-G[pre[i]][i])k=i;
  }
  for(i=0;i<n;++i)if(pre[i]!=-1)ret+=G[pre[i]][i];
  return ret;
}
```

5.15.2 左偏树优化

根为 1 的最小树形图。时间复杂度 $O(m\log n)$。

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<algorithm>
#include<cmath>
#include<queue>
#include<vector>
#define fortodo(i,f,t) for(i=(f);i<=(t);i++)
using namespace std;
typedef long long ll;
const int NEED_WAYS=1;
struct Info{
  int sig;
  ll cost,used;
  Info*compn[2];
  Info(){}
  Info(ll cost){
    sig=0;
    compn[0]=compn[1]=NULL;
    this->cost=cost;
    used=0;
  }
  Info(bool isBothPosi,Info*compn0,Info*compn1){
    sig=isBothPosi?1:-1;
    compn[0]=compn0;
    compn[1]=compn1;
    cost=compn[0]->cost+sig*compn[1]->cost;
    used=0;
  }
  void set(bool isBothPosi,Info*compn0,Info*compn1){
    sig=isBothPosi?1:-1;
    compn[0]=compn0;
    compn[1]=compn1;
    cost=compn[0]->cost+sig*compn[1]->cost;
    used=0;
  }
  void pick(){used++;}
  void inhr(){
    if(sig){
      compn[0]->used+=used;
      compn[1]->used+=sig*used;
      used=0;
    }
  }
}pool[10000000],*wkcpool;
Info emptyInfo(0);
vector<Info*>derived;
Info*Add(Info*a,Info*b){
  if(a==&emptyInfo)return b;
  if(b==&emptyInfo)return a;
  Info*nxt=wkcpool++;
  nxt->set(1,a,b);
```

```
  derived.push_back(nxt);
  return nxt;
}
Info*Sub(Info*a,Info*b){
  if(b==&emptyInfo)return a;
  Info*nxt=wkcpool++;
  nxt->set(0,a,b);
  derived.push_back(nxt);
  return nxt;
}
struct Leftist{
  Leftist*s[2];
  pair<Info*,ll>val;
  Info*Mask;
  ll Dist;
  Leftist(pair<Info*,ll>nval=pair<Info*,ll>(&emptyInfo,0)){
    s[0]=s[1]=NULL;
    Dist=1;
    val=nval;
    Mask=&emptyInfo;
  }
  ll Key(){return val.first->cost;}
};
ll dist(Leftist*cur){return cur?cur->Dist:0;}
void Push(Leftist*cur){
  if(cur->s[0])cur->s[0]->Mask=Add(cur->s[0]->Mask,cur->Mask);
  if(cur->s[1])cur->s[1]->Mask=Add(cur->s[1]->Mask,cur->Mask);
  cur->val.first=Sub(cur->val.first,cur->Mask);
  cur->Mask=&emptyInfo;
}
Leftist*Merge(Leftist*l,Leftist*r){
  if(l==NULL)return r;
  if(r==NULL)return l;
  Push(l);Push(r);
  if(l->Key()>r->Key())swap(l,r);
  l->s[1]=Merge(l->s[1],r);
  if(dist(l->s[0])<dist(l->s[1]))swap(l->s[0],l->s[1]);
  l->Dist=dist(l->s[1])+1;
  return l;
}
pair<Info*,ll>Extract(Leftist*&cur){
  Push(cur);
  pair<Info*,ll>ret=cur->val;
  cur=Merge(cur->s[0],cur->s[1]);
  return ret;
}
const int MAXN=1000010,MAXM=1001010;
int N,M;
Leftist*heap[MAXN],listpool[MAXM],*curlist;
struct UFS{
  int F[MAXN];
  void Init(){int i;fortodo(i,1,N)F[i]=i;}
  int Find(int x){return F[x]=F[x]==x?x:Find(F[x]);}
  void Union(int x,int y){F[Find(y)]=Find(x);}
  bool Cnx(int x,int y){return Find(x)==Find(y);}
};
UFS weak,strong;
```

```
priority_queue<int,vector<int>,greater<int> >activeStrong;
bool inside[MAXN];
inline void actins(int x){
  if(inside[x])return;
  inside[x]=1;
  activeStrong.push(x);
}
void Join(int u,int v){
  strong.Union(u,v);
  heap[u]=Merge(heap[u],heap[v]);
  heap[v]=NULL;
  actins(u);
}
Info baseInfo[MAXM],*prevCost[MAXN];
int pre[MAXN];
ll Ans;
bool Connected(){
  int i;
  fortodo(i,1,N)if(!weak.Cnx(1,i))return 0;
  return 1;
}
inline void addedge(int u,int v,ll w){
  M++;
  baseInfo[M]=Info(w);
  if(v==1)return;
  Leftist*p=curlist++;
  p->s[0]=p->s[1]=NULL;
  p->Dist=1;
  p->val=make_pair(&baseInfo[M],u);
  p->Mask=&emptyInfo;
  heap[v]=Merge(heap[v],p);
}
void solve(){
  int i,wkc;
  scanf("%d%d",&N,&wkc);
  wkcpool=pool;
  M=0;
  derived.clear();
  curlist=listpool;
  fortodo(i,1,N){
    heap[i]=NULL;
    pre[i]=0;
    prevCost[i]=NULL;
    inside[i]=0;
  }
  while(wkc--){
    int x,y,z;
    scanf("%d%d%d",&x,&y,&z);
    addedge(x,y,z);
  }
  while(!activeStrong.empty())activeStrong.pop();
  fortodo(i,2,N)actins(i);
  weak.Init();
  strong.Init();
  Ans=0;
  while(!activeStrong.empty()){
    int s0=activeStrong.top();
```

```
    if(heap[s0]==NULL){
      activeStrong.pop();
      continue;
    }
    pair<Info*,ll>pdi=Extract(heap[s0]);
    int S=strong.Find(pdi.second);
    if(S==s0)continue;
    if(!weak.Cnx(pdi.second,s0)){
      pre[s0]=pdi.second;
      prevCost[s0]=pdi.first;
      Ans+=pdi.first->cost;
      pdi.first->pick();
      weak.Union(pdi.second,s0);
      activeStrong.pop();
      continue;
    }
    vector<int>LIZ;
    LIZ.clear();
    LIZ.push_back(strong.Find(pdi.second));
    while(LIZ.back()!=s0)LIZ.push_back(strong.Find(pre[LIZ.back()]));
    Ans+=pdi.first->cost;
    pdi.first->pick();
    if(heap[s0])heap[s0]->Mask=Add(heap[s0]->Mask,pdi.first);
    int SZ=LIZ.size();
    fortodo(i,0,SZ-2)if(heap[LIZ[i]])
      heap[LIZ[i]]->Mask=Add(heap[LIZ[i]]->Mask,prevCost[LIZ[i]]);
    fortodo(i,0,SZ-2)Join(s0, LIZ[i]);
  }
  if(!Connected()){
    puts("NO");
    return;
  }
  puts("YES");
  printf("%lld\n",Ans);
  if(NEED_WAYS){
    vector<int>VI;
    VI.clear();
    for(vector<Info*>::reverse_iterator rit=derived.rbegin();rit!=derived.rend();rit++)
      (*rit)->inhr();
    int cnt=0;
    fortodo(i,1,M)if(baseInfo[i].used&&baseInfo[i].cost){
      VI.push_back(i);
      cnt++;
    }
    fortodo(i,0,cnt-1)printf("%d\n",VI[i]);//输入的第几条边
  }
  for(vector<Info*>::reverse_iterator rit=derived.rbegin();rit!=derived.rend();rit++)
    delete*rit;
}
```

5.16 构造双连通图

一个有桥的连通图，如何把它通过加边变成边双连通图？方法为首先求出所有的桥，然后删除这些桥边，剩下的每个连通块都是一个双连通子图。把每个双连通子图收缩为一个顶点，

再把桥边加回来，最后的这个图一定是一棵树，边连通度为 1。

统计出树中度为 1 的节点的个数，即为叶节点的个数，记为 $leaf$。则至少在树上添加 $\frac{leaf+1}{2}$ 条边，就能使树达到边双连通。具体方法为，首先把两个最近公共祖先最远的两个叶节点之间连接一条边，这样可以把这两个点到祖先的路径上所有点收缩到一起，因为一个形成的环一定是双连通的。然后再找两个最近公共祖先最远的两个叶节点，这样一对一对找完，恰好是 $\frac{leaf+1}{2}$ 次，把所有点收缩到了一起。

5.17 一般图最大匹配

用带花树求编号从 0 开始的 n 个点的图的最大匹配，时间复杂度 $O(n^3)$。

$mate[]$ 为配偶结点的编号，没有匹配上的点为 -1。

传入结点个数 n 及各结点的出边表 $G[]$, 返回匹配点对的数量 ret。

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
#include<queue>
using namespace std;
const int N=510;
int n,m,x,y;vector<int>g[N];
namespace Blossom{
int mate[N],n,ret,nxt[N],f[N],mark[N],vis[N],t;queue<int>Q;
int F(int x){return x==f[x]?x:f[x]=F(f[x]);}
void merge(int a,int b){f[F(a)]=F(b);}
int lca(int x,int y){
  for(t++;;swap(x,y))if(~x){
    if(vis[x=F(x)]==t)return x;vis[x]=t;
    x=mate[x]!=-1?nxt[mate[x]]:-1;
  }
}
void group(int a,int p){
  for(int b,c;a!=p;merge(a,b),merge(b,c),a=c){
    b=mate[a],c=nxt[b];
    if(F(c)!=p)nxt[c]=b;
    if(mark[b]==2)mark[b]=1,Q.push(b);
    if(mark[c]==2)mark[c]=1,Q.push(c);
  }
}
void aug(int s,const vector<int>G[]){
  for(int i=0;i<n;++i)nxt[i]=vis[i]=-1,f[i]=i,mark[i]=0;
  while(!Q.empty())Q.pop();Q.push(s);mark[s]=1;
  while(mate[s]==-1&&!Q.empty()){
    int x=Q.front();Q.pop();
    for(int i=0,y;i<(int)G[x].size();++i){
      if((y=G[x][i])!=mate[x]&&F(x)!=F(y)&&mark[y]!=2){
        if(mark[y]==1){
          int p=lca(x,y);
          if(F(x)!=p)nxt[x]=y;
          if(F(y)!=p)nxt[y]=x;
          group(x,p),group(y,p);
        }else if(mate[y]==-1){
          nxt[y]=x;
```

```
        for(int j=y,k,l;~j;j=l)k=nxt[j],l=mate[k],mate[j]=k,mate[k]=j;
        break;
      }else nxt[y]=x,Q.push(mate[y]),mark[mate[y]]=1,mark[y]=2;
    }
  }
 }
}
int solve(int _n,const vector<int>G[]){
  n=_n;memset(mate,-1,sizeof mate);
  for(int i=t=0;i<n;++i)if(mate[i]==-1)aug(i,G);
  for(int i=ret=0;i<n;++i)ret+=mate[i]>i;
  printf("%d\n",ret);
  for(int i=0;i<n;i++)printf("%d ",mate[i]+1);
  return ret;
}
}
int main(){
  scanf("%d%d",&n,&m);
  while(m--)scanf("%d%d",&x,&y),x--,y--,g[x].push_back(y),g[y].push_back(x);
  Blossom::solve(n,g);
}
```

5.18 图的绝对中心

求图的绝对中心, $g[][]$ 为邻接矩阵, 把没有的边权赋为 inf。返回一个 pair, 表示绝对中心在这条边 (s_1, s_2) 上，ds_1, ds_2 记录 s_1 和 s_2 距离绝对中心的距离, 按需返回。

最小直径生成树就是求出每个点到绝对中心的距离，然后找最短路树。

```
const int MAXN = 1000, inf = 1e9;
void floyd(int n, int g[][MAXN], int d[][MAXN]) {
  for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n; ++j) {
      d[i][j] = g[i][j];
    }
  }
  for (int k = 0; k < n; ++k) {
    for (int i = 0; i < n; ++i) {
      for (int j = 0; j < n; ++j) {
        d[i][j] = std::min(d[i][j], d[i][k] + d[k][j]);
      }
    }
  }
}
std::pair<int, int> KarivHakimi(int n, int g[][MAXN]) {
  static int rk[MAXN][MAXN], d[MAXN][MAXN];
  double ds1 = 0, ds2 = 0;
  floyd(n, g, d);
  for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n; ++j) rk[i][j] = j;
    std::sort(rk[i], rk[i] + n, [&](int a, int b) {
      return d[i][a] < d[i][b];
    });
  }
  int ret = inf, s1 = -1, s2 = -1;
```

```
  for (int u = 0; u < n; ++u) {
    if (d[u][rk[u][n - 1]] * 2 < ret) {
      ret = d[u][rk[u][n - 1]] * 2;
      s1 = s2 = u;
      ds1 = ds2 = 0;
    }
    for (int v = 0; v < n; ++v) {
      if (g[u][v] == inf) continue;
      for (int k = n - 1, i = n - 2; i >= 0; --i) {
        if (d[v][rk[u][i]] > d[v][rk[u][k]]) {
          int tmp = d[u][rk[u][i]] + d[v][rk[u][k]] + g[u][v];
          if (tmp < ret) {
            ret = tmp;
            s1 = u, s2 = v;
            ds1 = 0.5 * tmp - d[u][rk[u][i]];
            ds2 = g[u][v] - ds1;
          }
          k = i;
        }
      }
    }
  }
  return std::make_pair(s1, s2);
}
```

5.19 Hopcroft

```
#include<iostream>
using namespace std;
namespace Hopcroft{
const int N=100010,M=200010;//最大的单侧点个数
int cnt,pos[N],neg[N];//pos[]为左侧点所匹配到的右侧点编号，从0开始
//neg[]反之，没有匹配到对应的点则为-1
//传入左侧点个数 n 和左侧点至右侧点的边表e[]，返回匹配点对的数量cnt
//复杂度O(sqrt(n)*m)
int lx[N],ly[N],q[N],n,g[N],v[M],nxt[M],ed;
void init(int _n){n=_n;for(int i=ed=0;i<n;i++)g[i]=0;}
void add(int x,int y){v[++ed]=y;nxt[ed]=g[x];g[x]=ed;}
bool dfs(int x){
  int c=lx[x]+1,y=lx[x]=-1;
  for(int i=g[x];i;i=nxt[i])if(ly[y=v[i]]==c){
    ly[y]=-1;
    if(~neg[y]&&!dfs(neg[y]))continue;
    pos[neg[y]=x]=y;
    return 1;
  }
  return 0;
}
int work(){
  int i,x,y;
  fill(pos,pos+n,-1);fill(neg,neg+n,-1);
  for(x=cnt=0;x<n;x++)for(i=g[x];i;i=nxt[i]){
    if(~neg[y=v[i]])continue;
    pos[neg[y]=x]=y;
    cnt++;
```

```
      break;
  }
  while(1){
    int h=0,t=0,ok=0;
    fill(lx,lx+n,-1),fill(ly,ly+n,-1);
    for(x=0;x<n;x++)if(pos[x]<0)lx[q[t++]=x]=0;
    while(h!=t){
      for(i=g[x=q[h++]];i;i=nxt[i]){
        if(~ly[y=v[i]])continue;
        ly[y]=1+lx[x];
        if(~neg[y]&&~lx[neg[y]])continue;
        if(~neg[y])lx[q[t++]=neg[y]]=1+ly[y];else ok=1;
      }
    }
    if(!ok)return cnt;
    for(x=0;x<n;x++)if(pos[x]<0&&dfs(x))cnt++;
  }
}
}
```

5.20 KM

左边 nl 个点 $1..nl$，右边 nr 个点 $1..nr$，两点间代价为 $add(x,y,z)$ 表示左 x，右 y。
最后输出 ans 为最大费用最大流，$lk[i]$ 表示左边第 i 个点匹配右边哪个，0 表示不匹配。
最大费用流时，NOT=0；最大费用最大流时，NOT=-1LL*N*M。

```
const int N=405,M=1000000010;
const ll NOT=-1LL*N*M,INF=3LL*N*M;
int n,nl,nr,m,z,py,x,y,i,j,p,lk[N],pre[N];
bool vy[N];
ll lx[N],ly[N],d,w[N][N],slk[N],ans;
void add(int x,int y,ll z){if(w[y][x]<z)w[y][x]=z;}
int main(){
  scanf("%d%d%d",&nl,&nr,&m);
  n=max(nl,nr);
  for(i=0;i<=n;i++)lk[i]=pre[i]=lx[i]=ly[i]=slk[i]=0;
  for(i=0;i<=n;i++)for(j=0;j<=n;j++)w[i][j]=NOT;
  while(m--)scanf("%d%d%d",&x,&y,&z),add(x,y,z);
  for(i=1;i<=n;i++)for(j=1;j<=n;j++)lx[i]=max(lx[i],w[i][j]);
  for(i=1;i<=n;i++){
    for(j=1;j<=n;j++)slk[j]=INF,vy[j]=0;
    for(lk[py=0]=i;lk[py];py=p){
      vy[py]=1;d=INF;x=lk[py];
      for(y=1;y<=n;y++)if(!vy[y]){
        if(lx[x]+ly[y]-w[x][y]<slk[y])slk[y]=lx[x]+ly[y]-w[x][y],pre[y]=py;
        if(slk[y]<d)d=slk[y],p=y;
      }
      for(y=0;y<=n;y++)if(vy[y])lx[lk[y]]-=d,ly[y]+=d;else slk[y]-=d;
    }
    for(;py;py=pre[py])lk[py]=lk[pre[py]];
  }
  for(ans=0,i=1;i<=n;i++){
    ans+=lx[i]+ly[i];
    if(w[lk[i]][i]==NOT)ans-=NOT;
```

```
  }
  printf(”%lld\n”,ans);
  for(i=1;i<=nl;i++)printf(”%d ”,w[lk[i]][i]!=NOT?lk[i]:0);
}
```

5.21 强连通竞赛图哈密顿回路

给定一个 N 个点的竞赛图，对于每个点输出一条以它为起点的最长简单路径，时间复杂度 $O(N^2)$。

```
#include<cstdio>
#define N 2010
int n,i,j,x,q[N],h,t,f[N],nxt[N],d[N],rk[N],go[N];bool g[N][N],v[N],G[N][N];
namespace Hamiltonian{
int n,m,i,j,k,o,a[N],nxt[N],q[N],tmp[N];bool g[N][N];
void init(){n=0;}
void add(int x){a[++n]=x;}
void solve(){
  for(i=1;i<=n;i++)for(j=1;j<=n;j++)g[i][j]=::g[a[i]][a[j]];
  for(i=1;i<=n;i++)nxt[i]=0;
  for(o=i=1;i<=n;i++)if(i!=o){
    if(g[i][o])nxt[i]=o,o=i;
    else{
      for(j=o;nxt[j]&&g[nxt[j]][i];j=nxt[j]);
      nxt[i]=nxt[j];
      nxt[j]=i;
    }
  }
  for(i=1,j=o;i<=n;i++,j=nxt[j])q[i]=j;
  for(j=n;j>1&&!g[q[j]][q[1]];j--);
  while(j<n){
    for(i=1;i<=j;i++)if(g[q[j+1]][q[i]])break;
    if(i<=j){
      for(m=0,k=i;k<=j;k++)tmp[++m]=q[k];
      for(k=1;k<i;k++)tmp[++m]=q[k];
      for(k=1;k<=m;k++)q[k]=tmp[k];
      j++;
      continue;
    }
    for(i=j+2;;i++){
      for(k=1;k<=j;k++)if(g[q[i]][q[k]])break;
      if(k<=j)break;
    }
    for(m=0,o=k;o<=j;o++)tmp[++m]=q[o];
    for(o=1;o<k;o++)tmp[++m]=q[o];
    for(o=1;o<=m;o++)q[o]=tmp[o];
    j=i;
  }
  for(i=1;i<n;i++)::nxt[a[q[i]]]=a[q[i+1]];
  ::nxt[a[q[n]]]=a[q[1]];
}
}
void dfs1(int x){
  v[x]=1;
  for(int i=1;i<=n;i++)if(!v[i]&&g[x][i])dfs1(i);
```

```
  q[++t]=x;
}
void dfs2(int x,int y){
  v[x]=0,f[x]=y;
  Hamiltonian::add(x);
  for(int i=1;i<=n;i++)if(v[i]&&g[i][x])dfs2(i,y);
}
inline void work(int x){
  int i;
  for(t=0,i=1;i<=n;i++)v[i]=0;
  while(x){
    while(!v[x])v[q[++t]=x]=1,x=nxt[x];
    x=go[x];
  }
  write(t);
  for(i=1;i<=t;i++)*ou++=' ',write(q[i]);
  *ou++='\n';
}
int main(){
  read(n);
  for(i=1;i<=n;i++)for(j=1;j<=n;j++)read(g[i][j]);
  for(i=1;i<=n;i++)if(!v[i])dfs1(i);
  for(i=t;i;i--)if(v[q[i]]){
    Hamiltonian::init();
    dfs2(q[i],q[i]);
    Hamiltonian::solve();
  }
  for(i=1;i<=n;i++)for(j=1;j<=n;j++)if(f[i]!=f[j]&&g[i][j])G[f[i]][f[j]]=1;
  for(i=1;i<=n;i++)for(j=1;j<=n;j++)if(G[i][j])d[j]++;
  for(t=0,h=i=1;i<=n;i++)if(f[i]==i&&!d[i])q[++t]=i;
  while(h<=t)for(x=q[h++],i=1;i<=n;i++)if(G[x][i])if(!(--d[i]))q[++t]=i;
  for(i=1;i<=t;i++)rk[q[i]]=i;
  for(i=1;i<=n;i++)for(j=1;j<=n;j++)if(g[i][j]&&rk[f[j]]==rk[f[i]]+1){go[i]=j;break;}
  for(i=1;i<=n;i++)work(i);
}
```

5.22 最小割判定

判断边是否为某一最小割集的边：

在残余网络 (还有流量的边) 中求强连通分量，顶点不在同一 SCC 且满流的边。

判断边是否为全部最小割集的边：

在上一条的基础上，还要满足起点与 S 在同一 SCC，且终点与 T 在同一 SCC。

5.23 二分图匹配判定

5.23.1 关键点

关键点即一定在最大匹配中的点。

求出任意一个最大匹配，那么只需要考虑哪些匹配点不一定在。

假设是考虑左侧的点，右侧类似：

将匹配边从右往左，非匹配边从左到右，从左侧每个未匹配点开始 DFS 到的匹配点都不是关键点。

5.23.2 关键边

求出任意一个最大匹配，将匹配边从右到左，剩余边从左到右，求出 SCC。

对于一条边：

若它位于当前匹配中，那么若两端点位于同一 SCC，则是可能在，否则必定在。

若它不位于当前匹配中，那么若两端点位于同一 SCC，则是可能在，否则必定不在。

5.24 动态桥边个数

维护一个 n 个点的空无向图，m 次操作，每次翻转一条边的存在情况，统计每个时刻桥边个数。$O(m\log m)$。

```
typedef pair<int,int>P;
const int N=100010,M=100010,K=19;
int n,m,i,ans[M];
bool use[M],ex[M],cut[M],vis[N];
int g[N],v[M<<1],w[M<<1],nxt[M<<1],ed,f[N],dfn[N],low[N],num,from[N];
int G[N],V[N<<1],W[N<<1],NXT[N<<1],ED,id[N],sum[N],vip[N];
int O,ce,all,pos[M],q[K][M];
map<P,int>T;
struct E{
  int x,y,w;
  E(){}
  E(int _x,int _y,int _w){x=_x,y=_y,w=_w;}
}e[K][M];
inline int ask(int x,int y){
  if(x==y)return 0;
  if(x>y)swap(x,y);
  int&t=T[P(x,y)];
  if(!t)e[0][t=++ce]=E(x,y,0);
  return t;
}
inline void add(int x,int y,int z){v[++ed]=y;w[ed]=z;nxt[ed]=g[x];g[x]=ed;}
inline void ADD(int x,int y,int z){V[++ED]=y;W[ED]=z;NXT[ED]=G[x];G[x]=ED;}
void tarjan(int x){
  dfn[x]=low[x]=++num;
  for(int i=g[x];i;i=nxt[i])if(!dfn[v[i]]){
    f[v[i]]=w[i],tarjan(v[i]);
    if(low[x]>low[v[i]])low[x]=low[v[i]];
  }else if(f[x]!=w[i]&&low[x]>dfn[v[i]])low[x]=dfn[v[i]];
  if(f[x]&&low[x]==dfn[x])cut[f[x]]=1;
}
void dfs(int x,int y){
  from[x]=y;
  for(int i=g[x];i;i=nxt[i])if(!from[v[i]]&&!cut[w[i]])dfs(v[i],y);
}
inline void newedge(int x,int y,int w){all-=w;e[O][++ce]=E(x,y,w);}
void compress(int x,int y){
  int d=0;
  vis[x]=1;
  for(int i=G[x];i;i=NXT[i]){
    int u=V[i];
    if(u==y)continue;
    sum[u]=sum[x]+W[i];
```

```
    compress(u,x);
    if(!id[u])continue;
    d++;
    id[x]^=id[u];
  }
  if(d>1)vip[x]=1;
  if(vip[x]){
    for(int i=G[x];i;i=NXT[i]){
      int u=V[i];
      if(u==y)continue;
      int t=id[u];
      if(t)newedge(x,t,sum[t]-sum[x]);
    }
    id[x]=x;
  }
}
void solve(int o,int l,int r,int n,int m,int pre){
  O=o+1;
  int i;
  if(l==r){
    for(i=1;i<=m;i++)cut[i]=0;
    for(ed=0,i=1;i<=n;i++)g[i]=f[i]=dfn[i]=low[i]=from[i]=0;
    num=0;
    e[o][q[o][l]].w^=1;
    for(i=1;i<=m;i++)if(e[o][i].w){
      int x=e[o][i].x,y=e[o][i].y;
      add(x,y,i),add(y,x,i);
    }
    for(i=1;i<=n;i++)if(!dfn[i])tarjan(i);
    for(i=1;i<=m;i++)if(cut[i])pre+=e[o][i].w;
    e[o][q[o][l]].w^=1;
    ans[l]=pre;
    return;
  }
  for(i=1;i<=m;i++)use[i]=cut[i]=pos[i]=0;
  for(ed=0,i=1;i<=n;i++)g[i]=f[i]=dfn[i]=low[i]=from[i]=0;
  num=0;
  int cnt=0;
  for(i=l;i<=r;i++)use[q[o][i]]=1;
  for(i=1;i<=m;i++)if(!use[i]&&e[o][i].w){
    int x=e[o][i].x,y=e[o][i].y;
    add(x,y,i),add(y,x,i);
  }
  for(i=1;i<=n;i++)if(!dfn[i])tarjan(i);
  for(i=1;i<=n;i++)if(!from[i])dfs(i,++cnt);
  for(ED=0,i=1;i<=cnt;i++)vis[i]=vip[i]=G[i]=id[i]=sum[i]=0;
  ce=all=0;
  for(i=1;i<=m;i++)if(!use[i]&&e[o][i].w){
    int x=e[o][i].x,y=e[o][i].y;
    x=from[x],y=from[y];
    if(x==y)continue;
    ADD(x,y,e[o][i].w),ADD(y,x,e[o][i].w);
    all+=e[o][i].w;
  }
  for(i=l;i<=r;i++){
    int t=q[o][i];
    if(!t)continue;
```

```
    vip[from[e[o][t].x]]=vip[from[e[o][t].y]]=1;
  }
  for(i=1;i<=cnt;i++)if(vip[i]&&!vis[i])compress(i,0);
  int mid=(l+r)>>1,_ce=ce,cv=0;
  for(i=1;i<=cnt;i++)if(vip[i])vip[i]=++cv;
  pre+=all;
  for(i=1;i<=ce;i++)e[o+1][i].x=vip[e[o+1][i].x],e[o+1][i].y=vip[e[o+1][i].y];
  for(i=1;i<=m;i++)if(use[i]){
    int x=e[o][i].x,y=e[o][i].y;
    e[o+1][++ce]=E(vip[from[x]],vip[from[y]],e[o][i].w);
    pos[i]=ce;
  }
  for(i=l;i<=r;i++)q[o+1][i]=pos[q[o][i]];
  solve(o+1,l,mid,cv,ce,pre);
  ce=_ce;
  for(i=1;i<=m;i++)use[i]=0;
  for(i=l;i<=r;i++)use[q[o][i]]=1;
  for(i=1;i<=m;i++)if(use[i])ex[i]=e[o][i].w;
  for(i=l;i<=mid;i++)ex[q[o][i]]^=1;
  for(i=1;i<=m;i++)if(use[i])e[o+1][++ce].w=ex[i];
  solve(o+1,mid+1,r,cv,ce,pre);
}
int main(){
  scanf("%d%d",&n,&m);
  for(i=1;i<=m;i++){
    char s[9];
    int x,y;
    scanf("%s%d%d",s,&x,&y);
    q[0][i]=ask(x,y);
  }
  solve(0,1,m,n,ce,0);
  for(i=1;i<=m;i++)printf("%d\n",ans[i]);
}
```

5.25 平均数最小环

给定一个 n 个点 m 条边 (边权可能为负) 的有向图，$O(nm)$ 求边权平均数最小的环。

```
const int N=3005,M=10005;
const double inf=1e18;
int n,m,i,j,u[M],v[M];double w[M],f[N][N],ans=1e9,now,tmp;
inline void up(double&a,double b){a>b?(a=b):0;}
int main(){
  scanf("%d%d",&n,&m);
  for(i=1;i<=m;i++)scanf("%d%d%lf",&u[i],&v[i],&w[i]);
  for(i=1;i<=n;i++)for(j=1;j<=n;j++)f[i][j]=inf;
  for(i=0;i<n;i++)for(j=1;j<=m;j++)up(f[i+1][v[j]],f[i][u[j]]+w[j]);
  for(i=1;i<=n;i++)if(f[n][i]<inf/2){
    now=-1e9;
    for(j=0;j<n;j++)if(f[j][i]<inf/2){
      tmp=1.0*(f[n][i]-f[j][i])/(n-j);
      if(now<tmp)now=tmp;
    }
    up(ans,now);
  }
```

```
  printf("%.8f",ans);
}
```

5.26 团和独立集

将一个 n 个点 m 个点的简单无向图分成两个非空集合 A 和 B，要求 A 是团，B 是独立集，求方案数。

一个方案合法当且仅当团点数 $\times$(团点数 -1)$+$ 独立集度数和 $=$ 团度数和，且存在可行方案满足团是度数最大的若干个点。找到可行方案后，要么是团里出去一个点，要么是独立集里出去一个点，要么两边各出去一个点。

```
const int N=5005;
int n,i,j,x,y,ans,d[N],q[N],s[N];
bool cmp(int x,int y){return d[x]>d[y];}
int main(){
  read(n);
  for(i=1;i<=n;i++){
    read(d[i]);
    for(j=d[i];j;j--)read(x);
  }
  for(i=1;i<=n;i++)q[i]=i;
  sort(q+1,q+n+1,cmp);
  for(i=1;i<=n;i++)s[i]=s[i-1]+d[q[i]];
  for(i=0;i<=n;i++)if(i*(i-1)+s[n]-s[i]==s[i]){ans=1;break;}
  if(!ans)return puts("0"),0;
  for(x=0,j=1;j<=i;j++)if(d[q[i]]==d[q[j]])x++;
  for(y=0,j=i+1;j<=n;j++)if(d[q[i]]==d[q[j]])y++;
  ans+=x*y;
  for(j=1;j<=i;j++)if((i-1)*(i-2)+s[n]-s[i]+d[q[j]]==s[i]-d[q[j]])ans++;
  for(j=i+1;j<=n;j++)if((i+1)*i+s[n]-s[i]-d[q[j]]==s[i]+d[q[j]])ans++;
  if(n*(n-1)==s[n])ans--;
  if(s[n]==0)ans--;
  printf("%d",ans);
}
```

5.27 动态最小生成树

qx,qy 表示将输入的第 qx 条边的边权修改为 qy。

```
const int qsize=maxm+3*maxq;
int x[qsize],y[qsize],z[qsize],qx[maxq],qy[maxq],n,m,Q;
int a[maxn],*tz,kx[maxn],ky[maxn],kt,vd[maxn],id[maxm],app[maxm];bool extra[maxm];
int F(int x){
  int y=x,z;while(a[y])y=a[y];
  while(z=a[x])a[x]=y,x=z;
  return y;
}
bool cmp(int a,int b){return tz[a]<tz[b];}
void solve(int*qx,int*qy,int Q,int n,int*x,int*y,int*z,int m,long long ans){
  if(Q==1){
    for(int i=1;i<=n;i++)a[i]=0;
    z[qx[0]]=qy[0];
```

```
    for(int i=0;i<m;i++)id[i]=i;tz=z;
    sort(id,id+m,cmp);
    for(int i=0;i<m;i++){
      int ri=F(x[id[i]]),rj=F(y[id[i]]);
      if(ri!=rj)ans+=z[id[i]],a[ri]=rj;
    }
    printf(”%lld\n”,ans);
    return;
  }
  int ri,rj;
  kt=0;
  for(int i=1;i<=n;i++)a[i]=0;
  for(int i=0;i<Q;i++){
    ri=F(x[qx[i]]),rj=F(y[qx[i]]);
    if(ri!=rj)a[ri]=rj;
  }
  int tm=0;
  for(int i=0;i<m;i++)extra[i]=1;
  for(int i=0;i<Q;i++)extra[qx[i]]=0;
  for(int i=0;i<m;i++)if(extra[i])id[tm++]=i;
  tz=z;sort(id,id+tm,cmp);
  for(int i=0;i<tm;i++){
    ri=F(x[id[i]]),rj=F(y[id[i]]);
    if(ri!=rj){
      a[ri]=rj;ans+=z[id[i]];
      kx[kt]=x[id[i]];ky[kt]=y[id[i]];kt++;
    }
  }
  for(int i=1;i<=n;i++)a[i]=0;
  for(int i=0;i<kt;i++)a[F(kx[i])]=F(ky[i]);
  int n2=0;
  for(int i=1;i<=n;i++)if(!a[i])vd[i]=++n2;
  for(int i=1;i<=n;i++)if(a[i])vd[i]=vd[F(i)];
  int m2=0,*Nx=x+m,*Ny=y+m,*Nz=z+m;
  for(int i=0;i<m;i++)app[i]=-1;
  for(int i=0;i<Q;i++)if(app[qx[i]]==-1){
    Nx[m2]=vd[x[qx[i]]];Ny[m2]=vd[y[qx[i]]];Nz[m2]=z[qx[i]];
    app[qx[i]]=m2++;
  }
  for(int i=0;i<Q;i++)z[qx[i]]=qy[i],qx[i]=app[qx[i]];
  for(int i=1;i<=n2;i++)a[i]=0;
  for(int i=0;i<tm;i++){
    ri=F(vd[x[id[i]]]),rj=F(vd[y[id[i]]]);
    if(ri!=rj){
      a[ri]=rj;Nx[m2]=vd[x[id[i]]];
      Ny[m2]=vd[y[id[i]]];
      Nz[m2++]=z[id[i]];
    }
  }
  int mid=Q>>1;
  solve(qx,qy,mid,n2,Nx,Ny,Nz,m2,ans);
  solve(qx+mid,qy+mid,Q-mid,n2,Nx,Ny,Nz,m2,ans);
}
int main(){
  scanf(”%d%d%d”,&n,&m,&Q);
  for(int i=0;i<m;i++)scanf(”%d%d%d”,&x[i],&y[i],&z[i]);//nodes index from 1
  for(int i=0;i<Q;i++)scanf(”%d%d”,&qx[i],&qy[i]),qx[i]--;
```

```
  if(Q)solve(qx,qy,Q,n,x,y,z,m,0);
}
```

5.28 判断二分图完美匹配方案数是否是 4 的倍数

给定一个两边各有 N 个点的二分图，判断完美匹配的个数是否是 4 的倍数，时间复杂度 $O(\frac{n^4}{w})$。

```
#include<cstdio>
#include<algorithm>
using namespace std;
typedef unsigned long long ull;
const int N=305,M=5;
int Case,n,m,i,j,k,a[N][N],q[N],ans;char ch[N];
ull b[M],c[M],g[N][M],f[N][M];int h[N],w[N],v[N],flag,cnt;
inline void clr(){
  int i,j;
  for(i=0;i<n;i++)for(j=0,h[i]=-1;j<=m;j++)g[i][j]=0;
  cnt=0;
}
inline void init(int a[]){
  int j;
  for(j=0;j<=m;j++)b[j]=0;
  for(j=0;j<n;j++)if(a[j])b[j>>6]|=1ULL<<(j&63);
}
inline bool ins(int o,int mode=0){
  int i,j;
  for(i=0;i<n;i++)if(b[i>>6]>>(i&63)&1){
    if(h[i]<0){
      h[i]=o;
      for(j=0;j<=m;j++)g[i][j]=b[j];
      cnt++;
      return 1;
    }
    if(mode)w[h[i]]=1;
    for(j=0;j<=m;j++)b[j]^=g[i][j];
  }
  return 0;
}
bool solve(){
  scanf("%d",&n);
  ans=0;
  for(i=0;i<n;i++){
    scanf("%s",ch);
    for(j=0;j<n;j++)a[i][j]=ch[j]-'0';
  }
  for(i=0;i<n;i++){
    for(j=0;j<n;j++)if(a[i][j])break;
    if(j==n)return 1;
  }
  if(n<=5){
    for(i=0;i<n;i++)q[i]=i;
    do{
      for(i=0;i<n;i++)if(!a[i][q[i]])break;
      if(i==n)ans++;
```

```
    }while(next_permutation(q,q+n));
    return ans%4==0;
  }
  m=(n-1)>>6;
  clr();
  for(i=0;i<n;i++){
    init(a[i]);
    for(j=0;j<n;j++)w[j]=0;
    for(j=0;j<=m;j++)f[i][j]=0;
    bool t=ins(i,1);
    f[i][i>>6]=1ULL<<(i&63);
    for(j=0;j<n;j++)if(w[j])for(k=0;k<=m;k++)f[i][k]^=f[j][k];
    if(!t)break;
  }
  if(i==n)return 0;
  for(j=0;j<n;j++)w[j]=f[i][j>>6]>>(j&63)&1;
  if(!w[0])for(i=1;i<n;i++)if(w[i]){
    swap(w[i],w[0]);
    for(j=0;j<n;j++)swap(a[0][j],a[i][j]);
    break;
  }
  clr();
  for(i=0;i<n;i++)v[i]=0;
  for(i=0;i<n;i++)if(w[i])for(j=0;j<n;j++)v[j]+=a[i][j];
  for(i=0;i<n;i++)v[i]=(v[i]>>1)&1;
  init(v);
  flag=0;
  if(!ins(0))flag=1;
  for(i=1;i<n&&!flag;i++){
    init(a[i]);
    if(!ins(0))flag=1;
  }
  if(!flag)ans++;
  for(i=1;i<n;i++)if(w[i]){
    clr();
    for(j=1;j<n;j++)if(i!=j)init(a[j]),ins(0);
    if(cnt<n-2)continue;
    for(j=0;j<n;j++)if(a[i][j]){
      for(k=0;k<n;k++)v[k]=k<=j?0:a[i][k];
      init(v);
      ins(1);
      if(cnt==n-1){
        if(h[j]==-1)ans--;
        else{
          for(k=0;k<=m;k++)b[k]=c[k]=g[j][k];
          b[j>>6]^=1ULL<<(j&63);
          h[j]=-1;
          if(ins(1))ans--;
          h[j]=0;
          for(k=0;k<=m;k++)g[j][k]=c[k];
        }
      }
      for(k=0;k<n;k++)if(h[k]==1)h[k]=-1,cnt--;
    }
  }
  return (ans*2)%4==0;
}
```

```
int main(){
  scanf("%d",&Case);
  while(Case--)puts(solve()?"YES":"NO");
}
```

5.29 线图识别

给定一个 n 个点，$m(n,m \le 10^6)$ 条边的简单无向图 G，构造一个简单无向图 G' 使得 $L(G')=G$。

```
#include<cstdio>
#include<algorithm>
using namespace std;
typedef pair<int,int>P;
typedef unsigned int U;
const int N=1000010,LIM=1<<22,BUF=14000000;
char Buf[BUF],*buf=Buf;
int g[N],to[N<<1],nxt[N<<1],ed;
P res[N];
bool was[N];
P used_in_dfs[50];
int nei[N],V[N<<1],NXT[N<<1],ED;
int i,cnt,used_colors;
int que[N],size;
bool can[N<<1];
int occ[N<<1],pool[N<<1],my_occ[N<<1],cpool;
U val[N<<1],hv[N];
int hnxt[N],h[LIM+5],ch;
inline void read(int&a){for(a=0;*buf<48;buf++);while(*buf>47)a=a*10+*buf++-48;}
inline void ins(U x){
  int y=x&(LIM-1);
  hv[++ch]=x;
  hnxt[ch]=h[y];
  h[y]=ch;
}
inline bool has(U x){
  for(int i=h[x&(LIM-1)];i;i=hnxt[i])if(h[i]==x)return 1;
  return 0;
}
inline void add(int x,int y){to[++ed]=y;nxt[ed]=g[x];g[x]=ed;}
inline bool have_common(const P&p,const P&q){
  return p.first==q.first||p.first==q.second||p.second==q.first||p.second==q.second;
}
bool dfs(int id){
  if(id==i+1)return 1;
  for(int x=0;x<used_colors;x++)for(int y=x+1;y<used_colors;y++){
    if(id==0&&(x!=0||y!=1))continue;
    if(id==1&&(x!=0||y!=2))continue;
    bool flag=1;
    for(int j=0;j<id;j++)if(used_in_dfs[j]==P(x,y)){flag=0;break;}
    if(!flag)continue;
    res[que[id]]=P(cnt+x,cnt+y);
    bool ok=1;
    for(int j=0;j<id;j++){
      bool is_nei=0;
```

```
      for(int k=nei[id];k;k=NXT[k])if(V[k]==que[j]){is_nei=1;break;}
      bool really_nei=have_common(res[que[id]],res[que[j]]);
      if(is_nei!=really_nei){ok=0;break;}
    }
    if(ok){
      used_in_dfs[id]=P(x,y);
      if(dfs(id+1))return 1;
    }
  }
  return 0;
}
inline void fix(int v) {
  int x=res[v].first,y=res[v].second;
  occ[x]++,occ[y]++;
  ins(val[x]^val[y]);
}
int main(){
  int n,m;
  fread(Buf,1,BUF,stdin);read(n),read(m);
  for(i=1;i<N*2;i++)val[i]=val[i-1]*233+17;
  for(i=0;i<m;i++){
    int x,y;
    read(x),read(y);
    x--,y--;
    add(x,y),add(y,x);
  }
  for(i=0;i<n;i++)res[i]=P(-1,-1);
  for(int start=0;start<n;start++){
    if(was[start])continue;
    size=1;
    que[0]=start;
    was[start]=1;
    used_colors=0;
    ED=0;
    for(i=0;i<size;i++){
      int v=que[i];
      for(int j=g[v];j;j=nxt[j])if(!was[to[j]])que[size++]=to[j],was[to[j]]=1;
      nei[i]=0;
      cpool=0;
      for(int j=g[v];j;j=nxt[j])if(res[to[j]]!=P(-1,-1)){
        V[++ED]=to[j];
        NXT[ED]=nei[i];
        nei[i]=ED;
        int x=res[to[j]].first,y=res[to[j]].second;
        if(!my_occ[x])pool[cpool++]=x;
        my_occ[x]++;
        if(!my_occ[y])pool[cpool++]=y;
        my_occ[y]++;
      }
      if(used_colors<=4){
        for(used_colors=2;used_colors<=8; used_colors++)if(dfs(0))break;
        if(used_colors>8)return puts("-1"),0;
        if(used_colors>4){
          cnt+=used_colors;
          for(int j=0;j<=i;j++)fix(que[j]);
        }
        for(int p=0;p<cpool;p++)my_occ[pool[p]]=0;
```

```
        continue;
      }
      for(int p=0;p<cpool;p++){
        if(my_occ[pool[p]]==occ[pool[p]])can[pool[p]]=1;
        my_occ[pool[p]]=0;
      }
      int best_cover_size=3;
      P best_pair=P(-1,-1);
      for(int att=0;att<2;att++){
        int my_first=att==0?res[V[nei[i]]].first:res[V[nei[i]]].second;
        if(!can[my_first])continue;
        int uncovered=0;
        for(int j=nei[i];j;j=NXT[j]){
          int u=V[j];
          if(res[u].first==my_first||res[u].second==my_first)continue;
          uncovered++;
          my_occ[res[u].first]++;
          my_occ[res[u].second]++;
        }
        int cover_size=1,my_second=-1;
        if(uncovered!=0){
          for(int j=nei[i];j;j=NXT[j]){
            int u=V[j];
            if(res[u].first==my_first||res[u].second==my_first)continue;
            for(int k=0;k<2;k++){
              int p=k?res[u].first:res[u].second;
              if(!can[p])continue;
              if(has(val[my_first]^val[p]))continue;
              if(my_occ[p]==uncovered){my_second=p;break;}
            }
            if(~my_second)break;
          }
          if(my_second!=-1)cover_size=2;else cover_size=3;
        }
        if(cover_size<best_cover_size){
          best_cover_size=cover_size;
          best_pair=P(my_first,my_second);
        }
        for(int j=nei[i];j;j=NXT[j]){
          int u=V[j];
          if(res[u].first==my_first||res[u].second==my_first)continue;
          my_occ[res[u].first]=my_occ[res[u].second]=0;
        }
      }
      if(best_cover_size==3)return puts("-1"),0;
      res[v]=best_pair;
      if(res[v].second==-1)res[v].second=cnt++;
      fix(v);
      for(int p=0;p<cpool;p++)can[pool[p]]=0;
    }
    if(used_colors<=4){
      cnt+=used_colors;
      for(int j=0;j<size;j++)fix(que[j]);
    }
  }
  for(i=0;i<n;i++)printf("%d %d\n",res[i].first+1,res[i].second+1);
}
```

5.30 一般图最大权匹配

输入范围 $n \leq 400$ 完全图。

```
#include<bits/stdc++.h>
#define cin kin
#define DIST(e) (lab[e.u]+lab[e.v]-g[e.u][e.v].w*2)
using namespace std;
typedef long long ll;
const int N=1023,INF=1e9;
struct Edge{
  int u,v,w;
} g[N][N];
int n,m,n_x,lab[N],match[N],slack[N],st[N],pa[N],flower_from[N][N],S[N],vis[N];
vector<int> flower[N];
deque<int> q;
void update_slack(int u,int x){
  if(!slack[x]||DIST(g[u][x])<DIST(g[slack[x]][x]))slack[x]=u;
}
void set_slack(int x){
  slack[x]=0;
  for(int u=1; u<=n; ++u)
    if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)update_slack(u,x);
}
void q_push(int x){
  if(x<=n)return q.push_back(x);
  for(int i=0; i<flower[x].size(); i++)q_push(flower[x][i]);
}
void set_st(int x,int b){
  st[x]=b;
  if(x<=n)return;
  for(int i=0; i<flower[x].size(); ++i)set_st(flower[x][i],b);
}
int get_pr(int b,int xr){
  int pr=find(flower[b].begin(),flower[b].end(),xr)-flower[b].begin();
  if(pr%2==1){
    reverse(flower[b].begin()+1,flower[b].end());
    return (int)flower[b].size()-pr;
  }
  else return pr;
}
void set_match(int u,int v){
  match[u]=g[u][v].v;
  if(u<=n)return;
  Edge e=g[u][v];
  int xr=flower_from[u][e.u],pr=get_pr(u,xr);
  for(int i=0; i<pr; ++i)set_match(flower[u][i],flower[u][i^1]);
  set_match(xr,v);
  rotate(flower[u].begin(),flower[u].begin()+pr,flower[u].end());
}
void augment(int u,int v){
  int xnv=st[match[u]];
  set_match(u,v);
  if(!xnv)return;
```

```
  set_match(xnv,st[pa[xnv]]);
  augment(st[pa[xnv]],xnv);
}
int get_lca(int u,int v){
  static int t=0;
  for(++t; u||v; swap(u,v)){
    if(u==0)continue;
    if(vis[u]==t)return u;
    vis[u]=t;
    u=st[match[u]];
    if(u)u=st[pa[u]];
  }
  return 0;
}
void add_blossom(int u,int lca,int v){
  int b=n+1;
  while(b<=n_x&&st[b])++b;
  if(b>n_x)++n_x;
  lab[b]=0,S[b]=0;
  match[b]=match[lca];
  flower[b].clear();
  flower[b].push_back(lca);
  for(int x=u,y; x!=lca; x=st[pa[y]])
    flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
  reverse(flower[b].begin()+1,flower[b].end());
  for(int x=v,y; x!=lca; x=st[pa[y]])
    flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
  set_st(b,b);
  for(int x=1; x<=n_x; ++x)g[b][x].w=g[x][b].w=0;
  for(int x=1; x<=n; ++x)flower_from[b][x]=0;
  for(int i=0; i<flower[b].size(); ++i){
    int xs=flower[b][i];
    for(int x=1; x<=n_x; ++x)
      if(g[b][x].w==0||DIST(g[xs][x])<DIST(g[b][x]))
        g[b][x]=g[xs][x],g[x][b]=g[x][xs];
    for(int x=1; x<=n; ++x)
      if(flower_from[xs][x])flower_from[b][x]=xs;
  }
  set_slack(b);
}
void expand_blossom(int b)  // S[b] == 1{
  for(int i=0; i<flower[b].size(); ++i)
    set_st(flower[b][i],flower[b][i]);
  int xr=flower_from[b][g[b][pa[b]].u],pr=get_pr(b,xr);
  for(int i=0; i<pr; i+=2){
    int xs=flower[b][i],xns=flower[b][i+1];
    pa[xs]=g[xns][xs].u;
    S[xs]=1,S[xns]=0;
    slack[xs]=0,set_slack(xns);
    q_push(xns);
  }
  S[xr]=1,pa[xr]=pa[b];
  for(int i=pr+1; i<flower[b].size(); ++i){
    int xs=flower[b][i];
    S[xs]=-1,set_slack(xs);
  }
  st[b]=0;
```

