Huantwofat之ACM模板库

目录：

39. 凸包面积模板
40. 完整高精度算法系统
41. 网络流 Dinic算法
42. 网络流SAP最大流
43. 网络流判圈
44. 威佐夫博奕模板
45. 无源无汇上下限网络流
46. 线段树 扫描线
47. 斜率dp模板
48. 匈牙利算法
49. 压位高精度系统
50. 硬币问题的（NV）模板
51. 有源汇的上下界最小流
52. 约瑟夫问题
53. 在可解时间之内判断回文串算法（非后缀数组）
54. 中国剩余定理模板
55. 字典树
56. 最小费用最大流模板
57. 最长递增子序列模板
58. 最长上升子序列nlogn通解（dp之线段树优化）
59. 母函数模板
60. 上下限网络流大攻略

1、树状数组模板

```
 int bit[100001],n;
int sum(int a)
{
    int s=0;
    while(a>=0)
    {
        s+=bit[a];
        a-=a&(-a);
    }
    return s;
}
void merg(int a,int b)
{
    while(a<=n)
    {
        bit[a]+=b;
        a+=a&(-a);
    }
    return ;
}
```

2、二维树状数组模板

```
#include<stdio.h>
int bit[1025][1025],n;
```

```
int low(int a)
{
    return a&(-a);
}
int sum(int x,int y)
{
    int s;
    int p;
    s=0;
    while(x>0)
    {
        p=y;
        while(p>0)
        {
            s+=bit[x][p];
            p=p-low(p);
        }
        x=x-low(x);
    }
    return s;
}
void merg(int x,int y,int s)
{
    int p;
    while(x<=n)
    {
        p=y;
        while(p<=n)
        {
            bit[x][p]+=s;
            p=p+low(p);
        }
        x=x+low(x);
    }
    return ;
}
```

3、线段树 扫描线

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<iostream>
#include<algorithm>
#include<cmath>
#define eps 1e-12
using namespace std;
struct pi
{
    int l;
    int r;
    int cover;
    double len;
}pp[2005];
struct line
```

```
{
    double up;
    double down;
    double x;
    int cover;
}pp1[255];
double y[255];
int n,len;
void build(int a,int p,int q)
{
    pp[a].l=p;
    pp[a].r=q;
    int mid;
    mid=(p+q)/2;
    pp[a].cover=0;
    pp[a].len=0;
    if(p+1==q)
        return ;
    build(2*a,p,mid);
    build(2*a+1,mid,q);
}
void maintan(int k)
{
    if(pp[k].cover>0)
    {
        pp[k].len=y[pp[k].r]-y[pp[k].l];
        return ;
    }
    if(pp[k].r==pp[k].l+1)
    {
        pp[k].len=0;
        return ;
    }
    pp[k].len=pp[2*k].len+pp[2*k+1].len;
}
void merg(int k,int l,int r,int cover)
{
    if(pp[k].l>r||pp[k].r<l)
        return ;
    if(pp[k].l>=l&&pp[k].r<=r)
    {
        pp[k].cover+=cover;
        maintan(k);
        return ;
    }
    merg(2*k,l,r,cover);
    merg(2*k+1,l,r,cover);
    maintan(k);
    return ;
}
int cmp(struct line a,struct line b)
{
    return   a.x<b.x;
```

```
}
int find(double a)
{
    int l=0,r=len,mid;
    while(l<=r)
    {
        mid=(l+r)/2;
        if(y[mid]==a)
            return mid;
        if(y[mid]>a)
        {
            r=mid-1;
        }
        else
            l=mid+1;
    }
    return l;
}
int main()
{
    int i,p,q,N=0;
    double x1,y1,x2,y2,s;
    while(1)
    {
        N++;
        cin>>n;
        if(n==0)
            break;
        for(i=0;i<n;i++)
        {
            cin>>x1>>y1>>x2>>y2;
            pp1[2*i].x=x1;
            pp1[2*i].up=y2;
            pp1[2*i].down=y1;
            pp1[2*i].cover=1;
            pp1[2*i+1].x=x2;
            pp1[2*i+1].up=y2;
            pp1[2*i+1].down=y1;
            pp1[2*i+1].cover=-1;
            y[2*i]=y1;
            y[2*i+1]=y2;
        }
        s=0;
        sort(y,y+2*n);
        sort(pp1,pp1+2*n,cmp);
        len=1;
        for(i=1;i<2*n;i++)
        {
            if(y[i-1]!=y[i])
            {
                y[len++]=y[i];
            }
        }
```

```
            len--;
            build(1,0,len);
            for(i=0;i<2*n-1;i++)
            {
                p=find(pp1[i].down);
                q=find(pp1[i].up);
                merg(1,p,q,pp1[i].cover);
                s+=(pp1[i+1].x-pp1[i].x)*pp[1].len;
            }
            printf("Test case #%d\n",N);
            printf("Total explored area: %0.2lf\n\n",s);
        }
        return 0;
        }
```

4、三维扫描线

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cstdlib>
#include<cmath>
#define maxn 50005
using namespace std;
typedef __int64 LL;
int y[maxn<<1];
int z[maxn<<1];
int tot;
struct pi{
    int x1;
    int y1;
    int z1;
    int x2;
    int y2;
    int z2;
}pp[maxn];
struct ppi{
    int x;
    int up;
    int down;
    int cover;
}pp1[maxn];
struct pppi{
    int l;
    int r;
    int cover;
    int len;
    int len1;
    int len2;
}pp2[maxn];
void build(int p,int l,int r){
    pp2[p].l=l;
    pp2[p].r=r;
    pp2[p].len=0;
    pp2[p].len1=0;
```

```
    pp2[p].len2=0;
    pp2[p].cover=0;
    if(r==l+1) return ;
    build(2*p,l,(l+r)/2);
    build(2*p+1,(l+r)/2,r);
}
void update(int k){
    if(pp2[k].cover>=3){
        pp2[k].len=y[pp2[k].r]-y[pp2[k].l];
        pp2[k].len1=0;
        pp2[k].len2=0;
        return ;
    }
    if(pp2[k].cover==2){
        if(pp2[k].l+1!=pp2[k].r){
            pp2[k].len=pp2[2*k].len1+pp2[2*k].len
+pp2[2*k].len2+pp2[2*k+1].len2+pp2[2*k+1].len1+pp2[2*k+1].len;
            pp2[k].len1=y[pp2[k].r]-y[pp2[k].l]-pp2[k].len;
            pp2[k].len2=0;
        }
        else{
            pp2[k].len=0;
            pp2[k].len1=y[pp2[k].r]-y[pp2[k].l];
            pp2[k].len2=0;
        }
    }
    else if(pp2[k].cover==1){
        if(pp2[k].l+1!=pp2[k].r){
            pp2[k].len=pp2[2*k].len1+pp2[2*k].len+pp2[2*k
+1].len1+pp2[2*k+1].len;
            pp2[k].len1=pp2[2*k].len2+pp2[2*k+1].len2;
            pp2[k].len2=y[pp2[k].r]-y[pp2[k].l]-pp2[k].len-
pp2[k].len1;
        }
        else{
            pp2[k].len=0;
            pp2[k].len1=0;
            pp2[k].len2=y[pp2[k].r]-y[pp2[k].l];
        }
    }
    else{
        if(pp2[k].l+1!=pp2[k].r){
            pp2[k].len=pp2[2*k].len+pp2[2*k+1].len;
            pp2[k].len1=pp2[2*k].len1+pp2[2*k+1].len1;
            pp2[k].len2=pp2[2*k].len2+pp2[2*k+1].len2;
        }
        else{
            pp2[k].len=0;
            pp2[k].len1=0;
            pp2[k].len2=0;
        }
    }
}
```

```
void merg(int p,int l,int r,int cover){
    if(pp2[p].l>r||pp2[p].r<l) return ;
    if(pp2[p].l>=l&&pp2[p].r<=r){
        pp2[p].cover+=cover;
        update(p);
        return ;
    }
    merg(2*p,l,r,cover);
    merg(2*p+1,l,r,cover);
    update(p);
}
void add(int x1,int y1,int x2,int y2){
    pp1[tot].x=x1;
    pp1[tot].up=y2;
    pp1[tot].down=y1;
    pp1[tot].cover=1;
    tot++;
    pp1[tot].x=x2;
    pp1[tot].up=y2;
    pp1[tot].down=y1;
    pp1[tot].cover=-1;
    tot++;
}
int cmp(ppi a,ppi b){
    if(a.x!=b.x) return a.x<b.x;
    if(a.cover>0) return 1;
    return 0;
}
LL solve(int n,int cal){
    int i,j,m,len,too,k;
    LL s,are;
    sort(y,y+cal);
    sort(z,z+cal);
    s=0;
    len=unique(y, y+cal)-y;
    too=unique(z,z+cal)-z;
    build(1,0,len-1);
    tot=0;
    for(i=0;i<too-1;i++){
        are=0;
        tot=0;
        for(j=0;j<n;j++){
            if(pp[j].z1<=z[i]&&pp[j].z2>z[i]){
                add(pp[j].x1,pp[j].y1,pp[j].x2,pp[j].y2);
            }
        }
        sort(pp1,pp1+tot,cmp);
        for(j=0;j<tot;j++){
            m=lower_bound(y, y+len, pp1[j].up)-y;
            k=lower_bound(y, y+len, pp1[j].down)-y;
            merg(1,k,m,pp1[j].cover);
            if(j<tot-1)
                are+=1LL*(pp1[j+1].x-pp1[j].x)*pp2[1].len;
```

```
        }
        s+=are*(z[i+1]-z[i]);
    }
    return s;
}
int main(){
    int i,j,n,t,N,cal;
    scanf("%d",&t);
    N=t;
    while(t--){
        scanf("%d",&n);
        cal=0;
        for(i=0;i<n;i++){
            scanf("%d%d%d%d%d
%d",&pp[i].x1,&pp[i].y1,&pp[i].z1,&pp[i].x2,&pp[i].y2,&pp[i].z
2);
            y[cal]=pp[i].y1;
            z[cal]=pp[i].z1;
            cal++;
            y[cal]=pp[i].y2;
            z[cal]=pp[i].z2;
            cal++;
        }
        if(n<=2){
            printf("Case %d: 0\n",N-t);
            continue;
        }
        printf("Case %d: %I64d\n",N-t,solve(n,cal));
    }
    }
```

5、扩展欧几里得

```
int x,y;
int exgcd(int a,int b)
{
    if(b==0)
    {
        x=1;
        y=0;
        return a;
    }
    int p,t;
    p=exgcd(b,a%b);
    t=x;
    x=y;
    y=t-a/b*y;
    return p;
    }
```

6、平面最近点分治模板

```
#include<stdio.h>
#include<math.h>
#include<algorithm>
using namespace std;
struct pi
```

```
{
    double x;
    double y;
}pp[100001];
struct pi pp1[100001];
double pinfan(int a,int b)
{
    if(a==b)
        return 0;
    return sqrt((pp[a].x-pp[b].x)*(pp[a].x-pp[b].x)+(pp[a].y-
pp[b].y)*(pp[a].y-pp[b].y));
}
double pinfan2(int a,int b)
{
    if(a==b)
        return 0;
    return sqrt((pp1[a].x-pp1[b].x)*(pp1[a].x-pp1[b].x)+
(pp1[a].y-pp1[b].y)*(pp1[a].y-pp1[b].y));
}int cmp1(struct pi a,struct pi b)
{
    return a.y<b.y;
}
double jisuan(int a,int b)
{
    int p=(a+b)/2,t,m,i,tot,j;
    if(a==b)
        return 0;
    double min=1000000,f,k;
    if(b-a==2)
    {
        f=pinfan(a,a+1);
        if(f<min)
            min=f;
        f=pinfan(a,b);
        if(f<min)
            min=f;
        f=pinfan(a+1,b);
        if(f<min)
            min=f;
        return min;
    }
    if(b-a==1)
        return pinfan(a,b);
    k=jisuan(a,p);
    f=jisuan(p,b);
    if(k>f)
        k=f;
    t=p;
    while(t<=b&&(pp[t].x-pp[p].x)<=k)
        t++;
    m=p;
    while(m>=a&&(pp[p].x-pp[t].x)<=k)
        m--;
```

```
    tot=0;
    for(i=m+1;i<=t-1;i++)
    {
        pp1[tot].x=pp[i].x;
        pp1[tot++].y=pp[i].y;
    }
    sort(pp1,pp1+tot,cmp1);
    for(i=0;i<tot;i++)
    {
        for(j=1;j<=7;j++)
        {
            if(i+j>=tot)
                break;
            f=pinfan2(i,i+j);
            if(f<min)
                min=f;
        }
    }
    if(min>k)
        min=k;
    return min;
}
int cmp(struct pi a,struct pi b)
{
    return a.x<b.x;
}
int main()
{
    int n,i;
    double s;
    while(1)
    {
        scanf("%d",&n);
        if(n==0)
            break;
        for(i=0;i<n;i++)
        {
            scanf("%lf%lf",&pp[i].x,&pp[i].y);
        }
        if(n==2)
        {
            printf("%.2lf\n",pinfan(0,1)/2);
            continue;
        }
        sort(pp,pp+n,cmp);
        s=jisuan(0,n-1);
        printf("%.2lf\n",s/2);
    }
    return 0;
    }
```

