
ACTINIUM'S ACM TEMPLATE

Last build at November 1, 2012

目录

1	注意事项	3
2	数据结构	4
2.1	倍增法求LCA	4
2.2	扫描线求面积并	6
2.3	矩形周长并	9
2.4	树状数组	12
2.4.1	树状数组区间加减, 区间查询	12
2.4.2	树状数组求最值	13
2.4.3	树状数组求第K大	14
2.5	可修改堆	15
2.6	莫队算法	17
2.7	划分树	19
2.8	二维RMQ	22
2.9	主席树	24
2.10	树链剖分	26
2.11	KD树	29
2.12	treap	32
2.13	DancingLinks	35
2.13.1	精确覆盖	35
2.13.2	重复覆盖	36
3	字符串	39
3.1	最小表示法	39
3.2	manacher	40
3.3	扩展KMP	41
3.4	KMP	42
3.5	AC自动机	43
3.6	后缀数组	45
4	图论	48
4.1	SPFA判负环	48
4.2	tarjan	49
4.2.1	求SCC	49
4.2.2	求割点、点双联通分支	49
4.2.3	求桥	52
4.2.4	求LCA	54
4.3	网络流	56
4.3.1	EK	56
4.3.2	Sap	58
4.3.3	sap邻接表	60
4.3.4	费用流	61
4.3.5	无源无汇上下界网络流	62
4.3.6	有源有汇上下界网络流	64
4.4	2-sat	68
4.5	A*求最短路	72
4.6	带花树	74
4.7	KM算法	78
4.8	欧拉回路通路	81
4.9	哈密尔顿回路	83
4.10	最小树形图	86
4.10.1	邻接表	86
4.10.2	邻接矩阵	88
4.11	稳定婚姻	91
4.12	最优比率生成树	93
5	DP	95
5.1	插头DP	95
5.2	数位DP	98
5.3	最大平均子段和	100
5.4	斯坦纳树	102
5.5	四边形不等式	104

6	数学	106
6.1	数学结论	106
6.2	数论基础	107
6.3	筛法求素数	110
6.4	同余方程	111
6.5	高次同余方程	112
6.6	高斯消元	115
6.6.1	高斯消元解实数方程	115
6.6.2	高斯消元解异或方程组	115
6.6.3	模意义下求行列式的值	117
6.7	POLYA	119
6.7.1	概念	119
6.7.2	题目	119
7	计算几何	122
7.1	几何基础	122
7.1.1	求两向量的叉积	122
7.1.2	求平面两点欧氏距离	122
7.1.3	判断点是否在线段上	122
7.1.4	判断两线段是否相交 (快速排斥+跨立实验)	122
7.1.5	判断double型变量的符号	123
7.1.6	求点到线段的最短距离(此做法过于naive,可用点积+叉积来求)	123
7.1.7	求两线段间最短距离	124
7.1.8	判断两直线是否相交 (共线及平行, 若相交则求出交点)	124
7.1.9	求两直线交点	125
7.1.10	求两线段交点	125
7.2	多边形	126
7.2.1	判断线段是否与矩形相交 (包括线段在矩形内部)	126
7.2.2	判断点是否在多边形内部 (包含在边上), 射线法	126
7.2.3	求简单多边形重心	127
7.2.4	graham_scan求凸包	127
7.2.5	旋转卡壳求凸包直径	128
7.2.6	凸包间最短距离 (旋转卡壳)	129
7.2.7	判断两凸多边形是否相交 (graham_scan求凸包+枚举边、点)	133
7.3	半平面交	137
7.3.1	求多边形内核	137
7.3.2	zzyO(n*logn)做法	139
7.3.3	两个简单多边形求面积并、交	142
7.4	圆	145
7.4.1	点类	145
7.4.2	圆类	146
7.4.3	圆面积交、并	147
7.4.4	简单多边形与圆求面积交	152
7.4.5	求线段与圆的交点	154
7.4.6	求两圆公切线	154
7.4.7	最小圆覆盖	155
7.4.8	单位圆覆盖	157
7.5	模拟退火	160
7.5.1	求多边形费马点	160
7.5.2	最小球覆盖	161
7.6	三维几何	164
7.6.1	三维凸包	164
7.6.2	求两球体积并	168
7.7	三维仿射变换	170
8	其他	174
8.1	矩阵乘	174
8.2	平面最近点对	175
8.3	读入外挂	177
8.4	JAVA分数类	178
8.5	魔方	181
8.6	Hashmap	184

1 注意事项

CodeBlocks修改终端:

只要Settings-->Environment-->Terminal to launch console programs

用“gnome-terminal -t \$TITLE -x”替换掉“xterm -T \$TITLE -e”

测试系统速度:

```
1 #include <iostream>
2 #include <cstdio>
3 #include <ctime>
4 using namespace std;
5 int main()
6 {
7     int i=0,j=0,k=0;
8     time_t start = clock(), end;
9     for (int i=0;i<316000000;i++){
10     }
11     end = clock();
12     printf("done, total time:%ld ms\n", (end - start)/1000);
13     return 0;
14 }
```

对拍程序:

```
1 while true;do
2 ./gen>in
3 ./bruteforce<in>ans
4 ./prog<in>out
5 if diff out ans;then
6 echo AC
7 else
8 echo WA
9 read p
10 fi
11 sleep 1
12 done
```

2 数据结构

2.1 倍增法求LCA

```
1 # include <iostream>
2 # include <cstring>
3 # include <cstdio>
4 # include <vector>
5 # define N 55010
6 using namespace std;
7 struct node
8 {
9     int u,w;
10     //node (){}
11     node (int uu=0,int ww=0):u(uu),w(ww){}
12 };
13 vector <node> tree[N];
14 int p[N][50],d[N],dist[N];
15 void dfs (int x,int pre)
16 {
17     int y,len=tree[x].size(),now;
18     for (int i=0;i<len;i++)
19     {
20         y=tree[x][i].u;
21         if (y==pre)
22             continue;
23         now=p[y][0]=x;
24         d[y]=d[x]+1;
25         dist[y]=dist[x]+tree[x][i].w;
26         for (int j=0;now!=-1 && p[now][j]!=-1;j++)
27         {
28             p[y][j+1]=p[now][j];
29             now=p[now][j];
30         }
31         dfs(y,x);
32     }
33 }
34 int LCA(int x,int y)
35 {
36     if (x==y)
37         return x;
38     int tem;
39     if (d[x]<d[y])
40     {
41         tem=x;
42         x=y;
43         y=tem;
44     }
45     int dis=d[x]-d[y],k=0;
46     while (dis)
47     {
48         if (dis&1)
49             x=p[x][k];
50         dis>>=1;
51         k++;
52     }
53     k=0;
54     while (x!=y)
55     {
56         if ( p[x][k]!=p[y][k] || (p[x][k]==p[y][k] && k==0) )
57         {
58             x=p[x][k];
59             y=p[y][k];
```

```

60         k++;
61     }else k--;
62 }
63 return x;
64 }
65 int main (void)
66 {
67     int n,u,v,c,q,m;
68     char s[20];
69     scanf ("%d%d",&n,&m);
70     for (int i=1;i<=n;i++)
71         tree[i].clear();
72     memset(p,-1,sizeof(p));
73     for (int i=0;i<m;i++)
74     {
75         scanf ("%d%d%d%s",&u,&v,&c,s);
76         tree[u].push_back(node(v,c));
77         tree[v].push_back(node(u,c));
78     }
79     dist[1]=0;
80     d[1]=0;
81     dfs(1,-1);
82     scanf ("%d",&q);
83     while (q--)
84     {
85         scanf ("%d%d",&u,&v);
86         printf ("%d\n",dist[u]+dist[v]-2*dist[LCA(u,v)]);
87     }
88     return 0;
89 }

```

2.2 扫描线求面积并

hdu 4419

扫描线+容斥原理

注意lazy标记不用向下传递，因为一条线段被插入之后还会被删除。还有线段树val值的维护

```
1  # include <cstring>
2  # include <cstdio>
3  # include <cstdlib>
4  # include <iostream>
5  # include <algorithm>
6  using namespace std;
7  # define N 20500
8  typedef long long LOL;
9  struct NODE
10 {
11     int x,y1,y2,id,c;
12 }a[N];
13 int val[8][N<<2],len[N<<2],cov[8][N<<2];
14 int x[N],y[N];
15 LOL sum[10],ans[10];
16 void build(int idx,int l,int r)
17 {
18     if (l==r)
19     {
20         len[idx]=y[l+1]-y[l];
21         return ;
22     }
23     int mid=(l+r)>>1;
24     build(idx<<1,l,mid);
25     build(idx<<1|1,mid+1,r);
26     len[idx]=len[idx<<1]+len[idx<<1|1];
27 }
28 inline void up(int now,int idx,int ll)
29 {
30     if (cov[now][idx])
31         val[now][idx]=len[idx];
32     else
33     {
34         if (ll)
35             val[now][idx]=val[now][idx<<1]+val[now][idx<<1|1];
36         else val[now][idx]=0;
37     }
38 }
39 inline int bin(int a[],int l,int r,int x)
40 {
41     int mid;
42     while (l<r)
43     {
44         mid=(l+r+1)>>1;
45         if (a[mid]>x)
46             r=mid-1;
47         else l=mid;
48     }
49     return l;
50 }
51 void update(int now,int idx,int l,int r,int x,int y,int c)
52 {
53     if (x<=l && y>=r)
54     {
55         cov[now][idx]+=c;
56         up(now,idx,r-1);
57         return ;
58     }
```

```

59     int mid=(l+r)>>1;
60     if (y<=mid)
61         update(now,idx<<1,l,mid,x,y,c);
62     else if (x>mid)
63         update(now,idx<<1|1,mid+1,r,x,y,c);
64     else
65     {
66         update(now,idx<<1,l,mid,x,mid,c);
67         update(now,idx<<1|1,mid+1,r,mid+1,y,c);
68     }
69     up(now,idx,r-1);
70 }
71 int cmp(NODE a,NODE b)
72 {
73     return a.x<b.x;
74 }
75 int main (void)
76 {
77     int t,n,ys=0;
78     char op[20];
79     scanf("%d",&t);
80     while (t--)
81     {
82         scanf("%d",&n);
83         int x1,x2,y1,y2,cnt=0;
84         for (int i=1;i<=n;i++)
85         {
86             scanf("%s%d%d%d",op,&x1,&y1,&x2,&y2);
87
88             ++cnt;
89             if (op[0]=='R')
90                 a[cnt].c=1;
91             else
92                 if (op[0]=='G')
93                     a[cnt].c=2;
94                 else a[cnt].c=4;
95
96             a[cnt].x=x1;
97             a[cnt].y1=y1,a[cnt].y2=y2;
98             a[cnt].id=1;
99             x[cnt]=x1,y[cnt]=y1;
100
101             ++cnt;
102             a[cnt].c=a[cnt-1].c;
103             a[cnt].x=x2;
104             a[cnt].y1=y1,a[cnt].y2=y2;
105             a[cnt].id=-1;
106             x[cnt]=x2,y[cnt]=y2;
107         }
108
109         sort(x+1,x+1+cnt);
110         sort(y+1,y+1+cnt);
111         sort(a+1,a+1+cnt,cmp);
112
113         int n1=1,n2=1;
114         for (int i=2;i<=cnt;i++)
115             if (x[i]!=x[i-1])
116                 x[++n1]=x[i];
117         for (int i=2;i<=cnt;i++)
118             if (y[i]!=y[i-1])
119                 y[++n2]=y[i];
120
121         memset(val,0,sizeof(val));

```



```

122     memset(cov,0,sizeof(cov));
123     memset(sum,0,sizeof(sum));
124     build(1,1,n2-1);
125
126
127     int cur=1;
128     for (int i=1;i<=n1;i++)
129     {
130         for (int p=1;p<8;p++)
131             sum[p]+=(LOL)val[p][1]*(LOL)(x[i]-x[i-1]);
132         while (cur<=cnt && a[cur].x==x[i])
133         {
134             y1=bin(y,1,n2,a[cur].y1);
135             y2=bin(y,1,n2,a[cur].y2)-1;
136             for (int p=1;p<8;p++)
137             {
138                 if (!(a[cur].c&p)) continue;
139                 update(p,1,1,n2-1,y1,y2,a[cur].id);
140             }
141             ++cur;
142         }
143     }
144     LOL tmp3=sum[1]+sum[2]-sum[3];
145     LOL tmp5=sum[1]+sum[4]-sum[5];
146     LOL tmp6=sum[2]+sum[4]-sum[6];
147     LOL tmp7=sum[7]-sum[1]-sum[2]-sum[4]+tmp3+tmp5+tmp6;
148     ans[1]=sum[1]-tmp3-tmp5+tmp7;
149     ans[2]=sum[2]-tmp3-tmp6+tmp7;
150     ans[3]=sum[4]-tmp5-tmp6+tmp7;
151     ans[4]=tmp3-tmp7;
152     ans[5]=tmp5-tmp7;
153     ans[6]=tmp6-tmp7;
154     ans[7]=tmp7;
155     cout<<"Case "<<+ys<<": "<<endl;
156     for (int i=1;i<8;i++)
157         cout<<ans[i]<<endl;
158 }
159 return 0;
160 }

```

2.3 矩形周长并

HDU 1828.cpp

矩形周长并，统计y轴上线段条数

注意cmp函数，先入再出

```
1  # include <iostream>
2  # include <cstdio>
3  # include <cstdlib>
4  # include <cstring>
5  # include <algorithm>
6  using namespace std;
7  # define N 10050
8  struct SEG
9  {
10     int x,y1,y2,flag;
11 }seg[N];
12 struct NODE
13 {
14     int lc,rc,len,sum,num,cover;
15 }tree[N<<2];
16 int x[N],y[N];
17 int cmp(SEG a,SEG b)
18 {
19     if (a.x!=b.x)
20         return a.x<b.x;
21     return a.flag>b.flag;
22 }
23 int bin_search(int a[],int l,int r,int x)
24 {
25     int mid;
26     while (l<r)
27     {
28         mid=(l+r)>>1;
29         if (x>a[mid])
30             l=mid+1;
31         else r=mid;
32     }
33     return l;
34 }
35 void build(int idx,int l,int r)
36 {
37     tree[idx].lc=tree[idx].rc=tree[idx].sum=tree[idx].cover=tree[idx].num=0;
38     if (l==r)
39         tree[idx].len=y[l+1]-y[l];
40     else
41     {
42         int mid=(l+r)>>1;
43         build(idx<<1,l,mid);
44         build(idx<<1|1,mid+1,r);
45         tree[idx].len=tree[idx<<1].len+tree[idx<<1|1].len;
46     }
47 }
48 inline void solve(int idx,int len)
49 {
50     if (tree[idx].cover)
51     {
52         tree[idx].sum=tree[idx].len;
53         tree[idx].num=2;
54         tree[idx].lc=tree[idx].rc=1;
55     }
56     else
57     {
58         if (len>1)
```

```

59     {
60         tree[idx].sum=tree[idx<<1].sum+tree[idx<<1|1].sum;
61         tree[idx].num=tree[idx<<1].num+tree[idx<<1|1].num;
62         if (tree[idx<<1].rc && tree[idx<<1|1].lc)
63             tree[idx].num-=2;
64         tree[idx].lc=tree[idx<<1].lc;
65         tree[idx].rc=tree[idx<<1|1].rc;
66     }
67     else
68     {
69         tree[idx].num=tree[idx].sum=0;
70         tree[idx].lc=tree[idx].rc=0;
71     }
72 }
73 }
74 void update(int idx,int l,int r,int x,int y,int c)
75 {
76     if (x<=l && y>=r)
77     {
78         tree[idx].cover+=c;
79         solve(idx,r-l+1);
80         return ;
81     }
82     int mid=(l+r)>>1;
83     if (y<=mid)
84         update(idx<<1,l,mid,x,y,c);
85     else
86         if (x>mid)
87             update(idx<<1|1,mid+1,r,x,y,c);
88         else
89         {
90             update(idx<<1,l,mid,x,mid,c);
91             update(idx<<1|1,mid+1,r,mid+1,y,c);
92         }
93     solve(idx,r-l+1);
94 }
95 int main (void)
96 {
97     int n;
98     while (cin>>n)
99     {
100         int cnt=0,x1,x2,y1,y2;
101         for (int i=1;i<=n;i++)
102         {
103             scanf ("%d%d%d%d",&x1,&y1,&x2,&y2);
104             seg[++cnt].x=x1,seg[cnt].y1=y1,seg[cnt].y2=y2,seg[cnt].flag=1;
105             x[cnt]=x1,y[cnt]=y1;
106             seg[++cnt].x=x2,seg[cnt].y1=y1,seg[cnt].y2=y2,seg[cnt].flag=-1;
107             x[cnt]=x2,y[cnt]=y2;
108         }
109         sort(x+1,x+1+cnt);
110         sort(y+1,y+1+cnt);
111         sort(seg+1,seg+1+cnt,cmp);
112         int nx=1,ny=1;
113         for (int i=2;i<=cnt;i++)
114         {
115             if (x[i]!=x[i-1])
116                 x[++nx]=x[i];
117             if (y[i]!=y[i-1])
118                 y[++ny]=y[i];
119         }
120         build(1,1,ny-1);
121         int ans=0,last=0,j=1,k;

```

```

122     for (int i=1;i<=nx;i++)
123     {
124         if (i>1)
125             ans+=(x[i]-x[i-1])*tree[1].num;// 加矩形长边
126         for (k=j;k<=cnt && seg[k].x==x[i];k++)
127         {
128             update(1,1,ny-1,bin_search(y,1,ny,seg[k].y1),bin_search(y,1,
129                 ny,seg[k].y2)-1,seg[k].flag);
130             ans+=abs(tree[1].sum-last);// 加矩形宽边，每次加的是增量
131             last=tree[1].sum;
132         }
133         j=k;
134     }
135     cout<<ans<<endl;
136 }
137 return 0;
138 }

```

2.4 树状数组

2.4.1 树状数组区间加减，区间查询

树状数组区间加一个值，然后询问区间的和。假设原数组为a，再维护一个数组b， b_i 记录前i个元素每个全部加了 b_i 的值。当我们查询前x项的值时，那么 $sum = \sum_{i=1}^x a_i + x * \sum_{i=x+1}^n b_i + \sum_{i=1}^x b_i * i$ ，均化为了求和的操作，那么可以用树状数组去维护 b_i 与 $b_i * i$ 的值。

```
1  #include<iostream>
2  #include<cstring>
3  #include<cstdio>
4  using namespace std;
5  int lowbit(int x)
6  {
7      return x&(-x);
8  }
9  typedef long long LOL;
10 const int N = 100010;
11 LOL a[N],d[N],c[N],sum[N];
12 int n;
13 void Add(int x,int v)
14 {
15     if(x==0) return ;
16     LOL t=v*x;
17     while(x<=n)
18     {
19         d[x]+=v;
20         c[x]+=t;
21         x+=lowbit(x);
22     }
23 }
24 LOL getsumd(int x)
25 {
26     LOL u=x;
27     LOL sum=0;
28     while(x>0){
29         sum+=d[x];
30         x-=lowbit(x);
31     }
32     return sum;
33 }
34 LOL getsumc(int x)
35 {
36     LOL u=x;
37     LOL sum=0;
38     while(x>0){
39         sum+=c[x];
40         x-=lowbit(x);
41     }
42     return sum;
43 }
44 LOL getsum(int x)
45 {
46     LOL s1=getsumd(n)-getsumd(x);
47     LOL s2=getsumc(x);
48     return sum[x]+(LOL)x*s1+s2;
49 }
50 char com[2];
51 int main(void)
52 {
53     int m,w;
54     scanf("%d%d",&n,&m);
55     for(int i=1;i<=n;i++)
```

```

56     {
57         scanf ("%lld",&a[i]);
58         sum[i]=sum[i-1]+a[i];
59     }
60     int x,y;
61     for(int i=1;i<=m;i++)
62     {
63         scanf ("%s%d%d",com,&x,&y);
64         if (com[0]=='Q')
65         {
66             printf ("%lld\n",getsum(y)-getsum(x-1));
67         }
68         else
69         {
70             scanf ("%d",&w);
71             Add(y,w);Add(x-1,-w);
72         }
73     }
74     return 0;
75 }

```

2.4.2 树状数组求最值

HDU1754

复杂度为 $O(n * \log n * \log n)$

```

1  #include<iostream>
2  #include<cstring>
3  #include<cstdio>
4  using namespace std;
5  const int N = 200010;
6  int a[N],c[N];
7  int lowbit(int x)
8  {
9      return x&(-x);
10 }
11 void init(int n)
12 {
13     for(int i=1;i<=n;i++){
14         c[i]=a[i];
15         for(int j=1;j<lowbit(i);j<=1){
16             c[i]=max(c[i],c[i-j]);
17         }
18     }
19 }
20 int query(int l,int r)
21 {
22     int ans=a[r];
23     while(l<=r){
24         ans=max(ans,a[r]);
25         if(r-lowbit(r)+1>=1){
26             ans=max(ans,c[r]);
27             r-=lowbit(r);
28         }else{
29             r--;
30         }
31     }
32     return ans;
33 }
34 void modify(int idx,int v,int n)
35 {
36     a[idx]=v;
37     for(int i=idx;i<=n;i+=lowbit(i)){

```

```

38         c[i]=v;
39         for(int j=1;j<lowbit(i);j<=1){
40             c[i]=max(c[i],c[i-j]);
41         }
42     }
43 }
44 int main(void)
45 {
46     int n,m,x,y;
47     while(scanf("%d%d",&n,&m)!=EOF)
48     {
49         for(int i=1;i<=n;i++){
50             scanf("%d",&a[i]);
51         }
52         init(n);
53         char op[2];
54         for(int i=1;i<=m;i++){
55             scanf("%s%d%d",op,&x,&y);
56             if(op[0]=='Q'){
57                 printf("%d\n",query(x,y));
58             }else{
59                 modify(x,y,n);
60             }
61         }
62     }
63     return 0;
64 }

```

2.4.3 树状数组求第K大

```

1 int find(int k)
2 {
3     int cnt=0,res=0;
4     for(int i=18;i>=0;i--){
5         res+=1<<i;
6         if(res>=maxn||cnt+c[res]>=k){
7             res--(1<<i);
8         }else{
9             cnt+=c[res];
10        }
11    }
12    return res+1;
13 }

```

2.5 可修改堆

```
1  #include<iostream>
2  #include<cstdio>
3  #include<cstring>
4  using namespace std;
5  const int N = 1000010;
6  struct HEAP
7  {
8      int x,node;
9  }heap[N];
10 int set[N];
11 int size=0;
12 int a[N];
13 void Swap(int x,int y)
14 {
15     swap(heap[x],heap[y]);
16     set[heap[x].node]=x;
17     set[heap[y].node]=y;
18 }
19 void up(int &pos)
20 {
21     while(pos>=2&&heap[pos].x<heap[pos>>1].x)
22     {
23         Swap(pos,pos>>1);
24         pos=pos>>1;
25     }
26 }
27 void down(int &pos)
28 {
29     int j=pos<<1;
30     while(j<=size)
31     {
32         if(j<size&&heap[j+1].x<heap[j].x)
33         {
34             j++;
35         }
36         if(heap[pos].x<=heap[j].x) break;
37         Swap(pos,j);
38         pos=j;j=pos<<1;
39     }
40 }
41 void insert(int size,int node,int val)
42 {
43     heap[size].x=val;
44     heap[size].node=node;
45     set[node]=size;
46     up(size);
47 }
48 int delmin(int size)
49 {
50     int tmp=heap[1].node;
51     Swap(1,size);
52     size--;
53     int pos=1;
54     down(pos);
55     return tmp;
56 }
57 void modify(int pos,int val)
58 {
59     heap[pos].x=val;
60     up(pos);
61     down(pos);
```



```
62 | }
63 | void del(int pos)
64 | {
65 |     Swap(pos,size);
66 |     size--;
67 |     up(pos);
68 |     down(pos);
69 | }
70 | int main(void)
71 | {
72 |     return 0;
73 | }
```

2.6 莫队算法

基本可以处理各种序列问题，复杂度 $O(n * \sqrt{n})$

```
1 #include<iostream>
2 #include<cstdio>
3 #include<cstring>
4 #include<algorithm>
5 #include<cmath>
6 using namespace std;
7 const int N =100010;
8 struct T
9 {
10     int x,y;
11     int id;
12 }tt[N];
13 int sz;
14 bool cmp(T a,T b)
15 {
16     if(a.x/sz==b.x/sz){
17         return a.y<b.y;
18     }
19     return a.x/sz<b.x/sz;
20 }
21 int a[N];
22 int ans[N];
23 int cnt[N];
24 int num[N];
25 int maxx;
26 void add(int x)
27 {
28     cnt[a[x]]++;
29     num[cnt[a[x]]-1]--;
30     num[cnt[a[x]]]++;
31     if(maxx<cnt[a[x]]){
32         maxx=cnt[a[x]];
33     }
34 }
35 void del(int x)
36 {
37     cnt[a[x]]--;
38     num[cnt[a[x]]+1]--;
39     num[cnt[a[x]]]++;
40     if(maxx==cnt[a[x]]+1){
41         if(num[cnt[a[x]]+1]==0){
42             maxx=cnt[a[x]];
43         }
44     }
45 }
46 int main(void)
47 {
48     int n,m;
49     scanf("%d%d",&n,&m);
50     sz=(int)sqrt((double)n);
51     for(int i=1;i<=n;i++){
52         scanf("%d",&a[i]);
53     }
54     for(int i=1;i<=m;i++){
55         scanf("%d%d",&tt[i].x,&tt[i].y);
56         tt[i].x++;tt[i].y++;
57         tt[i].id=i;
58     }
59     sort(tt+1,tt+1+m,cmp);
60     int cl=1,cr=1;
```

```

61 |     num[0]=n;
62 |     maxx=0;
63 |     add(1);
64 |     for(int i=1;i<=m;++i){
65 |         while(cr < tt[i].y) add(++ cr);
66 |         while(cl < tt[i].x) del(cl ++);
67 |         while(cl > tt[i].x) add(-- cl);
68 |         while(cr > tt[i].y) del(cr --);
69 |         ans[tt[i].id] = maxx;
70 |     }
71 |     for(int i=1;i<=m;++i) printf("%d\n", ans[i]);
72 |     return 0;
73 | }

```

2.7 划分树

```
1  #include<iostream>
2  #include<cstring>
3  #include<cstdio>
4  #include<algorithm>
5  using namespace std;
6  const int N = 100010;
7  struct Seg
8  {
9      int left,right;
10     int mid(){
11         return (left+right)>>1;
12     }
13 }tt[N<<2];
14 struct Q
15 {
16     int op,x,y,k,now;
17 }q[N<<1];
18 int val[25][N],toleft[25][N],sorted[N];
19 void build(int left,int right,int d,int idx)
20 {
21     tt[idx].left=left;
22     tt[idx].right=right;
23     if(left==right) return ;
24     int mid=tt[idx].mid();
25     int lsame=mid-left+1;
26     for(int i=left;i<=right;i++)
27     {
28         if(val[d][i]<sorted[mid])
29         {
30             lsame--;
31         }
32     }
33     int lpos=left,rpos=mid+1,same=0;
34     for(int i=left;i<=right;i++)
35     {
36         if(i==left)
37         {
38             toleft[d][i]=0;
39         }
40         else
41         {
42             toleft[d][i]=toleft[d][i-1];
43         }
44         if(val[d][i]<sorted[mid]){
45             toleft[d][i]++;
46             val[d+1][lpos++]=val[d][i];
47         }
48         else if(val[d][i]>sorted[mid])
49         {
50             val[d+1][rpos++]=val[d][i];
51         }
52         else
53         {
54             if(same<lsame)
55             {
56                 same++;
57                 toleft[d][i]++;
58                 val[d+1][lpos++]=val[d][i];
59             }
60             else
61             {
```

```

62         val[d+1][rpos++]=val[d][i];
63     }
64 }
65 }
66 build(left,mid,d+1,idx<<1);
67 build(mid+1,right,d+1,idx<<1|1);
68 }
69 int query(int left,int right,int k,int d,int idx)//left到right的第k小的数
70 {
71     if(left==right)
72     {
73         return val[d][left];
74     }
75     int s;
76     int ss;
77     if(left==tt[idx].left)
78     {
79         s=toleft[d][right];
80         ss=0;
81     }
82     else
83     {
84         s=toleft[d][right]-toleft[d][left-1];
85         ss=toleft[d][left-1];
86     }
87     if(s>=k)
88     {
89         int newl=tt[idx].left+ss;
90         int newr=tt[idx].left+ss+s-1;
91         return query(newl,newr,k,d+1,idx<<1);
92     }
93     else
94     {
95         int mid=tt[idx].mid();
96         int tmp=left-tt[idx].left-ss;
97         int newl=mid+1+tmp;
98         int newr=mid+1+tmp+right-left-s;
99         return query(newl,newr,k-s,d+1,idx<<1|1);
100     }
101 }
102 int find(int d,int id,int x,int idx)//前id个数小于等于x的有多少个
103 {
104     if(id<tt[idx].left) return 0;
105     if(tt[idx].left==tt[idx].right) return sorted[id]<=x;
106     int s=toleft[d][id];
107     int mid=tt[idx].mid();
108     if(sorted[mid]<=x)
109     {
110         return find(d+1,mid+id-tt[idx].left+1-s,x,idx<<1|1)+s;
111     }
112     else
113     {
114         return find(d+1,tt[idx].left+s-1,x,idx<<1);
115     }
116 }
117 char com[20];
118 int main(void)
119 {
120     int n,ys=0;
121     while(scanf("%d",&n)!=EOF)
122     {
123         int len=0;
124         for(int i=1;i<=n;i++)

```

```

125     {
126         scanf ("%s", com);
127         if (strcmp (com, "Insert") == 0)
128         {
129             scanf ("%d", &q[i].x);
130             q[i].op=1;
131             ++len;
132             val[0][len]=q[i].x;
133             sorted[len]=val[0][len];
134         }
135         else if (strcmp (com, "Query_1") == 0)
136         {
137             scanf ("%d%d%d", &q[i].x, &q[i].y, &q[i].k);
138             q[i].op=2;
139         }
140         else if (strcmp (com, "Query_2") == 0)
141         {
142             scanf ("%d", &q[i].k);
143             q[i].op=3;
144             q[i].x=1; q[i].y=len;
145         }
146         else
147         {
148             scanf ("%d", &q[i].k);
149             q[i].op=4;
150             q[i].x=1; q[i].y=len;
151         }
152     }
153     sort (sorted+1, sorted+1+len);
154     build (1, len, 0, 1);
155     printf ("Case %d:\n", ++ys);
156     long long sum1=0, sum2=0, sum3=0;
157     for (int i=1; i<=n; i++)
158     {
159         if (q[i].op==2)
160         {
161             sum1+=(long long)query (q[i].x, q[i].y, q[i].k, 0, 1);
162         }
163         else if (q[i].op==4)
164         {
165             sum3+=(long long)query (q[i].x, q[i].y, q[i].k, 0, 1);
166         }
167         else if (q[i].op==3)
168         {
169             sum2+=(long long)find (0, q[i].y, q[i].k, 1);
170         }
171     }
172     printf ("%lld\n%lld\n%lld\n", sum1, sum2, sum3);
173 }
174 return 0;
175 }

```

2.8 二维RMQ

```
1 #include<iostream>
2 #include<cstring>
3 #include<cstdio>
4 using namespace std;
5 const int M = 301;
6 int val[M][M];
7 int Max[9][9][M][M];
8 int idx[M];
9 void initRMQ(int n,int m){
10     for(int i=1;i<=n;i++){
11         for(int j=1;j<=m;j++){
12             Max[0][0][i][j]=val[i][j];
13         }
14     }
15     for(int i=0;i<=idx[n];i++){
16         int limit1=n+1-(1<<i);
17         for(int j=0;j<=idx[m];j++){
18             if(!i&&!j) continue;
19             int limit2=m+1-(1<<j);
20             for(int ii=1;ii<=limit1;ii++){
21                 for(int jj=1;jj<=limit2;jj++){
22                     if(i) Max[i][j][ii][jj]=min(Max[i-1][j][ii+(1<<i>>1)][jj],Max[i-1][j][ii][jj]);
23                     else Max[i][j][ii][jj]=min(Max[i][j-1][ii][jj],Max[i][j-1][ii][jj+(1<<j>>1)]);
24                 }
25             }
26         }
27     }
28 }
29 int query(int a,int b,int c,int d)
30 {
31     int n=idx[c-a+1],m=idx[d-b+1];
32     c--=(1<<n)-1;
33     d--=(1<<m)-1;
34     return min(min(Max[n][m][a][b],Max[n][m][a][d]),min(Max[n][m][c][b],Max[n][m][c][d]));
35 }
36 int main(void)
37 {
38     idx[0]=-1;
39     for(int i=1;i<=300;i++){
40         idx[i]=(i&(i-1))?idx[i-1]:idx[i-1]+1;
41     }
42     int T,n;
43     scanf("%d",&T);
44     while(T--){
45         scanf("%d",&n);
46         for(int i=1;i<=n;i++){
47             for(int j=1;j<=n;j++){
48                 scanf("%d",&val[i][j]);
49             }
50         }
51         initRMQ(n,n);
52         int a,b,c,d,m;
53         scanf("%d",&m);
54         while(m--){
55             scanf("%d%d%d%d",&a,&b,&c,&d);
56             printf("%d\n",query(a,b,c,d));
57         }
58     }
```

```
59 |     return 0;  
60 | }
```