```
}
bool on_found_Edge(const Edge &e){
  int u=st[e.u],v=st[e.v];
  if(S[v]==-1){
    pa[v]=e.u,S[v]=1;
    int nu=st[match[v]];
    slack[v]=slack[nu]=0;
    S[nu]=0,q_push(nu);
  }
  else if(S[v]==0){
    int lca=get_lca(u,v);
    if(!lca)return augment(u,v),augment(v,u),1;
    else add_blossom(u,lca,v);
  }
  return 0;
}
bool matching(){
  fill(S,S+n_x+1,-1),fill(slack,slack+n_x+1,0);
  q.clear();
  for(int x=1; x<=n_x; ++x)
    if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,q_push(x);
  if(q.empty())return 0;
  for(;;){
    while(q.size()){
      int u=q.front();
      q.pop_front();
      if(S[st[u]]==1)continue;
      for(int v=1; v<=n; ++v)
        if(g[u][v].w>0&&st[u]!=st[v]){
          if(DIST(g[u][v])==0){
            if(on_found_Edge(g[u][v]))return 1;
          }
          else update_slack(u,st[v]);
        }
    }
    int d=INF;
    for(int b=n+1; b<=n_x; ++b)
      if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
    for(int x=1; x<=n_x; ++x)
      if(st[x]==x&&slack[x]){
        if(S[x]==-1)d=min(d,DIST(g[slack[x]][x]));
        else if(S[x]==0)d=min(d,DIST(g[slack[x]][x])/2);
      }
    for(int u=1; u<=n; ++u){
      if(S[st[u]]==0){
        if(lab[u]<=d)return 0;
        lab[u]-=d;
      }
      else if(S[st[u]]==1)lab[u]+=d;
    }
    for(int b=n+1; b<=n_x; ++b)
      if(st[b]==b){
        if(S[st[b]]==0)lab[b]+=d*2;
        else if(S[st[b]]==1)lab[b]-=d*2;
      }
    q.clear();
    for(int x=1; x<=n_x; ++x)
```

```
      if(st[x]==x&&slack[x]&&st[slack[x]]!=x&&DIST(g[slack[x]][x])==0)
        if(on_found_Edge(g[slack[x]][x]))return 1;
    for(int b=n+1; b<=n_x; ++b)
      if(st[b]==b&&S[b]==1&&lab[b]==0)expand_blossom(b);
  }
  return 0;
}
pair<ll,int> weight_blossom(){
  fill(match,match+n+1,0);
  n_x=n;
  int n_matches=0;
  ll tot_weight=0;
  for(int u=0; u<=n; ++u)st[u]=u,flower[u].clear();
  int w_max=0;
  for(int u=1; u<=n; ++u)
    for(int v=1; v<=n; ++v){
      flower_from[u][v]=(u==v?u:0);
      w_max=max(w_max,g[u][v].w);
    }
  for(int u=1; u<=n; ++u)lab[u]=w_max;
  while(matching())++n_matches;
  for(int u=1; u<=n; ++u)
    if(match[u]&&match[u]<u)
      tot_weight+=g[u][match[u]].w;
  return make_pair(tot_weight,n_matches);
}
int main(){
  cin>>n>>m;
  for(int u=1; u<=n; ++u)
    for(int v=1; v<=n; ++v)
      g[u][v]=Edge {u,v,0};
  for(int i=0,u,v,w; i<m; ++i){
    cin>>u>>v>>w;
    g[u][v].w=g[v][u].w=w;
  }
  cout<<weight_blossom().first<<'\n';
  for(int u=1; u<=n; ++u)cout<<match[u]<<' ';
}
```

5.31 保序回归

定义 L_p 问题为给定 y 的偏序关系和权值 x，最小化 $\sum w_i|y_i - x_i|^p$ 的问题。

在 L_1 问题中，若任意一个 x 均不在 (a,b) 范围内, 那么一定存在原问题的一组最优解可以向 $\{a,b\}$ 取整得到 $a \le y_i \le b$ 下的最优解。

同时一定有 $y_i = x_j$，因此可以分治求出每个点的最优取值 (求 $[y_{mid}, y_{mid+1}]$ 问题的最优解)。

```
const int inf = 0x3f3f3f3f;
namespace flow {
struct edge {
  int to, cap, rev;
  edge(int t, int c, int r) {
    to = t;
    cap = c;
```

```
    rev = r;
  }
};
int n, source, sink, ans;
vector<vector<edge>> adj;
vector<int> dist, cur;
void init(int v, int s, int t) {
  n = v;
  source = s;
  sink = t;
  ans = 0;
  adj.clear();
  adj.resize(n);
  dist.resize(n);
  cur.resize(n);
}
void add(int x, int y, int c) {
  adj[x].push_back(edge(y, c, adj[y].size()));
  adj[y].push_back(edge(x, 0, adj[x].size() - 1));
}
bool bfs() {
  queue<int> q;
  for (int i = 0; i < n; ++i) {
    dist[i] = -1;
  }
  dist[source] = 0;
  q.push(source);
  while (!q.empty()) {
    int x = q.front();
    q.pop();
    for (auto e : adj[x]) {
      if (e.cap && !~dist[e.to]) {
        dist[e.to] = dist[x] + 1;
        if (e.to == sink) {
          return true;
        }
        q.push(e.to);
      }
    }
  }
  return false;
}
int dfs(int x, int f) {
  if (x == sink) {
    return f;
  }
  for (int &i = cur[x]; ~i; --i) {
    edge &e = adj[x][i];
    if (e.cap && dist[e.to] == dist[x] + 1) {
      int w = dfs(e.to, min(e.cap, f));
      if (w) {
        e.cap -= w;
        adj[e.to][e.rev].cap += w;
        return w;
      }
    }
  }
```

```
  return 0;
}
int max_flow() {
  while (bfs()) {
    for (int i = 0; i < n; ++i) {
      cur[i] = adj[i].size() - 1;
    }
    while (true) {
      int flow = dfs(source, inf);
      if (!flow) {
        break;
      }
      ans += flow;
    }
  }
  return ans;
}
vector<bool> min_cut() {
  max_flow();
  vector<bool> res(n);
  for (int i = 0; i < n; ++i) {
    res[i] = !~dist[i];
  }
  return res;
}
}
using flow::source;
using flow::sink;
using flow::init;
using flow::add;
using flow::min_cut;
long long solve(vector<int> a, vector<int> w, vector<vector<int>> adj) {
  // edge (u, v): x[u] <= x[v]
  // minimize \sum a[i]|x[i]-w[i]|
  int n = a.size();
  long long ans = 0;
  vector<int> id(n, -1);
  function<void(int, int, vector<int>)> solve = [&](int l, int r, vector<int> all) {
    if (all.empty()) {
      return;
    }
    if (all.size() == 1) {
      if (w[all[0]] < l) {
        ans += (long long) a[all[0]] * (l - w[all[0]]);
      }
      if (w[all[0]] > r) {
        ans += (long long) a[all[0]] * (w[all[0]] - r);
      }
      return;
    }
    if (l == r) {
      for (auto x : all) {
        ans += (long long) a[x] * abs(w[x] - l);
      }
      return;
    }
    int m = all.size(), mid = l + r >> 1;
```

```
    init(m + 2, m, m + 1);
    for (int i = 0; i < all.size(); ++i) {
      int x = all[i];
      id[x] = i;
      if (w[x] > mid) {
        add(source, i, a[x]);
      } else {
        add(i, sink, a[x]);
      }
    }
    for (auto x : all) {
      for (auto y : adj[x]) {
        if (~id[y]) {
          add(id[x], id[y], inf);
        }
      }
    }
    for (auto x : all) {
      id[x] = -1;
    }
    vector<bool> choose = min_cut();
    vector<int> left, right;
    for (int i = 0; i < all.size(); ++i) {
      if (choose[i]) {
        left.push_back(all[i]);
      } else {
        right.push_back(all[i]);
      }
    }
    solve(l, mid, left);
    solve(mid + 1, r, right);
  };
  vector<int> all(n);
  for (int i = 0; i < n; ++i) {
    all[i] = i;
  }
  solve(*min_element(w.begin(), w.end()), *max_element(w.begin(), w.end()), all);
  return ans;
}
```

5.32 拟阵交

给一个图，每条边有颜色和权值，第一个人只能看到红或者绿，第二个人只能看到绿或者蓝。对于每个 k，输出选择恰好 k 条边的情况下，两个人看到的图都连通的最小边权和。

设 M_1 为限制 1 的拟阵，一个方案合法当且仅当在限制 1 下连通，同理定义 M_2 为限制 2 的拟阵。

建立有向图，原图每条边作为一个点，并添加源汇 S 和 T。

对于上一个 k 的一组最优解 E 中的某条边 x，如果去掉它后仍然满足 M_1，则由 S 向 x 连边；若去掉它后仍然满足 M_2，则由 x 向 T 连边。

对于 E 中某条边 x 和不在 E 中的某条边 y，若将 x 换成 y 后满足 M_2，则由 x 向 y 连边；若满足 M_1，则由 y 向 x 连边。

用 SPFA 求出 S 到 T 的最短路，就能得到边数恰好减少 1 的最优解。时间复杂度 $O(n^4)$。

```
const int N=105,M=100000,inf=~0U>>1;
int n,m,i,S,T,x,y,g[N],v[N<<1],nxt[N<<1],ed,vis[N];
int cost[N],col[N],use[N],ans,fin[N];char ch[9];
inline void add(int x,int y){v[++ed]=y;nxt[ed]=g[x];g[x]=ed;}
void dfs(int x,char ban){
  if(vis[x])return;
  vis[x]=1;
  for(int i=g[x];i;i=nxt[i])if(use[i>>1]&&col[i>>1]!=ban)dfs(v[i],ban);
}
inline bool check(char ban){
  int i;
  for(i=1;i<=n;i++)vis[i]=0;
  dfs(1,ban);
  for(i=1;i<=n;i++)if(!vis[i])return 0;
  return 1;
}
namespace Matroid{
int g[N],v[M],nxt[M],ed,q[M],h,t,d[N],pre[N],w[N];bool in[N];
inline void add(int x,int y){v[++ed]=y;nxt[ed]=g[x];g[x]=ed;}
inline void ext(int x,int y,int z){
  if(d[x]<=y)return;
  d[x]=y;
  pre[x]=z;
  if(in[x])return;
  q[++t]=x;
  in[x]=1;
}
inline bool find(){
  int i,j;
  S=m+1,T=m+2;
  for(ed=0,i=1;i<=T;i++)g[i]=0;
  for(i=1;i<=m;i++)if(use[i]){
    w[i]=-cost[i];
    use[i]^=1;
    if(check(’R’))add(S,i);
    if(check(’B’))add(i,T);
    use[i]^=1;
  }else w[i]=cost[i];
  for(i=1;i<=m;i++)if(use[i])for(j=1;j<=m;j++)if(!use[j]){
    use[i]^=1,use[j]^=1;
    if(check(’B’))add(i,j);
    if(check(’R’))add(j,i);
    use[i]^=1,use[j]^=1;
  }
  for(i=1;i<=T;i++)d[i]=inf,in[i]=0;
  q[h=t=1]=S;
  d[S]=0,in[S]=1;
  while(h<=t){
    x=q[h++];
    //printf(”! %d %d %d\n”,x,d[x],pre[x]);
    for(i=g[x];i;i=nxt[i])ext(v[i],d[x]+w[v[i]],x);
    in[x]=0;
  }
  if(d[T]==inf)return 0;
  ans+=d[T];
```

```
  while(pre[T]!=S)use[T=pre[T]]^=1;
  return 1;
}
}
int main(){
  scanf("%d%d",&n,&m);
  for(ed=i=1;i<=m;i++){
    scanf("%d%d%d%s",&x,&y,&cost[i],ch);
    col[i]=ch[0];
    add(x,y),add(y,x);
    use[i]=1;
    ans+=cost[i];
  }
  if(!check('R')||!check('B')){
    for(i=1;i<=m;i++)puts("-1");
    return 0;
  }
  fin[m]=ans;
  for(i=m-1;i;i--)if(Matroid::find())fin[i]=ans;
  else{
    for(x=1;x<=i;x++)fin[x]=-1;
    break;
  }
  for(i=1;i<=m;i++)printf("%d\n",fin[i]);
}
```

6 博弈论

6.1 SG 函数的计算方法

一个局面的 SG 为 mex(后继局面的 SG)，mex 运算为集合中没出现的最小的自然数。几个局面的和的 SG 为单个的 SG 异或，SG 不为 0 时先手必胜，SG 为 0 时后手必胜。

6.2 Nim Game

n 堆石子，每次可以从一堆里面取任意个石子。对于一堆石子，SG 函数就是石子数。整个游戏的 SG 函数是每一堆石子的 SG 函数的异或和。

必胜：SG 不为 0，必败：SG 为 0。

6.3 Bash Game

每次最多取 m 个石子，其他同 Nim。一堆石子的 SG 函数为石子数 $\bmod(m+1)$。

必胜：SG 不为 0，必败：SG 为 0。

6.4 Nim-k Game

每次最多可以同时从 k 堆石子进行操作，这 k 堆可以取不同数量的石子。

一堆石子的 SG 函数为石子数，对每一个二进制位单独算，求 SG 函数每一个二进制位 1 的个数 $\bmod(k+1)$，如果都为 0，则必败，否则必胜。

6.5 Anti-Nim Game

不能取石子的一方获胜。

必胜：SG 不为 0 且至少有一堆石子数大于 1，SG 为 0 且每一堆石子数都不超过 1

必败：其余为必败。

6.6 Anti-SG Game

SG 游戏中最先不能行动的一方获胜。

必胜：SG 不为 0 且至少有一个游戏的 SG 大于 1，SG 为 0 且每一个游戏的 SG 都不超过 1

必败：其余为必败。

6.7 Staircase Nim

阶梯博弈，每次可以从一个阶梯上拿掉任意数量石子放到下一层阶梯，不能操作的为输。

SG 函数为奇数阶梯上的石子的异或和，如果移动偶数层的石子到奇数层，对手一定可以继续移动这些石子到偶数层，使得其 SG 不变。

必胜：SG 不为 0，必败：SG 为 0。

6.8 Lasker's Nim Game

n 堆石子，每次可以从一堆里面取任意个石子，或者选择某堆至少为 2 的石子，分成两堆非空石子。

$SG(0)=0, SG(1)=1, SG(2)=2, SG(3)=4$。

对于 $k\geq 1$，$SG(4k)=4k-1, SG(4k+1)=4k+1, SG(4k+2)=4k+2, SG(4k+3)=4k+4$。

6.9 Wythoff Game

有两堆石子，每次可以从一堆或者两堆里拿走一样数目的石子，不能取的为输。

必败态为 $(1,2),(3,5),(4,7),(6,10)...$

差为 $1,2,3,4.....$ 每一对数的第一个数为前面没出现的最小的正整数。

$a_k=\lfloor\frac{k(1+\sqrt{5})}{2}\rfloor, b_k=a_k+k$。

6.10 树上删边游戏

给定一棵 n 个点的有根树，每次可以删掉一个子树，则叶子节点的 SG 值为 0，非叶子节点的 SG 值为其所有孩子节点 (SG 值 +1) 的异或和。

6.11 无向图删边游戏

结论：把奇环缩成一个点加一条新边，把偶环缩成一个点，不影响 SG，然后套用树上删边游戏。

```
#include<cstdio>
const int N=1000010,M=1000010;
int T,n,m,mm,k,i,x,y;
int e[M][2],cut[M],g[N],v[M<<1],nxt[M<<1],ed;
int f[N],dfn[N],low[N],num,cnt,from[N],sg[N];
void add(int x,int y){v[++ed]=y;nxt[ed]=g[x];g[x]=ed;}
void addedge(int x,int y,int k){//添加k条(x,y)的边，注意这里重边处理方式不对
  if(!k)return;
  k=k&1?1:2;
  if(x==y)y=++n;
  while(k--){
    e[++m][0]=x;e[m][1]=y;
    add(x,y),add(y,x);
  }
}
void tarjan(int x){
  dfn[x]=low[x]=++num;
  for(int i=g[x];i;i=nxt[i])if(!dfn[v[i]]){
    f[v[i]]=i>>1,tarjan(v[i]);
    if(low[x]>low[v[i]])low[x]=low[v[i]];
  }else if(f[x]!=(i>>1)&&low[x]>dfn[v[i]])low[x]=dfn[v[i]];
  if(f[x]&&low[x]==dfn[x])cut[f[x]]=1;
}
void dfs(int x,int y){
  from[x]=y;
  for(int i=g[x];i;i=nxt[i])if(!from[v[i]]&&!cut[i>>1])dfs(v[i],y);
}
```

```
void cal(int x,int y){
  int z=0;
  for(int i=g[x];i;i=nxt[i])if(v[i]!=y)cal(v[i],x),sg[x]^=sg[v[i]];
  else z^=1;
  if(y)sg[x]+=z;
}
int main(){
  read(T);
  while(T——){
    read(n),read(mm),read(k);
    ed=1,m=0,n++;
    while(k——)read(x),addedge(1,x+1,2);
    while(mm——)read(x),read(y),read(k),addedge(x+1,y+1,k);
    tarjan(1);
    for(i=1;i<=n;i++)if(!from[i])dfs(i,++cnt);
    for(ed=0,i=1;i<=n;i++)g[i]=0;
    for(i=1;i<=m;i++)if(cut[i]){
      add(from[e[i][0]],from[e[i][1]]);
      add(from[e[i][1]],from[e[i][0]]);
    }
    for(i=1;i<=m;i++)if(!cut[i])sg[from[e[i][0]]]^=1;
    cal(from[1],0);
    puts(sg[from[1]]?”1”:”0”);
    for(ed=num=cnt=0,i=1;i<=n;i++)g[i]=sg[i]=f[i]=dfn[i]=low[i]=from[i]=0;
    for(i=1;i<=m;i++)cut[i]=0;
  }
  return 0;
}
```

6.12 翻硬币游戏

n 枚硬币排成一排，有的正面朝上，有的反面朝上。游戏者根据某些约束翻硬币（如：每次只能翻一或两枚，或者每次只能翻连续的几枚），但他所翻动的硬币中，最右边的必须是从正面翻到反面。谁不能翻谁输。

需要先开动脑筋把游戏转化为其他的取石子游戏之类的，然后用如下定理解决：

局面的 SG 值等于局面中每个正面朝上的棋子单一存在时的 SG 值的异或和。

6.12.1 每一次只能翻转一枚硬币

$SG(0)=0, SG(k)=1(k>0)$。

6.12.2 每一次可以翻转一枚或两枚硬币

$SG(n)=n$。

6.12.3 Twins Game

每次必须翻动两个硬币，而且这两个硬币的距离要在可行集 $S=1,2,3$ 中，相当于 Bash Game。

6.12.4 每一次必须翻连续的 n 个硬币

$SG(nk) = 1(k > 0)$，其他 SG 函数值为 0。

6.12.5 Ruler Game

每一次可以翻任意长度的连续一段硬币，$SG(x)$ 为 x 中包含的 2 的最高次幂，即 $SG(x) = \lfloor \log_2 x \rfloor + 1$。

6.13 K 倍动态减法游戏

有一个整数 $S(S \geq 2)$，两个人想让它变成 0。首先，第一个人需要把 S 减掉一个正数 $x(0 < x < S)$。之后双方轮流把 S 减掉一个正整数，但都不能超过先前一回合对方减掉的数的 $K(1 \leq K \leq 100000)$ 倍，减到 0 的一方获胜。问谁会获得胜利，若胜利还要求先手第一步至少减去多少。

```
const int N=750000;
int i,j;ll n,k,a[N],b[N],ans;
int main(){
  scanf("%lld%lld",&n,&k);
  a[1]=b[1]=1;
  for(i=2,j=0;;i++){
    a[i]=b[i-1]+1;
    if(a[i]>=n)break;
    while(a[j+1]*k<a[i])j++;
    b[i]=a[i]+b[j];
  }
  while(a[i]>n)i--;
  if(a[i]==n)puts("lose");
  else{
    while(n){
      while(a[i]>n)i--;
      n-=a[i];
      ans=a[i];
    }
    printf("%lld",ans);
  }
}
```

6.14 Blue-Red Hackenbush

若干个 BW 串，W 手只能拿 W，B 手只能拿 B，每次拿走一个字符后其后缀也会消失，最先不能操作者输。

对于每个串计算 Surreal Number 并求和，若 > 0 无论先手是谁，W 都必胜；若 $= 0$ 则后手必胜；若 < 0 则无论先手是谁，B 都必胜。

```
ll cal(){
  int n;
  scanf("%d",&n);
  ll x=0,k=1LL<<50;int i;char a[99],b[99];
  for(i=1;i<=n;i++)scanf("%s",a),b[i]=a[0];
```

```
  for(i=1;i<=n&&b[i]==b[1];i++)if(b[i]==’W’)x+=k;else x-=k;
  for(k>>=1;i<=n;i++,k>>=1)if(b[i]==’W’)x+=k;else x-=k;
  return x;
}
```

6.14.1 Surreal Number

```
struct SN{
  ll a;int bk;
  SN(ll _a=0,int _bk=0):a(_a),bk(_bk){
    while(bk>0&&!(a&1))a>>=1,--bk;
  }
  SN operator -()const{return SN(-a,bk);}
};
const SN inf=SN(1,-1);
inline SN operator+(SN a,SN b){
  if(a.bk<0)return a;
  if(b.bk<0)return b;
  while(a.bk<b.bk)++a.bk,a.a<<=1;
  while(b.bk<a.bk)++b.bk,b.a<<=1;
  return SN(a.a+b.a,a.bk);
}
inline SN operator-(SN a,SN b){
  if(a.bk<0)return a;
  if(b.bk<0)return -b;
  while(a.bk<b.bk)++a.bk,a.a<<=1;
  while(b.bk<a.bk)++b.bk,b.a<<=1;
  return SN(a.a-b.a,a.bk);
}
inline bool operator<(const SN&a,const SN&b){
  return (a-b).a<0;
}
inline bool operator>(const SN&a,const SN&b){
  return (a-b).a>0;
}
inline bool operator<=(const SN&a,const SN&b){
  return (a-b).a<=0;
}
inline bool operator>=(const SN&a,const SN&b){
  return (a-b).a>=0;
}
inline SN construct(const SN&l,const SN&r){
  if(l>=r)
    throw ”INVALID construction: left >= right”;
  if(l<0&&r>0)return 0;
  if((l>0&&l.bk<0)||(l<0&&r.bk<0))
    throw ”INVALID construction: both same inf”;
  ll reta=(l>=0?(l.a>>l.bk):((r.a-1)>>r.bk));
  int retbk=0;
  while(1){
    for(int i=retbk;i>=0;--i)
      if(SN(reta+(1LL<<i),retbk)<r)
        reta+=1LL<<i;
    if(SN(reta,retbk)>l)break;
    reta<<=1;++retbk;
```

```
  }
  return SN(reta,retbk);
}
SN dp(int x){
  SN l=-inf,r=inf;
  for(int y=x's next states){
    SN t=dp(y);
    l=max(l,t);
    r=min(r,t);
  }
  return construct(l, r);
}
int main(){
  SN ans=0;
  for(x=games)ans=ans+dp(x);
  if(ans>0)puts("Alice Wins");
  else if(ans<0) puts("Bob Wins");
  else puts("Second Player Wins");
}
```

6.15 高维组合游戏

等于每一维单独的 SG 值的 nim 积。

```
const int N=50;
ll nm[N][N];
ll nimmul(ll x,ll y,int o);
void niminit(){
  for(int i=0;i<N;i++)for(int j=0;j<N;j++)nm[i][j]=nimmul(i,j,0);
}
ll nimpow(ll x,ll y,int o=1){
  if(o&&x<N&&y<N)return nm[x][y];
  if(!x)return 0;
  if(x==1)return y==1;
  ll t=2;
  while(t*t<=x)t*=t;
  ll c1=nimpow(x/t,y/t,o),
     c2=nimpow(x/t,y%t,o);
  return (c1^c2)*t^nimpow(t>>1,c1,o);
}
ll nimmul(ll x,ll y,int o=1){
  if(o&&x<N&&y<N)return nm[x][y];
  if(x<y)swap(x,y);
  if(!y)return 0;
  if(x==1)return 1;
  ll t=2;
  while(t*t<=x)t*=t;
  ll c1=nimmul(x/t,y/t,o),
     c2=nimmul(x/t,y%t,o)^nimmul(x%t,y/t,o),
     c3=nimmul(x%t,y%t,o);
  return (c1^c2)*t^c3^nimpow(t>>1,c1,o);
}
```

6.16 Termites

一个序列，两个人轮流从最左或者最右拿一个数，谁总和大谁胜。

转化成求最大差值，对于连续的三个数 A, B, C，如果 $A \leq B$ 且 $B \geq C$，那么可以将这三个数合并为 $A - B + C$。从左往右维护一个栈，每次新加入数的时候不断合并栈顶三个元素，那么最后栈是先递减再递增，此时贪心拿两侧最大的数即为最佳策略。

6.17 Game of Sorting

给出一列数，两人轮流从最左或者最右拿走一个数，如果目前轮到 A 操作，这个序列单调不下降或者单调不上升，那么 A 输。

每次询问在某个区间 $[l, r]$ 玩游戏的结果。

对于一个区间 $[l, r]$，若它是单调的，则显然先手必败。

若去掉 l 或者 r 后变成了单调的，则显然先手必胜。

若 $[l + 1, r - 1]$ 是单调的，那么同理先手必败。

若 $[l, r - 2]$ 是单调的，那么先手若取 r 会导致后手必胜，后手同理，故两人会一直取 l 直到出现上面第一种或者第二种情况，可以根据奇偶性判断。

同理可以得出 $[l + 2, r]$ 是单调的情形的处理方法。

除此之外，有 $[l, r]$ 的答案等于 $[l + 1, r - 1]$ 的答案，二分找到最小的 k 满足 $[l + k, r - k]$ 可以由上面方法直接判断出胜负即可。

时间复杂度 $O(m \log n)$。

```
const int N=1000010;
int n,i,a[N],fl[N],fr[N],gl[N],gr[N],m,x,y;
inline int check(int l,int r){
  if(l>=r)return -1;
  if(fr[l]>=r)return -1;
  if(fr[l]==r-1)return 1;
  if(fl[r]==l+1)return 1;
  if(fr[l+1]==r-1)return -1;
  int x;
  if(fr[l]==r-2){
    if(fl[r-1]<fl[r])x=(l&1)^(fl[r-1]&1)^1;
    else x=(l&1)^(fl[r]&1);
    return x?1:-1;
  }
  if(fl[r]==l+2){
    if(fr[l+1]>fr[l])x=(r&1)^(fr[l+1]&1)^1;
    else x=(r&1)^(fr[l]&1);
    return x?1:-1;
  }
  return 0;
}
inline int cal(int x,int y){
  int l=0,r=(y-x)/2+5,mid,t,o;
  while(l<=r){
    mid=(l+r)>>1;
    t=check(x+mid,y-mid);
    if(!t)l=mid+1;else r=mid-1,o=t;
  }
```

```
  return o;
}
int main(){
  scanf(”%d”,&n);
  for(i=1;i<=n;i++)scanf(”%d”,&a[i]);
  for(fr[n]=gr[n]=n,i=n-1;i;i--){
    fr[i]=a[i]<=a[i+1]?fr[i+1]:i;
    gr[i]=a[i]>=a[i+1]?gr[i+1]:i;
  }
  for(fl[1]=gl[1]=1,i=2;i<=n;i++){
    fl[i]=a[i]<=a[i-1]?fl[i-1]:i;
    gl[i]=a[i]>=a[i-1]?gl[i-1]:i;
  }
  for(i=1;i<=n;i++)fl[i]=min(fl[i],gl[i]),fr[i]=max(fr[i],gr[i]);
  scanf(”%d”,&m);
  while(m--)scanf(”%d%d”,&x,&y),puts(cal(x,y)>0?”Alice”:”Bob”);
}
```

6.18 Xormites

给出一列数，两人轮流从最左或者最右拿走一个数，最后得分是拿过的数的异或和，谁高谁赢，预测结果。

首先求出所有数的异或和 sum，若先手拿到了 A，则后手必然拿到了 $sum \oplus A$。

若 $sum = 0$，则 $A = sum \oplus A$，必定平局。

否则找到 sum 最高位的 1，那么拿到奇数个 1 的一方获胜，可以将所有数转化为 0 和 1。

若 n 是偶数，那么将序列黑白染色，必定有一方异或和较大，且先手可以保证自己拿走全部黑或者全部白，故先手必胜。

否则 n 是奇数，若 a_1 和 a_n 都是 0，那么后手必然可以通过上述方法获胜。

因此先手第一步必须要拿走一个 1，接下来只能模仿对手行动，检查是否可能即可。

时间复杂度 $O(n)$。

```
int Case,n,i,k,sum,a[50010];
bool check(int L,int R){
  int l=L,r=R,i,cnt=0;
  while(l<r&&a[l]==a[r])l++,r--;
  for(i=l;i<=r;i+=2)if(a[i]^a[i+1])return 0;
  for(i=L;i<=R;i++)cnt+=a[i];
  return cnt/2%2==0;
}
int solve(){
  scanf(”%d”,&n);
  sum=0;
  for(i=1;i<=n;i++){
    scanf(”%d”,&a[i]);
    sum^=a[i];
  }
  if(!sum)return 0;
  if(n%2==0)return 1;
  for(k=30;!(sum>>k&1);k--);
  for(i=1;i<=n;i++)a[i]=a[i]>>k&1;
  if(!a[1]&&!a[n])return -1;
```

```
  if(a[1]&&check(2,n))return 1;
  if(a[n]&&check(1,n-1))return 1;
  return -1;
}
int main(){
  scanf("%d",&Case);
  while(Case--){
    int t=solve();
    if(t==1)puts("First");
    if(t==0)puts("Draw");
    if(t==-1)puts("Second");
  }
}
```

7 数学

7.1 Bell 数

B_n 表示把 n 个带标号的物品划分为若干不相交集合的方案数。

有递推式 $B_{n+1} = \sum_{k=0}^{n} C(n,k)B_k$。

下面为 $O(P^2 \log P)$ 计算 B_n 对 $999999598 = 2 \times 13 \times 5281 \times 7283$ 取模的代码。

```
#include<cstdio>
typedef long long ll;
const int N=7284,P=999999598;
ll n;int a[4]={2,13,5281,7283},f[N],s[2][N],i,j,x;
int cal(int x,ll n){
  int i,j,k,m=0,b[N],c[N],d[70];
  for(i=0;i<=x;i++)b[i]=f[i]%x;
  while(n)d[m++]=n%x,n/=x;
  for(i=1;i<m;i++)for(j=1;j<=d[i];j++){
    for(k=0;k<x;k++)c[k]=(b[k]*i+b[k+1])%x;
    c[x]=(c[0]+c[1])%x;
    for(k=0;k<=x;k++)b[k]=c[k];
  }
  return c[d[0]];
}
ll pow(ll a,ll b,ll p){ll t=1;for(a%=p;b;b>>=1LL,a=a*a%p)if(b&1LL)t=t*a%p;return t;}
ll bell(ll n){
  if(n<N)return f[n];
  ll t=0;
  for(int i=0;i<4;i++)t=(t+(P/a[i])*pow(P/a[i],a[i]-2,a[i])%P*cal(a[i],n)%P)%P;
  return t;
}
int main(){
  f[0]=f[1]=s[0][0]=1,s[0][1]=2;
  for(i=2,x=1;i<N;i++,x^=1)for(f[i]=s[x][0]=s[x^1][i-1],j=1;j<=i;j++)
    s[x][j]=(s[x^1][j-1]+s[x][j-1])%P;
  scanf("%lld",&n),printf("%lld",bell(n));
}
```

7.2 扩展欧几里得算法解同余方程

$ans[]$ 保存的为循环节内的所有解。

```
int exgcd(int a,int b,int&x,int&y){
  if(!b)return x=1,y=0,a;
  int d=exgcd(b,a%b,x,y),t=x;
  return x=y,y=t-a/b*y,d;
}
void cal(ll a,ll b,ll n){//ax=b(mod n)
  ll x,y,d=exgcd(a,n,x,y);
  if(b%d)return;
  x=(x%n+n)%n;
  ans[cnt=1]=x*(b/d)%(n/d);
  for(ll i=1;i<d;i++)ans[++cnt]=(ans[1]+i*n/d)%n;
}
```

7.3 同余方程组

```
int n,flag,k,m,a,r,d,x,y;
int main(){
  scanf("%d",&n);
  flag=k=1,m=0;
  while(n--){
    scanf("%d%d",&a,&r);//ans%a=r
    if(flag){
      d=exgcd(k,a,x,y);
      if((r-m)%d){flag=0;continue;}
      x=(x*(r-m)/d+a/d)%(a/d),y=k/d*a,m=((x*k+m)%y)%y;
      if(m<0)m+=y;
      k=y;
    }
  }
  printf("%d",flag?m:-1);//若flag=1，说明有解，解为ki+m，i为任意整数
}
```

7.4 线性基

7.4.1 异或线性基

若要查询第 k 小子集异或和，则把 k 写成二进制，对于是 1 的第 i 位，把从低位到高位第 i 个不为 0 的数异或进答案。

若要判断是否有非空子集的异或和为 0，如果不存在自由基，那么说明只有空集的异或值为 0，需要高斯消元来判断。

```
struct Base{
  int a[31];
  Base(){for(int i=0;i<31;i++)a[i]=0;}
  void ins(int x){for(int i=30;~i;i--)if(x>>i&1){if(a[i])x^=a[i];else{a[i]=x;break;}}}
  void ask(){//查询最大子集异或和
    int t=0;
    for(int i=30;~i;i--)up(t,t^a[i]);
    printf("%d\n",t);
  }
};
```

7.4.2 实数线性基

ins 返回要插入的数是否可以被之前的数线性表示出来，返回 1 表示不能，0 表示可以。

```
struct Base{
  double a[N][N];bool v[N];
  Base(){
    for(int i=0;i<N;i++)for(int j=0;j<N;j++)a[i][j]=0;
    for(int i=0;i<N;i++)v[i]=0;
  }
  bool ins(double*x){
    for(int i=0;i<m;i++)if(fabs(x[i])>1e-5){
      if(v[i]){
```

```
        double t=x[i]/a[i][i];
        for(int j=0;j<m;j++)x[j]-=t*a[i][j];
      }else{
        v[i]=1;
        for(int j=0;j<m;j++)a[i][j]=x[j];
        return 1;
      }
    }
    return 0;
  }
};
```

7.5 原根、指标、离散对数

设 P 为质数，G 为 P 的原根，则 $x^y \equiv b \pmod P$ 等价于 $y\ ind\ x \equiv b \pmod{P-1}$。其中 $G^{ind\ x} \equiv x \pmod P$。

7.5.1 求原根

```
int q[10000];
int pow(ll a,int b,int P){ll t=1;for(;b;b>>=1,a=(ll)a*a%P)if(b&1)t=t*a%P;return t;}
int getG(int n){
  int i,j,t=0;
  for(i=2;(ll)i*i<n-1;i++)if((n-1)%i==0)q[t++]=i,q[t++]=(n-1)/i;
  for(i=2;;i++){
    for(j=0;j<t;j++)if(pow(i,q[j],n)==1)break;
    if(j==t)return i;
  }
}
```

7.5.2 扩展 Baby Step Giant Step

求满足 $a^x \bmod m = r$ 的最小整数 x。

```
#include<cstdio>
#include<cmath>
#include<map>
#include<algorithm>
#include<tr1/unordered_map>
using namespace std::tr1;
using namespace std;
typedef long long ll;
typedef pair<int,int>P;
int phi(int n){
  int t=1,i;
  for(i=2;i*i<=n;i++)if(n%i==0)for(n/=i,t*=i-1;n%i==0;n/=i,t*=i);
  if(n>1)t*=n-1;
  return t;
}
int pow(ll a,int b,int m){ll t=1;for(;b;b>>=1,a=a*a%m)if(b&1)t=t*a%m;return t;}
int bsgs(int a,int r,int m){
  if(r>=m)return -1;
```

```
  int i,g,x,c=0,at=int(2+sqrt(m));
  for(i=0,x=1%m;i<50;i++,x=ll(x)*a%m)if(x==r)return i;
  for(g=x=1;__gcd(int(ll(x)*a%m),m)!=g;c++)g=__gcd(x=ll(x)*a%m,m);
  if(r%g)return -1;
  if(x==r)return c;
  unordered_map<int,int>u;
  g=phi(m/g),u[x]=0;g=pow(a,g-at%g,m);
  for(i=1;i<at;i++){
    u.insert(P(x=ll(x)*a%m,i));
    if(x==r)return c+i;
  }
  for(i=1;i<at;i++){
    unordered_map<int,int>::iterator t=u.find(r=ll(r)*g%m);
    if(t!=u.end())return c+i*at+t->second;
  }
  return -1;
}
```

7.6 Catalan 数

$h_1=1$，$h_n=\frac{h_{n-1}(4n-2)}{n+1}=\frac{C(2n,n)}{n+1}=C(2n,n)-C(2n,n-1)$。

在一个格点阵列中，从 $(0,0)$ 点走到 (n,m) 点且不经过对角线 $x=y$ 的方法数 $(x>y)$: $C(n+m-1,m)-C(n+m-1,m-1)$。

在一个格点阵列中，从 $(0,0)$ 点走到 (n,m) 点且不穿过对角线 $x=y$ 的方法数 $(x\geq y)$: $C(n+m,m)-C(n+m,m-1)$。

7.7 扩展 Cayley 公式

对于 n 个点，m 个连通块的图，假设每个连通块有 $a[i]$ 个点，那么用 $s-1$ 条边把它连通的方案数为 $n^{s-2}a[1]a[2]...a[m]$。

7.8 Jacobi's Four Square Theorem

设 $a^2+b^2+c^2+d^2=n$ 的自然数解个数为 $r4(n)$，$d(n)$ 为 n 的约数和，由 Jacobi's Four Square Theorem 可知，若 n 是奇数，则 $r4(n)=8d(n)$，否则 $r4(n)=24d(k)$，k 是 n 去除所有 2 后的结果。

7.9 复数

复数相乘的几何意义为长度相乘，极角相加。

```
struct comp{
double r,i;comp(double _r=0,double _i=0){r=_r;i=_i;}
comp operator+(const comp x){return comp(r+x.r,i+x.i);}
comp operator-(const comp x){return comp(r-x.r,i-x.i);}
comp operator*(const comp x){return comp(r*x.r-i*x.i,r*x.i+i*x.r);}
comp operator/(const comp x){
  double t=x.r*x.r+x.i*x.i;
  return comp((r*x.r+i*x.i)/t,(i*x.r-r*x.i)/t);
}};
```

7.10 高斯消元

n 个未知量，n 个方程，a 为增广矩阵，求逆矩阵时每次要 for 1 到 n 而不能因为减少常数丢弃正确性。

```
for(i=1;i<=n;i++){
  for(k=i,j=i+1;j<=n;j++)if(fabs(a[j][i])>fabs(a[k][i]))k=j;
  if(k!=i)for(j=i;j<=n+1;j++)t=a[i][j],a[i][j]=a[k][j],a[k][j]=t;
  for(j=i+1;j<=n;j++)for(t=a[j][i]/a[i][i],k=i;k<=n+1;k++)a[j][k]-=a[i][k]*t;
}
for(ans[n]=a[n][n+1]/a[n][n],i=n-1;i;i--){
  for(ans[i]=a[i][n+1],j=n;j>i;j--)ans[i]-=ans[j]*a[i][j];
  ans[i]/=a[i][i];
}
```

7.10.1 行列式

求矩阵 a 的 n 阶行列式对任意数 P 取模的结果。

```
ll det(int n){
  ll ans=1;bool flag=1;
  for(i=1;i<=n;i++)for(j=1;j<=n;j++)a[i][j]=(a[i][j]+P)%P;
  for(i=1;i<=n;i++){
    for(j=i+1;j<=n;j++)while(a[j][i]){
      ll t=a[i][i]/a[j][i];
      for(k=i;k<=n;k++)a[i][k]=(a[i][k]+P-t*a[j][k]%P)%P;
      for(k=i;k<=n;k++)swap(a[i][k],a[j][k]);
      flag^=1;
    }
    ans=ans*a[i][i]%P;
    if(!ans)return 0;
  }
  if(!flag)ans=(P-ans);
  return ans;
}
```

7.10.2 Matrix-Tree 定理

对于一张图，建立矩阵 C，$C[i][i]=i$ 的度数，若 i,j 之间有边，那么 $C[i][j]=-1$，否则为 0。这张图的生成树个数等于矩阵 C 的 $n-1$ 阶行列式的值。

7.11 康托展开

输入 n，查询某个排列的排名以及字典序第 m 小的排列。

```
#include<cstdio>
int n,q,i,j,t,a[22];long long f[22],m,ans;char op[5];
int main(){
  scanf("%d%d",&n,&q);
  for(f[1]=1,i=2;i<n;i++)f[i]=f[i-1]*i;
  while(q--){
    scanf("%s",op);
```

```
    if(op[0]=='P'){
      scanf("%lld",&m);
      for(i=1;i<=n;i++)a[i]=i;
      for(m--,j=1;j<n;j++){
        printf("%d ",a[m/f[n-j]+1]);
        for(i=m/f[n-j]+1;i<=n-j;i++)a[i]=a[i+1];
        m%=f[n-j];
      }
      printf("%d\n",a[1]);
    }else{
      for(i=1;i<=n;i++)scanf("%d",&a[i]);
      for(ans=0,i=1;i<n;i++){
        for(t=0,j=n;j>i;j--)if(a[j]<a[i])t++;
        ans=(ans+t)*(n-i);
      }
      printf("%lld\n",ans+1);
    }
  }
}
```

7.12 自适应 Simpson

给定一个函数 $f(x)$，求 $[a,b]$ 区间内 $f(x)$ 到 x 轴所形成区域的面积。

根据辛普森公式，有 S 近似等于 $\frac{b-a}{6}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]$。

```
double simpson(double l,double r){return (f(l)+f(r)+4*f((l+r)/2.0))*(r-l)/6.0;}
double rsimpson(double l,double r){
  double mid=(l+r)/2.0;
  if(fabs(simpson(l,r)-simpson(l,mid)-simpson(mid,r))<eps)
    return simpson(l,mid)+simpson(mid,r);
  return rsimpson(l,mid)+rsimpson(mid,r);
}
```

7.13 线性规划

n 个约束条件，m 个未知数，求 $\sum_{i=1}^{m} a[0][i]x[i]$ 的最大值。

约束条件：$\sum_{j=1}^{m}(-a[i][j]) \times x[j] \leq a[i][0]$。

若要求最小值，则需进行对偶，即把目标函数的系数与约数条件右边的数交换，然后把矩阵转置。

```
#define rep(i,l,n) for(int i=l;i<=n;i++)
const int N=1810,M=610,inf=~0U>>2;
int n,m,a[N][M],nxt[M];
void cal(int l,int e){
  a[l][e]=-1;t=M-1;
  rep(i,0,m)if(a[l][i])nxt[t]=i,t=i;nxt[t]=-1;
  rep(i,0,n)if(i!=l&&(t=a[i][e])){
    a[i][e]=0;
    for(int j=nxt[M-1];~j;j=nxt[j])a[i][j]+=a[l][j]*t;
  }
}
int work(){
```

```
  for(;;){int min=inf,l=0,e=0;
    rep(i,1,m)if(a[0][i]>0){e=i;break;}
    if(!e)return a[0][0];
    rep(i,1,n)if(a[i][e]<0&&a[i][0]<min)min=a[i][0],l=i;
    cal(l,e);
  }
}
```

7.14 实数线性规划

求 $\max\{cx|Ax \le b, x \ge 0\}$ 的解。

```
typedef vector<double>VD;
VD simplex(vector<VD>A,VD b,VD c){
  int n=A.size(),m=A[0].size()+1,r=n,s=m-1;
  vector<VD>D(n+2,VD(m+1,0));vector<int>ix(n+m);
  for(int i=0;i<n+m;i++)ix[i]=i;
  for(int i=0;i<n;i++){
    for(int j=0;j<m-1;j++)D[i][j]=-A[i][j];
    D[i][m-1]=1;D[i][m]=b[i];
    if(D[r][m]>D[i][m])r=i;
  }
  for(int j=0;j<m-1;j++)D[n][j]=c[j];
  D[n+1][m-1]=-1;
  for(double d;;){
    if(r<n){
      int t=ix[s];ix[s]=ix[r+m];ix[r+m]=t;
      D[r][s]=1.0/D[r][s];vector<int>speedUp;
      for(int j=0;j<=m;j++)if(j!=s){
        D[r][j]*=-D[r][s];
        if(D[r][j])speedUp.push_back(j);
      }
      for(int i=0;i<=n+1;i++)if(i!=r){
        for(int j=0;j<speedUp.size();j++)
        D[i][speedUp[j]]+=D[r][speedUp[j]]*D[i][s];
        D[i][s]*=D[r][s];
      }
    }r=-1;s=-1;
    for(int j=0;j<m;j++)if(s<0||ix[s]>ix[j])
      if(D[n+1][j]>eps||(D[n+1][j]>-eps&&D[n][j]>eps))s=j;
    if(s<0)break;
    for(int i=0;i<n;i++)if(D[i][s]<-eps)
      if(r<0||(d=D[r][m]/D[r][s]-D[i][m]/D[i][s])<-eps
         ||(d<eps&&ix[r+m]>ix[i+m]))r=i;
    if(r<0)return VD();//无边界
  }
  if(D[n+1][m]<-eps)return VD();//无解
  VD x(m-1);
  for(int i=m;i<n+m;i++)if(ix[i]<m-1)x[ix[i]]=D[i-m][m];
  return x;//最优值在D[n][m]
}
```

7.15 调和级数

$\sum_{i=1}^{n} \frac{1}{i}$ 在 n 较大时约等于 $\ln n + r$，r 为欧拉常数，约等于 0.5772156649015328。

7.16 曼哈顿距离的变换

$|x_1 - x_2| + |y_1 - y_2| = \max(|(x_1 + y_1) - (x_2 + y_2)|, |(x_1 - y_1) - (x_2 - y_2)|)$

7.17 拉格朗日乘数法

偏导数：

对于一个多元函数，关于某个元 x 的偏导，就是将其他的量看作常数，然后这个多元函数就很显然地变成了一元函数，之后我们对这个一元函数求导，导数就是原多元函数关于 x 的偏导了。

拉格朗日乘数法：

设 $\bigtriangledown f$ 表示 f 的偏导，对于多元函数 f 和约束条件 g，多元函数 f 取到极值，当且仅当存在负数 λ，使得 f 关于每个元的偏导 $\bigtriangledown f$，以及对应的 g 都满足，$\bigtriangledown f = \lambda \times \bigtriangledown g$。

7.18 线性递推逆元

```
for(r[0]=r[1]=1,i=2;i<P;i++){
  r[i]=-r[P%i]*(P/i)%P;
  while(r[i]<0)r[i]+=P;
}
```

7.19 组合数取模

7.19.1 Lucas 定理

```
#include<cstdio>
#define P 10007
int T,n,m,i,f[P],r[P];
int C(int n,int m){return n<m?0:f[n]*r[n-m]%P*r[m]%P;}
int lucas(int n,int m){
  if(n<m)return 0;
  if(!m||n==m)return 1;
  return C(n%P,m%P)*lucas(n/P,m/P)%P;
}
int main(){
  for(r[0]=r[1]=f[0]=f[1]=1,i=2;i<P;i++){
    f[i]=f[i-1]*i%P,r[i]=-r[P%i]*(P/i)%P;
    while(r[i]<0)r[i]+=P;
  }
  for(i=2;i<P;i++)r[i]=r[i]*r[i-1]%P;
  for(scanf("%d",&T);T--;printf("%d\n",lucas(n,m)))scanf("%d%d",&n,&m);
}
```

7.19.2 P 是质数的幂

B 表示质数，P 表示模数，$cal(n)$ 将返回 $n!$，以 $a \times B^b$ 形式表示。

```
ll n,x,y,P,B,s[2000000];
ll exgcd(ll a,ll b){
  if(!b)return x=1,y=0,a;
  ll d=exgcd(b,a%b),t=x;
  return x=y,y=t-a/b*y,d;
}
ll rev(ll a,ll P){exgcd(a,P);while(x<0)x+=P;return x%P;}
ll pow(ll a,ll b,ll P){ll t=1;for(;b;b>>=1LL,a=a*a%P)if(b&1LL)t=t*a%P;return t;}
struct Num{
  ll a,b;
  Num(){a=1,b=0;}
  Num(ll _a,ll _b){a=_a,b=_b;}
  Num operator*(Num x){return Num(a*x.a%P,b+x.b);}
  Num operator/(Num x){return Num(a*rev(x.a,P)%P,b-x.b);}
}now[2];
Num cal(ll n){return n?Num(s[n%P]*pow(s[P],n/P,P)%P,n/B)*cal(n/B):Num(1,0);}
void pre(){for(i=s[0]=1;i<P;i++)if(i%B)s[i]=s[i-1]*i%P;else s[i]=s[i-1];s[P]=s[P-1];}
int main(){
  B=2,P=512,pre();
  cal(n);
}
```

7.20 超立方体相关

n 维超立方体有 $2^{n-i} \times C(n,i)$ 个 i 维元素。

7.21 平面图欧拉公式

对于连通的平面图，有区域数 $F =$ 点数 $E-$ 边数 $V + 1$。

7.22 线性筛

对于线性筛求积性函数，只需考虑质数的情况，质数的幂的情况即可。