7、KMP模板

```
#include<stdio.h>
#include<string.h>
```

```
#include<iostream>
using namespace std;
char c[100005];
int nex[100005];
int main()
{
    int i,j,m;
    cin>>c;
    m=strlen(c);
    i=0;j=-1;
    nex[0]=-1;
    while(i<m)
    {
        if(j==-1||c[i]==c[j])
        {
            i++;
            j++;
            nex[i]=j;
        }
        else
        {
            j=nex[j];
        }
    }
    return 0;
    }
```

8、凸包面积模板

```
#include<cstdio>
#include<iostream>
#include<algorithm>
#include<cmath>
#define eps 1e-8
using namespace std;
struct pi
{
    double x;
    double y;
}pp[10005];
pi pp1[10005];
int tot;
int dd(double x,double y)
{
    return fabs(x-y)<eps;
}
int xy(double x,double y)
{
    return x+eps<y;
}
int yx(double x,double y)
{
    return y+eps<x;
}
int xdy(double x,double y)
```

```
{
    return x<y+eps;
}
int ydx(double x,double y)
{
    return y<x+eps;
}
double cross(pi p1,pi p2,pi p3)
{
    double a;
    a=(p2.x-p1.x)*(p3.y-p1.y)-(p2.y-p1.y)*(p3.x-p1.x);
    return a;
}
double dist(pi a,pi b)
{
    return sqrt((double)(b.x-a.x)*(b.x-a.x)+(b.y-a.y)*(b.y-
a.y));
}
int cmp1(pi a,pi b)
{
    if(dd(a.x,b.x))
        return xy(a.y,b.y);
    return xy(a.x,b.x);
}
int cmp2(pi a,pi b)
{
    int p;
    p=cross(pp[0],a,b);
    if(p==0)
    {
        return xy(dist(pp[0],a),dist(pp[0],b));
    }
    return yx(p,0);
}
double mianji(void)
{
    int i;
    double s;
    s=0;
    for(i=2;i<=tot-1;i++)
    {
        s+=cross(pp1[1],pp1[i],pp1[i+1]);
    }
    return s/2.0;
}
double ghama(int n)
{
    tot=0;
    sort(pp,pp+n,cmp1);
    sort(pp+1,pp+n,cmp2);
    pp1[++tot]=pp[0];
    pp1[++tot]=pp[1];
    for(int i=2;i<n;i++)
```

```
    {
        if(ydx(cross(pp1[tot],pp[i],pp1[tot-1]),0))
        {
            pp1[++tot]=pp[i];
            continue;
        }
        while(xy(cross(pp1[tot],pp[i],pp1[tot-1]),0))
            tot--;
        pp1[++tot]=pp[i];
    }
    return mianji();
}
int main()
{
    int i,n,q;
    double s,j,p;
    while(cin>>n)
    {
        for(i=0;i<n;i++)
        {
            scanf("%lf%lf",&j,&p);
            pp[i].x=j;
            pp[i].y=p;
        }
        s=ghama(n);
        q=floor((s+eps)/50+eps);
        printf("%d\n",q);
    }
    return 0;
    }
```

9、巴什博弈模板

```
//只有一堆n个物品,两个人轮流从这堆物品中取物,规定每次至少取一个,最多取m个.最后取光者得胜.

//若(m+1) | n，则先手必败，否则先手必胜。

//显然,如果n=m+1,那么由于一次最多只能取m个,所以,无论先取者拿走多少个,后取者都能够一次拿走剩余的物品,后者取胜.因此我们发现了如何取胜的法则：如果n=(m+1)r+s,(r为任意自然数,s≤m),那么先取者要拿走s个物品,如果后取者拿走k(≤m)个,那么先取者再拿走m+1-k个,结果剩下(m+1)(r-1)个,以后保持这样的取法,那么先取者肯定获胜.总之,要保持给对手留下(m+1)的倍数,就能最后获胜.
```

10、莫比乌斯反演

```
#include<stdio.h>
#include<string.h>
#include<iostream>
#include<algorithm>
#define max 100000
using namespace std;
int u[max+20];
long long f[max+20];
long long g[max+20];
bool vis[max+20];
```

```
int mmax(int a,int b)
{
    int p;
    p=a;
    if(b<a)
        p=b;
    return p;
}
int main()
{
    int n,i,j,k,p,t,N,m;
    fill(u,u+max+19,1);
    u[1]=1;
    memset(vis,0,sizeof(vis));
    for(i=2;i<=max+18;i++)
    {
        if(vis[i])
            continue;
        if(u[i]==0)
            continue;
        for(j=i;j<=max+2;j=j+i)
        {
            if(u[j]==0)
            {
                continue;
            }
            if((j/i)%i==0)
            {
                u[j]=0;
            }
            else
            {
                u[j]=-u[j];
            }
            vis[j]=1;
        }
    }
    return 0;
    }
```

11、博弈论之Grungy数及sg函数

```
#include<stdio.h>
#include<string.h>
#define N 10000
int grun[N],b[N];
bool ha[N];
void grundy(int n)
{
    int i,j;
    memset(grun,0,sizeof(grun));
    for(i=1;i<=N;i++)
    {
        memset(ha,0,sizeof(ha));
        for(j=0;j<n;j++)
```

```
        {
            if(i>=b[j])
            {
                ha[grun[i-b[j]]]=1;
            }
        }
        for(j=0;j<=N;j++)
        {
            if(!ha[j])
                break;
        }
        grun[i]=j;
    }
}
//1.可选步数为1~m的连续整数，直接取模即可，SG(x) = x % (m+1);

//2.可选步数为任意步，SG(x) = x;

//3.可选步数为一系列不连续的数，用GetSG()
```

12、网络流 Dinic算法

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<vector>
#include<string.h>
#include<queue>
#define inf 100000000
using namespace std;
struct pi
{
    int to;
    int cost;
    int rev;
}pp;
vector<pi >g[505];
queue<int>q;
int s,t;
int line[505],leve[505];
int map[505][505];
int min(int a,int b)
{
    int p;
    p=a;
    if(b<a)
        p=b;
    return p;
}
void add(int a,int b,int cos)
{
    pp.to=b;
    pp.cost=cos;
    pp.rev=(int)g[b].size();
    g[a].push_back(pp);
```

```
    pp.to=a;
    pp.cost=0;
    pp.rev=(int)g[a].size()-1;
    g[b].push_back(pp);
    return ;
}
void bfs(void)
{
    q.push(s);
    int p,f,i;
    while(!q.empty())
    {
        p=q.front();
        q.pop();
        f=(int)g[p].size();
        for(i=0;i<f;i++)
        {
            pi &e=g[p][i];
            if(e.cost>0&&line[e.to]<0)
            {
                line[e.to]=line[p]+1;
                q.push(e.to);
            }
        }
    }
    return ;
}
int dfs(int v,int f)
{
    int p;
    if(v==t)
        return f;
    for(int &i=leve[v];i<g[v].size();i++)
    {
        pi &e=g[v][i];
        if(line[v]<line[e.to]&&e.cost>0)
        {
            p=dfs(e.to,min(f,e.cost));
            if(p>0)
            {
                e.cost-=p;
                g[e.to][e.rev].cost+=p;
                return p;
            }
        }
    }
    return 0;
}
long long  dinic()
{
    int f;
    long long flow=0;
    while(1)
```

```
    {
        memset(line,-1,sizeof(line));
        memset(leve,0,sizeof(leve));
        line[s]=0;
        bfs();
        if(line[t]<0)
            return flow;
        f=dfs(s,inf);
        if(f==0)
            continue;
        flow+=f;
        while((f=dfs(s,inf))>0)
        {
            flow+=f;
        }
    }
}
int main()
{
    int i;
    for(i=1;i<=s;i++)
        g[i].clear();
    }
```

13、polya定理模板

```
//设G是p个对象的一个置换群，用m种颜色涂染p个对象，则不同染色方案为:
//l=(sigma(m^(c(g[i]))))/|G|;c[g[i]]为c[i]循环节个数。
//其中G={g1 ,…gs}   c(gi )为置换gi的循环节数(i=1…s)
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<math.h>
using namespace std;
int gcd(int a,int b)
{
    while(b^=a^=b^=a%=b)
        ;
    return a;
}
//为环时的polay
void polya(int n,int m)
{
    int s=0,i,j,p;
    for(i=1;i<=n;i++)
    {
        s+=(int)pow((double)m,(double)gcd(n,i));
    }
    if(n&1)
        s+=n*(int)pow((double)m,(double)(n+1)/2);
    else
    {
        p=(int)pow((double)m,(double)n/2);
        s+=n/2*p+n/2*p*m;
    }
```

```
    s=s/n/2;
    return ;
}
//1. 对于旋转，有c(gi) = gcd(n,i)，i为转动几颗珠子。
//2. 对于翻转，当n为奇数时，c(gi) = n/2+1；当n为偶数时，有n/2个的循环
节数为n/2+1，有n/2个的循环节数为n/2。
//如果旋转4下，及正方形的旋转//l=(m^(n^2)+2*m^([(n^2+3)/4])
+m^([(n^2+1)/2]))
//为正方形时
int polya2(int n,int m)
{
    return ((int)pow((double)m,n*n)+2*(int)pow((double)m,(n*n
+3)/4)+(int)pow((double)m,(n*n+1)/2))/4;
     }
```

14、gcd模板

```
int gcd(int a, int b)
{
    while ( b ^= a ^= b ^= a %= b );
    return a;
}
```

15、四边形是否为凸模板

```
//用一个点是否在三角形内判断用面积小面积之和是否等于大面积
#include <queue>
#include <stack>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <limits.h>
#include <string.h>
#include <algorithm>
using namespace std;
const int MAX = 10010;
const double eps = 1e-6;
struct pi
{
    int x;
    int y;
}pp[MAX];
int chaji(int a,int b,int c)
{
    return (pp[b].x-pp[a].x)*(pp[c].y-pp[a].y)-(pp[b].y-
pp[a].y)*(pp[c].x-pp[a].x);
}
int panduan(int a,int b,int c,int d)
{
    if(abs(chaji(a,b,c))==abs(chaji(d,a,b))+abs(chaji(d,b,c))
+abs(chaji(d,c,a)))
    {
        return 0;
    }
    if(abs(chaji(d,b,c))==abs(chaji(a,b,c))+abs(chaji(a,c,d))
+abs(chaji(a,d,b)))
```

```
    {
        return 0;
    }
    if(abs(chaji(a,d,c))==abs(chaji(b,a,c))+abs(chaji(b,c,d))
+abs(chaji(b,d,a)))
    {
        return 0;
    }
    if(abs(chaji(a,b,d))==abs(chaji(c,a,b))+abs(chaji(c,b,d))
+abs(chaji(c,d,a)))
    {
        return 0;
    }
    return 1;
     }
```

16、威佐夫博奕模板

```
//有两堆各若干个物品,两个人轮流从某一堆或同时从两堆中取同样多的物品,规定每次至少取一个,多者不限,最后取光者得胜.这种规则下游戏是颇为复杂的。我们用（a[k]，b[k]）（a[k] ≤ b[k] ,k=0，1，2，...,n)表示两堆物品的数量并称其为局势。如果甲面对（0，0），那么甲已经输了，这种局势我们称为奇异局势。
//奇异局势下先手必败，非奇异局势下先手必胜。
//这种情况下是颇为复杂的.我们用(ak,bk)(ak ≤bk ,k=0,1,2,...,n)表示两堆物品的数量并称其为局势,如果甲面对(0,0),那么甲已经输了,这种局势我们称为奇异局势.前几个奇异局势是：(0,0)、(1,2)、(3,5)、(4,7)、(6,10)、(8,13)、(9,15)、(11,18)、(12,20).
//可以看出,a0=b0=0,ak是未在前面出现过的最小自然数,而bk= ak + k,奇异局势有如下三条性质：
//1、任何自然数都包含在一个且仅有一个奇异局势中.
//由于ak是未在前面出现过的最小自然数,所以有ak > ak-1 ,而bk= ak + k > ak-1 + k-1 = bk-1 > ak-1 .所以性质1.成立.
//2、任意操作都可将奇异局势变为非奇异局势.
//事实上,若只改变奇异局势(ak,bk)的某一个分量,那么另一个分量不可能在其他奇异局势中,所以必然是非奇异局势.如果使(ak,bk)的两个分量同时减少,则由于其差不变,且不可能是其他奇异局势的差,因此也是非奇异局势.
//3、采用适当的方法,可以将非奇异局势变为奇异局势.
//假设面对的局势是(a,b),若b = a,则同时从两堆中取走a 个物体,就变为了奇异局势(0,0)；如果a = ak ,b > bk,那么,取走b - bk个物体,即变为奇异局势；如果a = ak , b < bk ,则同时从两堆中拿走ak - ab - ak个物体,变为奇异局势( ab - ak , ab - ak+ b - ak)；如果a > ak ,b= ak + k,则从一堆中拿走多余的数量a - ak 即可；如果a < ak ,b= ak + k,分两种情况,第一种,a=aj (j < k),从第二堆里面拿走b - bj 即可；第二种,a=bj (j < k),从第二堆里面拿走b - aj 即可.
//从如上性质可知,两个人如果都采用正确操作,那么面对非奇异局势,先拿者必胜；反之,则后拿者取胜.
//那么任给一个局势(a,b),怎样判断它是不是奇异局势呢？我们有如下公式：