2.9 主席树

```
1  #include<iostream>
2  #include<cstring>
3  #include<cstdio>
4  #include<algorithm>
5  using namespace std;
6  const int N = 200001,M=200001,NN=3000000;
7  struct Seg
8  {
9      int l,r,lson,rson,s;
10 }a[NN];
11 int root[N];
12 int b[N],v[N],now[N];
13 int m;
14 int bin(int x)
15 {
16     int left=1,right=m;
17     while(left<=right)
18     {
19         int mid=(left+right)>>1;
20         if(v[mid]==x)
21         {
22             return mid;
23         }
24         if(v[mid]<x) left=mid+1;
25         else right=mid-1;
26     }
27     return 0;
28 }
29 int tot=0;
30 int build(int l,int r)
31 {
32     int k=++tot;
33     a[k].l=l;a[k].r=r;a[k].s=0;
34     if(l==r)
35     {
36         return k;
37     }
38     int mid=(l+r)>>1;
39     a[k].lson=build(l,mid);
40     a[k].rson=build(mid+1,r);
41     return k;
42 }
43 int change(int p,int x,int w)
44 {
45     int k=++tot;
46     a[k].l=a[p].l;a[k].r=a[p].r;
47     a[k].lson=a[p].lson;a[k].rson=a[p].rson;
48     a[k].s=a[p].s+w;
49     if(a[k].l==a[k].r) return k;
50     int mid=(a[k].l+a[k].r)>>1;
51     if(mid<x)
52     {
53         a[k].rson=change(a[p].rson,x,w);
54     }
55     else
56     {
57         a[k].lson=change(a[p].lson,x,w);
58     }
59     return k;
60 }
61 int query(int p,int q,int k)
```

```

62 {
63     if(a[p].l==a[p].r) return a[p].l;
64     int now=a[a[q].lson].s-a[a[p].lson].s;
65     if(now>=k)
66     {
67         return query(a[p].lson,a[q].lson,k);
68     }
69     else
70     {
71         return query(a[p].rson,a[q].rson,k-now);
72     }
73 }
74 int main(void)
75 {
76     int n,q,x,y,k;
77     scanf ("%d%d",&n,&q);
78     for(int i=1;i<=n;i++)
79     {
80         scanf ("%d",&b[i]);
81         v[i]=b[i];
82     }
83     sort(v+1,v+1+n);
84     m=1;
85     for(int i=2;i<=n;i++)
86     {
87         if(v[i]!=v[i-1])
88         {
89             v[++m]=v[i];
90         }
91     }
92     for(int i=1;i<=n;i++)
93     {
94         now[i]=bin(b[i]);
95     }
96     root[0]=build(1,m);
97     for(int i=1;i<=n;i++)
98     {
99         root[i]=change(root[i-1],now[i],1);
100     }
101     for(int i=1;i<=q;i++)
102     {
103         scanf ("%d%d%d",&x,&y,&k);
104         printf ("%d\n",v[query(root[x-1],root[y],k)]);
105     }
106     return 0;
107 }

```

2.10 树链剖分

```
1 // #pragma comment(linker, "/STACK:1024000000,1024000000")
2 #include<iostream>
3 #include<cstring>
4 #include<cstdio>
5 #include<vector>
6 using namespace std;
7 const int N = 500010;
8 int sz[N], dep[N], fa[N], son[N], val[N], top[N];
9 int g[N], d[N];
10 vector<int> edge[N];
11 int loc=0;
12 void dfs(int x, int dp, int pa)
13 {
14     fa[x]=pa;
15     sz[x]=1;
16     dep[x]=dp;
17     son[x]=0;
18     for(int i=0; i<edge[x].size(); i++)
19     {
20         int u=edge[x][i];
21         if(u!=pa)
22         {
23             dfs(u, dp+1, x);
24             sz[x]+=sz[u];
25             if(sz[son[x]]<sz[u])
26             {
27                 son[x]=u;
28             }
29         }
30     }
31 }
32 void find_link(int x, int tp)
33 {
34     val[x]=++loc;
35     d[loc]=g[x];
36     top[x]=tp;
37     if(son[x]!=0)
38     {
39         find_link(son[x], tp);
40     }
41     for(int i=0; i<edge[x].size(); i++)
42     {
43         int u=edge[x][i];
44         if(u!=son[x]&&fa[u]==x)
45         {
46             find_link(u, u);
47         }
48     }
49 }
50 int add[N<<2], sum[N<<2];
51 void up(int idx)
52 {
53     sum[idx]=sum[idx<<1]+sum[idx<<1|1];
54 }
55 void down(int idx, int l)
56 {
57     if(add[idx]!=0)
58     {
59         add[idx<<1]+=add[idx];
60         add[idx<<1|1]+=add[idx];
61         sum[idx<<1]+=add[idx]*(1-(l>>1));
```

```

62         sum[idx<<1|1]+=add[idx]*(1>>1);
63         add[idx]=0;
64     }
65 }
66 void build(int left,int right,int idx)
67 {
68     add[idx]=0;
69     if(left==right)
70     {
71         sum[idx]=d[left];
72         return ;
73     }
74     int mid=(left+right)>>1;
75     build(left,mid,idx<<1);
76     build(mid+1,right,idx<<1|1);
77     up(idx);
78 }
79 void update(int left,int right,int L,int R,int v,int idx)
80 {
81     if(left>=L&&right<=R)
82     {
83         add[idx]+=v;
84         sum[idx]+=v*(right-left+1);
85         return ;
86     }
87     down(idx,right-left+1);
88     int mid=(left+right)>>1;
89     if(mid<L)
90     {
91         update(mid+1,right,L,R,v,idx<<1|1);
92     }
93     else if(mid>=R)
94     {
95         update(left,mid,L,R,v,idx<<1);
96     }
97     else
98     {
99         update(left,mid,L,mid,v,idx<<1);
100         update(mid+1,right,mid+1,R,v,idx<<1|1);
101     }
102     up(idx);
103 }
104 int query(int left,int right,int id,int idx)
105 {
106     if(left==right)
107     {
108         return sum[idx];
109     }
110     down(idx,right-left+1);
111     int mid=(left+right)>>1;
112     if(mid<id)
113     {
114         return query(mid+1,right,id,idx<<1|1);
115     }
116     else
117     {
118         return query(left,mid,id,idx<<1);
119     }
120 }
121 void modify(int va,int vb,int v)
122 {
123     int f1=top[va],f2=top[vb];
124     while(f1!=f2)

```

```

125     {
126         if (dep[f1] < dep[f2])
127         {
128             swap(f1, f2);
129             swap(va, vb);
130         }
131         update(1, loc, val[f1], val[va], v, 1);
132         va = fa[f1]; f1 = top[va];
133     }
134     if (dep[va] < dep[vb]) swap(va, vb);
135     update(1, loc, val[vb], val[va], v, 1);
136 }
137 int main(void)
138 {
139     int n, m, q, x, y, w;
140     char op[2];
141     while (scanf("%d%d%d", &n, &m, &q) != EOF)
142     {
143         for (int i = 1; i <= n; i++)
144         {
145             scanf("%d", &g[i]);
146             edge[i].clear();
147         }
148         for (int i = 1; i <= m; i++)
149         {
150             scanf("%d%d", &x, &y);
151             edge[x].push_back(y);
152             edge[y].push_back(x);
153         }
154         loc = 0;
155         dfs(1, 0, -1);
156         find_link(1, 1);
157         build(1, n, 1);
158         for (int i = 1; i <= q; i++)
159         {
160             scanf("%s", op);
161             if (op[0] == 'I')
162             {
163                 scanf("%d%d%d", &x, &y, &w);
164                 modify(x, y, w);
165             }
166             else if (op[0] == 'Q')
167             {
168                 scanf("%d", &x);
169                 int ans = query(1, n, val[x], 1);
170                 printf("%d\n", ans);
171             }
172             else
173             {
174                 scanf("%d%d%d", &x, &y, &w);
175                 modify(x, y, -w);
176             }
177         }
178     }
179 }
180 return 0;
181 }

```

2.11 KD树

```
1  #include<iostream>
2  #include<cstring>
3  #include<cstdio>
4  #include<algorithm>
5  #include<queue>
6  #include<vector>
7  using namespace std;
8  const int N = 100010;
9  const int oo = 20010;
10 int Div[N];
11 struct P
12 {
13     int a[5];
14     int num;
15     int dis;
16 }tt[N],p[N],tmp,tx;
17 int m,now,k;
18 int minn[10],maxx[10];
19 struct Pcmp
20 {
21     bool operator()(P a,P b)
22     {
23         return a.dis<b.dis;
24     }
25 };
26 priority_queue<P,vector<P>,Pcmp> que;
27 bool cmp(P a,P b)
28 {
29     return a.a[now]<b.a[now];
30 }
31 int dist(P a,P b)
32 {
33     int sum=0;
34     for(int i=0;i<k;i++)
35     {
36         sum+=(a.a[i]-b.a[i])*(a.a[i]-b.a[i]);
37     }
38     return sum;
39 }
40 void build(int l,int r,P p[])
41 {
42     if(l>r) return ;
43     int mid=(l+r)>>1;
44     for(int i=0;i<k;i++)
45     {
46         minn[i]=oo;
47         maxx[i]=-oo;
48     }
49     for(int i=1;i<=r;i++)
50     {
51         for(int j=0;j<k;j++)
52         {
53             minn[j]=min(minn[j],p[i].a[j]);
54             maxx[j]=max(maxx[j],p[i].a[j]);
55         }
56     }
57     now=0;
58     for(int i=1;i<=r;i++)
59     {
60         if(maxx[i]-minn[i]>maxx[now]-minn[now])
61         {
```

```

62         now=i;
63     }
64 }
65 Div[mid]=now;
66 nth_element(p+l,p+mid,p+r+1,cmp);
67 build(l,mid-1,p);
68 build(mid+1,r,p);
69 }
70 void find(int l,int r,P a,P p[])
71 {
72     if(l>r) return ;
73     int mid=(l+r)>>1;
74     int dis=dist(a,p[mid]);
75     if(queue.size()<m)
76     {
77         tx=p[mid];tx.dis=dis;
78         queue.push(tx);
79     }
80     else if(queue.top().dis>dis)
81     {
82         tx=p[mid];tx.dis=dis;
83         queue.pop();
84         queue.push(tx);
85     }
86     int d=a.a[Div[mid]]-p[mid].a[Div[mid]];
87     int l1=l,l2=mid+1,r1=mid-1,r2=r;
88     if(d>0) swap(l1,l2),swap(r1,r2);
89     find(l1,r1,a,p);
90     if(queue.size()<m||d*d<queue.top().dis) find(l2,r2,a,p);
91 }
92 int main(void)
93 {
94     int n,q;
95     //freopen("1009.in","r",stdin);
96     //freopen("1.out","w",stdout);
97     while(scanf("%d%d",&n,&k)!=EOF)
98     {
99         for(int i=1;i<=n;i++)
100         {
101             for(int j=0;j<k;j++)
102             {
103                 scanf("%d",&tt[i].a[j]);
104                 p[i].a[j]=tt[i].a[j];
105             }
106         }
107         build(1,n,p);
108         scanf("%d",&q);
109         while(q--)
110         {
111             for(int i=0;i<k;i++)
112             {
113                 scanf("%d",&tmp.a[i]);
114             }
115             scanf("%d",&m);
116             while(!queue.empty()) queue.pop();
117             vector<P> v;
118             find(1,n,tmp,p);
119             while(!queue.empty()) v.push_back(queue.top()),queue.pop();
120             printf("the closest %d points are:\n",m);
121             for(int i=v.size()-1;i>=0;i--)
122             {
123                 for(int j=0;j<k;j++)
124                 {

```

```

125 |                                     printf ("%d%c",v[i].a[j],j==k-1?'\n':' ');
126 |                                     }
127 |                                 }
128 |                             }
129 |                         }
130 |                     return 0;
131 | }

```


2.12 treap

```
1  #include<iostream>
2  #include<cstring>
3  #include<cstdio>
4  #include<cstdlib>
5  using namespace std;
6  const int N = 1000010;
7  struct Node
8  {
9      int val,fix,left,right;
10     int size,weight;
11 }tn[N];
12 int loc,root;
13 void init()
14 {
15     loc=0,root=0,srand(1992);
16 }
17 int newnode(int v)
18 {
19     ++loc;
20     tn[loc].left=0;tn[loc].right=0;
21     tn[loc].size=tn[loc].weight=1;
22     tn[loc].val=v;
23     tn[loc].fix=rand();
24     return loc;
25 }
26 void up(int x)
27 {
28     tn[x].size=tn[tn[x].left].size+tn[tn[x].right].size+tn[x].weight;
29 }
30 void lr(int &x)
31 {
32     int y=tn[x].right;
33     tn[x].right=tn[y].left;
34     tn[y].left=x;
35     up(x);
36     up(y);
37     x=y;
38 }
39 void rr(int &x)
40 {
41     int y=tn[x].left;
42     tn[x].left=tn[y].right;
43     tn[y].right=x;
44     up(x);
45     up(y);
46     x=y;
47 }
48 void insert(int &x,int v)
49 {
50     if(x==0){
51         x=newnode(v);
52         return ;
53     }
54     tn[x].size++;
55     if(tn[x].val==v){
56         tn[x].weight++;
57     }else if(tn[x].val<v){
58         insert(tn[x].right,v);
59         if(tn[tn[x].right].fix<tn[x].fix){
60             lr(x);
61         }
```

```

62     }else{
63         insert(tn[x].left,v);
64         if(tn[tn[x].left].fix<tn[x].fix){
65             rr(x);
66         }
67     }
68 }
69 bool contain(int x,int v)
70 {
71     if(!x){
72         return false;
73     }
74     if(tn[x].val==v){
75         return true;
76     }else if(tn[x].val<v){
77         return contain(tn[x].right,v);
78     }else{
79         return contain(tn[x].left,v);
80     }
81 }
82 void erase(int &x)
83 {
84     if(tn[x].left+tn[x].right==0){
85         x=0;
86     }else if(tn[x].left*tn[x].right==0){
87         x=tn[x].left+tn[x].right;
88     }else if(tn[tn[x].left].fix<tn[tn[x].right].fix){
89         rr(x);
90         erase(tn[x].right);
91     }else{
92         lr(x);
93         erase(tn[x].left);
94     }
95 }
96 void remove(int &x,int v)
97 {
98     if(!contain(x,v)){
99         return ;
100     }
101     tn[x].size--;
102     if(tn[x].val>v){
103         remove(tn[x].left,v);
104     }else if(tn[x].val<v){
105         remove(tn[x].right,v);
106     }else{
107         tn[x].weight--;
108         if(tn[x].weight==0){
109             erase(x);
110         }
111     }
112 }
113 int find_kth(int x,int v)
114 {
115     if(tn[x].size<v){
116         return -1;
117     }
118     if(tn[tn[x].left].size>=v){
119         return find_kth(tn[x].left,v);
120     }else if(tn[x].weight+tn[tn[x].left].size>=v){
121         return v;
122     }else{
123         return find_kth(tn[x].right,v-tn[x].weight-tn[tn[x].left].size);
124     }

```

```

125 | }
126 | void Debug(int x)
127 | {
128 |     if(tn[x].left!=0){
129 |         Debug(tn[x].left);
130 |     }
131 |     printf("%d的为dvalue%, 为dfix%, 左儿子为d%d右儿子
      |     为,%d\n",x,tn[x].val,tn[x].fix,tn[x].left,tn[x].right);
132 |     if(tn[x].right!=0){
133 |         Debug(tn[x].right);
134 |     }
135 | }
136 | int main(void)
137 | {
138 |     return 0;
139 | }

```

2.13 DancingLinks

2.13.1 精确覆盖

```
1  #include<iostream>
2  #include<cstring>
3  #include<cstdio>
4  #define N 1010
5  #define M N*N
6  int R[M],L[M],U[M],D[M];
7  int C[M],S[N],O[N],row[M];
8  int size;
9  int ak;
10 void remove(int c)
11 { //删除一列, 并删除同列覆盖的每行
12     int i,j;
13     L[R[c]]=L[c];
14     R[L[c]]=R[c];
15     for(i=D[c];i!=c;i=D[i]){
16         for(j=R[i];j!=i;j=R[j]){
17             U[D[j]]=U[j];
18             D[U[j]]=D[j];
19             S[C[j]]--;
20         }
21     }
22 }
23
24 void resume(int&c)
25 { //恢复一列及此列覆盖的行
26     int i,j;
27     for(i=U[c];i!=c;i=U[i]){
28         for(j=L[i];j!=i;j=L[j]){
29             U[D[j]]=D[U[j]]=j;
30             S[C[j]]++;
31         }
32     } R[L[c]]=L[R[c]]=c;
33 }
34 int Dfs(int k)
35 {
36     int min,c,i,j;
37     if(R[0]==0){ //得到结果
38         ak=k;return 1;
39     }
40
41     for(min=1000000,c=0,i=R[0];i!=0;i=R[i]){
42         if(S[i]<min) min=S[i],c=i; //选取列元素数最少的
43     }
44     remove(c);
45     for(i=D[c];i!=c;i=D[i]){
46
47         for(j=R[i];j!=i;j=R[j])
48             remove(C[j]); //删除
49
50         O[k]=row[i]; //记录结果
51         if(Dfs(k+1))
52             return 1;
53
54         for(j=L[i];j!=i;j=L[j])
55             resume(C[j]); //恢复
56     }
57     resume(c);
58     return 0;
59 }
```

```

60 int main(void)
61 {
62     int n,m,num,x;
63     while (scanf ("%d%d",&n,&m) != EOF)
64     {
65         memset(S, 0, sizeof(S));
66         for(int i=1;i<=m;i++)
67         {
68             R[i-1]=L[i+1]=U[i]=D[i]=i;
69         }
70         R[m]=0; L[0]=m;
71         size=m+1;
72         for(int i=1;i<=n;i++)
73         {
74             int rowh=-1;
75             scanf ("%d",&num);
76             for(int j=1;j<=num;j++)
77             {
78                 scanf ("%d",&x);
79                 C[size]=x;
80                 D[U[x]]=size;
81                 U[size]=U[x];
82                 D[size]=x;
83                 U[x]=size;
84                 S[x]++;
85                 row[size]=i;
86                 if(rowh==-1)
87                 {
88                     L[size]=R[size]=size;
89                     rowh=size;
90                 }
91                 else
92                 {
93                     R[size]=rowh;
94                     L[size]=L[rowh];
95                     R[L[rowh]]=size;
96                     L[rowh]=size;
97                 }
98                 size++;
99             }
100         }
101         int ans=Dfs(0);
102         if(ans==0)
103         {
104             printf("NO\n");
105         }
106         else
107         {
108             printf("%d",ak);
109             for(int i=0;i<ak;i++)
110             {
111                 printf(" %d",O[i]);
112             }
113             printf("\n");
114         }
115     }
116     return 0;
117 }

```

2.13.2 重复覆盖

```

1 #include<iostream>
2 #include<cstring>
3 #include<cstdio>

```

```

4  #include<climits>
5  using namespace std;
6  #define N 100
7  #define M N*N
8  int R[M],L[M],U[M],D[M];
9  int C[M],S[N],O[N],row[M];
10 int Col[M];
11 int size=0;
12 void remove(int &c) {
13     for(int i = D[c]; i != c ; i = D[i]) {
14         L[R[i]] = L[i];
15         R[L[i]] = R[i];
16     }
17 }
18 void resume(int &c) {
19     for(int i = U[c]; i != c ; i = U[i]) {
20         L[R[i]] = i;
21         R[L[i]] = i;
22     }
23 }
24 int h() {
25     bool hash[51];
26     memset(hash,false,sizeof(hash));
27     int ret = 0;
28     for(int c = R[0]; c != 0 ; c = R[c]) {
29         if(!hash[c]) {
30             ret ++;
31             hash[c] = true;
32             for(int i = D[c] ; i != c ; i = D[i]) {
33                 for(int j = R[i] ; j != i ; j = R[j]) {
34                     hash[Col[j]] = true;
35                 }
36             }
37         }
38     }
39     return ret;
40 }
41 int ans=INT_MAX;
42 void dfs(int deep) {
43     if(deep + h() >= ans) {
44         return ;
45     }
46     if(R[0] == 0) {
47         ans=min(ans,deep);
48         return ;
49     }
50     int idx , i , j , minnum = 99999;
51     for(i = R[0] ; i != 0 ; i = R[i]) {
52         if(S[i] < minnum) {
53             minnum = S[i];
54             idx = i;
55         }
56     }
57     for(i = D[idx]; i != idx; i = D[i]) {
58         remove(i);
59         for(j = R[i]; j != i ; j = R[j]) {
60             remove(j);
61         }
62         dfs(deep+1);
63         for(j = L[i]; j != i ; j = L[j]) {
64             resume(j);
65         }
66         resume(i);

```

```

67     }
68     return ;
69 }
70 int main(void)
71 {
72     int n,m,num,x;
73     scanf ("%d%d",&n,&m);
74     memset (S,0,sizeof (S));
75     for(int i=1;i<=n;i++){
76         R[i-1]=L[i+1]=U[i]=D[i]=i;
77     }
78     R[n]=0;L[0]=n;
79     size=n+1;
80     for(int i=1;i<=m;i++){
81         int rowh=-1;
82         scanf ("%d",&num);
83         for(int j=1;j<=num;j++){
84             scanf ("%d",&x);
85             D[U[x]]=size;
86             U[size]=U[x];
87             D[size]=x;
88             U[x]=size;
89             S[x]++;
90             row[size]=i;
91             Col[size]=x;
92             if(rowh==-1)
93             {
94                 L[size]=R[size]=size;
95                 rowh=size;
96             }
97             else
98             {
99                 R[size]=rowh;
100                L[size]=L[rowh];
101                R[L[rowh]]=size;
102                L[rowh]=size;
103            }
104            size++;
105        }
106    }
107    dfs(0);
108    printf ("%d\n",ans);
109    return 0;
110 }

```

3 字符串

3.1 最小表示法

返回的是最小表示法得到的起始值的位置（以0开始）

```
1 int minirepresent(char *s)
2 {
3     int len=strlen(s);
4     int k=0,i=0,j=1;
5     while(j<len&& k<len)
6     {
7         if(s[(i+k)%len]==s[(j+k)%len])
8         {
9             k++;
10        }
11        else
12        {
13            if(s[(i+k)%len]<s[(j+k)%len])
14            {
15                j+=k+1;
16            }
17            else
18            {
19                i=max(j,i+k+1);
20                j=i+1;
21            }
22            k=0;
23        }
24    }
25    return i;
26 }
```


3.2 manacher

可以求出以某位为中心的回文串的最长回文串的长度（若是偶数，则中心在'#'上），最后的结果为p[id]-1;

```
1 void manacher(char *s)
2 {
3     int len=strlen(s);
4     p[0]=1;p[1]=1;p[2]=2;
5     int id=2;
6     for(int i=3;i<len;i++)
7     {
8         int u=2*id-i;
9         if(p[u]+i<p[id]+id)
10        {
11            p[i]=p[u];
12        }
13        else
14        {
15            int j=p[id]+id-i;
16            while(i+j<len&& s[i+j]==s[i-j])
17            {
18                j++;
19            }
20            p[i]=j;
21            id=i;
22        }
23    }
24 }
25 void init(char *s,int len)
26 {
27     str[0]='$';
28     str[1]='#';
29     for(int i=0;i<len;i++)
30     {
31         str[i*2+2]=s[i];
32         str[i*2+3]='#';
33     }
34     str[len*2+2]='\0';
35 }
```

3.3 扩展KMP

next[i]表示以i为起点和自己串匹配的最长公共前缀

```
1 void get_next(char *s)
2 {
3     int len=strlen(s),id;
4     next[0]=len;
5     next[1]=0;
6     int k=0;
7     while(k+1<len&&s[k]==s[k+1])
8     {
9         k++;
10        next[1]++;
11    }
12    id=1;
13    for(int i=2;i<len;i++)
14    {
15        int u=i-id;
16        if(next[u]+i>=next[id]+id)
17        {
18            int j=next[id]+id-i;
19            if(j<0) j=0;
20            while(j+i<len&&s[j]==s[j+i]) j++;
21            next[i]=j;
22            id=i;
23        }
24        else
25        {
26            next[i]=next[u];
27        }
28    }
29 }
```

t是模式串, s是原串,extend[i]表示以i为起点与模式串最长公共前缀的长度

```
1 void solve(char *t,char *s)//
2 {
3     get_next(t);
4     int m=strlen(t);
5     int n=strlen(s);
6     int k=0;
7     while(k<min(n,m)&&t[k]==s[k])
8     {
9         k++;
10    }
11    extend[0]=k;
12    int id=0;
13    for(int i=1;i<n;i++)
14    {
15        int u=i-id;
16        if(i+next[u]<extend[id]+id)
17        {
18            extend[i]=next[u];
19        }
20        else
21        {
22            int j=extend[id]+id-i;
23            if(j<0) j=0;
24            while(j+i<n&&t[j]==s[j+i]) j++;
25            extend[i]=j;
26            id=i;
27        }
28    }
29 }
```

3.4 KMP

```
1 void get_next(char *s)
2 {
3     int len=strlen(s);
4     next[0]=-1;
5     for(int i=1;i<len;i++)
6     {
7         int u=next[i-1];
8         while(u>=0&& s[u+1]!=s[i])
9         {
10             u=next[u];
11         }
12         if(s[u+1]==s[i])
13         {
14             u++;
15         }
16         next[i]=u;
17     }
18 }
19 int solve(char *s,char *t)
20 {
21     get_next(t);
22     int sum=0;
23     int u=-1;
24     int n=strlen(s),m=strlen(t);
25     for(int i=0;i<n;i++){
26         while(u>=0&& s[i]!=t[u+1]){
27             u=next[u];
28         }
29         if(s[i]==t[u+1]){
30             u++;
31         }
32         if(u==m-1){
33             sum++;
34             u=next[u];
35         }
36     }
37     return sum;
38 }
```

3.5 AC自动机

```
1 #include<iostream>
2 #include<cstring>
3 #include<cstdio>
4 #include<algorithm>
5 const int MAX_NODE = 50*10000+10;
6 const int CHILD_NUM = 26;
7 struct Aho_Corasick
8 {
9     int trie[MAX_NODE][CHILD_NUM];
10    int word[MAX_NODE];
11    int fail[MAX_NODE];
12    int vis[MAX_NODE];
13    int Q[MAX_NODE];
14    int sw[128];
15    int sz;
16    void Initialize(){
17        fail[0]=0;
18        for(int i=0;i<CHILD_NUM;i++){
19            sw[i+(int)'a']=i;
20        }
21    }
22    void Reset(){
23        memset(trie[0],0,sizeof(trie[0]));
24        word[0]=0;
25        memset(vis,0,sizeof(vis));
26        memset(word,0,sizeof(word));
27        sz=0;
28    }
29    void Insert(char *s)
30    {
31        int r=0,len=strlen(s);
32        for(int i=0;i<len;i++){
33            int idx=sw[s[i]];
34            if(!trie[r][idx]){
35                trie[r][idx]=++sz;
36                for(int j=0;j<CHILD_NUM;j++){
37                    trie[sz][j]=0;
38                }
39                word[sz]=0;fail[sz]=0;vis[sz]=0;
40            }
41            r=trie[r][idx];
42        }
43        word[r]++;
44    }
45    void Build(){
46        int front=0,rear=0;
47        for(int i=0;i<CHILD_NUM;i++){
48            int u=trie[0][i];
49            if(u){
50                fail[u]=0;
51                Q[rear++]=u;
52            }
53        }
54        while(front<rear){
55            int u=Q[front++];
56            for(int i=0;i<CHILD_NUM;i++){
57                int v=trie[u][i];
58                if(v){
59                    fail[v]=trie[fail[u]][i];
60                    Q[rear++]=v;
61                }else{
```

```

62         trie[u][i]=trie[fail[u]][i];
63     }
64 }
65 }
66 }
67 int Work(char *s)
68 {
69     int n=strlen(s);
70     int sum=0;
71     int r=0;
72     for(int i=0;i<n;i++){
73         int idx=sw[s[i]];
74         r=trie[r][idx];
75         int u=r;
76         while(u!=0&&!vis[u]){
77             sum+=word[u];
78             vis[u]=1;
79             u=fail[u];
80         }
81     }
82     return sum;
83 }
84 }AC;
85 char str[60],s[1000010];
86 int main(void)
87 {
88     int T,n;
89     AC.Initialize();
90     scanf("%d",&T);
91     while(T--){
92         AC.Reset();
93         scanf("%d",&n);
94         for(int i=1;i<=n;i++){
95             scanf("%s",str);
96             AC.Insert(str);
97         }
98         AC.Build();
99         scanf("%s",s);
100         printf("%d\n",AC.Work(s));
101     }
102     return 0;
103 }

```

3.6 后缀数组

```
1  #include<iostream>
2  #include<cstring>
3  #include<cstdio>
4  #include<cmath>
5  #define lowbit(x) (x&(-x))
6  using namespace std;
7  const int N = 10000;
8  const int CH = 256;//字母表大小
9  char str[N];
10 int p[N],s[N],c[N],pn[N],cn[N],height[N],cnt[N];
11 int val[N];
12 void init()
13 {
14     int n=strlen(str);
15     for(int i=0;i<n;i++){
16         s[i]=(int)str[i];
17     }
18     s[n]=0;
19 }
20 void Debug(int a[],int n)
21 {
22     printf("/*****\n");
23     for(int i=0;i<n;i++){
24         {
25             printf("%d ",a[i]);
26         }
27     }
28     printf("/*****\n");
29 }
30 void build_sa(int s[],int n,int alphabet)//s[]为字符串转化的数组，n为字符串长度
    加1,alp为字母表大小
31 {
32     memset(cnt,0,sizeof(cnt));
33     for(int i=0;i<n;i++)
34         ++cnt[s[i]];
35     for(int i=1;i<alphabet;i++)
36         cnt[i]+=cnt[i-1];
37     for(int i=n-1;i>=0;i--)
38         p[--cnt[s[i]]]=i;
39     c[p[0]]=0;
40     int classes=1;
41     for(int i=1;i<n;i++)
42     {
43         if(s[p[i]]!=s[p[i-1]]) ++classes;
44         c[p[i]]=classes-1;
45     }
46     //第一次排序结束
47     for(int h=0;(1<=h)<n;h++)
48     {
49         int m=1<=h;
50         for (int i=0; i<n; ++i) {
51             pn[i] = p[i] - (1<=h);
52             if (pn[i] < 0) pn[i] += n;
53         }
54         memset(cnt,0,sizeof(cnt));
55         for(int i=0;i<n;i++)
56             ++cnt[c[pn[i]]];
57         for(int i=1;i<classes;i++)
58             cnt[i]+=cnt[i-1];
59         for(int i=n-1;i>=0;i--)
60             p[--cnt[c[pn[i]]]]=pn[i];
```

```

61         cn[p[0]]=0;
62         classes=1;
63         for(int i=1;i<n;i++)
64         {
65             if(c[p[i]]!=c[p[i-1]]||c[p[i]+m]!=c[p[i-1]+m])
66                 ++classes;
67             cn[p[i]]=classes-1;
68         }
69         memcpy(c,cn,n*sizeof(int));
70         if(classes==n) break;//有所优化
71     }
72 }
73 void get_lcp(int n,int s[])//大小为n,height从1开始到n为真正有用的值,height[i]表
    示sa[i]与sa[i-1]的最长公共前缀
74 {
75     memset(height,0,sizeof(height));
76     for(int i=0;i<n;i++){
77         if(c[i]==0) continue;
78         int st=max(height[c[i-1]]-1,0);
79         int j=i+st,k=p[c[i]-1]+st;
80         while(j<n&&k<n&&s[j]==s[k])
81         {
82             st++;j++;k++;
83         }
84         height[c[i]]=st;
85     }
86 }
87 void rmq_init(int n,int height[])
88 {
89     for(int i=1;i<=n;i++)
90     {
91         val[i]=height[i];
92     }
93     for(int i=1;i<=n;i++)
94     {
95         for(int j=i;j<=n;j+=lowbit(j))
96         {
97             val[j]=min(val[j],height[i]);
98         }
99     }
100 }
101
102 int get_rmq(int l,int r)
103 {
104     if(l>r) swap(l,r);
105     l++;
106     int ans=height[r];
107     while(r>=l)
108     {
109         if(r-lowbit(r)+1>=l)
110         {
111             ans=min(ans,val[r]);
112             r-=lowbit(r);
113         }
114         else
115         {
116             ans=min(ans,height[r]);
117             r--;
118         }
119     }
120     return ans;
121 }
122 int main(void)

```

```

123 {
124     int T;
125     scanf("%d",&T);
126     while(T--)
127     {
128         scanf("%s",str);
129         init();
130         int n=strlen(str);
131         build_sa(s,n+1,128);
132         get_lcp(n,s);
133         long long sum=0;
134         for(int i=0;i<=n;i++)
135         {
136             sum+=n-p[i]-height[i];
137         }
138         cout<<sum<<endl;
139         //~ Debug(p,n+1);
140         //~ Debug(c,n+1);
141     }
142     return 0;
143 }

```


4 图论

4.1 SPFA判负环

如果没有负环的话跑的超慢，解决办法是初始贪心初始化，每个点由周围的点更新一次

```
1 void spfa(int x)
2 {
3     if(flag) return ;
4     instack[x]=1;
5     for(int cur=first[x]; cur!=-1; cur=edge[cur].next)
6     {
7         int u=edge[cur].v;
8         if (dis[u]>dis[x]+edge[cur].val)
9         {
10             dis[u]=dis[x]+edge[cur].val;
11             if(!instack[u]) spfa(u);
12             else{
13                 flag=1;
14                 return ;
15             }
16         }
17     }
18     instack[x]=0;
19 }
```