```
for(mu[1]=phi[1]=1,i=2;i<N;i++){
  if(!v[i])p[tot++]=i,mu[i]=-1,phi[i]=i-1;
  for(j=0;i*p[j]<N&&j<tot;j++){
    v[i*p[j]]=1;
    if(i%p[j]){
      mu[i*p[j]]=-mu[i];
      phi[i*p[j]]=phi[i]*(p[j]-1);
    }else{
      mu[i*p[j]]=0;
      phi[i*p[j]]=phi[i]*p[j];
      break;
    }
  }
}
```

7.23 数论函数变换

常见积性函数：

$id(i) = i$

$e(i) = [i = 1]$

$d(i) = i$ 的约数个数

$\sigma(i) = i$ 的约数之和

一些性质：

$n = \sum_{d|n} \varphi(d)$

$e(n) = \sum_{d|n} \mu(d)$

$\sum_{i=1}^{n} \sum_{j=1}^{m} i \times j[\gcd(i,j) = d] = \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{m}{d} \rfloor} id \times jd[\gcd(i,j) = 1]$

$\mu \times id = \varphi$

$\mu^2(n) = \sum_{d^2|n} \mu(d)$

$\sum_{i=1}^{n} d(n) = \sum_{i=1}^{n} \lfloor \frac{n}{i} \rfloor$

莫比乌斯反演：

$f(n) = \sum_{d|n} g(d)$

$g(n) = \sum_{d|n} \mu(d) f(\frac{n}{d})$

$f(n) = \sum_{i=1}^{n} t(i) g(\lfloor \frac{n}{i} \rfloor)$

$g(n) = \sum_{i=1}^{n} \mu(i) t(i) f(\lfloor \frac{n}{i} \rfloor)$

7.23.1 疯狂的前缀和

对于快速计算 φ 的前缀和：

$S_n = \frac{n(n+1)}{2} - \sum_{d=1}^{n} S_{\frac{n}{d}}$

对于快速计算 μ 的前缀和：

$S_n = 1 - \sum_{d=1}^{n} S_{\frac{n}{d}}$

n 小于 $maxn^{\frac{2}{3}}$ 时线性筛预处理，否则记忆化搜索，单次计算时间复杂度 $O(n^{\frac{2}{3}})$。

```
#include<cstdio>
#include<algorithm>
#include<map>
#define M 1670000
using namespace std;
typedef unsigned long long ll;
struct P{
  ll x;int y;
  P(){x=y=0;}
  P(ll _x,int _y){x=_x,y=_y;}
  void operator+=(P b){x+=b.x,y+=b.y;}
  void operator-=(P b){x-=b.x,y-=b.y;}
  P operator*(int b){return P(x*b,y*b);}
  void write(){printf("%llu %d\n",x,y);}
}pre[M];
int n,i,j,p[M],tot,N,ask[10],Q;bool v[M];map<int,P>T;
P sum(int n){
  if(n<N)return pre[n];
  if(T.find(n)!=T.end())return T[n];
```

```
  P t=P(((ll)n+1)*n/2,1);
  for(int i=2,j=0;i<=n&&j<n;i=j+1)t-=sum(n/i)*((j=n/(n/i))-i+1);
  return T[n]=t;
}
int main(){
  for(scanf("%d",&Q);i<Q;n=max(n,ask[i++]))scanf("%d",&ask[i]);
  while((ll)N*N*N<n)N++;N*=N;
  for(pre[1]=P(1,1),i=2;i<N;i++){
    if(!v[i])p[tot++]=i,pre[i]=P(i-1,-1);
    for(j=0;i*p[j]<N&&j<tot;j++){
      v[i*p[j]]=1;
      if(i%p[j])pre[i*p[j]]=P(pre[i].x*(p[j]-1),-pre[i].y);
      else{pre[i*p[j]]=P(pre[i].x*p[j],0);break;}
    }
  }
  for(i=2;i<=N;i++)pre[i]+=pre[i-1];
  for(i=0;i<Q;i++)sum(ask[i]).write();
}
```

7.24 快速傅里叶变换

下列模板中 n 必须为 2 的幂。

7.24.1 FFT

```
#include<cstdio>
#include<cmath>
#include<algorithm>
using namespace std;
struct comp{
  double r,i;comp(double _r=0,double _i=0){r=_r;i=_i;}
  comp operator+(const comp x){return comp(r+x.r,i+x.i);}
  comp operator-(const comp x){return comp(r-x.r,i-x.i);}
  comp operator*(const comp x){return comp(r*x.r-i*x.i,r*x.i+i*x.r);}
};
const double pi=acos(-1.0);
void FFT(comp a[],int n,int t){
  for(int i=1,j=0;i<n-1;i++){
    for(int s=n;j^=s>>=1,~j&s;);
    if(i<j)swap(a[i],a[j]);
  }
  for(int d=0;(1<<d)<n;d++){
    int m=1<<d,m2=m<<1;
    double o=pi/m*t;comp _w(cos(o),sin(o));
    for(int i=0;i<n;i+=m2){
      comp w(1,0);
      for(int j=0;j<m;j++){
        comp &A=a[i+j+m],&B=a[i+j],t=w*A;
        A=B-t;B=B+t;w=w*_w;
      }
    }
  }
  if(t==-1)for(int i=0;i<n;i++)a[i].r/=n;
}
```

7.24.2 NTT

998244353 原根为 3，1004535809 原根为 3，786433 原根为 10，880803841 原根为 26。

```
#include<cstdio>
typedef long long ll;
const int N=262144,K=17;
int n,m,i,k;
int a[N+10],b[N+10],tmp[N+10],tmp2[N+10];
int P=998244353,G=3,g[K+1],ng[K+10],inv[N+10],inv2;
int pow(int a,int b){int t=1;for(;b;b>>=1,a=(ll)a*a%P)if(b&1)t=(ll)t*a%P;return t;}
void NTT(int*a,int n,int t){
  for(int i=1,j=0;i<n-1;i++){
    for(int s=n;j^=s>>=1,~j&s;);
    if(i<j){int k=a[i];a[i]=a[j];a[j]=k;}
  }
  for(int d=0;(1<<d)<n;d++){
    int m=1<<d,m2=m<<1,_w=t==1?g[d]:ng[d];
    for(int i=0;i<n;i+=m2)for(int w=1,j=0;j<m;j++){
      int&A=a[i+j+m],&B=a[i+j],t=(ll)w*A%P;
      A=B-t;if(A<0)A+=P;
      B=B+t;if(B>=P)B-=P;
      w=(ll)w*_w%P;
    }
  }
  if(t==-1)for(int i=0,j=inv[n];i<n;i++)a[i]=(ll)a[i]*j%P;
}
//给定a,求a的逆元b
void getinv(int*a,int*b,int n){
  if(n==1){b[0]=pow(a[0],P-2);return;}
  getinv(a,b,n>>1);
  int k=n<<1,i;
  for(i=0;i<n;i++)tmp[i]=a[i];
  for(i=n;i<k;i++)tmp[i]=b[i]=0;
  NTT(tmp,k,1),NTT(b,k,1);
  for(i=0;i<k;i++){
    b[i]=(ll)b[i]*(2-(ll)tmp[i]*b[i]%P)%P;
    if(b[i]<0)b[i]+=P;
  }
  NTT(b,k,-1);
  for(i=n;i<k;i++)b[i]=0;
}
//给定a,求a的对数函数，且a[0]=1
void getln(int*a,int*b,int n){
  getinv(a,tmp2,n);
  int k=n<<1,i;
  for(i=0;i<n-1;i++)b[i]=(ll)a[i+1]*(i+1)%P;
  for(i=n-1;i<k;i++)b[i]=0;
  NTT(b,k,1),NTT(tmp2,k,1);
  for(i=0;i<k;i++)b[i]=(ll)b[i]*tmp2[i]%P;
  NTT(b,k,-1);
  for(i=n-1;i;i--)b[i]=(ll)b[i-1]*inv[i]%P;b[0]=0;
}
//给定a,求a的指数函数，且a[0]=0
void getexp(int*a,int*b,int n){
  if(n==1){b[0]=1;return;}
  getexp(a,b,n>>1);
```

```
  getln(b,tmp,n);
  int k=n<<1,i;
  for(i=0;i<n;i++){tmp[i]=a[i]-tmp[i];if(tmp[i]<0)tmp[i]+=P;}
  if((++tmp[0])==P)tmp[0]=0;
  for(i=n;i<k;i++)tmp[i]=b[i]=0;
  NTT(tmp,k,1),NTT(b,k,1);
  for(i=0;i<k;i++)b[i]=(ll)b[i]*tmp[i]%P;
  NTT(b,k,-1);
  for(i=n;i<k;i++)b[i]=0;
}
//给定a,求a的平方根，b且a[0]=1
void getroot(int*a,int*b,int n){
  if(n==1){b[0]=1;return;}
  getroot(a,b,n>>1);
  getinv(b,tmp2,n);
  int k=n<<1,i;
  for(i=0;i<n;i++)tmp[i]=a[i];
  for(i=n;i<k;i++)tmp[i]=b[i]=0;
  NTT(tmp,k,1),NTT(b,k,1),NTT(tmp2,k,1);
  for(i=0;i<k;i++)b[i]=((ll)b[i]*b[i]+tmp[i])%P*inv2%P*tmp2[i]%P;
  NTT(b,k,-1);
  for(i=n;i<k;i++)b[i]=0;
}
int main(){
  for(g[K]=pow(G,(P-1)/N),ng[K]=pow(g[K],P-2),i=K-1;~i;i--)
    g[i]=(ll)g[i+1]*g[i+1]%P,ng[i]=(ll)ng[i+1]*ng[i+1]%P;
  for(inv[1]=1,i=2;i<=N;i++)inv[i]=(ll)(P-inv[P%i])*(P/i)%P;inv2=inv[2];
  scanf("%d%d",&n,&m);
  for(k=1;k<=n;k<<=1);
  for(i=0;i<=n;i++)scanf("%d",&a[i]);
  getln(a,b,k);
  for(i=0;i<k;i++)b[i]=(ll)b[i]*m%P;
  getexp(b,a,k);
  for(i=0;i<k;i++)printf("%d ",a[i]);
}
```

7.24.3 多项式求幂

找到最低的不为 0 的那一项，把整个多项式除以它，然后 ln+exp 在 $O(n\log n)$ 时间内求出幂，再乘回去即可。

7.24.4 拉格朗日反演

若 F 与 G 互为复合逆，即互为反函数，根据拉格朗日反演可得，
$[x^n]F(x)=[x^{n-1}]\frac{(\frac{x}{G(x)})^n}{n}$，其中 $[x^n]$ 表示第 n 项的系数。

7.25 蔡勒公式

$w=(\lfloor\frac{c}{4}\rfloor-2c+y+\lfloor\frac{y}{4}\rfloor+\lfloor\frac{13(m+1)}{5}\rfloor+d-1) \bmod 7$

w：0 星期日，1 星期一，2 星期二，3 星期三，4 星期四，5 星期五，6 星期六。

c：世纪减 1（年份前两位数）。

y：年（后两位数）。

m：月（$3 \le m \le 14$，即在蔡勒公式中，1、2 月要看作上一年的 13、14 月来计算）。

d：日。

7.26 皮克定理

给定顶点坐标均是整点（或正方形格点）的简单多边形，皮克定理说明了其面积 S 和内部格点数目 n、边上格点数目 s 的关系：$S = n + \frac{s}{2} - 1$。

7.27 组合数 lcm

$$(n+1)lcm(C(n,0), C(n,1), ..., C(n,k)) = lcm(n+1, n, n-1, n-2, ..., n-k+1)$$

7.28 区间 lcm 的维护

对于一个数，将其分解质因数，若有因子 p^k，那么拆分出 k 个数 $p, p^2, ..., p^k$，权值都为 p，那么查询区间 $[l, r]$ 内所有数的 lcm 的答案 = 所有在该区间中出现过的数的权值之积，可持久化线段树维护即可。

7.29 中国剩余定理

n 个同余方程，第 i 个为 $x \equiv b[i] \pmod{a[i]}$，且 $a[i]$ 两两互质，那么可以通过中国剩余定理合并。

```
ll rev(ll a,ll p){exgcd(a,p);while(x<0)x+=p;return x;}
ll CRT(ll*a,ll*b,int n){
  ll ans,P=1;
  for(int i=0;i<n;i++)P*=a[i];
  for(int i=0;i<n;i++)ans=(ans+(P/a[i])*rev(P/a[i],a[i])%P*b[i]%P)%P;
  while(ans<0)ans+=P;
  return ans;
}
```

7.30 欧拉函数

```
int phi(int n){
  int t=1,i;
  if(!(n&1))for(n>>=1;!(n&1);n>>=1,t<<=1);
  for(i=3;i*i<=n;i+=2)if(n%i==0)for(n/=i,t*=i-1;n%i==0;n/=i,t*=i);
  if(n>1)t*=n-1;
  return t;
}
```

7.31 快速沃尔什变换

```
void FWT(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];
    //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];
    //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;
  }
}
```

7.31.1 K 进制异或卷积

```
const int N=540000,K=3;
int n,m,P,i,w[K];
inline int po(int a,int b,int P){
  int t=1;
  for(;b;b>>=1,a=1LL*a*a%P)if(b&1)t=1LL*t*a%P;
  return t;
}
int exgcd(int a,int b,int&x,int&y){
  if(!b)return x=1,y=0,a;
  int d=exgcd(b,a%b,x,y),t=x;
  return x=y,y=t-a/b*y,d;
}
inline int inv(int a,int P){
  int x,y;
  exgcd(a,P,x,y);
  return (x%P+P)%P;
}
void FWT(int*a,int n,int t){
  static int b[N];
  for(int l=K;l<=n;l*=K){
    for(int i=0;i<n;i++)b[i]=a[i];
    for(int i=0,h=l/K;i<n;i+=l)for(int j=0;j<h;j++)for(int k=0;k<K;k++){
      int v=0;
      for(int o=0;o<K;o++)v=(1LL*b[i+j+o*h]*w[(k*o%K*t+K)%K]+v)%P;
      a[i+j+k*h]=v;
    }
  }
  if(t==-1){
    int o=inv(n,P);
    for(int i=0;i<n;i++)a[i]=1LL*a[i]*o%P;
  }
}
inline int getG(int n,int phi,int K){
  int i,j,t=0;
  static int q[100000];
  for(i=2;1LL*i*i<phi;i++)if(phi%i==0)q[t++]=i,q[t++]=phi/i;
  for(i=2;;i++){
    for(j=0;j<t;j++)if(po(i,q[j],n)==1)break;
    if(j==t)return po(i,phi/K,n);
  }
```

```
}
int CRT(int*a,int*b,int n){
  int ans=0,P=1;
  for(int i=0;i<n;i++)P*=a[i];
  for(int i=0;i<n;i++)ans=(1LL*P/a[i]*inv(P/a[i],a[i])%P*b[i]+ans)%P;
  return (ans%P+P)%P;
}
int getroot(int P,int K){
  static int a[100],b[100];
  int cnt=0;
  for(int i=2;i*i<=P;i++)if(P%i==0){
    int t=1;
    while(P%i==0)P/=i,t*=i;
    a[cnt]=t;
    b[cnt]=getG(t,t/i*(i-1),K);
    cnt++;
  }
  if(P>1){
    a[cnt]=P;
    b[cnt]=getG(P,P-1,K);
    cnt++;
  }
  return CRT(a,b,cnt);
}
int main(){
  read(m),read(P);
  for(n=1,i=0;i<m;i++)n*=3;
  w[0]=1;
  w[1]=getroot(P,K);
  for(i=2;i<K;i++)w[i]=1LL*w[i-1]*w[1]%P;
}
```

7.32 幂和

$$\sum_{i=1}^{n} i^1 = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$$

$$\sum_{i=1}^{n} i^3 = \left[\frac{n(n+1)}{2}\right]^2 = \frac{1}{4}n^4 + \frac{1}{2}n^3 + \frac{1}{4}n^2$$

$$\sum_{i=1}^{n} i^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30} = \frac{1}{5}n^5 + \frac{1}{2}n^4 + \frac{1}{3}n^3 - \frac{1}{30}n$$

$$\sum_{i=1}^{n} i^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12} = \frac{1}{6}n^6 + \frac{1}{2}n^5 + \frac{5}{12}n^4 - \frac{1}{12}n^2$$

$$\sum_{i=1}^{n} i^6 = \frac{n(n+1)(2n+1)(3n^4+6n^3-3n+1)}{42} = \frac{1}{7}n^7 + \frac{1}{2}n^6 + \frac{1}{2}n^5 - \frac{1}{6}n^3 + \frac{1}{42}n$$

```
//求sum_{i=1}^n i^m
```

```
#include<cstdio>
const int N=1000010,P=1000000007;
int n,m,i,j,v[N],p[N],tot,f[N],r[N],b[N],A[N],B[N],ret,tmp;
int pow(int a,int b){int t=1;for(;b;b>>=1,a=1LL*a*a%P)if(b&1)t=1LL*t*a%P;return t;}
int main(){
  scanf("%d%d",&n,&m);m++;
  for(f[1]=1,i=2;i<=m;i++){
    if(!v[i])f[i]=pow(i,m-1),p[tot++]=i;
    for(j=0;j<tot;j++){
      if(i*p[j]>m)break;
      v[i*p[j]]=1,f[i*p[j]]=1LL*f[i]*f[p[j]]%P;
      if(i%p[j]==0)break;
    }
  }
  for(i=2;i<=m;i++)f[i]=(f[i-1]+f[i])%P;
  if((n%=P)<=m)return printf("%d",f[n]),0;
  for(r[0]=r[1]=1,i=2;i<=m;i++)r[i]=1LL*(P-r[P%i])*(P/i)%P;
  for(i=2;i<=m;i++)r[i]=1LL*r[i-1]*r[i]%P;
  for(i=1;i<=m+1;i++)b[i]=(n-i+1+P)%P;
  for(A[0]=B[m+2]=i=1;i<=m+1;i++)A[i]=1LL*A[i-1]*b[i]%P;
  for(i=m+1;i;i--)B[i]=1LL*B[i+1]*b[i]%P;
  for(i=0;i<=m;i++){
    tmp=1LL*f[i]*r[m-i]%P*r[i]%P*A[i]%P*B[i+2]%P;
    if((m-i)&1)ret=(ret-tmp+P)%P;else ret=(ret+tmp)%P;
  }
  printf("%d",ret);
}
```

7.33 斯特林数

7.33.1 第一类斯特林数

第一类 Stirling 数 $S(p,k)$ 的一个的组合学解释是：将 p 个物体排成 k 个非空循环排列的方法数。

$S(p,k)$ 的递推公式：$S(p,k)=(p-1)S(p-1,k)+S(p-1,k-1), 1\le k\le p-1$

边界条件：$S(p,0)=0, p\ge 1 \quad S(p,p)=1, p\ge 0$

7.33.2 第二类斯特林数

第二类 Stirling 数 $S(p,k)$ 的一个的组合学解释是：将 p 个物体划分成 k 个非空的不可辨别的（可以理解为盒子没有编号）集合的方法数。

$S(p,k)$ 的递推公式：$S(p,k)=kS(p-1,k)+S(p-1,k-1), 1\le k\le p-1$

边界条件：$S(p,0)=0, p\ge 1 \quad S(p,p)=1, p\ge 0$

也有卷积形式：

$$S(n,m)=\frac{1}{m!}\sum_{k=0}^{m}(-1)^k C(m,k)(m-k)^n=\sum_{k=0}^{m}\frac{(-1)^k(m-k)^n}{k!(m-k)!}=\sum_{k=0}^{m}\frac{(-1)^k}{k!}\times\frac{(m-k)^n}{(m-k)!}$$

7.34 各种情况下小球放盒子的方案数

k 个球	m 个盒子	是否允许有空盒子	方案数
各不相同	各不相同	是	m^k
各不相同	各不相同	否	$m!\text{Stirling2}(k,m)$
各不相同	完全相同	是	$\sum_{i=1}^{m}\text{Stirling2}(k,i)$
各不相同	完全相同	否	$\text{Stirling2}(k,m)$
完全相同	各不相同	是	$C(m+k-1,k)$
完全相同	各不相同	否	$C(k-1,m-1)$
完全相同	完全相同	是	$\frac{1}{(1-x)(1-x^2)...(1-x^m)}$ 的 x^k 项的系数
完全相同	完全相同	否	$\frac{x^m}{(1-x)(1-x^2)...(1-x^m)}$ 的 x^k 项的系数

7.35 错排公式

$D_1=0, D_2=1, D_n=(n-1)(D_{n-2}+D_{n-1})$

7.36 Prufer 编码

对于一棵有 n 个点的带标号的无根树，设 $d[i]$ 为 i 点的度数，那么可以用一个唯一的长度为 $n-2$ 的序列来表示这棵树，其中 i 出现了 $d[i]-1$ 次。若每个点的度数可行的取值是一个集合，则可以通过指数型生成函数来完成计数。

7.37 二项式反演

$$
\begin{aligned}
f(n) &= \sum_{k=0}^{n} C(n,k)g(k) \\
g(n) &= \sum_{k=0}^{n} (-1)^{n-k}C(n,k)f(k)
\end{aligned}
$$

7.38 x^k 的转化

$$x^k=\sum_{i=1}^{k} Stirling2(k,i)i!C(x,i)$$

7.39 快速计算素数个数

$N \leq 10^{12}$。

```
#include<cstdio>
#include<algorithm>
#include<vector>
#include<cmath>
#define PB push_back
using namespace std;
```

```
typedef long long ll;
vector<int>vis,prime,g;vector<vector<int> >f;
ll sqr(ll x){return x*x;}
int getsqrt(ll n){
  ll t=sqrt(n);
  while(sqr(t+1)<=n)t++;
  return t;
}
int getcbrt(ll n){
  ll t=cbrt(n);
  while((t+1)*(t+1)*(t+1)<=n)t++;
  return t;
}
void sieve(int n){
  vis.assign(n+1,0);
  prime.clear();
  for(int i=2;i<=n;i++){
    if(!vis[i]){
      prime.PB(i);
      vis[i]=i;
    }
    for(int j=0;j<(int)(prime.size());j++){
      if(prime[j]>vis[i]||prime[j]>n/i)break;
      vis[i*prime[j]]=prime[j];
    }
  }
}
ll PI(ll n);
ll F(ll n,int m){
  if(!m)return n;
  if(prime[m]>n)return 0;
  if(m<(int)(f.size())&&n<(int)(f[m].size()))return f[m][n];
  if(sqr(prime[m])>n)return PI(n)-m+1;
  return F(n,m-1)-F(n/prime[m-1],m-1);
}
ll PI(ll n){
  if(n<=prime.back())return g[n];
  int i=PI(getcbrt(n-1)+1);ll tmp=F(n,i)+i-1;
  while(1){
    ll t=prime[i];
    if(t*t>n)break;
    tmp-=PI(n/t)-i;
    i++;
  }
  return tmp;
}
void init(ll n){
  sieve(getsqrt(n+1)*2);
  g.assign(prime.back()+1,0);
  int i,j;
  for(i=j=0;i<=prime.back();g[i++]=j)if(i==prime[j])j++;
  int A=131072,B=min((int)prime.size(),64);
  f.assign(B,vector<int>(A));
  for(j=0;j<B;j++)for(i=0;i<A;i++)if(j==0)f[j][i]=i;
  else f[j][i]=f[j-1][i]-f[j-1][i/prime[j-1]];
}
int main(){
```

```
  init(100000000000LL);
  ll n;
  while(~scanf("%lld",&n))printf("%lld\n",PI(n));
}
```

7.40 素数的幂的和

求 n 以内所有素数的 k 次方之和。

```
#include<cstdio>
#include<cmath>
#include<vector>
using namespace std;
typedef long long ll;
// return the sum of p^k for all p <= m, where m is in form floor(n / i)
// for m <= sqrt{n}, stored in ssum[m]; for m > sqrt{n} stored in lsum[n / m]
// note: if you need all correct value of ssum and lsum, please remove "mark"
// and make "delta" always be 1
inline ll sub_mod(ll x,ll y,ll mod){return(x-y+mod)%mod;}
inline ll mul_mod(ll a,ll b,ll mod){
  if(mod<int(2e9))return a*b%mod;
  ll k=(ll)((long double)a*b/mod);
  ll res=a*b-k*mod;
  res%=mod;
  if(res<0)res+=mod;
  return res;
}
inline ll pow_mod(ll a,ll n,ll m){
  ll res=1;
  for(a%=m;n;n>>=1){
    if(n&1)res=mul_mod(res,a,m);
    a=mul_mod(a,a,m);
  }
  return res;
}
pair<vector<ll>,vector<ll>>prime_count(ll n,ll k,ll mod){
  auto pow_sum=[](ll n,ll k,ll mod){
    if(k==0)return n;
    if(k==1)return n*(n+1)/2%mod;
  };
  const ll v=static_cast<ll>(sqrt(n));
  vector<ll>ssum(v+1),lsum(v+1);
  vector<bool>mark(v+1);
  for(int i=1;i<=v;++i){
    ssum[i]=pow_sum(i,k,mod)-1;
    lsum[i]=pow_sum(n/i,k,mod)-1;
  }
  for(ll p=2;p<=v;++p){
    if(ssum[p]==ssum[p-1])continue;
    ll psum=ssum[p-1],q=p*p,ed=min(v,n/q);
    ll pk=pow_mod(p,k,mod);
    int delta=(p&1)+1;
    for(int i=1;i<=ed;i+=delta)if(!mark[i]){
      ll d=i*p;
      if(d<=v){
```

```
        lsum[i]=sub_mod(lsum[i],sub_mod(lsum[d],psum,mod)*pk%mod,mod);
      }else{
        lsum[i]=sub_mod(lsum[i],sub_mod(ssum[n/d],psum,mod)*pk%mod,mod);
      }
    }
    for(ll i=q;i<=ed;i+=p*delta)mark[i]=1;
    for(ll i=v;i>=q;--i){
      ssum[i]=sub_mod(ssum[i],sub_mod(ssum[i/p],psum,mod)*pk%mod,mod);
    }
  }
  return {move(ssum),move(lsum)};
}
int main(){
  ll n,k,mod;
  scanf("%lld%lld%lld",&n,&k,&mod);
  auto it=prime_count(n,k,mod);
  printf("%lld",it.second[1]);
}
```

7.41 求不超过 n 的模 P 为每个数的素数个数

```
#include<cstdio>
#include<cmath>
#include<vector>
#include<algorithm>
typedef long long ll;
const int N=200010,M=12;
ll ss[N][M],ls[N][M];int p[N],pc;
void sieve(){
  int i,j;
  for(i=2;i<N;i++){
    if(!p[i])p[pc++]=i;
    for(j=0;j<pc&&i*p[j]<N;j++){
      p[i*p[j]]=1;
      if(i%p[j]==0)break;
    }
  }
}
void prime_count(ll n,ll m){
  const ll v=static_cast<ll>(sqrt(n));
  int i,j;
  for(i=1;i<=v;i++){
    for(j=0;j<m;j++){
      ss[i][j]=i>=j?(i-j)/m+1:0;
      ls[i][j]=n/i>=j?(n/i-j)/m+1:0;
    }
    --ss[i][0],--ls[i][0];
    --ss[i][1],--ls[i][1];
  }
  std::vector<bool>mark(v+1);
  for(int it=0;it<pc&&::p[it]<=v;it++){
    ll p=::p[it],q=p*p,ed=std::min(v,n/q);
    int delta=(p&1)+1;
    for(i=1;i<=ed;i+=delta)if(!mark[i]){
      ll d=i*p;
```

```
      if(d<=v)for(j=0;j<m;j++)ls[i][p*j%m]-=ls[d][j]-ss[p-1][j];
      else for(j=0;j<m;j++)ls[i][p*j%m]-=ss[n/d][j]-ss[p-1][j];
    }
    for(ll i=q;i<=ed;i+=p*delta)mark[i]=1;
    for(ll i=v;i>=q;i--)for(j=0;j<m;j++)ss[i][p*j%m]-=ss[i/p][j]-ss[p-1][j];
  }
  for(i=0;i<m;i++)printf("%lld\n",ls[1][i]);
}
int main(){
  sieve();
  ll n,m;
  scanf("%lld%lld",&n,&m);
  prime_count(n,m);
}
```

7.42 Best Theorem

Best Theorem：有向图中以 i 为起点的欧拉回路个数为以 i 为根的树形图个数 $\times$((每个点度数 -1)!)。

Matrix Tree Theorem：以 i 为根的树形图个数 $=$ 基尔霍夫矩阵去掉第 i 行第 i 列的行列式。

无向图：对于边 $(x,y), a[x][x]++, a[y][y]++, a[x][y]--, a[y][x]--$。

从某个点 i 出发并回到 i 的欧拉回路个数 $=$ 以 i 为起点的欧拉回路个数 $\times i$ 的度数。

```
const int N=110,M=200010,P=1000003;
int n,m,i,j,k,x,y,in[N],ou[N],vis[N],g[N][N];ll f[M],a[N][N],ans;
void dfs(int x){
  vis[x]=++m;
  for(int i=1;i<=n;i++)if(g[x][i]&&!vis[i])dfs(i);
}
ll det(int n){
  ll ans=1;bool flag=1;
  for(i=1;i<=n;i++)for(j=1;j<=n;j++)a[i][j]=(a[i][j]%P+P)%P;
  for(i=1;i<=n;i++){
    for(j=i+1;j<=n;j++)while(a[j][i]){
      ll t=a[i][i]/a[j][i];
      for(k=i;k<=n;k++)a[i][k]=(a[i][k]+P-t*a[j][k]%P)%P;
      for(k=i;k<=n;k++)swap(a[i][k],a[j][k]);
      flag^=1;
    }
    ans=ans*a[i][i]%P;
    if(!ans)return 0;
  }
  if(!flag)ans=P-ans;
  return ans;
}
int solve(){
  for(m=0,i=1;i<=n;i++)in[i]=ou[i]=vis[i]=0;
  for(i=0;i<=n;i++)for(j=0;j<=n;j++)a[i][j]=g[i][j]=0;
  int ed=0;
  for(i=1;i<=n;i++)for(read(k);k--;g[i][j]++)read(j),ed++;
  for(dfs(i=1);i<=n;i++)if(!vis[i]&&g[i])return 0;
  for(i=1;i<=n;i++)for(j=1;j<=n;j++)if(g[i][j]){
```

```
    x=vis[i],y=vis[j];
    ou[x]+=g[i][j];in[y]+=g[i][j];
    a[x-1][y-1]-=g[i][j],a[x-1][x-1]+=g[i][j];
  }
  for(i=1;i<=m;i++)if(in[i]!=ou[i])return 0;
  if(m==1)return f[g[1][1]];
  ans=det(m-1)*in[1];
  for(i=1;i<=m;i++)ans=ans*f[in[i]-1]%P;
  return ans;
}
int main(){
  for(f[0]=i=1;i<M;i++)f[i]=f[i-1]*i%P;
  while(1){
    read(n);
    if(!n)return 0;
    printf("%d\n",solve());
  }
}
```

7.43 法雷序列

求两分数间分母最小的分数，$1 \le a, b, c, d \le 10^9$。

```
#include<cstdio>
#include<algorithm>
using namespace std;
typedef long long ll;
typedef pair<ll,ll>P;
ll a,b,c,d,t;
ll gcd(ll a,ll b){return b?gcd(b,a%b):a;}
P cal(ll a,ll b,ll c,ll d){
  ll x=a/b+1;
  if(x*d<c)return P(x,1);
  if(!a)return P(1,d/c+1);
  if(a<=b&&c<=d){
    P t=cal(d,c,b,a);
    swap(t.first,t.second);
    return t;
  }
  x=a/b;
  P t=cal(a-b*x,b,c-d*x,d);
  t.first+=t.second*x;
  return t;
}
int main(){
  while(~scanf("%lld%lld%lld%lld",&a,&b,&c,&d)){
    t=gcd(a,b),a/=t,b/=t;
    t=gcd(c,d),c/=t,d/=t;
    P p=cal(a,b,c,d);
    printf("%lld/%lld\n",p.first,p.second);
  }
}
```

7.44 分数拟合小数

给定一个 $[0,1)$ 之间的 n 位小数 $0.r$，拟合出一个分数。注意不要爆 long long。

```
typedef pair<ll,ll>P;
ll n,r,a,b,c,d,t;
P cal(ll a,ll b,ll c,ll d){
  ll x=a/b+1;
  if(x*d<c)return P(x,1);
  if(!a)return P(1,d/c+1);
  if(a<=b&&c<=d){
    P t=cal(d,c,b,a);
    swap(t.first,t.second);
    return t;
  }
  x=a/b;
  P t=cal(a-b*x,b,c-d*x,d);
  t.first+=t.second*x;
  return t;
}
int main(){
  while(~scanf("%lld 0.%lld",&n,&r)){
    if(!r){puts("1");continue;}
    for(t=10;n--;t*=10);
    for(a=r*10-5,b=t;a%5==0&&b%5==0;a/=5,b/=5);
    for(c=r*10+5,d=t;c%5==0&&d%5==0;c/=5,d/=5);
    P p=cal(a,b,c,d);
    printf("%lld\n",min(b,p.second));
  }
}
```

7.45 FFT 模任意质数

```
const int N=524300,P=1000003,M=1000;
int n,i,j,k,pos[N],A[N],B[N],C[N];
namespace FFT{
struct comp{
  long double r,i;comp(long double _r=0,long double _i=0){r=_r;i=_i;}
  comp operator+(const comp x){return comp(r+x.r,i+x.i);}
  comp operator-(const comp x){return comp(r-x.r,i-x.i);}
  comp operator*(const comp x){return comp(r*x.r-i*x.i,r*x.i+i*x.r);}
  comp conj(){return comp(r,-i);}
}A[N],B[N];
int a0[N],b0[N],a1[N],b1[N];
const long double pi=acos(-1.0);
void FFT(comp a[],int n,int t){
  for(int i=1;i<n;i++)if(i<pos[i])swap(a[i],a[pos[i]]);
  for(int d=0;(1<<d)<n;d++){
    int m=1<<d,m2=m<<1;
    long double o=pi*2/m2*t;comp _w(cos(o),sin(o));
    for(int i=0;i<n;i+=m2){
      comp w(1,0);
      for(int j=0;j<m;j++){
        comp&A=a[i+j+m],&B=a[i+j],t=w*A;
        A=B-t;B=B+t;w=w*_w;
```

```
      }
    }
  }
  if(t==-1)for(int i=0;i<n;i++)a[i].r/=n;
}
void FFT(comp a[],int n,int o){//预处理单位根提升精度
  for(int i=0;i<n;i++)w[i]=comp(cos(pi*i/n),sin(pi*i/n));
  for(int i=0;i<n;i++)ww[i]=w[i],ww[i].i*=-1;
  for(int i=1;i<n;i++)if(i<pos[i])swap(a[i],a[pos[i]]);
  for(int d=0,k=__builtin_ctz(n);(1<<d)<n;d++){
    int m=1<<d,m2=m<<1;
    for(int i=0;i<n;i+=m2)for(int j=0;j<m;j++){
      comp&A=a[i+j+m],&B=a[i+j],t=(o==1?w[j<<(k-d)]:ww[j<<(k-d)])*A;
      A=B-t;B=B+t;
    }
  }
  if(o==-1)for(int i=0;i<n;i++)a[i].r/=n;
}
void mul(int*a,int*b,int*c){//c=a*b
  int i,j;
  for(i=0;i<k;i++)A[i]=comp(a[i],b[i]);
  FFT(A,k,1);
  for(i=0;i<k;i++){
    j=(k-i)&(k-1);
    B[i]=(A[i]*A[i]-(A[j]*A[j]).conj())*comp(0,-0.25);
  }
  FFT(B,k,-1);
  for(i=0;i<k;i++)c[i]=((long long)(B[i].r+0.5))%P;
}
//输入两个多项式，求a*b mod P，保存在c中，c不能为a或b
void mulmod(int*a,int*b,int*c){
  int i;
  for(i=0;i<k;i++)a0[i]=a[i]/M,b0[i]=b[i]/M;
  for(mul(a0,b0,a0),i=0;i<k;i++){
    c[i]=1LL*a0[i]*M*M%P;
    a1[i]=a[i]%M,b1[i]=b[i]%M;
  }
  for(mul(a1,b1,a1),i=0;i<k;i++){
    c[i]=(a1[i]+c[i])%P,a0[i]=(a0[i]+a1[i])%P;
    a1[i]=a[i]/M+a[i]%M,b1[i]=b[i]/M+b[i]%M;
  }
  for(mul(a1,b1,a1),i=0;i<k;i++)c[i]=(1LL*M*(a1[i]-a0[i]+P)+c[i])%P;
}
}
int main(){
  read(n);
  for(k=1;k<n;k<<=1);k<<=1;
  j=__builtin_ctz(k)-1;
  for(i=0;i<k;i++)pos[i]=pos[i>>1]>>1|((i&1)<<j);
  for(i=0;i<n;i++)read(A[i]);
  for(i=0;i<n;i++)read(B[i]);
  FFT::mulmod(A,B,C);
  for(i=0;i<n+n-1;i++)printf("%d ",C[i]);
}
```

7.46 拉格朗日四平方和定理

给定一个整数 $N(0 \le N \le 10^{18})$，求 N 最少可以拆成多少个完全平方数的和。

```
#define C 2730
#define S 3
ll n;map<ll,bool>f;
ll gcd(ll a,ll b){return b?gcd(b,a%b):a;}
ll mul(ll a,ll b,ll n){return(a*b-(ll)(a/(long double)n*b+1e-3)*n+n)%n;}
ll pow(ll a,ll b,ll n){
  ll d=1;
  a%=n;
  while(b){
    if(b&1)d=mul(d,a,n);
    a=mul(a,a,n);
    b>>=1;
  }
  return d;
}
bool check(ll a,ll n){
  ll m=n-1,x,y;int i,j=0;
  while(!(m&1))m>>=1,j++;
  x=pow(a,m,n);
  for(i=1;i<=j;x=y,i++){
    y=pow(x,2,n);
    if((y==1)&&(x!=1)&&(x!=n-1))return 1;
  }
  return y!=1;
}
bool miller_rabin(int times,ll n){
  ll a;
  if(n==1)return 0;
  if(n==2)return 1;
  if(!(n&1))return 0;
  while(times--)if(check(rand()%(n-1)+1,n))return 0;
  return 1;
}
ll pollard_rho(ll n,int c){
  ll i=1,k=2,x=rand()%n,y=x,d;
  while(1){
    i++,x=(mul(x,x,n)+c)%n,d=gcd(y-x,n);
    if(d>1&&d<n)return d;
    if(y==x)return n;
    if(i==k)y=x,k<<=1;
  }
}
void findfac(ll n,int c){
  if(n==1)return;
  if(miller_rabin(S,n)){
    if(n%4==3)f[n]^=1;
    return;
  }
  ll m=n;
  while(m==n)m=pollard_rho(n,c--);
  findfac(m,c),findfac(n/m,c);
}
int solve(ll n){
```

```
  ll t=(ll)sqrt(n);
  for(ll i=t-2;i<=t+2;i++)if(i*i==n)return 1;
  t=n;
  while(1){
    if(t%8==7)return 4;
    if(t%4==0)t/=4;else break;
  }
  f.clear();
  findfac(n,C);
  for(map<ll,bool>::iterator it=f.begin();it!=f.end();it++)if(it->second)return 3;
  return 2;
}
int main(){
  int T;
  scanf("%d",&T);
  while(T--){
    scanf("%lld",&n);
    printf("%d\n",solve(n));
  }
}
```

7.47 Pell 方程

$x^2 - dy^2 = 1 (1 \le d \le 10^5)$, 求 (x, y) 的最小正整数解。

```
n=int(raw_input())
j=1
while j*j<n:
  j+=1
if j*j==n:
  print j,1
if j*j>n:
  p=[0 for i in range(0,1001)]
  q=[0 for i in range(0,1001)]
  a=[0 for i in range(0,1001)]
  g=[0 for i in range(0,1001)]
  h=[0 for i in range(0,1001)]
  p[1]=q[0]=h[1]=1
  p[0]=q[1]=g[1]=0
  a[2]=j-1
  i=2
  while 1:
    g[i]=-g[i-1]+a[i]*h[i-1]
    h[i]=(n-g[i]*g[i])/h[i-1]
    a[i+1]=(g[i]+a[2])/h[i]
    p[i]=a[i]*p[i-1]+p[i-2]
    q[i]=a[i]*q[i-1]+q[i-2]
    if(p[i]*p[i]-n*q[i]*q[i]==1):
      print p[i],q[i]
      break
    i+=1
```

7.48 O(1) 求 GCD

```cpp
#include<cstdio>
const int N=1000005,M=1005;
int n,m,i,j,k,f[M][M],d[N][3],p[N/10],tot;bool v[N];
int gcd_normal(int x,int y){
  if(!x||!y)return x+y;
  return gcd_normal(y,x%y);
}
inline int gcdfast(int a,int b){
  if(!a||!b)return a+b;
  int c=1;
  while(a-b){
    if(a&1){
      if(b&1){
        if(a>b)a=(a-b)>>1;else b=(b-a)>>1;
      }else b>>=1;
    }else{
      if(b&1)a>>=1;else c<<=1,a>>=1,b>>=1;
    }
  }
  return c*a;
}
inline int gcdO1(int x,int y){
  if(!x||!y)return x+y;
  int t=1;
  for(int*j=d[x],i=0,k;i<3&&y>1;i++,j++){
    if(*j==1)continue;
    else if(*j<=m)k=f[*j][y%*j];
    else if(y%*j==0)k=*j;
    else continue;
    t*=k,y/=k;
  }
  return t;
}
int main(){
  n=1000000,m=1000;
  for(i=0;i<=m;i++)f[0][i]=f[i][0]=f[i][i]=i;
  for(i=2;i<=m;i++)for(j=1;j<i;j++)f[i][j]=f[j][i]=f[i-j][j];
  for(d[1][0]=d[1][1]=d[1][2]=1,i=2;i<=n;i++){
    if(!v[i])p[tot++]=i,d[i][0]=i,d[i][1]=d[i][2]=1;
    for(j=0;j<tot;j++){
      if(i*p[j]>n)break;
      v[k=i*p[j]]=1;
      d[k][0]=d[i][0],d[k][1]=d[i][1],d[k][2]=d[i][2];
      if(d[k][0]*p[j]<=m)d[k][0]*=p[j];
      else if(d[k][1]*p[j]<=m)d[k][1]*=p[j];
      else d[k][2]*=p[j];
      if(i%p[j]==0)break;
    }
  }
}
```

7.49 拉格朗日插值法

传入 $y=f(x)$ 上的 n 个点, 拟合出对应的一元 $n-1$ 次方程, 返回各项系数。

类型需支持加, 减, 乘, 除, 取反, 加等于这六种操作。

```
template<typename T>
std::vector<T> interpolation(const T x[], const T y[], int n){
  std::vector<T> u(y, y + n), ret(n), sum(n);
  ret[0] = u[0], sum[0] = 1;
  for (int i = 1; i < n; ++i) {
    for (int j = n - 1; j >= i; --j) {
      u[j] = (u[j] - u[j - 1]) / (x[j] - x[j - i]);
    }
    for (int j = i; j; --j) {
      sum[j] = -sum[j] * x[i - 1] + sum[j - 1];
      ret[j] += sum[j] * u[i];
    }
    sum[0] = -sum[0] * x[i - 1];
    ret[0] += sum[0] * u[i];
  }
  return ret;
}
```

7.50 二次剩余

求解方程: $x^2 \equiv n \pmod p$，无解返回 -1, 否则返回其中一个解 r, 另一个解是 $p-r$。

```
LL ToneLLi_Shanks(LL n, LL p) {
  if (n == 0) return 0;
  if (p == 2) return (n & 1) ? 1 : -1;
  if (pow_mod(n, p >> 1, p) != 1) return -1;
  if (p & 2) return pow_mod(n, p + 1 >> 2, p);
  int s = __builtin_ctzll(p ^ 1);
  LL q = p >> s, z = 2;
  for (; pow_mod(z, p >> 1, p) == 1; ++z);
  LL c = pow_mod(z, q, p);
  LL r = pow_mod(n, q + 1 >> 1, p);
  LL t = pow_mod(n, q, p), tmp;
  for (int m = s, i; t != 1;) {
    for (i = 0, tmp = t; tmp != 1; ++i) tmp = tmp * tmp % p;
    for (; i < --m;) c = c * c % p;
    r = r * c % p;
    c = c * c % p;
    t = t * c % p;
  }
  return r;
}
```

7.51 一般积性函数求和

$f(p)=A$, $f(p^c)=Poly(c)$，时间复杂度 $O(\frac{n^{\frac{3}{4}}}{\log n})$，要开空间为 $2\sqrt{n}$。

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
typedef long long ll;
```

```
const int N=700000;
int p[N/10+9],tot;bool v[N+9];
int lim,xn,u[N+9],l[N+9];ll n,x[N+9],g[N+9],f[N+9];
const int A=4;
int Poly(int c){return 3*c+1;}
void init(int n){
  for(int i=2;i<=n;i++){
    if(!v[i])p[++tot]=i;
    for(int j=1;j<=tot&&i*p[j]<=n;j++){
      v[i*p[j]]=1;
      if(i%p[j]==0)break;
    }
  }
}
ll sqr(ll x){return x*x;}
int pos(ll x){return x<=lim?x:xn-n/x+1;}
void calg(){
  for(int i=1;i<=xn;i++)g[i]=x[i];
  fill(l+1,l+xn+1,0);
  for(int i=1,k=1;i<=tot;i++){
    ll t=sqr(p[i]);
    while(k<=xn&&x[k]<t)k++;
    for(int j=xn;j>=k;l[j--]=i){
      int d=pos(x[j]/p[i]);
      g[j]-=g[d]-(min(u[d],i)-l[d]-1);
    }
  }
  for(int i=1;i<=xn;i++)g[i]-=min(u[i],tot+1)-l[i]-1;
  fill(g+1,g+lim+1,0);
  for(int i=xn;i>lim;i--)g[i]--,g[i]*=A;
  for(int i=1;i<=xn;++i)g[i]++;
}
void calf(){
  fill(f+1,f+xn+1,1);
  fill(l+1,l+xn+1,0);
  for(int i=tot,k=xn;i;i--){
    ll t=sqr(p[i]);
    while(k>1&&x[k-1]>=t)k--;
    for(int j=xn;j>=k;j--){
      ll pc=p[i];
      for(int c=1;pc<=x[j];c++,pc*=p[i]){
        int d=pos(x[j]/pc);
        f[j]+=Poly(c)*(f[d]+A*max(0,u[d]-max(l[d],i)-1));
      }
      if(!l[j])l[j]=i;
    }
  }
  for(int i=1;i<=xn;i++)f[i]+=A*max(0,u[i]-l[i]-1);
  for(int i=xn;i;i--)f[i]-=f[i-1];
}
ll ask(){
  for(lim=xn=tot=0;sqr(lim+1)<=n;lim++);
  for(ll i=1,j;i<=n;i=j)x[++xn]=(j=n/(n/i))++;
  while(p[tot+1]<=lim)tot++;
  for(int i=1,j=1;i<=xn;u[i++]=j)while(j<=tot&&p[j]<=x[i])j++;
  calg(),calf();
  ll ret=0;
```

```
  for(int i=1;i<=xn;i++)ret+=f[i]*g[xn-i+1];
  return ret;
}
int main(){
  init(N);
  int T;
  for(scanf("%d",&T);T--;printf("%lld\n",ask()))scanf("%lld",&n);
}
```

7.52 第 k 小的期望

$f_n(k)$ 表示有 n 个变量，和为 1，第 k 小的期望。

$$\begin{aligned} f_n(k) &= \frac{1}{n^2} + (1-\frac{1}{n})f_{n-1}(k-1) \\ f_n(1) &= \frac{1}{n^2} \end{aligned}$$

7.53 固定 k 个点为根的带标号有根树森林计数

固定 k 个点作为根的 n 个点的带标号有根树森林的方案数是 $k \times n^{n-k-1}$。

7.54 斯特林近似公式

$n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$

7.55 伯努利数

$$\sum_{i=1}^{n} i^m = \frac{1}{m+1}\sum_{k=0}^{m} C(m+1,k)B_k n^{m+1-k}$$

注意当 $m=0$ 时不成立。

```
for(inv[0]=i=1;i<=T+5;i++)inv[i]=pow(i,P-2);
for(C[0][0]=i=1;i<=T+5;i++)for(C[i][0]=j=1;j<=i;j++)C[i][j]=(C[i-1][j-1]+C[i-1][j])%P;
for(i=0;i<=T;i++){
  B[i]=i+1;
  for(j=0;j<i;j++)B[i]=(B[i]-1LL*C[i+1][j]*B[j])%P;
  B[i]=(1LL*inv[i+1]*B[i]%P+P)%P;
}
```