//ak =[k(1+√5)/2](下取整), bk= ak + k (k∈N)
//奇妙的是其中出现了有关黄金分割数的式子：(1+√5)/2 =1.618...,若两堆物品个数分别为x,y(x<y)，则k=y-x，再判断x是否等于[(y-x)*( √5+1)/2] 即可得知是否是奇异局势。
#include<stdio.h>
#include<iostream>
#include<algorithm>
```

```
#include<math.h>
#define eps 1e-10
using namespace std;
const double jin=(1.0+sqrt(5.0))/2;
int main()
{
    int n,p,k,f;
    while(scanf("%d%d",&n,&p)!=EOF)
    {
        if(n==p)
        {
            printf("1\n");
            continue;
        }
        if(n>p)
            swap(n,p);
        k=p-n;
        f=(int)floor(k*jin+eps);
        if(f==n)
            printf("1\n");
        else
            printf("0\n");
    }
    return 0;
    }
```

17、LCA之离线算法（tarjan）

```
#include<stdio.h>
#include<algorithm>
#include<vector>
#define maxn 1000000
using namespace std;
int fa[maxn],lca[maxn];//lca[maxn]记录某条边的祖先值，而非节点
struct pi
{
    int to;
    int cost;
    int num;
}pp;
vector<pi>g[maxn];
vector<pi>gg[maxn];//输入待处理的值
int vis[maxn];
int find(int a)
{
    if(fa[a]==a)
        return a;
    return fa[a]=find(fa[a]);
}
void add(int a,int b,int cost)
{
    pp.to=b;
    pp.cost=cost;
    g[a].push_back(pp);
    pp.to=a;
```

```
    pp.cost=cost;
    g[b].push_back(pp);
}
void add1(int a,int b,int tot)
{
    pp.to=b;
    pp.num=tot;
    gg[a].push_back(pp);
    pp.to=a;
    pp.num=tot;
    gg[b].push_back(pp);
}
void LCA(int u)
{
    int k,i,p;
    k=gg[u].size();
    fa[u]=u;
    for(i=0;i<k;i++)
    {
        pp=gg[u][i];
        if(vis[pp.to])
        {
            lca[pp.num]=find(pp.to);//运用回溯，当正好回溯到一点时
他的儿子节点的祖先为他，一层一层的往上回溯。
        }
    }
    vis[u]=1;
    k=g[u].size();
    for(i=0;i<k;i++)
    {
        pp=g[u][i];
        if(!vis[pp.to])
        {
            p=pp.to;
            LCA(p);
            fa[p]=u;
        }
    }
    return ;
    }
```

18、最长递增子序列模板

```
#include<stdio.h>
#include<iostream>
#define N 100
using namespace std;
int main(void)
{
    //存储原字符串
    char str[N];
    //b[j]存储以j为长度的递增子序列的结尾元素
    char b[N];
    int cases;
    cout<<"请输入案例个数："<<endl;
```

```
    cin>>cases;
    while(cases--)
    {
        cout<<"请输入字符串："<<endl;
        cin>>str;
        //初始化各变量
        int i;
        int length = strlen(str);
        b[0] = '0'; //b[0]为最小，假设输入的字符全部是字母
        b[1] = str[0];//以1为长度的递增子序列的结尾元素都是str[0]
        int first,mid,end;//分别为二分查找的首，中，尾位置
        int maxLen = 1; //为目前递增子序列最大长度
        for(i=1;i<length;i++)
        {
            first = 0, end = maxLen;
            while(first<=end)
            {
                mid = (first+end)/2;
                if(b[mid]<str[i])
                    first = mid +1;
                else
                    end = mid -1;
            }
            b[first] = str[i];
            if(first>maxLen) maxLen++;
        }
        cout<<"最长递增子序列的长度为："<<maxLen<<endl;
        cout<<"最长递增子序列为："<<endl;
        for(i=1;i<=maxLen;i++)
            cout<<b[i];
        cout<<endl;
    }
    return 0;
    }
```

19、卢斯卡定理模板（大数组合）

```
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<iostream>
#include<algorithm>
using namespace std;
__int64 p;
__int64 jieji[200005];
__int64 ppow(__int64 n,__int64 m)
{

    __int64 s=1,k=n%p;
    while(m>0)
    {
        if(m&1)
        {
            s=(s*k)%p;
        }
```

```
        k=(k*k)%p;
        m>>=1;
    }
    return s;
}
__int64  zuhe(__int64 n,__int64 m)
{
    if(m==0)
        return 1;
    __int64 s=1;
    __int64 q,f;
    while(n>0&&m>0)
    {
        q=n%p;
        f=m%p;
        if(q<f)
            return 0;
        s=((s*jieji[q])%p*ppow((jieji[f]*jieji[q-f])%p,p-2))
%p;
        n=n/p;
        m=m/p;
    }
    return s;
}
int main()
{
    __int64 n,m,t,k;
    int i;
    cin>>t;
    while(t--)
    {
        scanf("%I64d%I64d%I64d",&n,&m,&p);
        jieji[0]=1;
        jieji[1]=1;
        for(i=2;i<=p;i++)
            jieji[i]=(jieji[i-1]*i)%p;
        k=zuhe(n+m,n);
        printf("%I64d\n",k);
    }
    return 0;
}
```

20、LCA在线算法

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<vector>
#include<string.h>
#define max 400005
using namespace std;
int deap[max],vis[max],dis[max],kk[32][max];
int maxlog;
struct pi
{
```

```
    int to;
    int cost;
}pp;
vector<pi>g[max];
void init(int v,int p,int d)
{
    vis[v]=1;
    deap[v]=d;
    int i,k;
     kk[0][v]=p;
    for(i=1;i<maxlog;i++)
    {
        if(kk[i-1][v]<0)
            kk[i][v]=-1;
        else
        {
            kk[i][v]=kk[i-1][kk[i-1][v]];
        }
    }
    k=g[v].size();
    for(i=0;i<k;i++)
    {
        if(!vis[g[v][i].to])
        {
            dis[g[v][i].to]=dis[v]+g[v][i].cost;
            init(g[v][i].to,v,d+1);
        }
    }
    return ;
}
int find(int a,int b)
{
    if(deap[a]>deap[b])
        swap(a,b);
    int i,f;
    f=deap[b]-deap[a];
    for(i=0;i<maxlog;i++)
    {
        if((f>>i)&1)
            b=kk[i][b];
    }
    if(b==a)
        return a;
    for(i=maxlog-1;i>=0;i--)
    {
        if(kk[i][a]!=kk[i][b])
        {
            a=kk[i][a];
            b=kk[i][b];
        }
    }
    return kk[0][a];
}
```

```
int solve(int a,int b)
{
    int f;
    f=find(a,b);
    return dis[a]+dis[b]-2*dis[f];
    }
```

21、点的外接矩形最小周长和面积模板

```
#include <iostream>
#include <cstdio>
#include <cmath>
#include <algorithm>
#include <vector>
using namespace std;
const double eps=1e-6;
const double inf=1e10;
int dcmp(double x)
{
    if(fabs(x)<eps) return 0;
    else return x<0?-1:1;
}
struct point
{
    double x,y;
    point(double x=0,double y=0):x(x),y(y){}
};
point operator-(point a,point b){return point(a.x-b.x,a.y-
b.y);}
point operator+(point a,point b){return point(a.x+b.x,a.y
+b.y);}
point operator*(point a,double p){return point(a.x*p,a.y*p);}
bool operator<(const point& a,const point& b)
{
    return a.x<b.x||(a.x==b.x&&a.y<b.y);
}
bool operator==(const point& a,const point& b)
{
    return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0;
}
double cross(point a,point b){return a.x*b.y-a.y*b.x;}
double dot(point a,point b){return a.x*b.x+a.y*b.y;}
double length(point a){return sqrt(dot(a,a));}
point normal(point a)
{
    double l=length(a);
    return point(a.x/l,a.y/l);
}
double distoline(point p,point a,point b)
{
    point v1=b-a,v2=p-a;
    return fabs(cross(v1,v2))/length(v1);
}
vector<point> p;
double mina,minp;
```

```
vector<point> convex(vector<point>& p)
{
    sort(p.begin(),p.end());
    int n=p.size();
    vector<point> ch(n+1);
    int m=0;
    for(int i=0;i<n;i++)
    {
        while(m>1&&cross(ch[m-1]-ch[m-2],p[i]-ch[m-2])<=0)
m--;
        ch[m++]=p[i];
    }
    int k=m;
    for(int i=n-2;i>=0;i--)
    {
        while(m>k&&cross(ch[m-1]-ch[m-2],p[i]-ch[m-2])<=0)
m--;
        ch[m++]=p[i];
    }
    if(n>1) m--;
    ch.resize(m);
    return ch;
}
void rotating_calipers(vector<point>& points)
{
    vector<point> p=convex(points);
    int n=p.size();
    p.push_back(p[0]);
    mina=inf;minp=inf;
    int l=1,r=1,u=1;
    for(int i=0;i<n;i++)
    {
        point edge=normal(p[(i+1)%n]-p[i]);
        while(dot(edge,p[r%n]-p[i])<dot(edge,p[(r+1)%n]-p[i]))
r++;
        while(u<r||cross(edge,p[u%n]-p[i])<cross(edge,p[(u
+1)%n]-p[i])) u++;
        while(l<u||dot(edge,p[l%n]-p[i])>dot(edge,p[(l+1)%n]-
p[i])) l++;
        double w=dot(edge,p[r%n]-p[i])-dot(edge,p[l%n]-p[i]);
        double h=distoline(p[u%n],p[i],p[(i+1)%n]);
        mina=min(mina,w*h);
        minp=min(minp,2*(w+h));
    }
}
int main()
{
    int n;
    while(~scanf("%d",&n))
    {
        if(!n) break;
        p.clear();
        for(int i=0;i<n;i++)
```

```
        {
            double x,y;
            scanf("%lf%lf",&x,&y);
            p.push_back(point(x,y));
        }
        rotating_calipers(p);
        printf("%.2f %.2f\n",mina,minp);
    }
    return 0;
    }
```

22、网络流SAP最大流

```
#include <iostream>
#include <cstdio>
#include <climits>
#include <cstring>
#include <algorithm>
using namespace std;
typedef  struct {int v,next,val;} edge;
const int MAXN=20010;
const int MAXM=500010;
edge e[MAXM];
int p[MAXN],eid;
inline void init(){memset(p,-1,sizeof(p));eid=0;}
//有向
inline void insert1(int from,int to,int val)
{
    e[eid].v=to;e[eid].val=val;
    e[eid].next=p[from];
    p[from]=eid++;
    swap(from,to);
    e[eid].v=to;e[eid].val=0;
    e[eid].next=p[from];
    p[from]=eid++;
}
//无向
inline void insert2(int from,int to,int val)
{
    e[eid].v=to;e[eid].val=val;
    e[eid].next=p[from];
    p[from]=eid++;
    swap(from,to);
    e[eid].v=to;e[eid].val=val;
    e[eid].next=p[from];
    p[from]=eid++;
}
int n,m;//n为点数 m为边数
int h[MAXN];
int gap[MAXN];
int source,sink;
inline int dfs(int pos,int cost)
{
    if (pos==sink) return cost;
```

```
    int j,minh=n-1,lv=cost,d;
    for (j=p[pos];j!=-1;j=e[j].next)
    {
        int v=e[j].v,val=e[j].val;
        if(val>0)
        {
            if (h[v]+1==h[pos])
            {
                if (lv<e[j].val) d=lv;
                else d=e[j].val;

                d=dfs(v,d);
                e[j].val-=d;
                e[j^1].val+=d;
                lv-=d;
                if (h[source]>=n) return cost-lv;
                if (lv==0) break;
            }
            if (h[v]<minh)    minh=h[v];
        }
    }
    if (lv==cost)
    {
        --gap[h[pos]];
        if (gap[h[pos]]==0) h[source]=n;
        h[pos]=minh+1;
        ++gap[h[pos]];
    }
    return cost-lv;
}

int isap(int st,int ed)
{

    source=st;sink=ed;
    int ret=0;
    memset(gap,0,sizeof(gap));
    memset(h,0,sizeof(h));
    gap[st]=n;
    while (h[st]<n)
    {
        ret+=dfs(st,INT_MAX);
    }
    return ret;
}
int main()
{
    while(cin>>m>>n)
    {
        init();
        for(int i=0;i<m;i++)
        {
            int u,v,c;
```

```
            scanf("%d%d%d",&u,&v,&c);
            insert1(u,v,c);
        }
        printf("%d\n",isap(1,n));
    }
    return 0;
    }
```

23、多重背包（单调队列优化）

```
#include<stdio.h>
#include<string.h>
#define MAXN 105
struct Queue
{
    int num,value;
}que[250005];
int head,tail;
int v[MAXN],w[MAXN],c[MAXN];
int dp[250005];
void enqueue (int x,int y)
{
    while (head<=tail && que[tail].value<y) tail--;
    que[++tail].num=x;que[tail].value=y;
}
int main()
{
    int i,j,d,sum,n,tempsum;
    while (scanf("%d",&n) && n>=0)
    {
        sum=0;
        for (i=1 ; i<=n ; ++i)
        {
            scanf("%d%d",&v[i],&c[i]);
            w[i]=v[i];
            sum+=w[i]*c[i];
        }
        tempsum=sum;
        sum/=2;
        for (i=1 ; i<=sum ; ++i) dp[i]=0;
        for (i=1 ; i<=n ; ++i)
        {
            if (c[i] > sum/w[i]) c[i]=sum/w[i];
            for (d=0 ; d<w[i] ; ++d)
            {
                head=1;tail=0;
                for (j=0 ; j<=(sum-d)/w[i] ; ++j)
                {
                    enqueue(j , dp[j*w[i]+d]-j*v[i]);
                    while (que[head].num<j-c[i] && head<=tail)
head++;
                    dp[j*w[i]+d]=que[head].value+j*v[i];
                }
            }
        }
```

```
        printf("%d %d/n",tempsum-dp[sum],dp[sum]);
    }
    return 0;
    }
```