4.2 tarjan

4.2.1 求SCC

```
1  #include<stdio.h>
2  #include<string.h>
3  #define N 10005
4  #define M 50005
5  int first[N],next[M],end[M],dfn[N],ins[N],low[N],stack[N],color[N],cnt[N],
    degree[N],top,c,indx;
6
7  void tarjan(int u)
8  {
9      int i,v;
10
11      stack[++top]=u;
12      ins[u]=1;
13      dfn[u]=low[u]=++indx;
14
15      for (i=first[u];i;i=next[i])
16      {
17          v=end[i];
18
19          if (dfn[v]==0)
20          {
21              tarjan(v);
22              if (low[u]>low[v])
23                  low[u]=low[v];
24          }
25
26          else if (ins[v]==1&&low[u]>dfn[v])
27              low[u]=dfn[v];
28      }
29
30      if (low[u]==dfn[u])
31      {
32          c++;
33
34          do
35          {
36              v=stack[top--];
37              color[v]=c;
38              cnt[c]++;
39              ins[v]=0;
40          }
41          while (u!=v);
42      }
43  }
44
45  void solve()
46  {
47      int i;
48
49      for (i=1;i<=n;i++)
50          if (dfn[i]==0)
51              tarjan(i);
52  }
```

4.2.2 求割点、点双联通分支

```
1  #define N 2000 // 注意缩点后新建的图的顶点数最大可能达到 2*n-1
2  #define M 600000
3  using namespace std;
```

```

4
5 struct Edge
6 {
7     int u,v;
8
9     Edge(){}
10    Edge(int a,int b):u(a),v(b){}
11 };
12
13 int first[N],next[M],end[M];
14 int cnt_edge;
15 int dfn[N],low[N],color[N];
16 int cut[N]; // cut[i]>0表示i是割点, 删掉i点后原图将分成cut[i]+1个联通块
17 Edge stack[M];
18 int idx,c,b,top;
19 vector<int>dpt[N]; // 保存每个点双联通分支中的点。注意割点会包含在多个点双联通分支中
20 int qfir[N],qnex[M],qend[M]; // 保存缩点后的图
21 int cnt_Qedge;
22
23 void addEdge(int u,int v)
24 {
25     end[cnt_edge]=v;
26     next[cnt_edge]=first[u];
27     first[u]=cnt_edge++;
28 }
29
30 void addQedge(int u,int v)
31 {
32     qend[cnt_Qedge]=v;
33     qnex[cnt_Qedge]=qfir[u];
34     qfir[u]=cnt_Qedge++;
35 }
36
37 void tarjan(int u)
38 {
39     int i,v,flag;
40     Edge e;
41
42     dfn[u]=low[u]=++idx;
43     flag=0;
44
45     for (i=first[u];i;i=next[i])
46     {
47         v=end[i];
48
49         if (dfn[v]==0)
50         {
51             stack[++top]=Edge(u,v);
52
53             tarjan(v);
54
55             low[u]=min(low[u],low[v]);
56
57             if (dfn[u]<=low[v])
58             {
59                 cut[u]++;
60                 c++;
61
62                 do
63                 {
64                     e=stack[top--];
65
66                     if (color[e.u]!=c)

```

```

67         {
68             color[e.u]=c;
69             dpt[c].push_back(e.u);
70         }
71         if (color[e.v]!=c)
72         {
73             color[e.v]=c;
74             dpt[c].push_back(e.v);
75         }
76
77         }while (e.u!=u||e.v!=v);
78     }
79 }
80 else
81 {
82     low[u]=min(low[u],dfn[v]);
83     if (dfn[u]>dfn[v])
84         stack[++top]=Edge(u,v);
85 }
86 }
87 }
88
89 main()
90 {
91     int n,m,i,j,u,v;
92
93     while (scanf("%d%d",&n,&m)!=EOF)
94     {
95         memset(first,0,sizeof(first));
96         memset(qfir,0,sizeof(qfir));
97         memset(dfn,0,sizeof(dfn));
98         memset(cut,0,sizeof(cut));
99         memset(color,0,sizeof(color));
100         cnt_edge=1;
101         cnt_Qedge=1;
102         idx=top=c=0;
103
104         for (i=0;i<N;i++)
105             dpt[i].clear();
106
107         for (i=0;i<m;i++)
108         {
109             scanf("%d%d",&u,&v);
110             addEdge(u,v);
111             addEdge(v,u);
112         }
113
114         for (i=1;i<=n;i++)
115             if (dfn[i]==0)
116             {
117                 tarjan(i);
118                 cut[i]--;
119             }
120
121         b=c;
122
123         for (i=1;i<=n;i++)
124             if (cut[i])
125                 color[i]=++c; // 缩点后新图中割点形成的点的编号>b, 联通分支形成的点的
                                // 编号<=b
126
127         for (i=1;i<=b;i++)
128             for (j=0;j<dpt[i].size();j++)

```

```

129         {
130             u=dpt[i][j];
131
132             if (cut[u])
133             {
134                 addQedge(color[u],i); // 缩点建图，每个割点形成的点向周围联通分量
                                     形成的点双向连边
135                 addQedge(i,color[u]);
136             }
137         }
138
139         for (i=1;i<=b;i++)
140         {
141             for (j=0;j<dpt[i].size();j++)
142                 printf("%d ",dpt[i][j]);
143             printf("\n");
144         }
145     }
146
147     return 0;
148 }

```

4.2.3 求桥

```

1  #include<stdio.h>
2  #include<string.h>
3  #define N 5005
4  #define M 20005
5
6  typedef struct
7  {
8      int u,v;
9  }Edge;
10
11  Edge bridge[M];
12  int first[N],next[M],end[M],dfn[N],low[N],degree[N],father[N];
13  int idx,cnt;
14
15  void reset(int n)
16  {
17      int i;
18
19      for (i=0;i<=n;i++)
20          father[i]=i;
21  }
22
23  int find(int a)
24  {
25      if (father[a]==a)
26          return a;
27      else
28          return father[a]=find(father[a]);
29  }
30
31  void merge(int a,int b)
32  {
33      int fa=find(a);
34      int fb=find(b);
35
36      if (fa!=fb)
37          father[fa]=fb;
38  }
39
40  int tarjan(int u,int fu)

```

```

41 {
42     int i,v,flag;
43
44     dfn[u]=low[u]=++idx;
45     flag=0;
46
47     for (i=first[u];i;i=next[i])
48     {
49         v=end[i];
50
51         if (flag==0&&v==fu)    // 处理返祖边的影响
52         {
53             flag=1;
54             continue;
55         }
56
57         if (dfn[v]==0)
58         {
59             tarjan(v,u);
60
61             if (low[u]>low[v])
62                 low[u]=low[v];
63
64             if (dfn[u]<low[v])    //dfn[u]<low[v]说明u v为桥
65             {
66                 bridge[cnt].u=u;
67                 bridge[cnt].v=v;
68                 cnt++;
69             }
70             else
71                 merge(u,v);
72         }
73         else if (low[u]>dfn[v])
74             low[u]=dfn[v];
75     }
76 }
77
78 main()
79 {
80     int n,m,i,u,v,ans;
81
82     scanf("%d%d",&n,&m);
83
84     memset(first,0,sizeof(first));
85     memset(dfn,0,sizeof(dfn));
86     memset(degree,0,sizeof(degree));
87     reset(n);
88     cnt=idx=0;
89
90     for (i=1;i<=m+m;)
91     {
92         scanf("%d%d",&u,&v);
93         end[i]=v;
94         next[i]=first[u];
95         first[u]=i;
96         i++;
97         end[i]=u;
98         next[i]=first[v];
99         first[v]=i;
100        i++;
101    }
102
103    tarjan(1,0);

```

```

104
105     for (i=0;i<cnt;i++)
106     // 求最少需要添加多少条边, 使原图无桥: 缩点后度为1 (叶子) 的点的个数为ans, 则需添
        加(ans+1)/2条边。
107     {
108         u=bridge[i].u;
109         v=bridge[i].v;
110
111         degree[find(u)]++;
112         degree[find(v)]++;
113     }
114
115     ans=0;
116
117     for (i=1;i<=n;i++)
118         if (degree[i]==1)
119             ans++;
120
121     printf("%d\n", (ans+1)/2);
122
123     return 0;
124 }

```

4.2.4 求LCA

```

1  #include<stdio.h>
2  #include<string.h>
3  #define N 1000
4  #define M 1000000
5  int father[N],mark[N],res[N],ans[N];
6  int first[N],next[N],end[N],degree[N],qhead[N],qnext[M],qtail[M],qnum[M];
7  int find(int a)
8  {
9      if (father[a]==a)
10         return a;
11     else
12         return father[a]=find(father[a]);
13 }
14
15 void merge(int a,int b)
16 {
17     father[b]=a;
18 }
19
20 void LCA(int u)
21 {
22     int i,v,num;
23
24     father[u]=u;
25
26     for (i=first[u];i;i=next[i])
27     {
28         v=end[i];
29         LCA(v);
30         merge(u,v);
31     }
32
33     mark[u]=1;
34     for (i=qhead[u];i;i=qnext[i])
35     {
36         v=qtail[i];
37         num=qnum[i];
38

```

```

39 |         if (mark[v]==1)
40 |         {
41 |             ans[num]=find(v);
42 |             res[find(v)]++;
43 |         }
44 |     }
45 | }

```


4.3 网络流

4.3.1 EK

```
1  # include <cstring>
2  # include <cstdio>
3  # include <cstdlib>
4  # include <iostream>
5  # include <queue>
6  # define N 20500
7  # define M 4000500
8  # define oo 200000000
9  using namespace std;
10 struct EDGE
11 {
12     int val,v,next;
13 }edge[M];
14 struct DOT
15 {
16     int x,y,c;
17 }dot[110];
18 int vis[110],map[110][110];
19 int first[N],prev[N],curedge[N];
20 int m,w,n,d,cnt_edge;
21 void addedge(int u,int v,int c)
22 {
23     edge[cnt_edge].next=first[u];
24     edge[cnt_edge].v=v;
25     edge[cnt_edge].val=c;
26     first[u]=cnt_edge++;
27
28     edge[cnt_edge].next=first[v];
29     edge[cnt_edge].v=u;
30     edge[cnt_edge].val=0;
31     first[v]=cnt_edge++;
32 }
33 int sqr(int x)
34 {
35     return x*x;
36 }
37 int EK_MaxFlow(int st,int en)
38 {
39     int maxflow=0;
40     queue<int>q;
41     while (1)
42     {
43         while (!q.empty())
44             q.pop();
45         memset(prev,-1,sizeof(prev));
46         q.push(st);
47         curedge[st]=oo;
48         while (!q.empty())
49         {
50             int x=q.front();
51             q.pop();
52             for (int cur=first[x];cur!=-1;cur=edge[cur].next)
53             {
54                 int v=edge[cur].v;
55                 if (prev[v]==-1 && edge[cur].val)
56                 {
57                     prev[v]=x;
58                     curedge[v]=cur;
59                     q.push(v);
60                 }
```

```

61         }
62         if (prev[en] != -1)
63             break;
64     }
65     if (prev[en] == -1)
66         return maxflow;
67     int minflow = oo;
68     for (int v = en; v != st; v = prev[v])
69         minflow = min(minflow, edge[curedge[v]].val);
70     for (int v = en; v != st; v = prev[v])
71     {
72         edge[curedge[v]].val -= minflow;
73         edge[curedge[v]^1].val += minflow;
74     }
75     maxflow += minflow;
76 }
77 return maxflow;
78 }
79 int bfs(void)
80 {
81     queue<int> q;
82     int x;
83     q.push(0);
84     while (!q.empty())
85     {
86         x = q.front();
87         q.pop();
88         for (int i = 1; i <= n + 1; i++)
89             if (map[x][i] && !vis[i])
90             {
91                 vis[i] = 1;
92                 q.push(i);
93             }
94     }
95     return vis[n + 1];
96 }
97 int main (void)
98 {
99     cin >> n >> m >> d >> w;
100     for (int i = 1; i <= n; i++)
101         scanf("%d%d%d", &dot[i].x, &dot[i].y, &dot[i].c);
102     if (d >= w)
103     {
104         printf("1\n");
105         return 0;
106     }
107     for (int i = 1; i <= n; i++)
108     {
109         if (dot[i].c == 0)
110             continue;
111         if (dot[i].y <= d)
112             map[0][i] = 1;
113         if (w - dot[i].y <= d)
114             map[i][n + 1] = 1;
115         for (int j = i; j <= n; j++)
116             if (dot[j].c)
117                 if (sqr(dot[i].x - dot[j].x) + sqr(dot[i].y - dot[j].y) <= d * d)
118                     map[i][j] = map[j][i] = 1;
119     }
120     if (!bfs())
121     {
122         printf("IMPOSSIBLE\n");
123         return 0;

```

```

124     }
125     cnt_edge=0;
126     memset(first,-1,sizeof(first));
127     int st=0,en=1,tot=1,ans=0,cnt=0;
128     for (;cnt<m;cnt+=EK_MaxFlow(st,en))
129     {
130         ans++;
131         for (int i=1;i<=n;i++)
132         {
133             if (dot[i].c==0)
134                 continue;
135             addedge(i+tot,i+n+tot,dot[i].c);
136             if (ans>1)
137             {
138                 for (int j=1;j<=n;j++)
139                     if (map[j][i])
140                         addedge(tot-n+j,tot+i,dot[i].c);
141                 if (map[i][n+1])
142                     addedge(tot-n+i,en,dot[i].c);
143             }
144             if (map[0][i])
145                 addedge(st,i+tot,dot[i].c);
146         }
147         tot+=n+n;
148     }
149     cout<<ans<<endl;
150     return 0;
151 }

```

4.3.2 Sap

```

1  # include <cstring>
2  # include <cstdlib>
3  # include <cstdio>
4  # include <iostream>
5  using namespace std;
6  # define M 485000
7  # define N 20500
8  # define oo 200000000
9  struct EDGE
10 {
11     int v,val,next;
12 }edge[M];
13 int first[N],dis[N],gap[N],pre[N],cur[N];
14 int cnt_edge,NP0;
15 inline void addedge(int u,int v,int c1,int c2)
16 {
17     edge[cnt_edge].next=first[u];
18     edge[cnt_edge].v=v;
19     edge[cnt_edge].val=c1;
20     first[u]=cnt_edge++;
21
22     edge[cnt_edge].next=first[v];
23     edge[cnt_edge].v=u;
24     edge[cnt_edge].val=c2;
25     first[v]=cnt_edge++;
26 }
27 int Sap(int st,int en)
28 {
29     memset(dis,0,sizeof(int)*(NP0+1));
30     memset(gap,0,sizeof(int)*(NP0+1));
31     for (int i=0;i<NP0;i++)
32         cur[i]=first[i];
33     int u=pre[st]=st,maxflow=0,aug=oo;

```

```

34     gap[0]=NPO;
35     while (dis[st]<NPO)
36     {
37 loop:   for (int &i=cur[u];i!=-1;i=edge[i].next)
38         {
39             int v=edge[i].v;
40             if (edge[i].val && dis[u]==dis[v]+1)
41             {
42                 aug=min(aug,edge[i].val);
43                 pre[v]=u;
44                 u=v;
45                 if (v==en)
46                 {
47                     maxflow+=aug;
48                     for (u=pre[u];v!=st;u=pre[u])
49                     {
50                         edge[cur[u]].val-=aug;
51                         edge[cur[u]^1].val+=aug;
52                         v=u;
53                     }
54                     aug=0;
55                 }
56                 goto loop;
57             }
58         }
59     int mindis=NPO;
60     for (int i=first[u];i!=-1;i=edge[i].next)
61     {
62         int v=edge[i].v;
63         if (edge[i].val && mindis>dis[v])
64         {
65             cur[u]=i;
66             mindis=dis[v];
67         }
68     }
69     if (--gap[dis[u]]==0)
70         break;
71     gap[dis[u]=mindis+1]++;
72     u=pre[u];
73 }
74 return maxflow;
75 }
76 int main (void)
77 {
78     int n,m,x,y,z;
79     cin>>n>>m;
80     int st=0,en=n+1;
81     NPO=en+1;
82     memset(first,-1,sizeof(first));
83     cnt_edge=0;
84     for (int i=1;i<=n;i++)
85     {
86         scanf("%d%d",&x,&y);
87         addedge(st,i,x,0);
88         addedge(i,en,y,0);
89     }
90     for (int i=1;i<=m;i++)
91     {
92         scanf("%d%d%d",&x,&y,&z);
93         addedge(x,y,z,z);
94     }
95     cout<<Sap(st,en)<<endl;
96     return 0;

```

97 | }

4.3.3 sap邻接表

```
1 //memset(first,-1,sizeof(first));
2 //cnt_edge=0
3 struct EDGE
4 {
5     int v,val;
6 }edge[M];
7 int next[M];
8 int first[N],d[N],gap[N];
9 int cnt_edge,NPO,ok;
10 void addedge(LOL u,LOL v,LOL c)
11 {
12     next[cnt_edge]=first[u];
13     edge[cnt_edge].v=v;
14     edge[cnt_edge].val=c;
15     first[u]=cnt_edge++;
16
17     next[cnt_edge]=first[v];
18     edge[cnt_edge].v=u;
19     edge[cnt_edge].val=0;
20     first[v]=cnt_edge++;
21 }
22 LOL MIN(LOL x,LOL y)
23 {
24     return x>y?y:x;
25 }
26 LOL relable(LOL x)
27 {
28     LOL mm=bignum;
29     for (int i=first[x];i!=-1;i=next[i])
30         if (edge[i].val)
31             mm=MIN(mm,d[edge[i].v]+1);
32     return mm==bignum?NPO:mm;
33 }
34 LOL solve(int cur ,int t,LOL min)
35 {
36     if (cur==t && min!=bignum)
37         return min;
38     for (int i=first[cur];i!=-1;i=next[i])
39     {
40         int u=edge[i].v;
41         if (edge[i].val && d[cur]==d[u]+1)
42         {
43             LOL p=MIN(solve(u,t,MIN(min,edge[i].val)),min);
44             if (p)
45             {
46                 edge[i].val-=p;
47                 edge[i^1].val+=p;
48                 return p;
49             }
50         }
51     }
52     LOL x=relable(cur);
53     gap[x]++;
54     if (--gap[d[cur]]==0)
55         ok=1;
56     d[cur]=x;
57     return 0;
58 }
59 int maxflow(int s,int t)
60 {
```

```

61     int ans=0;
62     ok=0;
63     memset(gap,0,sizeof(gap));
64     memset(d,0,sizeof(d));
65     while (d[s]<NPO && !ok)
66         ans+=solve(s,t,bignum);
67     return ans;
68 }

```

4.3.4 费用流

```

1  struct EDGE
2  {
3      int val,v,cost;
4  }edge[M];
5  int vis[N],pv[N],pe[N],q[N],dis[N],first[N];
6  int next[M];
7  int NPO,cnt_edge;
8  int MIN(int x,int y)
9  {
10     return x>y?y:x;
11 }
12 void addedge(int u,int v,int c,int cost)
13 {
14     next[cnt_edge]=first[u];
15     edge[cnt_edge].val=c;
16     edge[cnt_edge].v=v;
17     edge[cnt_edge].cost=cost;
18     first[u]=cnt_edge++;
19
20     next[cnt_edge]=first[v];
21     edge[cnt_edge].val=0;
22     edge[cnt_edge].v=u;
23     edge[cnt_edge].cost=-cost;
24     first[v]=cnt_edge++;
25 }
26 int mincost (int s,int t)
27 {
28     int x,u,cur;
29     int head,tail;
30     int flow,cost,min;
31     for (flow=0,cost=0;;)
32     {
33         memset(pv,-1,sizeof(int)*(NPO+10));
34         memset(vis,0,sizeof(int)*(NPO+10));
35         for (int i=0;i<=NPO;i++)
36             dis[i]=bignum;
37         dis[s]=0;
38         q[1]=s;
39         vis[s]=1;
40         for (head=0,tail=1;head++<tail;)
41         {
42             x=q[head%N];
43             vis[x]=0;
44             for (cur=first[x];cur!=-1;cur=next[cur])
45             {
46                 u=edge[cur].v;
47                 if (edge[cur].val && dis[x]+edge[cur].cost<dis[u])
48                 {
49                     dis[u]=dis[x]+edge[cur].cost;
50                     if (!vis[u])
51                     {
52                         vis[u]=1;
53                         q[(++tail)%N]=u;

```

```

54         }
55         pv[u]=x;
56         pe[u]=cur;
57     }
58 }
59 }
60 if (pv[t]==-1)
61     break;
62 for (cur=t,min=bignum;cur!=s;cur=pv[cur])
63     min=MIN(min,edge[pe[cur]].val);
64 for (cur=t;cur!=s;cur=pv[cur])
65 {
66     edge[pe[cur]].val-=min;
67     edge[pe[cur]^1].val+=min;
68 }
69 flow+=min;
70 cost+=dis[t]*min;
71 }
72 return cost;
73 }

```

4.3.5 无源无汇上下界网络流

z oj 2314

主要思想：每一个点流进来的流=流出去的流

对于每一个点i，令

$M_i = \text{sum}(i\text{点所有流进来的下界流}) - \text{sum}(i\text{点所有流出去的下界流})$

如果 M_i 大于0，代表此点必须还要流出去 M_i 的自由流，那么我们从源点连一条 M_i 的边到该点。

如果 M_i 小于0，代表此点必须还要流进来 M_i 的自由流，那么我们从该点连一条 M_i 的边到汇点。

如果求S到T的最大流，看是否满流(S的相邻边都流满)

```

1  # include <cstring>
2  # include <cstdlib>
3  # include <cstdio>
4  # include <iostream>
5  using namespace std;
6  # define N 400
7  # define M 500000
8  # define oo 2000000000
9  struct EDGE
10 {
11     int v,val,next;
12 }edge[M];
13 int low[M];
14 int w[N],first[N],dis[N],gap[N],preV[N],curE[N];
15 int cnt_edge,NPO;
16 inline void addedge(int u,int v,int c)
17 {
18     edge[cnt_edge].next=first[u];
19     edge[cnt_edge].v=v;
20     edge[cnt_edge].val=c;
21     first[u]=cnt_edge++;
22
23     edge[cnt_edge].next=first[v];
24     edge[cnt_edge].v=u;
25     edge[cnt_edge].val=0;
26     first[v]=cnt_edge++;
27 }
28 int Sap(int st,int en)
29 {
30     memset(dis,0,sizeof(int)*(NPO+1));
31     memset(gap,0,sizeof(int)*(NPO+1));
32     for (int i=0;i<NPO;i++)
33         curE[i]=first[i];
34     int u=preV[st]=st,maxflow=0,minflow=oo;

```

```

35 gap[0]=NPO;
36 while (dis[st]<NPO)
37 {
38     loop:
39     for (int &i=curE[u];i!=-1;i=edge[i].next)
40     {
41         int v=edge[i].v;
42         if (edge[i].val && dis[u]==dis[v]+1)
43         {
44             minflow=min(minflow,edge[i].val);
45             preV[v]=u;
46             u=v;
47             if (v==en)
48             {
49                 maxflow+=minflow;
50                 for (u=preV[u];v!=st;u=preV[u])
51                 {
52                     edge[curE[u]].val-=minflow;
53                     edge[curE[u]^1].val+=minflow;
54                     v=u;
55                 }
56                 minflow=oo;
57             }
58             goto loop;
59         }
60     }
61     int mindis=NPO;
62     for (int i=first[u];i!=-1;i=edge[i].next)
63     {
64         int v=edge[i].v;
65         if (edge[i].val && mindis>dis[v])
66         {
67             curE[u]=i;
68             mindis=dis[v];
69         }
70     }
71     if (--gap[dis[u]]==0)
72         break;
73     gap[dis[u]=mindis+1]++;
74     u=preV[u];
75 }
76 return maxflow;
77 }
78 int main (void)
79 {
80     int t,n,m;
81     cin>>t;
82     while (t--)
83     {
84         cin>>n>>m;
85         int u,v,l,h;
86         cnt_edge=0;
87         memset(first,-1,sizeof(first));
88         memset(w,0,sizeof(w));
89         int st=0,en=n+1,sum=0;
90         NPO=en+1;
91         for (int i=1;i<=m;i++)
92         {
93             scanf("%d%d%d%d",&u,&v,&l,&h);
94             addedge(u,v,h-1);
95             low[i]=1;
96             w[v]+=1;
97             w[u]-=1;

```



```

98     }
99     for (int i=1;i<=n;i++)
100         if (w[i]<0)
101         {
102             sum-=w[i];
103             addedge(i,en,-w[i]);
104         }
105         else
106             addedge(st,i,w[i]);
107     if (Sap(st,en)<sum)
108         printf("NO\n");
109     else
110     {
111         printf("YES\n");
112         for (int i=1;i<=m;i++)
113             printf("%d\n",edge[i+i-1].val+low[i]);
114     }
115 }
116 return 0;
117 }

```

4.3.6 有源有汇上下界网络流

poj_2396

有源有汇上下界网络流

由汇向源连一条 $(0, \infty)$ 的边，下界为0，上界为 ∞

然后转化成无源无汇上下界网络流问题

```

1  # include <cstring>
2  # include <cstdlib>
3  # include <cstdio>
4  # include <iostream>
5  # include <algorithm>
6  # include <queue>
7  # include <vector>
8  # include <map>
9  # include <cmath>
10 # include <time.h>
11 # include <set>
12 using namespace std;
13 # define N 400
14 # define M 500000
15 # define oo 2000000
16 struct EDGE
17 {
18     int v,val,next;
19 }edge[M];
20 int w[N],first[N],dis[N],gap[N],preV[N],curE[N];
21 int low[N][N],high[N][N],num[N][N];
22 int row[N],col[N];
23 int cnt_edge,NPO,flag;
24 inline void addedge(int u,int v,int c)
25 {
26     edge[cnt_edge].next=first[u];
27     edge[cnt_edge].v=v;
28     edge[cnt_edge].val=c;
29     first[u]=cnt_edge++;
30
31     edge[cnt_edge].next=first[v];
32     edge[cnt_edge].v=u;
33     edge[cnt_edge].val=0;
34     first[v]=cnt_edge++;
35 }

```

```

36 int Sap(int st,int en)
37 {
38     memset(dis,0,sizeof(int)*(NPO+1));
39     memset(gap,0,sizeof(int)*(NPO+1));
40     for (int i=0;i<NPO;i++)
41         curE[i]=first[i];
42     int u=preV[st]=st,maxflow=0,minflow=oo;
43     gap[0]=NPO;
44     while (dis[st]<NPO)
45     {
46         loop:
47         for (int &i=curE[u];i!=-1;i=edge[i].next)
48         {
49             int v=edge[i].v;
50             if (edge[i].val && dis[u]==dis[v]+1)
51             {
52                 minflow=min(minflow,edge[i].val);
53                 preV[v]=u;
54                 u=v;
55                 if (v==en)
56                 {
57                     maxflow+=minflow;
58                     for (u=preV[u];v!=st;u=preV[u])
59                     {
60                         edge[curE[u]].val-=minflow;
61                         edge[curE[u]^1].val+=minflow;
62                         v=u;
63                     }
64                     minflow=oo;
65                 }
66                 goto loop;
67             }
68         }
69         int mindis=NPO;
70         for (int i=first[u];i!=-1;i=edge[i].next)
71         {
72             int v=edge[i].v;
73             if (edge[i].val && mindis>dis[v])
74             {
75                 curE[u]=i;
76                 mindis=dis[v];
77             }
78         }
79         if (--gap[dis[u]]==0)
80             break;
81         gap[dis[u]=mindis+1]++;
82         u=preV[u];
83     }
84     return maxflow;
85 }
86 void update(int x,int y,char ch,int z)
87 {
88     if (ch=='=')
89     {
90         if (!(low[x][y]<=z && high[x][y]>=z))
91             flag=0;
92         low[x][y]=high[x][y]=z;
93     }
94     else
95     if (ch=='<')
96     {
97         high[x][y]=min(high[x][y],z-1);
98         if (low[x][y]>high[x][y])

```

```

99         flag=0;
100     }
101     else
102     if (ch=='>')
103     {
104         low[x][y]=max(low[x][y],z+1);
105         if (low[x][y]>high[x][y])
106             flag=0;
107     }
108 }
109 int main (void)
110 {
111     int t,n,m,ys=0;
112     //~ freopen("a.in","r",stdin);
113     //~ freopen("a.out","w",stdout);
114     cin>>t;
115     while (t--)
116     {
117         int sum1=0,sum2=0;
118         cin>>n>>m;
119         for (int i=1;i<=n;i++)
120         {
121             cin>>row[i];
122             sum1+=row[i];
123         }
124         for (int i=1;i<=m;i++)
125         {
126             cin>>col[i];
127             sum2+=col[i];
128         }
129         for (int i=1;i<=n;i++)
130             for (int j=1;j<=m;j++)
131                 low[i][j]=0,high[i][j]=oo;
132         int q,x,y,z;
133         flag=1;
134         char s[10];
135         cin>>q;
136         while (q--)
137         {
138             cin>>x>>y>>s>>z;
139             if (x==0 && y==0)
140             {
141                 for (int i=1;i<=n;i++)
142                     for (int j=1;j<=m;j++)
143                         update(i,j,s[0],z);
144             }
145             else
146             if (x==0)
147             {
148                 for (int i=1;i<=n;i++)
149                     update(i,y,s[0],z);
150             }
151             else
152             if (y==0)
153             {
154                 for (int i=1;i<=m;i++)
155                     update(x,i,s[0],z);
156             }
157             else
158                 update(x,y,s[0],z);
159         }
160         if (ys)
161             cout<<endl;

```

```

162     else ys=1;
163     if (sum1!=sum2 || !flag)
164         cout<<"IMPOSSIBLE"<<endl;
165     else
166     {
167         int st=0,en=n+m+1,cnt=0;
168         int ss=en+1,ee=en+2;
169         NPO=en+3;
170         cnt_edge=0;
171         memset(first,-1,sizeof(first));
172         memset(w,0,sizeof(w));
173         for (int i=1;i<=n;i++)
174             for (int j=1;j<=m;j++)
175             {
176                 addedge(i,n+j,high[i][j]-low[i][j]);
177                 w[n+j]+=low[i][j];
178                 w[i]-=low[i][j];
179                 num[i][j]=++cnt;
180             }
181         for (int i=1;i<=n;i++)
182         {
183             addedge(st,i,0);
184             w[i]+=row[i];
185             w[st]-=row[i];
186         }
187         for (int i=1;i<=m;i++)
188         {
189             addedge(i+n,en,0);
190             w[en]+=col[i];
191             w[i+n]-=col[i];
192         }
193         //add a oo edge from en to st
194         addedge(en,st,oo);
195
196         int sum=0;
197         for (int i=0;i<=n+m+1;i++)
198         {
199             if (w[i]>0)
200             {
201                 addedge(ss,i,w[i]);
202                 sum+=w[i];
203             }
204             else
205                 addedge(i,ee,-w[i]);
206         }
207         if (Sap(ss,ee)!=sum)
208             cout<<"IMPOSSIBLE"<<endl;
209         else
210         {
211             for (int i=1;i<=n;i++)
212                 for (int j=1;j<=m;j++)
213                     printf("%d%c",low[i][j]+edge[num[i][j]+num[i][j]-1].
214                             val,j==m?'':' ');
215         }
216     }
217     return 0;
218 }

```

4.4 2-sat

2-sat模版

poj_3683

如果 i,j 矛盾, 则`addege(i,j+n),addege(j,i+n)`

$i,j+n$ 矛盾 `addege(i,j),addege(j+n,i+n)`

$i+n,j$ 矛盾 `addege(i+n,j+n),addege(j,i)`

$i+n,j+n$ 矛盾 `addege(i+n,j),addege(j+n,i)`

`pd()`调用2-sat判断是否合法

调用`initialize()`初始化

`print_solution()`输出方案, 如果`ans[i]==1`则选择左点, 否则选择右点

```
1 # include <cstring>
2 # include <iostream>
3 # include <cstdio>
4 # include <cstdlib>
5 # define N 2500
6 # define M 2000000
7 using namespace std;
8 struct EDGE
9 {
10     int u,v,next;
11 }edge[M],edge2[M];
12 struct node
13 {
14     int s,e,len;
15 }a[N];
16 int first[N],DFN[N],divi[N],LOW[N],instack[N],stack[N];
17 int cf[N],choice[N],ind[N],q[N],ans[N],first2[N];
18 int cnt,cnt_edge,cnt_edge2,top,nowt;
19 inline void addege(int u,int v)
20 {
21     edge[cnt_edge].u=u;
22     edge[cnt_edge].v=v;
23     edge[cnt_edge].next=first[u];
24     first[u]=cnt_edge++;
25 }
26 inline void addege2(int u,int v)
27 {
28     edge2[cnt_edge2].v=v;
29     edge2[cnt_edge2].next=first2[u];
30     first2[u]=cnt_edge2++;
31 }
32 void tarjan(int u)
33 {
34     int v;
35     DFN[u]=LOW[u]=++nowt;
36     stack[++top]=u;
37     instack[u]=1;
38     for (int cur=first[u];cur!=-1;cur=edge[cur].next)
39     {
40         v=edge[cur].v;
41         if (!DFN[v])
42         {
43             tarjan(v);
44             LOW[u]=min(LOW[v],LOW[u]);
45         }
46         else
47             if (instack[v])
48                 LOW[u]=min(LOW[u],DFN[v]);
49     }
50     if (DFN[u]==LOW[u])
51     {
52         ++cnt;
```

```

53         do
54         {
55             v=stack[top--];
56             instack[v]=0;
57             divi[v]=cnt;
58         }
59         while (u!=v);
60     }
61 }
62 void initialize(void)
63 {
64     memset(first,-1,sizeof(first));
65     memset(first2,-1,sizeof(first2));
66     memset(instack,0,sizeof(instack));
67     memset(divi,0,sizeof(divi));
68     memset(DFN,0,sizeof(DFN));
69     memset(LOW,0,sizeof(LOW));
70     memset(choice,0,sizeof(choice));
71     memset(ind,0,sizeof(ind));
72     memset(ans,0,sizeof(ans));
73     nowt=cnt=top=0;
74     cnt_edge=cnt_edge2=0;
75 }
76 bool pd(int n)
77 {
78     for (int i=1;i<=2*n;i++)
79         if (!DFN[i])
80             tarjan(i);
81     for (int i=1;i<=n;i++)
82         if (divi[i]==divi[i+n])
83             return false;
84     return true;
85 }
86 void print_solution(int n)
87 {
88     for (int i=1;i<=n;i++)
89     {
90         cf[divi[i]]=divi[i+n];
91         cf[divi[i+n]]=divi[i];
92     }
93     for (int i=0;i<cnt_edge;i++)
94         if (divi[edge[i].u]!=divi[edge[i].v])// 反向连边
95         {
96             addedge2(divi[edge[i].v],divi[edge[i].u]);
97             ind[divi[edge[i].u]]++;
98         }
99     int head=0,tail=0,x;
100     for (int i=1;i<=cnt;i++)//topsort
101         if (!ind[i])
102             q[++tail]=i;
103     while (head++<tail)
104     {
105         x=q[head];
106         if (!choice[x])
107         {
108             choice[x]=1;
109             choice[cf[x]]=-1;
110         }
111         for (int cur=first2[x];cur!=-1;cur=edge2[cur].next)
112             if (--ind[edge2[cur].v]==0)
113                 q[++tail]=edge2[cur].v;
114     }
115     for (int i=1;i<=n;i++)