7.56 类欧几里得算法

$$
\begin{aligned}
&cal(a,b,c,n):\\
&f = \sum_{i=0}^{n}\lfloor\frac{ai+b}{c}\rfloor\\
&g = \sum_{i=0}^{n} i\lfloor\frac{ai+b}{c}\rfloor\\
&h = \sum_{i=0}^{n}\lfloor\frac{ai+b}{c}\rfloor^2
\end{aligned}
$$

```
#include<cstdio>
typedef long long ll;
const int P=1000000007;
ll inv2,inv6,a,b,c,l,r;
struct E{
  ll f,g,h;
  E(){}
  E(ll _f,ll _g,ll _h){f=_f,g=_g,h=_h;}
};
ll pow(ll a,ll b){ll t=1;for(;b;b>>=1,a=1LL*a*a%P)if(b&1)t=1LL*t*a%P;return t;}
E cal(ll a,ll b,ll c,ll n){
  if (!a)return E(0,0,0);//请注意本行有误，请自行修改 0,0,0
  E x,y;
  if(a>=c||b>=c){
    x=cal(a%c,b%c,c,n);
    y.f=(a/c*n%P*(n+1)%P*inv2+b/c*(n+1)+x.f)%P;
    y.g=(a/c*n%P*(n+1)%P*(n*2+1)%P*inv6+b/c*(n+1)%P*n%P*inv2+x.g)%P;
    y.h=a/c*(a/c)%P*n%P*(n+1)%P*(n*2+1)%P*inv6%P;
    (y.h+=b/c*(b/c)%P*(n+1))%=P;
    (y.h+=a/c*(b/c)%P*n%P*(n+1))%=P;
    (y.h+=2LL*(a/c)%P*x.g)%=P;
    (y.h+=2LL*(b/c)%P*x.f)%=P;
    (y.h+=x.h)%=P;
    y.f=(y.f+P)%P;
    y.g=(y.g+P)%P;
    y.h=(y.h+P)%P;
    return y;
  }
  ll m=(a*n+b)/c;
  x=cal(c,c-b-1,a,m-1);
  y.f=(n*m-x.f)%P;
  y.g=((n+1)*n%P*m-x.f-x.h)%P;
  y.g=y.g*inv2%P;
  y.h=(n*m%P*(m+1)-2LL*x.g-2LL*x.f-y.f)%P;
  y.f=(y.f+P)%P;
  y.g=(y.g+P)%P;
  y.h=(y.h+P)%P;
  return y;
}
int main(){
  inv2=pow(2,P-2);
  inv6=pow(6,P-2);
```

```
  scanf("%lld%lld%lld%lld%lld",&a,&c,&b,&l,&r);
  printf("%lld",(cal(a,b,c,r).g-cal(a,b,c,l-1).g+P)%P);
}
```

7.57 置换开根

给一个 n 的置换，求它开 m 次方的结果。

```
const int N=1000010;
int n,m,i,j,k,o,x,l,d,a[N],g[N],nxt[N],t,q[N],b[N],ans[N];bool v[N];
int cal(int x){
  int i,k=m,t=1;
  for(i=2;i*i<=x;i++)if(x%i==0){
    while(x%i==0)x/=i,t*=i;
    while(k%i==0)k/=i,t*=i;
  }
  if(x>1)for(t*=x;k%x==0;k/=x,t*=x);
  return t;
}
int main(){
  read(n),read(m);
  for(i=1;i<=n;i++)read(a[i]);
  for(i=1;i<=n;i++)if(!v[i]){
    t=v[i]=1;
    for(j=a[i];j!=i;j=a[j])v[j]=1,t++;
    nxt[i]=g[t],g[t]=i;
  }
  for(i=1;i<=n;i++)if(g[i]){
    for(t=0,j=g[i];j;j=nxt[j])q[++t]=j;
    d=gcd(l=cal(i),m);
    if(t%d)return puts("-1"),0;
    for(x=1;x<=t;x+=d){
      for(j=0;j<d;j++)for(k=0,o=q[x+j];k<i;k++,o=a[o])b[(j+1LL*k*m)%l]=o;
      for(j=0;j<l;j++)ans[b[j]]=b[(j+1)%l];
    }
  }
  for(i=1;i<=n;i++)printf("%d\n",ans[i]);
}
```

7.58 反欧拉函数

给定 $n(n \leq 10^{10})$ 求所有满足 $\varphi(x) = n$ 的 x。

```
typedef long long ll;
const int N=1000000,M=5000;
int T,lim,m,d,cnt,i,j,p[N/10],tot,small[N],big[N],g[M][700];
bool v[N];ll n,a[M],b[M],q[N];
bool check(ll n){
  if(n<N)return !v[n];
  for(int i=2;1LL*i*i<=n;i++)if(n%i==0)return 0;
  return 1;
}
void dfs(int x,ll S,ll p){
  if(p==1){q[cnt++]=S;return;}
```

```
  x=g[p<=lim?small[p]:big[n/p]][x];
  if(x==m)return;
  dfs(x+1,S,p);
  for(dfs(x+1,S*=a[x],p/=a[x]-1);p%a[x]==0;dfs(x+1,S*=a[x],p/=a[x]));
}
int main(){
  for(i=2;i<N;i++){
    if(!v[i])p[tot++]=i;
    for(j=0;j<tot&&i*p[j]<N;j++){
      v[i*p[j]]=1;
      if(i%p[j]==0)break;
    }
  }
  scanf("%d",&T);
  while(T--){
    scanf("%lld",&n);
    if(n==1){puts("2\n1 2");continue;}
    for(lim=1;1LL*(lim+1)*(lim+1)<=n;lim++);
    for(cnt=m=d=0,i=1;i<=lim;i++)if(n%i==0){
      if(check(i+1))a[m++]=i+1;
      b[d++]=i;
      if(1LL*i*i!=n){
        if(check(n/i+1))a[m++]=n/i+1;
        b[d++]=n/i;
      }
    }
    std::sort(a,a+m),std::sort(b,b+d);
    for(i=0;i<d;i++){
      if(b[i]<=lim)small[b[i]]=i;else big[n/b[i]]=i;
      for(g[i][m]=m,j=m-1;~j;j--)g[i][j]=b[i]%(a[j]-1)?g[i][j+1]:j;
    }
    dfs(0,1,n);
    std::sort(q,q+cnt);
    printf("%d\n",cnt);
    if(cnt)for(printf("%lld",q[0]),i=1;i<cnt;i++)printf(" %lld",q[i]);
    puts("");
  }
}
```

7.59 毕达哥拉斯三元组

枚举所有满足 $x^2+y^2=z^2, 1 \le x < y < z \le n$，且 $\gcd(x,y,z)=1$ 的三元组。

```
int n,lim,i,j,x,y,z,v[100000];
void divide(int n,int p){
  int i,j;
  for(i=2;i*i<=n;i++)if(n%i==0){
    while(n%i==0)n/=i;
    for(j=i;j<=lim;j+=i)v[j]+=p;
  }
  if(n>1)for(j=n;j<=lim;j+=n)v[j]+=p;
}
int main(){
  scanf("%d",&n);
  for(lim=1;lim*lim<=n;lim++);
```

```
  for(i=2;i<=lim;i++){
    divide(i,1);
    for(j=i&1?2:1;j<i;j+=2){
      if(i*i+j*j>n)break;
      if(v[j])continue;
      x=i*i-j*j;
      y=2*i*j;
      z=i*i+j*j;
      if(x>y)swap(x,y);
      //x<y<z,gcd(x,y,z)=1
      printf("%d %d %d\n",x,y,z);
    }
    divide(i,-1);
  }
}
```

7.60 Stern-Brocot Tree

所有既约分数形成的排序二叉树。

```
P ask(){
  int l0=0,l1=1,r0=1,r1=0,m0,m1;
  while(1){
    m0=l0+r0,m1=l1+r1;
    if(m1>100000)break;
    if(equal(m0/m1))return P(m0,m1);
    if(should_bigger())l0=m0,l1=m1;else r0=m0,r1=m1;
  }
  compare(l0/l1,r0/r1);
}
```

7.61 Berlekamp-Massey

Berlekamp-Massey 求解线性递推。

```
#include<cstdio>
#include<vector>
using namespace std;
#define rep(i,a,n) for(int i=a;i<n;i++)
#define pb push_back
#define SZ(x) ((int)(x).size())
typedef vector<int>VI;
typedef long long ll;
const ll P=1000000007;
ll powmod(ll a,ll b){ll t=1;a%=P;for(;b;b>>=1,a=a*a%P)if(b&1)t=t*a%P;return t;}
int _,n;
namespace linear_seq{
const int N=10010;
ll res[N],base[N],_c[N],_md[N];
vector<int>Md;
void mul(ll*a,ll*b,int k){
  rep(i,0,k+k)_c[i]=0;
  rep(i,0,k)if(a[i])rep(j,0,k)_c[i+j]=(_c[i+j]+a[i]*b[j])%P;
  for(int i=k+k-1;i>=k;i--)if(_c[i])
```

```
    rep(j,0,SZ(Md))_c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%P;
  rep(i,0,k)a[i]=_c[i];
}
int solve(ll n,VI a,VI b){//a系数 b初值 b[n+1]=a[0]*b[n]+...
  printf("%d\n",SZ(b));
  rep(i,0,SZ(b))printf("b[%d]=%d\n",i,b[i]);
  printf("%d\n",SZ(a));
  printf("b[n]");
  rep(i,0,SZ(a)){
    if(!i)putchar('=');else putchar('+');
    printf("%d*b[n-%d]",a[i],i+1);
  }
  puts("");
  ll ans=0,pnt=0;
  int k=SZ(a);
  rep(i,0,k)_md[k-1-i]=-a[i];_md[k]=1;
  Md.clear();
  rep(i,0,k)if(_md[i])Md.push_back(i);
  rep(i,0,k)res[i]=base[i]=0;
  res[0]=1;
  while((1LL<<pnt)<=n)pnt++;
  for(int p=pnt;p>=0;p--){
    mul(res,res,k);
    if((n>>p)&1){
      for(int i=k-1;i>=0;i--)res[i+1]=res[i];res[0]=0;
      rep(j,0,SZ(Md))res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%P;
    }
  }
  rep(i,0,k)ans=(ans+res[i]*b[i])%P;
  if(ans<0)ans+=P;
  return ans;
}
VI BM(VI s){
  VI C(1,1),B(1,1);
  int L=0,m=1,b=1;
  rep(n,0,SZ(s)){
    ll d=0;
    rep(i,0,L+1)d=(d+(ll)C[i]*s[n-i])%P;
    if(!d)++m;
    else if(2*L<=n){
      VI T=C;
      ll c=P-d*powmod(b,P-2)%P;
      while(SZ(C)<SZ(B)+m)C.pb(0);
      rep(i,0,SZ(B))C[i+m]=(C[i+m]+c*B[i])%P;
      L=n+1-L;B=T;b=d;m=1;
    }else{
      ll c=P-d*powmod(b,P-2)%P;
      while(SZ(C)<SZ(B)+m)C.pb(0);
      rep(i,0,SZ(B))C[i+m]=(C[i+m]+c*B[i])%P;
      ++m;
    }
  }
  return C;
}
int gao(VI a,ll n){
  VI c=BM(a);
  c.erase(c.begin());
```

```
  rep(i,0,SZ(c))c[i]=(P-c[i])%P;
  return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
}
}
int main(){
  for(scanf("%d",&_);_;_--){
    scanf("%d",&n);
    printf("%d\n",linear_seq::gao(VI{2,24,96,416,1536,5504,18944,64000,
212992,702464},n-1));
  }
}
```

7.62 ax+by+c=0 的整数解个数

求满足 $ax+by+c=0$，且 $xl \le x \le xr, yl \le y \le yr$ 的整数解个数。

```
#include<iostream>
#include<algorithm>
using namespace std;
typedef long long ll;
ll a,b,c,xl,xr,yl,yr;
ll exgcd(ll a,ll b,ll&x,ll&y){
  if(!b)return x=1,y=0,a;
  ll d=exgcd(b,a%b,x,y),t=x;
  return x=y,y=t-a/b*y,d;
}
ll solve(ll a,ll b,ll c,ll xl,ll xr,ll yl,ll yr){
  if(xl>xr)return 0;
  if(yl>yr)return 0;
  if(!a&&!b){
    if(c)return 0;
    return (xr-xl+1)*(yr-yl+1);
  }
  if(!b){
    swap(a,b);
    swap(xl,yl);
    swap(xr,yr);
  }
  if(!a){
    if(c%b)return 0;
    ll y=-c/b;
    if(y<yl||y>yr)return 0;
    return xr-xl+1;
  }
  ll x,y,d=exgcd((a%abs(b)+abs(b))%abs(b),abs(b),x,y);
  if(c%d)return 0;
  x=(x%abs(b)+abs(b))%abs(b)*((((-c)%abs(b))+abs(b))%abs(b)/d)%abs(b/d);
  d=abs(b/d);
  ll kl=(xl-x)/d-3,kr=(xr-x)/d+3;
  while(x+kl*d<xl)kl++;
  while(x+kr*d>xr)kr--;
  ll A=(-yl*b-a*x-c)/(a*d),B=(-yr*b-a*x-c)/(a*d);
  if(A>B)swap(A,B);
  kl=max(kl,A-3);
  kr=min(kr,B+3);
```

```
  while(kl<=kr){
    ll y=(-c-a*x-a*d*kl)/b;
    if(yl<=y&&y<=yr)break;
    kl++;
  }
  while(kl<=kr){
    ll y=(-c-a*x-a*d*kr)/b;
    if(yl<=y&&y<=yr)break;
    kr--;
  }
  if(kl>kr)return 0;
  return kr-kl+1;
}
int main(){
  cin>>a>>b>>c>>xl>>xr>>yl>>yr;
  cout<<solve(a,b,c,xl,xr,yl,yr);
}
```

7.63 切比雪夫多项式

$n!! = n(n-2)(n-4)\ldots$。

若 n 为偶数，则：

$$\cos(nx) = \sum_{k=0}^{\frac{n}{2}} (-1)^{\frac{n-2k}{2}} \frac{n \cdot (n+2k-2)!!}{(2k)!(n-2k)!!} \cos^{2k} x$$

若 n 为奇数，则：

$$\cos(nx) = \sum_{k=1}^{\frac{n+1}{2}} (-1)^{\frac{n+1-2k}{2}} \frac{n \cdot (n+2k-3)!!}{(2k-1)!(n+1-2k)!!} \cos^{2k-1} x$$

另外 3 个公式：

$$\begin{aligned}
\sin(n\theta) &= \sum_{r=0,2r+1\le n} (-1)^r C(n,2r+1) \cos^{n-2r-1}(\theta) \sin^{2r+1}(\theta) \\
\cos(n\theta) &= \sum_{r=0,2r\le n} (-1)^r C(n,2r) \cos^{n-2r}(\theta) \sin^{2r}(\theta) \\
\tan(n\theta) &= \frac{\sum_{r=0,2r+1\le n} (-1)^r C(n,2r+1) \tan^{2r+1}(\theta)}{\sum_{r=0,2r\le n} (-1)^r C(n,2r) \tan^{2r}(\theta)}
\end{aligned}$$

7.64 斜率绝对值为 1 的折线计数

从 (a,b) 走到 (c,d)，每次从 (x,y) 只能往 $(x+1,y-1)$ 或 $(x+1,y+1)$ 走，且 $y \ge 0$，问方案数。

旋转坐标系后用组合数求方案数，对于不合法的情况等价于将起点沿 $y=-1$ 翻转后的情况。

```
int cal(int n,int a,int b){return C(n,a)-C(n,b);}
int solve(int a,int b,int c,int d){
  if(a>c)return 0;
```

```
  if(a==c)return b==d;
  c-=a;
  int k=c+d-b;
  if(k%2)return 0;
  k/=2;
  if(b+k>mx)mx=b+k;
  d+=b+2;
  return cal(c,k,(c+d)/2);
}
```

7.65 等比矩阵求和

```
void sum(int n,int g[][N],int a[][N]){//a=sum_{i=0}^{n-1}g^i
  static int s[N][N],f[N][N];
  clear(a);
  s=g;
  for(n--;n;n>>=1,s=g*s+s,g*=g)if(n&1)a=g*a+s;
  a+=I;
}
```

7.66 等价类容斥

考虑容斥，$Bell(p)$ 枚举所有等价情况。对于一种情况，强制了一个等价类里面的数都要相同，其它的可以相同也可以不同。

容斥系数为：(-1) 的 (p− 等价类个数) 次方 $\times$(每个等价类大小 -1)! 之积。

贝尔数前若干项的表：1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597, 27644437, 190899322, 1382958545。

7.67 掷硬币

7.67.1 二人游戏

两个人玩游戏，分别有个 01 模式串 S 和 T，每次随机 50% 概率生成 0 和 1，直到包含 S 或 T 作为子串时，被包含的人胜利。

若 $A=B$，那么平局。

若 A 是 B 的子串，那么 A 赢。

若 B 是 A 的子串，那么 B 赢。

否则比较 $B:B-B:A$ 与 $A:A-A:B$ 的大小即可，大的胜利。

$A:B=\sum_{k=1}^{\min(|A|,|B|)}2^k[A$ 长度为 k 的后缀 $=B$ 长度为 k 的前缀 $]$。

A 的胜率 $/B$ 的胜率 $=(B:B-B:A)/(A:A-A:B)$。

```
const int N=100010;
int n,m;
inline void getnxt(char*a,int*f,int n){
  int i,j;
  for(f[1]=j=0,i=2;i<=n;f[i++]=j){
    while(j&&a[j+1]!=a[i])j=f[j];
    if(a[j+1]==a[i])j++;
```

```
  }
}
//is a substring of b?
inline bool occ(char*a,int n,char*b,int m){
  static int f[N];
  getnxt(a,f,n);
  for(int i=1,j=0;i<=m;i++){
    while(j&&a[j+1]!=b[i])j=f[j];
    if(a[j+1]==b[i]){
      j++;
      if(j==n)return 1;
    }
  }
  return 0;
}
//s+=p*(a:b)
inline void cal(char*a,int n,char*b,int m,int*s,int p){
  int i;
  static char c[N<<1];
  static int f[N<<1];
  for(i=1;i<=m;i++)c[i]=b[i];
  for(i=1;i<=n;i++)c[i+m]=a[i];
  getnxt(c,f,n+m);
  for(i=f[n+m];i;i=f[i])if(i<=n&&i<=m)s[i]+=p;
}
inline void fix(int*a,int n){
  for(int i=1;i<=n;i++)if(a[i]<0)a[i]+=2,a[i+1]--;
}
inline int solve(){
  //same
  int i,len=n>m?n:m;
  static char a[N],b[N];
  static int u[N],d[N];
  scanf("%s%s",a+1,b+1);
  if(n==m){
    for(i=1;i<=n;i++)if(a[i]!=b[i])break;
    if(i>n)return 0;
  }
  //a is substring of b
  if(occ(a,n,b,m))return -1;
  //b is substring of a
  if(occ(b,m,a,n))return 1;
  for(i=1;i<=len;i++)u[i]=d[i]=0;
  cal(b,m,b,m,u,1),cal(b,m,a,n,u,-1);
  cal(a,n,a,n,d,1),cal(a,n,b,m,d,-1);
  fix(u,len),fix(d,len);
  for(i=len;i;i--)if(u[i]!=d[i])return u[i]>d[i]?-1:1;
  return 0;
}
int main(){
  while(~scanf("%d%d",&n,&m)){
    if(!n)return 0;
    int t=solve();
    if(t<0)puts("Hamlet");
    if(!t)puts("Equal");
    if(t>0)puts("Laertes");
  }
```

```
}
```

7.67.2 多人游戏

多个人有两两不同的等长 01 模式串，问每个人的胜率。

设 N 表示还没有人获胜的状态。

对于 A 来说，枚举每个 B 的每个后缀。

有 $P(NA) = P(A) + \sum_B \sum_S P(BT)$。

其中 S 是 B 的后缀，且是 A 的前缀，T 是 A 中除去 S 的后缀，列出 $n+1$ 个变量 $n+1$ 个方程进行高斯消元即可。

对于预处理，可以拼接 AB 后，沿着 KMP 的 next 链进行查找。

时间复杂度 $O(n^3)$。

```
const int N=305;
int n,m,i,j,k,f[N<<1];double po[N],g[N][N],ans[N],t;char a[N][N],b[N<<1];
double kmp(int x,int y){
  int i,j;
  for(i=1;i<=m;i++)b[i]=a[x][i],b[i+m]=a[y][i];
  for(j=0,i=2;i<=m+m;f[i++]=j){
    while(j&&b[j+1]!=b[i])j=f[j];
    if(b[j+1]==b[i])j++;
  }
  double t=0;
  for(i=f[m+m];i;i=f[i])if(i<=m)t+=po[m-i];
  return t;
}
int main(){
  scanf("%d%d",&n,&m);
  for(i=1;i<=n;i++)scanf("%s",a[i]+1);
  for(po[0]=1,i=1;i<=m;i++)po[i]=po[i-1]/2.0;
  for(i=1;i<=n+1;i++)g[0][i]=1;
  for(i=1;i<=n;i++){
    g[i][0]=-1;
    for(j=1;j<=n;j++)g[i][j]=kmp(i,j);
  }
  for(i=0;i<=n;i++){
    for(k=i,j=i+1;j<=n;j++)if(fabs(g[j][i])>fabs(g[k][i]))k=j;
    if(k!=i)for(j=i;j<=n+1;j++)swap(g[i][j],g[k][j]);
    for(j=i+1;j<=n;j++)for(t=g[j][i]/g[i][i],k=i;k<=n+1;k++)g[j][k]-=g[i][k]*t;
  }
  for(ans[n]=g[n][n+1]/g[n][n],i=n-1;i;i--){
    for(ans[i]=g[i][n+1],j=n;j>i;j--)ans[i]-=ans[j]*g[i][j];
    ans[i]/=g[i][i];
  }
  for(i=1;i<=n;i++)printf("%.10f\n",ans[i]);
}
```

7.67.3 作为子串出现的概率

将 m 个长度不超过 n 的字符串按在长度为 n 的随机串中作为子串出现的概率从大到小排序。

对于每个串，从大到小记录长度至少为 $2l-n$ 的非 l 的 border 长度 vector，那么 vector 越小的出现概率越高。

```
const int N=100010,M=12;
int n,m,i,q[M];
struct E{
  int m,a[N];char s[N];
  void init(){
    static int nxt[N];
    scanf("%s",s+1);
    int l=strlen(s+1),i,j;
    for(nxt[1]=j=0,i=2;i<=l;nxt[i++]=j){
      while(j&&s[j+1]!=s[i])j=nxt[j];
      if(s[j+1]==s[i])j++;
    }
    m=0;
    for(i=nxt[l];i>0&&i>=l+l-n;i=nxt[i])a[++m]=i;
  }
}a[M];
bool cmp(int x,int y){
  for(int i=1;i<=a[x].m&&i<=a[y].m;i++)
    if(a[x].a[i]!=a[y].a[i])return a[x].a[i]<a[y].a[i];
  if(a[x].m!=a[y].m)return a[x].m<a[y].m;
  return x<y;
}
int main(){
  scanf("%d%d",&n,&m);
  for(i=1;i<=m;i++)a[i].init(),q[i]=i;
  sort(q+1,q+m+1,cmp);
  for(i=1;i<=m;i++)puts(a[q[i]].s+1);
}
```

7.67.4 作为子串出现的期望长度

给一个字符串 S 和字符集大小 n，问随机生成字符串 T 到多少位的时候包含 S 作为子串，输出期望长度。

对于任意的 $i(1 \le i \le |S|)$，如果 S 长度为 i 的前后缀相等，则答案加上 n^i。

7.68 泰勒展开

$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + ... + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$

7.69 约数和

$\sum_{i=1}^{A}\sum_{j=1}^{B}\sum_{k=1}^{C} d(ijk) \bmod 10^9+7$，$A, B, C \le 100000$。

```
const int N=100010,M=45,P=1000000007;
int T,A,B,C,n,i,j,k,p[N],tot,v[N],d[N],f[N],g[M+5][M+5][M+5];
inline void up(int&a,int b){a=a+b<P?a+b:a+b-P;}
inline void del(int&a,int b){a=a>=b?a-b:a-b+P;}
int dfs(int x,int A,int B,int C){
  if(!p[x])return 1LL*f[A]*f[B]%P*f[C]%P;
  int o=p[x],n=max(A,max(B,C));
```

```
  if(o>n&&n<=M)return g[A][B][C];
  if(!A&&!B&&!C)return 0;
  int ret=dfs(x+1,A,B,C);
  if(A>=o&&B>=o)del(ret,dfs(x+1,A/o,B/o,C));
  if(B>=o&&C>=o)del(ret,dfs(x+1,A,B/o,C/o));
  if(A>=o&&C>=o)del(ret,dfs(x+1,A/o,B,C/o));
  if(A>=o&&B>=o&&C>=o){
    int t=dfs(x+1,A/o,B/o,C/o);
    up(t,t);
    up(ret,t);
  }
  return ret;
}
int main(){
  for(i=1;i<N;i++)d[i]=1;
  for(i=2;i<N;i++)if(!v[i])for(p[tot++]=i,j=1;1LL*i*j<N;j++)v[i*j]=1,d[i*j]+=d[j];
  for(i=1;i<N;i++)f[i]=f[i-1]+d[i];
  reverse(p,p+tot);
  for(i=1;i<=M;i++)for(j=1;j<=M;j++)for(k=1;k<=M;k++)g[i][j][k]=d[i*j*k];
  for(i=1;i<=M;i++)for(j=1;j<=M;j++)for(k=1;k<=M;k++)up(g[i][j][k],g[i-1][j][k]);
  for(i=1;i<=M;i++)for(j=1;j<=M;j++)for(k=1;k<=M;k++)up(g[i][j][k],g[i][j-1][k]);
  for(i=1;i<=M;i++)for(j=1;j<=M;j++)for(k=1;k<=M;k++)up(g[i][j][k],g[i][j][k-1]);
  scanf(”%d”,&T);
  while(T--){
    scanf(”%d%d%d”,&A,&B,&C);
    for(n=0;p[n]>A&&p[n]>B&&p[n]>C;n++);
    printf(”%d\n”,dfs(n,A,B,C));
  }
}
```

7.70 单位根反演

如果 p 为质数且 $k|(p-1)$，找到 p 的原根 g，单位根 $w=g^{\frac{p-1}{k}}$，则 $\sum_{i=0}^{k-1}w^{ix}$ 当 x 是 k 的倍数时为 k，当 x 不是 k 的倍数时为 0。

$$\begin{aligned}
[k|x]f_x &= \frac{1}{k}\sum_{i=0}^{k-1}w^{ix}f_x \\
\sum_{i=0}^{n}[k|i]C(n,i)f_i &= \frac{1}{k}\sum_{x=0}^{k-1}\sum_{i=0}^{n}C(n,i)f_iw^{xi} \\
F(x) &= x^{-n}(xI+A)^n=\sum_{i=0}^{n}C(n,i)x^{-i}A^i \\
ans &= \frac{1}{k}\sum_{i=0}^{n}C(n,i)A^i(\sum_{j=0}^{k-1}w^{ij}) \\
&= \frac{1}{k}[F(w^{-0})+F(w^{-1})+\cdots+F(w^{-(k-1)})]
\end{aligned}$$

7.71 求 $1\times3\times5\times7\ldots \bmod 2^{64}$

```
const ull A = 7083242339890757633ULL;
const ull B = 5560996899424501760ULL;
```

```
const ull S = 1 << 16;
ull solve(ull x) {
  ull p = (x + 1) / S / 2, sumQ;
  if(p % 3 == 0) sumQ = p / 3 * (4 * p * p - 1);
  else {
    if((p * 2 - 1) % 3 == 0) {
      sumQ = (p * 2 - 1) / 3 * (p * 2 + 1) * p;
    } else {
      sumQ = (p * 2 + 1) / 3 * (p * 2 - 1) * p;
    }
  }
  ull ret = quickPow(A, p) + B * quickPow(A, p - 1) * sumQ * S * S;
  for(ull i = 2ull * S * p + 1; i <= x; i += 2) ret = ret * i;
  return ret;
}
```

7.72 组合数模 p^k

```
#include<cstdio>
#include<map>
using namespace std;
typedef long long ll;
const int P=5,K=13,MO=1220703125;
int i,j;
ll n,m,p[K],C[K][K];
struct Poly{
  ll a[K];
  Poly(){for(int i=0;i<K;i++)a[i]=0;}
  Poly operator*(const Poly&b)const{
    Poly c;
    for(int i=0;i<K;i++)for(int j=0;j<=i;j++)c.a[i]=(1LL*a[j]*b.a[i-j]+c.a[i])%MO;
    return c;
  }
  Poly ext(ll x){
    Poly b;
    x%=MO;
    for(int i=p[0]=1;i<K;i++)p[i]=1LL*p[i-1]*x%MO;
    for(int i=0;i<K;i++)for(int j=i;j<K;j++)
b.a[i]=(1LL*a[j]*p[j-i]%MO*C[j][i]+b.a[i])%MO;
    return b;
  }
};
map<ll,Poly>h;
inline Poly G(ll n){//G(n).a[0]=prod_{i=1..n,i%P>0}i
  if(h.find(n)!=h.end())return h[n];
  Poly r,t;
  r.a[0]=1;
  if(n<100){
    for(int i=1;i<=n;i++)if(i%P)t.a[0]=i,t.a[1]=1,r=r*t;
  }else{
    ll o=n/2/P*P;
    Poly u=G(o-1),v=u.ext(o);
    r=u*v;
    for(ll i=o<<1;i<=n;i++)if(i%P)t.a[0]=i,t.a[1]=1,r=r*t;
  }
```

```
  return h[n]=r;
}
inline ll F(ll n){
  ll r=1;
  for(ll i=1;i<=n;i*=P)r=1LL*r*G(n/i).a[0]%MO;
  return r;
}
inline ll po(ll a,ll b){ll t=1;for(;b;b>>=1,a=a*a%MO)if(b&1)t=t*a%MO;return t;}
inline ll getC(ll x,ll y){
  if(x<y)return 0;
  ll c=0;
  for(ll i=P;i<=x;i*=P)c+=x/i-y/i-(x-y)/i;
  if(c>=K)return 0;
  ll r=1;
  for(int i=1;i<=c;i++)r=1LL*r*P%MO;
  ll mo=MO/P*(P-1);
  return 1LL*r*F(x)%MO*po(1LL*F(y)*F(x-y)%MO,mo-1)%MO;
}
int main(){
  for(C[0][0]=i=1;i<K;i++)for(C[i][0]=j=1;j<=i;j++)C[i][j]=(C[i-1][j]+C[i-1][j-1])%MO;
  while(~scanf("%lld%lld",&n,&m))printf("%lld\n",getC(n,m));
}
```

7.73 无根树计数

每个点度数不超过 $m(m \leq 4)$。

以重心为根 DP 求出无标号有根树的个数，则最大子树大小要 $< \frac{n+1}{2}$。

设 $f[i][j]$ 表示 i 个点的子树，根有 j 个儿子的方案数。用背包进行转移，转移时枚举该子树放了多少次，用隔板法求出方案数。

设 $g[i]$ 表示根有 $< m$ 个儿子的方案数，$h[i]$ 表示根有 $\leq m$ 个儿子的方案数。则 $ans = h[n]$，特别地当 n 是偶数时，还可能有两个重心，此时两边方案数都为 $g[\frac{n}{2}]$。

```
void solve(){
  f[1][0].set(1);
  one.set(1);
  mx=(n+1)/2-1;
  for(i=1;i<=mx;i++){
    t.set(0);
    for(j=0;j<m;j++)t+=f[i][j];
    if(t.iszero())continue;
    w[1].copy(t);
    w[2].set(0);
    w[3].set(0);
    w[4].set(0);
    for(j=2;j<=m;j++){
      t+=one;
      w[j]=w[j-1]*t;
    }
    w[2]/=2;
    w[3]/=6;
    w[4]/=24;
    for(j=n;j>i;j--)for(k=1;k<=m;k++)for(o=1;o<=k&&i*o<j;o++)f[j][k]+=f[j-i*o][k-o]*w[o];
  }
```

```
  for(i=1;i<=n;i++){
    for(j=0;j<m;j++)g[i]+=f[i][j];
    h[i].copy(g[i]);
    h[i]+=f[i][m];
  }
  ans.copy(h[n]);
  if(n%2==0){
    n>>=1;
    if(!g[n].iszero()){
      t.copy(g[n]);
      g[n]+=one;
      t=t*g[n];
      t/=2;
      ans+=t;
    }
  }
}
```

7.74 快速分治 NTT

```
#include<cstdio>
#include<cstring>
#include<algorithm>
typedef long long ll;
const int N=1048576*2,K=20,P=998244353,G=3;


typedef unsigned int uint32;
typedef long long int64;
typedef unsigned long long uint64;
typedef uint32 word;
typedef uint64 dword;
typedef int sword;

const int word_bits=sizeof(word)*8;
word mod,Modinv,r2;

// Montgomery modular multiplication - about 7x faster
// ensure mod is an odd number, use after call set_mod method
struct UnsafeMod{
  word x;
  UnsafeMod(): x(0) {}
  UnsafeMod(word _x): x(init(_x)) {}

  UnsafeMod& operator += (const UnsafeMod& rhs) {
    (x += rhs.x) >= mod && (x -= mod);
    return *this;
  }
  UnsafeMod& operator -= (const UnsafeMod& rhs) {
    sword(x -= rhs.x) < 0 && (x += mod);
    return *this;
  }
  UnsafeMod& operator *= (const UnsafeMod& rhs) {
    x = reduce(dword(x) * rhs.x);
    return *this;
```

```
  }
  UnsafeMod operator + (const UnsafeMod &rhs) const {
    return UnsafeMod(*this) += rhs;
  }
  UnsafeMod operator - (const UnsafeMod &rhs) const {
    return UnsafeMod(*this) -= rhs;
  }
  UnsafeMod operator * (const UnsafeMod &rhs) const {
    return UnsafeMod(*this) *= rhs;
  }
  UnsafeMod pow(uint64 e) const {
    UnsafeMod ret(1);
    for (UnsafeMod base = *this; e; e >>= 1, base *= base) {
      if (e & 1) ret *= base;
    }
    return ret;
  }
  word get() const {
    return reduce(x);
  }
  static word modulus() {
    return mod;
  }
  static word init(word w) {
    return reduce(dword(w) * r2);
  }
  static void set_mod(word m) {
    mod = m;
    Modinv = mul_inv(mod);
    r2 = -dword(mod) % mod;
  }
  static word reduce(dword x) {
    word y = word(x >> word_bits) - word((dword(word(x) * Modinv) * mod) >> word_bits);
    return sword(y) < 0 ? y + mod : y;
  }
  static word mul_inv(word n, int e = 6, word x = 1) {
    return !e ? x : mul_inv(n, e - 1, x * (2 - x * n));
  }
}TWO;

int n,m,i,j,k;
UnsafeMod a[N+10],b[N+10],tmp[N+10];
UnsafeMod g[K+1],ng[K+10],gw[N+10],ngw[N+10];
bool v[K+10];UnsafeMod vb[N+10];
int pos[N+10],inv[N+10],inv2;
inline void NTT(UnsafeMod*a,int n,int t){
  int j=__builtin_ctz(n)-1;
  for(int i=0;i<n;i++)pos[i]=pos[i>>1]>>1|((i&1)<<j);
  for(int i=1;i<n;i++)if(i<pos[i])std::swap(a[i],a[pos[i]]);
  for(int d=0;(1<<d)<n;d++){
    int m=1<<d,m2=m<<1;
    UnsafeMod*_w=t==1?gw:ngw;
    _w+=m;
    for(int i=0;i<n;i+=m2){
      UnsafeMod*w=_w;
      for(int j=i;j<m+i;j++,w++){
        UnsafeMod t=*w*a[j+m];
```

```
        a[j+m]=a[j]-t;
        a[j]+=t;
      }
    }
  }
  if(t==-1){
    UnsafeMod j=inv[n];
    for(int i=0;i<n;i++)a[i]*=j;
  }
}
//给定 a, 求 a 的逆元 b
void getinv(UnsafeMod*a,UnsafeMod*b,int n){
  if(n==1){b[0]=a[0].pow(P-2);return;}
  getinv(a,b,n>>1);
  int k=n<<1,i;
  for(i=0;i<n;i++)tmp[i]=a[i];
  for(i=n;i<k;i++)tmp[i]=b[i]=0;
  NTT(tmp,k,1),NTT(b,k,1);
  for(i=0;i<k;i++)b[i]*=TWO-tmp[i]*b[i];
  NTT(b,k,-1);
  for(i=n;i<k;i++)b[i]=0;
}
UnsafeMod f[N+5],A[N+5],B[N+5];
void solve(int l,int r){
  if(r-l+1<32){
    for(int i=l;i<r;i++)for(int j=i+1;j<=r;j++)f[j]+=f[i]*A[j-i];
    for(int i=l;i<=r;i++)f[i]=233;
    return;
  }
  if(l==r){f[l]=233;return;}
  int mid=(l+r)>>1,i,n=r-l+1,k;
  solve(l,mid);
  for(k=1;k<n;k<<=1);
  memcpy(A,f+l,sizeof(A[0])*(mid-l+1));
  memset(A+mid-l+1,0,sizeof(A[0])*(k-mid+l-1));
  NTT(A,k,1);
  int Log=__builtin_ctz(k);
  if(!v[Log]){
    v[Log]=1;
    for(i=0;i<k;i++)B[i]=i;
    NTT(B,k,1);
    memcpy(vb+k,B,sizeof(B[0])*k);
    for(i=0;i<k;i++)A[i]*=B[i];
  }else for(i=0;i<k;i++)A[i]*=vb[i+k];
  NTT(A,k,-1);
  for(i=r;i>mid;i--)f[i]+=A[i-l];
  solve(mid+1,r);
}
int main(){
  UnsafeMod::set_mod(P);
  TWO=2;
  for(g[K]=((UnsafeMod)G).pow((P-1)/N),ng[K]=g[K].pow(P-2),i=K-1;~i;i--)
g[i]=g[i+1]*g[i+1],ng[i]=ng[i+1]*ng[i+1];
  for(i=0;i<=K;i++){
    gw[1<<i]=ngw[1<<i]=1;
    for(j=1;j<1<<i;j++){
      gw[(1<<i)+j]=gw[(1<<i)+j-1]*g[i];
```

```
      ngw[(1<<i)+j]=ngw[(1<<i)+j-1]*ng[i];
    }
  }
  for(inv[1]=1,i=2;i<=N;i++)inv[i]=1LL*(P-inv[P%i])*(P/i)%P;inv2=inv[2];

  /*k=1<<20;
  for(i=0;i<k;i++)a[i]=i+233;
  a[0]=1;
  getinv(a,b,k);
  for(i=0;i<k&&i<10;i++)printf("%u ",b[i].get());*/

  solve(1,1000000);
  //solve(1,100000);
  for(i=0;i<k&&i<10;i++)printf("%u ",f[i].get());
  return 0;
}
```

7.75 万能欧几里得

求 $\sum_{x=1}^{L} A^x B^{\lfloor \frac{Px+R}{Q} \rfloor}$，其中 A, B 是 $N \times N$ 的矩阵。

```
#include<bits/stdc++.h>
using namespace std;
typedef unsigned long long u64;
typedef long long i64;
const int N=21,mod=998244353;
int d;
struct mat{
  int v[N][N];
  mat(){memset(v,0,sizeof(v));}
  mat(int x){memset(v,0,sizeof(v));for(int i=1;i<=d;++i)v[i][i]=x;}
  int*operator[](int x){return v[x];}
  void rev(){for(int i=1;i<=d;++i)for(int j=1;j<i;++j)swap(v[i][j],v[j][i]);}
  friend mat operator+(mat a,mat b){for(int i=1;i<=d;++i)for(int j=1;j<=d;++j)
a[i][j]=(a[i][j]+b[i][j])%mod;return a;}
  friend mat operator*(mat a,mat b){
    u64 s;mat c;int i,j,k;b.rev();
    for(i=1;i<=d;++i)for(j=1;j<=d;c[i][j++]=s%mod)
for(s=0,k=1;k<=d;s+=u64(a[i][k])*b[j][k],++k)if(!(k&15))s%=mod;
    return c;
  }
};
template<class T>
struct supergcd{
  struct node{
    T mr,ml,sum;
    node():mr(1),ml(1),sum(0){}
    node(T mr,T ml,T sum):mr(mr),ml(ml),sum(sum){}
    friend node operator+(node a,node b){
      node c;
      c.ml=a.ml*b.ml,c.mr=a.mr*b.mr,c.sum=a.sum+a.ml*b.sum*a.mr;
      return c;
    }
    friend node operator*(node a,i64 b){
      if(!b)return node();
```

```
      node s=a*(b>>1);s=s+s;
      if(b&1)s=s+a;
      return s;
    }
  };
  i64 div(i64 a,i64 b,i64 c,i64 d){
    i64 res=((long double)a*b+c)/d;
    while(a*b+c-d*(res+1)>=0)res++;
    while(a*b+c-d*res<0)res--;
    return res;
  }
  node solve(i64 P,i64 Q,i64 R,i64 L,node A,node B){
    assert(P>=0&&Q>0&&R>=0&&L>=0);
    R%=Q;
    if(!L)return node();
    if(P>=Q)return solve(P%Q,Q,R,L,A,A*(P/Q)+B);
    i64 M=div(L,P,R,Q);
    if(!M)return B*L;
    return B*((Q-R-1)/P)+A+solve(Q,P,-R-1+Q,M-1,B,A)+B*(L-div(Q,M,-R+P-1,P)+1);
  }
  T work(T A,T B,i64 P,i64 Q,i64 R,i64 L){
    node NU,NR;
    NU.mr=B,NR.ml=A,NR.sum=A;
    return (NU*(R/Q)+solve(P,Q,R,L,NU,NR)).sum;
  }
};
int main(){
  i64 P,Q,L,R;mat A,B;
  cin>>P>>Q>>R>>L>>d;
  for(int i=1;i<=d;++i)for(int j=1;j<=d;++j)cin>>A[i][j];
  for(int i=1;i<=d;++i)for(int j=1;j<=d;++j)cin>>B[i][j];
  mat C=supergcd<mat>().work(A,B,P,Q,R,L);
  for(int i=1;i<=d;++i)for(int j=1;j<=d;++j)cout<<C[i][j]<<" \n"[j==d];
}
```

7.76 Min25 类欧几里得

求 $\sum_{x=0}^{n} x^{e_1} \lfloor \frac{ax+b}{c} \rfloor^{e_2}$，其中 $e_1 + e_2 \leq 50$。

```
#include <cstdio>
#include <stack>
#define getchar getchar_unlocked
#define putchar putchar_unlocked

using namespace std;

typedef long long i64;

int get_int() {
  int c, n;
  while ((c = getchar()) < '0');
  n = c - '0';
  while ((c = getchar()) >= '0') n = n * 10 + (c - '0');
  return n;
}
```

```cpp

const int K = 50;
const int mod = 1e9 + 7;
const i64 lmod = i64(mod) << 32;
const int signs[2] = {1, mod - 1};
int invs[K + 2], binom[K + 2][K + 2], B[K + 1], polys[K + 1][K + 2];

int add(int a, int b) {
  return (a += b - mod) < 0 ? a + mod : a;
}

i64 add64(i64 a, i64 b) {
  return (a += b - lmod) < 0 ? a + lmod : a;
}

int mul(int a, int b) {
  return i64(a) * b % mod;
}

int power_sum(int e, int x) {
  int ret = 0;
  for (int i = 0; i < e + 2; ++i) ret = add(mul(ret, x), polys[e][i]);
  return mul(ret, invs[e + 1]);
}

void init() {
  invs[0] = invs[1] = 1;
  for (int i = 2; i <= K + 1; ++i) invs[i] = mul(invs[mod % i], mod - mod / i);
  for (int i = 0; i <= K + 1; ++i) {
    binom[i][0] = 1;
    for (int j = 1; j <= i; ++j) binom[i][j] = add(binom[i - 1][j - 1], binom[i - 1][j]);
  }
  B[0] = 1;
  for (int i = 1; i <= K; ++i) {
    int s = 0;
    for (int j = 0; j < i; ++j) s = add(s, mul(binom[i + 1][j], B[j]));
    B[i] = mul(mul(s, invs[i + 1]), signs[1]);
  }
  for (int i = 0; i <= K; ++i) {
    for (int j = 0; j <= i; ++j)
polys[i][j] = mul(mul(binom[i + 1][j], B[j]), signs[j & 1]);
    polys[i][i + 1] = 0;
  }
  polys[0][1] = 1;
}

struct T {
  int n, a, b, c;
  T() {}
  T(int _n, int _a, int _b,int _c) {
    n = _n;
    a = _a;
    b = _b;
    c = _c;
  }
};

```

```
int scary_sum(int N, int a, int b, int c, int e1, int e2) {
  stack<T> stac;

  while (1) {
    stac.push(T(N, a, b, c));
    if (N < 0 || a == 0) break;
    if (a >= c) {
      a %= c;
    } else if (b >= c) {
      b %= c;
    } else {
      N = (i64(a) * N + b) / c - 1;
      b = c - 1 - b; swap(a, c);
    }
  }

  const int S = e1 + e2;
  int curr[K + 1][K + 1] = {}, next[K + 1][K + 1] = {};
  while (!stac.empty()) {
    T o = stac.top(); stac.pop();
    int N = o.n;
    int a = o.a;
    int b = o.b;
    int c = o.c;
    if (N < 0) {
      ;
    } else if (a == 0) {
      int q = b / c;
      for (int e1 = 0; e1 <= S; ++e1) {
        int s = power_sum(e1, N);
        for (int e2 = 0; e2 <= S - e1; ++e2) next[e1][e2] = s, s = mul(s, q);
      }
    } else if (a >= c || b >= c) {
      int q = (a >= c) ? a / c : b / c;
      int d = (a >= c) ? 1 : 0;
      for (int e1 = 0; e1 <= S; ++e1) {
        for (int e2 = 0; e2 <= S - e1; ++e2) {
          i64 s = 0; int p = 1;
          for (int i2 = 0; i2 <= e2; ++i2) {
            s = add64(s, i64(p) * mul(binom[e2][i2], curr[e1 + i2 * d][e2 - i2]));
            p = mul(p, q);
          }
          next[e1][e2] = s % mod;
        }
      }
    } else {
      static int cumu[K + 1][K + 1];
      for (int e2 = 0; e2 <= S - 1; ++e2) {
        for (int e1 = 0; e1 <= S - e2 - 1; ++e1) {
          i64 s = 0;
          for (int j = 0; j <= e1 + 1; ++j) {
            s = add64(s, i64(polys[e1][e1 + 1 - j]) * curr[e2][j]);
          }
          cumu[e1][e2] = mul(s % mod, invs[e1 + 1]);
        }
      }
      const int M = (i64(a) * N + b) / c;
```

```
      for (int e1 = 0; e1 <= S; ++e1) {
        int p = power_sum(e1, N);
        for (int e2 = 0; e2 <= S - e1; ++e2) {
          i64 t = 0;
          for (int i2 = 0; i2 < e2; ++i2) {
            t = add64(t, i64(cumu[e1][i2]) * binom[e2][i2]);
          }
          next[e1][e2] = add(p, mod - t % mod);
          p = mul(p, M);
        }
      }
    }
    swap(curr, next);
  }
  return curr[e1][e2];
}

int main() {
  init();
  int e = get_int();
  int l = get_int();
  int r = get_int();
  int a = get_int();
  int b = get_int();
  int c = get_int();
  int ans = scary_sum(r, a, b, c, 0, e);
  ans -= scary_sum(l - 1, a, b, c, 0, e);
  ans += mod;
  ans %= mod;
  printf("%d",ans);
}
```

7.77 积分表

Integrals of Rational Functions

$\int \frac{1}{1+x^2}dx = \tan^{-1} x$ $\quad \int \frac{1}{a^2+x^2}dx = \frac{1}{a}\tan^{-1}\frac{x}{a}$ $\quad \int \frac{x}{a^2+x^2}dx = \frac{1}{2}\ln|a^2+x^2|$ $\quad \int \frac{x^2}{a^2+x^2}dx = x - a\tan^{-1}\frac{x}{a}$ $\quad \int \frac{x^3}{a^2+x^2}dx = \frac{1}{2}x^2 - \frac{1}{2}a^2\ln|a^2+x^2|$

$\int \frac{1}{ax^2+bx+c}dx = \frac{2}{\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}}$ $\quad \int \frac{1}{(x+a)(x+b)}dx = \frac{1}{b-a}\ln\frac{a+x}{b+x},\ a \neq b$ $\quad \int \frac{x}{(x+a)^2}dx = \frac{a}{a+x} + \ln|a+x|$ $\quad \int \frac{x}{ax^2+bx+c}dx = \frac{1}{2a}\ln|ax^2+bx+c| - \frac{b}{a\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}}$

Integrals with Roots

$\int \frac{x}{\sqrt{x\pm a}}dx = \frac{2}{3}(x\mp 2a)\sqrt{x\pm a}$ $\quad \int \sqrt{\frac{x}{a-x}}dx = -\sqrt{x(a-x)} - a\tan^{-1}\frac{\sqrt{x(a-x)}}{x-a}$ $\quad \int \sqrt{\frac{x}{a+x}}dx = \sqrt{x(a+x)} - a\ln\left[\sqrt{x}+\sqrt{x+a}\right]$ $\quad \int x\sqrt{x^2\pm a^2}dx = \frac{1}{3}\left(x^2\pm a^2\right)^{3/2}$

$\int x\sqrt{ax+b}dx = \frac{2}{15a^2}(-2b^2+abx+3a^2x^2)\sqrt{ax+b}$ $\quad \int \sqrt{x(ax+b)}dx = \frac{1}{4a^{3/2}}\left[(2ax+b)\sqrt{ax(ax+b)}\ \ -b^2\ln\left|a\sqrt{x}+\sqrt{a(ax+b)}\right|\right]$ $\quad \int \sqrt{x^2\pm a^2}dx = \frac{1}{2}x\sqrt{x^2\pm a^2} \pm \frac{1}{2}a^2\ln\left|x+\sqrt{x^2\pm a^2}\right|$