24、硬币问题的（NV）模板

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<string.h>
using namespace std;
bool dp[100005];
int num[100005];
int a[105],b[105];
int main()
{
    int n,i,j,m,sum;
    double s;
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        for(i=0;i<n;i++)
            scanf("%d",&a[i]);
        for(i=0;i<n;i++)
            scanf("%d",&b[i]);
        memset(dp,0,sizeof(dp));
        sum=0;
        dp[0]=1;
        for(i=0;i<n;i++)
        {
            memset(num,0,(m+1)*sizeof(num[0]));
            for(j=a[i];j<=m;j++)
            {
                if(!dp[j]&&dp[j-a[i]]&&num[j-a[i]]<b[i])
                {
                    dp[j]=1;
                    num[j]=num[j-a[i]]+1;
                    sum++;
                }
            }
        }
        s=(double)sum/m;
        printf("%.3lf%%\n",s*100);
    }
    return 0;
    }
```

25、强连通分量（tarjan算法）

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
#define maxn 1105
using namespace std;
```

```
int
low[maxn],dnf[maxn],que[maxn],tear,head[maxn],c[maxn],mark[max
n];
bool map[maxn][maxn];
vector<int>g[maxn];
void init(int n){
    for(int i=1;i<=n;i++) g[i].clear();
}
int cal,tot;
void tarjin(int p){
    dnf[p]=low[p]=cal++;
    int i,k;
    k=g[p].size();
    que[tear++]=p;
    for(i=0;i<k;i++){
        int v=g[p][i];
        if(!dnf[v]){
            tarjin(v);
            low[p]=min(low[p],low[v]);//如果儿子就比较low
        }
        else if(!mark[v]){
            low[p]=min(low[p],dnf[v]);//与栈里面的点进行比较，如果
mark[i]有值就代表已经出栈了。
        }
    }
        if(low[p]==dnf[p]){
        while(tear>0){
            mark[que[tear-1]]=tot;
            if(que[tear-1]==p){
                tear--;
                break;
            }
            tear--;
        }
        tot++;
    }
}
void solve(int n){
    int i;
    tear=0;
    cal=1;
    tot=1;
    memset(dnf,0,sizeof(dnf));
    memset(c,0,sizeof(c));
    memset(mark,0,sizeof(mark));
    for(i=1;i<=n;i++){
        tear=0;
        if(!dnf[i]) tarjin(i);
    }
    for(i=1;i<=n;i++){
        c[mark[i]]++;
    }
 }
```

```
int main()
{
    int i,j,n,m,p,t;
    scanf("%d",&t);
    while(t--){
        scanf("%d%d",&n,&m);
        init(n);
        for(i=0;i<m;i++){
            scanf("%d%d",&p,&j);
            g[p].push_back(j);
        }
        solve(n);
    }
    }
```

26、欧拉错排公式

```
void init()
{
    s[0]=0; s[1]=0; s[2]=1;
    int i;
    for (i=3;i<=100;i++)
        s[i]=(i-1)*(s[i-1]+s[i-2])%mod;
    }
```

27、三分模板

```
//三分求极值法
//    二分法早就失去了他的意义了。不过还是可以用三分法来实现的，就是二分中再来二分。比如我们定义了L和R，m = (L + R) / 2，mm = (mid + R) / 2;如果mid靠近极值点，则R = mm；否则就是mm靠近极值点，则L = m;这样的话，极值还是可以求的
#include<stdio.h>
#include<iostream>
#include<algorithm>
using namespace std;
const double EPS = 1e-10;
double calc(double n)
{
    return;
}
double solve(double L, double R)
{
    double M, RM;
    while (L + EPS < R)
    {
        M = (L + R) / 2;
        RM = (M + R) / 2;
        if (calc(M) < calc(RM)) //计算最小值
            R = RM;
        else
            L = M;
    }
    return R;
    }
```

28、最小费用最大流模板

```
#include<stdio.h>
```

```
#include<string.h>
#include<algorithm>
#include<iostream>
#include<vector>
#include<queue>
#include<queue>
#define inf 0x3f
using namespace std;
struct pi
{
    int to;
    int cap;
    int cost;
    int rev;
};
vector<pi>g[205];
int dis[205];
int pre[205],pree[205];
int sink,source;
int vis[205];
int a[205];
char c[205];
void add(int a,int b,int cap,int cost)
{
    pi pp;
    pp.to=b;
    pp.cap=cap;
    pp.cost=cost;
    pp.rev=(int)g[b].size();
    g[a].push_back(pp);
    pp.to=a;
    pp.cap=0;
    pp.cost=-cost;
    pp.rev=(int)g[a].size()-1;
    g[b].push_back(pp);
    return ;
}
int spfa(void)
{
    int i,p,k;
    memset(dis,0x3f ,sizeof(dis));
    memset(vis,0,sizeof(vis));
    memset(pre,-1,sizeof(pre));
    memset(pree,-1,sizeof(pree));
    dis[source]=0;
    vis[source]=1;
    queue<int>q;
    pi pp;
    q.push(source);
    while(!q.empty())
    {
        p=q.front();
        k=(int)g[p].size();
```

```
            q.pop();
            vis[p]=0;
            if(p==sink)
                continue;
            for(i=0;i<k;i++)
            {
                pp=g[p][i];
                if(pp.cap>0&&dis[pp.to]>dis[p]+pp.cost)
                {
                    dis[pp.to]=dis[p]+pp.cost;
                    pre[pp.to]=p;
                    pree[pp.to]=i;
                    if(!vis[pp.to])
                    {
                        q.push(pp.to);
                        vis[pp.to]=1;
                    }
                }
            }
        }
        return pre[sink]!=-1;
    }
    int min(int a,int b)
    {
        int p;
        p=a;
        if(b<a)
            p=b;
        return p;
    }
    int minflow(void)
    {
        int i,f;
        int res=0;
        while(spfa())
        {
            f=inf;
            for(i=sink;i!=source;i=pre[i])
            {
                f=min(f,g[pre[i]][pree[i]].cap);
            }
            res+=f*dis[sink];
            for(i=sink;i!=source;i=pre[i])
            {
                pi &e=g[pre[i]][pree[i]];
                e.cap-=f;
                g[i][e.rev].cap+=f;
            }
        }
        return res;
    }
```

29、中国剩余定理模板

```
#include<stdio.h>
#include<math.h>
#include<algorithm>
using namespace std;
int x, y,q;
int exgcd(int a,int b)
{
    int p;
    if(b==0)
    {
        x=1;
        y=0;
        return a;
    }
    q=exgcd(b,a%b);
    p=x;
    x=-y;
    y=-p-(a/b)*y;
    return q;
}
int main()
{
    int p,e,i,d,count,q,t,f,k,N=0;
    while(1)
    {
        N++;
        scanf("%d%d%d%d",&p,&e,&i,&d);
        if(p==-1&&e==-1&&i==-1&&d==-1)
            break;
        if(p==0)
            p=23;
        if(e==0)
            e=28;
        if(i==0)
            i=33;
        q=23*28*33;
        count=0;
        t=28*33;
        f=23*33;
        k=23*28;
        exgcd(t,23);
        if(x<0)
            x=x%23+23;
        count+=p*x*t;
        exgcd(f,28);
        if(x<0)
            x=x%28+28;
        count+=f*x*e;
        exgcd(k,33);
        if(x<0)
            x=x%33+33;
        count+=k*x*i-d;
        count=count%q;
```

```
        if(count<=0)
            count+=q;
        printf("Case %d: the next triple peak occurs in %d
days.\n",N,count);
    }
    return 0;
    }
```

30、后缀数组（dc3）

```
/*
 一定要注意把几个串连起来的时候连接点千万别相同
 */
#include<cstdio>
#include<cstring>
#include<vector>
#include<algorithm>
#define maxn 200100
using namespace std;
int r[maxn];
int Rank[maxn],sa[maxn],height[maxn];
int wa[maxn],wb[maxn],wv[maxn],ws[maxn];
char a[maxn],b[maxn];
int cmp(int *r,int a,int b,int le)
{
    return r[a]==r[b]&&r[a+le]==r[b+le];
}
void da(int *r,int *sa,int n,int m)
{
    int i,j,p,*x=wa,*y=wb,*t;
    for ( i = 0; i < m; i++) ws[i]=0;
    for ( i = 0; i < n; i++) ws[ x[i] = r[i] ]++;
    for ( i = 1; i < m; i++) ws[i]+=ws[i-1];
    for ( i = n-1; i >= 0; i--) sa[--ws[x[i]]]=i;
    for ( j = 1,p=1; p <n ; j*=2,m=p)
    {
      for ( p = 0,i=n-j; i < n; i++) y[p++]=i;
      for ( i = 0; i < n; i++) if(sa[i]>=j) y[p++]=sa[i]-j;
      for ( i = 0; i < n; i++) wv[i]=x[y[i]];
      for ( i = 0; i < m; i++) ws[i]=0;
      for ( i = 0; i < n; i++) ws[wv[i]]++;
      for ( i = 1; i < m; i++) ws[i]+=ws[i-1];
      for ( i = n-1; i >= 0 ; i--) sa[--ws[wv[i]]]=y[i];
      for (t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)
         x[sa[i]] = cmp(y,sa[i-1],sa[i],j)?p-1:p++;
    }
    return ;
}
void calheight( int *r,int *sa,int n)
{
    int i,j,k=0;
    for ( i = 1; i <=n ; i++) Rank[sa[i]]=i;
    for(i=0;i<n;height[Rank[i++]]=k)
      for(k?k--:0,j=sa[Rank[i]-1];r[i+k]==r[j+k];k++);
    return ;
```

```
}
struct pi
{
    int min;
}pp[4*maxn];
void build(int le,int ri,int tot)
{
    if(le==ri)
    {
        pp[tot].min=height[le];
        return ;
    }
    int mid;
    mid=(le+ri)/2;
    build(le,mid,2*tot);
    build(mid+1,ri,2*tot+1);
    pp[tot].min=min(pp[2*tot].min,pp[2*tot+1].min);
    return ;
}
int query(int le,int ri,int tot,int ll,int rr)
{
    int p,q;
    p=100000000;
    q=100000000;
    if(le>ri)
        return 0;
    if(le<=ll&&ri>=rr)
    {
        return pp[tot].min;
    }
    int mid;
    mid=(ll+rr)/2;
    if(le<=mid)
    {
        p=query(le,ri,2*tot,ll,mid);
    }
    if(ri>mid)
    {
        q=query(le,ri,2*tot+1,mid+1,rr);
    }
    if(p==100000000&&q==100000000)
        return 0;
    return min(p,q);
}
int main()
{
    int i,n,k,f,le,ri,mid,p;
    while(scanf("%d%d",&n,&k)!=EOF)
    {
        for(i=0;i<n;i++)
        {
            scanf("%d",&f);
            r[i]=f+1;/*
```

```
                        一定要注意把几个串连起来的时候连接点千万别相同
                        */
        }
            r[n]=0;//为了使rank从1开始，防止height[rank[i]-1]越
界。
        da(r,sa,n+1,20002);
        calheight(r,sa,n);
        for(i=2;i<=n;i++)
            height[i];      //height[i]数组的定义sa[i-1]和sa[i]的
最长公共前缀，而rank从1到n,所以height数组从2到n。
    }
    return 0;
    }
```

31、双调欧几里得旅行商问题

```
//d[i][j]=d[i-1][j]+p[i][i-1];

//d[i][i-1]=min(d[i-1][j]+p[j][i]);
void work_p()
{
    for (int i=1;i<n;i++)
        for (int j=i+1;j<=n;j++)
            p[i][j]=p[j][i]=cnt(a[i].x,a[i].y,a[j].x,a[j].y);
}
void DP()
{
    d[1][1]=0;
    for (int i=2;i<=n;i++)
        d[i][1]=p[i][1];
    for (int i=2;i<n;i++)
    {
        d[i+1][i]=INT_MAX;
        for (int j=1;j<=i-1;j++)
        {
            d[i+1][j]=d[i][j]+p[i][i+1];
            d[i+1][i]=min(d[i+1][i],d[i][j]+p[j][i+1]);
        }
    }
    }
```

32、约瑟夫问题

```
/*
 f[1]=0;
 f[i]=(f[i-1]+m)%i;  (i>1)
*/
#include <stdio.h>
 int main() {
      int n, m, i, s = 0;
         printf ("N M = ");
         scanf("%d%d", &n, &m);
         for (i = 2; i <= n; i++)
             {
                     s = (s + m) % i;
                }
         printf ("\nThe winner is %d\n", s+1);
```

```
        }
```