```

```

116         if (choice[divi[i]]==1)
117             ans[i]=1;
118     }
119     bool intersect(int x1,int y1,int x2,int y2)
120     {
121         if ((x2<y1 && y2>x1) || (x1<y2 && y1>x2))
122             return true;
123         return false;
124     }
125     int main (void)
126     {
127         int n,x,y;
128         scanf("%d",&n);
129         initialize();
130         for (int i=1;i<=n;i++)
131         {
132             scanf("%d:%d",&x,&y);
133             a[i].s=x*60+y;
134             scanf("%d:%d",&x,&y);
135             a[i].e=x*60+y;
136             scanf("%d",&a[i].len);
137             if (a[i].s+a[i].len>a[i].e)
138             {
139                 printf("NO\n");
140                 return 0;
141             }
142         }
143         for (int i=1;i<=n;i++)
144             for (int j=i+1;j<=n;j++)
145             {
146                 if (intersect(a[i].s,a[i].s+a[i].len,a[j].s,a[j].s+a[j].len))
147                 {
148                     addedge(i,j+n);
149                     addedge(j,i+n);
150                 }
151                 if (intersect(a[i].s,a[i].s+a[i].len,a[j].e-a[j].len,a[j].e))
152                 {
153                     addedge(i,j);
154                     addedge(j+n,i+n);
155                 }
156                 if (intersect(a[i].e-a[i].len,a[i].e,a[j].s,a[j].s+a[j].len))
157                 {
158                     addedge(i+n,j+n);
159                     addedge(j,i);
160                 }
161                 if (intersect(a[i].e-a[i].len,a[i].e,a[j].e-a[j].len,a[j].e))
162                 {
163                     addedge(i+n,j);
164                     addedge(j+n,i);
165                 }
166             }
167         if (!pd(n))
168             printf("NO\n");
169         else
170         {
171             printf("YES\n");
172             print_solution(n);
173             for (int i=1;i<=n;i++)
174             {
175                 if (ans[i])
176                 {
177                     x=a[i].s;
178                     y=a[i].s+a[i].len;

```

```

179         }
180     else
181     {
182         x=a[i].e-a[i].len;
183         y=a[i].e;
184     }
185     printf ("%02d:%02d %02d:%02d\n",x/60,x%60,y/60,y%60);
186 }
187 }
188 return 0;
189 }

```


4.5 A*求最短路

```
1  # include <cstring>
2  # include <queue>
3  # include <cstdlib>
4  # include <cstdio>
5  # include <iostream>
6  # define oo 200000000
7  # define N 1050
8  # define M 105000
9  using namespace std;
10 struct EDGE
11 {
12     int v,w,next;
13 }E[M],E1[M];
14 int first[N],first1[N],dist[N],vis[N],deg[N];
15 struct Po
16 {
17     int v,w;
18     bool operator <(const Po &a)const
19     {
20         return w+dist[v]>a.w+dist[a.v];
21     }
22 }Point;
23
24 inline void addedge(EDGE E[],int first[],int i,int u,int v,int c)
25 {
26     E[i].next=first[u];
27     E[i].v=v;
28     E[i].w=c;
29     first[u]=i;
30 }
31 void Dij(EDGE E[],int first[],int s,int n)
32 {
33     priority_queue <Po> Q;
34     Po tem;
35     int u,now;
36     memset(vis,0,sizeof(vis));
37     for (int i=1;i<=n;i++)
38         dist[i]=oo;
39     dist[s]=0;
40     tem.v=s;
41     tem.w=0;
42     Q.push(tem);
43     while (!Q.empty())
44     {
45         now=Q.top().v;
46         Q.pop();
47         if (vis[now])
48             continue;
49         vis[now]=1;
50         for (int cur=first[now];cur!=-1;cur=E[cur].next)
51         {
52             u=E[cur].v;
53             if (dist[u]>dist[now]+E[cur].w)
54             {
55                 dist[u]=dist[now]+E[cur].w;
56                 tem.v=u;
57                 tem.w=dist[u];
58                 Q.push(tem);
59             }
60         }
61     }
```

```

62 }
63 int A_star(EDGE E[],int first[],int s,int t,int k)
64 {
65     priority_queue <Po> Q;
66     Po tem,now;
67     memset(deg,0,sizeof(deg));
68     tem.v=s;
69     tem.w=0;
70     Q.push(tem);
71     while (!Q.empty())
72     {
73         tem=Q.top();
74         Q.pop();
75         deg[tem.v]++;
76         if (deg[tem.v]==k)
77             return tem.w+dist[tem.v];
78         for (int cur=first[tem.v];cur!=-1;cur=E[cur].next)
79         {
80             now.v=E[cur].v;
81             now.w=tem.w+E[cur].w;
82             if (deg[now.v]<k)
83                 Q.push(now);
84         }
85     }
86     return -1;
87 }
88 int main (void)
89 {
90     int n,m,s,t,k;
91     int x,y,z;
92     cin>>n>>m;
93     memset(first,-1,sizeof(first));
94     memset(first1,-1,sizeof(first1));
95     for (int i=0;i<m;i++)
96     {
97         scanf("%d%d%d",&x,&y,&z);
98         addedge(E,first,i,x,y,z);
99         addedge(E1,first1,i,y,x,z);
100     }
101     cin>>s>>t>>k;
102     if (s==t)
103         k++;
104     Dij(E1,first1,t,n);
105     cout<<A_star(E,first,s,t,k)<<endl;
106     return 0;
107 }

```

4.6 带花树

求一般图最大匹配数的带花树算法

```
1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  #define N 1000
5  #define M 800000
6  using namespace std;
7
8  int first[N],next[M],end[M];
9  int cnt_edge;
10 int n;
11
12 char mark[N],blossom[N];
13 int que[N],pre[N],base[N],match[N]; //match数组记录匹配点
14
15 void addEdge(int u,int v) // 无向图求最大匹配需要双向加边
16 {
17     end[cnt_edge]=v;
18     next[cnt_edge]=first[u];
19     first[u]=cnt_edge++;
20 }
21
22 void argument(int u)
23 {
24     int k,v;
25
26     while (~u)
27     {
28         v=pre[u];
29         k=match[v];
30         match[u]=v;
31         match[v]=u;
32         u=k;
33     }
34 }
35
36
37 void changeblossom(int b,int u)
38 {
39     int v;
40
41     while (base[u]!=b)
42     {
43         v=match[u];
44         blossom[base[v]]=blossom[base[u]]=1;
45         u=pre[v];
46
47         if (base[u]!=b)
48             pre[u]=v;
49     }
50 }
51
52
53 int findbase(int u,int v)
54 {
55     char inp[N];
56
57     memset(inp,0,sizeof(inp));
58
59     while (1)
60     {
```

```

61         inp[u]=1;
62         if (match[u]==-1)
63             break;
64
65         u=base[pre[match[u]]];
66     }
67
68     while (!inp[v])
69         v=base[pre[match[v]]];
70
71     return v;
72
73 }
74
75 int contract(int u,int v)
76 {
77     int b;
78
79     memset(blossom,0,sizeof(blossom));
80     b=findbase(base[u],base[v]);
81     changeblossom(b,u);
82     changeblossom(b,v);
83
84     if (base[u]!=b)
85         pre[u]=v;
86     if (base[v]!=b)
87         pre[v]=u;
88
89     return b;
90
91 }
92
93 int bfs(int p)
94 {
95     int i,j,head,tail,u,v,b;
96
97     memset(pre,-1,sizeof(pre));
98     memset(mark,0,sizeof(mark));
99
100     for (i=0;i<n;i++)
101         base[i]=i;
102
103     que[0]=p;
104     mark[p]=1;
105     head=0,tail=1;
106
107     while (head<tail)
108     {
109         u=que[head++];
110
111         for (i=first[u];i;i=next[i])
112         {
113             v=end[i];
114
115             if (base[u]!=base[v]&&v!=match[u])
116             {
117                 if (v==p|| (match[v]!=-1&&pre[match[v]]!=-1))
118                 {
119                     b=contract(u,v);
120
121                     for (j=0;j<n;j++)
122                         if (blossom[base[j]])
123                             {

```

```

124         base[j]=b;
125
126         if (mark[j]==0)
127         {
128             mark[j]=1;
129             que[tail++]=j;
130         }
131     }
132 }
133 else if (pre[v]==-1)
134 {
135     pre[v]=u;
136
137     if (match[v]==-1)
138     {
139         argument(v);
140         return 1;
141     }
142     else
143     {
144         que[tail++]=match[v];
145         mark[match[v]]=1;
146     }
147 }
148 }
149 }
150 }
151
152 return 0;
153 }
154
155
156 int max_match() // 返回最大匹配数
157 {
158     int i,res=0;
159
160     memset(match,-1,sizeof(match));
161
162     for (i=0;i<n;i++)
163         if (match[i]==-1)
164             res+=bfs(i);
165
166     return res;
167 }
168
169 //ural1099: n个点(编号范围0~n-1)的无向图, 求最大匹配数, 并输出最大匹配方案。单组数据, 读入边
    以EOF结束。
170 int main()
171 {
172     int i,u,v,ans;
173
174     memset(first,0,sizeof(first));
175     cnt_edge=1;
176
177     scanf("%d",&n);
178     while (scanf("%d%d",&u,&v)!=EOF)
179     {
180         u--,v--;
181         addEdge(u,v);
182         addEdge(v,u);
183     }
184
185     ans=max_match();

```

```

186
187     printf ("%d\n", ans*2);
188
189     for (i=0; i<n; i++)
190         if (match[i] != -1 && i < match[i])
191             printf ("%d %d\n", i+1, match[i]+1);
192
193     return 0;
194 }

```

4.7 KM算法

```
1  #include<cstdio>
2  #include<cmath>
3  #include<cstring>
4  #define N 105
5  #define oo 0x7fffffff
6  using namespace std;
7  struct Point
8  {
9      int x;
10     int y;
11 };
12
13 char str[N][N],mx[N],my[N];
14 int r,c,n,nx,ny,map[N][N],lx[N],ly[N],slack[N],res[N];
15
16 int find(int u)
17 {
18     int i,v,t;
19
20     mx[u]=1;
21
22     for (i=1;i<=n;i++)
23         if (my[i]==0)
24         {
25             v=i;
26             t=lx[u]+ly[v]-map[u][v];
27
28             if (t==0)
29             {
30                 my[v]=1;
31                 if (res[v]==0||find(res[v]))
32                 {
33                     res[v]=u;
34                     return 1;
35                 }
36             }
37             else
38             {
39                 if (slack[v]>t)
40                     slack[v]=t;
41             }
42         }
43
44     return 0;
45 }
46
47 void KM()
48 {
49
50
51     int i,j,d;
52
53     memset(res,0,sizeof(res));
54     memset(lx,0,sizeof(lx));
55     memset(ly,0,sizeof(ly));
56
57     for (i=1;i<=n;i++)
58         for (j=1;j<=n;j++)
59             if (lx[i]<map[i][j])
60                 lx[i]=map[i][j];
61 }
```

```

62     for (i=1;i<=n;i++)
63     {
64         for (j=1;j<=n;j++)
65             slack[j]=oo;
66
67         while (1)
68         {
69             memset(mx,0,sizeof(mx));
70             memset(my,0,sizeof(my));
71
72             if (find(i))
73                 break;
74             else
75             {
76                 d=oo;
77                 for (j=1;j<=n;j++)
78                     if (my[j]==0&&d>slack[j])
79                         d=slack[j];
80
81                 for (j=1;j<=n;j++)
82                     if (mx[j])
83                         lx[j]-=d;
84
85                 for (j=1;j<=n;j++)
86                     if (my[j])
87                         ly[j]+=d;
88                 else
89                     slack[j]-=d;
90             }
91         }
92     }
93 }
94
95 main()
96 {
97     int i,j,r,c,ans;
98     Point px[N],py[N];
99     while (scanf("%d%d",&r,&c),r||c)
100     {
101         for (i=0;i<r;i++)
102             scanf("%s",str[i]);
103
104         n=nx=ny=0;
105         for (i=0;i<r;i++)
106             for (j=0;j<c;j++)
107                 if (str[i][j]=='m')
108                 {
109                     px[++nx].x=i+1;
110                     px[nx].y=j+1;
111                 }
112                 else if (str[i][j]=='H')
113                 {
114                     py[++ny].x=i+1;
115                     py[ny].y=j+1;
116                 }
117         n=nx;
118
119         for (i=1;i<=n;i++)
120             for (j=1;j<=n;j++)
121                 map[i][j]=- (abs(px[i].x-py[j].x)+abs(px[i].y-py[j].y));
122
123         KM();
124

```



```

125         for (ans=0,i=1;i<=n;i++)
126             ans+=map[res[i]][i];
127
128         printf("%d\n",-ans);
129     }
130     return 0;
131 }

```

4.8 欧拉回路通路

当图中存在欧拉回路（通路）时，使用USACO算法能够找到一个可行方案

```
1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  #include<stack>
5  #define N 20000
6  #define M 110000
7  using namespace std;
8
9  int first[N],next[M],end[M],vis[M];
10 int cnt_edge;
11 stack<int>ans;
12
13 void addEdge(int u,int v)
14 {
15     end[cnt_edge]=v;
16     next[cnt_edge]=first[u];
17     first[u]=cnt_edge++;
18 }
19
20 // 无向图需要双向加边，欧拉通路需要选好起点，最后逆序（弹栈）输出
21 void USACO(int u)
22 {
23     int i,v;
24
25     for (i=first[u];i;i=next[i])
26     {
27         v=end[i];
28
29         if (vis[i]==0)
30         {
31             vis[i]=1;
32             USACO(v);
33         }
34     }
35
36     ans.push(u);
37 }
38
39 //poj2230
40 int main()
41 {
42     int n,m,u,v,i,j;
43
44     memset(first,0,sizeof(first));
45     memset(vis,0,sizeof(vis));
46     cnt_edge=1;
47
48     scanf("%d%d",&n,&m);
49     for (i=0;i<m;i++)
50     {
51         scanf("%d%d",&u,&v);
52         addEdge(u,v);
53         addEdge(v,u);
54     }
55
56     USACO(1);
57
58     while (!ans.empty())
59     {
60         printf("%d\n",ans.top());
```

```
61 |         ans.pop();
62 |     }
63 |
64 |     return 0;
65 | }
```

4.9 汉密尔顿回路

哈密顿回路 Dirac's Theorem: 对于一张顶点个数为 n 的无向图, 若 $n \geq 3$ 且每个顶点的度不小于 $(n+1)/2$, 那么一定存在哈密顿回路。 以下为满足此性质的图中求哈密顿回路的算法, 时间复杂度为 $O(n^2)$ 。

```
1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  #define N 200
5  using namespace std;
6
7  struct node
8  {
9      int id;
10     node *next;
11
12     node (int u=0,node *x=NULL)
13     {
14         id=u;
15         next=x;
16     }
17 };
18
19 node *hs,*he,*cur; //hs为链的头指针, he为尾指针
20 char map[N][N],mark[N]; //map[u][v]为1则顶点u、v关联
21 int n,m,size;
22
23 void init()
24 {
25     memset(map,0,sizeof(map));
26     memset(mark,0,sizeof(mark));
27     hs=he=cur=NULL;
28 }
29
30 void reverse(node *v)
31 {
32     if (v==he)
33     {
34         cur=he;
35         return;
36     }
37
38     reverse(v->next);
39     cur->next=v;
40     cur=v;
41 }
42
43 void change(node *v1,node *v2)
44 {
45     v1->next=he;
46     reverse(v2);
47 }
48
49 void solve()
50 {
51     int i;
52
53     hs=new node(1,NULL);
54     mark[1]=1;
55
56     for (i=2;i<=n;i++)
57         if (map[i][1])
58         {
59             he=new node(i,NULL);
```

```

60
61         hs->next=he;
62         mark[i]=1;
63         break;
64     }
65
66     size=2;
67     while (1)
68     {
69         for (i=1;i<=n;i++)
70         {
71             if (mark[i])
72                 continue;
73
74             if (map[i][hs->id]&&mark[i]==0)
75             {
76                 mark[i]=1;
77                 size++;
78                 hs=new node(i,hs);
79             }
80
81             if (map[i][he->id]&&mark[i]==0)
82             {
83                 mark[i]=1;
84                 size++;
85                 node *tmp=he;
86                 he=new node(i,NULL);
87                 tmp->next=he;
88             }
89         }
90
91         if (map[hs->id][he->id]==0)
92         {
93             node *tmp=hs->next;
94
95             while (tmp->next!=he&&tmp!=he)
96             {
97                 if (map[hs->id][tmp->next->id]==1&&map[he->id][tmp->id]==1)
98                 {
99                     change(tmp,tmp->next);
100                     he=cur;
101                     he->next=NULL;
102                     cur=NULL;
103                     break;
104                 }
105                 tmp=tmp->next;
106             }
107         }
108
109         if (size==n)
110             break;
111
112         for (i=1;i<=n;i++)
113         {
114             if (mark[i])
115                 continue;
116
117             node *tmp=hs;
118
119             while (tmp!=NULL)
120             {
121                 if (map[i][tmp->id])
122                     {

```

```

123         he->next=hs;
124         hs=tmp->next;
125         he=new node(i,NULL);
126         tmp->next=he;
127         mark[i]=1;
128         size++;
129         break;
130     }
131     tmp=tmp->next;
132 }
133
134     if (map[hs->id][he->id]==0)
135         break;
136 }
137 }
138 }
139
140 void print()
141 {
142     node *tmp=hs;
143
144     printf("%d",tmp->id);
145     tmp=tmp->next;
146
147     while (tmp!=NULL)
148     {
149         printf(" %d",tmp->id);
150         tmp=tmp->next;
151     }
152
153     printf("\n");
154 }
155
156 //hdu4337: 求满足Dirac's Theorem中条件的图 的哈密顿回路的方案
157 int main()
158 {
159     int i,u,v;
160
161     while (scanf("%d%d",&n,&m)!=EOF)
162     {
163         init();
164
165         for (i=0;i<m;i++)
166         {
167             scanf("%d%d",&u,&v);
168             map[u][v]=map[v][u]=1;
169         }
170
171         solve();
172         print();
173     }
174
175     return 0;
176 }

```

4.10 最小树形图

4.10.1 邻接表

```
1  #include<iostream>
2  #include<cstdio>
3  #include<cstring>
4  #include<cmath>
5  #define oo 1000000000
6  #define N 1005
7  #define M 2000100
8  using namespace std;
9
10 struct Point
11 {
12     int x,y,z;
13 };
14
15 struct Edge
16 {
17     int u,v,w;
18 };
19
20 Edge edge[M];
21 Point pt[N];
22 int pre[N],in[N],color[N],mark[N];
23 int cnt_edge;
24
25 void addEdge(int u,int v,int w)
26 {
27     edge[cnt_edge].u=u;
28     edge[cnt_edge].v=v;
29     edge[cnt_edge].w=w;
30     cnt_edge++;
31 }
32
33
34 int dis(Point a,Point b)
35 {
36     return abs(a.x-b.x)+abs(a.y-b.y)+abs(a.z-b.z);
37 }
38
39 int ZhuLiu(int root,int n,int m)
40 {
41     int i,j,u,v,w,cnt;
42     int res=0;
43
44     while (1)
45     {
46         for (i=1;i<=n;i++)
47             in[i]=oo;
48         for (i=0;i<m;i++)
49         {
50             u=edge[i].u;
51             v=edge[i].v;
52             w=edge[i].w;
53
54             if (in[v]>w&&u!=v)
55             {
56                 in[v]=w;
57                 pre[v]=u;
58             }
59         }
60     }
```

```

61     for (i=1;i<=n;i++)
62         if (i!=root&&in[i]==oo)
63             return -1;
64
65     memset(mark,0,sizeof(mark));
66     memset(color,0,sizeof(color));
67     mark[root]=1;
68     in[root]=0;
69     cnt=0;
70
71     for (i=1;i<=n;i++)
72     {
73         res+=in[i];
74         v=i;
75         while (mark[v]!=i&&color[v]==0&&v!=root)
76         {
77             mark[v]=i;
78             v=pre[v];
79         }
80
81         if (v!=root&&color[v]==0)
82         {
83             cnt++;
84             for (u=pre[v];u!=v;u=pre[u])
85                 color[u]=cnt;
86             color[v]=cnt;
87         }
88     }
89
90     if (cnt==0)
91         break;
92     for (i=1;i<=n;i++)
93         if (color[i]==0)
94             color[i]=++cnt;
95     for (i=0;i<m;i++)
96     {
97         v=edge[i].v;
98         edge[i].u=color[edge[i].u];
99         edge[i].v=color[edge[i].v];
100         if (edge[i].u!=edge[i].v)
101             edge[i].w-=in[v];
102     }
103
104     n=cnt;
105     root=color[root];
106 }
107
108 return res;
109 }
110
111 int main()
112 {
113     int n,m,i,u,v,w,mx,my,mz;
114
115     while (scanf("%d%d%d%d",&n,&mx,&my,&mz),n||mx||my||mz)
116     {
117         cnt_edge=0;
118         for (i=1;i<=n;i++)
119             scanf("%d%d%d",&pt[i].x,&pt[i].y,&pt[i].z);
120
121         for (i=1;i<=n;i++)
122         {
123             u=i;

```



```

124         scanf("%d",&m);
125         while (m--)
126         {
127             scanf("%d",&v);
128             if (u==v)
129                 continue;
130             w=dis(pt[u],pt[v])*my;
131             if (pt[u].z<pt[v].z)
132                 w+=mz;
133
134             addEdge(u,v,w);
135         }
136         addEdge(n+1,i,pt[i].z*mx);
137     }
138
139     printf("%d\n",ZhuLiu(n+1,n+1,cnt_edge));
140 }
141
142
143     return 0;
144 }

```

4.10.2 邻接矩阵

```

1  #include<iostream>
2  #include<cstdio>
3  #include<cstring>
4  #include<cmath>
5  #include<algorithm>
6  #define oo 1e15
7  #define eps 1e-6
8  #define N 105
9  #define M 10010
10 using namespace std;
11
12 struct Point
13 {
14     double x,y;
15 };
16 Point pt[N];
17
18 double dis(Point a,Point b)
19 {
20     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
21 }
22
23 double map[N][N];
24 int pre[N];
25 char mark[N],flag[N];
26 int n;
27 void dfs(int u)
28 {
29     int i;
30
31     mark[u]=1;
32     for (i=1;i<=n;i++)
33         if (mark[i]==0&&map[u][i]<oo)
34             dfs(i);
35 }
36
37 int connect(int root)
38 {
39     int i;
40

```

```

41     memset(mark,0,sizeof(mark));
42     dfs(root);
43
44     for (i=1;i<=n;i++)
45         if (mark[i]==0)
46             return 0;
47     return 1;
48 }
49
50 double ZhuLiu(int root)
51 {
52     int i,j,k;
53     double res=0;
54
55     memset(flag,0,sizeof(flag));
56
57     while (1)
58     {
59         for (i=1;i<=n;i++)
60         {
61             if (flag[i]||i==root)
62                 continue;
63
64             pre[i]=i;
65             map[i][i]=oo;
66             for (j=1;j<=n;j++)
67                 if (!flag[j]&&map[j][i]<map[pre[i]][i])
68                     pre[i]=j;
69         }
70
71         for (i=1;i<=n;i++)
72         {
73             if (flag[i]||i==root)
74                 continue;
75
76             memset(mark,0,sizeof(mark));
77             mark[root]=1;
78             j=i;
79
80             do
81             {
82                 mark[j]=1;
83                 j=pre[j];
84             }while (!mark[j]);
85
86             if (j==root)
87                 continue;
88
89             i=j;
90             res+=map[pre[i]][i];
91             for (j=pre[i];j!=i;j=pre[j])
92             {
93                 flag[j]=1;
94                 res+=map[pre[j]][j];
95             }
96
97             for (j=1;j<=n;j++)
98                 if (!flag[j]&&map[j][i]<oo)
99                     map[j][i]-=map[pre[i]][i];
100
101             for (j=pre[i];j!=i;j=pre[j])
102                 for (k=1;k<=n;k++)
103                     if (!flag[k])

```

```

104         {
105             map[i][k]=min(map[i][k],map[j][k]);
106             if (map[k][j]<oo)
107                 map[k][i]=min(map[k][i],map[k][j]-map[pre[j]][j]
108                     );
109         }
110         break;
111     }
112     if (i>n)
113     {
114         for (j=1;j<=n;j++)
115             if (!flag[j]&&j!=root)
116                 res+=map[pre[j]][j];
117         break;
118     }
119 }
120
121 return res;
122 }
123
124 int main()
125 {
126     int m,i,j,u,v;
127
128     while (scanf("%d%d",&n,&m)!=EOF)
129     {
130         for (i=0;i<=n;i++)
131             for (j=0;j<=n;j++)
132                 map[i][j]=oo;
133
134         for (i=1;i<=n;i++)
135             scanf("%lf%lf",&pt[i].x,&pt[i].y);
136         for (i=0;i<m;i++)
137         {
138             scanf("%d%d",&u,&v);
139             map[u][v]=min(map[u][v],dis(pt[u],pt[v]));
140         }
141
142         if (connect(1))
143             printf("%.2f\n",ZhuLiu(1));
144         else
145             printf("poor snoopy\n");
146     }
147
148     return 0;
149 }

```

4.11 稳定婚姻

解题思路：此题为一道典型的稳定婚姻问题，如果男A与女C匹配，男B与女D 匹配，但是A更喜欢D，D更喜欢A，那么这个婚姻就不是稳定的。稳定婚姻问题的算法步骤大致如下：（男士优先）

1.男士先选择自己最爱的人去求婚，如果有多个男士的最爱相同，那么女士就选择更爱的那位男士，那么其他的男士在这次求婚中失败。

2.上次求婚失败的男士再选择自己次爱的女士进行求婚，如果这位女士没有匹配男士，那么这两个人就进行匹配，如果这位女士有匹配的男士，但是如果这位女士更喜欢这位正在求婚的男士，那么这位女士就可以抛弃原有的男士而与这位她更爱的男士匹配。而原有的男士在这次匹配中失败。

3.上次求婚失败的男士(包括被女士抛弃的男士)再次选择自己次爱的女士求婚，知道所有的男士与女士全部匹配则结束。此算法被证明必定存在解，刚才所写的步骤是基于男士优先的。如果是女士向男士求婚，那么就是基于女士优先的，算法步骤和上面的基本相同，只要将男女调换就可以了。

```
1  #include<iostream>
2  #include<cstring>
3  #include<cstdio>
4  #include<map>
5  #include<string>
6  using namespace std;
7  const int N = 1010;
8  map<string,int> gmp,bmp;
9  string s;
10 int b[N][N],g[N][N],match[N],vis[N],v[N][N];
11 string boy[N],girl[N];
12 int n;
13 int num;
14 void find(int x)
15 {
16     int t=0;
17     for(int i=1;i<=n;i++){
18         if(!v[i][x]&&(t==0||g[x][t]>g[x][i])){
19             t=i;
20         }
21     }
22     v[t][x]=1;
23     if(!match[t]){
24         match[t]=x;vis[x]=1;num++;
25     }else{
26         if(b[t][x]<b[t][match[t]]){
27             vis[match[t]]=0;
28             match[t]=x;vis[x]=1;
29         }
30     }
31 }
32 void solve()
33 {
34     num=0;
35     memset(vis,0,sizeof(vis));
36     memset(v,0,sizeof(v));
37     memset(match,0,sizeof(match));
38     while(num<n){
39         for(int i=1;i<=n;i++){
40             if(!vis[i]) find(i);
41         }
42     }
43 }
44 int main(void)
45 {
46     while(scanf("%d",&n)!=EOF)
47     {
48         bmp.clear(),gmp.clear();
49         memset(g,0,sizeof(g));
50         memset(b,0,sizeof(b));
51         int k=0;
```

```

52     for(int i=1;i<=n;i++){
53         cin>>s;
54         boy[i]=s;
55         bmp[s]=i;
56         for(int j=1;j<=n;j++){
57             cin>>s;
58             if(gmp[s]==0){
59                 gmp[s]=++k;
60                 girl[k]=s;
61             }
62             b[i][gmp[s]]=j;
63         }
64     }
65     for(int i=1;i<=n;i++){
66         cin>>s;
67         int t=gmp[s];
68         for(int j=1;j<=n;j++){
69             cin>>s;
70             g[t][bmp[s]]=j;
71         }
72     }
73     solve();
74     for(int i=1;i<=n;i++){
75         cout<<boy[i]<<" "<<girl[match[i]]<<endl;
76     }
77 }
78 return 0;
79 }

```

4.12 最优比率生成树

poj_2728

dis[i][j]是i到j的距离

cost[i][j]修建i,j这条边的花费。

要求总花费比总距离最小（单位距离花费最小）

```
1 # include <cstdio>
2 # include <iostream>
3 # include <cstring>
4 # include <cmath>
5 # define N 1050
6 # define bignum 200000000
7 using namespace std;
8 double x[N],y[N],z[N],val[N],dis[N][N],cost[N][N],map[N][N];
9 int vis[N];
10 inline double prim(double x,int n)
11 {
12     for (int i=1;i<n;i++)
13         for (int j=i+1;j<=n;j++)
14             map[i][j]=map[j][i]=cost[i][j]-x*dis[i][j];
15     memset(vis,0,sizeof(int)*(n+10));
16     for (int i=2;i<=n;i++)
17         val[i]=map[1][i];
18     vis[1]=1;
19     double ans=0,Min;
20     int now;
21     for (int p=1;p<n;p++)
22     {
23         Min = bignum;
24         for (int i=2;i<=n;i++)
25             if (!vis[i] && val[i]<Min)
26             {
27                 now=i;
28                 Min=val[i];
29             }
30         vis[now]=1;
31         ans+=Min;
32         for (int i=2;i<=n;i++)
33             if (!vis[i])
34                 val[i]=min(val[i],map[now][i]);
35     }
36     return ans;
37 }
38 int main (void)
39 {
40     int n;
41     while (scanf("%d",&n)!=EOF && n)
42     {
43         double Min=bignum;
44         for (int i=1;i<=n;i++)
45         {
46             scanf("%lf%lf%lf",&x[i],&y[i],&z[i]);
47             for (int j=1;j<i;j++)
48             {
49                 cost[i][j]=cost[j][i]=fabs(z[i]-z[j]);
50                 dis[i][j]=dis[j][i]=sqrt((x[i]-x[j])*(x[i]-x[j])+(y[i]-y[j])
51                     *(y[i]-y[j]));
52                 if ( fabs(dis[i][j])>1e-2 && cost[i][j]/dis[i][j]<Min)
53                     Min=cost[i][j]/dis[i][j];
54             }
55         }
56         double left=Min,right,mid,sum1=0,sum2=0;
57         for (int i=2;i<=n;i++)
```

```

57     {
58         sum1+=cost[i][1];
59         sum2+=dis[i][1];
60     }
61     right=sum1/sum2;
62     while (right-left>1e-4)
63     {
64         mid=(right+left)/2;
65         if (prim(mid,n)>0)
66             left=mid;
67         else right=mid;
68     }
69     printf("%.3f\n",left);
70 }
71 return 0;
72 }

