ACM-ICPC 模板整理

HDU_Supportornot

2017 年 1 月 6 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);
  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;
```

```cpp
    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 $O(qn^{\frac{3}{2}})$。

```cpp
#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(st[y]-1,p),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));
  }
}
```

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());
}
```

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[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

```
int dfn[N],low[N],num,cut[N],q[N],t,sub;
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){//x是割点，接下来一行输出所有该点双连通分量内的点
      cut[x]=1;
      while(q[t]!=x)printf("%d ",q[t--]);
      printf("%d\n",x);
    }
```

```
  }else if(low[x]>dfn[v[i]])low[x]=dfn[v[i]];
}
```

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。

```
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;
}
```

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
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++;
  }
}
```

5.13 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。

```
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];
      }
      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;
}
```

5.16 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 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);
}
```

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 0，SG 为 0 且每一堆石子数都为 1

必败：其余为必败。

6.6 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。

6.7 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。

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

6.8 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。

6.9 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
  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;
}
```

6.10 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。

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 $999999598=2\times13\times5281\times7283$ 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 $(x\ge y)$：$C(n+m,m)-C(n+m,m-1)$。

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。

```
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 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。

7.15 曼哈顿距离的变换

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

7.16 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，$\bigtriangledown f=\lambda\times\bigtriangledown g$。

7.17 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;
}
```

7.18 组合数取模

7.18.1 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);
}
```

7.18.2 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);
}
```

7.19 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。

7.20 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 $V + 1$。

7.21 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;
        }
    }
}
```

7.22 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：

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

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

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

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

7.22.1 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();
}
```

7.23 快速傅里叶变换

下列模板中 n 必须为 2 的幂。

7.23.1 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;
}
```

7.23.2 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]);
}
```

7.23.3 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。

7.23.4 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。

7.24 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。

7.25 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：$S=n+\frac{s}{2}-1$。

7.26 组合数 lcm

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

7.27 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。

7.28 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;
}
```

7.29 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;
}
```

7.30 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 幂和

$$
\begin{aligned}
\sum_{i=1}^{n} i^1 &= \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n \\
\sum_{i=1}^{n} i^2 &= \frac{n(n+1)(2n+1)}{6} = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n \\
\sum_{i=1}^{n} i^3 &= \left[\frac{n(n+1)}{2}\right]^2 = \frac{1}{4}n^4 + \frac{1}{2}n^3 + \frac{1}{4}n^2 \\
\sum_{i=1}^{n} i^4 &= \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30} = \frac{1}{5}n^5 + \frac{1}{2}n^4 + \frac{1}{3}n^3 - \frac{1}{30}n \\
\sum_{i=1}^{n} i^5 &= \frac{n^2(n+1)^2(2n^2+2n-1)}{12} = \frac{1}{6}n^6 + \frac{1}{2}n^5 + \frac{5}{12}n^4 - \frac{1}{12}n^2 \\
\sum_{i=1}^{n} i^6 &= \frac{n(n+1)(2n+1)(3n^4+6n^3-3n+1)}{42} = \frac{1}{7}n^7 + \frac{1}{2}n^6 + \frac{1}{2}n^5 - \frac{1}{6}n^3 + \frac{1}{42}n
\end{aligned}
$$

```
//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);
}
```

7.32 斯特林数

7.32.1 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：$S(p,0)=0, p\ge 1\quad S(p,p)=1, p\ge 0$

7.32.2 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：

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

7.33 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

7.34 错排公式

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

7.35 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。

7.36 二项式反演

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

7.37 x^k 的转化

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

7.38 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));
}
```

7.39 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。

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());
  }
}
```

7.40 法雷序列

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

```
#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);
  }
}
```

7.41 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 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]);
}
```

7.42 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));
  }
}
```

7.43 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
```

7.44 O(1) 求 GCD

```
#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;
    }
  }
}
```

7.45 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;
}
```

7.46 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 (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;
}
```

7.47 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);
}
```

7.48 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。

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

7.49 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 $k \times n^{n-k-1}$。

7.50 斯特林近似公式

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

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];
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];}
```

8.5 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;
struct node{int st,en,lk,son[S];int len(){return min(en,pos+1)-st;}}tree[N<<1];
int n;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()){
    ace+=tree[node].len(),aclen-=tree[node].len(),acnode=node;
    return 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 y){
  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]);puts(":");
  for(int i=0;i<S;i++)if(tree[x].son[i])show(tree[x].son[i],dep+2,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('$');
  show(root,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;
}
```

8.10 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);
    }
  }
}
```

8.11 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));
  }
}
```

8.12 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;
}
```

8.13 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;
}
```

8.14 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);
  }
}
```

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]);
    }
}
```