33、回文串的hash解法

```
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <iostream>
#include<algorithm>
#define maxn 1000010
#define inf 31
using namespace std;
char s[maxn];
int  N;
unsigned long long bit[maxn],f[maxn],t[maxn];
void init(void) {
    bit[0] = 1;
    for (int i = 1; i <=maxn-10;i++) {
        bit[i] = (unsigned long long)bit[i-1]*inf;
    }
}
int low(int x){
    return x & -x;
}
void add(int x, unsigned long long c[], unsigned long long
val) {
    for (int i = x; i <= N; i += low(i)) {
        c[i] += val;
    }
}
unsigned long long sum(int x, unsigned long long c[]) {
    unsigned long long ret = 0;
    for (int i = x; i > 0; i -= low(i)) {
        ret += c[i];
    }
    return ret;
}
int panduan(int a, int b) {
    int x = a-1, y = N-b, z;
    z = max(x, y);
    unsigned long long ll = sum(b, f) - sum(a-1, f);
    unsigned long long rr = sum(N-a+1, t) - sum(N-b, t);
    ll *= bit[z-x];
    rr *= bit[z-y];
    return ll == rr;
}
void merg(int x, int val) {
    add(x, f, (val-s[x])*bit[x-1]);
    add(N+1-x, t, (val-s[x])*bit[N-x]);
    s[x] = val;
}
int main() {
    char cc[5], ch[5];
    int a, b,Q;
    init();
```

```
    while (scanf("%s", s+1)!=EOF) {
        N = (int)strlen(s+1);
        memset(f, 0, sizeof (long long) * (N+1));
        memset(t, 0, sizeof (long long) * (N+1));
        for (int i = 1, j = N; i <= N; ++i, --j) {
            s[i] -= 'a';
            add(i, f, s[i]*bit[i-1]);
            add(j, t, s[i]*bit[j-1]);
        }
        scanf("%d", &Q);
        for(int i=0;i<Q;i++) {
            scanf("%s", cc);
            if (cc[0] == 'Q') {
                scanf("%d %d", &a, &b);
                if(panduan(a, b)){
                    printf("Yes\n");
                }
                else{
                    printf("No\n");
                }
            } else {
                scanf("%d %s", &a, ch);
                merg(a, ch[0]-'a');
            }
        }
        printf("\n");
    }
    return 0;
    }
```

34、在可解时间之内判断回文串算法（非后缀数组）

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<string.h>
#define maxn 5005//几乎是极限
using namespace std;
char a[5005];
int pan[5005][5005];
int main()
{
    int n,i,k;
    cin>>a;
    n=strlen(a);
    for(i=0;i<n;i++)
        pan[i][i]=1;
    for(k=2;k<=n;k++){
        for(i=0;i<n;i++){
            if(i+k-1>=n)
                break;
            if(k==2&&a[i]==a[i+k-1])
                pan[i][i+k-1]=1;
            else if(k>2&&a[i]==a[i+k-1]&&pan[i+1][i+k-2])
                pan[i][i+k-1]=1;
```

```
        }
    }
}
```

35、数据输入加速

```
template <class T>
inline 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();
}
```

36、字典树

```
#include <stdio.h>
#include <string.h>
#define MAX(a,b) ((a)>(b)?(a):(b))
#define NODE 3200010
#define N 100010
int n;
__int64 v[N];
__int64 node;
__int64 next[NODE][2];
__int64 end[NODE];
void add(__int64 cur,__int64 k)
{
    memset(next[node],0,sizeof(next[node]));
    end[node]=0;
    next[cur][k]=node++;
}
__int64 cal(__int64 x)
{
    int i;
    __int64 cur=0;
    __int64 k;
    for(i=32;i>=0;i--)
    {
        k=(((__int64)1<<i)&x)?0:1;
        if(next[cur][k]) cur=next[cur][k];
        else    cur=next[cur][1-k];
    }
    return (x^end[cur]);
}
int main()
{
    int  i,j,k,t,NN,m;
    __int64 cur ,p,x;
    scanf("%d",&t);
    NN=t;
    while(t--)
    {
        scanf("%d%d",&n,&m);
        node=1;
        memset(next[0],0,sizeof(next[0]));
```

```
        for(i=0;i<n;i++)
        {
            scanf("%I64d",&x);
            v[i]=x;
            cur=0;
            for(j=32;j>=0;j--)
            {
                k=(((__int64)1<<j)&x)?1:0;
                if(next[cur][k]==0) add(cur,k);
                cur=next[cur][k];
            }
            end[cur]=x;
        }
        printf("Case #%d:\n",NN-t);
        for(i=0;i<m;i++)
        {
            scanf("%I64d",&p);
            printf("%I64d\n",cal(p)^p);
        }
    }
    return 0;
    }
```

37、平面的划分

```
/*已知经过同一个的n个平面，任意三个平面不经过同一条直线，若这n个平面将空间分成f（n）个部分，则f（3）= f(n)=
 (1)、f(3)=8，见上图。
 (2)、当n>3时，每增加一个面，这面就要与前面n-1个面都相交，因为过同一点，两平面如果有一个公共点就有一条公共直线，这样就会把前面平面划分的空间一分为二，f(n)-f(n-1)=2(n-1),然后累加得f(n)=n^2-n+2*/
```

38、高斯消元（解方程）

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cmath>
using namespace  std;
double b[205][205],x[205],a[25][25];
void guass(int equ,int val){
    int i,j,n,p ,k;
    double s2,q;
    p=0;
    for(i=0;i<equ&&p<val;i++,p++){
        n=i;
        for(j=i+1;j<equ;j++){
            if(fabs(b[j][p])>fabs(b[n][p])) n=j;
        }
        if(n!=i){
            for(j=p;j<val;j++){
                swap(b[i][j],b[n][j]);
            }
            swap(x[i],x[n]);
        }
        if(fabs(b[i][p])<1e-6){
            i--;
```

```
            continue;
        }
        for(j=i+1;j<equ;j++){
            if(fabs(b[j][p])<1e-9) continue;
            q=b[j][p]/b[i][p];
            for(k=p;k<val;k++){
                b[j][k]=q*b[i][k]-b[j][k];
            }
            x[j]=q*x[i]-x[j];
        }
    }
    for(i=equ-1;i>=0;i--){
        s2=x[i];
        for(j=val-1;j>i;j--) s2-=b[i][j]*x[j];
        x[i]=s2/b[i][i];
    }
}
int aabs(int p,int f){
    if(f>p) return f-p;
    return p-f;
}
int main()
{
    int i,j,n,m,p,k,f,v,N=0;
    while(1){
        scanf("%d%d%d",&m,&n,&p);
        if(m==0&&n==0&&p==0) break;
        if(N!=0) printf("\n");
        N++;
        for(i=0;i<n;i++){
            for(j=0;j<m;j++)
                scanf("%lf",&a[i][j]);
        }
        memset(b,0,sizeof(b));
        for(i=0;i<n;i++){
            for(j=0;j<m;j++){
                v=0;
                for(f=0;f<n;f++){
                    for(k=0;k<m;k++){
                        if(aabs(i,f)+aabs(j,k)<=p){
                            b[i*m+j][f*m+k]=1;
                            v++;
                        }
                    }
                }
                x[i*m+j]=v*a[i][j];
            }
        }
        guass(n*m,m*n);
        for(i=0;i<n;i++){
            for(j=0;j<m;j++){
                printf("%8.2lf",x[i*m+j]);
            }
```

```
            printf("\n");
        }
    }
    }
```

39、高斯消元 开关类问题（异或方程）

```
#include<cstdio>
#include<algorithm>
#include<cstring>
#define LL __int64
using namespace std;
int b[55][55],x[55],c[55][55];
int flag;//判断是否无解。
int guass(int equ,int val){
    int i,j,n,m,p,k,q;
    p=0;
    for(i=0;i<equ&&p<val;i++,p++){
        k=b[i][p];
        n=i;
        for(j=i+1;j<equ;j++){
            if(b[j][p]>k) {k=b[j][p];
                n=j;
            }
        }
        if(n!=i){
            for(j=p;j<val;j++){
                swap(b[i][j],b[n][j]);
            }
            swap(x[i],x[n]);
        }
        if(b[i][p]==0) {
            i--;
            continue;
        }
        for(j=i+1;j<equ;j++){
            if(b[j][p]==0) continue;
            for(m=p;m<val;m++){
                b[j][m]=b[i][m]^b[j][m];
            }
            x[j]=x[i]^x[j];
        }
    }
    q=i;
    m=0;
    for(i=0;i<equ;i++){
        p=0;
        for(j=0;j<val;j++){
            if(b[i][j]!=0){
                p=1;
                break;
            }
        }
        if(p==0&&x[i]!=0){
            flag=1;
```

```
            return 0;
        }
        if(p) m++;
    }
    return val-q;//返回不确定变元个数
     }
```

40、四边形优化模板

```
/*
对于dp[i][j]=dp[i][k]+d[k][j]+w[i][j]的dp方程，如果满足w[i][j]
+w[i'][j']<=w[i'][j]+w[i][j']    (i'<=i<=j<=j')则w[i][j]是凸的，
也就是说，对于dp[i][j]的决策s[i][j],必然满足不等式s[i][j-1]<=s[i]
[j]<=s[i+1][j].所以求决策时只需要循环从s[i][j-1]到s[i+1][j]就行，然
后求s[i][j],注意循环长度。
 区间dp一般用四边形优化
 */
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
int dp[1005][1005];
int s[1005][1005];
struct pi{
    int x;
    int y;
}pp[1005];
int main()
{
    int i,j,n,m,k;
    while(scanf("%d",&n)!=EOF){
        memset(dp,0x3f,sizeof(dp));
        memset(s,0,sizeof(s));
        for(i=1;i<=n;i++) {scanf("%d%d",&pp[i].x,&pp[i].y);
            s[i][i]=i;
            dp[i][i]=0;
        }
        for(i=1;i<=n-1;i++){//四边形优化一定要优化长度
            for(j=1;j+i<=n;j++){
                for(k=s[j][j+i-1];k<=s[j+1][j+i];k++){
                    m=pp[k].y-pp[j+i].y+pp[k+1].x-pp[j].x;
                    if(dp[j][j+i]>dp[j][k]+dp[k+1][j+i]+m){
                        dp[j][j+i]=dp[j][k]+dp[k+1][j+i]+m;
                        s[j][j+i]=k;
                    }
                }
            }
        }
        printf("%d\n",dp[1][n]);
    }
}
```

41、斜率dp模板

```
/*
```

我们假设k<j<i。如果在j的时候决策要比在k的时候决策好，那么也是就是dp[j]+M+(sum[i]-sum[j])^2<dp[k]+M+(sum[i]-sum[k])^2。(因为是最小花费嘛，所以优就是小于)

两边移项一下，得到：(dp[j]+num[j]^2-(dp[k]+num[k]^2))/(2*(num[j]-num[k]))<sum[i]。我们把dp[j]-num[j]^2看做是yj，把2*num[j]看成是xj。

那么不就是yj-yk/xj-xk<sum[i]么？　　左边是不是斜率的表示?

那么yj-yk/xj-xk<sum[i]说明了什么呢？　我们前面是不是假设j的决策比k的决策要好才得到这个表示的？　如果是的话，那么就说明g[j,k]=yj-jk/xj-xk<sum[i]代表这j的决策比k的决策要更优。

关键的来了：现在从左到右，还是设k<j<i，如果g[i,j]<g[j,k]，那么j点便永远不可能成为最优解，可以直接将它踢出我们的最优解集。为什么呢?

我们假设g[i,j]<sum[i]，那么就是说i点要比j点优，排除j点。

如果g[i,j]>=sum[i]，那么j点此时是比i点要更优，但是同时g[j,k]>g[i,j]>sum[i]。这说明还有k点会比j点更优，同样排除j点。

排除多余的点，这便是一种优化!