```

5 DP

5.1 插头DP

```
1  # include <cstring>
2  # include <cstdlib>
3  # include <cstdio>
4  # include <iostream>
5  # define HashMod 1999997
6  # define LEN 700000
7  # define N 15
8  using namespace std;
9  int HHash[HashMod];
10 int state[2][LEN],tot[2];
11 long long sum[2][LEN];
12 int a[N][N];
13 int n,m,nn,mm;
14 inline void Hash_in(int k,int s,long long data)
15 {
16     int hashpos=s%HashMod;
17     while (HHash[hashpos])
18     {
19         if (state[k][HHash[hashpos]]==s)
20         {
21             sum[k][HHash[hashpos]]+=data;
22             return;
23         }
24         hashpos++;
25         if (hashpos==HashMod) hashpos=0;
26     }
27     HHash[hashpos]=++tot[k];
28     state[k][tot[k]]=s;
29     sum[k][tot[k]]=data;
30 }
31 long long solve (void)
32 {
33     int k=0,jz[N];
34     int s,temps,p,q,bracket,w;
35     long long data,ans=0;
36     for (int i=0;i<N;i++)
37         jz[i]=i<1;
38     tot[0]=1;
39     state[0][1]=0;
40     sum[0][1]=1;
41     for (int i=1;i<=n;i++)
42     {
43         for (int j=1;j<=m;j++)
44         {
45             k^=1;
46             tot[k]=0;
47             memset(HHash,0,sizeof(HHash));
48             memset(state[k],0,sizeof(state[k]));
49             memset(sum[k],0,sizeof(sum[k]));
50             for (int now=1;now<=tot[k^1];now++)
51             {
52                 s=state[k^1][now];
53                 data=sum[k^1][now];
54                 p=(s>>jz[j-1])%4;
55                 q=(s>>jz[j])%4;
56                 if (!a[i][j])
57                 {
58                     if (p==0 && q==0)
59                         Hash_in(k,s,data);
```



```

60         continue;
61     }
62     if (p==2 && q==1)
63     {
64         temps=s-2*(1<<jz[j-1])-(1<<jz[j]);
65         Hash_in(k,temps,data);
66         continue;
67     }
68     if (p==1 && q==2)
69     {
70         if (i==nn && j==mm)
71             ans+=data;
72         continue;
73     }
74     if (p==1 && q==1)
75     {
76         bracket=1;
77         temps=s-(1<<jz[j-1])-(1<<jz[j]);
78         for (int x=j+1;x<=m;x++)
79         {
80             w=(s>>jz[x])%4;
81             if (w==1)
82                 bracket++;
83             if (w==2)
84                 bracket--;
85             if (bracket==0)
86             {
87                 temps=temps-(1<<jz[x]);
88                 break;
89             }
90         }
91         Hash_in(k,temps,data);
92         continue;
93     }
94     if (p==2 && q==2)
95     {
96         bracket=1;
97         temps=s-2*(1<<jz[j-1])-2*(1<<jz[j]);
98         for (int x=j-2;x>=0;x--)
99         {
100             w=(s>>jz[x])%4;
101             if (w==1)
102                 bracket--;
103             if (w==2)
104                 bracket++;
105             if (bracket==0)
106             {
107                 temps=temps+(1<<jz[x]);
108                 break;
109             }
110         }
111         Hash_in(k,temps,data);
112         continue;
113     }
114     if (p==0 && q==0)
115     {
116         if (a[i][j+1] && a[i+1][j])
117         {
118             temps=s+(1<<jz[j-1])+2*(1<<jz[j]);
119             Hash_in(k,temps,data);
120         }
121         continue;
122     }

```

```

123         if (p==0 && q>0)
124         {
125             if (a[i][j+1])
126                 Hash_in(k,s,data);
127             if (a[i+1][j])
128             {
129                 temps=s-q*(1<<jz[j])+q*(1<<jz[j-1]);
130                 Hash_in(k,temps,data);
131             }
132         }
133         if (p>0 && q==0)
134         {
135             if (a[i+1][j])
136                 Hash_in(k,s,data);
137             if (a[i][j+1])
138             {
139                 temps=s-p*(1<<jz[j-1])+p*(1<<jz[j]);
140                 Hash_in(k,temps,data);
141             }
142             continue;
143         }
144     }
145 }
146 for (int now=1;now<=tot[k];now++)
147     state[k][now]<=2;
148 }
149 return ans;
150 }
151 int main (void)
152 {
153     char ss[50];
154     memset(a,0,sizeof(a));
155     cin>>n>>m;
156     for (int i=1;i<=n;i++)
157     {
158         scanf("%s",ss+1);
159         for (int j=1;j<=m;j++)
160         {
161             a[i][j]=ss[j]=='.';
162             if (a[i][j])
163             {
164                 nn=i;
165                 mm=j;
166             }
167         }
168     }
169     cout<<solve()<<endl;
170     return 0;
171 }

```

5.2 数位DP

CF55D

```
1 #include<iostream>
2 #include<cstring>
3 #include<cstdio>
4 using namespace std;
5 #ifdef WINDOWS
6 #define LOL __int64
7 #else
8 #define LOL long long
9 #endif
10 LOL dp[50][2600][50];
11 int c[2600],g[2600][30];
12 int v[50];
13 int b[30],a[30];
14 int gcd(int a,int b)
15 {
16     if(b==0) return a;
17     return gcd(b,a%b);
18 }
19 int lcm(int a,int b)
20 {
21     if(a>b) swap(a,b);
22     if(a==0) return b;
23     return a*b/gcd(a,b);
24 }
25 LOL dfs(int l,int pre,int lcm,bool z)
26 {
27     if(l==0) return pre%lcm==0;
28     if(!z&&dp[l][pre][c[lcm]]!=-1) return dp[l][pre][c[lcm]];
29     LOL ans=0;
30     int u=z?a[l]:9;
31     for(int i=0;i<=u;i++)
32     {
33         int npre=(pre*10+i)%2520,nlcm=g[lcm][i];
34         ans+=dfs(l-1,npre,nlcm,i==u&&z);
35     }
36     if(!z)
37     {
38         dp[l][pre][c[lcm]]=ans;
39     }
40     return ans;
41 }
42 LOL solve(LOL x)
43 {
44     int n=0;
45     while(x!=0)
46     {
47         a[++n]=x%10;
48         x/=10;
49     }
50     return dfs(n,0,1,1);
51 }
52 int main(void)
53 {
54     memset(dp,0,sizeof(dp));
55     b[0]=1;
56     for(int i=1;i<=20;i++)
57     {
58         b[i]=(b[i-1]*10)%2520;
59     }
60     for(int i=1,r=-1;i<=2520;i++)
```

```

61     {
62         if (2520%i==0)
63         {
64             r++;
65             v[r]=i;
66         }
67         c[i]=r;
68     }
69     for(int j=0;j<10;++j){
70         for(int i=1;i<=2520;++i)
71             g[i][j]=j?i*j/gcd(i,j):i;
72     }
73     memset(dp,-1,sizeof(dp));
74     int T;
75     LOL a,b;
76     cin>>T;
77     while(T--)
78     {
79         cin>>a>>b;
80         cout<<solve(b)-solve(a-1)<<endl;
81     }
82     return 0;
83 }

```

5.3 最大平均子段和

```
1 # include <stdio>
2 # include <cstring>
3 # include <cmath>
4 # define N 1060000
5 using namespace std ;
6 __int64 sum [ N ] ,q[ N ];
7 int GetInt()
8 {
9     char ch=getchar();
10    while(ch<'0' || ch>'9')
11        ch=getchar();
12    int num=0;
13    while(ch>='0' && ch<='9')
14    {
15        num=num*10+ch-'0';
16        ch=getchar();
17    }
18    return num;
19 }
20 double MAX (double x , double y)
21 {
22     return x>y?x:y;
23 }
24 int main ( void )
25 {
26     __int64 x1,y1,x2,y2;
27     int head , tail ;
28     int n , k ,x;
29     while (scanf ("%d%d",&n,&k) !=EOF)
30     {
31         for (int i=1;i<=n;i++)
32         {
33             x=GetInt();
34             sum[i]=sum[i-1]+x;
35         }
36         tail = 0 ;
37         head = 1 ;
38         double ans = 0 ;
39         for (int i=k;i<=n;i++)
40         {
41             int now = i-k;
42             while (head<tail)
43             {
44                 x1=now-q[tail];
45                 y1=sum[now]-sum[q[tail]];
46                 x2=q[tail]-q[tail-1];
47                 y2=sum[q[tail]]-sum[q[tail-1]];
48                 if (x1*y2-x2*y1>=0)
49                     tail--;
50                 else break;
51             }
52             q[++tail]=now;
53             while (head<tail)
54             {
55                 x1=i-q[head+1];
56                 y1=sum[i]-sum[q[head+1]];
57                 x2=q[head+1]-q[head];
58                 y2=sum[q[head+1]]-sum[q[head]];
59                 if (x1*y2-x2*y1<=0)
60                     head++;
61                 else break ;
```

```

62         }
63         ans = MAX (ans,double(sum[i]-sum[q[head]])/double (i-q[head]));
64     }
65     printf("%.2lf\n",ans);
66 }
67 return 0 ;
68 }

```

5.4 斯坦纳树

```
1 # include <cstring>
2 # include <cstdio>
3 # include <cstdlib>
4 # include <iostream>
5 using namespace std;
6 # define oo 200000000
7 # define N 55
8 int dp[2000][N],val[2000];
9 int f[N][N],num[N],l[N],n,k;
10 inline int lowbit(int x)
11 {
12     return x&(-x);
13 }
14 int judge(int x)
15 {
16     int cnt1=0,cnt2=0;
17     for (int i=0;i<n;i++)
18     {
19         if (num[i]==-1) continue;
20         if (x&(1<<num[i]))
21         {
22             if (num[i]<k) cnt1++;
23             else cnt2++;
24         }
25     }
26     return (cnt1==cnt2);
27 }
28 int main (void)
29 {
30     int t,m,u,v,c;
31     scanf("%d",&t);
32     while (t--)
33     {
34         scanf("%d%d%d",&n,&m,&k);
35         for (int i=0;i<n;i++)
36             for (int j=0;j<n;j++)
37                 if (i!=j)
38                     f[i][j]=oo;
39         for (int i=1;i<=m;i++)
40         {
41             scanf("%d%d%d",&u,&v,&c);
42             u--,v--;
43             f[u][v]=min(f[u][v],c);
44             f[v][u]=min(f[v][u],c);
45         }
46         for (int p=0;p<n;p++)
47             for (int i=0;i<n;i++)
48                 for (int j=0;j<n;j++)
49                     f[i][j]=min(f[i][j],f[i][p]+f[p][j]);
50         memset(num,-1,sizeof(num));
51         memset(dp,-1,sizeof(dp));
52         int tot=0;
53         for (int i=0;i<k;i++)
54             num[i]=tot++;
55         for (int i=n-k;i<n;i++)
56             num[i]=tot++;
57         for (int i=0;i<n;i++)
58         {
59             int s=0;
60             if (num[i]!=-1)
```

```

62         s=(1<<num[i]);
63         dp[s][i]=0;
64     }
65     int all=(1<<(k<<1));
66     for (int i=1;i<all;i++)
67         val[i]=oo;
68     for (int i=1;i<all;i++)
69     {
70         for (int j=0;j<n;j++)
71         {
72             for (int s=(i-1)&i;s;s=(s-1)&i)
73             {
74                 if (dp[s][j]==-1 || dp[i^s][j]==-1) continue;
75                 if (dp[i][j]==-1 || dp[i][j]>dp[s][j]+dp[i^s][j])
76                     dp[i][j]=dp[s][j]+dp[i^s][j];
77             }
78             if (dp[i][j]==-1) continue;
79             val[i]=min(val[i],dp[i][j]);
80             for (int q=0;q<n;q++)
81                 if (f[j][q]<oo)
82                     if (dp[i][q]==-1 || dp[i][q]>dp[i][j]+f[j][q])
83                         dp[i][q]=dp[i][j]+f[j][q];
84         }
85     }
86     for (int i=0;i<all;i++)
87         if (judge(i))
88             for (int j=0;j<i;j++)
89                 if (judge(j) && ((i&j)==j))
90                     val[i]=min(val[i],val[j]+val[i^j]);
91     if (val[all-1]<oo)
92         printf("%d\n",val[all-1]);
93     else printf("No solution\n");
94 }
95 return 0;
96 }

```


5.5 四边形不等式

状态转移方程如 $m(i, j) = m(i, k-1) + m(k, j) + w(i, j)$, 假设 $a \leq b < c \leq d$, 如果满足 $w(a, c) + w(b, d) \leq w(b, c) + w(a, d)$, 那么称 w 满足四边形不等式。若 w 满足四边形不等式, 则 m 也满足四边形不等式, 其决策 $s(i, j)$ 满足: $s(i, j-1) \leq s(i, j) \leq s(i+1, j)$

hdu3516 Consider a two-dimensional space with a set of points (x_i, y_i) that satisfy $x_i < x_j$ and $y_i > y_j$ for all $i < j$. We want to have them all connected by a directed tree whose edges go toward either right (x positive) or upward (y positive). Find a tree connecting all given points with the shortest total length of edges.

```
1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4 #define N 1010
5 #define oo 0x3fffffff
6 using namespace std;
7
8 int dp[N][N], s[N][N], x[N], y[N];
9
10 int main()
11 {
12     int n, i, j, k;
13
14     while (scanf("%d", &n) != EOF)
15     {
16         for (i=0; i<n; i++)
17             scanf("%d%d", &x[i], &y[i]);
18
19         if (n==1)
20         {
21             printf("0\n");
22             continue;
23         }
24
25         for (j=1; j<n; j++)
26         {
27             dp[j-1][j]=x[j]-x[j-1]+y[j-1]-y[j];
28             s[j-1][j]=j-1;
29
30             for (i=j-2; i>=0; i--)
31             {
32                 dp[i][j]=oo;
33
34                 for (k=s[i][j-1]; k<=s[i+1][j]; k++)
35                     if (dp[i][j]>dp[i][k]+dp[k+1][j]+x[k+1]-x[i]+y[k]-y[j])
36                     {
37                         dp[i][j]=dp[i][k]+dp[k+1][j]+x[k+1]-x[i]+y[k]-y[j];
38                         s[i][j]=k;
39                     }
40             }
41         }
42
43         printf("%d\n", dp[0][n-1]);
44     }
45
46     return 0;
47 }
```

```
1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4 #define N 301
5 #define M 32
6 #define oo 0x3fffffff
7 using namespace std;
```

```

8
9 int dp[M][N],s[M][N],w[N][N],p[N];
10
11 int main()
12 {
13     int n,m,i,j,k;
14
15     scanf("%d%d",&n,&m);
16
17     for (i=1;i<=n;i++)
18         scanf("%d",&p[i]);
19
20     for (i=1;i<=n;i++)
21     {
22         w[i][i]=0;
23         for (j=i+1;j<=n;j++)
24             w[i][j]=w[i][j-1]+p[j]-p[(i+j)/2];
25     }
26
27     memset(dp,0,sizeof(dp));
28
29     for (i=0;i<=n;i++)
30     {
31         dp[min(i,m+1)][i]=0;
32         s[min(i,m+1)][i]=i;
33     }
34
35     for (i=1;i<=n;i++)
36     {
37         dp[0][i]=oo;
38         for (j=min(i-1,m);j>0;j--)
39         {
40             dp[j][i]=oo;
41             for (k=s[j][i-1];k<=s[j+1][i];k++)
42                 if (dp[j][i]>dp[j-1][k-1]+w[k][i])
43                 {
44                     dp[j][i]=dp[j-1][k-1]+w[k][i];
45                     s[j][i]=k;
46                 }
47         }
48     }
49
50     printf("%d\n",dp[m][n]);
51
52     return 0;
53 }

```

6 数学

6.1 数学结论

当 $b \geq \phi(c)$ 时:

$$a^b = a^{b \% \phi(c) + \phi(c)} \pmod{c}$$

6.2 数论基础

```
1 #include <cstdio>
2 #include <cmath>
3 #include <cstring>
4 #include <iostream>
5 #define M 9901
6 using namespace std;
7 typedef __int64 typec;
8 ///teoy's number theory template
9 ///functions
10 /*****
11 gcd
12 *****/
13 typec gcd(typec a, typec b)
14 {
15     if(b==0) return a;
16     return gcd(b,a%b);
17 }
18 /*****
19 Extend_GCD
20 *****/
21 typec extendGCD(typec a, typec b, typec& x, typec& y)
22 {
23     if(!b) return x = 1, y = 0, a;
24     typec res = extendGCD(b, a % b, x, y), tmp = x;
25     x = y, y = tmp - (a / b) * y;
26     return res;
27 }
28 /*****求对的逆元
29 ap
30 *****/
31 typec inverse(typec a, typec p)
32 {
33     typec x, y;
34     y = extendGCD(a, p, x, y);
35     return x < 0 ? x += p : x;
36 }
37 /*****
38 abss for abs();
39 kgcd for quick gcd;
40 *****/
41 typec abss(typec a)
42 {
43     if(a<0) return -a;
44     return a;
45 }
46 typec kgcd(typec a,typec b)
47 {
48     if(a==0) return b;
49     if(b==0) return a;
50     if(!(a&1)&&!(b&1)) return kgcd(a>>1,b>>1)<<1;
51     else if(!(b&1)) return kgcd(a,b>>1);
52     else if(!(a&1)) return kgcd(a>>1,b);
53     else return kgcd(abss(a-b),min(a,b));
54 }
55
56
57 /*****
58 for x^k
59 *****/
60 typec power(typec x,typec k)
61 {
```

```

62     typec ans=1;
63     while(k)
64     {
65         if(k&1) ans*=x;
66         x*=x;
67         k>>=1;
68     }
69     return ans;
70 }
71 /*****
72 for (x^k)%mod
73 *****/
74 typec powermod(typec x,typec k,typec mod)
75 {
76     typec ans=1;
77     while(k)
78     {
79         if(k&1)
80         {
81             ans=(ans*x)%mod;
82         }
83         x=(x*x)%mod;
84         k>>=1;
85     }
86     return ans;
87 }
88 /*****
89 prime table
90 O(n)
91 prime[0] for the primes number in the PRIMERANGE
92 prime[i] for i th prime number
93 *****/
94 const int PRIMERANGE = 100000;
95 int prime[PRIMERANGE + 1];
96 int getPrime()
97 {
98     memset (prime, 0, sizeof (int) * (PRIMERANGE + 1));
99     for (int i = 2; i <= PRIMERANGE; i++)
100     {
101         if (!prime[i]) prime[++prime[0]] = i;
102         for (int j = 1; j <= prime[0] && prime[j] <= PRIMERANGE / i; j++)
103         {
104             prime[prime[j]*i] = 1;
105             if (i % prime[j] == 0) break;
106         }
107     }
108     return prime[0];
109 }
110 /*****
111 IsPrime table
112 IsPrime[i]==true if(i is not a prime)
113 Pmaxn is the range of numbers;
114 *****/
115 const int Pmaxn=20000000;
116 bool IsPrime[Pmaxn];
117 void Isprime()
118 {
119     memset(IsPrime,false,sizeof(IsPrime));
120     for(int i=2;i<=Pmaxn;i++)
121     {
122         if(!IsPrime[i])
123         {
124             for(int j=i;j<=Pmaxn;j++)

```

```

125         {
126             IsPrime[i*j]=true;
127         }
128     }
129 }
130 }
131 /*****
132 euler function
133 *****/
134 typec euler(typec x)
135 {
136     typec res=x;
137     for(int i=2;i*i<(typec)(x*1.0)+1;i++)
138     {
139         if(x%i==0)
140         {
141             res=res/i*(i-1);
142             while(x%i==0) x/=i;
143         }
144     }
145     if(x>1) res=res/x*(x-1);
146     return res;
147 }
148 ///you should init the prime table before
149 int factor[100][3], facCnt;
150 int getFactors(int x)
151 {
152     facCnt = 0;
153     int tmp = x;
154     for(int i = 1; prime[i] <= tmp / prime[i]; i++)
155     {
156         factor[facCnt][1] = 1, factor[facCnt][2] = 0;
157         if(tmp % prime[i] == 0)
158             factor[facCnt][0] = prime[i];
159         while(tmp % prime[i] == 0)
160             factor[facCnt][2]++, factor[facCnt][1] *= prime[i], tmp /= prime[i];
161         if(factor[facCnt][1] > 1) facCnt++;
162     }
163     if(tmp != 1)
164         factor[facCnt][0] = tmp, factor[facCnt][1] = tmp, factor[facCnt][2] = 1;
165     return facCnt;
166 }

```

6.3 筛法求素数

1. Pri数组中的素数是递增的,当i能整除Pri[j], 那么i*Pri[j+1]这个合数肯定被Pri[j]乘以某个数筛掉。 fir[i]代表i的最小素因子, 可以几乎线性的分解i. 2. 线性求1 n之间所有数的欧拉函数:

```
1      if ((i/fir[i])%fir[i]==0)
2          ouler[i]=ouler[i/fir[i]]*fir[i];
3      else ouler[i]=ouler[i/fir[i]]*(fir[i]-1);
```

3. 线性求1 n之间所有数约数个数 ei表示n的第i个质因数的个数.

```
1      if (i%Pri[j]==0)
2      {
3          divnum[i*Pri[j]]=divsum[i]/(e[i]+1)*(e[i]+2); //最小素因子次数加1
4          e[i*Pri[j]]=e[i]+1;
5      }
6      else
7      {
8          divnum[i*Pri[j]]=divnum[i]*2;    //满足积性函数条件
9          e[i*Pri[j]]=1;
10     }
```

```
1 # define RANGE 100000
2 int init(void)
3 {
4     int cnt=0,x;
5     for (int i=2;i<=RANGE;i++)
6     {
7         if (!Pri[i])
8         {
9             fir[i]=Pri[++cnt]=i;
10            pos[i]=cnt;
11        }
12        for (int j=1;j<=cnt;j++)
13        {
14            x=i*Pri[j];
15            if (x>RANGE) break;
16            Pri[x]=1;
17            fir[x]=Pri[j];
18            if (i%Pri[j]==0) break;
19        }
20    }
21    return cnt;
22 }
```

6.4 同余方程

```
1  # include <cstring>
2  # include <cstdio>
3  # include <cstdlib>
4  # include <iostream>
5  using namespace std;
6  typedef long long LOL;
7  LOL extend_gcd(LOL a, LOL b, LOL &x, LOL &y)
8  {
9      if (!b)
10     {
11         x=1,y=0;
12         return a;
13     }
14     LOL tem,r;
15     r=extend_gcd(b,a%b,x,y);
16     tem=x;
17     x=y;
18     y=tem-a/b*y;
19     return r;
20 }
21 int main (void)
22 {
23     int n;
24     while (scanf("%d",&n)!=EOF)
25     {
26         LOL r=0,a=1,r1,a1,x,y,tem,t;
27         int flag=0;
28         for (int i=1;i<=n;i++)
29         {
30             scanf("%lld%lld",&a1,&r1);
31             if (flag) continue;
32             tem=extend_gcd(a,a1,x,y);
33             if ((r1-r)%tem)
34                 flag=1;
35             else
36             {
37                 t=a1/tem;
38
39                 x=((r1-r)/tem*x%t+t)%t;
40                 r+=x*a;
41                 a*=(a1/tem);
42                 r=(r%a+a)%a;
43             }
44         }
45         if (!flag)
46             printf("%lld\n",r);
47         else printf("-1\n");
48     }
49     return 0;
50 }
```


6.5 高次同余方程

poj 3243 hdu 2815 poj 2417

这个是求扩展离散对数问题。 $X^Y \bmod Z = K$, 给出X,Z,K,求Y。

当Z时素数的时候直接用baby-step 算法即可。

方程 $a^x = b \pmod{c}$, 可以进行一系列的转化。假设

$$d = \gcd(a, c)$$

由

$$a^{x-1} * a = b \pmod{c}$$

知道 a^{x-1} 要存在必须满足 $\gcd(a, c) \mid b$, 如果满足这个条件, 那么我们可以在方程两边同时除以d, 方程是不变的。因为

$$a^x = b + k * c$$

再除以公约数d, 得到方程

$$a^{x-1} * \frac{a}{d} = \frac{b}{d} + k * \frac{c}{d}$$

假设我们除了k次, 那么方程转化为

$$a^{x-k} * \frac{a^k}{d^k} = \frac{b}{d^k} + k * \frac{c}{d^k}$$

令 $d = \frac{a^k}{d^k}, b' = \frac{b}{d^k}, c' = \frac{c}{d^k}, x' = x - k$, 方程转化为:

$$a^{x'} * d = b' \pmod{c'}$$

得到:

$$a^{x'} = b' * d^{-1} \pmod{c'}$$

现在直接用baby-step解方程

$$a^{x'} = b' * d^{-1} \pmod{c'}$$

即可。注意到 $x = x' + k$, 如果存在x小于k的解, 那么x'小于0, 但是baby-step是不会求负的次数的, 所以需要先枚举一下是否存在小于k的解, 由于输入的数据不会超过 10^9 的, 假设k不超过50进行枚举即可了。

```
1  #include <iostream>
2  #include <stdio.h>
3  #include <cmath>
4  using namespace std;
5  typedef long long LOL;
6  const int maxn = 65535;
7  struct Hashh
8  {
9      int a,b,next;
10 }Hash[maxn<<1];
11
12 int flag[maxn+100]; //注意要赋初值0
13 int top,idx; //注意要赋初值maxn
14
15 void ins(int a,int b)
16 {
17     int k=b&maxn;
18     if(flag[k]!=idx) //第b&maxn个槽为空
19     {
20         flag[k]=idx;
21         Hash[k].a=a;
22         Hash[k].b=b;
23         Hash[k].next=-1;
24         return ;
25     }
26     //第b&maxn个槽不为空
27     while(Hash[k].next!=-1) // 到链表的最后一个
28     {
29         if(Hash[k].b==b) return; //若b已经存在, 返回
30         k=Hash[k].next;
31     }
```

```

32     Hash[k].next=++top;
33     Hash[top].next=-1;
34     Hash[top].a=a;
35     Hash[top].b=b;
36 }
37
38 int find(int b)
39 {
40     int k=b&maxn;
41     if(flag[k]!=idx) return -1;//为空
42     while(k!=-1)
43     {
44         if(Hash[k].b==b)
45             return Hash[k].a;
46         k=Hash[k].next;
47     }
48     return -1;
49 }
50
51 int gcd(int a,int b)
52 {
53     if(b==0) return a;
54     return gcd(b,a%b);
55 }
56
57 int exgcd(int a,int b,int &x,int &y)
58 {
59     if(0==b)
60     {
61         x=1;
62         y=0;
63         return a;
64     }
65     int d=exgcd(b,a%b,x,y);
66     int t=x;
67     x=y;
68     y=t-a/b*y;
69     return d;
70 }
71
72 int exmod(LOL a,int b,int c)
73 {
74     LOL ret=1%c;
75     a%=c;
76     while(b)
77     {
78         if(b&1) ret=ret*a%c;
79         a=a*a%c;
80         b>>=1;
81     }
82     return ret;
83 }
84
85 int invmod(int a,int b,int n)
86 {
87     int x,y,e;
88     exgcd(a,n,x,y);
89     e=(LOL)x*b%n;
90     return e<0?e+n:e;
91 }
92
93 int babystep(int a,int b,int c)
94 {

```

```

95     top=maxn;idx++;
96     LOL buf=1%c,K;
97     LOL D=buf;
98     int tmp,w,d=0;
99     for(int i=0;i<=100;i++)
100     {
101
102         if(buf==b) return i;
103         buf=buf*a%c;
104     }
105     while((tmp=gcd(a,c))!=1)
106     {
107         if(b%tmp) return -1;
108         ++d;
109         c/=tmp;
110         b/=tmp;
111         D=D*a/tmp%c;
112     }
113     int m=(int)ceil(sqrt((double)(c-1)));
114     buf=1%c;
115     for(int i=0;i<=m;i++)
116     {
117         ins(i,buf);
118         buf=buf*a%c;
119     }
120     K=exmod((LOL)a,m,c);
121     for(int i=0;i<=m;i++)
122     {
123         tmp=invmod((int)D,b,c);
124         w=find(tmp);
125         if(tmp>=0 && (w!=-1))
126             return i*m+w+d;
127         D=(D*K%c+c)%c;
128     }
129     return -1;
130 }
131
132
133 int main()
134 {
135     int a,b,c,tmp;
136     // a^X = b % c
137     while(scanf("%d%d%d",&c,&a,&b)!=EOF && (a||b||c))
138     {
139         //~ if(b>=c)
140         //~ {
141             //~ printf("no solution\n");
142             //~ continue;
143         //~ }
144         tmp=babystep(a,b,c);
145         if(tmp<0) printf("no solution\n");
146         else printf("%d\n",tmp);
147     }
148     return 0;
149 }

```

6.6 高斯消元

6.6.1 高斯消元解实数方程

```
1 int Gauss(int n, int m, double a[][N*N])
2 {
3     int i, j, r, c, pvt;
4     double maxp;
5     for (r=0, c=0; r<n && c<m; ++r, ++c)
6     {
7         for (maxp=0, i=r; i < n; ++i)
8             if (fabs(a[i][c])>fabs(maxp)) maxp = a[pvt=i][c];
9         if (sgn(maxp)==0)
10        {
11            r--;
12            continue;
13        }
14        if (pvt != r)
15            for (j = r; j <= m; ++j) swap(a[r][j], a[pvt][j]);
16        for (j = c+1; j <= m; ++j)
17        {
18            a[r][j] /= maxp;
19            for (i = r+1; i < n; ++i)
20                a[i][j] -= a[i][c]*a[r][j];
21        }
22    }
23    for (i = r; i < n; ++i)
24        if (sgn(a[i][m])) return -1;
25    if (r < m) return m-r;
26    for (i = m-1; i >= 0; --i)
27        for (j = i+1; j < m; ++j)
28            a[i][m] -= a[j][m]*a[i][j];
29    return 0;
30 }
```

6.6.2 高斯消元解异或方程组

高斯消元解异或方程+枚举变元

```
1 #include<iostream>
2 #include<cstdio>
3 #include<cstring>
4 #include<vector>
5 #define PB push_back
6 #define MP make_pair
7 using namespace std;
8 const int M = 16;
9 const int maxn=M*M;
10 typedef pair<int,int> PII;
11 char s[M][M];
12 int num[4];
13 int dx[4][8];
14 int dy[4][8];
15 int N;
16 int a[maxn][maxn],b[maxn][maxn],c[maxn],d[maxn];
17 int n,m;
18 int get(int x,int y)
19 {
20     return x*m+y;
21 }
22 void build(int op)
23 {
24     memset(a,0,sizeof(a));
25     for(int i=0;i<n;i++){
26         for(int j=0;j<m;j++){
```

```

27         int u=get(i,j);
28         if(s[i][j]=='1'){
29             a[u][N]=1;
30         }else{
31             a[u][N]=0;
32         }
33         for(int k=0;k<num[op];k++){
34             int tx=i+dx[op][k];
35             int ty=j+dy[op][k];
36             if(tx<0||tx>=n||ty<0||ty>=m) continue;
37             int v=get(tx,ty);
38             a[v][u]=1;
39         }
40     }
41 }
42 }
43 void debug(int a[maxn][maxn])
44 {
45     for(int i=0;i<N;i++){
46         for(int j=0;j<=N;j++){
47             printf("%d ",a[i][j]);
48         }
49         printf("\n");
50     }
51 }
52 int gauss()
53 {
54     int col,row;
55     vector<int>vec;
56     vector<PII>use;
57     int ans=n*m+1;
58     for(col=0,row=0;col<N&&row<N;col++){
59         int mark=row;
60         for(int i=row+1;i<N;i++){
61             if(a[i][col]){
62                 mark=i;
63             }
64         }
65         for(int i=0;i<N+1;i++){
66             swap(a[row][i],a[mark][i]);
67         }
68         if(!a[row][col]){
69             vec.PB(col);
70             continue;
71         }
72         use.PB(MP(row,col));
73         for(int i=row+1;i<N;i++){
74             if(!a[i][col]) continue;
75             for(int j=0;j<N+1;j++){
76                 a[i][j]^=a[row][j];
77             }
78         }
79         row++;
80     }
81     //debug(a);
82     for(int i=row;i<N;i++){
83         if(a[i][N]){
84             return n*m+1;
85         }
86     }
87     int sz=vec.size();
88     for(int i=0;i<(1<<sz);i++){
89         for(int j=0;j<N;j++){

```

```

90         for(int k=0;k<N+1;k++){
91             b[j][k]=a[j][k];
92         }
93     }
94     memset(d,0,sizeof(d));
95     int cnt=0;
96     for(int j=0;j<sz;j++){
97         int u;
98         if(i&(1<j)){
99             u=1;
100         }else u=0;
101         d[vec[j]]=u;cnt+=d[vec[j]];
102     }
103     for(int j=row-1;j>=0;j--){
104         int y=b[j][N];
105         for(int k=use[j].second+1;k<N;k++){
106             y^=(b[j][k]&& d[k]);
107         }
108         d[use[j].second]=y;
109         cnt+=d[use[j].second];
110     }
111     ans=min(ans,cnt);
112 }
113 return ans;
114 }

```

6.6.3 模意义下求行列式的值

```

1  #include<iostream>
2  #include<cstring>
3  #include<cstdio>
4  using namespace std;
5  const int N = 210;
6  typedef long long LOL;
7  LOL a[N][N];
8  LOL solve(int n,LOL p)
9  {
10     LOL ans=1;
11     for(int i=1;i<=n;i++){
12         for(int j=i+1;j<=n;j++){
13             while(a[j][i]!=0){
14                 LOL t=a[i][i]/a[j][i];
15                 for(int k=i;k<=n;k++){
16                     a[i][k]=a[i][k]-a[j][k]*t;
17                     a[i][k]%=p;
18                 }
19                 for(int k=i;k<=n;k++){
20                     swap(a[i][k],a[j][k]);
21                 }
22                 ans=-ans;
23             }
24         }
25         if(a[i][i]==0) return 0;
26         else ans=((ans*a[i][i])%p+p)%p;
27     }
28     return ans;
29 }
30 int main(void)
31 {
32     int n;
33     LOL p;
34     while(scanf("%d%lld",&n,&p)!=EOF){

```

```

35         for(int i=1;i<=n;i++){
36             for(int j=1;j<=n;j++){
37                 scanf("%lld",&a[i][j]);
38                 a[i][j]%=p;
39             }
40         }
41         LOL ans=solve(n,p);
42         ans=(ans%p+p)%p;
43         cout<<ans<<endl;
44     }
45     return 0;
46 }

```

6.7 POLYA

6.7.1 概念

定义 1 置换: 设 X 是一个有限集, 取 X 为包含前 n 个正整数的集合 $1, 2, \dots, n$, X 的每个置换 i_1, i_2, \dots, i_n 可视为 X 到其自身定义的一个一对一的函数

$$f: X \rightarrow X$$

其中 $f(1) = i_1, f(2) = i_2, \dots, f(n) = i_n$

定义 2 置换群: 如果 S_n 中的置换的非空子集 G 满足如下三条性质, 则定义它为 X 的一个置换群:

- 1) (合成运算的封闭性) 对 G 中所有的置换 f 与 g , $f \circ g$ 也属于 G 。
- 2) (单位元) S_n 中的恒等置换 τ 属于 G 。
- 3) (逆元的封闭性) 对 G 中的每一个置换 f , 它的逆 f^{-1} 也属于 G 。

定义 3 着色等价: 令 G 是作用在集合 X 上的一个置换群, 通常取 X 为前 n 个正整数的集合。令 C 是 X 的一个着色集合。设 c_1, c_2 是 C 中的两种着色, 如果 G 中存在一个置换 f , 使得

$$f * c_1 = c_2$$

则称 c_1 (在 G 的作用下) 等价于 c_2 。

定义 4 稳定核: 使着色 c 保持不变的所有置换的集合 $G(c)$ 称为 c 的稳定核。

定理 1 对于每一种着色 c , c 的稳定核 $G(c)$ 是一个置换群, 而且对 G 中任意置换 f 与 g , $g * c = f * c$ 当且仅当 $f^{-1} \circ g$ 属于 $G(c)$ 。

证明:

- 1) 因为 f 和 g 都使 c 保持不变, 则 $(g \circ f)(c) = c$, 满足合成运算的封闭性。
 - 2) 显然单位元 τ 使所有着色不变。
 - 3) 如果 f 使得 c 不变, 那么 f^{-1} 也使得 c 不变, 于是 $G(c)$ 具有对逆元的封闭性。所以 $G(c)$ 是一个置换群。
- 假设 $f * c = g * c$, 可得

$$(f^{-1} \circ g) * c = f^{-1} * (g * c) = f^{-1} * (f * c) = (f^{-1} \circ f) * c = \tau * c = c$$

6.7.2 题目

POJ1286 题目大意: 用红蓝绿三种颜色去染 n 元环, 问方案数。

解法: 旋转同构 $\sum_{i=1}^n 3^{\gcd(n,i)}$ 翻转同构, 如果 n 为奇数, $3^{n/2+1}$, 如果 n 为偶数, $(n/2) * 3^{n/2} + (n/2) * 3^{n/2+1}$

POJ2409 题目大意: 用 c 种颜色染 n 元环, 问方案数。

解法: 旋转同构 $\sum_{i=1}^n c^{\gcd(n,i)}$ 翻转同构, 如果 n 为奇数, $c^{n/2+1}$, 如果 n 为偶数, $(n/2) * c^{n/2} + (n/2) * c^{n/2+1}$

POJ2154 题目大意: 用 n 种颜色染 n 元环, $1 \leq n \leq 1000000000$, 并且有3500组数据。

解法: 解法和上一个相同, 但是因为 n 过大, 枚举 n 的复杂度会超时, 我们转过来想对于 x 有多少 $i (1 \leq i \leq n)$, 使得 $\gcd(n, i) = x$, 这个的个数为 $\phi(n/x)$ ($\phi(x)$ 表示 x 的欧拉函数, 即小于等于 x 的数有多少个)。这样枚举 n 的约数, 然后就能算出结果, 复杂度为 $O(\sqrt{n})$

```
1 for(int i=1;i*i<=n;i++){
2     if(n%i==0&&i*i!=n){
3         ans+=gao(n,i-1)*euler(n/i);
4         ans+=gao(n,n/i-1)*euler(i);
5         ans%=p;
6     }
7     if(i*i==n){
8         ans+=gao(n,i-1)*euler(i);
9         ans%=p;
10    }
11 }
```

优化: 通过DFS直接枚举出 n 的所有约数和约数的欧拉函数。

```
1 int ans=0;
2 void dfs(int now,int euler,int val)
3 {
4     if(now>=facCnt){
5         ans=(ans+(gao(n,n/val-1)*euler)%mod;
6         ans%=mod;
7     }
```



```

7         return ;
8     }
9     int pre=1,sum=1;
10    for(int i=0;i<=fac[now][1];i++){
11        dfs(now+1,euler*pre%mod,val*sum);
12        if(i==0){
13            pre*=(fac[now][0]-1);
14        }else{
15            pre*=fac[now][0];
16        }
17        sum*=fac[now][0];
18    }
19 }

```

立方体染色 立方体面: $1*1^6 + 6*1^2*4^1 + 3*1^2*2^2 + 8*3^2 + 6*2^3$

立方体楞: $1*1^{12} + 6*4^3 + 3*2^6 + 8*3^4 + 6*2^5*1^2$

立方体点: $1*1^8 + 6*4^2 + 9*2^4 + 8*1^2*3^2$

对立方体面来说: 如果用三种颜色染色,

$$I = (3^6 + 6*3^3 + 3*3^4 + 8*3^2 + 6*3^3)/24 = 57$$

如果限定每种颜色用两次, 那么

$$I = (r+b+g)^6 + 6*(r+g+b)^2*(r^4+b^4+g^4) + 3*(r+b+g)^2*(r^2+b^2+g^2)^2 + 8*(r^3+b^3+g^3)^2 + 6*(r^2+b^2+g^2)^3 \quad (1)$$

然后方案数就是 $r^2*b^2*g^2$ 的系数

UVA11255 题目大意: 用三种颜色, 每种a,b,c个去染 $n = a + b + c$ 的环, 问方案数。

解法: 同上, 用dfs求系数。

```

1  int a[3],n;
2  LOL com[45][45];
3  LOL gao(int g,int t)
4  {
5      if(a[0]%t!=0||a[1]%t!=0||a[2]%t!=0) return 0;
6      LOL ans=0;
7      int ta=a[0]/t,tb=a[1]/t,tc=a[2]/t;
8      ans=com[ta+tb+tc][ta]*com[tb+tc][tb];
9      return ans;
10 }
11 int main(void)
12 {
13     int T;
14     for(int i=0;i<=40;i++){
15         com[i][0]=1;
16     }
17     for(int i=1;i<=40;i++){
18         for(int j=1;j<=i;j++){
19             com[i][j]=com[i-1][j]+com[i-1][j-1];
20         }
21     }
22     scanf("%d",&T);
23     while(T--){
24         n=0;
25         for(int i=0;i<3;i++){
26             scanf("%d",&a[i]);
27             n+=a[i];
28         }
29         LOL ans=0;
30         for(int i=1;i<=n;i++){
31             int g=gcd(i,n);
32             ans+=gao(g,n/g);
33         }
34         if(n%2==0){
35             ans+=n/2*gao(n/2,2);

```

```

36         for(int i=0;i<3;i++){
37             for(int j=0;j<3;j++){
38                 a[i]--;a[j]--;
39                 ans+=n/2*gao((n-2)/2,2);
40                 a[i]++;a[j]++;
41             }
42         }
43     }
44     else{
45         for(int i=0;i<3;i++){
46             a[i]--;
47             ans+=n*gao(n/2,2);
48             a[i]++;
49         }
50     }
51     cout<<ans/(2*n)<<endl;
52 }
53 return 0;
54 }
55 }

```

7 计算几何

7.1 几何基础

7.1.1 求两向量的叉积

```
1 | int cross(int x1,int y1,int x2,int y2) //int型
2 | {
3 |     return x1*y2-x2*y1;
4 | }
5 |
6 | double cross(double x1,double y1,double x2,double y2) //double型
7 | {
8 |     return x1*y2-x2*y1;
9 | }
10 |
11 | double cross(Point a,Point b) // 向量叉积
12 | {
13 |     return a.x*b.y-b.x*a.y;
14 | }
```

7.1.2 求平面两点欧氏距离

```
1 | double distance(Point a,Point b)
2 | {
3 |     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
4 | }
```

7.1.3 判断点是否在线段上

```
1 | int on_segment(Segment seg,Point k)
2 | {
3 |     Point a=seg.s,Point b=seg.e;
4 |
5 |     if (cross(k.x-a.x,k.y-a.y,b.x-a.x,b.y-a.y)==0&&
6 |         k.x>=min(a.x,b.x)&&k.x<=max(a.x,b.x)&&
7 |         k.y>=min(a.y,b.y)&&k.y<=max(a.y,b.y))
8 |
9 |         return 1;
10 |
11 |     else
12 |         return 0;
13 | }
```

7.1.4 判断两线段是否相交 (快速排斥+跨立实验)

```
1 | int segment_intersect(Segment seg1,Segment seg2) //endpoint exclusive
2 | {
3 |     Point s1=seg1.s,e1=seg1.e,s2=seg2.s,e2=seg2.e;
4 |
5 |     if (max(s1.x,e1.x)>min(s2.x,e2.x)&&
6 |         max(s2.x,e2.x)>min(s1.x,e1.x)&&
7 |         max(s1.y,e1.y)>min(s2.y,e2.y)&&
8 |         max(s2.y,e2.y)>min(s1.y,e1.y)&&
9 |         cross(e1.x-s1.x,e1.y-s1.y,s2.x-s1.x,s2.y-s1.y)*cross(e1.x-s1.x,e1.y-
10 |             s1.y,e2.x-s1.x,e2.y-s1.y)<0&&
11 |         cross(e2.x-s2.x,e2.y-s2.y,s1.x-s2.x,s1.y-s2.y)*cross(e2.x-s2.x,e2.y-
12 |             s2.y,e1.x-s2.x,e1.y-s2.y)<0)
13 |
14 |         return 1;
15 |
16 |     else
17 |         return 0;
18 | }
```

```

18 int segment_intersect(Segment seg1,Segment seg2) //endpoint inclusive
19 {
20     Point s1=seg1.s,e1=seg1.e,s2=seg2.s,e2=seg2.e;
21
22     if (max(s1.x,e1.x)>=min(s2.x,e2.x)&&
23         max(s2.x,e2.x)>=min(s1.x,e1.x)&&
24         max(s1.y,e1.y)>=min(s2.y,e2.y)&&
25         max(s2.y,e2.y)>=min(s1.y,e1.y)&&
26         cross(e1.x-s1.x,e1.y-s1.y,s2.x-s1.x,s2.y-s1.y)*cross(e1.x-s1.x,e1.y-
            s1.y,e2.x-s1.x,e2.y-s1.y)<=0&&
27         cross(e2.x-s2.x,e2.y-s2.y,s1.x-s2.x,s1.y-s2.y)*cross(e2.x-s2.x,e2.y-
            s2.y,e1.x-s2.x,e1.y-s2.y)<=0)
28
29         return 1;
30
31     else
32         return 0;
33 }

```

7.1.5 判断double型变量的符号

```

1 #define eps 1e-8
2 int dlcmp(double x)
3 {
4     return x<-eps?-1:x>eps;
5 }

```

7.1.6 求点到线段的最短距离(此做法过于naive,可用点积+叉积来求)

```

1 #include<cmath>
2 #include<algorithm>
3 #define eps 1e-6
4 struct Point
5 {
6     double x,y;
7 };
8
9 struct Segment
10 {
11     Point s,e;
12 };
13
14 struct Line //ax+by+c=0;
15 {
16     double a,b,c;
17     Line(double d1=1,double d2=-1,double d3=0) {a=d1;b=d2;c=d3;}
18 };
19
20 int dlcmp(double x)
21 {
22     return x<-eps?-1:x>eps;
23 }
24 double dis(Point a,Point b)
25 {
26     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
27 }
28
29 Line make_line(Point a,Point b)
30 {
31     Line l;
32     int sign=1;
33
34     l.a=b.y-a.y;
35

```

```

36     if (l.a<0)
37     {
38         sign=-1;
39         l.a=l.a*sign;
40     }
41
42     l.b=sign*(a.x-b.x);
43     l.c=sign*(a.y*b.x-a.x*b.y);
44     return l;
45 }
46
47 double dis_point_segment(Point a,Segment seg)
48 {
49     double A=dis(a,seg.s);
50     double B=dis(a,seg.e);
51     double C=dis(seg.s,seg.e);
52
53     if (!dlcmp(A+B-C))
54         return 0;
55
56     if (!dlcmp(A+C-B)||!dlcmp(B+C-A))
57         return min(A,B);
58
59     if (dlcmp(A*A+C*C-B*B)<=0||dlcmp(B*B+C*C-A*A)<=0)
60         return min(A,B);
61
62     Line l=make_line(seg.s,seg.e);
63     double t=fabs(a.x*l.a+a.y*l.b+l.c);
64     t/=sqrt(l.a*l.a+l.b*l.b);
65
66     return t;
67 }
68 }

```

7.1.7 求两线段间最短距离

```

1 double dis_segments(Segment seg1,Segment seg2)
2 {
3     double m1=dis_point_segment(seg1.s,seg2);
4
5     double m2=dis_point_segment(seg1.e,seg2);
6     double m3=dis_point_segment(seg2.s,seg1);
7     double m4=dis_point_segment(seg2.e,seg1);
8
9     return min(min(m1,m2),min(m3,m4));
10 }

```

7.1.8 判断两直线是否相交（共线及平行，若相交则求出交点）

```

1 #include<math.h>
2 #define eps 1e-8
3 struct Point
4 {
5     double x;
6     double y;
7 };
8
9 struct Line
10 {
11     Point s;
12     Point e;
13     Point v;
14 };
15

```

```

16 | Line l1,l2;
17 |
18 | double x,y交点坐标;//
19 |
20 | int line_intersect(Line l1,Line l2)
21 | {
22 |     Point vec;
23 |     double r;
24 |
25 |     vec.x=l1.s.x-l2.s.x;
26 |     vec.y=l1.s.y-l2.s.y;
27 |
28 |     if (fabs(l1.v.x*l2.v.y-l2.v.x*l1.v.y)<eps)
29 |     {
30 |         if (fabs(l1.v.x*vec.y-vec.x*l1.v.y)<eps)
31 |             return 2;    //共线
32 |         else
33 |             return 0;    //平行
34 |     }
35 |     else
36 |     {
37 |         r=((l1.s.x-l2.s.x)*l2.v.y-(l1.s.y-l2.s.y)*l2.v.x)/(l1.v.x*l2.v.y-l1.
38 |             v.y*l2.v.x);
39 |         x=-r*l1.v.x+l1.s.x;
40 |         y=-r*l1.v.y+l1.s.y;
41 |         return 1;        //相交
42 |     }
43 | }

```

7.1.9 求两直线交点

```

1 | Point line_intersect(Point s1,Point e1,Point s2,Point e2)
2 | {
3 |     Point v1,v2,res;
4 |     double r;
5 |
6 |     v1=s1-e1;
7 |     v2=s2-e2;
8 |
9 |     r=((s1.x-s2.x)*v2.y-(s1.y-s2.y)*v2.x)/(v1.x*v2.y-v1.y*v2.x);
10 |     res.x=-r*v1.x+s1.x;
11 |     res.y=-r*v1.y+s1.y;
12 |
13 |     return res;
14 | }

```

7.1.10 求两线段交点

```

1 | Point line_intersect(Point s1,Point e1,Point s2,Point e2)
2 | {
3 |     Point res;
4 |
5 |     double cs=fabs(cross(s2,e1,s1));
6 |     double ce=fabs(cross(e2,e1,s1));
7 |
8 |     res.x=(ce*s2.x+cs*e2.x)/(cs+ce);
9 |     res.y=(ce*s2.y+cs*e2.y)/(cs+ce);
10 |
11 |     return res;
12 | }

```

7.2 多边形

7.2.1 判断线段是否与矩形相交（包括线段在矩形内部）

```
1 struct Rectangle
2 {
3     Point lt;//lefttop
4     Point rb;//rightbottom
5 };
6
7 int segment_rectangle_intersect(Segment l,Rectangle r)
8 {
9     Segment d1,d2;//rectangle's diagonal
10
11     d1.s=r.lt;
12     d1.e=r.rb;
13     d2.s.x=d1.e.x;
14     d2.s.y=d1.s.y;
15     d2.e.x=d1.s.x;
16     d2.e.y=d1.e.y;
17
18     if (l.s.x>=r.lt.x&&1.s.x<=r.rb.x&&
19         l.s.y<=r.lt.y&&1.s.y>=r.rb.y||
20         l.e.x>=r.lt.x&&1.e.x<=r.rb.x&&
21         l.e.y<=r.lt.y&&1.e.y>=r.rb.y)
22
23         return 1;
24
25     if (segment_intersect(l,d1)|| segment_intersect(l,d2)) //
26         segment_intersect(endpoint inclusive)
27         return 1;
28
29     return 0;
30 }
```

7.2.2 判断点是否在多边形内部（包含在边上）,射线法

```
1 int point_on_segment(Point o,Point a,Point b)//点在线段上
2 {
3     if (dlcmp(cross(a-o,b-o))==0&&
4         o.x>=min(a.x,b.x)&&o.x<=max(a.x,b.x)&&
5         o.y>=min(a.y,b.y)&&o.y<=max(a.y,b.y))
6
7         return 1;
8     else
9         return 0;
10 }
11
12 int point_in_polygon(Point o,Point pln[],int n)//判断点在多边形内部
13 {
14     int i,j,cnt=0;
15     Point far(o,o.y);
16     int d1,d2,d3,d4;
17
18     for (i=0;i<n;i++)
19     {
20         j=(i+1)%n;
21         if (point_on_segment(o,pln[i],pln[j]))
22             return 1;
23
24         d1=dlcmp(cross(far-o,pln[i]-o));
25         d2=dlcmp(cross(far-o,pln[j]-o));
26         d3=dlcmp(cross(pln[j]-pln[i],o-pln[i]));
27         d4=dlcmp(cross(pln[j]-pln[i],far-pln[i]));
```

```

28
29         if (d1*d2<0&& d3*d4<0)
30             cnt++;
31         else if ((d1*d2==0&& d3*d4<0)&&
32                 dlcmp(o.y-max(pln[i].y,pln[j].y))==0)
33             cnt++;
34     }
35
36     if (cnt&1)
37         return 1;
38     else
39         return 0;
40 }

```

7.2.3 求简单多边形重心

```

1 Point get_center(Point pt[],int n)
2 {
3     double sum,area;
4     Point res(0,0),o(0,0);
5     int i;
6
7     sum=0;
8     for (i=0;i<n;i++)
9     {
10         area=cross(pt[i]-o,pt[(i+1)%n]-o);
11         res=res+(pt[i]+pt[(i+1)%n])/3*area;
12         sum+=area;
13     }
14
15     res=res/sum;
16     return res;
17 }

```

7.2.4 graham_scan求凸包

测试报告: hdu1348 1392

```

1 #include<cmath>
2 #include<algorithm>
3 #define eps 1e-6
4 #define N 50005
5 using namespace std;
6
7 struct Point
8 {
9     double x,y;
10 };
11
12 Point pt[N],pln[N];
13
14 int dlcmp(double x)
15 {
16     return x<-eps?-1:x>eps;
17 }
18
19 double cross(Point a,Point b,Point s)
20 {
21     double x1=a.x-s.x,y1=a.y-s.y;
22     double x2=b.x-s.x,y2=b.y-s.y;
23
24     return x1*y2-x2*y1;
25 }
26
27 double dis(Point a,Point b)

```



```

28 {
29     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
30 }
31
32 int cmp(Point a,Point b)
33 {
34     if (dlcmp(cross(a,b,pt[0]))==1||
35         dlcmp(cross(a,b,pt[0]))==0&&
36         dis(pt[0],a)<dis(pt[0],b))
37
38         return 1;
39     else
40         return 0;
41 }
42
43 int graham_scan(int n)
44 {
45     int i,top,t;
46
47     if (n<=1)
48         return n;
49
50     for (t=0,i=1;i<n;i++)
51         if (dlcmp(pt[i].y-pt[t].y)==-1||
52             dlcmp(pt[i].y-pt[t].y)==0&&
53             dlcmp(pt[i].x-pt[t].x)==-1)
54
55             t=i;
56
57     swap(pt[0],pt[t]);
58
59     sort(pt+1,pt+n,cmp);
60
61     top=2;
62     for (i=0;i<2;i++)
63         pln[i]=pt[i];
64
65     for (i=2;i<n;i++)
66     {
67         while (top>1&&dlcmp(cross(pln[top-1],pt[i],pln[top-2]))<=0)
68             top--;
69         pln[top++]=pt[i];
70     }
71
72     return top;
73 }

```

7.2.5 旋转卡壳求凸包直径

测试报告: poj2187

```

1 double rotating_calipers(Point pln[],int n)
2 {
3     int p,q;
4     double res;
5
6     pln[n]=pln[0];
7     res=0;
8
9     for (p=0,q=1;p<n;p++)
10    {
11        while (cross(pln[p+1],pln[q],pln[p])<
12              cross(pln[p+1],pln[q+1],pln[p]))

```

```

13
14         q=(q+1)%n;
15
16         res=max(res,max(dis(pln[p],pln[q]),dis(pln[p+1],pln[q+1])));
17     }
18
19     return res;
20 }

```

7.2.6 凸包间最短距离 (旋转卡壳)

测试报告: poj3608

```

1  const double eps=1e-6;
2  const double oo=1e100;
3  const double PI=3.141592657589793;
4
5  struct Point
6  {
7      double x,y;
8  };
9
10
11 struct Segment
12 {
13     Point s,e;
14 };
15
16 struct Line    //ax+by+c=0;
17 {
18     double a,b,c;
19
20     Line(double d1=1,double d2=-1,double d3=0) {a=d1;b=d2;c=d3;}
21 };
22
23 double dis(Point a,Point b)
24 {
25     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
26 }
27
28 double cross1(double x1,double y1,double x2,double y2)
29 {
30     return x1*y2-x2*y1;
31 }
32
33 double cross2(Point a,Point b)
34 {
35     return a.x*b.y-b.x*a.y;
36 }
37
38 double cross3(Point a,Point b,Point s)
39 {
40
41     double x1=a.x-s.x,y1=a.y-s.y;
42     double x2=b.x-s.x,y2=b.y-s.y;
43
44     return x1*y2-x2*y1;
45 }
46
47 int dlcmp(double x)
48 {
49     return x<-eps?-1:x>eps;
50 }

```

```

51
52 double polygon_area(Point pln[],int n)
53 {
54     double sum=0;
55     int i;
56
57     for (i=0;i<n;i++)
58         sum+=cross2(pln[i],pln[(i+1)%n]);
59
60     return sum/2;
61 }
62
63 void reverse_clockwise(Point pln[],int n)
64 {
65     int i;
66
67     for (i=0;i<=(n-1)/2;i++)
68         swap(pln[i],pln[n-i-1]);
69 }
70
71 double calculate_degree(Point a,Point b,double cur)
72 {
73     double ang,res;
74     Point p;
75
76     p.x=b.x-a.x;
77     p.y=b.y-a.y;
78
79     if (!dlcmp(p.x))
80     {
81         if (p.y>0)
82             ang=PI/2;
83         else
84             ang=3*PI/2;
85     }
86     else
87     {
88         ang=atan(p.y/p.x);
89         if (p.x<0)
90             ang+=PI;
91     }
92
93     while (ang<0)
94         ang+=2*PI;
95
96     if (ang>=PI)
97         cur+=PI;
98
99     if (ang>cur)
100         res=ang-cur;
101     else
102         res=PI-(cur-ang);
103
104     while (res>=PI)
105         res-=PI;
106
107     if (!dlcmp(res-PI))
108         res=0;
109
110     return res;
111 }
112
113

```

```

114
115 Line make_line(Point a,Point b)
116 {
117     Line l;
118     int sign=1;
119
120     l.a=b.y-a.y;
121
122     if (l.a<0)
123     {
124         sign=-1;
125         l.a=l.a*sign;
126     }
127
128     l.b=sign*(a.x-b.x);
129     l.c=sign*(a.y*b.x-a.x*b.y);
130
131     return l;
132 }
133
134 double dis_point_segment(Point a,Segment seg)
135 {
136     double A=dis(a,seg.s);
137     double B=dis(a,seg.e);
138     double C=dis(seg.s,seg.e);
139
140     if (!dlcmp(A+B-C))
141         return 0;
142
143     if (!dlcmp(A+C-B)||!dlcmp(B+C-A))
144         return min(A,B);
145
146     if (dlcmp(A*A+C*C-B*B)<=0||dlcmp(B*B+C*C-A*A)<=0)
147         return min(A,B);
148
149     Line l=make_line(seg.s,seg.e);
150     double t=fabs(a.x*l.a+a.y*l.b+l.c);
151     t/=sqrt(l.a*l.a+l.b*l.b);
152
153     return t;
154 }
155
156 double dis_segments(Segment seg1,Segment seg2)
157 {
158     double m1=dis_point_segment(seg1.s,seg2);
159     double m2=dis_point_segment(seg1.e,seg2);
160     double m3=dis_point_segment(seg2.s,seg1);
161     double m4=dis_point_segment(seg2.e,seg1);
162
163     return min(min(m1,m2),min(m3,m4));
164 }
165
166
167 double dis_polygons(Point pln1[],int n1,Point pln2[],int n2)
168 {
169     int i,j,k,p1=0,p2=0;
170     double res=oo,arg=0,cur=0;
171     double ang1,ang2,tmp;
172     Segment seg1,seg2;
173
174     if (polygon_area(pln1,n1)<0)
175         reverse_clockwise(pln1,n1);
176     if (polygon_area(pln2,n2)<0)

```

```

177         reverse_clockwise(pln2,n2);
178
179     for (i=1;i<n1;i++)
180         if (pln1[i].y<pln1[p1].y)
181             p1=i;
182
183     for (i=1;i<n2;i++)
184         if (pln2[i].y>pln2[p2].y)
185             p2=i;
186
187     while (arg<=360)
188     {
189         while (cur>=PI)
190             cur-=PI;
191
192         if (!dlcmp(cur-PI))
193             cur=0;
194
195         ang1=calculate_degree(pln1[p1],pln1[(p1+1)%n1],cur);
196         ang2=calculate_degree(pln2[p2],pln2[(p2+1)%n2],cur);
197
198         if (!dlcmp(ang1-ang2))
199         {
200             cur+=ang1;
201             arg+=ang1;
202
203             seg1.s=pln1[p1],seg1.e=pln1[(p1+1)%n1];
204             seg2.s=pln2[p2],seg2.e=pln2[(p2+1)%n2];
205
206             tmp=dis_segments(seg1,seg2);
207             res=min(res,tmp);
208
209             p1=(p1+1)%n1;
210             p2=(p2+1)%n2;
211         }
212         else if (dlcmp(ang1-ang2)>0)
213         {
214             cur+=ang2;
215             arg+=ang2;
216
217             seg2.s=pln2[p2],seg2.e=pln2[(p2+1)%n2];
218             tmp=dis_point_segment(pln1[p1],seg2);
219             res=min(tmp,res);
220
221             p2=(p2+1)%n2;
222         }
223         else
224         {
225             cur+=ang1;
226             arg+=ang1;
227             seg1.s=pln1[p1],seg1.e=pln1[(p1+1)%n1];
228             tmp=dis_point_segment(pln2[p2],seg1);
229
230             res=min(tmp,res);
231             p1=(p1+1)%n1;
232         }
233     }
234
235     return res;
236 }
237
238 main()
239 {

```

```

240     int n1,n2,i;
241     double ans;
242     Point pln1[N],pln2[N];
243
244     while (scanf("%d%d",&n1,&n2),n1||n2)
245     {
246         for (i=0;i<n1;i++)
247             scanf("%lf%lf",&pln1[i].x,&pln1[i].y);
248         for (i=0;i<n2;i++)
249             scanf("%lf%lf",&pln2[i].x,&pln2[i].y);
250
251         ans=dis_polygons(pln1,n1,pln2,n2);
252
253         printf("%.5lf\n",ans);
254     }
255
256     return 0;
257 }

```

7.2.7 判断两凸多边形是否相交 (graham_scan求凸包+枚举边、点)

```

1  #include<iostream>
2  #include<cstdio>
3  #include<algorithm>
4  #define N 110
5  using namespace std;
6
7  struct Point
8  {
9      int x,y;
10 };
11
12 struct Polygon
13 {
14     Point p[N];
15     int n;
16 };
17
18 Point pt[N];
19 int stack[N];
20
21 int cross(Point a,Point b,Point s)
22 {
23     return (a.x-s.x)*(b.y-s.y)-(b.x-s.x)*(a.y-s.y);
24 }
25
26 int dist(Point a,Point b)
27 {
28     return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
29 }
30
31 int cmp(Point a,Point b)
32 {
33     if (cross(a,b,pt[0])>0||cross(a,b,pt[0])==0&&dist(a,pt[0])<dist(b,pt[0]))
34         return 1;
35     else
36         return 0;
37 }
38
39 int on_segment(Point s,Point e,Point o)
40 {

```

```

41     if (cross(s,e,o)==0&&o.x>=min(s.x,e.x)&&
42         o.x<=max(s.x,e.x)&&o.y>=min(s.y,e.y)&&o.y<=max(s.y,e.y))
43
44         return 1;
45     else
46         return 0;
47 }
48
49 int graham_scan(int n)
50 {
51     int i,top,t;
52
53     if (n<=1)
54     {
55         stack[0]=0;
56         return n;
57     }
58
59     for (t=0,i=1;i<n;i++)
60         if (pt[i].y<pt[t].y||pt[i].y==pt[t].y&&pt[i].x<pt[t].x)
61             t=i;
62
63     swap(pt[0],pt[t]);
64
65     sort(pt+1,pt+n,cmp);
66
67     top=2;
68     for (i=0;i<2;i++)
69         stack[i]=i;
70
71     for (i=2;i<n;i++)
72     {
73         while (top>1&&cross(pt[stack[top-1]],pt[i],pt[stack[top-2]])<=0)
74             top--;
75         stack[top++]=i;
76     }
77
78     return top;
79 }
80
81 int segment_intersect(Point s1,Point e1,Point s2,Point e2)
82 {
83     if (max(s1.x,e1.x)>=min(s2.x,e2.x)&&
84         max(s2.x,e2.x)>=min(s1.x,e1.x)&&
85         max(s1.y,e1.y)>=min(s2.y,e2.y)&&
86         max(s2.y,e2.y)>=min(s1.y,e1.y)&&
87         (double)cross(s2,e1,s1)*(double)cross(e2,e1,s1)<=0&&
88         (double)cross(s1,e2,s2)*(double)cross(e1,e2,s2)<=0)
89
90         return 1;
91     else
92         return 0;
93 }
94
95 int point_inside(Point o,Polygon pln)
96 {
97     int i,a1=0,a2=0,n=pln.n;
98
99     pln.p[n]=pln.p[0];
100
101     for (i=0;i<n;i++)
102         a1+=abs(cross(pln.p[i],pln.p[i+1],o));
103     for (i=1;i<n;i++)

```

```

104         a2+=abs(cross(pln.p[i],pln.p[i+1],pln.p[0]));
105
106     if (a1==a2)
107         return 1;
108
109     else
110         return 0;
111 }
112
113 int convex_polygon_intersect(Polygon pln1,Polygon pln2)
114 {
115     int i,j;
116
117     pln1.p[pln1.n]=pln1.p[0];
118     pln2.p[pln2.n]=pln2.p[0];
119
120     for (i=0;i<pln1.n;i++)
121         for (j=0;j<pln2.n;j++)
122             if (segment_intersect(pln1.p[i],pln1.p[i+1],pln2.p[j],pln2.p[j
123                                     +1]))
124                 return 1;
125
126     if (point_inside(pln1.p[0],pln2)||point_inside(pln2.p[0],pln1))
127         return 1;
128
129     return 0;
130 }
131
132 int main()
133 {
134     int n,m,i,vertexnum,ans;
135     Polygon pln1,pln2;
136
137     while (cin>>n>>m,n||m)
138     {
139         for (i=0;i<n;i++)
140             cin>>pt[i].x>>pt[i].y;
141         vertexnum=graham_scan(n);
142         pln1.n=vertexnum;
143
144         for (i=0;i<vertexnum;i++)
145             pln1.p[i]=pt[stack[i]];
146
147         for (i=0;i<m;i++)
148             cin>>pt[i].x>>pt[i].y;
149
150         vertexnum=graham_scan(m);
151         pln2.n=vertexnum;
152
153         for (i=0;i<vertexnum;i++)
154             pln2.p[i]=pt[stack[i]];
155
156         if (pln1.n==1&&pln2.n==1)
157             ans=1;
158         else if (pln1.n==1&&pln2.n==2)
159         {
160             if (on_segment(pln2.p[0],pln2.p[1],pln1.p[0]))
161                 ans=0;
162             else
163                 ans=1;
164         }
165         else if (pln1.n==2&&pln2.n==1)
166         {

```



```

166         if (on_segment(pln1.p[0],pln1.p[1],pln2.p[0]))
167             ans=0;
168         else
169             ans=1;
170     }
171     else if (pln1.n==2&&pln2.n==2)
172     {
173         if (segment_intersect(pln1.p[0],pln1.p[1],pln2.p[0],pln2.p[1]))
174             ans=0;
175         else
176             ans=1;
177     }
178     else if (pln1.n==1)
179     {
180         if (point_inside(pln1.p[0],pln2))
181             ans=0;
182         else
183             ans=1;
184     }
185     else if (pln2.n==1)
186     {
187         if (point_inside(pln2.p[0],pln1))
188             ans=0;
189         else
190             ans=1;
191     }
192     else
193     {
194         if (convex_polygon_intersect(pln1,pln2)==0)
195             ans=1;
196         else
197             ans=0;
198     }
199
200     if (ans==0)
201         cout<<"NO"<<endl;
202     else
203         cout<<"YES"<<endl;
204 }
205
206 return 0;
207 }