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。

```
#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( 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;}
};

//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
bool point_in_polygon(vec *v,int n,vec c) {
  double s = 0;
  for(int i=0;i<n;++i) {
    vec a = v[i] - c,b = v[(i+1)%n] - c;
    double ang = acos( a*b/(a.len() * b.len()) );
    s += cross(a,b) < 0 ? ang : -ang;
  }
  return sgn(s) != 0;// s=0在多边形外,s=pi在边缘上,否则在多边形内
}

//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;
}

//----------------------------------------------------------------
//       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 $O(n^2 \log n)$。

```
#include<cstdio>
#include<cmath>
#include<algorithm>
using namespace std;
typedef pair<double,double>P;
const int N=1010;
const double PI=acos(-1.0),eps=1e-8;
int n,i,j,del[N],t;
double a[N],b[N],r[N],d,x,y,u,ans;
P q[N],tmp;
double sqr(double x){return x*x;}
double dis(double x1,double y1,double x2,double y2){
  return sqrt(sqr(x1-x2)+sqr(y1-y2));
}
double angle(double a,double b,double c){
  return acos((sqr(a)+sqr(b)-sqr(c))/(2*a*b));
}
int sig(double x){
  if(fabs(x)<eps)return 0;
  return x>0?1:-1;
}
void cal(int i,double L,double R){
  double x1=a[i]+r[i]*cos(L),y1=b[i]+r[i]*sin(L),
         x2=a[i]+r[i]*cos(R),y2=b[i]+r[i]*sin(R);
  ans+=r[i]*r[i]*(R-L-sin(R-L))+x1*y2-x2*y1;
}
int main(){
  scanf("%d",&n);
  for(i=0;i<n;i++){
    scanf("%lf%lf%lf",&a[i],&b[i],&r[i]);
    for(j=0;j<i;j++)if(!sig(r[i]-r[j])&&!sig(a[i]-a[j])&&!sig(b[i]-b[j])){
      r[i]=0;
      break;
    }
  }
  for(i=0;i<n;i++)for(j=0;j<n;j++)
    if(i!=j&&sig(r[j]-r[i])>=0&&sig(dis(a[i],b[i],a[j],b[j])-r[j]+r[i])<=0){
      del[i]=1;
      break;
    }
  for(i=0;i<n;i++)if(sig(r[i])&&!del[i]){
    for(t=j=0;j<n;j++)if(i!=j){
      d=dis(a[i],b[i],a[j],b[j]);
      if(sig(d-r[i]-r[j])>=0||sig(d-fabs(r[i]-r[j]))<=0)continue;
      x=atan2(b[j]-b[i],a[j]-a[i]),y=angle(r[i],d,r[j]);
      tmp=P(x-y,x+y);
      if(sig(tmp.first)<=0&&sig(tmp.second)<=0)q[t++]=P(2*PI+tmp.first,2*PI+tmp.second);
      else if(sig(tmp.first)<0)q[t++]=P(2*PI+tmp.first,2*PI),q[t++]=P(0,tmp.second);
      else q[t++]=tmp;
```

```
    }
    if(t)sort(q,q+t);
    for(u=0,j=0;j<t;j++)
      if(sig(u-q[j].first)>=0)u=max(u,q[j].second);
      else cal(i,u,q[j].first),u=q[j].second;
    if(!sig(u))ans+=r[i]*r[i]*2*PI;else cal(i,u,2*PI);
  }
  ans/=2;
  printf("%.3f",ans);
}
```

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
}
```

10.9 Descartes' Theorem

平面上 4 个圆相切于不同的 6 个点时，四个圆的半径满足

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

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());
}
```

11 黑科技与杂项

11.1 开栈

11.1.1 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));
}
```

11.1.2 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));
```

11.1.3 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
```

11.2 I/O 优化

11.2.1 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');
}
```

11.2.2 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;
}
```

11.2.3 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 位运算及其运用

11.3.1 枚举子集

枚举 i 的非空子集 j。

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

11.3.2 求 1 的个数

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

11.3.3 求前缀 0 的个数

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

11.3.4 求后缀 0 的个数

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

11.4 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);
  }
}
```

11.5 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=1000000000000000LL;
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);
}
```

11.6 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);
}
```

11.7 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;
  }
}
```

11.8 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]
  }
}
```

11.9 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 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]);
  }
}
```

11.11 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];
}
```

11.12 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;
    }
  }
};
```

11.13 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

11.13.1 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());
  }
}
```

11.13.2 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。

```cpp
#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 << " ";
}
```

11.14 pb_ds 的红黑树

```cpp
#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

```
Big*** add(Big*** b)//加上b
Big*** subtract(Big*** b)//减去b
Big*** multiply(Big*** b)//乘b
Big*** divided(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
```