接下来看看如何找最优解。

设k<j<i。

由于我们排除了g[i,j]<=g[j,k]的情况，所以整个有效点集呈现一种上凸性质，即k j的斜率要大于j i的斜率。

这样，从左到右，斜率之间就是单调递减的了。当我们的最优解取得在j点的时候，那么k点不可能再取得比j点更优的解了，于是k点也可以排除。换句话说，j点之前的点全部不可能再比j点更优了，可以全部从解集中排除。

于是对于这题我们对于斜率优化做法可以总结如下:

1，用一个单调队列来维护解集。

2，假设队列中从头到尾已经有元素a b c。那么当d要入队的时候，我们维护队列的上凸性质，即如果g[d,c]<g[c,b]，那么就将c点删除。直到找到g[d,x]>=g[x,y]为止，并将d点加入在该位置中。

3，求解时候，从队头开始，如果已有元素a b c，当i点要求解时，如果g[b,a]<sum[i]，那么说明b点比a点更优，a点可以排除，于是a出队。最后dp[i]=getDp(q[head])。

*/

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cmath>
using namespace std;
typedef int LL;
LL a[500005];
LL dp[500005];
int q[500005];
LL s[500005];
LL get1(int n,int m){
    return dp[n]-dp[m]+s[n]*s[n]-s[m]*s[m];
}
LL get2(int n,int m){
    return s[n]-s[m];
}
int main()
{
    int i,n,m,tear,rear;
    while(scanf("%d%d",&n,&m)!=EOF){
        s[0]=0;
        dp[0]=0;
        for(i=1;i<=n;i++){
            scanf("%d",&a[i]);
            s[i]=s[i-1]+a[i];
        }
        rear=0;
        tear=0;
        q[tear++]=0;
        for(i=1;i<=n;i++){
            while(rear+1<tear){
                if(get1(q[rear+1],q[rear])<=2*s[i]*get2(q[rear
+1],q[rear])) rear++;
                else break;
            }
            dp[i]=dp[q[rear]]+(s[i]-s[q[rear]])*(s[i]-
s[q[rear]])+m;
            while(tear>rear+1){

if(get1(i,q[tear-1])*get2(q[tear-1],q[tear-2])<=get1(q[tear-1]
,q[tear-2])*get2(i,q[tear-1])) tear--;
                else break;
            }
            q[tear++]=i;
        }
        printf("%d\n",dp[n]);
    }
}
```

42、分解大数质因数模板

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
```

```
#include<time.h>
#include<iostream>
#include<algorithm>
#define LL long long
using namespace std;
//
************************************************************************
**
// Miller_Rabin 算法进行素数测试
//速度快，而且可以判断 <2^63的数
//
************************************************************************
**
const int S=20;//随机算法判定次数，S越大，判错概率越小
//计算 (a*b)%c.   a,b都是long long的数，直接相乘可能溢出的
//  a,b,c <2^63
LL mult_mod(LL a,LL b,LL c)
{
    a%=c;
    b%=c;
    LL ret=0;
    while(b)
    {
        if(b&1){ret+=a;ret%=c;}
        a<<=1;
        if(a>=c)a%=c;
        b>>=1;
    }
    return ret;
}
//计算  x^n %c
LL pow_mod(LL x,LL n,LL mod)//x^n%c
{
    if(n==1)return x%mod;
    x%=mod;
    LL tmp=x;
    LL ret=1;
    while(n)
    {
        if(n&1) ret=mult_mod(ret,tmp,mod);
        tmp=mult_mod(tmp,tmp,mod);
        n>>=1;
    }
    return ret;
}
//以a为基,n-1=x*2^t      a^(n-1)=1(mod n)  验证n是不是合数
//一定是合数返回true,不一定返回false
bool check(LL a,LL n,LL x,LL t)
{
    LL ret=pow_mod(a,x,n);
    LL last=ret;
    for(int i=1;i<=t;i++)
    {
```

```
        ret=mult_mod(ret,ret,n);
        if(ret==1&&last!=1&&last!=n-1) return true;//合数
        last=ret;
    }
    if(ret!=1) return true;
    return false;
}

// Miller_Rabin()算法素数判定
//是素数返回true.(可能是伪素数，但概率极小)
//合数返回false;

bool Miller_Rabin(LL n)
{
    if(n<2)return false;
    if(n==2)return true;
    if((n&1)==0) return false;//偶数
    LL x=n-1;
    LL t=0;
    while((x&1)==0){x>>=1;t++;}
    for(int i=0;i<S;i++)
    {
        LL a=rand()%(n-1)+1;//rand()需要stdlib.h头文件
        if(check(a,n,x,t))
            return false;//合数
    }
    return true;
}
//****************************************************
//pollard_rho 算法进行质因数分解
//****************************************************
LL factor[100];//质因数分解结果（刚返回时是无序的）
int tol;//质因数的个数。数组小标从0开始

LL gcd(LL a,LL b)
{
    if(a==0)return 1;//???????
    if(a<0) return gcd(-a,b);
    while(b)
    {
        LL t=a%b;
        a=b;
        b=t;
    }
    return a;
}
LL Pollard_rho(LL x,LL c)
{
    LL i=1,k=2;
    LL x0=rand()%x;
    LL y=x0;
    while(1)
    {
```

```
            i++;
            x0=(mult_mod(x0,x0,x)+c)%x;
            LL d=gcd(y-x0,x);
            if(d!=1&&d!=x) return d;
            if(y==x0) return x;
            if(i==k){y=x0;k+=k;}
        }
    }
    //对n进行素因子分解
    void findfac(LL n)
    {
        if(Miller_Rabin(n))//素数
        {
            factor[tol++]=n;
            return;
        }
        LL p=n;
        while(p>=n)p=Pollard_rho(p,rand()%(n-1)+1);
        findfac(p);
        findfac(n/p);
    }
    int main()
    {
        long long n;
        int t;
        cin>>t;
        while(t--){
            cin>>n;
            tol=0;
        findfac(n);
        if(tol==1){
            printf("Prime\n");
            continue;
        }
        sort(factor,factor+tol);
            cout<<factor[0]<<endl;
        }
    }
```

43、浮点数高精度

```
//用log取对数，然后用exp取次方
```

44、强连通分量（桥）

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
#define maxn 1005
using namespace std;
int vis[maxn],low[maxn],dnf[maxn],yes[maxn],fa[maxn];
vector<int > g[maxn];
int map[1005][1005];
void init(int n){
    for(int i=1;i<=n;i++) g[i].clear();
}
```

```
int zi[maxn],pp;
int cal;
void tarjin(int p,int pa,int n){
    int k,i;
    low[p]=dnf[p]=cal++;
    k=g[p].size();
    for(i=0;i<k;i++){
        int v=g[p][i];
        if(!dnf[v]){
            tarjin(v,p,n);
            low[p]=min(low[p],low[v]);
            if(low[v]>dnf[p]){//判断是否是桥
                pp++;//桥的个数
            }
        }
        else if(v!=pa){//一定要注意求桥时不能遍历父亲节点，因为判断条
件是大于，否则变成了环。
            low[p]=min(low[p],dnf[v]);
        }
    }
}
void solve(int n){
    int i,m;
    cal=1;
    for(i=1;i<=n;i++){
        if(!dnf[i])
            tarjin(i,-1,i);
    }
}
```

45、强连通分量（割点）

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
#define maxn 1005
using namespace std;
int vis[maxn],low[maxn],dnf[maxn],yes[maxn],fa[maxn];
vector<int > g[maxn];
int map[1005][1005];
void init(int n){
    for(int i=1;i<=n;i++) g[i].clear();
}
int zi[maxn],pp;
int cal;
void tarjin(int p,int pa,int n){
    int k,i,q=0;
    low[p]=dnf[p]=cal++;
    k=g[p].size();
    for(i=0;i<k;i++){
        int v=g[p][i];
        if(!dnf[v]){
            tarjin(v,p,n);
```

```
                low[p]=min(low[p],low[v]);
                if(low[v]>=low[p]){
                    yes[p]=1;//表示p是割点，可以包含p的环
                }
                q++;
            }
            else if(v!=pa){
                low[p]=min(low[p],dnf[v]);
            }
        }
        if(p==n) zi[p]=q;//根节点不加1.
        else zi[p]=q+1;//有多个分子图，也就是连着几块联通块。
}
void solve(int n){
        int i,m;
        memset(zi,0,sizeof(zi));
        cal=1;
        for(i=1;i<=n;i++){
        if(!dnf[i])
        tarjin(i,-1,i);
        }
}
```

46、布丰投针问题

```
/*间距为D的平行直线，一根长为l的针（l<D),针与直线相交的概率是2*l/
(pai*d).
如果是一个凸包，则概率是C/(pai*d),即为sigma(2*l/(pai*d))/2.因为直线与
凸包相交必然与两条边相交，所以概率除以2.
*/
```

47、调和级数求和

```
#include<cstdio>
#include<cstring>
#include<cmath>
#include<iostream>
#include<algorithm>
using namespace  std;
int main()
{
        int i,n;
        double s;
        while(cin>>n){
            if(n<=1000000){
                s=0;
                for(i=1;i<=n;i++) s+=1.0/i;
            }
            else{
                s=0.5772156649+log(n+1.0);
            }
            printf("%.4lf\n",s);
        }
}
```

48、网络流判圈

```
#include<stdio.h>
#include<iostream>
```

```
#include<algorithm>
#include<vector>
#include<string.h>
#include<queue>
#define inf 100000000
#define LL long long
#define maxn 820
using namespace std;
struct pi
{
    int to;
    int cost;
    int rev;
}pp;
vector<pi >g[maxn];
queue<int>q;
int c[450][450];
int s,t;
int line[maxn],leve[maxn];
bool vis[452][452];
int min(int a,int b)
{
    int p;
    p=a;
    if(b<a)
        p=b;
    return p;
}
void add(int a,int b,int cos)
{
    pp.to=b;
    pp.cost=cos;
    pp.rev=(int)g[b].size();
    g[a].push_back(pp);
    pp.to=a;
    pp.cost=0;
    pp.rev=(int)g[a].size()-1;
    g[b].push_back(pp);
    return ;
}
void bfs(void)
{
    q.push(s);
    int p,f,i;
    while(!q.empty())
    {
        p=q.front();
        q.pop();
        f=(int)g[p].size();
        for(i=0;i<f;i++)
        {
            pi &e=g[p][i];
            if(e.cost>0&&line[e.to]<0)
```

```
            {
                line[e.to]=line[p]+1;
                q.push(e.to);
            }
        }
    }
    return ;
}
int dfs(int v,int f)
{
    int p;
    if(v==t)
        return f;
    for(int &i=leve[v];i<g[v].size();i++)
    {
        pi &e=g[v][i];
        if(line[v]<line[e.to]&&e.cost>0)
        {
            p=dfs(e.to,min(f,e.cost));
            if(p>0)
            {
                e.cost-=p;
                g[e.to][e.rev].cost+=p;
                return p;
            }
        }
    }
    return 0;
}
long long  dinic()
{
    int f;
    long long flow=0;
    while(1)
    {
        memset(line,-1,sizeof(line));
        memset(leve,0,sizeof(leve));
        line[s]=0;
        bfs();
        if(line[t]<0)
            return flow;
        f=dfs(s,inf);
        if(f==0)
            continue;
        flow+=f;
        while((f=dfs(s,inf))>0)
        {
            flow+=f;
        }
    }
}
int main()
{
```

```
int i,j,n,m,p,k,f,f1,f2;
LL flow,s1,s2;
while(scanf("%d%d%d",&n,&m,&k)!=EOF){
    for(i=0;i<=n+m+1;i++) g[i].clear();
    s1=0;
    s2=0;
    f=0;
    for(i=1;i<=n;i++){
        scanf("%d",&p);
        add(0,i,p);
        if(p>k*m||p<0){
            f=1;
        }
        s1+=p;
    }
    for(i=1;i<=m;i++){
        scanf("%d",&p);
        if(p>k*n||p<0){
            f=1;
        }
        add(n+i,n+m+1,p);
        s2+=p;
    }
    if(f){
        printf("Impossible\n");
        continue;
    }
    if(s1!=s2){
        printf("Impossible\n");
        continue;
    }
    for(i=1;i<=n;i++){
        for(j=1;j<=m;j++){
            add(i,n+j,k);
        }
    }
    s=0;
    t=n+m+1;
    flow=dinic();
    if(flow!=s1){
        printf("Impossible\n");
        continue;
    }
    for(i=1;i<=n;i++){
        p=(int)g[i].size();
        for(j=0;j<p;j++){
            pp=g[i][j];
            if(pp.to>n&&pp.to<=n+m)
                c[i][pp.to-n]=pp.cost;//用残量网络判断是否含环
        }
    }
    memset(vis,0,sizeof(vis));//如果残量网络有两行全未达到极值说明图中流量不唯一，也就是说明图中含环
```

```
            int flag=0;
            for(i=1;i<=n;i++){
                for(j=1;j<=m;j++){
                    for(p=j+1;p<=m;p++){
                        f1=0,f2=0;
                        if(c[i][j]!=0&&c[i][p]!=k){
                            if(vis[p][j]){
                                flag=1;
                                break;
                            }
                            f1=1;
                        }
                        if(flag) break;
                        if(c[i][j]!=k&&c[i][p]!=0){
                            if(vis[j][p]){
                                flag=1;
                                break;
                            }
                            f2=1;
                        }
                        if(f1) vis[j][p]=1;
                        if(f2) vis[p][j]=1;
                    }
                    if(flag) break;
                }
                if(flag) break;
            }
            if(flag){
                printf("Not Unique\n");
            }
            else{
                printf("Unique\n");
                for(i=1;i<=n;i++){
                    for(j=1;j<=m;j++){
                        if(j==1){
                            printf("%d",k-c[i][j]);
                        }
                        else{
                            printf(" %d",k-c[i][j]);
                        }
                    }
                    printf("\n");
                }
            }
        }
    }
```

49、set函数

```
#include<cstdio>
#include<cstring>
#include<set>
#include<algorithm>
using namespace std;
struct pp{
```

```
    int id;
    int pi;
};
struct cmp{
    bool operator()(const pp& a,const pp &b)const{
        return a.pi<b.pi;
    }
};
pp ppi;
set<pp,cmp>q;
set<pp,cmp>::iterator it;
int main()
{
    int n,p,k;
    while(1){
        scanf("%d",&n);
        if(!n) break;
        if((n==2||n==3)&&q.size()==0){
            printf("0\n");
            continue;
        }
        if(n==1){
            scanf("%d%d",&k,&p);
            ppi.id=k;
            ppi.pi=p;
            q.insert(ppi);
            continue;
        }
        if(n==2){
            it=q.end();
            it--;
            printf("%d\n",it->id);
            q.erase(*it);
        }
        else{
            it=q.begin();
            printf("%d\n",it->id);
            q.erase(*it);
        }
    }
    return 0;
}
```