```

7.3 半平面交

7.3.1 求多边形内核

测试报告: poj3130 3335 3384

复杂度 $O(n^2)$

```
1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  #include<cmath>
5  #define eps 1e-8
6  #define N 1010
7  using namespace std;
8
9  struct Point
10 {
11     double x,y;
12
13     Point(){};
14     Point(double a,double b):x(a),y(b){}
15
16     Point operator - (const Point a) const {return Point(x-a.x,y-a.y);}
17 };
18 Point kernel[N],pt[N];
19
20 int dlcmp(double x)
21 {
22     if (x<=-eps)
23         return -1;
24     else
25         return x>eps?1:0;
26 }
27
28 double cross(Point v1,Point v2) //向量叉积
29 {
30     return v1.x*v2.y-v2.x*v1.y;
31 }
32
33 Point line_intersect(Point s1,Point e1,Point s2,Point e2) //求两点所在直线的交点
34 {
35     Point v1,v2,res;
36     double r;
37
38     v1=s1-e1;
39     v2=s2-e2;
40
41     r=((s1.x-s2.x)*v2.y-(s1.y-s2.y)*v2.x)/(v1.x*v2.y-v1.y*v2.x);
42     res.x=-r*v1.x+s1.x;
43     res.y=-r*v1.y+s1.y;
44
45     return res;
46 }
47
48 void rev_points(Point p[],int n)
49 {
50     int i;
51
52     for (i=0;i<n/2;i++)
53         swap(p[i],p[n-i-1]);
54 }
55
56 double polygon_area(Point pln[],int n) //求多边形有向面积,顺时针为负,逆时针为正
```

```

58 {
59     int i;
60     double res=0;
61
62     for (i=0;i<n;i++)
63         res+=cross(pln[i],pln[(i+1)%n]);
64
65     return res/2.0;
66 }
67
68 void cut_polygon(Point org[],int on,Point des[],int &dn,Point s,Point e) //多
    边形切割
69 {
70     int i;
71     int d1,d2;
72
73     dn=0;
74
75     for (i=0;i<on;i++)
76     {
77         d1=dlcmp(cross(e-s,org[i]-s));
78         d2=dlcmp(cross(e-s,org[(i+1)%on]-s));
79
80         if (d1>=0)
81             des[dn++]=org[i];
82         if (d1*d2<0)
83             des[dn++]=line_intersect(s,e,org[i],org[(i+1)%on]);
84     }
85 }
86
87 void polygon_kernel(Point org[],int on,Point kernel[],int &kn) //求多边形内核
88 {
89     int i,j,dn;
90     Point des[N];
91
92     if (dlcmp(polygon_area(org,on))<0)
93         rev_points(org,on);
94
95     for (i=0;i<on;i++)
96         kernel[i]=org[i];
97     kn=on;
98
99     for (i=0;i<on;i++)
100     {
101         cut_polygon(kernel,kn,des,dn,org[i],org[(i+1)%on]);
102
103         for (j=0;j<dn;j++)
104             kernel[j]=des[j];
105         kn=dn;
106     }
107 }
108
109 //poj3335
110 int main()
111 {
112     int t,i,pn,kn;
113
114     scanf("%d",&t);
115     while (t--)
116     {
117         scanf("%d",&pn);
118
119         for (i=0;i<pn;i++)

```

```

120         scanf("%lf%lf",&pt[i].x,&pt[i].y);
121
122         polygon_kernel(pt,pn,kernel,kn);
123
124         if (kn==0)
125             printf("NO\n");
126         else
127             printf("YES\n");
128     }
129
130     return 0;
131 }

```

7.3.2 zzyO($n \cdot \log n$)做法

测试报告: poj2451

```

1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  #include<vector>
5  #include<string>
6  #include<queue>
7  #include<cmath>
8  #define N 60000
9  #define eps 1e-8
10 using namespace std;
11
12 int dlcmp(double x)
13 {
14     return x<-eps?-1:x>eps;
15 }
16
17 struct Point
18 {
19     double x,y;
20
21     Point(){}
22     Point(double a,double b):x(a),y(b){}
23
24     Point operator + (const Point a) const {return Point(x+a.x,y+a.y);}
25     Point operator - (const Point a) const {return Point(x-a.x,y-a.y);}
26 };
27
28 double cross(Point v1,Point v2)
29 {
30     return v1.x*v2.y-v2.x*v1.y;
31 }
32
33 Point line_intersect(Point s1,Point e1,Point s2,Point e2)
34 {
35     Point v1,v2,res;
36     double r;
37
38     v1=s1-e1;
39     v2=s2-e2;
40     r=((s1.x-s2.x)*v2.y-(s1.y-s2.y)*v2.x)/(v1.x*v2.y-v1.y*v2.x);
41
42     res.x=-r*v1.x+s1.x;
43     res.y=-r*v1.y+s1.y;
44
45     return res;
46 }

```

```

47
48 struct Vector
49 {
50     Point s,e;
51     double ang;
52
53     Vector(){}
54     Vector(Point a,Point b):s(a),e(b){ang=atan2(e.y-s.y,e.x-s.x);}
55
56     bool operator == (const Vector &v) const
57     {
58         return dlcmp(ang-v.ang)==0;
59     }
60
61     bool operator < (const Vector &v) const
62     {
63         if (dlcmp(ang-v.ang)==0)
64             return dlcmp(cross(v.e-v.s,e-v.s))>=0;
65         else
66             return dlcmp(ang-v.ang)<0;
67     }
68 };
69
70 struct HalfPlane
71 {
72     Vector plane[N];
73     int n;
74
75     HalfPlane():n(0){}
76
77     void add(Vector v)
78     {
79         plane[n++]=v;
80     }
81
82     void add(Point s,Point e)
83     {
84         Vector v(s,e);
85         plane[n++]=v;
86     }
87
88     int check(Vector &v1,Vector &v2,Vector &v0)
89     {
90         Point o=line_intersect(v1.s,v1.e,v2.s,v2.e);
91
92         return dlcmp(cross(o-v0.s,v0.e-v0.s))>0;
93     }
94
95     //首先需要保证存在交集
96     void work(Point pln[],int &pn)
97     {
98         int i,cnt,head,tail;
99         Vector deque[N];
100
101         sort(plane,plane+n);
102         for (cnt=i=1;i<n;i++)
103             if (!(plane[i]==plane[i-1]))
104                 plane[cnt++]=plane[i];
105         n=cnt;
106
107         head=1,tail=2;
108         deque[head]=plane[0];
109         deque[tail]=plane[1];

```

```

110
111     for (i=2;i<n;i++)
112     {
113         while (head<tail&&check(deque[tail-1],deque[tail],plane[i]))
114             tail--;
115         while (head<tail&&check(deque[head+1],deque[head],plane[i]))
116             head++;
117         deque[++tail]=plane[i];
118     }
119
120     while (head<tail&&check(deque[tail-1],deque[tail],deque[head]))
121         tail--;
122     while (head<tail&&check(deque[head+1],deque[head],deque[tail]))
123         head++;
124     deque[--head]=deque[tail];
125
126     for (pn=0,i=head+1;i<=tail;i++)
127         pln[pn++]=line_intersect(deque[i-1].s,deque[i-1].e,deque[i].s,
128                                 deque[i].e);
129 };
130
131 Point pln[N];
132 HalfPlane w;
133
134 double get_area(Point pln[],int n)
135 {
136     int i;
137     double res=0;
138
139     for (i=1;i<n-1;i++)
140         res+=cross(pln[i]-pln[0],pln[i+1]-pln[0]);
141
142     return fabs(res/2);
143 }
144
145 //poj2451
146 int main()
147 {
148     int n,pn,i;
149     Point s,e;
150     double ans;
151
152     s=Point(0,0);e=Point(10000,0);w.add(s,e);
153     s=Point(10000,0);e=Point(10000,10000);w.add(s,e);
154     s=Point(10000,10000);e=Point(0,10000);w.add(s,e);
155     s=Point(0,10000);e=Point(0,0);w.add(s,e);
156
157     scanf("%d",&n);
158     for (i=0;i<n;i++)
159     {
160         scanf("%lf%lf%lf%lf",&s.x,&s.y,&e.x,&e.y);
161         w.add(s,e);
162     }
163
164     w.work(pln,pn);
165     ans=get_area(pln,pn);
166
167     printf("%.1f\n",ans);
168
169     return 0;
170 }

```

7.3.3 两个简单多边形求面积并、交

```
1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  #include<cmath>
5  #define eps 1e-8
6  #define N 550
7  using namespace std;
8
9  struct Point
10 {
11     double x,y;
12
13     Point (){}
14     Point (double a,double b):x(a),y(b){}
15
16     Point operator - (const Point a) const {return Point(x-a.x,y-a.y);}
17 };
18
19 Point zero(0,0);
20 int dlcmp(double x) {return x<=-eps?-1:x>eps;}
21 double cross(Point v1,Point v2) {return v1.x*v2.y-v2.x*v1.y;}
22
23 struct Polygon
24 {
25     Point p[N];
26     int n;
27
28     Polygon():n(0){}
29     void clear(){n=0;}
30     void add(Point a){p[n++]=a;}
31
32     double area()
33     {
34         double res=0;
35         for (int i=1;i<n-1;i++)
36             res+=cross(p[i]-p[0],p[i+1]-p[0]);
37         return fabs(res/2);
38     }
39 };
40
41 Polygon A,B,rec;
42
43
44 Point line_intersect(Point s1,Point e1,Point s2,Point e2) //两直线交点
45 {
46     Point v1,v2,res;
47     double r;
48
49     v1=s1-e1;
50     v2=s2-e2;
51     r=((s1.x-s2.x)*v2.y-(s1.y-s2.y)*v2.x)/(v1.x*v2.y-v1.y*v2.x);
52     res.x=-r*v1.x+s1.x;
53     res.y=-r*v1.y+s1.y;
54
55     return res;
56 }
57
58 void cut(Point s,Point e) //半平面交
59 {
60     int i,j,d1,d2;
61     Polygon ker;
```

```

62
63     for (i=0;i<rec.n;i++)
64     {
65         j=(i+1)%rec.n;
66         d1=dlcmp(cross(e-s,rec.p[i]-s));
67         d2=dlcmp(cross(e-s,rec.p[j]-s));
68
69         if (d1>=0)
70             ker.add(rec.p[i]);
71         if (d1*d2<0)
72             ker.add(line_intersect(s,e,rec.p[i],rec.p[j]));
73     }
74
75     rec=ker;
76 }
77
78 double calc(Point p1,Point p2,Point q1,Point q2)
79 {
80     int dp=dlcmp(cross(p1,p2)),dq=dlcmp(cross(q1,q2));
81     int sgn=dp*dq;
82
83     if (sgn==0)
84         return 0;
85
86     rec.clear();
87     rec.add(zero); rec.add(p1); rec.add(p2);
88     if (dp<0)
89         swap(rec.p[1],rec.p[2]);
90     if (dq>0)
91     {
92         cut(zero,q1);
93         cut(q1,q2);
94         cut(q2,zero);
95     }
96     else
97     {
98         cut(zero,q2);
99         cut(q2,q1);
100        cut(q1,zero);
101    }
102
103    return sgn*rec.area();
104 }
105
106 double solve()
107 {
108     double res=A.area()+B.area();
109     double sum=0;
110     int i,j;
111
112     //对两个多边形三角剖分，分别求两个三角形的面积交
113     for (i=0;i<A.n;i++)
114         for (j=0;j<B.n;j++)
115             sum+=calc(A.p[i],A.p[(i+1)%A.n],B.p[j],B.p[(j+1)%B.n]);
116     res-=fabs(sum); //fabs(sum)为两个多边形的面积交
117
118     return res; //面积并
119 }
120 //hdu3060(题目数据有误)
121 int main()
122 {
123     int n,m,i;
124     Point pt;

```



```

125     double ans;
126
127     while (scanf("%d%d",&n,&m)!=EOF)
128     {
129         A.clear();B.clear();
130         for (i=0;i<n;i++)
131         {
132             scanf("%lf%lf",&pt.x,&pt.y);
133             A.add(pt);
134         }
135         for (i=0;i<m;i++)
136         {
137             scanf("%lf%lf",&pt.x,&pt.y);
138             B.add(pt);
139         }
140
141         ans=solve();
142
143         printf("%.2f\n",ans+eps);
144     }
145
146     return 0;
147 }

```

7.4 圆

7.4.1 点类

```
1 struct Point
2 {
3     double x,y;
4
5     Point(){}
6     Point(double a,double b):x(a),y(b){}
7     Point operator + (const Point a) const {return Point(x+a.x,y+a.y);}
8     Point operator - (const Point a) const {return Point(x-a.x,y-a.y);}
9     Point operator * (const double a) const {return Point(x*a,y*a);}
10    Point operator / (const double a) const {return Point(x/a,y/a);}
11
12    bool operator < (const Point a) const
13    {
14        if (dlcmp(x-a.x)==0)
15            return dlcmp(x-a.y)<0;
16        else
17            return dlcmp(x-a.x)<0;
18    }
19    bool operator == (const Point a) const
20    {
21        return !dlcmp(x-a.x)&&!dlcmp(y-a.y);
22    }
23
24    //向量长度定为d
25    Point trunc(double d)
26    {
27        double dis(Point,Point);
28        double len=dis(*this,Point(0,0));
29        return Point(x*d/len,y*d/len);
30    }
31
32    //坐标逆时针旋转a度
33    Point rotate(double a)
34    {
35        return Point(x*cos(a)-y*sin(a),y*cos(a)+x*sin(a));
36    }
37 };
38
39 double dis(Point a,Point b)
40 {
41     return sqrt(sqr(a.x-b.x)+sqr(a.y-b.y));
42 }
43
44 double cross(Point a,Point b,Point s)
45 {
46     double x1=a.x-s.x,y1=a.y-s.y;
47     double x2=b.x-s.x,y2=b.y-s.y;
48
49     return x1*y2-x2*y1;
50 }
51
52 double cross(Point a,Point b)
53 {
54     return a.x*b.y-b.x*a.y;
55 }
56
57 double dot(Point a,Point b,Point s)
58 {
59     double x1=a.x-s.x,y1=a.y-s.y;
```

```

60     double x2=b.x-s.x,y2=b.y-s.y;
61
62     return x1*x2+y1*y2;
63 }
64
65 double dot(Point a,Point b)
66 {
67     return a.x*b.x+a.y*b.y;
68 }

```

7.4.2 圆类

```

1  struct Circle
2  {
3      Point o;
4      double r;
5
6
7      Circle(){}
8      Circle(Point a,double l):o(a),r(l){}
9
10     double area(){return sqr(r)*PI;}
11 };
12
13 //判断圆a是否含于圆b
14 int inner_circle(Circle a,Circle b)
15 {
16     if (dlcmp(a.r-b.r)>0)
17         return 0;
18     return dlcmp(dis(a.o,b.o)+a.r-b.r)<=0;
19 }
20
21 //以base点为基点，极角排序，排序前base需赋初值
22 Point base;
23 int cmp(const Point a,const Point b)
24 {
25     return atan2(a.y-base.y,a.x-base.x)<atan2(b.y-base.y,b.x-base.x);
26 }
27
28 //向量a,b的夹角
29 double vec_angle(Point a,Point b)
30 {
31     double tmp=dot(a,b)/(dis(a,Point(0,0))*dis(b,Point(0,0)));
32     if (dlcmp(tmp-1)>=0) tmp=1;
33     if (dlcmp(tmp+1)<=0) tmp=-1;
34
35     return acos(tmp);
36 }
37
38 //计算由a到b逆时针方向的弓形面积
39 double arc_area(Point a,Point b,Circle c)
40 {
41     double theta=vec_angle(a-c.o,b-c.o);
42     double sf=sqr(c.r)*theta/2.0;
43     double st=sqr(c.r)*sin(theta)/2.0;
44
45     if (dlcmp(cross(a,b,c.o))>0)
46         return sf-st;
47     else
48         return c.area()-sf+st;
49 }
50
51 double arc_area(double th,double r)

```

```

52 | {
53 |     return 0.5*sqr(r)*(th-sin(th));
54 | }

```

7.4.3 圆面积交、并

测试报告: sgu435 hdu3239 spojCRCU spojCRCUT

```

1 | //求两圆交点, 排除相切的情况, 不考虑内含
2 | int inter_circle_or(Circle c1,Circle c2,Point &p1,Point &p2)
3 | {
4 |     double len=dis(c1.o,c2.o);
5 |
6 |     if (dlcmp(len-c1.r-c2.r)>=0)
7 |         return 0;
8 |
9 |     double s=(sqr(c1.r)-sqr(c2.r)+sqr(len))/len/2;
10 |    double h=sqrt(sqr(c1.r)-sqr(s));
11 |    Point vec=c2.o-c1.o;
12 |    Point p0=c1.o+vec.trunc(s);
13 |
14 |    p1=p0+vec.rotate(PI/2).trunc(h);
15 |    p2=p0-vec.rotate(PI/2).trunc(h);
16 |    return 1;
17 | }
18 |
19 | //求两圆交点, 不排除相切的情况, 不考虑内含
20 | int inter_circle_and(Circle c1,Circle c2,Point &p1,Point &p2)
21 | {
22 |     double len=dis(c1.o,c2.o);
23 |
24 |     if (dlcmp(len-c1.r-c2.r)>0)
25 |         return 0;
26 |
27 |     double s=(sqr(c1.r)-sqr(c2.r)+sqr(len))/len/2;
28 |     double h=sqrt(sqr(c1.r)-sqr(s));
29 |     Point vec=c2.o-c1.o;
30 |     Point p0=c1.o+vec.trunc(s);
31 |
32 |     p1=p0+vec.rotate(PI/2).trunc(h);
33 |     p2=p0-vec.rotate(PI/2).trunc(h);
34 |     return 1;
35 | }
36 |
37 | struct Circles
38 | {
39 |     int n;
40 |     Circle c[MAXN];
41 |
42 |     Circles():n(0){}
43 |     void add(Circle cc) {c[n++]=cc;}
44 |     void clear() {n=0;}
45 |
46 |     //初始化圆的面积并, 去掉能被其他圆覆盖的圆
47 |     void init_or()
48 |     {
49 |         char mark[MAXN]={0};
50 |
51 |         int i,j,cnt=0;
52 |
53 |         for (i=0;i<n;i++)
54 |             for (j=0;j<n;j++)
55 |                 if (i!=j&&!mark[j]&&inner_circle(c[i],c[j]))
56 |                     {

```

```

57         mark[i]=1;
58         break;
59     }
60
61     for (i=0;i<n;i++)
62         if (!mark[i])
63             c[cnt++]=c[i];
64     n=cnt;
65 }
66
67 //初始化圆的面积并，去掉能把其他圆覆盖的圆
68 void init_and()
69 {
70     char mark[MAXN]={0};
71
72     int i,j,cnt=0;
73
74     for (i=0;i<n;i++)
75         for (j=0;j<n;j++)
76             if (i!=j&&!mark[j]&&inner_circle(c[j],c[i]))
77             {
78                 mark[i]=1;
79                 break;
80             }
81
82     for (i=0;i<n;i++)
83         if (!mark[i])
84             c[cnt++]=c[i];
85     n=cnt;
86 }
87
88 //判断圆弧是否被其他圆覆盖
89 int isvalid_or(Point a,Point b,int num)
90 {
91     Point vec,p;
92     int i;
93
94     vec=a-b;
95     p=c[num].o+vec.rotate(PI/2).trunc(c[num].r);
96
97     for (i=0;i<n;i++)
98         if (i!=num&&dlcmp(dis(p,c[i].o)-c[i].r)<0)
99             return 0;
100     return 1;
101 }
102
103 //判断点是否被其他圆覆盖
104 int isvalid_and(Point a)
105 {
106     int i;
107
108     for (i=0;i<n;i++)
109         if (dlcmp(dis(a,c[i].o)-c[i].r)>0)
110             return 0;
111     return 1;
112 }
113
114 //判断圆弧是否被其他圆覆盖
115 int isvalid_and(Point a,Point b,int num)
116 {
117     Point vec,p;
118     int i;
119

```

```

120         vec=a-b;
121         p=c[num].o+vec.rotate(PI/2).trunc(c[num].r);
122
123         return isvalid_and(p);
124     }
125
126     //计算圆的面积并
127     double area_or()
128     {
129         int i,j,k;
130         vector<Point>s[MAXN];
131         Point a,b;
132         double sa=0,sp=0;
133
134         init_or();
135
136         for (i=0;i<n;i++)
137             for (j=i+1;j<n;j++)
138                 if (inter_circle_or(c[i],c[j],a,b))
139                 {
140                     s[i].push_back(a);
141                     s[i].push_back(b);
142                     s[j].push_back(a);
143                     s[j].push_back(b);
144                 }
145
146
147         for (i=0;i<n;i++)
148         {
149             if (s[i].empty())
150             {
151                 sa+=c[i].area();
152                 continue;
153             }
154
155             base=c[i].o;
156             sort(s[i].begin(),s[i].end(),cmp);
157             s[i].resize(unique(s[i].begin(),s[i].end())-s[i].begin());
158             if (s[i].front()==s[i].back())
159                 s[i].pop_back();
160
161             for (j=0;j<s[i].size();j++)
162             {
163                 k=(j+1)%s[i].size();
164
165                 if (isvalid_or(s[i][j],s[i][k],i))
166                 {
167                     sa+=arc_area(s[i][j],s[i][k],c[i]);
168                     sp+=cross(s[i][j],s[i][k],Point(0,0));
169                 }
170             }
171         }
172
173         return sa+fabs(sp)/2.0;
174     }
175
176     //计算圆的面积交，若交集为空，返回-1.0，若交集为一点，保存交点到res中
177     double area_and(Point &res)
178     {
179         int i,j,k;
180         vector<Point>s[MAXN];
181         Point a,b;
182         double sa=0,sp=0;

```

```

183
184     init_and();
185     if (n==1)
186         return c[0].area();
187
188     for (i=0;i<n;i++)
189         for (j=i+1;j<n;j++)
190             if (inter_circle_and(c[i],c[j],a,b))
191             {
192                 s[i].push_back(a);
193                 s[i].push_back(b);
194                 s[j].push_back(a);
195                 s[j].push_back(b);
196             }
197             else
198                 return -1.0;
199
200     for (i=0;i<n;i++)
201     {
202         base=c[i].o;
203         sort(s[i].begin(),s[i].end(),cmp);
204
205         s[i].resize(unique(s[i].begin(),s[i].end())-s[i].begin());
206         if (s[i].front()==s[i].back())
207             s[i].pop_back();
208
209         if (s[i].size()==1)
210         {
211             if (isvalid_and(s[i][0]))
212                 res=s[i][0];
213             continue;
214         }
215
216         for (j=0;j<s[i].size();j++)
217         {
218             if (isvalid_and(s[i][j]))
219                 res=s[i][j];
220
221             k=(j+1)%s[i].size();
222
223             if (isvalid_and(s[i][j],s[i][k],i))
224             {
225                 sa+=arc_area(s[i][j],s[i][k],c[i]);
226                 sp+=cross(s[i][j],s[i][k],Point(0,0));
227             }
228         }
229     }
230
231     return sa+fabs(sp)/2.0;
232 }
233
234
235 //计算被覆盖i次的面积
236 double ans[MAXN],pre[MAXN]; //ans[i]保存被覆盖i次的面积
237
238 void get_area()
239 {
240     int i,j,k;
241
242     memset(ans,0,sizeof(ans));
243
244     vector<pair<double,int> >v;
245

```

```

246     for (i=0;i<n;i++)
247     {
248         v.clear();
249         v.push_back(make_pair(-PI,1));
250         v.push_back(make_pair(PI,-1));
251
252         for (j=0;j<n;j++)
253             if (i!=j)
254             {
255                 Point q=c[j].o-c[i].o;
256                 double ab=dis(q,Point(0,0)),ac=c[i].r,bc=c[j].r;
257
258                 if (dlcmp(ab+ac-bc)<=0)
259                 {
260                     v.push_back(make_pair(-PI,1));
261                     v.push_back(make_pair(PI,-1));
262                     continue;
263                 }
264
265                 if (dlcmp(ab+bc-ac)<=0||dlcmp(ab-ac-bc)>0)
266                     continue;
267
268                 double th=atan2(q.y,q.x);
269                 double fai=acos((ac*ac+ab*ab-bc*bc)/(2.0*ac*ab));
270
271                 double a0=th-fai;
272                 if (dlcmp(a0+PI)<0)
273                     a0+=2*PI;
274                 double a1=th+fai;
275                 if (dlcmp(a1-PI)>0)
276                     a1-=2*PI;
277
278                 if (dlcmp(a0-a1)>0)
279                 {
280                     v.push_back(make_pair(a0,1));
281                     v.push_back(make_pair(PI,-1));
282                     v.push_back(make_pair(-PI,1));
283                     v.push_back(make_pair(a1,-1));
284                 }
285                 else
286                 {
287                     v.push_back(make_pair(a0,1));
288                     v.push_back(make_pair(a1,-1));
289                 }
290             }
291
292         sort(v.begin(),v.end());
293
294         int cur=0;
295
296         for (j=0;j<v.size();j++)
297         {
298             if (cur&&dlcmp(v[j].first-pre[cur]))
299             {
300                 ans[cur]+=arc_area(v[j].first-pre[cur],c[i].r);
301                 Point pa(c[i].o.x+c[i].r*cos(pre[cur]),c[i].o.y+c[i].r*
302                     sin(pre[cur]));
303                 Point pb(c[i].o.x+c[i].r*cos(v[j].first),c[i].o.y+c[i].r
304                     *sin(v[j].first));
305                 ans[cur]+=0.5*cross(pa,pb);
306             }
307             cur+=v[j].second;
308             pre[cur]=v[j].first;

```



```

307         }
308     }
309
310     for (i=1;i<=n;i++)
311         ans[i]-=ans[i+1];
312 }
313 };

```

7.4.4 简单多边形与圆求面积交

```

1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  #include<cmath>
5  #define N 200
6  #define eps 1e-8
7  using namespace std;
8
9  const double PI=acos(-1.0);
10
11 struct Point
12 {
13     double x,y;
14 };
15
16 Point pt[N];
17 int n;
18
19 int dlcmp(double x)
20 {
21     return x<-eps?-1:x>eps;
22 }
23
24 double sqr(double x)
25 {
26     return x*x;
27 }
28
29 double dis(Point a,Point b)
30 {
31     return sqrt(sqr(a.x-b.x)+sqr(a.y-b.y));
32 }
33
34 double outer(Point a,Point b,Point c)
35 {
36     return (a.x-c.x)*(b.y-c.y)-(a.y-c.y)*(b.x-c.x);
37 }
38
39 double inner(Point a,Point b,Point c)
40 {
41     return (a.x-c.x)*(b.x-c.x)+(a.y-c.y)*(b.y-c.y);
42 }
43
44 double calc_area(Point a,Point b,Point c,double r)
45 {
46     double A,B,C,x,y,tS;
47
48     A=dis(b,c);
49     B=dis(a,c);
50     C=dis(b,a);
51
52     if (A<r&&B<r)
53         return outer(a,b,c)/2;

```

```

54
55     else if (A<r&&B>=r)
56     {
57         x=(inner(a,c,b)+sqrt(sqr(r)*sqr(C)-sqr(outer(a,c,b))))/C;
58         tS=outer(a,b,c)/2;
59
60         return asin(tS*(1-x/C)*2/r/B)*sqr(r)/2+tS*x/C;
61     }
62     else if (A>=r&&B<r)
63     {
64         y=(inner(b,c,a)+sqrt(sqr(r)*sqr(C)-sqr(outer(b,c,a))))/C;
65         tS=outer(a,b,c)/2;
66
67         return asin(tS*(1-y/C)*2/r/A)*sqr(r)/2+tS*y/C;
68     }
69     else if (fabs(outer(a,b,c))>=r*C||inner(b,c,a)<=0||inner(a,c,b)<=0)
70     {
71         if (inner(a,b,c)<0)
72         {
73             if (outer(a,b,c)<0)
74                 return (-PI-asin(outer(a,b,c)/A/B))*sqr(r)/2;
75             else
76                 return (PI-asin(outer(a,b,c)/A/B))*sqr(r)/2;
77         }
78         else
79             return asin(outer(a,b,c)/A/B)*sqr(r)/2;
80     }
81     else
82     {
83         x=(inner(a,c,b)+sqrt(sqr(r)*sqr(C)-sqr(outer(a,c,b))))/C;
84         y=(inner(b,c,a)+sqrt(sqr(r)*sqr(C)-sqr(outer(b,c,a))))/C;
85         tS=outer(a,b,c)/2;
86
87         return (asin(tS*(1-x/C)*2/r/B)+asin(tS*(1-y/C)*2/r/A))*sqr(r)/2+tS
            *((y+x)/C-1);
88     }
89 }
90
91 //计算一般多边形与圆的交面积 (将多边形划分为三角形, 然后有向三角形与圆求有向面积交)
92 double solve(Point o,double r)
93 {
94     int i,j;
95     double res,sum;
96     Point tri[3];
97
98     res=0;
99     for (i=1;i<n-1;i++)
100     {
101         tri[0]=pt[0];
102         tri[1]=pt[i];
103         tri[2]=pt[i+1];
104         sum=0;
105
106         for (j=0;j<3;j++)
107             sum+=calc_area(tri[j],tri[(j+1)%3],o,r);
108
109         //sum为三角形与圆交的有向面积
110         res+=sum;
111     }
112
113     return fabs(res);
114 }
115

```

```

116 //poj3675
117 int main()
118 {
119     double x0,y0,v,vx,vy,g,r,t,theta,ans;
120     Point o;
121     int i;
122
123     o.x=o.y=0;
124
125     while (scanf("%lf",&r)!=EOF)
126     {
127         scanf("%d",&n);
128         for (i=0;i<n;i++)
129             scanf("%lf%lf",&pt[i].x,&pt[i].y);
130
131         ans=solve(o,r);
132
133         printf("%.2f\n",ans);
134     }
135
136     return 0;
137 }

```

7.4.5 求线段与圆的交点

若求直线与圆的交点类似，无需讨论 k_1 、 k_2 的取值范围

```

1 int inter_circle_segment(Circle c,Point a,Point b,Point &p1,Point &p2)
2 {
3     Point vec=b-a;
4     double A=sqr(vec.x)+sqr(vec.y);
5     double B=2*(vec.x*(a.x-c.o.x)+vec.y*(a.y-c.o.y));
6     double C=sqr(a.x-c.o.x)+sqr(a.y-c.o.y)-sqr(c.r);
7     double delta=sqr(B)-4*A*C;
8
9     if (dlcmp(delta)<0)
10         return 0;
11
12     double k1=(-B-sqrt(fabs(delta)))/(2*A);
13     double k2=(-B+sqrt(fabs(delta)))/(2*A);
14     int res=0;
15
16
17     if (dlcmp(k1)>=0&&dlcmp(k1-1)<=0)
18     {
19         res++;
20         p1=a+vec*k1;
21     }
22
23     if (dlcmp(k2)>=0&&dlcmp(k2-1)<=0)
24     {
25         res++;
26
27         if (res==1)
28             p1=a+vec*k2;
29         else
30             p2=a+vec*k2;
31     }
32
33     return res;
34 }

```

7.4.6 求两圆公切线

```

1 //求两相离的圆的两条内共切线
2 void get_InCommonTangent(Circle c1,Circle c2,Point &s1,Point &e1,Point &s2,
   Point &e2)
3 {
4     double l=dis(c1.o,c2.o);
5     double d=l*c1.r/(c1.r+c2.r);
6     double tmp=c1.r/d;
7     tmp=fix(tmp);
8     double theta=acos(tmp);
9     Point vec=c2.o-c1.o;
10
11     vec=vec.trunc(c1.r);
12     s1=c1.o+vec.rotate(theta);
13     s2=c1.o+vec.rotate(-theta);
14
15     vec=c1.o-c2.o;
16     vec=vec.trunc(c2.r);
17     e1=c2.o+vec.rotate(theta);
18     e2=c2.o+vec.rotate(-theta);
19 }
20
21 //求两相离的圆的两条外公切线
22 void get_OutCommonTangent(Circle c1,Circle c2,Point &s1,Point &e1,Point &s2,
   Point &e2)
23 {
24     double l=dis(c1.o,c2.o);
25     double d=fabs(c1.r-c2.r);
26     double theta=acos(d/l);
27
28     if (dlcmp(c1.r-c2.r)>0)
29         swap(c1,c2);
30
31     Point vec=c1.o-c2.o;
32     vec=vec.trunc(c1.r);
33     s1=c1.o+vec.rotate(theta);
34     s2=c1.o+vec.rotate(-theta);
35     vec=vec.trunc(c2.r);
36     e1=c2.o+vec.rotate(theta);
37     e2=c2.o+vec.rotate(-theta);
38 }

```

7.4.7 最小圆覆盖

测试报告: zoj1450 hysbz1336 1337 hdu3007 3932

```

1 #define eps 1e-8
2 #define MAX_P 2000
3 struct Point
4 {
5     double x,y;
6
7     Point operator-(Point &a)
8     {
9         Point t;
10
11         t.x=x-a.x;
12         t.y=y-a.y;
13
14         return t;
15     }
16 };
17
18 struct Circle
19 {
20     double r;

```

```

21     Point center;
22 };
23
24 struct Triangle
25 {
26     Point t[3];
27 };
28
29 Point pt[MAX_P];    //点集
30 Circle c;           //最小圆
31
32 double distance(Point a,Point b)
33 {
34     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
35 }
36
37 double cross(Point a,Point b)
38 {
39     return a.x*b.y-b.x*a.y;
40 }
41
42 double triangle_area(Triangle tri)    //三角形面积
43 {
44     Point v1=tri.t[1]-tri.t[0];
45     Point v2=tri.t[2]-tri.t[0];
46
47     return fabs(cross(v1,v2))/2;
48 }
49
50 Circle circumcircle_triangle(Triangle tri)    //三角形外接圆
51 {
52     Circle res;
53
54     double a,b,c,c1,c2;
55     double xA,yA,xB,yB,xC,yC;
56
57     a=distance(tri.t[0],tri.t[1]);
58     b=distance(tri.t[1],tri.t[2]);
59     c=distance(tri.t[2],tri.t[0]);
60
61     res.r=a*b*c/triangle_area(tri)/4;
62
63     xA=tri.t[0].x; yA=tri.t[0].y;
64     xB=tri.t[1].x; yB=tri.t[1].y;
65     xC=tri.t[2].x; yC=tri.t[2].y;
66
67     c1=(xA*xA+yA*yA-xB*xB-yB*yB)/2;
68     c2 = (xA*xA+yA*yA-xC*xC-yC*yC)/2;
69
70     res.center.x=(c1*(yA-yC)-c2*(yA-yB))/ ((xA-xB)*(yA-yC)-(xA-xC)*(yA-yB));
71     res.center.y = (c1*(xA-xC)-c2*(xA-xB))/ ((yA-yB)*(xA-xC)-(yA-yC)*(xA-xB));
72
73     return res;
74 }
75
76 Circle mincircle_triangle(int trinum, Triangle tri)
77 {
78     Circle res;
79
80     if (trinum==0)
81         res.r=-2;
82     else if (trinum==1)