$\int \sqrt{x^3(ax+b)}dx = \left[\frac{b}{12a} - \frac{b^2}{8a^2x} + \frac{x}{3}\right]\sqrt{x^3(ax+b)} + \frac{b^3}{8a^{5/2}}\ln\left|a\sqrt{x}+\sqrt{a(ax+b)}\right|$ $\quad \int \sqrt{a^2-x^2}dx = \frac{1}{2}x\sqrt{a^2-x^2} + \frac{1}{2}a^2\tan^{-1}\frac{x}{\sqrt{a^2-x^2}}$ $\quad \int \frac{x^2}{\sqrt{x^2\pm a^2}}dx = \frac{1}{2}x\sqrt{x^2\pm a^2} \mp \frac{1}{2}a^2\ln\left|x+\sqrt{x^2\pm a^2}\right|$

$\int \frac{1}{\sqrt{x^2\pm a^2}}dx = \ln\left|x+\sqrt{x^2\pm a^2}\right|$ $\quad \int \frac{1}{\sqrt{a^2-x^2}}dx = \sin^{-1}\frac{x}{a}$ $\quad \int \frac{x}{\sqrt{x^2\pm a^2}}dx = \sqrt{x^2\pm a^2}$ $\quad \int \frac{x}{\sqrt{a^2-x^2}}dx = -\sqrt{a^2-x^2}$ $\quad \int \sqrt{ax^2+bx+c}dx = \frac{b+2ax}{4a}\sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx{+}c)}\right|$

$\int x\sqrt{ax^2+bx+c} = \frac{1}{48a^{5/2}}\left(2\sqrt{a}\sqrt{ax^2+bx+c}\ \times\left(-3b^2+2abx+8a(c+ax^2)\right)\ +3(b^3-4abc)\ln\left|b+2ax+2\sqrt{a}\sqrt{ax^2+bx+c}\right|\right)$ $\quad \int \frac{1}{\sqrt{ax^2+bx+c}}dx = \frac{1}{\sqrt{a}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right|$

$\int \frac{x}{\sqrt{ax^2+bx+c}}dx = \frac{1}{a}\sqrt{ax^2+bx+c} - \frac{b}{2a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right|$ $\quad \int \frac{dx}{(a^2+x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2+x^2}}$

Integrals with Logarithms

$\int \ln(ax+b)dx = \left(x+\frac{b}{a}\right)\ln(ax+b) - x, a\neq 0$

$\int \frac{\ln ax}{x}dx = \frac{1}{2}(\ln ax)^2$ $\quad \int \ln(x^2+a^2)\,\mathrm{d}x = x\ln(x^2+a^2) + 2a\tan^{-1}\frac{x}{a} - 2x$ $\quad \int \ln(x^2-a^2)\,\mathrm{d}x = x\ln(x^2-a^2) + a\ln\frac{x+a}{x-a} - 2x$ $\quad \int x\ln(ax+b)dx = \frac{bx}{2a} - \frac{1}{4}x^2 + \frac{1}{2}\left(x^2-\frac{b^2}{a^2}\right)\ln(ax+b)$

$\int \ln\left(ax^2+bx+c\right)dx = \frac{1}{a}\sqrt{4ac-b^2}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}} - 2x + \left(\frac{b}{2a}+x\right)\ln\left(ax^2+bx+c\right)$ $\quad \int x\ln\left(a^2-b^2x^2\right)dx = -\frac{1}{2}x^2 + \frac{1}{2}\left(x^2-\frac{a^2}{b^2}\right)\ln\left(a^2-b^2x^2\right)$

Integrals with Exponentials

$\int x^n e^{ax}\,\mathrm{d}x = \frac{x^n e^{ax}}{a} - \frac{n}{a}\int x^{n-1}e^{ax}\,\mathrm{d}x$ $\quad \int xe^{-ax^2}\ \mathrm{d}x = -\frac{1}{2a}e^{-ax^2}$

Integrals with Trigonometric Functions

$\int \sin^3 ax dx = -\frac{3\cos ax}{4a} + \frac{\cos 3ax}{12a}$ $\quad \int \cos^2 ax dx = \frac{x}{2} + \frac{\sin 2ax}{4a}$

$\int \cos^3 ax dx = \frac{3\sin ax}{4a} + \frac{\sin 3ax}{12a}$ $\quad \int \cos ax\sin bx dx = \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)}, a\neq b$ $\quad \int \sin^2 ax\cos bx dx = -\frac{\sin[(2a-b)x]}{4(2a-b)} + \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)}$ $\quad \int \sin^2 x\cos x dx = \frac{1}{3}\sin^3 x$

$\int \cos^2 ax\sin bx dx = \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} - \frac{\cos[(2a+b)x]}{4(2a+b)}$ $\quad \int \cos^2 ax\sin ax dx = -\frac{1}{3a}\cos^3 ax$ $\quad \int \sin^2 ax\cos^2 bx dx = \frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} + \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)}$ $\quad \int \sin^2 ax\cos^2 ax dx = \frac{x}{8} - \frac{\sin 4ax}{32a}$

$\int \tan ax dx = -\frac{1}{a}\ln\cos ax$ $\quad \int \tan^2 ax dx = -x + \frac{1}{a}\tan ax$ $\quad \int \tan^3 ax dx = \frac{1}{a}\ln\cos ax + \frac{1}{2a}\sec^2 ax$ $\quad \int \sec x dx = \ln|\sec x+\tan x| = 2\tanh^{-1}\left(\tan\frac{x}{2}\right)$ $\quad \int \sec^2 ax dx = \frac{1}{a}\tan ax$

$\int \sec^3 x\,\mathrm{d}x = \frac{1}{2}\sec x\tan x + \frac{1}{2}\ln|\sec x+\tan x|$ $\quad \int \sec x\tan x dx = \sec x$ $\quad \int \sec^2 x\tan x dx = \frac{1}{2}\sec^2 x$ $\quad \int \sec^n x\tan x dx = \frac{1}{n}\sec^n x, n\neq 0$ $\quad \int \csc x dx = \ln\left|\tan\frac{x}{2}\right| = \ln|\csc x - \cot x| + C$

$\int \csc^2 ax dx = -\frac{1}{a}\cot ax$ $\quad \int \csc^3 x dx = -\frac{1}{2}\cot x\csc x + \frac{1}{2}\ln|\csc x - \cot x|$ $\quad \int \csc^n x\cot x dx = -\frac{1}{n}\csc^n x, n\neq 0$ $\quad \int \sec x\csc x dx = \ln|\tan x|$

Products of Trigonometric Functions and Monomials

$\int x\cos x dx = \cos x + x\sin x$ $\quad \int x\cos ax dx = \frac{1}{a^2}\cos ax + \frac{x}{a}\sin ax$ $\quad \int x^2\cos x dx = 2x\cos x + \left(x^2-2\right)\sin x$ $\quad \int x^2\cos ax dx = \frac{2x\cos ax}{a^2} + \frac{a^2x^2-2}{a^3}\sin ax$ $\quad \int x\sin x dx = -x\cos x + \sin x$

$\int x\sin ax dx = -\frac{x\cos ax}{a} + \frac{\sin ax}{a^2}$ $\quad \int x^2\sin x dx = \left(2-x^2\right)\cos x + 2x\sin x$ $\quad \int x^2\sin ax dx = \frac{2-a^2x^2}{a^3}\cos ax + \frac{2x\sin ax}{a^2}$

Products of Trigonometric Functions and Exponentials

$\int e^x\sin x dx = \frac{1}{2}e^x(\sin x - \cos x)$ $\quad \int e^{bx}\sin ax dx = \frac{1}{a^2+b^2}e^{bx}(b\sin ax - a\cos ax)$ $\quad \int e^{bx}\cos ax dx = \frac{1}{a^2+b^2}e^{bx}(a\sin ax + b\cos ax)$ $\quad \int xe^x\sin x dx = \frac{1}{2}e^x(\cos x - x\cos x + x\sin x)$

$\int xe^x\cos x dx = \frac{1}{2}e^x(x\cos x - \sin x + x\sin x)$ $\quad \int e^x\cos x dx = \frac{1}{2}e^x(\sin x + \cos x)$

8 字符串匹配

8.1 KMP

输入长度为 n 的模式串 a 以及长度为 m 的匹配串 b，下标从 0 开始，依次输出每个匹配成功的位置。

```
int nxt[N];
void kmp(int n,char*a,int m,char*b){
  int i,j;
  for(nxt[0]=j=-1,i=1;i<n;nxt[i++]=j){
    while(~j&&a[j+1]!=a[i])j=nxt[j];
    if(a[j+1]==a[i])j++;
  }
  for(j=-1,i=0;i<m;i++){
    while(~j&&a[j+1]!=b[i])j=nxt[j];
    if(a[j+1]==b[i])j++;
    if(j==n-1)printf("%d ",i),j=nxt[j];
  }
}
```

8.2 最小表示法

```
int n,i,t,a[N<<1];
int minrep(){
  int i=0,j=1,k=0,t;
  while(i<n&&j<n&&k<n)if(t=a[(i+k)%n]-a[(j+k)%n]){
    if(t>0)i+=k+1;else j+=k+1;
    if(i==j)j++;
    k=0;
  }else k++;
  return i<j?i:j;
}
int main(){
  for(scanf("%d",&n);i<n;i++)scanf("%d",&a[i]),a[i+n]=a[i];
  for(t=minrep(),i=0;i<n;i++)printf(i<n-1?"%d ":"%d",a[i+t]);
}
```

8.3 AC 自动机

s 是 t 的后缀等价于 t 串终止节点能通过 fail 指针走到 s 终止节点，即 t 串终止节点是 s 终止节点在 fail 树上的孩子。

```
int tot,son[N][26],id[N],fail[N],q[N];
int n;char s[N];//fail要清空
void insert(int p){
  for(int l=strlen(s),x=0,i=0,w;i<l;i++){
    if(!son[x][w=s[i]-'a'])son[x][w]=++tot;x=son[x][w];
    if(i==l-1)id[x]=p;
  }
}
void make(){
```

```
  int h=1,t=0,i,j,x;fail[0]=-1;
  for(i=0;i<26;i++)if(son[0][i])q[++t]=son[0][i];
  while(h<=t)for(x=q[h++],i=0;i<26;i++)
    if(son[x][i])fail[son[x][i]]=son[fail[x]][i],q[++t]=son[x][i];
    else son[x][i]=son[fail[x]][i];
}
void find(){
  for(int l=strlen(s),x=0,i=0,w,j;i<l;i++){
    x=son[x][w=s[i]-’a’];
    for(j=x;j;j=fail[j])if(id[j])printf(”%d ”,id[j]);
  }
}
int main(){
  scanf(”%d”,&n);
  for(int i=1;i<=n;i++)scanf(”%s”,s),insert(i);
  while(1)scanf(”%s”,s),find(),puts(””);
}
```

8.4 回文串

8.4.1 Manacher

对于一个位置 i，$[i-f[i]+1,i+f[i]-1]$ 是最长的以 i 为中心的奇回文串，$g[i]-i$ 是最长的以 i 为开头的回文串长度。

```
int n,m,i,r,p,f[N],g[N];char a[N],s[N];
int min(int a,int b){return a<b?a:b;}
void up(int&a,int b){if(a<b)a=b;}
int main(){
  while(1){
    scanf(”%s”,a+1),n=strlen(a+1);
    for(i=1;i<=n;i++)s[i<<1]=a[i],s[i<<1|1]=’#’;
    s[0]=’$’,s[1]=’#’,s[m=(n+1)<<1]=’@’;
    for(r=p=0,f[1]=1,i=2;i<m;i++){
      for(f[i]=r>i?min(r-i,f[p*2-i]):1;s[i-f[i]]==s[i+f[i]];f[i]++);
      if(i+f[i]>r)r=i+f[i],p=i;
    }
    for(i=0;i<=m;i++)g[i]=0;
    for(i=2;i<m;i++)up(g[i-f[i]+1],i+1);
    for(i=1;i<=m;i++)up(g[i],g[i-1]);
    ans=0;
    for(i=2;i<m;i+=2)printf(”%d ”,g[i]-i);puts(””);
  }
}
```

8.4.2 Palindromic Tree

N：串长。

S：字符集大小。

$text[1..all]$：字符串。

$son[x][y]$：第 x 个点所代表的回文串两边加上字符 y 后的回文串。

$fail[x]$：第 x 个点所代表的回文串的最长回文后缀。

$cnt[x]$：第 x 个点所代表的回文串的出现次数（需建完树后 $count()$ 一遍才可以）。

$len[x]$：第 x 个点所代表的回文串的长度。

```
const int N=300010,S=26;
int all,son[N][S],fail[N],cnt[N],len[N],text[N],last,tot;
int newnode(int l){
  for(int i=0;i<S;i++)son[tot][i]=0;
  cnt[tot]=0,len[tot]=l;
  return tot++;
}
void init(){
  last=tot=all=0;
  newnode(0),newnode(-1);
  text[0]=-1,fail[0]=1;
}
int getfail(int x){
  while(text[all-len[x]-1]!=text[all])x=fail[x];
  return x;
}
void add(int w){
  text[++all]=w;
  int x=getfail(last);
  if(!son[x][w]){
    int y=newnode(len[x]+2);
    fail[y]=son[getfail(fail[x])][w];
    son[x][w]=y;
  }
  cnt[last=son[x][w]]++;
}
void count(){for(int i=tot-1;~i;i--)cnt[fail[i]]+=cnt[i];}
```

8.4.3 Palindromic Tree 优化 DP

给定两个串 S 和 T，选择最少的 S 中的区间翻转使得 $S=T$，输出方案。

将两个串拼在一起，转化为偶回文串最少划分。

```
char s[N];
int S[N],n,fail[N],e[N][26],len[N],ec,lst,anc[N],diff[N],m,dp[N],from[N],pre[N];
inline int newnode(int l){pre[++ec]=-1;len[ec]=l;return ec;}
inline void init(){
  lst=1;while(n)S[n--]=0;S[0]=-1;
  while(~ec)len[ec]=fail[ec]=0,memset(e[ec--],0,104);
  newnode(0);newnode(-1);fail[0]=1;
}
inline int match(int p){while(S[n]!=S[n-len[p]-1])p=fail[p];return p;}
inline void extend(int c){
  S[++n]=c;int cur=match(lst);
  if(!e[cur][c]){
    int p=newnode(len[cur]+2);fail[p]=e[match(fail[cur])][c];e[cur][c]=p;
    diff[p]=len[p]-len[fail[p]];anc[p]=diff[p]==diff[fail[p]]?anc[fail[p]]:fail[p];
  }
  lst=e[cur][c];
}
int main(){
  int i,x;scanf("%s",s+1);m=strlen(s+1);for(i=1;i<=m;i++)S[(i<<1)-1]=s[i]-'a';
```

```
  scanf("%s",s+1);for(i=1;i<=m;i++)S[i<<1]=s[i]-'a';m<<=1;init();
  for(i=1;i<=m;i++){
    extend(S[i]);dp[i]=N;
    for(int x=lst;x;x=anc[x]){
      if(anc[x]!=fail[x])pre[x]=pre[fail[x]];else pre[x]=-1;
      int p=i-len[anc[x]]-diff[x];if(pre[x]==-1||pre[x]!=-1&&dp[p]<dp[pre[x]])pre[x]=p;
      if((~i&1)&&~pre[x]&&dp[pre[x]]+1<dp[i])dp[i]=dp[pre[x]]+1,from[i]=pre[x];
    }
    if((~i&1)&&S[i]==S[i-1]&&dp[i-2]<dp[i])dp[i]=dp[i-2],from[i]=i-2;
  }
  if(dp[m]==N)puts("-1"),exit(0);printf("%d\n",dp[m]);
  for(i=m;i;i=from[i])if(from[i]+2!=i)printf("%d %d\n",from[i]+2>>1,i>>1);
  return 0;
}
```

8.4.4 区间本质不同回文子串计数

```
const int N=100010;
int sz,last,go[N][26],len[N],diff[N],link[N],series_link[N],baby[N];
int n,m,i,j,k,x,y,f[N],bit[N];
int ans[N];
vector<pair<int,int> >poses[N],que[N];
char s[N];
inline int go_until_pal(int i,int v){
  while(s[i]!=s[i-len[v]-1])v=link[v];
  return v;
}
inline pair<vector<int>,vector<int> >add_char(int i){
  last=go_until_pal(i,last);
  int&v=go[last][s[i]-'a'];
  if(v==-1){
    v=sz++;
    len[v]=len[last]+2;
    if(!last)link[v]=1;
    else{
      last=go_until_pal(i,link[last]);
      link[v]=go[last][s[i]-'a'];
    }
    diff[v]=len[v]-len[link[v]];
    if(diff[v]==diff[link[v]]){
      baby[v]=baby[link[v]];
      series_link[v]=series_link[link[v]];
    }else{
      baby[v]=v;
      series_link[v]=link[v];
    }
  }
  last=v;
  pair<vector<int>,vector<int> >ans;
  for(int u=v;u!=1;u=series_link[u]){
    int B=baby[u];
    vector<pair<int,int> >&vec=poses[B];
    int left=i-len[u]+1;
    if(vec.size()&&vec.back().first==left)vec.back().second=len[u];
    else vec.push_back(make_pair(left,len[u]));
```

```
      ans.first.push_back(i-len[B]+1);
      if(vec.size()!=1){
        int L=vec[vec.size()-2].first,
            len_=vec[vec.size()-2].second,
            R=L+len_-1,
            L_=R-len[u]+1;
        ans.second.push_back(L_);
      }
      if(vec.size()!=1)if(vec[vec.size()-1].second==vec[vec.size()-2].second){
        vec[vec.size()-2]=vec[vec.size()-1];
        vec.pop_back();
      }
    }
  return ans;
}
inline void add(int x,int p){for(f[x]+=p;x<=n;x+=x&-x)bit[x]+=p;}
inline int ask(int x){int t=0;for(;x;x-=x&-x)t+=bit[x];return t;}
int main(){
  scanf("%s%d",s+1,&m);
  n=strlen(s+1);
  sz=2,last=0,len[0]=-1;
  for(i=0;i<=n+5;i++)memset(go[i],-1,sizeof(go[i]));
  for(i=0;i<m;i++){
    read(x),read(y);
    que[y].push_back(make_pair(x,i));
  }
  for(i=1;i<=n;i++){
    pair<vector<int>,vector<int> >vec=add_char(i);
    vector<int>adding=vec.first,deleting=vec.second;
    for(j=0;j<deleting.size();j++){
      if(!f[k=deleting[j]])continue;
      add(k,-1);
    }
    for(j=0;j<adding.size();j++){
      if(f[k=adding[j]]==1)continue;
      add(k,1);
    }
    for(j=0;j<que[i].size();j++)ans[que[i][j].second]=sz-2-ask(que[i][j].first-1);
  }
  for(i=0;i<m;i++)printf("%d\n",ans[i]);
}
```

在线版本：

```
const int N=100010,M=8500000;
int sz,last,go[N][26],len[N],diff[N],link[N],series_link[N],baby[N];
int type,n,m,i,j,k,x,y,now,f[N],cnt[N],T[N],tot,l[M],r[M],v[M];
vector<pair<int,int> >poses[N];
char s[N];
inline int go_until_pal(int i,int v){
  while(s[i]!=s[i-len[v]-1])v=link[v];
  return v;
}
inline pair<vector<int>,vector<int> >add_char(int i){
  last=go_until_pal(i,last);
  int&v=go[last][s[i]-'a'];
  if(v==-1){
```

```
    v=sz++;
    len[v]=len[last]+2;
    if(!last)link[v]=1;
    else{
      last=go_until_pal(i,link[last]);
      link[v]=go[last][s[i]-'a'];
    }
    diff[v]=len[v]-len[link[v]];
    if(diff[v]==diff[link[v]]){
      baby[v]=baby[link[v]];
      series_link[v]=series_link[link[v]];
    }else{
      baby[v]=v;
      series_link[v]=link[v];
    }
  }
  last=v;
  pair<vector<int>,vector<int> >ans;
  for(int u=v;u!=1;u=series_link[u]){
    int B=baby[u];
    vector<pair<int,int> >&vec=poses[B];
    int left=i-len[u]+1;
    if(vec.size()&&vec.back().first==left)vec.back().second=len[u];
    else vec.push_back(make_pair(left,len[u]));
    ans.first.push_back(i-len[B]+1);
    if(vec.size()!=1){
      int L=vec[vec.size()-2].first,
          len_=vec[vec.size()-2].second,
          R=L+len_-1,
          L_=R-len[u]+1;
      ans.second.push_back(L_);
    }
    if(vec.size()!=1)if(vec[vec.size()-1].second==vec[vec.size()-2].second){
      vec[vec.size()-2]=vec[vec.size()-1];
      vec.pop_back();
    }
  }
  return ans;
}
int ins(int x,int a,int b,int c,int p){
  int y=++tot;
  v[y]=v[x]+p;
  if(a==b)return y;
  int mid=(a+b)>>1;
  if(c<=mid)l[y]=ins(l[x],a,mid,c,p),r[y]=r[x];else l[y]=l[x],r[y]=ins(r[x],mid+1,b,c,p);
  return y;
}
int ask(int x,int a,int b,int d){
  if(b<=d)return v[x];
  int mid=(a+b)>>1,t=ask(l[x],a,mid,d);
  if(d>mid)t+=ask(r[x],mid+1,b,d);
  return t;
}
int main(){
  read(type),read(n),read(m);
  scanf("%s",s+1);
  sz=2,last=0,len[0]=-1;
```

```
  for(i=0;i<=n+5;i++)f[i]=0,memset(go[i],-1,sizeof(go[i]));
  for(i=1;i<=n;i++){
    T[i]=T[i-1];
    pair<vector<int>,vector<int> >vec=add_char(i);
    vector<int>adding=vec.first,deleting=vec.second;
    for(j=0;j<deleting.size();j++){
      if(!f[k=deleting[j]])continue;
      f[k]--;
      T[i]=ins(T[i],1,n,k,-1);
    }
    for(j=0;j<adding.size();j++){
      if(f[k=adding[j]]==1)continue;
      f[k]++;
      T[i]=ins(T[i],1,n,k,1);
    }
    cnt[i]=sz;
  }
  while(m--){
    read(x),read(y);
    if(type)x^=now,y^=now;
    now=cnt[y]-2;
    if(x>1)now-=ask(T[y],1,n,x-1);
    printf("%d\n",now);
  }
}
```

8.5 后缀数组

n：串长。

m：字符集大小。

$s[0..n-1]$：字符串。

$sa[1..n]$：字典序第 i 小的是哪个后缀。

$rank[0..n-1]$：后缀 i 的排名。

$height[i]$：$lcp(sa[i], sa[i-1])$。

```
int n,rank[N],sa[N],height[N],tmp[N],cnt[N];char s[N];
void suffixarray(int n,int m){
  int i,j,k;n++;
  for(i=0;i<n*2+5;i++)rank[i]=sa[i]=height[i]=tmp[i]=0;
  for(i=0;i<m;i++)cnt[i]=0;
  for(i=0;i<n;i++)cnt[rank[i]=s[i]]++;
  for(i=1;i<m;i++)cnt[i]+=cnt[i-1];
  for(i=0;i<n;i++)sa[--cnt[rank[i]]]=i;
  for(k=1;k<=n;k<<=1){
    for(i=0;i<n;i++){
      j=sa[i]-k;
      if(j<0)j+=n;
      tmp[cnt[rank[j]]++]=j;
    }
    sa[tmp[cnt[0]=0]]=j=0;
    for(i=1;i<n;i++){
      if(rank[tmp[i]]!=rank[tmp[i-1]]||rank[tmp[i]+k]!=rank[tmp[i-1]+k])cnt[++j]=i;
      sa[tmp[i]]=j;
```

```
    }
    memcpy(rank,sa,n*sizeof(int));
    memcpy(sa,tmp,n*sizeof(int));
    if(j>=n-1)break;
  }
  for(j=rank[height[i=k=0]=0];i<n-1;i++,k++)
    while(~k&&s[i]!=s[sa[j-1]+k])height[j]=k--,j=rank[sa[j]+1];
}
```

8.6 后缀树

在线往右添加字符。

```
const int inf=1<<25,S=256,N=5000;
int root,last,pos,need,remain,acnode,ace,aclen;
int n;
struct node{int st,en,lk,son[S];int len(){return min(en,pos+1)-st;}}tree[N<<1];
char text[N],tmp[N];
int new_node(int st,int en=inf){
  node nd;
  nd.st=st;nd.en=en;
  for(int i=nd.lk=0;i<S;i++)nd.son[i]=0;
  tree[++last]=nd;
  return last;
}
char acedge(){return text[ace];}
void addedge(int node){
  if(need)tree[need].lk=node;
  need=node;
}
bool down(int node){
  if(aclen>=tree[node].len())
    return ace+=tree[node].len(),aclen-=tree[node].len(),acnode=node,1;
  return 0;
}
void init(){
  need=last=remain=ace=aclen=0;
  root=acnode=new_node(pos=-1,-1);
}
void extend(char c){
  text[++pos]=c;need=0;remain++;
  while(remain){
    if(!aclen)ace=pos;
    if(!tree[acnode].son[acedge()])
      tree[acnode].son[acedge()]=new_node(pos),addedge(acnode);
    else{
      int nxt=tree[acnode].son[acedge()];
      if(down(nxt))continue;
      if(text[tree[nxt].st+aclen]==c){aclen++;addedge(acnode);break;}
      int split=new_node(tree[nxt].st,tree[nxt].st+aclen);
      tree[acnode].son[acedge()]=split;
      tree[split].son[c]=new_node(pos);
      tree[nxt].st+=aclen;
      tree[split].son[text[tree[nxt].st]]=nxt;
      addedge(split);
```

```
    }
    remain--;
    if(acnode==root&&aclen)aclen--,ace=pos-remain+1;
    else acnode=tree[acnode].lk?tree[acnode].lk:root;
  }
}
void show(int x,int dep,int sum,int y){
  sum+=tree[x].len();
  for(int i=0;i<dep;i++)putchar('-');
  for(int i=tree[x].st;i<min(tree[x].en,pos+1);i++)printf("%c",text[i]+'a');printf(":");
  printf("id=%d,[%d,%d],len=%d,maxsuf=%d ",x,tree[x].st,
min(tree[x].en,pos+1)-1,sum,f[x]);
  if(sum&&min(tree[x].en,pos+1)==pos+1){
    printf(" is suffix %d\n",pos-sum+1);
  }else puts("");
  for(int i=0;i<S;i++)if(tree[x].son[i])show(tree[x].son[i],dep+2,sum,i);
}
int search(){
  scanf("%s",tmp+1);
  n=strlen(tmp+1);
  int x=root,i=1,j;
  while(i<=n){
    if(tree[x].son[tmp[i]]){
      x=tree[x].son[tmp[i]];
      j=tree[x].st;
      while(i<=n&&j<min(tree[x].en,pos+1))if(tmp[i]==text[j])i++,j++;else return 0;
    }else return 0;
  }
  return 1;
}
int main(){
  init();
  scanf("%s",tmp+1);
  n=strlen(tmp+1);
  for(int i=1;i<=n;i++)extend(tmp[i]);extend('$');
  pos--;
  show(root,0,0,0);
  while(1)printf("%d\n",search());
}
```

8.7 后缀自动机

在线往右添加字符。

```
struct SuffixAutomaton{
  enum{N_CHAR=26,MX_LEN=1100000};
  struct Node{Node *fail,*next[N_CHAR];int val,right;};
  Node mempool[MX_LEN*2];int n_node;
  Node*new_node(int v){
    Node*p=&mempool[n_node++];
    for(int i=0;i<N_CHAR;++i)p->next[i]=0;
    return p->fail=0,p->right=0,p->val=v,p;
  }
  Node*root,*last;
  SuffixAutomaton(){clear();}
```

```
  void clear(){root=last=new_node(n_node=0);}
  void add(int c){
    Node*p=last,*np=new_node(p->val+1);
    while(p&&!p->next[c])p->next[c] = np,p = p->fail;
    if(!p)np->fail=root;
    else{
      Node*q=p->next[c];
      if(p->val+1==q->val)np->fail=q;
      else{
        Node*nq=new_node(p->val+1);
        for(int i=0;i<N_CHAR;++i)nq->next[i]=q->next[i];
        nq->fail=q->fail,q->fail=np->fail=nq;
        while(p&&p->next[c]==q)p->next[c]=nq,p=p->fail;
      }
    }
    last=np,np->right=1;
  }
  Node*go(const char*s){
    int cL=0;//与s匹配的长度
    Node*p=root;
    for(int i=0;s[i];++i){
      int c=s[i]-'a';
      if(p->next[c])p=p->next[c],++cL;
      else{
        while(p&&!p->next[c])p=p->fail;
        if(!p)cL=0,p=root;else cL=p->val+1,p=p->next[c];
      }
    }
    return p;
  }
  int d[MX_LEN*2];Node*b[MX_LEN*2];
  void topological_sort(){
    for(int i=0;i<=n_node;++i)d[i]=0;
    int mx_val=0;
    for(int i=0;i<n_node;++i)mx_val=max(mx_val,mempool[i].val),d[mempool[i].val]++;
    for(int i=1;i<=mx_val;++i)d[i]+=d[i-1];
    for(int i=0;i<n_node;++i)b[--d[mempool[i].val]]=&mempool[i];
  }
  void update_right(){
    topological_sort();
    for(int i=n_node-1;i;--i)b[i]->fail->right+=b[i]->right;
  }
};
```

8.8 后缀自动机 - Claris

```
int tot=1,last=1,pre[N<<1],son[N<<1][S],ml[N<<1];
void extend(int w){
  int p=++tot,x=last,r,q;
  for(ml[last=p]=ml[x]+1;x&&!son[x][w];x=pre[x])son[x][w]=p;
  if(!x)pre[p]=1;
  else if(ml[x]+1==ml[q=son[x][w]])pre[p]=q;
  else{
    pre[r=++tot]=pre[q];memcpy(son[r],son[q],sizeof son[r]);
    ml[r]=ml[x]+1;pre[p]=pre[q]=r;
```

```
    for(;x&&son[x][w]==q;x=pre[x])son[x][w]=r;
  }
}
```

8.9 后缀自动机统计子串出现次数

给一棵有根树，每条边有个字符。询问串 S 中每个区间在树上的出现次数之和。

```
#include<cstdio>
#include<cstring>
const int N=800010,M=N*2;
char ch,a[N*10];
int tot=1,last=1,pre[M],son[M][3],ml[M],d[M],h,t,q[M],cnt[M];
int n,i,x,l,fin[N];long long f[M],ans;
void extend(int w){
  int p=++tot,x=last,r,q;
  for(ml[last=p]=ml[x]+1,cnt[p]=1;x&&!son[x][w];x=pre[x])son[x][w]=p;
  if(!x)pre[p]=1;
  else if(ml[x]+1==ml[q=son[x][w]])pre[p]=q;
  else{
    pre[r=++tot]=pre[q];memcpy(son[r],son[q],sizeof son[r]);
    ml[r]=ml[x]+1;cnt[r]=0;pre[p]=pre[q]=r;
    for(;x&&son[x][w]==q;x=pre[x])son[x][w]=r;
  }
}
int main(){
  read(n);
  fin[1]=1;
  for(i=2;i<=n;i++){
    read(x);while(!((ch=getchar())>='a'&&ch<='c'));
    last=fin[x],extend(ch-'a'),fin[i]=last;
  }
  for(i=1;i<=tot;i++)d[pre[i]]++;
  for(i=h=1;i<=tot;i++)if(!d[i])q[++t]=i;
  while(h<=t){
    x=q[h++];
    cnt[pre[x]]+=cnt[x];
    if(pre[x]&&!(--d[pre[x]]))q[++t]=pre[x];
  }
  for(i=1;i<=t;i++)f[i]=1LL*(ml[i]-ml[pre[i]])*cnt[i];
  for(i=t;i;i--)f[q[i]]+=f[pre[q[i]]];
  scanf("%s",a);n=strlen(a);
  for(x=1,i=0;i<n;i++){
    ch=a[i]-'a';
    while(x>1&&!son[x][ch])x=pre[x],l=ml[x];
    if(!son[x][ch])x=1,l=0;
    else x=son[x][ch],l++,ans+=f[pre[x]]+1LL*(l-ml[pre[x]])*cnt[x];
  }
  printf("%lld",ans);
}
```

8.10 后缀平衡树

在线往左添加字符，一个串 S 的出现次数 = 字典序小于 $S*$ 的后缀个数 − 字典序小于 S 的后缀个数，其中 $*$ 为字符集中没出现的字符，且比任意字符都要大。

len：串长。

$s[i]$：从右往左第 i 个字符。

比较从右往左第 i 个字符开始的后缀与从右往左第 j 个字符开始的后缀的字典序等价于比较 $tm[i]$ 与 $tm[j]$。

$ins(len)$：插入从右往左第 len 个字符开始的后缀。

```
typedef unsigned long long ll;
const ll inf=1ULL<<63;
const double A=0.8;
ll tl[N],tr[N],tm[N];
int size[N],son[N][2],f[N],v[N],tot,root,id[N],cnt;
char s[N];
bool cmp(int a,int b){return s[a]==s[b]?tm[a-1]>tm[b-1]:s[a]>s[b];}
int ins(int x,int p){
  int b=cmp(p,v[x]);
  if(!son[x][b]){
    son[x][b]=++tot;f[tot]=x;v[tot]=p;
    if(!b)tl[tot]=tl[x],tr[tot]=tm[x];else tl[tot]=tm[x],tr[tot]=tr[x];
    tm[tot]=(tl[tot]+tr[tot])>>1;
    return tot;
  }else return ins(son[x][b],p);
}
void dfs(int x){
  if(son[x][0])dfs(son[x][0]);
  id[++cnt]=x;
  if(son[x][1])dfs(son[x][1]);
}
int build(int fa,int l,int r,ll a,ll b){
  int mid=(l+r)>>1,x=id[mid];
  f[x]=fa;son[x][0]=son[x][1]=0;size[x]=1;tl[x]=a;tr[x]=b;tm[x]=(a+b)>>1;
  if(l==r)return x;
  if(l<mid)size[x]+=size[son[x][0]=build(x,l,mid-1,a,tm[x])];
  if(r>mid)size[x]+=size[son[x][1]=build(x,mid+1,r,tm[x],b)];
  return x;
}
int rebuild(int x){cnt=0;dfs(x);return build(f[x],1,cnt,tl[x],tr[x]);}
void insert(int p){
  if(!root){root=tot=size[1]=1;v[1]=p;tr[1]=inf,tm[1]=inf>>1;return;}
  int x=ins(root,p);
  int deep=0,z=x;while(z)size[z]++,z=f[z],deep++;
  if(deep<log(tot)/log(1/A))return;
  while(1.0*size[son[x][0]]<A*size[x]&&1.0*size[son[x][1]]<A*size[x])x=f[x];
  if(!x)return;
  if(x==root){root=rebuild(x);return;}
  int y=f[x],b=son[y][1]==x,now=rebuild(x);
  son[y][b]=now;
}
```

8.11 Basic Factor Dictionary

输入一个串，$O(\log n)$ 询问区间 $[a, b]$ 的最短循环节，询问下标从 1 开始。

```
#include<cstdio>
#include<cstring>
#include<vector>
#include<algorithm>
using namespace std;
#define PB push_back
#define int long long
#define PII pair<int,int>
#define FI first
#define SE second
#define R(i,n) for(int i=0;i<n;i++)
#define ALL(x) (x).begin(),(x).end()
#define SZ(x) ((int)(x).size())
#define MAX 100010
struct ciag{
  int a,il,b;
  ciag(){}
  ciag(int _a,int _il,int _b){a=_a,il=_il,b=_b;}
  inline bool add(int x){
    if(il==1){
      b=x-a;
      il=2;
      return 1;
    }
    if(a+il*b==x){
      il++;
      return 1;
    }
    return 0;
  }
  inline int ost(){return a+(il-1)*b;}
};
int exgcd(int a,int b,int&x,int&y){
  if(a<b)return exgcd(b,a,y,x);
  if(b==0)return x=1,y=0,a;
  int t,d=exgcd(b,a%b,t,x);
  return y=t-x*(a/b),d;
}
inline int floordiv(int a,int b){
  if(b<0)a=-a,b=-b;
  int d=a/b,m=a-d*b;
  if(m<0)d--;
  return d;
}
inline int ceildiv(int a,int b){
  int r=floordiv(a,b);
  if(a%b)r++;
  return r;
}
inline ciag marek(ciag&a,ciag&b){
  ciag wynik;
  int n,m,da=b.a-a.a,g=exgcd(a.b,b.b,n,m);
  if(da%g)return ciag(1,0,1);
```

```
  n*=da/g;
  wynik.b=a.b/g*b.b;
  int elem=a.a+a.b*n,maxA=a.ost(),maxB=b.ost(),
      minIle=max(ceildiv(a.a-elem,wynik.b),ceildiv(b.a-elem,wynik.b)),
      maxIle=min(floordiv(maxA-elem,wynik.b),floordiv(maxB-elem,wynik.b));
  if(minIle>maxIle)return ciag(1,0,1);
  wynik.a=elem+minIle*wynik.b;
  wynik.il=maxIle-minIle+1;
  return wynik;
}
vector<vector<ciag> >ciagi[MAX];
int n,len,ans,kmr[19][MAX];
vector<pair<PII,int> >x;
char z[MAX];
vector<int>wys[MAX];
inline void mapuj(int j){
  sort(ALL(x));
  int id=0;
  R(i,SZ(x)){
    if(i&&x[i-1].FI!=x[i].FI)id++;
    kmr[j][x[i].SE]=id;
    wys[id].PB(x[i].SE);
  }
  ciagi[j].resize(id+1);
  R(i,id+1){
    for(vector<int>::iterator it=wys[i].begin();it!=wys[i].end();it++)
      if(ciagi[j][i].empty()||!ciagi[j][i].back().add(*it))
        ciagi[j][i].PB(ciag(*it,1,0));
    wys[i].clear();
  }
}
inline void licz_kmr(){
  R(i,n)x.PB(pair<PII,int>(PII(z[i],0),i));
  mapuj(0);
  int krok=1,j=0;
  while(krok<n){
    x.clear();
    R(i,n-krok)x.PB(pair<PII,int>(PII(kmr[j][i],kmr[j][i+krok]),i));
    mapuj(++j);
    krok<<=1;
  }
}
inline int pierw(int j,int k,int x){
  int l=-1,r=SZ(ciagi[j][k]);
  while(l+1!=r){
    int m=(l+r)>>1;
    if(ciagi[j][k][m].ost()>=x)r=m;else l=m;
  }
  return r;
}
inline ciag przetnij(ciag xx,ciag y,int a,int b,int k){
  xx.a=b-xx.ost();
  y.a=k+y.a-a;
  if(xx.il==1)xx.b=1;
  if(y.il==1)y.b=1;
  return ciag(marek(xx,y));
}
```

```
inline bool spr(int a,int b,int j){
  int k=1<<j,aa=kmr[j][a],bb=kmr[j][b-k],pa=pierw(j,aa,b-2*k),pbb=pierw(j,bb,a),res=0;
  while(pa<SZ(ciagi[j][aa])){
    if(ciagi[j][aa][pa].a>b-k)break;
    int pb=pbb;
    while(pb<SZ(ciagi[j][bb])){
      if(ciagi[j][bb][pb].a>a+k)break;
      ciag pom=przetnij(ciagi[j][aa][pa],ciagi[j][bb][pb],a,b,k);
      if(pom.il!=0){
        int wyn=pom.ost(),lim=min(2*k,b-a-1);
        if(wyn>lim){
          int cof=(wyn-lim+pom.b-1)/pom.b;
          wyn-=cof*pom.b;
        }
        if(pom.a<=wyn)res=max(res,wyn);
      }
      pb++;
    }
    pa++;
  }
  if(res>=k){
    ans=len-res;
    return 1;
  }
  return 0;
}
inline void zap(int a,int b){
  ans=len;
  int j=0,dl=b-a;
  while((1<<(j+1))<dl)j++;
  while(~j&&!spr(a,b,j))j--;
}
#undef int
int main(){
  int T,C=0,q;
  scanf("%d",&T);
  while(T--){
    printf("Case #%d:\n",++C);
    for(int i=0;i<MAX;i++)ciagi[i].clear(),wys[i].clear();
    x.clear();memset(kmr,0,sizeof kmr);
    scanf("%s%d",z,&q);n=strlen(z);
    licz_kmr();
    while(q--){
      int a,b;
      scanf("%d%d",&a,&b);
      len=b-a+1;
      zap(a-1,b);
      printf("%d\n",(int)ans);
    }
  }
}
```

8.12 可持久化 KMP

维护一个字符串 S，一开始是空串，进行 m 次操作，每次操作包含两个整数 x_i, c_i，表示这次操作的字符串为在第 x_i 次操作之后的字符串末尾添加一个字符 c_i 所形成的字符串。

请在每次操作完毕之后，求出该次操作得到的字符串最短的循环节的长度。

```
const int N=300005,M=N*20*3;
int n,m,i,x,y,z,d[N],nxt,T[N],v[M],l[M],r[M],tot;
int change(int x,int a,int b,int c,int p){
  int y=++tot;
  if(a==b){
    v[y]=p;
    return y;
  }
  int mid=(a+b)>>1;
  if(c<=mid)l[y]=change(l[x],a,mid,c,p),r[y]=r[x];
  else l[y]=l[x],r[y]=change(r[x],mid+1,b,c,p);
  return y;
}
int ask(int x,int a,int b,int c){
  if(a==b)return v[x];
  int mid=(a+b)>>1;
  return c<=mid?ask(l[x],a,mid,c):ask(r[x],mid+1,b,c);
}
int main(){
  scanf("%d%d",&n,&m);
  for(i=1;i<=n;i++){
    scanf("%d%d",&x,&y);
    d[i]=d[x]+1;
    z=ask(T[x],0,n,x);
    nxt=ask(z,1,m,y);
    printf("%d\n",d[i]-d[nxt]);
    T[i]=change(T[x],0,n,x,change(z,1,m,y,i));
    T[i]=change(T[i],0,n,i,ask(T[i],0,n,nxt));
  }
}
```

8.13 扩展 KMP

返回 $z[i] = lcp(suf[i], suf[0])$。

```
std::vector<int> ext_kmp(char *s, int n) {
  std::vector<int> z(n, 0);
  for (int i = 1, x = 0, y = 0; i < n; ++i) {
    if (i <= y) z[i] = std::min(y - i, z[i - x]);
    while (i + z[i] < n && s[i + z[i]] == s[z[i]]) ++z[i];
    if (y <= i + z[i]) x = i, y = i + z[i];
  }
  z[0] = n;
  return z;
}
```

8.14 循环最长公共子序列

给定两个串 A 和 B，可以旋转 B，求最长公共子序列，时间复杂度 $O(nm)$。

```
const int N = 4000 + 10;
int dp[N][N], from[N][N];
int clcs(char s[], char t[]) {
  int n = strlen(s), m = strlen(t);
  auto eq = [&](int a, int b) {
    return s[(a - 1) % n] == t[(b - 1) % m];
  };
  dp[0][0] = from[0][0] = 0;
  for (int i = 0; i <= n * 2; ++i) {
    for (int j = 0; j <= m; ++j) {
      dp[i][j] = 0;
      if (j && dp[i][j - 1] > dp[i][j]) {
        dp[i][j] = dp[i][j - 1];
        from[i][j] = 0;
      }
      if (i && j && eq(i, j) && dp[i - 1][j - 1] + 1 > dp[i][j]) {
        dp[i][j] = dp[i - 1][j - 1] + 1;
        from[i][j] = 1;
      }
      if (i && dp[i - 1][j] > dp[i][j]) {
        dp[i][j] = dp[i - 1][j];
        from[i][j] = 2;
      }
    }
  }
  int ret = 0;
  for (int i = 0; i < n; ++i) {
    ret = std::max(ret, dp[i + n][m]);
    int x = i + 1, y = 0;
    while (y <= m && from[x][y] == 0) ++y;
    for (; y <= m && x <= n * 2; ++x) {
      from[x][y] = 0, --dp[x][m];
      if (x == n * 2) break;
      for (; y <= m; ++y) {
        if (from[x + 1][y] == 2) break;
        if (y + 1 <= m && from[x + 1][y + 1] == 1) {
          ++y; break;
        }
      }
    }
  }
  return ret;
}
```

8.15 生成 Lyndon Words

按字典序从小到大生成长度不超过 n，字符集大小为 m 的所有 Lyndon words。

```
void lyndon_generate(int n, int m) {
  char z = 'a' + m - 1, s[1000];
  s[0] = 'a' - 1;
```

```
  for (int i = 1, x = 1; ; ++i) {
    s[x - 1]++; s[x] = 0;
    puts(s);
    for (int j = x; j < n; ++j) s[j] = s[j - x];
    for (x = n; s[x - 1] == z; --x);
  }
}
```

8.16 ALCS

输入 S 和 T，对于 T 的每个区间，输出 S 和它的 LCS。时间复杂度 $O(n^2)$。

```
#include<vector>
#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>
using namespace std;
const int N = 2005;
int n, m;
char s[N], t[N];
int from[N][N], cnt[N], ans[N][N], dp[N][N];
int main() {
    memset(from, -1, sizeof(from));
    memset(dp, -1, sizeof(dp));
    scanf("%s%s", s, t);
    n = strlen(s), m = strlen(t);
    dp[0][0] = 0, from[0][0] = 0;
    for (int i = 0; i <= m; ++i) {
        for (int j = 0; j <= n; ++j) {
            if (j && dp[i][j - 1] > dp[i][j]) {
                dp[i][j] = dp[i][j - 1];
                from[i][j] = 0;
            }
            if (i && j && s[j - 1] == t[i - 1] && dp[i - 1][j - 1] + 1 > dp[i][j]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
                from[i][j] = 1;
            }
            if (i && dp[i - 1][j] > dp[i][j]) {
                dp[i][j] = dp[i - 1][j];
                from[i][j] = 2;
            }
        }
    }
    for (int i = 1; i <= m; ++i) {
        cnt[i - 1] = 0;
        for (int j = 0; j <= n; ++j) {
            cnt[i - 1] += !!from[i][j];
        }
        --cnt[i - 1];
    }
    for (int i = 0; i < m; ++i) {
/*
for (int j = i; j <= m; ++j) {
    for (int k = 0; k <= n; ++ k) {
```

```
        putchar(from[j][k] == 1 ? '\\' : ' ');
        putchar(from[j][k] == 2 ? '|' : ' ');
    }
    puts("");
    for (int k = 0; k <= n; ++ k) {
        putchar(from[j][k] == 0 ? '-' : ' ');
        putchar('*');
    }
    printf("\n");
}
printf("\n");
*/
        for (int j = i; j < m; ++j) {
            ans[i][j] = cnt[j];
        }
        int x = i, y = 0;
        from[i + 1][0] = 0;
        while (y <= n) {
            if (x < m && from[x + 1][y] == 2) {
                from[x + 1][y] = 0;
                --cnt[x];
                ++x;
            } else if (x < m && from[x + 1][y + 1] == 1) {
                from[x + 1][y + 1] = 0;
                --cnt[x];
                ++x, ++y;
            } else {
                ++y;
            }
        }
    }
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < m; ++j) {
            if (j < i) {
                printf("  ");
            } else {
                printf("%d%c", ans[i][j], j == m - 1 ? '\n' : ' ');
            }
        }
    }
}
```

8.17 Shift And

维护下标 0 开始数字串，支持 3 种操作：

0. 在位置 x_i 处插入一个字符串 y_i。

1. 删除位置 $[x_i, y_i)$ 的字符串。

2. 查询位置 $[x_i, y_i)$ 的字符串包含多少次给定的子串 z_i。

```
typedef unsigned int U;
const int N=1000010;
int T,op,x,y,i,j,n,cb,m,g[65536];char s[N];
inline int popcount(U x){return g[x>>16]+g[x&65535];}
struct BIT{
```

```
  U v[N/32+5];
  //初始化为 0
  void clr(){for(int i=0;i<=cb;i++)v[i]=0;}
  //返回第x位的值
  U get(int x){return v[x>>5]>>(x&31)&1;}
  //将第x位设置为y
  void set(int x,U y){if((v[x>>5]>>(x&31)&1)^y)v[x>>5]^=1U<<(x&31);}
  //翻转第x位
  void flip(int x){v[x>>5]^=1U<<(x&31);}
  //将从x开始的后缀后移y位，不清空移走部分
  void shr(int x,int y){
    int A=y>>5,B=y&31,C=(32-B)&31,D=x>>5;
    v[cb+A+1]=0;
    for(int i=cb;i>D;i--){
      if(C)v[i+A+1]|=v[i]>>C;
      v[i+A]=v[i]<<B;
    }
    for(int i=(D<<5)+31;i>=x;i--)set(i+y,get(i));
  }
  //将从x+y开始的后缀前移y位，不清空移走部分
  void shl(int x,int y){
    int A=y>>5,B=y&31,C=(32-B)&31,D=x>>5,E=(D<<5)+31;
    for(int i=x;i<=E;i++)set(i,get(i+y));
    for(int i=D+1;i<=cb;i++){
      v[i]=v[i+A]>>B;
      if(C)v[i]|=v[i+A+1]<<C;
    }
  }
  //复制
  void copy(int x,int y,const BIT&p){for(int i=x;i<=y;i++)v[i]=p.v[i];}
  //与运算
  void And(int x,int y,const BIT&p){for(int i=x;i<=y;i++)v[i]&=p.v[i];}
  //对于第[x,y]块后移一格
  void shift(int x,int y){
    for(int i=y;i>=x;i--){
      v[i]<<=1;
      if(i&&(v[i-1]>>31&1))v[i]|=1;
    }
  }
  //询问[x,y]之间 1 的个数
  int count(int x,int y){
    int A=x>>5,B=y>>5,C,ret=0;
    if(A==B){
      for(int i=x;i<=y;i++)if(v[A]>>(i&31)&1)ret++;
      return ret;
    }
    for(int i=A+1;i<B;i++)ret+=popcount(v[i]);
    C=(A<<5)+31;
    for(int i=x;i<=C;i++)if(v[A]>>(i&31)&1)ret++;
    C=B<<5;
    for(int i=C;i<=y;i++)if(v[B]>>(i&31)&1)ret++;
    return ret;
  }
}f[10],S;
inline void getstr(){
  scanf("%s",s);
  m=strlen(s);
```

```
  for(i=0;i<m;i++)s[i]-='0';
}
inline void ins(int x){
  for(i=0;i<10;i++)f[i].shr(x,m);
  for(i=0;i<10;i++)for(j=0;j<m;j++)f[i].set(x+j,s[j]==i);
  n+=m;
  cb=n>>5;
}
inline void del(int x,int y){
  if(x==y)return;
  n-=y-x;
  cb=n>>5;
  for(i=0;i<10;i++)f[i].shl(x,y-x);
}
inline int ask(int x,int y){
  if(y-x<m)return 0;
  y--;
  int A=x>>5,B=y>>5;
  S.copy(A,B,f[s[0]]);
  for(i=1;i<m;i++)S.shift(A,B),S.And(A,B,f[s[i]]);
  return S.count(x+m-1,y);
}
int main(){
  for(i=1;i<65536;i++)g[i]=g[i>>1]+(i&1);
  scanf("%d",&T);
  while(T--){
    scanf("%d%d",&op,&x);
    if(op==0){
      getstr();
      ins(x);
    }else if(op==1){
      scanf("%d",&y);
      del(x,y);
    }else{
      scanf("%d",&y);
      getstr();
      printf("%d\n",ask(x,y));
    }
  }
}
```