50、无源无汇上下限网络流

```
/*一种方法是 添加附加源汇S,T  对于某点 u, 设 M(u)=sigma(B[i,u])-
sigma(B[u,j]) ，
 则根据流量平衡条件有 M(u)同时等于 sigma(g[u,j])-sigma(g[i,u])
 若M(u)<0，即sigma(g[u,j]) < sigma(g[i,u]) 进入u的流量比从u 出去的
多，
 所以 u -> T 连容量为  -(sigma(B[i,u])-sigma(B[u,j]) ) 的边
 同理. M(u)>0时，即 S->u 连容量为 sigma(B[i,u])-sigma(B[u,j])  的
边.
 然后再 对于任意边(i,u)/(u,j) 连一条 C[u,v]-B[u,v]的边.
```

```
 这样 只需对新的网络求一遍最大流即可. 若出附加源点的边都满流即是存在可行流，反之不然.
 满流的必要条件是显然的. 不满流不能保证加上B[,]后流量平衡. 前面都白费了.
*/
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<vector>
#include<string.h>
#include<queue>
#define LL int
#define inf 100000000
#define maxn 1500
using namespace std;
struct pi
{
    int to;
    int cost;
    int rev;
    int id;
}pp;
vector<pi >g[maxn];
queue<int>q;
int s,t;
int line[maxn],leve[maxn];
int min(int a,int b)
{
    int p;
    p=a;
    if(b<a)
        p=b;
    return p;
}
void add(int a,int b,int cos,int id)
{
    pp.to=b;
    pp.cost=cos;
    pp.id=id;
    pp.rev=(int)g[b].size();
    g[a].push_back(pp);
    pp.to=a;
    pp.cost=0;
    pp.id=-1;
    pp.rev=(int)g[a].size()-1;
    g[b].push_back(pp);
    return ;
}
void bfs(void)
{
    q.push(s);
    int p,f,i;
    while(!q.empty())
    {
```

```
        p=q.front();
        q.pop();
        f=(int)g[p].size();
        for(i=0;i<f;i++)
        {
            pi &e=g[p][i];
            if(e.cost>0&&line[e.to]<0)
            {
                line[e.to]=line[p]+1;
                q.push(e.to);
            }
        }
    }
    return ;
}
int dfs(int v,int f)
{
    int p;
    if(v==t)
        return f;
    for(int &i=leve[v];i<g[v].size();i++)
    {
        pi &e=g[v][i];
        if(line[v]<line[e.to]&&e.cost>0)
        {
            p=dfs(e.to,min(f,e.cost));
            if(p>0)
            {
                e.cost-=p;
                g[e.to][e.rev].cost+=p;
                return p;
            }
        }
    }
    return 0;
}
LL  dinic()
{
    int f;
    LL flow=0;
    while(1)
    {
        memset(line,-1,sizeof(line));
        memset(leve,0,sizeof(leve));
        line[s]=0;
        bfs();
        if(line[t]<0)
            return flow;
        f=dfs(s,inf);
        if(f==0)
            continue;
        flow+=f;
        while((f=dfs(s,inf))>0)
```

```
        {
            flow+=f;
        }
    }
}
int a[20005];
struct ppi{
    int x;
    int y;
    int co1;
    int co2;
}pp1[100005];
int main()
{
    int i,j,n,m,p,k,f,q;
    s=0;
    while(scanf("%d%d",&n,&m)!=EOF){
        t=n+1;
        for(i=0;i<=n+1;i++) g[i].clear();
        for(i=0;i<m;i++){
            scanf("%d%d%d
%d",&pp1[i].x,&pp1[i].y,&pp1[i].co1,&pp1[i].co2);
            add(pp1[i].x,pp1[i].y,pp1[i].co2-pp1[i].co1,i);
            a[pp1[i].x]+=pp1[i].co1;
            a[pp1[i].y]-=pp1[i].co1;
        }
        for(i=1;i<=n;i++){
            if(a[i]>0) add(i,t,a[i],0);
            else add(s,i,-a[i],0);
        }//必要流量出流量比较多的与T连边，入的多的与S连边，跑一遍最大
流。
        p=dinic();
        memset(a,0,sizeof(a));
        k=0;
        f=g[0].size();
        for(i=0;i<f;i++){//如果与源点连的边都满流则为可行流，否则不可
行。
            if(g[0][i].cost!=0){
                k=1;
                break;
            }
        }
        if(k) printf("NO\n");
        else{
            for(i=1;i<=n;i++){
                k=g[i].size();
                for(j=0;j<k;j++){
                    if(g[i][j].id!=-1&&g[i][j].to!=s&&g[i]
[j].to!=t){
                        a[g[i][j].id]=pp1[g[i][j].id].co2-g[i]
[j].cost;//基础流量加上必要流量，注意网络里这条边的流量就是除基础流量外的
跑的流量。
                    }
                }
```

```
            }
            printf("YES\n");
            for(i=0;i<m;i++){
                printf("%d\n",a[i]);
            }
        }
    }
}
```

51、匈牙利算法

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
#include<vector>
#define maxn 1500
using namespace std;
int match[maxn];
vector<int >g[maxn];
bool use[maxn];
bool dfs(int u){
    int p,i,j,n,k;
    use[u]=1;
    n=g[u].size();
    for(i=0;i<n;i++){
        k=g[u][i];
        p=match[k];
        if(p<0||!use[p]&&dfs(p)){
            match[u]=g[u][i];
            match[g[u][i]]=u;
            return 1;
        }
    }
    return 0;
}
int hungry(int n){
    int m=0,i,j;
    memset(match,-1,sizeof(match));
    for(i=0;i<n;i++){
        if(match[i]<0){
            memset(use,0,sizeof(use));
            if(dfs(i)){
                m++;
            }
        }
    }
    return m;
}
int main()
{
    int i,j,n,m,p,k;
}
```

52、压位高精度系统

```
#include<stdio.h>
```

```
#include<stdlib.h>
#include<string.h>
#define mm 100000000
#define LL long long
char a1[10005],b1[10005];
int x[1300],y[1300],z[1300],jie[10005]
[1250],v[1500],u[1500],w[1500];
int d[10005];
int f1,f2;
int max(int a,int b){
    if(a<b) return b;
    return a;
}
int plus(int *a,int *b,int *c,int n,int m){
    int i,j;
    if(n<m){
        for(i=n+1;i<=m;i++) a[i]=0;
    }
    else{
        for(i=m+1;i<=n;i++) b[i]=0;
    }
    memset(z,0,(max(n,m)+2)*sizeof(z[0]));
    for(i=0;i<=max(n,m);i++){
        z[i]+=b[i]+a[i];
        if(z[i]>=mm){
            z[i+1]+=z[i]/mm;
            z[i]=z[i]%mm;
        }
    }
    j=max(n,m)+1;
    while(z[j]>=mm){
        z[j+1]+=z[j]/mm;
        z[j]%=mm;
        j++;
    }
    while(j>=0&&z[j]==0) j--;
    if(j<0) j=0;
    for(i=0;i<=j;i++) c[i]=z[i];
    return j;
}
int minus(int *a,int *b,int *c,int n,int m){//必须a>=b
    int i,p;
    memset(z,0,(n+2)*sizeof(z[0]));
    for(i=m+1;i<=n;i++) b[i]=0;
    for(i=0;i<=n;i++){
        z[i]+=a[i]-b[i];
        if(z[i]<0){
            z[i+1]--;
            z[i]+=mm;
        }
    }
    p=n;
    while(p>=0&&z[p]==0) p--;
```

```
    if(p<0) p=0;
    for(i=0;i<=p;i++) c[i]=z[i];
    return p;
}
int mul(int *a,int p,int *c,int n){
    int i,m;
    LL s1;
    memset(z,0,(n+2)*sizeof(z[0]));
    for(i=0;i<=n;i++){
        s1=(LL)a[i]*p+z[i];
        if(s1>=mm){
            z[i+1]+=s1/mm;
            z[i]=s1%mm;
        }
        else z[i]=s1;
    }
    m=n+1;
    while(z[m]>=mm){
        z[m+1]+=z[m]/mm;
        z[m]%=mm;
    }
    while(m>=0&&z[m]==0) m--;
    if(m<0) m=0;
    for(i=m;i>=0;i--) c[i]=z[i];
    c[m+1]=0;
    c[m+2]=0;
    return m;
}
int main()
{
    int i,j,n,m,p,q;
    while(scanf("%d %d",&n,&m)!=EOF){
        p=0;
        while(n>0){
            x[p++]=n%mm;
            n=n/mm;
        }
       /* q=0;
        while(m>0){
            y[q++]=m%mm;
            m=m/mm;
        }
        p--;*/
        q--;
        memset(z,0,sizeof(z));
        j=mul(x,m,z,p);
        if(j<0){
            z[0]=0;
            j=0;
        }
        for(i=j;i>=0;i--){
            if(i==j) printf("%d",z[i]);
            else{
```

```
                printf("%08d",z[i]);
            }
        }
        printf("\n");
    }
}
```

53、完整高精度算法系统

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
char a1[10005],b1[10005],x1[10005],q[10005],f[10005];
int d[10005];
void plus(char *a,char *b,char *c){
    int s,t,p,i;
    memset(d,0,sizeof(d));
    s=(int)strlen(a);
    t=(int)strlen(b);
    p=0;
    s--;
    t--;
    while(s>=0&&t>=0){
        d[p]+=a[s]-'0'+b[t]-'0';
        if(d[p]>=10){
            d[p+1]+=d[p]/10;
            d[p]%=10;
        }
        p++;
        s--;
        t--;
    }
    if(s<0&&t>=0){
        while(t>=0){
            d[p]+=b[t]-'0';
            d[p+1]+=d[p]/10;
            d[p]%=10;
            t--;
            p++;
        }
    }
    else if(s>=0&&t<0){
        while(s>=0){
            d[p]+=a[s]-'0';
            d[p+1]+=d[p]/10;
            d[p]%=10;
            p++;
            s--;
        }
    }
    while(p>=0&&d[p]==0) p--;
    if(p<0){
        c[0]='0';
        c[1]='\0';
        return ;
```

```
    }
    for(i=0;i<=p;i++) c[i]=d[p-i]+'0';
    c[p+1]='\0';
}
void minus(char *a,char *b,char *c){
    int i,n,m,p,k;
    n=(int)strlen(a);
    m=(int)strlen(b);
    p=0;
    while(p<n&&a[0]=='0'){
        p++;
    }
    if(p==n){
        a[0]='0';
        a[1]='\0';
    }
    else{
        for(i=0;i<n-p;i++) a[i]=a[i+p];
        a[n-p]='\0';
    }
    p=0;
    while(p<m&&b[0]=='0') p++;
    if(p==m) b[1]='\0';
    else{
        for(i=0;i<m-p;i++) b[i]=b[i+p];
        b[m-p]='\0';
    }
    n=strlen(a);
    m=strlen(b);
    p=0;
    if(n<m){
        p=1;
    }
    else if(n==m){
        if(strcmp(a,b)<0) p=1;
    }
    memset(d,0,sizeof(d));
    k=0;
    if(p==1){
        for(i=n-1;i>=0;i--){
            d[k]+=b[m-n+i]-a[i];
            while(d[k]<0){
                d[k]=d[k]+10;
                d[k+1]--;
            }
            k++;
        }
        for(i=m-n-1;i>=0;i--){
            d[k]+=b[i]-'0';
            while(d[k]<0){
                d[k]+=10;
                d[k+1]--;
            }
```

```
            k++;
        }
        k++;
        while(k>=0&&d[k]==0) k--;
        if(k<0){
            c[0]='0';
            c[1]='\0';
        }
        else{
            c[0]='-';
            for(i=k;i>=0;i--) c[i+1]=d[k-i]+'0';
            c[k+2]='\0';
        }
        return ;
    }
    for(i=m-1;i>=0;i--){
        d[k]+=a[n-m+i]-b[i];
        while(d[k]<0){
            d[k]+=10;
            d[k+1]--;
        }
        k++;
    }
    for(i=n-m-1;i>=0;i--){
        d[k]+=a[i]-'0';
        while(d[k]<0){
            d[k]+=10;
            d[k+1]--;
        }
        k++;
    }
    k++;
    while(k>=0&&d[k]==0) k--;
    if(k<0){
        c[0]='0';
        c[1]='\0';
    }
    else{
        for(i=0;i<=k;i++) c[i]=d[k-i]+'0';
        c[k+1]='\0';
    }
    return ;
}
void mul(char *a,char *b,char *c){
    int i,j,n,m,k;
    n=strlen(a);
    m=strlen(b);
    memset(d,0,sizeof(d));
    for(i=m-1;i>=0;i--){
        for(j=n-1;j>=0;j--){
            d[(m-1-i)+(n-1-j)]+=(a[j]-'0')*(b[i]-'0');
            if(d[(m-1-i)+(n-1-j)]>=10){
                d[(m-1-i)+(n-1-j)+1]+=d[(m-1-i)+(n-1-j)]/10;
```

```
                d[(m-1-i)+(n-1-j)]%=10;
            }
        }
    }
    k=n+m-1;
    while(d[k]>=10){
        d[k+1]+=d[k]/10;
        d[k]=d[k]%10;
        k++;
    }
    while(d[k]==0&&k>=0) k--;
    if(k<0){
        c[0]='0';
        c[1]='\0';
    }
    else{
        for(i=0;i<=k;i++) c[i]=d[k-i]+'0';
        c[k+1]='\0';
    }
}
void mi(char *a,int p,char *c){
    strcpy(q,a);
    c[0]='1';
    c[1]='\0';
    while(p>0){
        if(p&1){
            mul(c,q,c);
        }
        mul(q,q,q);
        p>>=1;
    }
}
void chu(char *a,int p,char *c){
    memset(d,0,sizeof(d));
    int n,m,i,j;
    n=strlen(a);
    m=0;
    int k=0;
    for(i=0;i<n;i++){
        m=m*10+a[i]-'0';
        if(m>=p){
            d[k]+=m/p;
            k++;
            m=m%p;
        }
    }
    for(i=0;i<k;i++) c[i]=d[i]+'0';
    c[k]='\0';
}
```

54、后缀数组罗穗蹇模板（dc3）

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

```
#include<iostream>
#define maxn 1000003
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int wa[maxn],wb[maxn],wv[maxn],ws[maxn];
int r[maxn],sa[maxn];
int c0(int *r,int a,int b)
{return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];}
int c12(int k,int *r,int a,int b)
{if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
    else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];}
void sort(int *r,int *a,int *b,int n,int m)
{
    int i;
    for(i=0;i<n;i++) wv[i]=r[a[i]];
    for(i=0;i<m;i++) ws[i]=0;
    for(i=0;i<n;i++) ws[wv[i]]++;
    for(i=1;i<m;i++) ws[i]+=ws[i-1];
    for(i=n-1;i>=0;i--) b[--ws[wv[i]]]=a[i];
    return;
}
void dc3(int *r,int *sa,int n,int m)
{
    int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
    r[n]=r[n+1]=0;
    for(i=0;i<n;i++) if(i%3!=0) wa[tbc++]=i;
    sort(r+2,wa,wb,tbc,m);
    sort(r+1,wb,wa,tbc,m);
    sort(r,wa,wb,tbc,m);
    for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
        rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
    if(p<tbc) dc3(rn,san,tbc,p);
    else for(i=0;i<tbc;i++) san[rn[i]]=i;
    for(i=0;i<tbc;i++) if(san[i]<tb) wb[ta++]=san[i]*3;
    if(n%3==1) wb[ta++]=n-1;
    sort(r,wb,wa,ta,m);
    for(i=0;i<tbc;i++) wv[wb[i]=G(san[i])]=i;
    for(i=0,j=0,p=0;i<ta && j<tbc;p++)
        sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
    for(;i<ta;p++) sa[p]=wa[i++];
    for(;j<tbc;p++) sa[p]=wb[j++];
    return;
}
int rank[maxn],height[maxn];
void calheight(int *r,int *sa,int n)
{
    int i,j,k=0;
    for(i=1;i<=n;i++) rank[sa[i]]=i;
    for(i=0;i<n;height[rank[i++]]=k)
        for(k?k--:0,j=sa[rank[i]-1];r[i+k]==r[j+k];k++);
    return;
}
int RMQ[maxn];
```

```
int mm[maxn];
int best[20][maxn];
void initRMQ(int n)
{
    int i,j,a,b;
    for(mm[0]=-1,i=1;i<=n;i++)
        mm[i]=((i&(i-1))==0)?mm[i-1]+1:mm[i-1];
    for(i=1;i<=n;i++) best[0][i]=i;
    for(i=1;i<=mm[n];i++)
        for(j=1;j<=n+1-(1<<i);j++)
        {
            a=best[i-1][j];
            b=best[i-1][j+(1<<(i-1))];
            if(RMQ[a]<RMQ[b]) best[i][j]=a;
            else best[i][j]=b;
        }
    return;
}
int askRMQ(int a,int b)
{
    int t;
    t=mm[b-a+1];b-=(1<<t)-1;
    a=best[t][a];b=best[t][b];
    return RMQ[a]<RMQ[b]?a:b;
}
int lcp(int a,int b)
{
    int t;
    a=rank[a];b=rank[b];
    if(a>b) {t=a;a=b;b=t;}
    return(height[askRMQ(a+1,b)]);
}
char c[maxn];
struct pi{
    int x;
    int y;
}pp[maxn];
int main()
{
    int i,j,n,m,p,k,le,ri,mid;
    while(1){
        scanf("%d",&k);
        if(k==0) break;
        scanf("%s",c);
        n=strlen(c);
        for(i=0;i<n;i++){
            r[i]=c[i]-'0';
        }
        r[n]=0;
        dc3(r,sa,n+1,200002);
        calheight(r, sa, n);
    }
}
```

55、有源汇的上下界最小流

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<vector>
#include<string.h>
#include<queue>
#define LL int
#define inf 100000000
#define maxn 150
using namespace std;
struct pi
{
    int to;
    int cost;
    int rev;
    int id;
}pp;
vector<pi >g[maxn];
queue<int>q;
int s,t;
int line[maxn],leve[maxn];
int min(int a,int b)
{
    int p;
    p=a;
    if(b<a)
        p=b;
    return p;
}
void add(int a,int b,int cos,int id)
{
    pp.to=b;
    pp.cost=cos;
    pp.rev=(int)g[b].size();
    pp.id=id;
    g[a].push_back(pp);
    pp.to=a;
    pp.cost=0;
    pp.rev=(int)g[a].size()-1;
    pp.id=-1;
    g[b].push_back(pp);
    return ;
}
void bfs(void)
{
    q.push(s);
    int p,f,i;
    while(!q.empty())
    {
        p=q.front();
        q.pop();
        f=(int)g[p].size();
```

```
        for(i=0;i<f;i++)
        {
            pi &e=g[p][i];
            if(e.cost>0&&line[e.to]<0)
            {
                line[e.to]=line[p]+1;
                q.push(e.to);
            }
        }
    }
    return ;
}
int dfs(int v,int f)
{
    int p;
    if(v==t)
        return f;
    for(int &i=leve[v];i<g[v].size();i++)
    {
        pi &e=g[v][i];
        if(line[v]<line[e.to]&&e.cost>0)
        {
            p=dfs(e.to,min(f,e.cost));
            if(p>0)
            {
                e.cost-=p;
                g[e.to][e.rev].cost+=p;
                return p;
            }
        }
    }
    return 0;
}
LL  dinic()
{
    int f;
    LL flow=0;
    while(1)
    {
        memset(line,-1,sizeof(line));
        memset(leve,0,sizeof(leve));
        line[s]=0;
        bfs();
        if(line[t]<0)
            return flow;
        f=dfs(s,inf);
        if(f==0)
            continue;
        flow+=f;
        while((f=dfs(s,inf))>0)
        {
            flow+=f;
        }
```

```
    }
}
struct ppi{
    int from;
    int to;
    int a;
    int b;
}pp1[100005];
char c[1005];
int a[105];
int b[100005];
int main()
{
    int i,j,n,m,p,k,N=0,le,ri,mid;
    while(scanf("%d%d",&n,&m)!=EOF){
        memset(a,0,sizeof(a));
        for(i=0;i<=n+1;i++) g[i].clear();
        for(i=0;i<m;i++){
            scanf("%d%d%d",&pp1[i].from,&pp1[i].to,&pp1[i].b);
            scanf("%d",&p);
            if(p==0){
                pp1[i].a=0;
            }
            else pp1[i].a=pp1[i].b;
            a[pp1[i].from]+=pp1[i].a;
            a[pp1[i].to]-=pp1[i].a;
        }
        s=0;
        t=n+1;
        le=0;
        ri=1000000000;
        while(le<=ri){//将t-》s连一条有上界的边，二分上界。
            mid=(le+ri)/2;
            for(i=0;i<=t;i++) g[i].clear();
            for(i=0;i<m;i++){
                add(pp1[i].from,pp1[i].to,pp1[i].b-
pp1[i].a,i);
            }
            for(i=1;i<=n;i++){
                if(a[i]==0) continue;
                if(a[i]>0) add(i,t,a[i],-1);
                else add(s,i,-a[i],-1);
            }
            add(n,1,mid,-1);
            dinic();
            p=0;
            k=g[s].size();
            for(i=0;i<k;i++){
                if(g[s][i].cost!=0){
                    p=1;
                    break;
                }
            }
```

```
            if(p) le=mid+1;
            else ri=mid-1;
        }
        if(le>=1000000000){
            printf("Impossible\n");
        }
        else{
            printf("%d\n",le);
            for(i=0;i<=t;i++) g[i].clear();
            for(i=0;i<m;i++){
                add(pp1[i].from,pp1[i].to,pp1[i].b-
pp1[i].a,i);
            }
            for(i=1;i<=n;i++){
                if(a[i]==0) continue;
                if(a[i]>0) add(i,t,a[i],-1);
                else add(s,i,-a[i],-1);
            }
            add(n,1,le,-1);
            dinic();
            for(i=1;i<=n;i++){
                k=g[i].size();
                for(j=0;j<k;j++){
                    if(g[i][j].id!=-1){
                        b[g[i][j].id]=pp1[g[i][j].id].b-g[i]
[j].cost;
                    }
                }
            }
            for(i=0;i<m;i++){
                if(i==0) printf("%d",b[i]);
                else printf(" %d",b[i]);
            }
            printf("\n");
        }
    }
}
```

56、上下限网络流

```
/*
 1、无源无汇可行流。

 由流量守恒sigma(g[u,i])+sigma(b[u,i])=sigma(g[i,v])
+sigma(b[i,v]).

 其中b[u,i]是流量下界，g[u,i]<=c[u,i]-b[u,i]。c[u,i]是流量上界，最后
得到

 sigma(g[u,i])=sigma(g[i,v])+p。如果p大于0，就添一条i到t流量为p的
边，其中，t是超级源点。

 如果小于0，就添一条s到i的流量为-p的边。跑一遍dinic，如果与s相连的边有一
条边不满流就不是可行流。
```

跑完最大流之后每天边的流量等于流量上界减去残余流量。

2.有源汇最大最小流。

一、最大流：

如果对于网路中流量为a，则连一条t->s流量下界为a的边变成无源无汇网络，则这个网络一定存在可行流。

所以可以二分a，也就是说，二分t-》s流量下界，判断是否是可行流即可。每条边流量即为无源无汇网络每条边流量；

二、最小流：同理最大流，二分t->s的流量上界，然后用可行流判断即可。
*/

57、枚举一个数二进制表示下的子集

```
#include<cstdio>
int main()
{
    int i,j,s;
    for(i=s;i;i=(i-1)&s);//枚举s子集
}
```

58、母函数模板

```
#include <iostream>

using namespace std;

const int _max = 10001;

// c1是保存各项质量砝码可以组合的数目

// c2是中间量，保存每一次的情况

int c1[_max], c2[_max];

int main()

{ //int n,i,j,k;

    int nNum; //

    int i, j, k;

    while(cin >> nNum)

    {

        for(i=0; i<=nNum; ++i) // -- ①

        {

            c1[i] = 1;

            c2[i] = 0;
```

```
        }
        for(i=2; i<=nNum; ++i) // —— ②// bie wang le i*i<num
na dao ti
        {
            for(j=0; j<=nNum; ++j) // —— ③
                for(k=0; k+j<=nNum; k+=i) // —— ④//bie wang le
k+=i*i na dao ti
                {
                    c2[j+k] += c1[j];
                }
            for(j=0; j<=nNum; ++j) // —— ⑤
            {
                c1[j] = c2[j];
                c2[j] = 0;
            }
        }
        cout << c1[nNum] << endl;
    }
    return 0;
}
```

59、最长上升子序列nlogn通解（dp之线段树优化）

对于任何一个子序列，我们可以轻易的列出dp方程。dp[i]=max(dp[i],dp[j]+1);其中a[j]<a[i].但这是n^2的必然T。我们在仔细地看一下这个方程，我们只是要求高度小于等于a[i]里面的最大的dp值，如果高度是整数，也就是求1~a[i]的高度里面dp值最大，这里已经很明显，这个就是一个线段树的区间查询的过程。但是高度很大数组开不下怎么办？离散化一下，区间便变成了1~n。

每次求dp值的时候先查询，先求出a[i]在整个有序序列中的位置（二分），假如是p，那就查询1~p之间最大的dp值。然后求出

dp[i]，再把dp[i]插入p这个位置的线段树中。

60、上下限网络流大攻略

1、无源无汇可行流。

由流量守恒sigma(g[u,i])+sigma(b[u,i])=sigma(g[i,v])+sigma(b[i,v]).

其中b[u,i]是流量下界，g[u,i]<=c[u,i]-b[u,i]。

c[u,i]是流量上界，最后得到sigma(g[u,i])=sigma(g[i,v])+p。如果p大于0，就添一条i到t流量为p的边，其中，t是超级源点。

如果小于0，就添一条s到i的流量为-p的边。跑一遍dinic，如果与s相连的边有一条边不满流就不是可行流。

跑完最大流之后每天边的流量等于流量上界减去残余流量。

2.有源汇最大最小流。

一、最大流：

如果对于网路中流量为a，则连一条t->s流量下界为a的边变成无源无汇网络，则这个网络一定存在可行流。

所以可以二分a，也就是说，二分t-》s流量下界，判断是否是可行流即可。每条边流量即为无源无汇网络每条边流量；

二、最小流：同理最大流，二分t->s的流量上界，然后用可行流判断即可。