9 随机化

9.1 Pollard Rho

```
#include<cstdio>
#include<algorithm>
#define C 2730
#define S 3
using namespace std;
typedef long long ll;
ll n;
ll gcd(ll a,ll b){return b?gcd(b,a%b):a;}
ll mul(ll a,ll b,ll n){return(a*b-(ll)(a/(long double)n*b+1e-3)*n+n)%n;}
ll pow(ll a, ll b, ll n){
  ll d=1;
  a%=n;
  while(b){
    if(b&1)d=mul(d,a,n);
    a=mul(a,a,n);
    b>>=1;
  }
  return d;
}
bool check(ll a,ll n){
  ll m=n-1,x,y;int i,j=0;
  while(!(m&1))m>>=1,j++;
  x=pow(a,m,n);
  for(i=1;i<=j;x=y,i++){
    y=pow(x,2,n);
    if((y==1)&&(x!=1)&&(x!=n-1))return 1;
  }
  return y!=1;
}
bool miller_rabin(int times,ll n){
  ll a;
  if(n==1)return 0;
  if(n==2)return 1;
  if(!(n&1))return 0;
  while(times--)if(check(rand()%(n-1)+1,n))return 0;
  return 1;
}
ll pollard_rho(ll n,int c){
  ll i=1,k=2,x=rand()%n,y=x,d;
  while(1){
    i++,x=(mul(x,x,n)+c)%n,d=gcd(y-x,n);
    if(d>1&&d<n)return d;
    if(y==x)return n;
    if(i==k)y=x,k<<=1;
  }
}
void findfac(ll n,int c){
  if(n==1)return;
  if(miller_rabin(S,n)){
    //找到了质因子n
    return;
  }
```

```
  ll m=n;
  while(m==n)m=pollard_rho(n,c--);
  findfac(m,c),findfac(n/m,c);
}
int main(){while(~scanf("%lld",&n))findfac(n,C);}
```

9.2 最小圆覆盖

给定 n 个点 $b[0], b[1], ..., b[n-1]$，返回最小的能覆盖所有点的圆，圆心为 O，半径为 R。

```
double R,eps=1e-10;
struct P{double x,y;}a[N],O;
double dis(P x,P y){return sqrt((x.x-y.x)*(x.x-y.x)+(x.y-y.y)*(x.y-y.y));}
P center(P x,P y,P z){
  double a1=y.x-x.x,b1=y.y-x.y,
         c1=(a1*a1+b1*b1)/2,a2=z.x-x.x,
         b2=z.y-x.y,c2=(a2*a2+b2*b2)/2,
         d=a1*b2-a2*b1;
  return (P){x.x+(c1*b2-c2*b1)/d,x.y+(a1*c2-a2*c1)/d};
}
void cal(int n,P*b){
  int i,j,k,n=0;
  for(i=0;i<n;i++)a[i]=b[i];
  for(i=0;i<n;i++)swap(a[rand()%n],a[i]);
  for(O=a[0],R=0,i=1;i<n;i++)if(dis(a[i],O)>R+eps)
    for(O=a[i],R=0,j=0;j<i;j++)if(dis(a[j],O)>R+eps){
      O=(P){(a[i].x+a[j].x)/2,(a[i].y+a[j].y)/2},R=dis(O,a[i]);
      for(k=0;k<j;k++)if(dis(a[k],O)>R+eps)O=center(a[k],a[j],a[i]),R=dis(O,a[i]);
    }
}
```

9.3 最小球覆盖

```
int n,cnt,i;double R,tmp;
struct P{
  double x,y,z;
  P(){}
  P(double _x,double _y,double _z){x=_x,y=_y,z=_z;}
  P operator+(const P&b){return P(x+b.x,y+b.y,z+b.z);}
  P operator-(const P&b){return P(x-b.x,y-b.y,z-b.z);}
  P operator*(double b){return P(x*b,y*b,z*b);}
  P operator/(double b){return P(x/b,y/b,z/b);}
}a[200000],b[4],O;
double dis(const P&a,const P&b){
  return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y)+(a.z-b.z)*(a.z-b.z);
}
double dot(const P&a,const P&b){
  return a.x*b.x+a.y*b.y+a.z*b.z;
}
void ball(){
  P q[3];double m[3][3],f[3],L[3],det;
  int i,j;O.x=O.y=O.z=R=0;
  switch(cnt){
```

```
  case 1:O=b[0];break;
  case 2:O=(b[0]+b[1])/2;R=dis(O,b[0]);break;
  case 3:
    for(i=0;i<2;i++)q[i]=b[i+1]-b[0];
    for(i=0;i<2;i++)for(j=0;j<2;j++)m[i][j]=dot(q[i],q[j])*2;
    for(i=0;i<2;i++)f[i]=dot(q[i],q[i]);
    if(fabs(det=m[0][0]*m[1][1]-m[0][1]*m[1][0])<eps)return;
    L[0]=(f[0]*m[1][1]-f[1]*m[0][1])/det;
    L[1]=(f[1]*m[0][0]-f[0]*m[1][0])/det;
    O=b[0]+q[0]*L[0]+q[1]*L[1];
    R=dis(O,b[0]);
    break;
  case 4:
    for(i=0;i<3;i++)q[i]=b[i+1]-b[0],f[i]=dot(q[i],q[i]);
    for(i=0;i<3;i++)for(j=0;j<3;j++)m[i][j]=dot(q[i],q[j])*2;
    det=m[0][0]*m[1][1]*m[2][2]
    +m[0][1]*m[1][2]*m[2][0]
    +m[0][2]*m[2][1]*m[1][0]
    -m[0][2]*m[1][1]*m[2][0]
    -m[0][1]*m[1][0]*m[2][2]
    -m[0][0]*m[1][2]*m[2][1];
    if(fabs(det)<eps)return;
    for(j=0;j<3;j++){
      for(i=0;i<3;i++)m[i][j]=f[i];
      L[j]=(m[0][0]*m[1][1]*m[2][2]
      +m[0][1]*m[1][2]*m[2][0]
      +m[0][2]*m[2][1]*m[1][0]
      -m[0][2]*m[1][1]*m[2][0]
      -m[0][1]*m[1][0]*m[2][2]
      -m[0][0]*m[1][2]*m[2][1])/det;
      for(i=0;i<3;i++)m[i][j]=dot(q[i],q[j])*2;
    }
    O=b[0];
    for(i=0;i<3;i++)O=O+q[i]*L[i];
    R=dis(O,b[0]);
  }
}
void minball(int n){
  ball();
  if(cnt<4)for(int i=0;i<n;i++)if(dis(O,a[i])-R>eps){
    b[cnt++]=a[i];minball(i);cnt--;
    if(i>0){
      P t=a[i];
      memmove(&a[1],&a[0],sizeof(P)*i);
      a[0]=t;
    }
  }
}
int main(){
  while(~scanf("%d",&n)){
    for(i=0;i<n;i++)scanf("%lf%lf%lf",&a[i].x,&a[i].y,&a[i].z);
    random_shuffle(a,a+n);R=-1;
    for(i=0;i<n;i++)if(dis(O,a[i])-R>eps)cnt=1,b[0]=a[i],minball(i);
    printf("%.4f %.4f %.4f %.4f\n",sqrt(R),O.x,O.y,O.z);
  }
}
```

10 计算几何

10.1 半平面交

```
const int N=600;
const double eps=1e-10;
struct P{
  double x,y;
  P(){x=y=0;}
  P(double _x,double _y){x=_x,y=_y;}
  P operator-(const P&a)const{return P(x-a.x,y-a.y);}
  P operator+(const P&a)const{return P(x+a.x,y+a.y);}
  P operator*(double a)const{return P(x*a,y*a);}
  void read(){scanf("%lf%lf",&x,&y);}
}p[N],a[N];
struct L{
  P p,v;double a;
  L(){}
  L(P _p,P _v){p=_p,v=_v;}
  bool operator<(const L&b)const{return a<b.a;}
  void cal(){a=atan2(v.y,v.x);}
}line[N],q[N];
int n,m,i,cl;
double cross(const P&a,const P&b){return a.x*b.y-a.y*b.x;}
//新的半平面，在这条向量a->b的逆时针方向
void newL(const P&a,const P&b){line[++cl]=L(a,b-a);}
bool left(const P&p,const L&l){return cross(l.v,p-l.p)>0;}
P pos(const L&a,const L&b){
  P x=a.p-b.p;double t=cross(b.v,x)/cross(a.v,b.v);
  return a.p+a.v*t;
}
double halfplane(){
  for(int i=1;i<=cl;i++)line[i].cal();
  sort(line+1,line+cl+1);
  int h=1,t=1;
  q[1]=line[1];
  for(int i=2;i<=cl;i++){
    while(h<t&&!left(p[t-1],line[i]))t--;
    while(h<t&&!left(p[h],line[i]))h++;
    if(fabs(cross(q[t].v,line[i].v))<eps)q[t]=left(q[t].p,line[i])?q[t]:line[i];
    else q[++t]=line[i];
    if(h<t)p[t-1]=pos(q[t],q[t-1]);
  }
  while(h<t&&!left(p[t-1],q[h]))t--;
  p[t]=pos(q[t],q[h]);
  if(t-h<=1)return 0;
  double ans=0;
  for(int i=h;i<t;i++)ans+=cross(p[i],p[i+1]);
  return (ans+cross(p[t],p[h]))/2;
}
int main(){
  scanf("%d",&n);
  while(n--){
    scanf("%d",&m);
    for(i=0;i<m;i++)a[i].read();
    for(i=0;i<m;i++)newL(a[i],a[(i+1)%m]);
```

```
  }
  printf("%.3f",halfplane());
}
```

10.2 最小矩形覆盖

求出凸包后旋转卡壳。注意不能有重复点。

```
#include<cstdio>
#include<cmath>
#include<algorithm>
#include<vector>
using namespace std;
typedef double DB;
const int N=88888;
const DB eps=1e-8,pi=acos(-1);
DB ans;
int n;
struct PT{
  DB x,y;
  PT(DB x=0,DB y=0):x(x),y(y){}
  void input(){scanf("%lf%lf",&x,&y);}
  bool operator<(const PT&p)const{
    if(fabs(x-p.x))return x<p.x;
    return y<p.y;
  }
  void output(){printf("%.10f %.10f\n",x,y);}
}p[N],q[N];
vector<PT>ret;
DB vect(PT p,PT p1,PT p2){
  return (p1.x-p.x)*(p2.y-p.y)-(p1.y-p.y)*(p2.x-p.x);
}
int convex_hull(PT*p,int n,PT*q){
  int i,k,m;
  sort(p,p+n);
  m=0;
  for(i=0;i<n;q[m++]=p[i++])while(m>1&&vect(q[m-2],q[m-1],p[i])<eps)m--;
  k=m;
  for(i=n-2;i>=0;q[m++]=p[i--])while(m>k&&vect(q[m-2],q[m-1],p[i])<eps)m--;
  return --m;
}
PT get(PT p,DB x){
  return PT(p.x*cos(x)-p.y*sin(x),p.x*sin(x)+p.y*cos(x));
}
bool is_ext(int id,PT pp){
  if(vect(p[id],PT(p[id].x+pp.x,p[id].y+pp.y),p[id+1])<-eps)return 0;
  if(vect(p[id],PT(p[id].x+pp.x,p[id].y+pp.y),p[(id-1+n)%n])<-eps)return 0;
  return 1;
}
PT inter(PT p1,PT p2,PT p3,PT p4){
  p2.x+=p1.x;
  p2.y+=p1.y;
  p4.x+=p3.x;
  p4.y+=p3.y;
  DB s=vect(p1,p2,p3),s1=vect(p1,p2,p4);
```

```
  DB t=s/(s-s1);
  return PT(p3.x+(p4.x-p3.x)*t,p3.y+(p4.y-p3.y)*t);
}
void solve(){
  int f[4];
  f[1]=f[2]=f[3]=0;
  for(int i=0;i<n;i++){
    f[0]=i;
    PT v[4];
    v[0]=PT(p[i+1].x-p[i].x,p[i+1].y-p[i].y);
    for(int j=1;j<4;j++)for(v[j]=get(v[0],pi/2*j);!is_ext(f[j],v[j]);f[j]=(f[j]+1)%n);
    vector<PT>tmp;
    for(int j=0;j<4;j++)tmp.push_back(inter(p[f[j]],v[j],p[f[(j+1)%4]],v[(j+1)%4]));
    DB tmps=0;
    for(int j=0;j<4;j++)tmps+=vect(tmp[0],tmp[j],tmp[(j+1)%4]);
    tmps=fabs(tmps);
    if(ans>tmps)ans=tmps,ret=tmp;
  }
}
int main(){
  while(~scanf("%d",&n)){
    if(!n)return 0;
    for(int i=0;i<n;i++)p[i].input();
    n=convex_hull(p,n,q);
    if(n<3)ans=0;else{
      for(int i=0;i<n;i++)p[i]=q[i];
      p[n]=p[0];
      ans=1e100;
      solve();
    }
    printf("%.4f\n",ans/2.0);
    //for(int i=0;i<4;i++)ret[i].output();
  }
}
```

10.3 三维凸包

```
#define PR 1e-8
#define N 620
struct TPoint{
  double x,y,z;
  TPoint(){}
  TPoint(double _x,double _y,double _z):x(_x),y(_y),z(_z){}
  TPoint operator+(const TPoint p){return TPoint(x+p.x,y+p.y,z+p.z);}
  TPoint operator-(const TPoint p){return TPoint(x-p.x,y-p.y,z-p.z);}
  TPoint operator*(const TPoint p){//叉积
    return TPoint(y*p.z-z*p.y,z*p.x-x*p.z,x*p.y-y*p.x);
  }
  TPoint operator*(double p){return TPoint(x*p,y*p,z*p);}
  TPoint operator/(double p){return TPoint(x/p,y/p,z/p);}
  double operator^(const TPoint p){return x*p.x+y*p.y+z*p.z;}//点积
}center;
struct fac{
  int a,b,c;//凸包一个面上的三个点的编号
  bool ok;//该面是否是最终凸包中的面
```

```
};
struct T3dhull{
  int n;//初始点数
  TPoint ply[N];//初始点
  int trianglecnt;//凸包上三角形数
  fac tri[N];//凸包三角形
  int vis[N][N];//点i到点j是属于哪个面
  double dist(TPoint a){//两点长度
    return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);
  }
  double area(TPoint a,TPoint b,TPoint c){//三角形面积*2
    return dist((b-a)*(c-a));
  }
  double volume(TPoint a,TPoint b,TPoint c,TPoint d){//四面体有向体积*6
    return (b-a)*(c-a)^(d-a);
  }
  double ptoplane(TPoint &p,fac &f){//正：点在面同向
    TPoint m=ply[f.b]-ply[f.a],n=ply[f.c]-ply[f.a],t=p-ply[f.a];
    return (m*n)^t;
  }
  void deal(int p,int a,int b){
    int f=vis[a][b];
    fac add;
    if(tri[f].ok){
      if((ptoplane(ply[p],tri[f]))>PR)dfs(p,f);else{
        add.a=b,add.b=a,add.c=p,add.ok=1;
        vis[p][b]=vis[a][p]=vis[b][a]=trianglecnt;
        tri[trianglecnt++]=add;
      }
    }
  }
  void dfs(int p,int cnt){//维护凸包，如果点p 在凸包外更新凸包
    tri[cnt].ok=0;
    deal(p,tri[cnt].b,tri[cnt].a);
    deal(p,tri[cnt].c,tri[cnt].b);
    deal(p,tri[cnt].a,tri[cnt].c);
  }
  bool same(int s,int e){//判断两个面是否为同一面
    TPoint a=ply[tri[s].a],b=ply[tri[s].b],c=ply[tri[s].c];
    return fabs(volume(a,b,c,ply[tri[e].a]))<PR
    &&fabs(volume(a,b,c,ply[tri[e].b]))<PR
    &&fabs(volume(a,b,c,ply[tri[e].c]))<PR;
  }
  void construct(){//构建凸包
    int i,j;
    trianglecnt=0;
    if(n<4)return;
    bool tmp=1;
    for(i=1;i<n;i++){//前两点不共点
      if((dist(ply[0]-ply[i]))>PR){
        swap(ply[1],ply[i]);tmp=0;break;
      }
    }
    if(tmp)return;
    tmp=1;
    for(i=2;i<n;i++){//前三点不共线
      if((dist((ply[0]-ply[1])*(ply[1]-ply[i])))>PR){
```

```
        swap(ply[2],ply[i]); tmp=0; break;
      }
    }
    if(tmp)return;
    tmp=1;
    for(i=3;i<n;i++)//前四点不共面
      if(fabs((ply[0]-ply[1])*(ply[1]-ply[2])^(ply[0]-ply[i]))>PR){
        swap(ply[3],ply[i]);tmp=0;break;
      }
    if(tmp)return;
    fac add;
    for(i=0;i<4;i++){//构建初始四面体
      add.a=(i+1)%4,add.b=(i+2)%4,add.c=(i+3)%4,add.ok=1;
      if((ptoplane(ply[i],add))>0) swap(add.b,add.c);
      vis[add.a][add.b]=vis[add.b][add.c]=vis[add.c][add.a]=trianglecnt;
      tri[trianglecnt++]=add;
    }
    for(i=4;i<n;i++){//构建更新凸包
      for(j=0;j<trianglecnt;j++)
        if(tri[j].ok&&(ptoplane(ply[i],tri[j]))>PR){
          dfs(i,j);break;
        }
    }
    int cnt=trianglecnt;
    trianglecnt=0;
    for(i=0;i<cnt;i++)
      if(tri[i].ok)
        tri[trianglecnt++]=tri[i];
  }
  double area(){//表面积
    double ret=0;
    for(int i=0;i<trianglecnt;i++)ret+=area(ply[tri[i].a],ply[tri[i].b],ply[tri[i].c]);
    return ret/2.0;
  }
  double volume(){//体积
    TPoint p(0,0,0);
    double ret=0;
    for(int i=0;i<trianglecnt;i++)
      ret+=volume(p,ply[tri[i].a],ply[tri[i].b],ply[tri[i].c]);
    return fabs(ret/6);
  }
  int facetri(){return trianglecnt;}//表面三角形数
  int facepolygon(){//表面多边形数
    int ans=0,i,j,k;
    for(i=0;i<trianglecnt;i++){
      for(j=0,k=1;j<i;j++){
        if(same(i,j)){k=0;break;}
      }
      ans+=k;
    }
    return ans;
  }
  TPoint barycenter(){//重心
    TPoint ans(0,0,0),o(0,0,0);
    double all=0;
    for(int i=0;i<trianglecnt;i++){
      double vol=volume(o,ply[tri[i].a],ply[tri[i].b],ply[tri[i].c]);
```

```
      ans=ans+(o+ply[tri[i].a]+ply[tri[i].b]+ply[tri[i].c])/4.0*vol;
      all+=vol;
    }
    return ans/all;
  }
  double ptoface(TPoint p,int i){//点到面的距离
    return fabs(volume(ply[tri[i].a],ply[tri[i].b],ply[tri[i].c],p)
                /area(ply[tri[i].a],ply[tri[i].b],ply[tri[i].c]));
  }
}a;
int main(){
  scanf("%d",&a.n);
  for(int i=0;i<a.n;i++)scanf("%lf%lf%lf",&a.ply[i].x,&a.ply[i].y,&a.ply[i].z);
  a.construct();
  center=a.barycenter();
  double tmp=1e15;
  for(int i=0;i<a.trianglecnt;i++)tmp=min(tmp,a.ptoface(center,i));
  printf("%.6f",tmp);
}
```

10.4 球缺

半径为 r，高度为 h 的球缺的体积为 $\frac{h^2(3r-h)\pi}{3}$。

10.5 2D 计算几何模板大全

```
#include<cstdio>
#include<algorithm>
#include<cmath>
using namespace std;
double eps=1e-9;
//------------------------------------------------------------
//       Fundamental
//------------------------------------------------------------
int sgn(double x) {
  if( x < -eps ) return -1;
  if( x >  eps ) return  1;
  return 0;
}
//二次方程
bool Quadratic(double A, double B, double C, double *t0, double *t1) {
    double discrim = B * B - 4.f * A * C;
    if (discrim < 0.) return false;
    double rootDiscrim = sqrt(discrim);
    double q;
    if (B < 0) q = -.5f * (B - rootDiscrim);
    else       q = -.5f * (B + rootDiscrim);
    *t0 = q / A;
    *t1 = C / q;
    if (*t0 > *t1) swap(*t0, *t1);
    return true;
}
struct vec {
  double x,y;
```

```
  vec(){x=y=0;}
  vec(double _x,double _y){x=_x,y=_y;}

  vec   operator + (vec v) {return vec(x+v.x,y+v.y);}
  vec   operator - (vec v) {return vec(x-v.x,y-v.y);}
  vec   operator * (double v) {return vec(x*v,y*v);}
  vec   operator / (double v) {return vec(x/v,y/v);}

  double operator * (vec v) {return x*v.x + y*v.y;}

  double len()     {return hypot(x,y); }
  double len_sqr() {return x*x + y*y; }

  //逆时针旋转
  vec rotate(double c) {return vec(x*cos(c)-y*sin(c),x*sin(c)+y*cos(c));}
  vec trunc (double l) {return (*this) * l / len();}
  vec rot90 () {return vec(-y,x);}
};
double cross(vec a,vec b) {return a.x*b.y - a.y*b.x;}
//计算 a,b 间的角度
double get_angle(vec a,vec b) {return fabs(atan2(fabs(cross(a,b)),a*b));}
//直线插值
vec lerp(vec a,vec b,double t) {return a * (1-t) + b * t;}

//判断点是否在线段上(包含端点)
bool point_on_segment(vec p,vec a,vec b) {
  return sgn( cross(b-a,p-a) ) == 0 && sgn( (p-a)*(p-b) ) <= 0;
}

//判断线段ab,pq间是否有交点
int has_intersection(vec a,vec b,vec p,vec q) {
  int d1 = sgn( cross(b-a,p-a) ),d2 = sgn( cross(b-a,q-a) );
  int d3 = sgn( cross(q-p,a-p) ),d4 = sgn( cross(q-p,b-p) );
  if( d1 * d2 < 0 && d3 * d4 < 0 )
    return  1; //有交点，且交点不在端点
  if( ( d1 == 0 && point_on_segment(p,a,b) )||
    ( d2 == 0 && point_on_segment(q,a,b) )||
    ( d3 == 0 && point_on_segment(a,p,q) )||
    ( d4 == 0 && point_on_segment(b,p,q) ))
    return -1; //重合或交点在端点上
  return 0;
}

//直线求交点，需保证p!=q,a!=b
int line_intersection(vec a,vec b,vec p,vec q,vec &o,double *t=0) {
  double U = cross(p-a,q-p);
  double D = cross(b-a,q-p);
  if( sgn(D) == 0 )
    return sgn(U)==0 ? 2:0;//重叠|平行
  o = a + (b-a) * (U/D);
  if(t) *t = U/D;
  return 1;
}

//点p到直线ab距离
double dist_point_to_line(vec p,vec a,vec b) {
  return fabs(cross(p-a,b-a))/(b-a).len();
```

```
}

//点p到线段ab距离
double dist_point_to_segment(vec p,vec a,vec b) {
  if( (b-a).len()>eps && sgn( (p-a)*(b-a) ) >= 0 && sgn( (p-b)*(a-b) ) >= 0 )
    return fabs(cross(p-a,b-a))/(b-a).len();
  return min( (p-a).len() , (p-b).len() );
}

//------------------------------------------------------------
//       Circle
//------------------------------------------------------------
struct circle {
  vec c;double r;
  circle(){c=vec(0,0),r=0;}
  circle(vec _c,double _r){c=_c,r=_r;}
  vec point(double a){return vec(c.x+r*cos(a),c.y+r*sin(a));}
};
const double PI=acos(-1.0);

//过点p做圆C的切线，返回切线个数。v[i]表示第i条切线的方向向量
int getTangents(vec p,circle C,vec*v){
  vec u=C.c-p;
  double dist=u.len();
  if(sgn(dist-C.r)<0)return 0;
  if(!sgn(dist-C.r)){
    v[0]=u.rot90();
    return 1;
  }
  double ang=asin(C.r/dist);
  v[0]=u.rotate(-ang);
  v[1]=u.rotate(ang);
  return 2;
}

//两圆的公切线
//返回切线的个数，-1表示有无数条公切线。
//a[i],b[i]表示第i条切线在圆A，圆B上的切点
int getTangents(circle A,circle B,vec*a,vec*b){
  int cnt=0;
  if(A.r<B.r)swap(A,B),swap(a,b);
  double d2=(A.c.x-B.c.x)*(A.c.x-B.c.x)+(A.c.y-B.c.y)*(A.c.y-B.c.y);
  double rdiff=A.r-B.r;
  double rsum=A.r+B.r;
  if(sgn(d2-rdiff*rdiff)<0)return 0;//内含
  double base=atan2(B.c.y-A.c.y,B.c.x-A.c.x);
  if(!sgn(d2)&&!sgn(A.r-B.r))return -1;//无限多条切线
  if(!sgn(d2-rdiff*rdiff)){//内切一条切线
    a[cnt]=A.point(base);
    b[cnt]=B.point(base);
    cnt++;
    return 1;
  }
  //有外共切线
  double ang=acos((A.r-B.r)/sqrt(d2));
  a[cnt]=A.point(base+ang);b[cnt]=B.point(base+ang);cnt++;
  a[cnt]=A.point(base-ang);b[cnt]=B.point(base-ang);cnt++;
```

```
  if(!sgn(d2-rsum*rsum)){//一条公切线
    a[cnt]=A.point(base);
    b[cnt]=B.point(PI+base);
    cnt++;
  }else if(sgn(d2-rsum*rsum)>0){//两条公切线
    double ang=acos((A.r+B.r)/sqrt(d2));
    a[cnt]=A.point(base+ang);b[cnt]=B.point(PI+base+ang);cnt++;
    a[cnt]=A.point(base-ang);b[cnt]=B.point(PI+base-ang);cnt++;
  }
  return cnt;
}

//圆直线交点，交点是lerp(a,b,*t0)和lerp(a,b,*t1)
bool circle_line_intersection(circle c,vec a,vec b,double *t0,double *t1) {
  vec d = b - a;
  double A = d * d;
  double B = d * (a-c.c) * 2.0;
  double C = (a-c.c).len_sqr() - c.r * c.r;
  return Quadratic(A,B,C,t0,t1);
}

//圆圆相交
bool circle_circle_intersection(circle a,circle b,vec &p1,vec &p2) {
  double d = (a.c-b.c).len();
  if( d > a.r + b.r || d < fabs(a.r-b.r) ) return false;//相离|内含
  double l = ( (a.c-b.c).len_sqr() + a.r*a.r - b.r*b.r ) / (2*d);
  double h = sqrt(a.r*a.r-l*l);
  vec vl = (b.c-a.c).trunc(l),vh = vl.rot90().trunc(h);
  p1 = a.c + vl + vh;
  p2 = a.c + vl - vh;
  return true;
}

//圆和三角形abo交的面积，o是圆心
double circle_triangle_intersection_area(circle c,vec a,vec b) {
  if( sgn(cross(a-c.c,b-c.c)) == 0 ) return 0;
  vec q[5];
  int len = 0;
  double t0,t1;
  q[len++] = a;
  if( circle_line_intersection(c,a,b,&t0,&t1) ) {
    if( 0 <= t0 && t0 <= 1 ) q[len++] = lerp(a,b,t0);
    if( 0 <= t1 && t1 <= 1 ) q[len++] = lerp(a,b,t1);
  }
  q[len++] = b;
  double s = 0;
  for(int i=1;i<len;++i) {
    vec z = (q[i-1] + q[i])/2;
    if( (z-c.c).len_sqr() <= c.r*c.r )
      s += fabs( cross(q[i-1]-c.c,q[i]-c.c) ) / 2;
    else
      s += c.r*c.r*get_angle(q[i-1]-c.c,q[i]-c.c) / 2;
  }
  return s;
}

//------------------------------------------------------------
```

```
//       Polygon
//------------------------------------------------------------------
//多边形与圆交的面积
double circle_polygon_intersection_area(circle c,vec *v,int n) {
  double s = 0;
  for(int i=0;i<n;++i) {
    int j = (i+1) % n;
    s += circle_triangle_intersection_area(c,v[i],v[j])
       * sgn( cross(v[i]-c.c,v[j]-c.c) );
  }
  return fabs(s);
}

//切割多边形
//顶点按逆时针给，保留有向直线a->b左侧的部分
int polygon_cut(vec *v,int n,vec a,vec &b,vec *o) {
  int len = 0;
  for(int i=0;i<n;++i) {
    if( cross(v[i]-a,b-a) <= 0 ) o[len++] = v[i];
    vec c;double t;
    if( line_intersection(v[i],v[(i+1)%n],a,b,c,&t) && t > 0 && t < 1 )
      o[len++] = c;
  }
  return len;
}

//判点在是否在多边形内或边上
bool point_in_polygon(vec*a,int n,vec p){
  int ret=0;
  for(int i=0;i<n;i++){
    vec u=a[i],v=a[(i+1)%n];
    if(point_on_segment(p,u,v))return 1;
    if(sgn(u.y-v.y)<=0)swap(u,v);
    if(sgn(p.y-u.y)>0||sgn(p.y-v.y)<=0)continue;
    ret+=sgn(cross(v-p,u-p))>0;
  }
  return ret&1;
}

//凸包，不可有重复点
bool cmpXY(vec a,vec b) {
  if( sgn(a.x-b.x) ) return a.x < b.x;
  return a.y < b.y;
}
int convex_hull(vec* v,int n,vec *z) {
  sort(v,v+n,cmpXY);
  int m = 0;
  for(int i=0;i<n;++i) {
    while( m > 1 && cross(z[m-1]-z[m-2],v[i]-z[m-2]) <= 0 ) --m;
    z[m++] = v[i];
  }
  int k = m;
  for(int i=n-2;i>=0;--i) {
    while( m > k && cross(z[m-1]-z[m-2],v[i]-z[m-2]) <= 0 ) --m;
    z[m++] = v[i];
  }
  if( n > 1 ) --m;
```

```
  return m;
}

//返回多边形的重心，面积为 0 时需要特别处理
point MassCenter(){
  point ans=point(0,0);
  if(sgn(area())==0)while(1);
  a[n]=a[0];
  for(int i=0;i<n;i++)ans=ans+(a[i]+a[i+1])*det(a[i+1],a[i]);
  return ans/area()/6.0;
}

//————————————————————————————————————————————
//       Misc
//————————————————————————————————————————————
//绕轴旋转矩阵,使用列向量,matrix::I()是单位阵
//注意：对应法线的变换矩阵是Inverse(Transpose(R))
//verified HDU 5388
matrix rotate(double x,double y,double z,double d) {
    double len = sqrt( x*x + y*y + z*z );
    x/=len;y/=len;z/=len;
    matrix K;
    K.v[0][1] = -z;K.v[0][2] =  y;
    K.v[1][0] =  z;K.v[1][2] = -x;
    K.v[2][0] = -y;K.v[2][1] =  x;
    return matrix::I() + K * sin(d) + K * K * (1 - cos(d));
}

//三角形面积，海伦公式，a,b,c为三边长
double get_triangle_area(double a,double b,double c) {
  double s = (a+b+c) / 2;
  return sqrt( s*(s-a)*(s-b)*(s-c) );
}
```

10.6 曼哈顿凸包

先输出周长再输出面积。

```
#include<cstdio>
#include<algorithm>
using namespace std;
int n,i,r,t,q[100010],A,B,C,D;long long ans;struct P{int x,y;}a[100010];
bool cmp(P a,P b){return a.x==b.x?a.y<b.y:a.x<b.x;}
int main(){
  A=C=~0U>>1,B=D=-A;
  for(scanf("%d",&n),i=1;i<=n;i++){
    scanf("%d%d",&a[i].x,&a[i].y);
    A=min(A,a[i].x);
    B=max(B,a[i].x);
    C=min(C,a[i].y);
    D=max(D,a[i].y);
  }
  sort(a+1,a+n+1,cmp);
  for(i=1;i<=n;i++)if(!t||a[i].y>r)r=a[i].y,q[++t]=i;
  for(i=q[r=t]+1;i<=n;q[++t]=i++)while(t>r&&a[i].y>=a[q[t]].y)t--;
```

```
  for(i=2;i<=t;i++)ans+=1LL*min(a[q[i-1]].y,a[q[i]].y)*(a[q[i]].x-a[q[i-1]].x);
  for(t=0,i=1;i<=n;i++)if(!t||a[i].y<r)r=a[i].y,q[++t]=i;
  for(i=q[r=t]+1,i=q[t]+1;i<=n;q[++t]=i++)while(t>r&&a[i].y<=a[q[t]].y)t--;
  for(i=2;i<=t;i++)ans-=1LL*max(a[q[i-1]].y,a[q[i]].y)*(a[q[i]].x-a[q[i-1]].x);
  printf("%lld\n%lld",2LL*B+2LL*D-2LL*A-2LL*C,ans);
}
```

10.7 圆的面积并

圆并算法，时间复杂度 $O(n^2 \log n)$。

```
//Area[i]表示覆盖次数大于等于i的面积
struct P{
  double x,y;
  P(){}
  P(double _x,double _y){x=_x,y=_y;}
  P operator+(const P&b)const{return P(x+b.x,y+b.y);}
  P operator-(const P&b)const{return P(x-b.x,y-b.y);}
  P operator*(double b)const{return P(x*b,y*b);}
  P operator/(double b)const{return P(x/b,y/b);}
  double det(const P&b)const{return x*b.y-y*b.x;}
  P rot90()const{return P(-y,x);}
  P unit(){return *this/abs();}
  double abs(){return hypot(x,y);}
};
struct Circle{
  P o;double r;
  bool contain(const Circle&v,const int&c)const{
    return sgn(r-(o-v.o).abs()-v.r)>c;
  }
  bool disjuct(const Circle&v,const int&c)const{//0严格，-1不严格
    return sgn((o-v.o).abs()-r-v.r)>c;
  }
};
//求圆与圆的交点，包含相切，假设无重圆
bool isCC(Circle a,Circle b,P&p1,P&p2){
  if(a.contain(b,0)||b.contain(a,0)||a.disjuct(b,0))return 0;
  double s1=(a.o-b.o).abs();
  double s2=(a.r*a.r-b.r*b.r)/s1;
  double aa=(s1+s2)/2,bb=(s1-s2)/2;
  P mm=(b.o-a.o)*(aa/(aa+bb))+a.o;
  double h=sqrt(max(0.0,a.r*a.r-aa*aa));
  P vv=(b.o-a.o).unit().rot90()*h;
  p1=mm+vv,p2=mm-vv;
  return 1;
}
struct EV{
  P p;double ang;int add;
  EV(){}
  EV(const P&_p,double _ang,int _add){p=_p,ang=_ang,add=_add;}
  bool operator<(const EV&a)const{return ang<a.ang;}
}eve[N*2];
int E,cnt,C,i,j;Circle c[N];
bool g[N][N],overlap[N][N];
double Area[N];
```

```
int cX[N],cY[N],cR[N];
bool contain(int i,int j){
  return (sgn(c[i].r-c[j].r)>0||
          sgn(c[i].r-c[j].r)==0&&i<j)&&c[i].contain(c[j],-1);
}
int main(){
  scanf("%d",&C);
  for(i=0;i<C;i++){
    scanf("%d%d%d",&cX[i],&cY[i],&cR[i]);
    c[i].o=P(cX[i],cY[i]);
    c[i].r=cR[i];
  }
  for(i=0;i<=C;i++)Area[i]=0;
  for(i=0;i<C;i++)for(j=0;j<C;j++)overlap[i][j]=contain(i,j);
  for(i=0;i<C;i++)for(j=0;j<C;j++)
    g[i][j]=!(overlap[i][j]||overlap[j][i]||c[i].disjuct(c[j],-1));
  for(i=0;i<C;i++){
    E=0;cnt=1;
    for(j=0;j<C;j++)if(j!=i&&overlap[j][i])cnt++;
    for(j=0;j<C;j++)if(i!=j&&g[i][j]){
      P aa,bb;
      isCC(c[i],c[j],aa,bb);
      double A=atan2(aa.y-c[i].o.y,aa.x-c[i].o.x);
      double B=atan2(bb.y-c[i].o.y,bb.x-c[i].o.x);
      eve[E++]=EV(bb,B,1);
      eve[E++]=EV(aa,A,-1);
      if(B>A)cnt++;
    }
    if(E==0)Area[cnt]+=PI*c[i].r*c[i].r;
    else{
      sort(eve,eve+E);
      eve[E]=eve[0];
      for(j=0;j<E;j++){
        cnt+=eve[j].add;
        Area[cnt]+=eve[j].p.det(eve[j+1].p)*0.5;
        double theta=eve[j+1].ang-eve[j].ang;
        if(theta<0)theta+=PI*2;
        Area[cnt]+=theta*c[i].r*c[i].r*0.5-sin(theta)*c[i].r*c[i].r*0.5;
      }
    }
  }
  for(i=1;i<=C;i++)printf("%d %.3f\n",i,Area[i]-Area[i+1]);
}
```

10.8 平面图

给定一张平面图，不保证连通，以及若干个点，进行平面图求域以及点定位，时间复杂度 $O(n\log n)$。

cnt 表示封闭区域个数，无限域编号为 0，$from[i]$ 表示第 i 条边所属区域，$id[i]$ 表示第 i 个询问点所属区域。

```
#include<cstdio>
#include<cmath>
#include<set>
```

```
#include<map>
#include<algorithm>
using namespace std;
const double eps=1e-8;
const int N=20010,M=50010;
int n,m,q,cnt,i,x,y;
map<int,int>T[20010];
int sgn(double x){
  if(fabs(x)<eps)return 0;
  return x>0?1:-1;
}
struct P{
  double x,y;
  P(){}
  P(double _x,double _y){x=_x,y=_y;}
  double operator*(const P&b){return x*b.y-y*b.x;}
}a[N],b[N];
struct E{
  int x,y;double o;
  E(){}
  E(int _x,int _y){x=_x,y=_y,o=atan2(a[y].x-a[x].x,a[y].y-a[x].y);}
}e[M];
bool del[M],ex[M];int from[M],id[N];
struct EV{
  double x;int y,t;
  EV(){}
  EV(double _x,int _y,int _t){x=_x,y=_y,t=_t;}
}ev[M<<1];
bool cmpEV(const EV&a,const EV&b){
  if(sgn(a.x-b.x))return a.x<b.x;
  return a.t<b.t;
}
namespace GetArea{
struct cmp{bool operator()(int a,int b){return e[a].o<e[b].o;}};
set<int,cmp>g[N];set<int,cmp>::iterator k;int i,j,q[M],t;
void work(){
  for(i=0;i<m+m;i++)if(!del[i]&&!ex[i]){
    for(q[t=1]=j=i;;q[++t]=j=*k){
      k=g[e[j].y].find(j^1);k++;
      if(k==g[e[j].y].end())k=g[e[j].y].begin();
      if(*k==i)break;
    }
    double s=0;
    for(j=1;j<=t;j++)s+=a[e[q[j]].x]*a[e[q[j]].y],del[q[j]]=1;
    if(sgn(s)<=0)continue;
    for(cnt++,j=1;j<=t;j++)from[q[j]]=cnt;
  }
}
}
namespace ScanLine{
struct cmp{
  bool operator()(int A,int B){
    if(e[A].x==e[B].x)return e[A].o>e[B].o;
    double x=min(a[e[A].x].x,a[e[B].x].x),
           yA=(a[e[A].x].y-a[e[A].y].y)*(x-a[e[A].y].x)/
              (a[e[A].x].x-a[e[A].y].x)+a[e[A].y].y,
           yB=(a[e[B].x].y-a[e[B].y].y)*(x-a[e[B].y].x)/
```

```
              (a[e[B].x].x-a[e[B].y].x)+a[e[B].y].y;
    return yA>yB;
  }
};
set<int,cmp>T;
int cnt,i,j,k,g[M],v[M],nxt[M],ed,vis[N],t,tmp[N];
bool cmpC(int x,int y){return a[x].x<a[y].x;}
void add(int x,int y){v[++ed]=y;nxt[ed]=g[x];g[x]=ed;}
void dfs(int x){
  vis[x]=1;
  if(a[x].y>a[t].y)t=x;
  for(int i=g[x];i;i=nxt[i])if(!vis[v[i]])dfs(v[i]);
}
double cal(int A,double x){
  return(a[e[A].x].y-a[e[A].y].y)*(x-a[e[A].y].x)/
        (a[e[A].x].x-a[e[A].y].x)+a[e[A].y].y;
}
void connect(){
  for(i=0;i<m+m;i++)add(e[i].x,e[i].y);
  for(i=1;i<=n;i++)if(!vis[i])dfs(t=i),ev[cnt++]=EV(a[t].x,t,2);
  for(i=0;i<m+m;i++)if(sgn(a[e[i].x].x-a[e[i].y].x)>0){
    ev[cnt++]=EV(a[e[i].y].x,i,1);
    ev[cnt++]=EV(a[e[i].x].x,i,0);
  }
  sort(ev,ev+cnt,cmpEV);
  a[n+1]=P(10010,10010);
  a[n+2]=P(-10010,10010);
  e[m+m]=E(n+1,n+2);
  T.insert(m+m);
  e[m+m+1]=E(n+2,n+1);
  n+=2,m++;
  for(ed=0,i=1;i<=n;i++)g[i]=0;
  for(i=0;i<cnt;i++){
    if(ev[i].t==0)T.erase(ev[i].y);
    if(ev[i].t==1)T.insert(ev[i].y);
    if(ev[i].t==2){
      a[n+1]=P(ev[i].x,a[ev[i].y].y+eps);
      a[n+2]=P(ev[i].x-1,a[ev[i].y].y+eps);
      e[m+m]=E(n+1,n+2);
      T.insert(m+m);
      set<int,cmp>::iterator j=T.find(m+m);
      j--,add(*j,ev[i].y);
      T.erase(m+m);
    }
  }
  int newm=m+m;
  for(i=0;i<m+m;i++){
    for(cnt=0,j=g[i];j;j=nxt[j]){
      if(!sgn(a[v[j]].x-a[e[i].x].x)){
        e[newm++]=E(v[j],e[i].x);
        e[newm++]=E(e[i].x,v[j]);
        continue;
      }
      if(!sgn(a[v[j]].x-a[e[i].y].x)){
        e[newm++]=E(v[j],e[i].y);
        e[newm++]=E(e[i].y,v[j]);
        continue;
```

```
      }
      tmp[++cnt]=v[j];
    }
    if(!cnt)continue;
    ex[i]=ex[i^1]=1;
    sort(tmp+1,tmp+cnt+1,cmpC);
    for(k=e[i].y,j=1;j<=cnt;k=n,j++){
      a[++n]=P(a[tmp[j]].x,cal(i,a[tmp[j]].x));
      e[newm++]=E(k,n);
      e[newm++]=E(n,k);
      e[newm++]=E(tmp[j],n);
      e[newm++]=E(n,tmp[j]);
    }
    e[newm++]=E(n,e[i].x);
    e[newm++]=E(e[i].x,n);
  }
  m=newm/2;
}
void location(){
  for(i=cnt=0;i<m+m;i++)if(!ex[i]&&sgn(a[e[i].x].x-a[e[i].y].x)>0){
    ev[cnt++]=EV(a[e[i].y].x,i,1);
    ev[cnt++]=EV(a[e[i].x].x,i,0);
  }
  for(i=0;i<q;i++)ev[cnt++]=EV(b[i].x,i,2);
  sort(ev,ev+cnt,cmpEV);
  T.clear();
  for(i=0;i<cnt;i++){
    if(ev[i].t==0)T.erase(ev[i].y);
    if(ev[i].t==1)T.insert(ev[i].y);
    if(ev[i].t==2){
      a[n+1]=P(ev[i].x,b[ev[i].y].y);
      a[n+2]=P(ev[i].x-1,b[ev[i].y].y);
      e[m+m]=E(n+1,n+2);
      T.insert(m+m);
      set<int,cmp>::iterator j=T.find(m+m);
      if(j!=T.begin())j--,id[ev[i].y]=from[*j];
      T.erase(m+m);
    }
  }
}
}
int getid(){
  int x,y;
  scanf("%d%d",&x,&y);
  if(T[x+10000][y])return T[x+10000][y];
  T[x+10000][y]=++n;
  a[n]=P(x,y);
  return n;
}
int main(){
  scanf("%d%d",&q,&m);
  for(i=0;i<q;i++)scanf("%lf%lf",&b[i].x,&b[i].y);
  for(i=0;i<m;i++){
    x=getid();
    y=getid();
    e[i<<1]=E(x,y);
    e[i<<1|1]=E(y,x);
```

```
  }
  ScanLine::connect();//通过加辅助线使得图连通
  for(i=0;i<m+m;i++)if(!ex[i])GetArea::g[e[i].x].insert(i);
  GetArea::work();//求出所有域
  ScanLine::location();//利用扫描线进行点定位
}
```

10.8.1 直线分割

平面的 $n(n \le 500)$ 条直线将平面分割成了若干区域，给出 $m(m \le 100000)$ 个点，求每个点所在区域的面积。为了防止出现面积无穷大的情况，有额外的四条直线框定了平面区域的大小，分别是 $x = L, y = L, x = -L, y = -L$。其中 L 是给定的正实数，所有的点都在这个框定的区域内。

```
#include<cstdio>
#include<cmath>
#include<set>
#include<map>
#include<algorithm>
using namespace std;
const double eps=1e-8;
const int N=130000,M=800010,ML=600;
int nl,n,m,q,cnt,i,j,x,y,tmp;double S[M],lim;
inline int sgn(double x){
  if(fabs(x)<eps)return 0;
  return x>0?1:-1;
}
struct P{
  double x,y;
  P(){}
  P(double _x,double _y){x=_x,y=_y;}
  double operator*(const P&b){return x*b.y-y*b.x;}
  P operator*(double v){return P(x*v,y*v);}
  P operator-(const P&b){return P(x-b.x,y-b.y);}
  P operator+(const P&b){return P(x+b.x,y+b.y);}
}a[N],b[N],pool[ML];
inline bool cmpP(const P&a,const P&b){return !sgn(a.x-b.x)?a.y<b.y:a.x<b.x;}
struct Line{
  double a,b,c;
  P A,B;
  Line(){}
  Line(double _a,double _b,double _c){a=_a,b=_b,c=_c;}
  void cal(){
    if(sgn(b)){
      A=P(10000,(-a*10000-c)/b);
      B=P(-10000,(a*10000-c)/b);
    }else{
      A=P(-c/a,10000);
      B=P(-c/a,-10000);
    }
  }
}line[ML];
bool lap[ML];
struct E{
```

```
  int x,y;double o;
  E(){}
  E(int _x,int _y){x=_x,y=_y,o=atan2(a[y].x-a[x].x,a[y].y-a[x].y);}
}e[M];
bool del[M];int from[M],id[N];
struct EV{
  double x;int y,t;
  EV(){}
  EV(double _x,int _y,int _t){x=_x,y=_y,t=_t;}
}ev[M<<1];
inline bool cmpEV(const EV&a,const EV&b){
  if(sgn(a.x-b.x))return a.x<b.x;
  return a.t<b.t;
}
namespace GetArea{
struct cmp{bool operator()(int a,int b){return e[a].o<e[b].o;}};
set<int,cmp>g[N];set<int,cmp>::iterator k;int i,j,q[M],t;
void work(){
  for(i=0;i<m+m;i++)if(!del[i]){
    for(q[t=1]=j=i;;q[++t]=j=*k){
      k=g[e[j].y].find(j^1);k++;
      if(k==g[e[j].y].end())k=g[e[j].y].begin();
      if(*k==i)break;
    }
    double s=0;
    for(j=1;j<=t;j++)s+=a[e[q[j]].x]*a[e[q[j]].y],del[q[j]]=1;
    if(sgn(s)<=0)continue;
    for(cnt++,j=1;j<=t;j++)from[q[j]]=cnt;
    S[cnt]=s;
  }
}
}
namespace ScanLine{
struct cmp{
  bool operator()(int A,int B){
    if(e[A].x==e[B].x)return e[A].o>e[B].o;
    double x=min(a[e[A].x].x,a[e[B].x].x),
           yA=(a[e[A].x].y-a[e[A].y].y)*(x-a[e[A].y].x)/
              (a[e[A].x].x-a[e[A].y].x)+a[e[A].y].y,
           yB=(a[e[B].x].y-a[e[B].y].y)*(x-a[e[B].y].x)/
              (a[e[B].x].x-a[e[B].y].x)+a[e[B].y].y;
    return yA>yB;
  }
};
set<int,cmp>T;
int cnt,i;
void location(){
  for(i=cnt=0;i<m+m;i++)if(sgn(a[e[i].x].x-a[e[i].y].x)>0){
    ev[cnt++]=EV(a[e[i].y].x,i,1);
    ev[cnt++]=EV(a[e[i].x].x,i,0);
  }
  for(i=0;i<q;i++)ev[cnt++]=EV(b[i].x,i,2);
  sort(ev,ev+cnt,cmpEV);
  T.clear();
  for(i=0;i<cnt;i++){
    if(ev[i].t==0)T.erase(ev[i].y);
    if(ev[i].t==1)T.insert(ev[i].y);
```

```
    if(ev[i].t==2){
      a[n+1]=P(ev[i].x,b[ev[i].y].y);
      a[n+2]=P(ev[i].x-1,b[ev[i].y].y);
      e[m+m]=E(n+1,n+2);
      T.insert(m+m);
      set<int,cmp>::iterator j=T.find(m+m);
      if(j!=T.begin())j--,id[ev[i].y]=from[*j];
      T.erase(m+m);
    }
  }
}
}
inline int getid(const P&x){
  int l=1,r=n,mid;
  while(l<=r){
    mid=(l+r)>>1;
    if(!sgn(x.x-a[mid].x)&&!sgn(x.y-a[mid].y))return mid;
    if(sgn(x.x-a[mid].x)>0||!sgn(x.x-a[mid].x)&&sgn(x.y-a[mid].y)>0)l=mid+1;else r=mid-1;
  }
  while(1);
}
inline void newedge(int x,int y){
  if(x==y)return;
  e[m<<1]=E(x,y);
  e[m<<1|1]=E(y,x);
  m++;
}
inline double cross(const P&a,const P&b){return a.x*b.y-a.y*b.x;}
inline void work(P a,P b,P p,P q){
  double U=cross(p-a,q-p),D=cross(b-a,q-p);
  if(!sgn(D))return;
  P o=a+(b-a)*(U/D);
  if(sgn(o.x+lim)<0||sgn(o.x-lim)>0||sgn(o.y+lim)<0||sgn(o.y-lim)>0)return;
  ::a[++n]=o;
}
inline void work2(P a,P b,P p,P q){
  double U=cross(p-a,q-p),D=cross(b-a,q-p);
  if(!sgn(D))return;
  P o=a+(b-a)*(U/D);
  if(sgn(o.x+lim)<0||sgn(o.x-lim)>0||sgn(o.y+lim)<0||sgn(o.y-lim)>0)return;
  pool[++tmp]=o;
}
int main(){
  scanf("%d%d%lf",&nl,&q,&lim);
  for(i=0;i<nl;i++)scanf("%lf%lf%lf",&line[i].a,&line[i].b,&line[i].c);
  line[nl++]=Line(1,0,-lim);
  line[nl++]=Line(1,0,lim);
  line[nl++]=Line(0,1,-lim);
  line[nl++]=Line(0,1,lim);
  for(i=0;i<q;i++)scanf("%lf%lf",&b[i].x,&b[i].y);
  for(i=0;i<nl;i++)for(j=0;j<i;j++)
    if(!sgn(line[i].a-line[j].a)&&
       !sgn(line[i].b-line[j].b)&&!sgn(line[i].c-line[j].c))lap[i]=1;
  for(i=0;i<nl;i++)line[i].cal();
  for(i=0;i<nl;i++)if(!lap[i])for(j=0;j<i;j++)
    if(!lap[j])work(line[i].A,line[i].B,line[j].A,line[j].B);
  sort(a+1,a+n+1,cmpP);
```

```
  for(i=1,j=0;i<=n;i++)if(i==1||sgn(a[i].x-a[i-1].x)||sgn(a[i].y-a[i-1].y))a[++j]=a[i];
  n=j;
  for(i=0;i<nl;i++)if(!lap[i]){
    tmp=0;
    for(j=0;j<nl;j++)if(i!=j&&!lap[j])work2(line[i].A,line[i].B,line[j].A,line[j].B);
    sort(pool+1,pool+tmp+1,cmpP);
    for(j=1;j<tmp;j++)newedge(getid(pool[j]),getid(pool[j+1]));
  }
  for(i=0;i<m+m;i++)GetArea::g[e[i].x].insert(i);
  GetArea::work();
  ScanLine::location();
  for(i=0;i<q;i++)printf("%.2f\n",S[id[i]]/2);
}
```

10.8.2 线段分割

平面上 $n(n \leq 100)$ 条不重合线段，再给定不在线段上的起点和终点，问最少穿过几次线段。

```
#include<cstdio>
#include<cmath>
#include<set>
#include<algorithm>
using namespace std;
const double eps=1e-12,inf=110000;
const int N=20010,M=100010;
int n,m,w,q,cnt,cur,i,j,x,idx[N],g[N],v[M],nxt[M],ed,d[N],que[N],h,t,S,T;
inline int sgn(double x){
  if(fabs(x)<eps)return 0;
  return x>0?1:-1;
}
struct P{
  double x,y;
  P(){}
  P(double _x,double _y){x=_x,y=_y;}
  P operator+(P b){return P(x+b.x,y+b.y);}
  P operator-(P b){return P(x-b.x,y-b.y);}
  P operator*(double b){return P(x*b,y*b);}
  P operator/(double b){return P(x/b,y/b);}
  double operator*(P b){return x*b.x+y*b.y;}
}a[N],b[N],st[N],en[N],pool[N];
inline bool cmpP(const P&a,const P&b){return !sgn(a.x-b.x)?a.y<b.y:a.x<b.x;}
inline double cross(P a,P b){return a.x*b.y-a.y*b.x;}
inline bool point_on_segment(P p,P a,P b){
  return sgn(cross(b-a,p-a))==0&&sgn((p-a)*(p-b))<=0;
}
inline int has_intersection(P a,P b,P p,P q){
  int d1=sgn(cross(b-a,p-a)),d2=sgn(cross(b-a,q-a));
  int d3=sgn(cross(q-p,a-p)),d4=sgn(cross(q-p,b-p));
  return d1*d2<0&&d3*d4<0;
}
inline P line_intersection(P a,P b,P p,P q){
  double U=cross(p-a,q-p),D=cross(b-a,q-p);
  return a+(b-a)*(U/D);
}
```

```
struct E{
  int x,y;double o;
  E(){}
  E(int _x,int _y){x=_x,y=_y,o=atan2(a[y].x-a[x].x,a[y].y-a[x].y);}
}e[M];
bool del[M],ex[M];int from[M],id[N];
struct EV{
  double x;int y,t;
  EV(){}
  EV(double _x,int _y,int _t){x=_x,y=_y,t=_t;}
}ev[M<<1];
inline bool cmpEV(const EV&a,const EV&b){
  if(sgn(a.x-b.x))return a.x<b.x;
  return a.t<b.t;
}
namespace GetArea{
struct cmp{bool operator()(int a,int b){return e[a].o<e[b].o;}};
set<int,cmp>g[N];set<int,cmp>::iterator k;int i,j,q[M],t;
void work(){
  for(i=0;i<m+m;i++)if(!del[i]&&!ex[i]){
    for(q[t=1]=j=i;;q[++t]=j=*k){
      k=g[e[j].y].find(j^1);k++;
      if(k==g[e[j].y].end())k=g[e[j].y].begin();
      if(*k==i)break;
    }
    double s=0;
    for(j=1;j<=t;j++)s+=cross(a[e[q[j]].x],a[e[q[j]].y]),del[q[j]]=1;
    if(sgn(s)<0)continue;
    for(cnt++,j=1;j<=t;j++)from[q[j]]=cnt;
  }
}
}
namespace ScanLine{
struct cmp{
  bool operator()(int A,int B){
    if(e[A].x==e[B].x)return e[A].o>e[B].o;
    double x=min(a[e[A].x].x,a[e[B].x].x),
           yA=(a[e[A].x].y-a[e[A].y].y)*(x-a[e[A].y].x)/
              (a[e[A].x].x-a[e[A].y].x)+a[e[A].y].y,
           yB=(a[e[B].x].y-a[e[B].y].y)*(x-a[e[B].y].x)/
              (a[e[B].x].x-a[e[B].y].x)+a[e[B].y].y;
    return yA>yB;
  }
};
set<int,cmp>T;
int cnt,i,j,k,g[M],v[M],nxt[M],ed,vis[N],t,tmp[N];
inline bool cmpC(int x,int y){return a[x].x<a[y].x;}
inline void add(int x,int y){v[++ed]=y;nxt[ed]=g[x];g[x]=ed;}
void dfs(int x){
  vis[x]=1;
  if(a[x].y>a[t].y)t=x;
  for(int i=g[x];i;i=nxt[i])if(!vis[v[i]])dfs(v[i]);
}
inline double cal(int A,double x){
  return(a[e[A].x].y-a[e[A].y].y)*(x-a[e[A].y].x)/
        (a[e[A].x].x-a[e[A].y].x)+a[e[A].y].y;
}
```

```
void connect(){
  for(i=0;i<m+m;i++)add(e[i].x,e[i].y);
  for(i=1;i<=n;i++)if(!vis[i])dfs(t=i),ev[cnt++]=EV(a[t].x,t,2);
  for(i=0;i<m+m;i++)if(sgn(a[e[i].x].x-a[e[i].y].x)>0){
    ev[cnt++]=EV(a[e[i].y].x,i,1);
    ev[cnt++]=EV(a[e[i].x].x,i,0);
  }
  sort(ev,ev+cnt,cmpEV);
  a[n+1]=P(inf,inf);
  a[n+2]=P(-inf,inf);
  e[m+m]=E(n+1,n+2);
  T.insert(m+m);
  e[m+m+1]=E(n+2,n+1);
  n+=2,m++;
  for(ed=0,i=1;i<=n;i++)g[i]=0;
  for(i=0;i<cnt;i++){
    if(ev[i].t==0)T.erase(ev[i].y);
    if(ev[i].t==1)T.insert(ev[i].y);
    if(ev[i].t==2){
      a[n+1]=P(ev[i].x,a[ev[i].y].y+eps);
      a[n+2]=P(ev[i].x-1,a[ev[i].y].y+eps);
      e[m+m]=E(n+1,n+2);
      T.insert(m+m);
      set<int,cmp>::iterator j=T.find(m+m);
      j--,add(*j,ev[i].y);
      T.erase(m+m);
    }
  }
  int newm=m+m;
  for(i=0;i<m+m;i++){
    for(cnt=0,j=g[i];j;j=nxt[j]){
      if(!sgn(a[v[j]].x-a[e[i].x].x)){
        e[newm++]=E(v[j],e[i].x);
        e[newm++]=E(e[i].x,v[j]);
        continue;
      }
      if(!sgn(a[v[j]].x-a[e[i].y].x)){
        e[newm++]=E(v[j],e[i].y);
        e[newm++]=E(e[i].y,v[j]);
        continue;
      }
      tmp[++cnt]=v[j];
    }
    if(!cnt)continue;
    ex[i]=ex[i^1]=1;
    sort(tmp+1,tmp+cnt+1,cmpC);
    for(k=e[i].y,j=1;j<=cnt;k=n,j++){
      a[++n]=P(a[tmp[j]].x,cal(i,a[tmp[j]].x));
      e[newm++]=E(k,n);
      e[newm++]=E(n,k);
      e[newm++]=E(tmp[j],n);
      e[newm++]=E(n,tmp[j]);
    }
    e[newm++]=E(n,e[i].x);
    e[newm++]=E(e[i].x,n);
  }
  m=newm/2;
```

```
}
void location(){
  for(i=cnt=0;i<m+m;i++)if(!ex[i]&&sgn(a[e[i].x].x-a[e[i].y].x)>0){
    ev[cnt++]=EV(a[e[i].y].x,i,1);
    ev[cnt++]=EV(a[e[i].x].x,i,0);
  }
  for(i=0;i<q;i++)ev[cnt++]=EV(b[i].x,i,2);
  sort(ev,ev+cnt,cmpEV);
  T.clear();
  for(i=0;i<cnt;i++){
    if(ev[i].t==0)T.erase(ev[i].y);
    if(ev[i].t==1)T.insert(ev[i].y);
    if(ev[i].t==2){
      a[n+1]=P(ev[i].x,b[ev[i].y].y);
      a[n+2]=P(ev[i].x-1,b[ev[i].y].y);
      e[m+m]=E(n+1,n+2);
      T.insert(m+m);
      set<int,cmp>::iterator j=T.find(m+m);
      if(j!=T.begin())j--,id[ev[i].y]=from[*j];
      T.erase(m+m);
    }
  }
}
}
inline int getid(P o){
  int l=1,r=n,mid;
  while(l<=r){
    mid=(l+r)>>1;
    if(!sgn(o.x-a[mid].x)&&!sgn(o.y-a[mid].y))return mid;
    if(sgn(o.x-a[mid].x)>0||!sgn(o.x-a[mid].x)&&sgn(o.y-a[mid].y)>0)l=mid+1;else r=mid-1;
  }
}
inline void cal0(P a,P b,P c,P d){
  if(!has_intersection(a,b,c,d))return;
  ::a[++n]=line_intersection(a,b,c,d);
}
inline void cal1(P a,P b,P c,P d){
  if(point_on_segment(c,a,b)){pool[++cur]=c;return;}
  if(point_on_segment(d,a,b)){pool[++cur]=d;return;}
  if(!has_intersection(a,b,c,d))return;
  pool[++cur]=line_intersection(a,b,c,d);
}
inline void add(int x,int y){v[++ed]=y;nxt[ed]=g[x];g[x]=ed;}
int main(){
  scanf("%d",&w);
  for(q=2;i<q;i++)scanf("%lf%lf",&b[i].x,&b[i].y);
  for(i=0;i<w;i++){
    scanf("%lf%lf%lf%lf",&st[i].x,&st[i].y,&en[i].x,&en[i].y);
    a[++n]=st[i];
    a[++n]=en[i];
  }
  for(i=0;i<w;i++)for(j=0;j<i;j++)cal0(st[i],en[i],st[j],en[j]);
  sort(a+1,a+n+1,cmpP);
  int _=0;
  for(i=1;i<=n;i++)if(i==1||sgn(a[i].x-a[i-1].x)||sgn(a[i].y-a[i-1].y))a[++_]=a[i];
  n=_;
  for(i=0;i<w;i++){
```

```
    pool[1]=st[i];
    pool[cur=2]=en[i];
    for(j=0;j<w;j++)if(i!=j)cal1(st[i],en[i],st[j],en[j]);
    sort(pool+1,pool+cur+1,cmpP);
    for(j=1;j<=cur;j++)idx[j]=getid(pool[j]);
    for(j=1;j<cur;j++)if(idx[j]!=idx[j+1]){
      e[m<<1]=E(idx[j],idx[j+1]);
      e[m<<1|1]=E(idx[j+1],idx[j]);
      m++;
    }
  }
  ScanLine::connect();
  for(i=0;i<m+m;i++)if(!ex[i])GetArea::g[e[i].x].insert(i);
  GetArea::work();
  ScanLine::location();
  for(i=0;i<m+m;i++)if(!ex[i])add(from[i],from[i^1]);
  d[que[h=t=1]=id[0]]=1;
  while(h<=t)for(i=g[x=que[h++]];i;i=nxt[i])if(!d[v[i]])d[que[++t]=v[i]]=d[x]+1;
  return printf("%d",d[id[1]]-1),0;
}
```

10.9 Descartes' Theorem

平面上 4 个圆相切于不同的 6 个点时，四个圆的曲率 $k=\frac{1}{r}$（若该圆与其他圆均外切，则曲率取正，否则取负）满足

$$(k_1+k_2+k_3+k_4)^2=2(k_1^2+k_2^2+k_3^2+k_4^2)$$

即

$$k_4=k_1+k_2+k_3\pm 2\sqrt{k_1k_2+k_2k_3+k_3k_1}$$

如果将圆心用复数 $z=x+yi$ 表示，那么有

$$(k_1z_1+k_2z_2+k_3z_3+k_4z_4)^2=2(k_1^2z_1^2+k_2^2z_2^2+k_3^2z_3^2+k_4^2z_4^2)$$

即

$$z_4=\frac{z_1k_1+z_2k_2+z_3k_3\pm 2\sqrt{k_1k_2z_1z_2+k_2k_3z_2z_3+k_1k_3z_1z_3}}{k_4}$$

在 n 维空间中，比如 2 维的圆和直线 $(k=0)$，3 维的球和平面 $(k=0)$，有

$$(\sum_{i=1}^{n+2}k_i)^2=n\sum_{i=1}^{n+2}k_i^2$$

10.10 动态凸包

平面上最开始只包含 3 个点，之后依次加入 n 个点，每加入一个点后输出凸包面积。

```
typedef pair<int,int>P;
const int inf=2000000000;
set<P>T0,T1;set<P>::iterator j,k,pre,nxt,q[100010];
```

```
int n,m,x,y;ll ans,now;
ll cross(P A,P B,P C){
  return 1LL*(B.second-A.second)*(C.first-B.first)-
         1LL*(C.second-B.second)*(B.first-A.first);
}
ll mul(P A,P B){return 1LL*A.first*B.second-1LL*A.second*B.first;}
void add0(int x,int y){
  if(T0.find(P(x,y))!=T0.end())return;
  T0.insert(P(x,y));
  pre=nxt=j=T0.find(P(x,y));
  pre--,nxt++;
  if(pre->second<inf&&nxt->second<inf){
    if(cross(*pre,*j,*nxt)<=0){T0.erase(j);return;}
    ans-=mul(*pre,*nxt);
  }
  if(pre->second<inf)ans+=mul(*pre,*j);
  if(nxt->second<inf)ans+=mul(*j,*nxt);
  m=0;
  while(pre->second<inf){
    k=pre;k--;
    if(k->second==inf||cross(*k,*pre,*j)>0)break;
    ans+=mul(*k,*j)-mul(*k,*pre)-mul(*pre,*j);
    q[++m]=pre;
    pre=k;
  }
  while(nxt->second<inf){
    k=nxt;k++;
    if(k->second==inf||cross(*j,*nxt,*k)>0)break;
    ans+=mul(*j,*k)-mul(*j,*nxt)-mul(*nxt,*k);
    q[++m]=nxt;
    nxt=k;
  }
  while(m)T0.erase(q[m--]);
}
void add1(int x,int y){
  if(T1.find(P(x,y))!=T1.end())return;
  T1.insert(P(x,y));
  pre=nxt=j=T1.find(P(x,y));
  pre--,nxt++;
  if(pre->second<inf&&nxt->second<inf){
    if(cross(*pre,*j,*nxt)>=0){T1.erase(j);return;}
    ans+=mul(*pre,*nxt);
  }
  if(pre->second<inf)ans-=mul(*pre,*j);
  if(nxt->second<inf)ans-=mul(*j,*nxt);
  m=0;
  while(pre->second<inf){
    k=pre;k--;
    if(k->second==inf||cross(*k,*pre,*j)<0)break;
    ans-=mul(*k,*j)-mul(*k,*pre)-mul(*pre,*j);
    q[++m]=pre;
    pre=k;
  }
  while(nxt->second<inf){
    k=nxt;k++;
    if(k->second==inf||cross(*j,*nxt,*k)<0)break;
    ans-=mul(*j,*k)-mul(*j,*nxt)-mul(*nxt,*k);
```

```
    q[++m]=nxt;
    nxt=k;
  }
  while(m)T1.erase(q[m--]);
}
int main(){
  T0.insert(P(-inf,inf)),T0.insert(P(inf,inf));
  T1.insert(P(-inf,inf)),T1.insert(P(inf,inf));
  for(n=0;n<3;n++)scanf("%d%d",&x,&y),add0(x,y),add1(x,y);
  for(scanf("%d",&n);n--;printf("%lld\n",-now)){
    scanf("%d%d",&x,&y),add0(x,y),add1(x,y);
    now=ans;
    j=T0.begin();j++;
    k=T1.begin();k++;
    now+=mul(*k,*j);
    j=T0.find(P(inf,inf));j--;
    k=T1.find(P(inf,inf));k--;
    now+=mul(*j,*k);
  }
}
```

10.11 四面体内切球公式

设四个点是 a, b, c, d, 那么内切球的球心可以表示为：

$$\frac{\left\|(b-a)\times(c-a)\right\|\times d+\left\|(b-a)\times(d-a)\right\|\times c+\left\|(c-a)\times(d-a)\right\|\times b+\left\|(c-b)\times(d-b)\right\|\times a}{\left\|(b-a)\times(c-a)\right\|+\left\|(b-a)\times(d-a)\right\|+\left\|(c-a)\times(d-a)\right\|+\left\|(c-b)\times(d-b)\right\|}$$

半径满足：

$$R=\frac{\left\|\begin{vmatrix} a_x & a_y & a_z & 1\\ b_x & b_y & b_z & 1\\ c_x & c_y & c_z & 1\\ d_x & d_y & d_z & 1\end{vmatrix}\right\|}{\left\|(b-a)\times(c-a)\right\|+\left\|(b-a)\times(d-a)\right\|+\left\|(c-a)\times(d-a)\right\|+\left\|(c-b)\times(d-b)\right\|}$$

10.12 长方体表面两点距离

```
int r;
void turn(int i,int j,int x,int y,int z,int x0,int y0,int L,int W,int H){
  if(z==0){
    int R=x*x+y*y;
    if(R<r)r=R;
  }else{
    if(i>=0&&i<2)turn(i+1,j,x0+L+z,y,x0+L-x,x0+L,y0,H,W,L);
    if(j>=0&&j<2)turn(i,j+1,x,y0+W+z,y0+W-y,x0,y0+W,L,H,W);
    if(i<=0&&i>-2)turn(i-1,j,x0-z,y,x-x0,x0-H,y0,H,W,L);
    if(j<=0&&j>-2)turn(i,j-1,x,y0-z,y-y0,x0,y0-H,L,H,W);
  }
}
int cal(int x1,int y1,int z1,int x2,int y2,int z2,int L,int W,int H){
```

```
  if(z1!=0&&z1!=H)if(y1==0||y1==W)
    swap(y1,z1),swap(y2,z2),swap(W,H);
  else
    swap(x1,z1),swap(x2,z2),swap(L,H);
  if(z1==H)z1=0,z2=H-z2;
  r=~0U>>1;
  turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
  return r;
}
```

10.13 3D 计算几何基本操作

```
typedef double flt;
const flt eps = 1e-8;
int sgn(flt x) {return x < -eps ? -1 : x > eps;}
struct Point {
  flt x, y, z;
  Point() {}
  Point(flt x, flt y, flt z): x(x), y(y), z(z) {}
  bool operator < (const Point &rhs) const {
    return x < rhs.x || (x == rhs.x && y < rhs.y)
           || (x == rhs.x && y == rhs.y && z < rhs.z);
  }
  Point operator + (const Point &rhs) const {
    return Point(x + rhs.x, y + rhs.y, z + rhs.z);
  }
  Point operator - (const Point &rhs) const {
    return Point(x - rhs.x, y - rhs.y, z - rhs.z);
  }
  Point operator * (const flt k) const {
    return Point(x * k, y * k, z * k);
  }
  Point operator / (const flt k) const {
    return Point(x / k, y / k, z / k);
  }
  flt dot(const Point &rhs) const {
    return x * rhs.x + y * rhs.y + z * rhs.z;
  }
  Point det(const Point &rhs) const {
    return Point(y * rhs.z - z * rhs.y,
                 z * rhs.x - x * rhs.z,
                 x * rhs.y - y * rhs.x);
  }
  flt sqrlen() const {
    return x * x + y * y + z * z;
  }
  flt len() const {
    return sqrt(sqrlen());
  }
  void read() {
    scanf("%lf%lf%lf", &x, &y, &z);
  }
  void print() {
    printf("%.10f %.10f %.10f\n", x, y, z);
  }
```

```
} A, B, C, D;
//判断点C是否在直线AB上
bool dotsInline(const Point &A, const Point &B, const Point &C) {
  return sgn((B - A).det(C - A).len()) == 0;
}
//判断直线AB和CD是否平行
bool is_parallel(const Point &A, const Point &B, const Point &C, const Point &D) {
  return sgn((B - A).det(D - C).len()) == 0;
}
//判断直线AB和CD的交点，如果直线异面，返回的是两直线的垂线在AB上的垂足
//需要保证 AB 和 CD 不平行
Point intersect(const Point &A, const Point &B, const Point &C, const Point &D) {
  Point p0 = (C - A).det(D - C), p1 = (B - A).det(D - C);
  return A + (B - A) * (p0.dot(p1) / p1.sqrlen());
}
```

10.14 经纬度求球面最短距离

```
//lati为纬度，longi为经度，R为半径
double dis(double lati1,double longi1,double lati2,double longi2,double R){
  double pi=acos(-1.0);lati1*=pi/180,longi1*=pi/180,lati2*=pi/180,longi2*=pi/180;
  double x1=cos(lati1)*sin(longi1),y1=cos(lati1)*cos(longi1),z1=sin(lati1);
  double x2=cos(lati2)*sin(longi2),y2=cos(lati2)*cos(longi2),z2=sin(lati2);
  double theta=acos(x1*x2+y1*y2+z1*z2);return R*theta;
}
```

10.15 三维旋转操作

```
//a点绕 Ob 向量逆时针旋转angle弧度
//angle,sin(),cos()先求出来，减少精度问题
P e1,e2,e3;P Rotate(P a,P b,double angle){
  b.std();//单位化，注意 b 不能为(0,0,0)
  e3=b;double lens=a*e3;//dot(a,e3)
  e1=a-e3*lens;
  if(e1.len()>1e-8)e1.std();else return a;
  e2=e1/e3;//det(e1,e3)
  double x1=a*e2,y1=a*e1,
         x=x1*cos(angle)-y1*sin(angle);
  double y=x1*sin(angle)+y1*cos(angle);
  return e3*lens+e1*y+e2*x;
}
```

10.16 DP 凸包

平面上 n 个白点，m 个黑点，从白点选择若干个形成面积最大的凸包，使得边上和内部不存在黑点。

注意到凸包上一圈的边的极角序递增，故将所有边极角排序。

枚举起点 S，问题等价于按递增序走回 S，且中间任何一个三角形 SAB 都不包含黑点。

判断三角形是否包含黑点只需要对每条边预处理其左侧的点集，然后 bitset 加速。

设 $f[i]$ 表示从 S 到 i 的最大叉积和，按递增序枚举每条边松弛即可。时间复杂度 $O(\frac{n^4}{64})$。

```
const int N=110,inf=~0U>>1;
int n,m,cnt,i,j,k,f[N],ans;bitset<N>w[N][N];
struct P{
  int x,y;double o;
  P(){}
  P(int _x,int _y,double _o){x=_x,y=_y,o=_o;}
  P operator-(const P&b){return P(x-b.x,y-b.y,0);}
  void read(){scanf("%d%d",&x,&y);}
}a[N],b[N],e[N*N];
bool cmp(const P&a,const P&b){return a.o<b.o;}
int cross(const P&a,const P&b){return a.x*b.y-a.y*b.x;}
void up(int&a,int b){a<b?(a=b):0;}
bool have(int x,int y,int z){return (w[x][y]&w[y][z]&w[z][x]).any();}
int main(){
  scanf("%d%d",&n,&m);
  for(i=1;i<=n;i++)a[i].read();
  for(i=1;i<=m;i++)b[i].read();
  for(i=1;i<=n;i++)for(j=1;j<=n;j++)if(i!=j)
    e[++cnt]=P(i,j,atan2(a[j].y-a[i].y,a[j].x-a[i].x));
  sort(e+1,e+cnt+1,cmp);
  for(i=1;i<=n;i++)for(j=1;j<=n;j++)if(i!=j)
    for(k=1;k<=m;k++)if(cross(b[k]-a[i],a[j]-a[i])<=0)w[i][j][k]=1;
  for(i=1;i<=n;i++){
    for(j=1;j<=n;j++)f[j]=-inf;
    f[i]=0;
    for(j=1;j<=cnt;j++)if(f[e[j].x]>-inf&&!have(i,e[j].x,e[j].y))
      up(f[e[j].y],f[e[j].x]+cross(a[e[j].x],a[e[j].y]));
    up(ans,f[i]);
  }
  printf("%d.%d0",ans/2,ans%2*5);
}
```

10.17 凸包切线

给定一个 n 个点的凸包，m 次独立询问假如新增一个点会导致凸包面积新增多少。

如果凸包大小 ≤ 2，那么特判即可。

否则凸包大小至少为 3，对于每个询问，首先在凸包上按极角二分出一条边，看看是否在凸包内或者边上。

如果不在，那么往左往右二分出两条切线的位置，然后用叉积前缀和回答询问即可。

时间复杂度 $O((n+m)\log n)$。

```
const int N=100010;
int n,m,ce,i,j,x,y,z,ca,cb,cnt;ll f[N<<1];
struct P{
  int x,y;
  P(){}
  P(int _x,int _y){x=_x,y=_y;}
  P operator-(const P&b){return P(x-b.x,y-b.y);}
  void operator-=(const P&b){*this=*this-b;}
  bool operator==(const P&b){return x==b.x&&y==b.y;}
  bool operator!=(const P&b){return x!=b.x||y!=b.y;}
```

```
}a[N],b,c[N<<1],O;
bool cmp(const P&a,const P&b){return a.x==b.x?a.y<b.y:a.x<b.x;}
ll cross(const P&a,const P&b){return 1LL*a.x*b.y-1LL*a.y*b.x;}
int convexhull(P*p,int n,P*q){
  int i,k,m;
  for(i=m=0;i<n;q[m++]=p[i++])while(m>1&&cross(q[m-1]-q[m-2],p[i]-q[m-2])<=0)m--;
  k=m;
  for(i=n-2;i>=0;q[m++]=p[i--])while(m>k&&cross(q[m-1]-q[m-2],p[i]-q[m-2])<=0)m--;
  return --m;
}
bool point_on_segment(P p,P a,P b){
  return !cross(b-a,p-a)&&1LL*(p.x-a.x)*(p.x-b.x)+1LL*(p.y-a.y)*(p.y-b.y)<=0;
}
int askl(int l,int r,P p){
  int t=l++,mid;
  while(l<=r){
    mid=(l+r)>>1;
    if(cross(c[mid]-p,c[(mid-1+n)%n]-c[mid])<=0)l=(t=mid)+1;else r=mid-1;
  }
  return t;
}
int askr(int l,int r,P p){
  int t=r--,mid;
  while(l<=r){
    mid=(l+r)>>1;
    if(cross(c[mid]-p,c[(mid+1)%n]-c[mid])>=0)r=(t=mid)-1;else l=mid+1;
  }
  return t;
}
ll solve(P p){
  if(n<2)return 0;
  if(n==2)return abs(f[0]+cross(c[1],p)+cross(p,c[0]));
  if(point_on_segment(p,c[0],c[n-1]))return f[n-1];
  int o=0;
  if(p.x>0){
    int l=1,r=n-1,mid;
    while(l<=r)if(cross(c[mid=(l+r)>>1],p)>=0)l=(o=mid)+1;else r=mid-1;
  }else if(p.y>0)o=n-1;
  if(p.x>=0&&cross(p-c[o],c[o+1]-p)<0)return f[n-1];
  if(p.x>=0&&point_on_segment(p,c[o],c[o+1]))return f[n-1];
  int l,r;
  if(p.x>0)l=askl(0,o,p),r=askr(o,n,p);else l=askl(m,n,p),r=askr(0,m,p);
  if(l>r)r+=n;
  //切点为l,r，作用区域为 l..r
  return f[n-1]+cross(c[l],p)+cross(p,c[r])-f[r-1]+f[l-1];
}
int main(){
  read(ca),read(cb);
  for(i=1;i<=ca;i++)read(a[i].x),read(a[i].y);
  cnt=ca;
  sort(a+1,a+ca+1,cmp);
  for(ca=0,i=1;i<=cnt;i++)if(i==1||a[i]!=a[i-1])a[++ca]=a[i];
  n=convexhull(a+1,ca,c);
  for(O=c[0],i=0;i<n;i++)c[i]-=O;
  for(i=0;i<n;i++)if(c[i].x>=c[m].x)m=i;
  for(i=0;i<n;i++)c[i+n]=c[i];
  for(i=0;i<n+n;i++){
```

```
    f[i]=cross(c[i],c[i+1]);
    if(i)f[i]+=f[i-1];
  }
  while(cb--){
    read(b.x),read(b.y);
    ll tmp=solve(b-O);
    printf("%lld.%lld\n",tmp/2,tmp%2*5);
  }
}
```

10.18 欧几里得最小生成树

三角剖分求出欧几里得最小生成树，注意代码比三角剖分多了一些边。incir 函数不能 inline。

```
const int N=100010;
const double eps=1e-9;
int sgn(double x){
  if(x>eps)return 1;
  if(x<-eps)return -1;
  return 0;
}
struct P{
  double x,y;
  P(){}
  P(double _x,double _y){x=_x,y=_y;}
  bool operator<(const P&a)const{return sgn(x-a.x)<0||sgn(x-a.x)==0&&sgn(y-a.y)<0;}
  P operator-(const P&a)const{return P(x-a.x,y-a.y);}
  double operator&(const P&a)const{return x*a.y-y*a.x;}
  double operator|(const P&a)const{return x*a.x+y*a.y;}
}p[N];
double check(const P&a,const P&b,const P&c){return (b-a)&(c-a);}
double dis2(const P&a){return a.x*a.x+a.y*a.y;}
bool cross(int a,int b,int c,int d){
  return sgn(check(p[a],p[c],p[d])*check(p[b],p[c],p[d]))<0&&
         sgn(check(p[c],p[a],p[b])*check(p[d],p[a],p[b]))<0;
}
struct P3{
  double x,y,z;
  P3(){}
  P3(double _x,double _y,double _z){x=_x,y=_y,z=_z;}
  bool operator<(const P3&a)const{return sgn(x-a.x)<0||sgn(x-a.x)==0&&sgn(y-a.y)<0;}
  P3 operator-(const P3&a)const{return P3(x-a.x,y-a.y,z-a.z);}
  double operator|(const P3&a)const{return x*a.x+y*a.y+z*a.z;}
  P3 operator&(const P3&a)const{return P3(y*a.z-z*a.y,z*a.x-x*a.z,x*a.y-y*a.x);}
}ori[N];
P3 check(const P3&a,const P3&b,const P3&c){return (b-a)&(c-a);}
P3 gp3(const P&a){return P3(a.x,a.y,a.x*a.x+a.y*a.y);}
int cal(double x){
  int y=x;
  for(int i=y-2;i<=y+2;i++)if(!sgn(x-i))return i;
}
bool incir(int a,int b,int c,int d){
  P3 aa=gp3(p[a]),bb=gp3(p[b]),cc=gp3(p[c]),dd=gp3(p[d]);
  if(sgn(check(p[a],p[b],p[c]))<0)swap(bb,cc);
```

```
  return sgn(check(aa,bb,cc)|(dd-aa))<0;
}
int n,m,i,j,et=1,la[N],tot,l,r,q[N<<2];
struct E{
  int to,l,r;
  E(){}
  E(int _to,int _l,int _r=0){to=_to,l=_l,r=_r;}
}e[N<<5];
void add(int x,int y){
  e[++et]=E(y,la[x]),e[la[x]].r=et,la[x]=et;
  e[++et]=E(x,la[y]),e[la[y]].r=et,la[y]=et;
}
void del(int x){
  e[e[x].r].l=e[x].l;
  e[e[x].l].r=e[x].r;
  la[e[x^1].to]==x?la[e[x^1].to]=e[x].l:1;
}
void delaunay(int l,int r){
  if(r-l<=2){
    for(int i=l;i<r;i++)for(int j=i+1;j<=r;j++)add(i,j);
    return;
  }
  int i,j,mid=(l+r)>>1,ld=0,rd=0,id,op;
  delaunay(l,mid),delaunay(mid+1,r);
  for(tot=0,i=l;i<=r;q[++tot]=i++)
    while(tot>1&&sgn(check(p[q[tot-1]],p[q[tot]],p[i]))<0)tot--;
  for(i=1;i<tot&&!ld;i++)if(q[i]<=mid&&mid<q[i+1])ld=q[i],rd=q[i+1];
  for(;add(ld,rd),1;){
    id=op=0;
    for(i=la[ld];i;i=e[i].l)
      if(sgn(check(p[ld],p[rd],p[e[i].to]))>0)
        if(!id||incir(ld,rd,id,e[i].to))op=-1,id=e[i].to;
    for(i=la[rd];i;i=e[i].l)
      if(sgn(check(p[rd],p[ld],p[e[i].to]))<0)
        if(!id||incir(ld,rd,id,e[i].to))op=1,id=e[i].to;
    if(op==0)break;
    if(op==-1){
      for(i=la[ld];i;i=e[i].l)
      if(cross(rd,id,ld,e[i].to))del(i),del(i^1),i=e[i].r;
      ld=id;
    }else{
      for(i=la[rd];i;i=e[i].l)
      if(cross(ld,id,rd,e[i].to))del(i),del(i^1),i=e[i].r;
      rd=id;
    }
  }
}
int main(){
  scanf("%d",&n);
  for(i=1;i<=n;i++){
    int x,y;
    scanf("%d%d",&x,&y);
    p[i]=P(x,y);
    ori[i]=P3(x,y,i);
  }
  sort(p+1,p+n+1);
  sort(ori+1,ori+n+1);
```

```
  delaunay(1,n);
  for(i=1;i<=n;i++)for(j=la[i];j;j=e[j].l){
    int x=cal(ori[i].z),y=cal(ori[e[j].to].z);
    DS::newedge(x,y,dis2(p[i]-p[e[j].to]));
  }
  DS::work();
}
```

10.19 直线与凸包交点

若直线与凸包相交，输出分成两部分里较小部分的面积，否则输出 0。

```
const int N=100010;
const double eps=1e-9,pi=acos(-1.0);
int n,n2,m,i;double f[N],sum[N<<1];
inline int sgn(double x){
  if(x>eps)return 1;
  if(x<-eps)return -1;
  return 0;
}
struct P{
  double x,y;
  P(){x=y=0;}
  P(double _x,double _y){x=_x,y=_y;}
  P operator+(const P&p)const{return P(x+p.x,y+p.y);}
  P operator-(const P&p)const{return P(x-p.x,y-p.y);}
  double operator*(const P&p)const{return x*p.y-y*p.x;}
  P operator*(double p)const{return P(x*p,y*p);}
  P operator/(double p)const{return P(x/p,y/p);}
  bool operator<(const P&p)const{
    if(!sgn(y-p.y))return x<p.x;
    return y<p.y;
  }
  double angle(){return atan2(y,x);}
  void read(){scanf("%lf%lf",&x,&y);}
}a[N],c[N<<1],u,v;
inline double vect(const P&p,const P&p1,const P&p2){
  return (p1.x-p.x)*(p2.y-p.y)-(p1.y-p.y)*(p2.x-p.x);
}
int convexhull(P*p,int n,P*q){
  int i,k,m;
  sort(p,p+n);
  if(n==1){
    q[0]=p[0];
    return 1;
  }
  m=0;
  for(i=0;i<n;q[m++]=p[i++])while(m>1&&vect(q[m-2],q[m-1],p[i])<eps)m--;
  k=m;
  for(i=n-2;i>=0;q[m++]=p[i--])while(m>k&&vect(q[m-2],q[m-1],p[i])<eps)m--;
  return --m;
}
inline int mid(double s){
  if(s<0)s+=pi*2;
  int L=1,R=n,m,r;
```

```
  while(L<=R){
    m=(L+R)/2;
    if(f[m]>s+eps)R=m-1;
    else if(f[m+1]>s-eps)return m;
    else L=m+1;
  }
  return n;
}
inline P cal(int L,int R,double&s1,double&s2){
  if(R<L)R+=n;
  if((u-v)*(c[L]-v)<-eps)swap(u,v);
  if((u-v)*(c[R]-v)>eps)swap(u,v);
  int s=L,e=R,m;
  while(s+1<e){
    m=(s+e)>>1;
    if((u-v)*(c[m]-v)>-eps)s=m;else e=m;
  }
  s1+=sum[s]-sum[L],s2+=sum[R]-sum[e];
  double t1=(c[e]-c[s])*(u-c[s]),t2=(c[s]-c[e])*(v-c[e]);
  P r=(u*t2+v*t1)/(t1+t2);
  s1+=c[s]*r,s2+=r*c[e];
  return r;
}
inline double solve(){
  u.read(),v.read();
  if(n<3)return 0;
  int k1=mid((u-v).angle()),k2=mid((v-u).angle());
  double t1=(u-v)*(c[k1]-v),t2=(u-v)*(c[k2]-v);
  if(t1*t2>-eps)return 0;
  double s1=0,s2=0;
  P a=cal(k1,k2,s1,s2),b=cal(k2,k1,s2,s1);
  s1+=a*b,s2+=b*a;
  return min(s1,s2)/2.0;
}
int main(){
  scanf(”%d”,&n);
  for(i=1;i<=n;i++)a[i].read();
  sort(a+1,a+n+1);
  for(m=0,i=1;i<=n;i++)if(i==1||sgn(a[i].x-a[i-1].x)||sgn(a[i].y-a[i-1].y))a[++m]=a[i];
  n=convexhull(a+1,m,c);
  for(i=0;i<n;i++)c[i+n]=c[i];
  c[n2=n+n]=c[0];
  for(i=1;i<=n2;i++)sum[i]=sum[i-1]+c[i-1]*c[i];
  for(i=1;i<=n+1;i++){
    f[i]=(c[i]-c[i-1]).angle();
    if(f[i]<f[i-1]-eps)f[i]+=pi*2;
  }
  scanf(”%d”,&m);
  while(m--)printf(”%.10f\n”,solve());
}
```

10.20 平面最近点对

```
#include<cstdio>
#include<algorithm>
```

```
using namespace std;
typedef long long ll;
const int N=100010;
const ll inf=1LL<<60;
int n,m,i,ansx,ansy;ll ans;
struct P{int x,y;}p[N],q[N];
inline bool cmpx(const P&a,const P&b){return a.x<b.x;}
inline bool cmpy(int a,int b){return p[a].y<p[b].y;}
inline ll dis(const P&a,const P&b){
  return 1LL*(a.x-b.x)*(a.x-b.x)+1LL*(a.y-b.y)*(a.y-b.y);
}
ll solve(P*p,int l,int r){
  ll d=inf;
  if(l+2>=r){
    for(int i=l;i<=r;i++)for(int j=i+1;j<=r;j++){
      ll t=dis(p[i],p[j]);
      if(t<d){
        d=t;
        if(t<ans)ans=t,ansx=i,ansy=j;
      }
    }
    return d;
  }
  int mid=(l+r)>>1,i,j,m=0;
  d=min(solve(p,l,mid),solve(p,mid+1,r));
  for(i=l;i<=r;i++)if(p[i].x>=p[mid].x-d&&p[i].x<=p[mid].x+d)a[m++]=i;
  sort(a,a+m,cmpy);
  for(i=0;i<m;i++)for(j=i+1;j<m;j++){
    if(p[a[j]].y-p[a[i]].y>=d)break;
    ll t=dis(p[a[i]],p[a[j]]);
    if(t<d){
      d=t;
      if(t<ans)ans=t,ansx=a[i],ansy=a[j];
    }
  }
  return d;
}
int main(){
  scanf("%d",&n);
  for(i=0;i<n;i++)scanf("%d%d",&p[i].x,&p[i].y);
  sort(p,p+n,cmpx);
  ans=inf;
  solve(p,0,n-1);
}
```

10.21 三维投影平面

```
point3 point3::unit()const{return*this /length();}
double mix(const point3&a,const point3&b,const point3&c){return dot(a,det(b,c));}
struct plane3{point3 a,b,c;plane3(){}
plane3(point3 a,point3 b,point3 c):a(a),b(b),c(c){}};
double vlen(point3 p){return p.length();}
point3 pvec(plane3 s){return det((s.a-s.b),(s.b-s.c));}
//平面法向量
point3 pvec(point3 s1,point3 s2,point3 s3){return det((s1-s2),(s2-s3));}
```

```
double ptoplane(point3 p,plane3 s){return fabs(dot(pvec(s),p-s.a))/vlen(pvec(s));}
//旋转点集，使法向量为 v 的平面水平
void rotate_to_horizontal(int n,point3*p,point3 v){
  if(!sgn(v.x)&&!sgn(v.y))return;
  double a=atan2(v.y, v.x),c=cos(a),s=sin(a);
  for(int i=0;i<n;i++){
    point3 t=p[i];
    p[i].x=t.x*c+t.y*s;
    p[i].y=t.y*c-t.x*s;
  }
  a=atan2(sqrt(v.x*v.x+v.y*v.y),v.z);c=cos(a),s=sin(a);
  for(int i=0;i<n;i++){
    point3 t=p[i];
    p[i].z=t.z*c+t.x*s;
    p[i].x=t.x*c-t.z*s;
  }
}
//怎么求点在底面上的投影
//先求出平面的法向量，以及点到平面的距离
//将点沿着法向量移动距离个单位（两个方向），选择移动后在平面内的点，即为投影
void doit(point3 p1,point3 p2,point3 p3,point3 a[]){
  point3 fa=pvec(p1,p2,p3),p[N];
  double DIS=0,dis[N];
  for(int i=0;i<n;i++){
    gmax(DIS,dis[i]=ptoplane(a[i],plane3(p1,p2,p3)));
    point3 pp1=a[i]+fa.unit()*dis[i],pp2=a[i]-fa.unit()*dis[i];
    if(!sgn(ptoplane(pp1,plane3(p1,p2,p3))))p[i]=pp1;else p[i]=pp2;
  }
  //找到了在平面 f 上的所有点的映射点集 p，旋转完之后就变成了水平面上的点集 p
  rotate_to_horizontal(n,p,fa);
}
```

10.22 木棍拼面积最大的多边形

```
ld cal(int S){
  int i,len=0,sum=0;
  for(i=0;i<n;i++)if(S>>i&1)sum+=a[i],b[++len]=a[i];
  if(len<=2)return -1;
  sort(b+1,b+len+1);
  if(sum<=b[len]*2)return -1;
  ld L=0.5*b[len],R=b[len]*len,ret=0;
  for(int _=60;_--;){
    ld mid=(L+R)/2;
    for(i=1;i<=len;i++)f[i]=f[i-1]+acos(1.0-0.5*b[i]*b[i]/mid/mid);
    if(f[len]>PI*2)L=mid;else R=mid;
  }
  if(fabs(f[len]-PI*2)>1e-8){
    L=0.5*b[len],R=sum*PI;
    for(int _=60;_--;){
      ld mid=(L+R)/2;
      for(i=1;i<=len;i++)f[i]=f[i-1]+acos(1.0-0.5*b[i]*b[i]/mid/mid);
      ld x=cos(f[len-1]),y=sin(f[len-1]);
      if(sqrt((x-1)*(x-1)+y*y)*mid<b[len]||f[len]>PI*2)L=mid;else R=mid;
    }
  }
```

```
  f[len]=0;
  for(i=1;i<=len;i++)ret+=sin(f[i]-f[i-1]);
  return ret*L*L/2;
}
```

11 黑科技与杂项

11.1 卡常

```
#define _FORTIFY_SOURCE 0
#pragma GCC optimize(”Ofast,no-stack-protector”)
#pragma GCC target(”sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx,avx,tune=native”)
struct FastIO{
  static const int S=1310720;
  int wpos,pos,len;char wbuf[S];
  FastIO():wpos(0){}
  inline int xchar(){
    static char buf[S];
    if(pos==len)pos=0,len=fread(buf,1,S,stdin);
    if(pos==len)return -1;
    return buf[pos++];
  }
  inline int xuint(){//-1表示空了
    int c=xchar();
    while(c<=32&&~c)c=xchar();
    if(c==-1)return -1000000000;
    int f=0;
    int a=0;
    for(;c<45;c=xchar());
    if(c==45)f=1,c=xchar();
    while(c>47)a=a*10+c-48,c=xchar();
    if(f)a=-a;
    return a;
  }
}io;
```

11.2 开栈

11.2.1 32 位 Win 下

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
extern int main2(void) __asm__ (”_main2”);
int main2(){
  char test[255<<20];
  memset(test,42,sizeof(test));
  printf(”:)\n”);
  exit(0);//注意：这里需要exit(0);来退出程序，否则会得到非零退出的错误，可能报RE的
}
int main(){
  int size=256<<20;//256MB
  char*p=(char*)malloc(size)+size;
  __asm__ __volatile__(
    ”movl  %0, %%esp\n”
    ”pushl $_exit\n”
    ”jmp _main2\n”
    :: ”r”(p));
}
```

11.2.2 64 位 Linux 下：(对 main() 中的汇编语句做修改)

```
  __asm__ __volatile__(
    "movq  %0, %%rsp\n"
    "pushq $exit\n"
    "jmp main2\n"  //注意到下需要，下要用win_main2linuxmain2
    :: "r"(p));
```

11.2.3 简化版本

一定要最后写一句 exit(0); 退出程序。

```
int size=256<<20;//256MB
char*p=(char*)malloc(size)+size;
__asm__ __volatile__("movq  %0, %%rsp\n" :: "r"(p));//64bit
```

11.3 I/O 优化

11.3.1 普通 I/O 优化

```
//适用于非负整数
template<class T>
void scan_d(T&ret){
  char c;ret=0;
  while((c=getchar())<'0'||c>'9');
  while(c>='0'&&c<='9')ret=ret*10+(c-'0'),c=getchar();
}
//适用于整数
template<class T>
bool scan_d(T&ret){
  char c;int sgn;
  if(c=getchar(),c==EOF)return 0;//EOF
  while(c!='-'&&(c<'0'||c>'9'))c=getchar();
  sgn=(c=='-')?-1:1;
  ret=(c=='-')?0:(c-'0');
  while(c=getchar(),c>='0'&&c<='9')ret=ret*10+(c-'0');
  ret*=sgn;
  return 1;
}
//适用于整数,(int,long long,float,double)
template<class T>
bool scan_d(T&ret){
  char c;int sgn;T bit=0.1;
  if(c=getchar(),c==EOF)return 0;
  while(c!='-'&&c!='.'&&(c<'0'||c>'9'))c=getchar();
  sgn=(c=='-')?-1:1;
  ret=(c=='-')?0:(c-'0');
  while(c=getchar(),c>='0'&&c<='9')ret=ret*10+(c-'0');
  if(c==' '||c=='\n'){ret*=sgn;return 1;}
  while(c=getchar(),c>='0'&&c<='9')ret+=(c-'0')*bit,bit/=10;
```

```
  ret*=sgn;
  return 1;
}
//输出外挂
void out(int x){
  if(x>9)out(x/10);
  putchar(x%10+'0');
}
```

11.3.2 文艺 I/O 优化

```
const int BUFSIZE=20<<20;//<<10KB,<<20MB
char Buf[BUFSIZE+1],*buf=Buf;
//fread(Buf,1,BUFSIZE,stdin)在读入之前写这句话;
//非负整数
template<class T>
void scan(T&a){
  for(a=0;*buf<'0'||*buf>'9';buf++);
  while(*buf>='0'&&*buf<='9'){a=a*10+(*buf-'0');buf++;}
}
//任何整数
template<class T>
void scan(T&a){
  int sgn=1;
  for(a=0;*buf<'0'||*buf>'9';buf++)if(*buf=='-')sgn=-1;
  while(*buf>='0'&&*buf<='9'){a=a*10+(*buf-'0');buf++;}
  a*=sgn;
}
//支持EOF，非负整数
const int BUFSIZE=20<<20;//<<10KB,<<20MB
char Buf[BUFSIZE+1],*buf=Buf;
size_t lastlen=0;
template<class T>
bool scan(T&a){
  for(a=0;(*buf<'0');buf++);
  if(buf-Buf>=lastlen)return 0;
  while(*buf>='0'){a=a*10+(*buf-'0');buf++;}
  return 1;
}
//lastlen=fread(Buf,1,BUFSIZE,stdin);在读入之前写这句话
//读入字符串
void scanString(char*c){
  for(;*buf=='\n'||*buf=='\t'||*buf==' ';buf++);
  while((*buf!='\n')&&(*buf!='\t')&&(*buf!=' ')){*(c++)=*(buf++);}
  *c=0;
}
```

11.3.3 二逼 I/O 优化

```
namespace IO{
const int MX=4096;
char buf[MX],t[50];
int bi=MX,bn=MX;
```

```
int read(char*s){//读入字符串
  while(bn){
    for(;bi<bn&&buf[bi]<=' ';bi++);
    if(bi<bn)break;
    bn=fread(buf,1,MX,stdin);
    bi=0;
  }
  int sn=0;
  while(bn){
    for(;bi<bn&&buf[bi]>' ';bi++)s[sn++]=buf[bi];
    if(bi<bn)break;
    bn=fread(buf,1,MX,stdin);
    bi=0;
  }
  s[sn]=0;
  return sn;
}
bool read(int&x){//读入并转化成变量，这里以int为例
  if(!read(t))return 0;
  x=atoi(t);//long long和double的读入同理，有atoll()和atof()
  return 1;
}
}
```

11.3.4 fread 判 EOF

```
struct FastIO{
  static const int S=1310720;
  int wpos,pos,len;char wbuf[S];
  FastIO():wpos(0){}
  inline int xchar(){
    static char buf[S];
    if(pos==len)pos=0,len=fread(buf,1,S,stdin);
    if(pos==len)return -1;
    return buf[pos++];
  }
  inline int xuint(){//-1 表示空了
    int c=xchar(),x=0;
    while(c<=32&&~c)c=xchar();
    if(c==-1)return -1;
    for(;'0'<=c&&c<='9';c=xchar())x=x*10+c-'0';
    return x;
  }
}io;
```

11.4 位运算及其运用

11.4.1 枚举子集

枚举 i 的非空子集 j。

```
for(j=i;j;j=(j-1)&i);
```

11.4.2 求 1 的个数

```
int __builtin_popcount(unsigned int x);
```

11.4.3 求前缀 0 的个数

```
int __builtin_clz(unsigned int x);
```

11.4.4 求后缀 0 的个数

```
int __builtin_ctz(unsigned int x);
```

11.5 石子合并

每次可以合并相邻两堆石子，代价为两堆石子的个数和，用 GarsiaWachs 算法求最小总代价。

```
#include<cstdio>
int a[50003],n,i,t,ans;
void combine(int k){
  int tmp=a[k]+a[k-1],i,j;ans+=tmp;
  for(i=k;i<t-1;i++)a[i]=a[i+1];
  for(t--,j=k-1;j>0&&a[j-1]<tmp;j--)a[j]=a[j-1];
  a[j]=tmp;
  while(j>=2&&a[j]>=a[j-2])i=t-j,combine(j-1),j=t-i;
}
int main(){
  while(1){
    scanf(”%d”,&n);
    if(!n)return 0;
    for(i=ans=0;i<n;i++)scanf(”%d”,a+i);
    for(t=i=1;i<n;i++){
      a[t++]=a[i];
      while(t>=3&&a[t-3]<=a[t-1])combine(t-2);
    }
    while(t>1)combine(t-1);
    printf(”%d\n”,ans);
  }
}
```

11.6 最小乘积生成树

把方案看成一个二维点，$x = \sum a, y = \sum b$，答案一定在下凸壳上。

找到 l, r 两个点，l 是 x 最小的，r 是 y 最小的，然后递归调用 $work(l, r)$：

1. 找到离该直线最远的点，那个点一定在下凸壳上。

2. 将边权设为 (a, b) 叉积 $(l - r)$。

3. 求出最小生成树作为点 mid。

4. 递归 $work(l,mid)$ 和 $work(mid,r)$。

```
#include<cstdio>
#include<algorithm>
#define N 210
#define M 10010
using namespace std;
typedef long long ll;
struct P{
  int x,y;P(){x=y=0;}P(int _x,int _y){x=_x,y=_y;}
  P operator-(const P&a){return P(x-a.x,y-a.y);}
}l,r;
ll cross(P a,P b){return (ll)a.x*(ll)b.y-(ll)a.y*(ll)b.x;}
struct E{int x,y,a,b,c;}a[M];
bool cmp(const E&a,const E&b){return a.c<b.c;}
int n,m,i,f[N];
ll ans=100000000000000LL;
int F(int x){return f[x]==x?x:f[x]=F(f[x]);}
P kruskal(){
  P p;int i;
  sort(a+1,a+m+1,cmp);
  for(i=1;i<=n;i++)f[i]=i;
  for(i=1;i<=m;i++)if(F(a[i].x)!=F(a[i].y))
    f[f[a[i].x]]=f[a[i].y],p.x+=a[i].a,p.y+=a[i].b;
  if((ll)p.x*(ll)p.y<ans)ans=(ll)p.x*(ll)p.y;
  return p;
}
void work(P l,P r){
  P t=l-r;
  for(int i=1;i<=m;i++)a[i].c=cross(P(a[i].a,a[i].b),t);
  P mid=kruskal();
  if(cross(mid-l,r-mid)>0)work(l,mid),work(mid,r);
}
int main(){
  scanf("%d%d",&n,&m);
  for(i=1;i<=m;i++)scanf("%d%d%d%d",&a[i].x,&a[i].y,&a[i].a,&a[i].b),a[i].x++,a[i].y++;
  for(i=1;i<=m;i++)a[i].c=a[i].a;
  l=kruskal();
  for(i=1;i<=m;i++)a[i].c=a[i].b;
  r=kruskal();
  work(l,r);
  printf("%lld",ans);
}
```

11.7 特征多项式加速线性递推

$a[n]=\sum_{i=1}^{m} c[m-i]a[n-i]$，给定 $a[0],a[1],...,a[m-1]$，求 $a[n]$。

用特征多项式 + 快速幂加速线性递推，时间复杂度 $O(m^2\log n)$。

```
const int M=2000,P=1000000007;
int n,m,i,j,x,w,b,t,a[M],c[M],v[M],u[M<<1],ans;
int main(){
  scanf("%d%d",&n,&m);
  for(i=m-1;~i;i--)scanf("%d",&c[i]),c[i]=(c[i]%P+P)%P;
  for(i=0;i<m;i++)scanf("%d",&a[i]),a[i]=(a[i]%P+P)%P;
```

```
  for(i=0;i<m;i++)v[i]=1;
  for(w=!!n,i=n;i>1;i>>=1)w<<=1;
  for(x=0;w;copy(u,u+m,v),w>>=1,x<<=1){
    fill_n(u,m<<1,0),b=!!(n&w),x|=b;
    if(x<m)u[x]=1;
    else{
      for(i=0;i<m;i++)for(j=0,t=i+b;j<m;j++,t++)u[t]=((ll)v[i]*v[j]+u[t])%P;
      for(i=(m<<1)-1;i>=m;i--)for(j=0,t=i-m;j<m;j++,t++)u[t]=((ll)c[j]*u[i]+u[t])%P;
    }
  }
  for(i=0;i<m;i++)ans=((ll)v[i]*a[i]+ans)%P;
  printf("%d",ans);
}
```

11.8 三元环的枚举

给定一张 n 个点 m 条边的无向图，在 $O(m\sqrt{m})$ 的时间内枚举所有三元环。

```
const int N=100010,M=200010,Base=(1<<21)-1;
struct edge{int v,w;edge*nxt;}epool[M],*ecur=epool,*g[N],*j,*k;
struct Edge{int x,y,w;Edge*nxt;}Epool[M],*Ecur=Epool,*G[Base+1],*l;
int n,m,i,d[N],x,y,lim,Hash;
struct Elist{int x,y,w;}e[M];
bool cmp(const Elist&a,const Elist&b){return a.x==b.x?a.y<b.y:a.x<b.x;}
int vis(int x,int y){
  for(l=G[(x<<8|y)&Base];l;l=l->nxt)if(l->x==x&&l->y==y)return l->w;
  return 0;
}
int main(){
  while(~scanf("%d%d",&n,&m)){
    while(lim*lim<m)lim++;
    for(i=1;i<=m;i++){
      scanf("%d%d",&x,&y);
      if(x<y)swap(x,y);
      e[i].x=x,e[i].y=y;
    }
    for(sort(e+1,e+m+1,cmp),i=1;i<=m;i++){
      d[x=e[i].x]++;
      ecur->v=y=e[i].y;ecur->w=i;ecur->nxt=g[x];g[x]=ecur++;
      Ecur->x=x;Ecur->y=y;Ecur->w=i;Ecur->nxt=G[Hash=(x<<8|y)&Base];G[Hash]=Ecur++;
    }
    for(i=3;i<=n;i++)for(j=g[i];j;j=j->nxt)if(d[x=j->v]<=lim){
      for(k=g[x];k;k=k->nxt)if(y=vis(i,k->v)){
        //三条边分别为e[j->w] e[k->w] e[y]
        //与x点相连的两条边分别为e[j->w] e[k->w]
        //与i点相连的两条边分别为e[j->w] e[y]
        //与k->v点相连的两条分别边为e[k->w] e[y]
      }
    }else for(k=j->nxt;k;k=k->nxt)if(y=vis(x,k->v)){
        //三条边分别为e[j->w] e[k->w] e[y]
        //与i点相连的两条边分别为e[j->w] e[k->w]
        //与x点相连的两条边分别为e[j->w] e[y]
        //与k->v点相连的两条边分别为e[k->w] e[y]
    }
    lim=0,ecur=epool,Ecur=Epool;
```

```
    for(i=1;i<=n;i++)d[i]=0,g[i]=NULL;
    for(i=1;i<=m;i++)G[(e[i].x<<8|e[i].y)&Base]=NULL;
  }
}
```

11.9 所有区间 gcd 的预处理

```
int n,i,j,a[N],l[N],v[N];
int fun(int x,int y){return __gcd(x,y);}
int main(){
  for(scanf("%d",&n),i=1;i<=n;i++)scanf("%d",&a[i]);
  for(i=1;i<=n;i++)for(v[i]=a[i],j=l[i]=i;j;j=l[j]-1){
    v[j]=fun(v[j],a[i]);
    while(l[j]>1&&fun(a[i],v[l[j]-1])==fun(a[i],v[j]))l[j]=l[l[j]-1];
    //[l[j]..j,i]区间内的值求fun均为v[j]
  }
}
```

11.10 无向图最小割

给定一张 n 个点 m 条带权边的无向图，点的编号为 0 到 $n-1$，用 Stoer-Wagner 算法求它的最小割，即最后的图至少有 2 个连通块。

```
const int N=502,inf=1000000000;
int v[N],w[N],c[N],g[N][N],S,T,now,n,m,x,y,z;
void search(){
  int i,j,k,t;
  for(i=0;i<n;i++)v[i]=w[i]=0;
  for(S=T=-1,i=0;i<n;i++){
    for(k=-inf,j=0;j<n;j++)if(!c[j]&&!v[j]&&w[j]>k)k=w[t=j];
    if(T==t)return;
    S=T,T=t,now=k,v[t]=1;
    for(j=0;j<n;j++)if(!c[j]&&!v[j])w[j]+=g[t][j];
  }
}
int stoerwagner(){
  int i,j,ans=inf;
  for(i=0;i<n;i++)c[i]=0;
  for(i=0;i<n-1;i++){
    search();
    if(now<ans)ans=now;
    if(ans==0)return 0;
    for(c[T]=1,j=0;j<n;j++)if(!c[j])g[S][j]+=g[T][j],g[j][S]+=g[j][T];
  }
  return ans;
}
int main(){
  scanf("%d%d",&n,&m);
  while(m--)scanf("%d%d%d",&x,&y,&z),g[x][y]+=z,g[y][x]+=z;
  printf("%d",stoerwagner());
}
```

11.10.1 堆优化

时间复杂度 $O(nm\log n)$。

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<queue>
using namespace std;
const int N=3010,M=100010*2,inf=0x3f3f3f3f;
int head[N],val[N],ecnt;
int to[M],nxt[M],weight[M];
bool vis[N];
int fa[N],lk[N];
int n,m;
void init(){
  memset(head+1,-1,sizeof(int)*n);
  memset(lk+1,-1,sizeof(int)*n);
  for(int i=1;i<=n;++i)fa[i]=i;
  ecnt=0;
}
void add_edge(int u,int v,int w){
  to[ecnt]=v;weight[ecnt]=w;nxt[ecnt]=head[u];head[u]=ecnt++;
  to[ecnt]=u;weight[ecnt]=w;nxt[ecnt]=head[v];head[v]=ecnt++;
}
int F(int u){return u==fa[u]?u:fa[u]=F(fa[u]);}
void merge(int u,int v){
  int p=u;
  while(~lk[p])p=lk[p];
  lk[p]=v;
  fa[v]=u;
}
int MinimumCutPhase(int cnt,int &s,int&t){
  memset(val+1,0,sizeof(int)*n);
  memset(vis+1,0,sizeof(bool)*n);
  priority_queue<pair<int,int> > que;
  t=1;
  while(--cnt){
    vis[s=t]=1;
    for(int u=s;~u;u=lk[u])for(int p=head[u];~p;p=nxt[p]){
      int v=F(to[p]);
      if(!vis[v])que.push(make_pair(val[v]+=weight[p],v));
    }
    t=0;
    while(!t){
      if(que.empty())return 0;//图不连通
      pair<int,int>pa=que.top();que.pop();
      if(val[pa.second]==pa.first)t=pa.second;
    }
  }
  return val[t];
}
int StoerWagner(){
  int res=inf;
  for(int i=n,s,t;i>1;--i){
    res=min(res,MinimumCutPhase(i,s,t));
    if(!res)break;
```

```
    merge(s,t);
  }
  return res;
}
int main(){
  while(~scanf("%d%d",&n,&m)){
    init();
    for(int i=0,u,v,w;i<m;++i){
      scanf("%d%d%d",&u,&v,&w);//1<=u,v<=n
      add_edge(u,v,w);
    }
    printf("%d\n",StoerWagner());
  }
}
```

11.11 分割回文串

输入一个长度为 n，从 0 开始的字符串 s，MinPalindromeSpilt(s) 后，$f[n][0]$ 表示该串分成偶数个回文串，至少能分成几个；$f[n][1]$ 表示该串分成奇数个回文串，至少能分成几个。若能分成 X 个，那么一定可以分成 $X+2$ 个。

```
char s[N];
int d[N][2],f[N][2];
struct P{
  int d[3];
  P(){}
  P(int a,int b,int c){d[0]=a;d[1]=b;d[2]=c;}
  int&operator[](int x){return d[x];}
}a[32],b[32],c[32];
void up(int f[][2],int x,int y){
  if(y<=0)return;
  int p=y&1;
  if(f[x][p]<0)f[x][p]=y;else f[x][p]=min(f[x][p],y);
}
void make(int f[][2],int x,int y){if(y>0)f[x][y&1]=y;}
void MinPalindromeSpilt(char*s){
  int n=strlen(s);
  memset(a,0,sizeof a);
  memset(b,0,sizeof b);
  memset(c,0,sizeof c);
  memset(d,0,sizeof d);
  memset(f,0,sizeof f);
  for(int i=0;i<=n;i++)d[i][0]=1000000000,d[i][1]=1000000001;
  for(int ca=0,j=0;j<n;j++){
    int cb=0,cc=0,r=-j-2;
    for(int u=0;u<ca;u++){
      int i=a[u][0];
      if(i>=1&&s[i-1]==s[j])a[u][0]--,b[cb++]=a[u];
    }
    for(int u=0;u<cb;u++){
      int i=b[u][0],d=b[u][1],k=b[u][2];
      if(i-r!=d){
        c[cc++]=P(i,i-r,1);
        if(k>1)c[cc++]=P(i+d,d,k-1);
```

```
      }else c[cc++]=P(i,d,k);
      r=i+(k-1)*d;
    }
    if(j>=1&&s[j-1]==s[j])c[cc++]=P(j-1,j-1-r,1),r=j-1;
    c[cc++]=P(j,j-r,1),ca=0;
    P&h=c[0];
    for(int u=1;u<cc;u++){
      P&x=c[u];
      if(x[1]==h[1])h[2]+=x[2];else a[ca++]=h,h=x;
    }
    a[ca++]=h;
    if((j+1)%2==0)f[j+1][0]=j+1,f[j+1][1]=1000000001;
    else f[j+1][0]=1000000000,f[j+1][1]=j+1;
    for(int u=0;u<ca;u++){
      int i=a[u][0],e=a[u][1],k=a[u][2];
      r=i+(k-1)*e;
      up(f,j+1,f[r][0]+1),up(f,j+1,f[r][1]+1);
      if(k>1)up(f,j+1,d[i+1-e][0]),up(f,j+1,d[i+1-e][1]);
      if(i+1-e>=0){
        if(k>1)up(d,i+1-e,f[r][0]+1),up(d,i+1-e,f[r][1]+1);
        else make(d,i+1-e,f[r][0]+1),make(d,i+1-e,f[r][1]+1);
      }
    }
  }
}
int main(){
  int T;
  for(scanf("%d",&T);T--;){
    scanf("%s",s);
    MinPalindromeSpilt(s);
    int n=strlen(s);
    printf("%d %d\n",f[n][0],f[n][1]);
  }
}
```

11.12 2-SAT 计数

$n(n \leq 50)$ 个布尔变量 $x_1, x_2, ..., x_n$，m 个限制，每个限制要求 a_i *or* b_i 为真，求方案数。输入里正数表示 x，否则表示非 x。

```
const int MAXN = 100;

vector<int> f[2*MAXN];
vector<int> g[MAXN];
int conv(int x) {
  if (x > 0) return 2 * (x - 1) + 1;
  return 2 * (-x - 1);
}

int deg[MAXN];
int forbidden[MAXN];
int value[MAXN];
int ok[MAXN][MAXN];
long long ans;
int n, m;
```

```

void set(int id, int val) {
  assert(value[id] == -1);
  value[id] = val;
  forbidden[2*id + 1 - val]++;
  for (int x : g[id]) {
    deg[x]--;
  }
  for (int x : f[id * 2 + val]) {
    forbidden[x]++;
  }
}

void unset(int id) {
  assert(value[id] != -1);
  forbidden[2*id + 1 - value[id]]--;
  for (int x : g[id]) {
    deg[x]++;
  }
  for (int x : f[id * 2 + value[id]]) {
    forbidden[x]--;
  }
  value[id] = -1;
}

long long cycle(int v, vector<int>& used) {
  vector<int> path;
  int i = v;
  int prev = -1;
  while (true) {
    //fprintf(stderr, "i = %d\n", i);
    if (used[i]) {
      assert(i == v);
      break;
    }
    path.push_back(i);
    used[i] = 1;
    int to = -1;
    for (int j : g[i]) {
      if (value[j] == -1 && j != prev) {
        to = j;
        break;
      }
    }
    if (to == -1) {
      break;
    }
    prev = i;
    i = to;
  }

  long long c0, c1;
  long long res = 0;
  for (int i = 0; i < 2; i++) {
    c0 = i ? 0 : 1;
    c1 = i ? 1 : 0;
    for (int j = 1; j < (int)path.size(); j++) {
```

```
      long long nc0, nc1;
      nc0 = nc1 = 0;
      if (ok[path[j-1]][path[j]] & 1) nc0 += c0;
      if (ok[path[j-1]][path[j]] & 2) nc0 += c1;
      if (ok[path[j-1]][path[j]] & 4) nc1 += c0;
      if (ok[path[j-1]][path[j]] & 8) nc1 += c1;
      c1 = nc1;
      c0 = nc0;
    }
    //printf("i = %d, path.size() = %d, c0 = " INT64 ", c1 = " INT64 "\n", i, (int)path.size(), c0, c1);
    if (i == 0) {
      if (ok[path[0]][path.back()] & 1) res += c0;
      if (ok[path[0]][path.back()] & 4) res += c1;
    } else {
      if (ok[path[0]][path.back()] & 2) res += c0;
      if (ok[path[0]][path.back()] & 8) res += c1;
    }
  }
  return res;
}

void go() {
  for (int i = 0; i < n; i++) {
    if (forbidden[2*i] && forbidden[2*i+1]) {
      return;
    }
  }
  int maxd = -1;
  for (int i = 0; i < n; i++) {
    if (value[i] != -1) continue;
    if (forbidden[2*i]) {
      set(i, 1);
      go();
      unset(i);
      return;
    }
    if (forbidden[2*i + 1]) {
      set(i, 0);
      go();
      unset(i);
      return;
    }
    if (maxd == -1 || deg[i] > deg[maxd]) {
      maxd = i;
    }
  }
  //printf("maxd = %d\n", maxd);
  if (maxd == -1) {
    ans += 1;
  } else if (deg[maxd] <= 2) {
    long long c = 1;
    vector<int> used(n);
    for (int i = 0; i < n; i++) {
      if (value[i] != -1 || used[i]) continue;
      //printf("! i = %d, deg[i] = %d\n", i, deg[i]);
      if (deg[i] == 1) {
```

```
        c *= cycle(i, used);
      }
    }
    for (int i = 0; i < n; i++) {
      if (value[i] != -1 || used[i]) continue;
      //printf("!! i = %d, deg[i] = %d\n", i, deg[i]);
      c *= cycle(i, used);
    }
    ans += c;
  } else {
    set(maxd, 0);
    go();
    unset(maxd);
    set(maxd, 1);
    go();
    unset(maxd);
  }
}

int main() {
  scanf("%d%d", &n, &m);

  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++) {
      ok[i][j] = 0xf;
    }

  for (int i = 0; i < m; i++) {
    int a, b;
    scanf("%d%d", &a, &b);
    a = conv(a);
    b = conv(b);
    if (a / 2 != b / 2) {
      f[a ^ 1].push_back(b ^ 1);
      f[b ^ 1].push_back(a ^ 1);
      g[a / 2].push_back(b / 2);
      g[b / 2].push_back(a / 2);
      ok[a/2][b/2] &= 0xf ^ (1 << ((1 - (a & 1)) + 2 * (1 - (b & 1))));
      ok[b/2][a/2] &= 0xf ^ (1 << (2 * (1 - (a & 1)) + (1 - (b & 1))));
    } else {
      assert(a != b);
    }
  }
  for (int i = 0; i < n; i++) {
    sort(g[i].begin(), g[i].end());
    g[i].erase(unique(g[i].begin(), g[i].end()), g[i].end());
    deg[i] = g[i].size();
  }
  memset(value, -1, sizeof(value));
  go();

  printf(INT64 "\n", ans);
}
```

11.13 高精度计算

```
#include<algorithm>
using namespace std;
const int N_huge=850,base=100000000;
char s[N_huge*10];
struct huge{
    typedef long long value;
    value a[N_huge];int len;
    void clear(){len=1;a[len]=0;}
    huge(){clear();}
    huge(value x){*this=x;}
    huge operator =(huge b){
        len=b.len;for (int i=1;i<=len;++i)a[i]=b.a[i]; return *this;
    }
    huge operator +(huge b){
        int L=len>b.len?len:b.len;huge tmp;
        for (int i=1;i<=L+1;++i)tmp.a[i]=0;
        for (int i=1;i<=L;++i){
            if (i>len)tmp.a[i]+=b.a[i];
            else if (i>b.len)tmp.a[i]+=a[i];
            else {
                tmp.a[i]+=a[i]+b.a[i];
                if (tmp.a[i]>=base){
                    tmp.a[i]-=base;++tmp.a[i+1];
                }
            }
        }
        if (tmp.a[L+1])tmp.len=L+1;
            else tmp.len=L;
        return tmp;
    }
    huge operator -(huge b){
        int L=len>b.len?len:b.len;huge tmp;
        for (int i=1;i<=L+1;++i)tmp.a[i]=0;
        for (int i=1;i<=L;++i){
            if (i>b.len)b.a[i]=0;
            tmp.a[i]+=a[i]-b.a[i];
            if (tmp.a[i]<0){
                tmp.a[i]+=base;--tmp.a[i+1];
            }
        }
        while (L>1&&!tmp.a[L])--L;
        tmp.len=L;
        return tmp;
    }
    huge operator *(huge b){
        int L=len+b.len;huge tmp;
        for (int i=1;i<=L;++i)tmp.a[i]=0;
        for (int i=1;i<=len;++i)
            for (int j=1;j<=b.len;++j){
                tmp.a[i+j-1]+=a[i]*b.a[j];
                if (tmp.a[i+j-1]>=base){
                    tmp.a[i+j]+=tmp.a[i+j-1]/base;
                    tmp.a[i+j-1]%=base;
                }
            }
        tmp.len=len+b.len;
```

```
        while (tmp.len>1&&!tmp.a[tmp.len])--tmp.len;
        return tmp;
    }
    pair<huge,huge> divide(huge a,huge b){
        int L=a.len;huge c,d;
        for (int i=L;i;--i){
            c.a[i]=0;d=d*base;d.a[1]=a.a[i];
            //while (d>=b){d-=b;++c.a[i];}
            int l=0,r=base-1,mid;
            while (l<r){
                mid=(l+r+1)>>1;
                if (b*mid<=d)l=mid;
                    else r=mid-1;
            }
            c.a[i]=l;d-=b*l;
        }
        while (L>1&&!c.a[L])--L;c.len=L;
        return make_pair(c,d);
    }
    huge operator /(value x){
        value d=0;huge tmp;
        for (int i=len;i;--i){
            d=d*base+a[i];
            tmp.a[i]=d/x;d%=x;
        }
        tmp.len=len;
        while (tmp.len>1&&!tmp.a[tmp.len])--tmp.len;
        return tmp;
    }
    value operator %(value x){
        value d=0;
        for (int i=len;i;--i)d=(d*base+a[i])%x;
        return d;
    }
    huge operator /(huge b){return divide(*this,b).first;}
    huge operator %(huge b){return divide(*this,b).second;}
    huge &operator +=(huge b){*this=*this+b;return *this;}
    huge &operator -=(huge b){*this=*this-b;return *this;}
    huge &operator *=(huge b){*this=*this*b;return *this;}
    huge &operator ++(){huge T;T=1;*this=*this+T;return *this;}
    huge &operator --(){huge T;T=1;*this=*this-T;return *this;}
    huge operator ++(int){huge T,tmp=*this;T=1;*this=*this+T;return tmp;}
    huge operator --(int){huge T,tmp=*this;T=1;*this=*this-T;return tmp;}
    huge operator +(value x){huge T;T=x;return *this+T;}
    huge operator -(value x){huge T;T=x;return *this-T;}
    huge operator *(value x){huge T;T=x;return *this*T;}
    //huge operator /(value x){huge T;T=x;return *this/T;}
    //huge operator %(value x){huge T;T=x;return *this%T;}
    huge operator *=(value x){*this=*this*x;return *this;}
    huge operator +=(value x){*this=*this+x;return *this;}
    huge operator -=(value x){*this=*this-x;return *this;}
    huge operator /=(value x){*this=*this/x;return *this;}
    huge operator %=(value x){*this=*this%x;return *this;}
    bool operator ==(value x){huge T;T=x;return *this==T;}
    bool operator !=(value x){huge T;T=x;return *this!=T;}
    bool operator <=(value x){huge T;T=x;return *this<=T;}
    bool operator >=(value x){huge T;T=x;return *this>=T;}
```

```
    bool operator <(value x){huge T;T=x;return *this<T;}
    bool operator >(value x){huge T;T=x;return *this>T;}
    huge operator =(value x){
        len=0;
        while (x)a[++len]=x%base,x/=base;
        if (!len)a[++len]=0;
        return *this;
    }
    bool operator <(huge b){
        if (len<b.len)return 1;
        if (len>b.len)return 0;
        for (int i=len;i;--i){
            if (a[i]<b.a[i])return 1;
            if (a[i]>b.a[i])return 0;
        }
        return 0;
    }
    bool operator ==(huge b){
        if (len!=b.len)return 0;
        for (int i=len;i;--i)
            if (a[i]!=b.a[i])return 0;
        return 1;
    }
    bool operator !=(huge b){return !(*this==b);}
    bool operator >(huge b){return !(*this<b||*this==b);}
    bool operator <=(huge b){return (*this<b)||(*this==b);}
    bool operator >=(huge b){return (*this>b)||(*this==b);}
    void str(char s[]){
        int l=strlen(s);value x=0,y=1;len=0;
        for (int i=l-1;i>=0;--i){
            x=x+(s[i]-’0’)*y;y*=10;
            if (y==base)a[++len]=x,x=0,y=1;
        }
        if (!len||x)a[++len]=x;
    }
    void read(){
        scanf(”%s”,s);this->str(s);
    }
    void print(){
        printf(”%d”,(int)a[len]);
        for (int i=len-1;i;--i){
            for (int j=base/10;j>=10;j/=10){
                if (a[i]<j)printf(”0”);
                    else break;
            }
            printf(”%d”,(int)a[i]);
        }
        printf(”\n”);
    }
}f[1005];
int main(){
  f[1]=f[2]=1;
  for(int i=3;i<=1000;i++)f[i]=f[i-1]+f[i-2];
}
```

11.14 高精度计算 - Claris

```
const int B=10000,MAXL=6000;
struct Num{
  int a[MAXL],len,fu;
  Num(){len=1,fu=a[1]=0;}
  Num operator+(Num b){
    Num c;
    c.len=max(len,b.len)+2;
    int i;
    for(i=1;i<=c.len;i++)c.a[i]=0;
    if(fu==b.fu){
      for(i=1;i<=len;i++)c.a[i]=a[i];
      for(i=1;i<=b.len;i++)c.a[i]+=b.a[i];
      for(i=1;i<=c.len;i++)if(c.a[i]>=B)c.a[i+1]++,c.a[i]-=B;
      while(c.len>1&&!c.a[c.len])c.len--;
      c.fu=fu;
    }else{
      bool flag=0;
      if(len==b.len){
        for(i=len;i;i--)if(a[i]!=b.a[i]){
          if(a[i]>b.a[i])flag=1;
          break;
        }
      }else{
        if(len>b.len)flag=1;
      }
      if(flag){
        for(i=1;i<=len;i++)c.a[i]=a[i];
        for(i=1;i<=b.len;i++)c.a[i]-=b.a[i];
        for(i=1;i<=c.len;i++)if(c.a[i]<0)c.a[i+1]--,c.a[i]+=B;
        while(c.len>1&&!c.a[c.len])c.len--;
        c.fu=fu;
      }else{
        for(i=1;i<=b.len;i++)c.a[i]=b.a[i];
        for(i=1;i<=len;i++)c.a[i]-=a[i];
        for(i=1;i<=c.len;i++)if(c.a[i]<0)c.a[i+1]--,c.a[i]+=B;
        while(c.len>1&&!c.a[c.len])c.len--;
        c.fu=b.fu;
      }
    }
    return c;
  }
  Num operator*(Num b){
    Num c;
    c.len=len+b.len+2;
    c.fu=fu^b.fu;
    int i,j;
    for(i=1;i<=c.len;i++)c.a[i]=0;
    for(i=1;i<=len;i++)for(j=1;j<=b.len;j++){
      c.a[i+j-1]+=a[i]*b.a[j];
      if(c.a[i+j-1]>=B){
        c.a[i+j]+=c.a[i+j-1]/B;c.a[i+j-1]%=B;
        if(c.a[i+j]>=B)c.a[i+j+1]+=c.a[i+j]/B,c.a[i+j]%=B;
      }
    }
```

```
    while(c.len>1&&!c.a[c.len])c.len--;
    return c;
  }
  ll operator%(ll b){
    static ll mo[MAXL];
    int i;
    for(i=len;i;i--)mo[i]=a[i];
    mo[0]=0;
    for(i=len;i;i--)mo[i-1]+=(mo[i]%b)*10000LL,mo[i]/=b;
    return mo[0];
  }
  bool iszero(){
    return len==1&&!a[1];
  }
  void read(){
    static char s[10010],ch;
    gets(s+1);
    int i,j,l=std::strlen(s+1);
    while(!((s[l]>='0')&&(s[l]<='9')))l--;
    if(s[1]=='-'){
      fu=1;
      len=(l+2)>>2;
      for(i=1;i<=len;i++)a[i]=0;
      for(i=2,j=l;i<j;i++,j--)ch=s[i],s[i]=s[j],s[j]=ch;
      for(i=l;i>=2;i--)(a[(i+2)>>2]*=10)+=(s[i]-'0');
    }else{
      fu=0;
      len=(l+3)>>2;
      for(i=1;i<=len;i++)a[i]=0;
      for(i=1,j=l;i<j;i++,j--)ch=s[i],s[i]=s[j],s[j]=ch;
      for(i=l;i;i--)(a[(i+3)>>2]*=10)+=(s[i]-'0');
    }
  }
  void write(){
    if(len==1&&!a[1])fu=0;
    if(fu)putchar('-');
    printf("%d",a[len]);
    for(int i=len-1;i;i--)printf("%04d",a[i]);
    puts("");
  }
  void set(int x){
    fu=0;
    if(x>=B){
      len=2;
      a[1]=x%B;
      a[2]=x/B;
    }else{
      len=1;
      a[1]=x;
    }
  }
};
```

11.15 Rope

push_back(x)：在末尾添加 x(x 是 char)

insert(pos,x)：在 pos 插入 x (x 是字符串,x 后面加个 int 参数可以只能 x 中插入几个)

erase(pos,x)：从 pos 开始删除 x 个

replace(pos,x)：从 pos 开始换成 x(x 是字符串,x 后面加个 int 参数可以只能 x 中的前几个)

substr(pos,x)：提取 pos 开始 x 个

copy(x)：复制 rope 中所有内容到 x 字符串

at(x)/[x]：访问第 x 个元素

注意事项：

1、rope 好像并不原声支持对一个字符串复制 n 遍的做法，要自己手写快速幂

2、rope 可以用 += 来做追加操作

3、rope 中访问、修改一个特定字符的操作是 $O(\log length)$ 的

4、rope 中 [] 运算符只能访问不能修改，需要修改要用 mutable_begin()+ 偏移量，得到迭代器再修改

11.15.1 示例 1

展开一个括号压缩的字符串。比如 z(rz)3r(rui)2cumt 展开为 zrzrzrzrruiruicumt。

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#include<stdlib.h>
#include<limits.h>
#include<math.h>
#include<algorithm>
using namespace std;
typedef long long ll;
#include<ext/rope> //header with rope
using namespace __gnu_cxx;//namespace with rope and some additional stuff
rope<char> tillRight(char *str,int &p,int &times){
  rope<char> ans=""; times=0;
  while(str[p]!=')') ans+=str[p++];
  p++;
  while(isdigit(str[p])){
    times=times*10+(str[p++]-'0');
  }
  p--;
  return ans;
}
rope<char> times(rope<char> &src,int times){
  rope<char> ans,tmp=src;
  while(times){
    if(times&1) ans+=tmp;
    times>>=1; tmp+=tmp;
  }
  return ans;
}
void expand(char * str, rope<char>& ss){
  int p=0;
```

```
  ss.clear();
  for(;str[p];p++){
    if(str[p]!='(') ss+=str[p];
    else{
      p++;
      int t;
      crope tmp=tillRight(str,p,t);
      ss.append(times(tmp,t));
    }
  }
}
char str[20005];
int main(){
  while(~scanf("%s",str)){
    rope<char> txt;
    expand(str,txt);
    printf("%s\n",txt.c_str());
  }
}
```

11.15.2 示例 2

给一个 100000 长的串和 100000 次查询，每次要把 $[l, r]$ 内的元素移动到序列开头。输出最后的序列。

```
#include<iostream>
#include<cstdio>
#include<ext/rope> //header with rope
using namespace std;
using namespace __gnu_cxx;//namespace with rope and some additional stuff
int main(){
  ios_base::sync_with_stdio(false);
  rope <int> v;//use as usual STL container
  int n, m;
  cin >> n >> m;
  for(int i = 1; i <= n; ++i)v.push_back(i);//initialization
  int l, r;
  for(int i = 0; i < m; ++i){
    cin >> l >> r;
    --l, --r;
    rope <int> cur = v.substr(l, r - l + 1);
    v.erase(l, r - l + 1);
    v.insert(v.mutable_begin(), cur);
  }
  for(rope <int>::iterator it = v.mutable_begin(); it != v.mutable_end(); ++it)
    cout << *it << " ";
}
```

11.16 pb_ds 的红黑树

```
#include<algorithm>
using namespace std;
#include<ext/pb_ds/assoc_container.hpp>
```

```
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_cxx;
using namespace __gnu_pbds;
#define rbset(T) tree<T,null_type,less<T>,rb_tree_tag,tree_order_statistics_node_update>
struct RBTree{
  rbset(int) rb;
  void init(){
    rb=rbset(int)();
  }
  void insert(int x){//插入
    rb.insert(x);
  }
  void remove(int x){//删除
    rb.erase(x);
  }
  int findKth(int x){//找第k大(k从0开始)
    return *rb.find_by_order(x);
  }
  int findElementRank(int x){//找>=x的第一个元素是排第几
    return rb.order_of_key(x);
  }
};
/*
    ordered_set X;
    X.insert(1);
    X.insert(2);
    X.insert(4);
    X.insert(8);
    X.insert(16);

    cout<<*X.find_by_order(1)<<endl; // 2
    cout<<*X.find_by_order(2)<<endl; // 4
    cout<<*X.find_by_order(4)<<endl; // 16
    cout<<(end(X)==X.find_by_order(6))<<endl; // true

    cout<<X.order_of_key(-5)<<endl;  // 0
    cout<<X.order_of_key(1)<<endl;   // 0
    cout<<X.order_of_key(3)<<endl;   // 2
    cout<<X.order_of_key(4)<<endl;   // 2
    cout<<X.order_of_key(400)<<endl; // 5
*/
```

12 Java

12.1 输入

12.1.1 声明一个输入对象 cin

```
Scanner cin=new Scanner(System.in);
```

12.1.2 输入一个 int 值

```
Int a=cin.nextInt();
```

12.1.3 输入一个大数

```
BigDecimal a=cin.nextBigDecimal();
```

12.1.4 EOF 结束

```
while(cin.hasNext())…{}
```

12.2 输出

输出任意类型的 str。

```
System.out.println(str);//有换行
System.out.print(str);//无换行
System.out.println”(“str);//输出字符串str
System.out.printf(”Hello,%s.Next year,you’ll be %d”,name,age);//C风格输出(无C风格输入)
```

12.3 大数类

12.3.1 赋值

```
BigInteger a=BigInteger.valueOf(12);
BigInteger b=new BigInteger(String.valueOf(12));
BigDecimal c=BigDecimal.valueOf(12.0);
BigDecimal d=new BigDecimal(”12.0”);//建议使用字符串以防止double类型导致的误差
```

也可以用上述方法构造一个临时对象用于参与运算。

```
b.add(BigInteger.valueOf(105));
```

12.3.2 比较

```
c.compareTo(BigDecimal.ZERO)==0//判断相等，c 等于0
c.compareTo(BigDecimal.ZERO)>0//判断大于，c大于0
c.compareTo(BigDecimal.ZERO)<0//判断小于，c小于0
```

12.3.3 基本运算

BigDecimal 除法要 divide(xxxxx,2,BigDecimal.ROUND_HALF_EVEN)，否则除不尽会异常。

```
Big*** add(Big*** b)//加上b
Big*** subtract(Big*** b)//减去b
Big*** multiply(Big*** b)//乘b
Big*** divide(Big*** b)//除b
Big*** pow(int b)//计算this^b，注意b只能是int类型
Big*** remainder(Big*** b)//mod b，即计算this%b
Big*** abs()//返回this的绝对值
Big*** negate()//返回-this
Big*** max(Big*** b)//返回this和b中的最大值
Big*** min(Big*** b)//返回this和b中的最小值
```

BigInteger 特有的函数：

```
gcd(BigInteger val)//返回一个BigInteger，其值是abs(this)和abs(val)的最大公约数
mod(BigInteger val)//求 this mod val
modInverse(BigInteger val)//求逆元，返回this^(-1) mod m
```

12.3.4 BigDecimal 的格式控制

toString() 将 BigDecimal 对象的数值转换成字符串。之后可配合字符串处理函数进行一些处理：

```
str.startWith("0")//以0开始
str.endWith("0")//以0结束
str.subString(int x,int y)//从x到y的str的子串
str.subString(int x))//从x到结尾的str的子串
c.stripTrailingZeros().toPlainString();//c去除末尾0，并转换成普通字符串
```

setScale(int newScale,RoundingMode roundingMode) 返回 BigDecimal，其标度（小数点后保留位数）为指定值，其非标度值通过此 BigDecimal 的非标度值乘以或除以十的适当次幂来确定，以维护其总值。(用法见下例）

CEILING	向正无限大方向舍入的舍入模式
DOWN	向零方向舍入的舍入模式
FLOOR	向负无限大方向舍入的舍入模式
HALF_DOWN	向最接近数字方向舍入的舍入模式，如果与两个相邻数字的距离相等，则向下舍入
HALF_EVEN	向最接近数字方向舍入的舍入模式，如果与两个相邻数字的距离相等，则向相邻的偶数舍入
HALF_UP	向最接近数字方向舍入的舍入模式，如果与两个相邻数字的距离相等，则向上舍入
UNNECESSARY	用于断言请求的操作具有精确结果的舍入模式，因此不需要舍入
UP	远离零方向舍入的舍入模式

12.3.5 创建 BigDecimal 对象

```
BigDecimal bigNumber=new BigDecimal(”89.1234567890123456789”);
BigDecimal bigRate=new BigDecimal(1000);
BigDecimal bigResult=new BigDecimal();//对象bigResult的值为0.0
```

12.3.6 对 bigNumber 的值乘以 1000，结果赋予 bigResult

```
bigResult=bigNumber.multiply(bigRate);
System.out.println(bigResult);
```

12.3.7 BigInteger 的进制转换

Java 支持的进制范围为 2 36(0 9+ 小写的 a z)。

```
BigInteger a=cin.nextBigInteger(2);//读入一个二进制数
System.out.println(a.toString(2));//输出二进制
```

12.4 小数四舍五入

```
import java.util.*;//输入输出所在的包
import java.math.*;//高精度整数/浮点数所在的包
public class Test{
public static void main(String[] args){
   double i=3.856;
   System.out.println(”四舍五入取整:(3.856)=”
     + new BigDecimal(i).setScale(0,BigDecimal.ROUND_HALF_UP));
   System.out.println(”四舍五入保留两位小数:(3.856)=”
     + new BigDecimal(i).setScale(2,BigDecimal.ROUND_HALF_UP));
}
}
```

12.5 高精度小数 A+B，输出最简结果

```
import java.math.*;
import java.util.*;
public class Main{
  public static void main(String []args){
    Scanner cin=new Scanner(System.in);
    BigDecimal a,b,c;
    while(cin.hasNext()){
      a=cin.nextBigDecimal();b=cin.nextBigDecimal();
      c=a.add(b);
      if(c.compareTo(BigDecimal.ZERO)==0){System.out.println("0"); continue;}
      //不能省，因为stripTrailingZeros()不能很好处理0.00这种情况(JDK8才修复)
      String str=c.stripTrailingZeros().toPlainString();
      if(str.endsWith(".")) str=str.substring(0,str.length()-1);
      System.out.println(str);
    }
  }
}
```

12.6 斐波那契数列

```
import java.util.*;
import java.math.*;
public class Main
{
    public static void main(String[] args){
        BigInteger f[] = new BigInteger[1005];//数组的用法和C#类似
        //二维数组BigInteger[][] f=new BigInteger[1005][1005];
        f[1]=BigInteger.valueOf(1);f[2]=BigInteger.valueOf(1);
        for(int i=3;i<=1000;i++)f[i]=f[i-2].add(f[i-1]);
        Scanner input=new Scanner(System.in);
        int n,t;
        //用for(...;t--;)替代会报错，因为要求第二个表达式必须返回bool
        for(t=input.nextInt();t>0;t--){
            n=input.nextInt();
            System.out.println(f[n]);
        }
    }
}
```

12.7 两个高精度浮点数比较是否相等

```
import java.math.*;
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner input=new Scanner(System.in);
        BigDecimal a,b;int result;
        while(input.hasNextBigDecimal()){
            a=input.nextBigDecimal();
            b=input.nextBigDecimal();
```

```
            result=a.compareTo(b);
            System.out.printf(result==0?"YES\r\n":"NO\r\n");
            //方法1：手工处理。win的oj上换行必须\r\n，否则PE
            //Linux下评测时用\n来表换行
            //方法2：Java中%n表示运行平台决定的换行符，智能的输出\r\n或者\n
        }
    }
}
```

12.8 高效的输入类

```
class FastScanner {
    BufferedReader br;
    StringTokenizer st;
    public FastScanner(InputStream in) {
        br = new BufferedReader(new InputStreamReader(in),16384);
        eat("");
    }
    private void eat(String s) {st = new StringTokenizer(s);}
    public String nextLine() {
        try {
            return br.readLine();
        } catch (IOException e) {
            return null;
        }
    }
    public boolean hasNext() {
        while (!st.hasMoreTokens()) {
            String s = nextLine();
            if(s == null)return false;
            eat(s);
        }
        return true;
    }
    public String next() {
        hasNext();
        return st.nextToken();
    }
    public int nextInt() {return Integer.parseInt(next());}
    public double nextDouble() {return Double.parseDouble(next());}
    //需要的其他类型比如long可以仿照
    //BigInteger建议这样写：BigInteger test=new BigInteger(in.next());
    //使用方法：FastScanner in=new FastScanner(System.in);
}
```

12.9 输出外挂

注意输出量很大的时候才有效果，否则会起一定的拖慢反作用。

```
PrintWriter out = new PrintWriter(
                  new BufferedWriter(new OutputStreamWriter(System.out)));
out.println();out.print();//输出使用照常
out.flush();//注意最后一定要追加，不然会WA
```

13 战术研究

13.1 注意点

1. 读新题的优先级高于一切
2. 读完题之后必须看一遍 clarification
3. 交题之前必须看一遍 clarification
4. 可能有 SPJ 的题目提交前也应该尽量做到与样例输出完全一致
5.WA 时需要检查 INF 是否设小
6. 构造题不可开场做
7. 每道题需至少有两个人确认题意
8. 上机之前做法需得到队友确认
9. 带有猜想性质的算法应放后面写
10. 当发现题目不会做但是过了一片时应冲一发暴力
11. 将待写的题按所需时间放入小根堆中，每次选堆顶的题目写
12. 交完题目后立马打印随后让出机器
13. 写题超过半小时应考虑是否弃题
14. 细节、公式等在上机前应在草稿纸上准备好，防止上机后越写越乱
15. 提交题目之前应检查 $solve(n, m)$ 是否等于 $solve(m, n)$
16. 检查是否所有东西都已经清空
17. 对于中后期题应该考虑一人写题，另一人在一旁辅助，及时发现手误
18. 最后半小时不能慌张
19. 对于取模的题，在输出之前一定要再取模一次进行保险

对于舍入输出，若 abs 不超过 eps，需要强行设置为 0 来防止 -0.0000 的出现

13.2 打表找规律方法

1. 直接找规律
2. 差分后找规律
3. 找积性
4. 点阵打表
5. 相除
6. 找循环节
7. 凑量纲
8. 猜想满足 $P(n)f(n) = Q(n)f(n-2) + R(n)f(n-1) + C$，其中 P, Q, R 都是关于 n 的二次多项式