```

```

83     {
84         res.center=tri.t[0];
85         res.r=0;
86     }
87     else if (trinum==2)
88     {
89         res.center.x=(tri.t[0].x+tri.t[1].x)/2;
90         res.center.y=(tri.t[0].y+tri.t[1].y)/2;
91         res.r=distance(tri.t[0],tri.t[1])/2;
92     }
93     else if (trinum==3)
94         res=circumcircle_triangle(tri);
95
96     return res;
97 }
98
99 void mincircle_pointset(int m,int trinum,Triangle tri) //求点集的最小覆盖圆
100 {
101     int i,j;
102     Point tmp;
103
104     c=mincircle_triangle(trinum,tri);
105
106     if (trinum==3)
107         return;
108
109     for (i=0;i<m;i++)
110         if (distance(pt[i],c.center)>c.r)
111         {
112             tri.t[trinum]=pt[i];
113
114             mincircle_pointset(i,trinum+1,tri);
115
116             tmp=pt[i];
117
118             for (j=i;j>=1;j--)
119                 pt[j]=pt[j-1];
120
121             pt[0]=tmp;
122         }
123 }
124
125 main()
126 {
127     int n,i,f1,f2;
128     Triangle tri;
129
130     while (scanf("%d%d%d",&f1,&f2,&n)!=EOF)
131     {
132         for (i=0;i<n;i++)
133             scanf("%lf%lf",&pt[i].x,&pt[i].y);
134
135         mincircle_pointset(n,0,tri);
136         printf("%lf %lf %lf\n",c.center.x,c.center.y,c.r);
137     }
138     return 0;
139 }

```

7.4.8 单位圆覆盖

测试报告: poj1981

```
1 #include<math.h>
```

```

2  #define eps 1e-8
3  #define MAX_P 505
4  const double r=1.0;//单位圆半径
5
6  struct Point
7  {
8      double x,y;
9  };
10
11 Point pt[MAX_P];
12
13 double distance(Point a,Point b)
14 {
15     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
16     //sqrt函数速度较慢，应尽量避免出现，此处可优化为距离的平方和的形式
17 }
18
19 Point find_center(Point a,Point b)
20 {
21     Point v,mid,center;
22     double d,s,ang;
23
24     v.x=a.x-b.x;
25     v.y=a.y-b.y;
26
27     mid.x=(a.x+b.x)/2;
28     mid.y=(a.y+b.y)/2;
29
30     d=distance(a,mid);
31     s=sqrt(r*r-d*d);          //优化为s=sqrt(r*r-d);
32
33     if (fabs(v.y)<eps)
34     {
35         center.x=mid.x;
36         center.y=mid.y+s;
37     }
38     else
39     {
40         ang=atan(-v.x/v.y);
41         center.x=mid.x+s*cos(ang);
42         center.y=mid.y+s*sin(ang);
43     }
44
45     return center;
46 }
47
48 main()
49 {
50
51     int n,i,j,k,ans,cnt;
52     double tmp;
53     Point center;
54
55     while (scanf("%d",&n),n)
56     {
57         for (i=0;i<n;i++)
58             scanf("%lf%lf",&pt[i].x,&pt[i].y);
59
60         ans=1;
61         for (i=0;i<n;i++)
62             for (j=i+1;j<n;j++)
63             {

```

```

64 |         if (distance(pt[i],pt[j])>2*r)    //优化
        为distance(pt[i],pt[j])>2*2*r*r
65 |         continue;
66 |
67 |         cnt=0;
68 |         center=find_center(pt[i],pt[j]);
69 |
70 |         for (k=0;k<n;k++)
71 |             if (distance(pt[k],center)<=r+eps)
72 |                 cnt++;
73 |
74 |         if (ans<cnt)
75 |             ans=cnt;
76 |     }
77 |
78 |     printf("%d\n",ans);
79 | }
80 |
81 | return 0;
82 | }

```


7.5 模拟退火

7.5.1 求多边形费马点

测试报告: poj2420

```
1 #include<iostream>
2 #include<cstdio>
3 #include<cmath>
4 #define eps 1e-6
5 #define N 105
6 using namespace std;
7
8 struct Point
9 {
10     double x,y;
11 };
12
13
14 double point_dis(Point a,Point b)
15 {
16     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
17 }
18
19
20 double sum_dis(Point pt[],int n,Point o)
21 {
22     double res=0;
23     int i;
24
25     for (i=0;i<n;i++)
26         res+=point_dis(pt[i],o);
27
28     return res;
29 }
30
31 double polygon_Fermatpoint(Point pln[],int n)
32 {
33     Point cp,np,tmp;
34     double min,step,d;
35     int flag;
36
37     cp=pln[0]; //cp保存当前更新后最优的费马点
38     min=sum_dis(pln,n,cp);
39     step=10000; //选取坐标范围的最大值
40
41     while (step>eps)
42     {
43         flag=1;
44         while (flag)
45         {
46             flag=0;
47             np=cp;
48
49             tmp=cp,tmp.x+=step;
50             d=sum_dis(pln,n,tmp);
51
52             if (min>d)
53                 min=d, np=tmp, flag=1;
54
55             tmp=cp,tmp.x-=step;
56             d=sum_dis(pln,n,tmp);
57
58             if (min>d)
```

```

59         min=d, np=tmp,flag=1;
60
61         tmp=cp,tmp.y+=step;
62         d=sum_dis(pln,n,tmp);
63
64         if (min>d)
65             min=d, np=tmp,flag=1;
66
67         tmp=cp,tmp.y-=step;
68         d=sum_dis(pln,n,tmp);
69
70         if (min>d)
71             min=d, np=tmp,flag=1;
72
73         cp=np;
74     }
75
76     step*=0.98;    //系数根据精度要求修改
77 }
78
79     return min;
80 }
81
82 main()
83 {
84     int n,i;
85     double min;
86     Point pt[N];
87
88     cin>>n;
89
90     for (i=0;i<n;i++)
91         cin>>pt[i].x>>pt[i].y;
92
93     min=polygon_Fermatpoint(pt,n);
94
95     printf("%.0f\n",min);
96
97     return 0;
98 }

```

7.5.2 最小球覆盖

测试报告: poj2069 cf106E

```

1  #include<iostream>
2  #include<cstdio>
3  #include<cmath>
4  #define oo 1e20
5  #define eps 1e-10
6  #define N 105
7  using namespace std;
8
9  struct Point
10 {
11     double x,y,z;
12 };
13
14 double dis(Point a,Point b)
15 {
16     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y)+(a.z-b.z)*(a.z-b.z));
17 }

```

```

18
19 int max_dis(Point pt[],int n,Point o)
20 {
21     int i,res;
22     double max,tmp;
23
24     max=0;
25     res=0;
26
27     for (i=0;i<n;i++)
28     {
29         tmp=dis(pt[i],o);
30
31         if (max<tmp)
32         {
33             max=tmp;
34             res=i;
35         }
36     }
37
38     return res;
39 }
40
41 int main()
42 {
43     Point pt[N],o;
44     int n,i,t;
45     double dx,dy,dz,step,r,tmp;
46
47     cin>>n;
48
49     for (i=0;i<n;i++)
50         cin>>pt[i].x>>pt[i].y>>pt[i].z;
51
52     step=10000;    //step选取最大的坐标范围
53     r=0;
54
55     if (n==1)
56     {
57         o.x=pt[0].x;
58         o.y=pt[0].y;
59         o.z=pt[0].z;
60     }
61     else
62     {
63         o.x=o.y=o.z=0;
64         while (step>eps)
65         {
66             t=max_dis(pt,n,o);
67             tmp=dis(pt[t],o);
68
69             if (r>tmp)
70                 r=tmp;
71
72             dx=(pt[t].x-o.x)/tmp;
73             dy=(pt[t].y-o.y)/tmp;
74             dz=(pt[t].z-o.z)/tmp;
75
76             o.x+=step*dx;
77             o.y+=step*dy;
78             o.z+=step*dz;
79
80             step*=0.9993;    //系数的选取根据具体精度调整

```

```
81 |         }
82 |     }
83 |     printf("%.6f %.6f %.6f\n",o.x,o.y,o.z);
84 |
85 |     return 0;
86 | }
```

7.6 三维几何

7.6.1 三维凸包

测试报告: hdu3662 poj3528

```
1 #include<iostream>
2 #include<cstdio>
3 #include<cstring>
4 #include<cmath>
5 #include<algorithm>
6 #define N 505
7 #define eps 1e-8
8 using namespace std;
9
10 struct Point
11 {
12     double x,y,z;
13     Point(){}
14     Point (double px,double py,double pz):x(px),y(py),z(pz){}
15     Point operator - (const Point p)
16     {
17         return Point(x-p.x,y-p.y,z-p.z);
18     }
19     Point operator * (const Point p)
20     {
21         return Point(y*p.z-z*p.y,z*p.x-x*p.z,x*p.y-y*p.x);
22     }
23     double operator ^ (const Point p)
24     {
25         return x*p.x+y*p.y+z*p.z;
26     }
27 };
28
29 struct ConvexPolygon3D
30 {
31     struct Face
32     {
33         int a,b,c;
34         bool flag;
35     };
36
37     int n;
38     Point pt[N];
39     int tri_num;
40     Face face[8*N];
41     int g[N][N];
42
43     double veclen(const Point &p)
44     {
45         return sqrt(p.x*p.x+p.y*p.y+p.z*p.z);
46     }
47
48     Point cross(const Point &a, const Point &b, const Point &c)
49     {
50         return Point((b.y-a.y)*(c.z-a.z)-(b.z-a.z)*(c.y-a.y),-((b.x-a.x)*(c.
51             z-a.z)-(b.z-a.z)*(c.x-a.x)),(b.x-a.x)*(c.y-a.y)-(b.y-a.y)*(c.x-a.
52             x));
53     }
54
55     double tri_area(Point a,Point b,Point c)
56     {
57         return veclen((a-c)*(b-c))/2;
```

```

58 double tetrahedron_volume(Point a,Point b,Point c,Point d)
59 {
60     return ((b-a)*(c-a)^(d-a))/6;
61 }
62
63 double dlcmp(Point &p,Face &f)
64 {
65     Point m=pt[f.b]-pt[f.a];
66     Point n=pt[f.c]-pt[f.a];
67     Point t=p-pt[f.a];
68
69     return (m*n)^t;
70 }
71
72 void deal(int a,int b,int p)
73 {
74     int f=g[a][b];
75     Face add;
76
77     if (face[f].flag)
78     {
79         if (dlcmp(pt[p],face[f])>eps)
80             dfs(p,f);
81         else
82         {
83             add.a=b;
84             add.b=a;
85             add.c=p;
86             add.flag=1;
87             g[p][b]=g[a][p]=g[b][a]=tri_num;
88             face[tri_num++]=add;
89         }
90     }
91 }
92
93 void dfs(int p,int now)
94 {
95     face[now].flag=0;
96     deal(face[now].b,face[now].a,p);
97     deal(face[now].c,face[now].b,p);
98     deal(face[now].a,face[now].c,p);
99 }
100
101 bool same(int s,int t)
102 {
103     Point &a=pt[face[s].a];
104     Point &b=pt[face[s].b];
105     Point &c=pt[face[s].c];
106
107     bool res=fabs(tetrahedron_volume(a,b,c,pt[face[t].a]))<eps&&
108             fabs(tetrahedron_volume(a,b,c,pt[face[t].b]))<eps&&
109             fabs(tetrahedron_volume(a,b,c,pt[face[t].c]))<eps;
110
111     return res;
112 }
113
114 void solve()
115 {
116     int i,j,tmp;
117     Face add;
118     bool flag;
119
120     tri_num=0;

```

```

121
122     if (n<4)
123         return;
124
125     flag=true;
126     for (i=1;i<n;i++)
127         if (veclen((pt[0]-pt[1])*(pt[1]-pt[i]))>eps)
128             {
129                 swap(pt[2],pt[i]);
130                 flag=false;
131                 break;
132             }
133
134     if (flag)
135         return;
136
137     flag=true;
138     for (i=2;i<n;i++)
139         if (fabs((pt[0]-pt[1])*(pt[1]-pt[2])^(pt[0]-pt[i]))>eps)
140             {
141                 swap(pt[3],pt[i]);
142                 flag=false;
143                 break;
144             }
145
146     if (flag)
147         return;
148
149     flag=true;
150     for (i=3;i<n;i++)
151         if (veclen(pt[0]-pt[i])>eps)
152             {
153                 swap(pt[1],pt[i]);
154                 flag=false;
155                 break;
156             }
157
158     if (flag)
159         return;
160
161     for (i=0;i<4;i++)
162     {
163         add.a=(i+1)%4;
164         add.b=(i+2)%4;
165         add.c=(i+3)%4;
166         add.flag=true;
167
168         if (dlcmp(pt[i],add)>0)
169             swap(add.b,add.c);
170
171         g[add.a][add.b]=g[add.b][add.c]=g[add.c][add.a]=tri_num;
172         face[tri_num++]=add;
173     }
174
175     for (i=4;i<n;i++)
176         for (j=0;j<tri_num;j++)
177             if (face[j].flag&&dlcmp(pt[i],face[j])>eps)
178                 {
179                     dfs(i,j);
180                     break;
181                 }
182
183     tmp=tri_num;

```

```

184         for (i=tri_num=0;i<tmp;i++)
185             if (face[i].flag)
186                 face[tri_num++]=face[i];
187     }
188
189     double area()
190     {
191         double res=0;
192
193         if (n==3)
194         {
195             Point p=cross(pt[0],pt[1],pt[2]);
196             res=veclen(p)/2;
197         }
198         else
199         {
200             for (int i=0;i<tri_num;i++)
201                 res+=tri_area(pt[face[i].a],pt[face[i].b],pt[face[i].c]);
202         }
203
204         return res;
205     }
206
207     double volume()
208     {
209         double res=0;
210         Point tmp(0,0,0);
211
212         for (int i=0;i<tri_num;i++)
213             res+=tetrahedron_volume(tmp,pt[face[i].a],pt[face[i].b],pt[face[i].c]);
214
215         return fabs(res);
216     }
217
218     Point get_center() //凸包重心
219     {
220         Point res(0,0,0),o(0,0,0),p;
221         double sum,vol;
222         int i;
223
224         sum=0;
225         for (i=0;i<tri_num;i++)
226         {
227             vol=tetrahedron_volume(o,pt[face[i].a],pt[face[i].b],pt[face[i].c]);
228             sum+=vol;
229             p=(pt[face[i].a]+pt[face[i].b]+pt[face[i].c])/4;
230             p.x*=vol; p.y*=vol; p.z*=vol;
231             res=res+p;
232         }
233
234         res=res/sum;
235         return res;
236     }
237
238     int triangle_num()
239     {
240         return tri_num;
241     }
242
243     int polygon_num()
244     {

```



```

245         int i,j,res,flag;
246
247         res=0;
248         for (i=0;i<tri_num;i++)
249         {
250             flag=1;
251             for (j=0;j<i;j++)
252                 if (same(i,j))
253                 {
254                     flag=0;
255                     break;
256                 }
257             res+=flag;
258         }
259
260         return res;
261     }
262 };
263
264 ConvexPolygon3D hull;
265
266 //点p到平面abc的距离
267 double dis_point_face(Point p,Point a,Point b,Point c)
268 {
269     Point vec=(b-a)*(c-a);
270     Point t=a-p;
271     double tmp=(vec^t)/(vec.len()*t.len());
272
273     return fabs(t.len()*tmp);
274 }
275
276 int main()
277 {
278     int i;
279
280     while(scanf("%d",&hull.n)!=EOF)
281     {
282
283         for(i=0;i<hull.n;i++)
284             scanf("%lf%lf%lf",&hull.pt[i].x,&hull.pt[i].y,&hull.pt[i].z);
285         hull.solve();
286
287         printf("%.3f\n",hull.area());
288     }
289
290     return 0;
291 }

```

7.6.2 求两球体积并

测试报告: zoj3500

```

1  #define PI (acos(-1.0))
2
3  struct Point
4  {
5      double x,y,z;
6  };
7
8  double dis(Point a,Point b)
9  {
10     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y)+(a.z-b.z)*(a.z-b.z))
        ;

```

```

11 }
12
13 double ball_volume_combination(Point o1,double r1,Point o2,double r2)
14 {
15     double d,R,r,p,l,H,h,res;
16     R=max(r1,r2),r=min(r1,r2);
17     d=dis(o1,o2);
18     res=PI*(R*R*R+r*r*r)*4/3;
19
20     if (R+r>d)
21     {
22         if (R-r<=d)
23         {
24             p=(R+r+d)/2;
25             l=sqrt(p*(p-R)*(p-r)*(p-d))*2/d;
26             H=sqrt(R*R-l*l);
27             h=sqrt(r*r-l*l);
28
29             res-=PI*(R*R*R*2/3-R*R*H+H*H*H/3);
30
31             if (R*R-r*r<=d*d)
32                 res-=PI*(r*r*r*2/3-r*r*h+h*h*h/3);
33             else
34                 res-=PI*r*r*r*4/3-PI*(r*r*r*2/3-r*r*h+h*h*h/3);
35         }
36         else
37             res-=PI*r*r*r*4/3;
38     }
39
40     return res;
41 }

```

7.7 三维仿射变换

```
1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  #include<vector>
5  #include<stack>
6  #include<cmath>
7  #define eps 1e-6
8  #define SZ 10
9  using namespace std;
10
11 const double PI=acos(-1.0);
12 int dlcmp(double x) {return x<-eps?-1:x>eps;}
13 double sqr(double x) {return x*x;}
14
15 struct Matrix
16 {
17     double m[SZ][SZ];
18
19     void Identity()
20     {
21         for (int i=0;i<SZ;i++)
22             for (int j=0;j<SZ;j++)
23                 m[i][j]=(i==j?1.0:0.0);
24     }
25
26     Matrix (){Identity();}
27
28     Matrix operator * (const Matrix &a)
29     {
30         Matrix res;
31
32         for (int i=0;i<SZ;i++)
33             for (int j=0;j<SZ;j++)
34             {
35                 res.m[i][j]=0;
36                 for (int k=0;k<SZ;k++)
37                     res.m[i][j]+=m[i][k]*a.m[k][j];
38             }
39
40         return res;
41     }
42 };
43
44 struct Point
45 {
46     double x,y,z;
47
48     Point (){}
49     Point (double a,double b,double c):x(a),y(b),z(c){}
50
51     Point operator + (const Point &a) const {return Point(x+a.x,y+a.y,z+a.z)}
52     Point operator - (const Point &a) const {return Point(x-a.x,y-a.y,z-a.z)}
53
54     Point operator * (const Matrix &a) const
55     {
56         Point res;
57
58         res.x=x*a.m[0][0]+y*a.m[1][0]+z*a.m[2][0]+a.m[3][0];
59         res.y=x*a.m[0][1]+y*a.m[1][1]+z*a.m[2][1]+a.m[3][1];
```

```

60         res.z=x*a.m[0][2]+y*a.m[1][2]+z*a.m[2][2]+a.m[3][2];
61
62         return res;
63     }
64
65     void norm()
66     {
67         double len=sqrt(sqr(x)+sqr(y)+sqr(z));
68         x/=len; y/=len; z/=len;
69     }
70 };
71
72 stack<Matrix>sm;
73 stack<int>sn;
74
75 Matrix pow(Matrix a,int k)
76 {
77     Matrix res;
78
79     while (k)
80     {
81         if (k&1)
82             res=res*a;
83         a=a*a;
84         k/=2;
85     }
86
87     return res;
88 }
89
90 Matrix get_trans(Point v)
91 {
92     Matrix res;
93     res.m[3][0]=v.x; res.m[3][1]=v.y; res.m[3][2]=v.z;
94
95     return res;
96 }
97
98 Matrix get_scale(Point v)
99 {
100     Matrix res;
101     res.m[0][0]=v.x; res.m[1][1]=v.y; res.m[2][2]=v.z;
102
103     return res;
104 }
105
106 Matrix get_rotate(Point v,double ang)
107 {
108     Matrix res;
109     double d=ang/180*PI;
110     v.norm();
111
112     res.m[0][0]=(1-cos(d))*v.x*v.x+cos(d);
113     res.m[0][1]=(1-cos(d))*v.x*v.y+sin(d)*v.z;
114     res.m[0][2]=(1-cos(d))*v.x*v.z-sin(d)*v.y;
115     res.m[1][0]=(1-cos(d))*v.y*v.x-sin(d)*v.z;
116     res.m[1][1]=(1-cos(d))*v.y*v.y+cos(d);
117     res.m[1][2]=(1-cos(d))*v.y*v.z+sin(d)*v.x;
118     res.m[2][0]=(1-cos(d))*v.z*v.x+sin(d)*v.y;
119     res.m[2][1]=(1-cos(d))*v.z*v.y-sin(d)*v.x;
120     res.m[2][2]=(1-cos(d))*v.z*v.z+cos(d);
121
122     return res;

```

```

123 }
124
125 int main()
126 {
127     int i,n,m;
128     Matrix cur,tmp;
129     double ang;
130     Point p,v;
131     char str[100];
132
133     while (scanf("%d",&n),n)
134     {
135         while (!sm.empty())
136             sm.pop();
137         while (!sn.empty())
138             sn.pop();
139
140         sm.push(Matrix());
141         sn.push(-1);
142
143         while (!sm.empty() && !sn.empty())
144         {
145             scanf("%s",str);
146
147             if (!strcmp(str,"translate"))
148             {
149                 scanf("%lf%lf%lf",&v.x,&v.y,&v.z);
150                 tmp=get_trans(v);
151                 sm.top()=sm.top()*tmp;
152             }
153             else if (!strcmp(str,"scale"))
154             {
155                 scanf("%lf%lf%lf",&v.x,&v.y,&v.z);
156                 tmp=get_scale(v);
157                 sm.top()=sm.top()*tmp;
158             }
159             else if (!strcmp(str,"rotate"))
160             {
161                 scanf("%lf%lf%lf%lf",&v.x,&v.y,&v.z,&ang);
162                 tmp=get_rotate(v,ang);
163                 sm.top()=sm.top()*tmp;
164             }
165             else if (!strcmp(str,"repeat"))
166             {
167                 scanf("%d",&m);
168                 sn.push(m);
169                 sm.push(Matrix());
170             }
171             else if (!strcmp(str,"end"))
172             {
173                 m=sn.top();
174                 sn.pop();
175                 tmp=sm.top();
176                 sm.pop();
177                 if (m!=-1)
178                     cur=tmp;
179                 else
180                 {
181                     tmp=pow(tmp,m);
182                     sm.top()=sm.top()*tmp;
183                 }
184             }
185         }
186     }

```

```

186     }
187
188     for (i=0;i<n;i++)
189     {
190         scanf ("%lf%lf%lf",&p.x,&p.y,&p.z);
191         p=p*cur;
192         printf("%.2f %.2f %.2f\n",p.x+eps,p.y+eps,p.z+eps);
193     }
194     printf("\n");
195 }
196
197 return 0;
198 }

```

8 其他

8.1 矩阵乘

注意初始化sz的大小

```
1 | const int N = 60;
2 | typedef long long LOL;
3 | LOL mod=1000000007ll;
4 | LOL c[N][N],a[N][N],b[N][N],g[N][N];
5 | int sz;
6 | void matcopy(LOL a[N][N],LOL b[N][N])
7 | {
8 |     for(int i=1;i<=sz;i++){
9 |         for(int j=1;j<=sz;j++){
10 |             a[i][j]=b[i][j];
11 |         }
12 |     }
13 | }
14 | void matmul(LOL a[N][N],LOL b[N][N])
15 | {
16 |     memset(c,0,sizeof(c));
17 |     for(int i=1;i<=sz;i++){
18 |         for(int j=1;j<=sz;j++){
19 |             if(a[i][j]){
20 |                 for(int k=1;k<=sz;k++){
21 |                     c[i][k]+=a[i][j]*b[j][k];
22 |                     if(c[i][k]>mod) c[i][k]%=mod;
23 |                 }
24 |             }
25 |         }
26 |     }
27 |     matcopy(a,c);
28 | }
29 | void matpow(LOL a[N][N],LOL x)
30 | {
31 |     memset(b,0,sizeof(b));
32 |     for(int i=1;i<=sz;i++){
33 |         b[i][i]=1;
34 |     }
35 |     while(x!=0)
36 |     {
37 |         if(x%2) matmul(b,a);
38 |         matmul(a,a);
39 |         x/=2;
40 |     }
41 |     matcopy(a,b);
42 | }
```

8.2 平面最近点对

测试报告: poj3714 hdu1007

```
1 #include<iostream>
2 #include<cstdio>
3 #include<cmath>
4 #include<algorithm>
5 #define oo 1e30
6 #define eps 1e-8
7 #define N 100005
8 using namespace std;
9 int stack[N];
10
11 struct Point
12 {
13     double x,y;
14 };
15
16 Point pt[N];
17
18 double dis(Point a,Point b)
19 {
20     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
21 }
22
23 bool cmp_x(Point a,Point b)
24 {
25     if (fabs(a.x-b.x)<eps)
26         return a.y<b.y;
27     else
28         return a.x<b.x;
29 }
30
31 bool cmp_y(int a,int b)
32 {
33     return pt[a].y<pt[b].y;
34 }
35
36 double find_minimum_points(Point pt[],int l,int r)
37 {
38     int i,j,top,mid;
39     double d1,d2,res;
40
41     if (r-l==1)
42         return dis(pt[l],pt[r]);
43     else if (r-l==2)
44         return min(dis(pt[l],pt[l+1]),min(dis(pt[l],pt[r]),dis(pt[l+1],pt[r]
45         ])));
46     else
47     {
48         mid=(l+r)/2;
49         d1=find_minimum_points(pt,l,mid);
50         d2=find_minimum_points(pt,mid+1,r);
51         res=min(d1,d2);
52
53         top=0;
54         for (i=mid;i>=l&&fabs(pt[mid+1].x-pt[i].x)<=res;i--)
55             stack[top++]=i;
56         for (i=mid+1;i<=r&&fabs(pt[mid].x-pt[i].x)<=res;i++)
57             stack[top++]=i;
58
59         if (top>0)
60             sort(stack,stack+top,cmp_y);
```



```

60
61     for (i=0;i<top;i++)
62         for (j=i+1;pt[stack[j]].y-pt[stack[i]].y<=res&& j<top;j++)
63             res=min(res,dis(pt[stack[i]],pt[stack[j]]));
64
65     return res;
66 }
67 }
68
69 double minimum_distance_pointset(Point pt[],int n)
70 {
71     int i;
72
73     sort(pt,pt+n,cmp_x);
74     for (i=1;i<n;i++)
75
76         if (fabs(pt[i].x-pt[i-1].x)<eps&&fabs(pt[i].y-pt[i-1].y)<eps)
77             return 0.0;
78
79     return find_minimum_points(pt,0,n-1);
80 }
81
82 int main()
83 {
84     int i,n;
85     double ans;
86
87     while (cin>>n,n)
88     {
89
90         for (i=0;i<n;i++)
91             scanf("%lf%lf",&pt[i].x,&pt[i].y);
92
93         ans=minimum_distance_pointset(pt,n);
94         printf("%.2f\n",ans/2);
95     }
96
97     return 0;
98 }

```

8.3 读入外挂

```
1 inline int ScanInt(void) {
2     int r = 0, c, d;
3     while (!isdigit(c = getchar()) && c != '-');
4     if (c != '-') r = c - '0'; d = c;
5     while (isdigit(c = getchar())) r = r * 10 + c - '0';
6     return d=='-'?-r:r;
7 }
8
9 inline Int64 ScanInt(void) {
10    Int64 r = 0, c, d;
11    while (!isdigit(c = getchar()) && c != '-');
12    if (c != '-') r = c - '0'; d = c;
13    while (isdigit(c = getchar())) r = r * 10ll + c - '0';
14    return d=='-'?-r:r;
15 }
```

8.4 JAVA分数类

```
1 import java.io.*;
2 import java.util.*;
3 import java.math.*;
4
5 class BigFraction {
6
7     BigFraction() {
8         numerator = BigInteger.ZERO;
9         Denominator = BigInteger.ONE;
10    }
11
12    BigFraction(BigInteger _numerator, BigInteger _Denominator) {
13        numerator = _numerator;
14        Denominator = _Denominator;
15    }
16
17    public BigFraction add(BigFraction bf) {
18        BigInteger n = numerator.multiply(bf.Denominator).add(Denominator.
19            multiply(bf.numerator));
20        BigInteger d = Denominator.multiply(bf.Denominator);
21
22        BigFraction ret = new BigFraction(n, d);
23        ret.simplify();
24
25        return ret;
26    }
27
28    public BigFraction subtract(BigFraction bf) {
29        BigInteger n = numerator.multiply(bf.Denominator).subtract(
30            Denominator.multiply(bf.numerator));
31        BigInteger d = Denominator.multiply(bf.Denominator);
32
33        BigFraction ret = new BigFraction(n, d);
34        ret.simplify();
35
36        return ret;
37    }
38
39    public BigFraction multiply(BigFraction bf) {
40        BigInteger n = numerator.multiply(bf.numerator);
41        BigInteger d = Denominator.multiply(bf.Denominator);
42
43        BigFraction ret = new BigFraction(n, d);
44        ret.simplify();
45
46        return ret;
47    }
48
49    public BigFraction divide(BigFraction bf) {
50        BigInteger n = numerator.multiply(bf.Denominator);
51        BigInteger d = Denominator.multiply(bf.numerator);
52
53        BigFraction ret = new BigFraction(n, d);
54        ret.simplify();
55
56        return ret;
57    }
58
59    public int compareTo(BigFraction bf) {
60        BigInteger ret = numerator.multiply(bf.Denominator).subtract(
61            Denominator.multiply(bf.numerator));
```

```

59         return ret.compareTo(BigInteger.ZERO);
60     }
61
62
63     public BigFraction abs() {
64         BigFraction ret = new BigFraction(numerator.abs(), Denominator.abs()
65         );
66         return ret;
67     }
68
69     public BigFraction negate() {
70         numerator = numerator.negate();
71         return this;
72     }
73
74     public boolean isInteger() {
75         return Denominator.equals(BigInteger.ONE);
76     }
77
78     // for simplify numerator and Denominator
79     private void simplify() {
80         BigInteger g = numerator.gcd(Denominator);
81         numerator = numerator.divide(g);
82         Denominator = Denominator.divide(g);
83         if (Denominator.compareTo(BigInteger.ZERO) < 0) {
84             numerator = numerator.negate();
85             Denominator = Denominator.negate();
86         }
87     }
88
89     public BigInteger numerator;
90     public BigInteger Denominator;
91     public static BigFraction ZERO = new BigFraction(BigInteger.ZERO,
92     BigInteger.ONE);
93     public static BigFraction ONE = new BigFraction(BigInteger.ONE,
94     BigInteger.ONE);
95 }
96
97 public class Main
98 {
99     static BigFraction c[][]=new BigFraction[120][120];
100     static BigFraction b[][]=new BigFraction[120][120];
101     public static void main(String args[])
102     {
103         for(int i=0;i<=101;i++){
104             for(int j=0;j<=101;j++){
105                 c[i][j]=BigFraction.ZERO;
106                 b[i][j]=BigFraction.ZERO;
107             }
108         }
109         for(int i=0;i<=101;i++){
110             c[i][0]=BigFraction.ONE;
111         }
112         for(int i=1;i<=101;i++){
113             for(int j=1;j<=i;j++){
114                 c[i][j]=c[i-1][j-1].add(c[i-1][j]);
115             }
116         }
117         b[0][1]=BigFraction.ONE;
118         for(int i=1;i<=100;i++){
119             for(int j=1;j<=i+1;j++){
120                 b[i][j]=c[i+1][j];
121             }
122             for(int j=0;j<i;j++){

```

```

119         for(int k=1;k<=j+1;k++){
120             b[i][k]=b[i][k].subtract(c[i+1][i+1-j].multiply(b[j][k])
121             );
122         }
123     }
124     for(int j=1;j<=i+1;j++){
125         b[i][j]=b[i][j].divide(c[i+1][1]);
126     }
127     Scanner cin=new Scanner(new BufferedInputStream(System.in));
128     BigInteger n;
129     int m;
130     while(cin.hasNext())
131     {
132         n=cin.nextBigInteger();
133         m=cin.nextInt();
134         BigFraction ans=BigFraction.ZERO;
135         BigInteger tmp=n;
136         for(int i=1;i<=m+1;i++){
137             BigFraction tx = BigFraction.ONE;
138             tx.numerator=tmp;
139             tx.Denominator=BigInteger.ONE;
140             ans=ans.add(tx.multiply(b[m][i]));
141             tmp=tmp.multiply(n);
142         }
143         System.out.println(ans.numerator);
144     }
145 }
146
147 }

```

8.5 魔方

```
1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  #include<vector>
5  #include<string>
6  #include<queue>
7  #include<cmath>
8  using namespace std;
9
10 int rubik[55];
11
12 //魔方初始化。根据题目要求决定是每个面的每个格子一个编号，还是每个面的格子标同一个编号
13 void init() {for (int i=1;i<=54;i++) rubik[i]=i;}
14
15 //对每个面进行顺时针旋转变度90
16 void rotate(int x1,int x2,int x3,int x4,int x5,int x6,int x7,int x8,int x9)
17 {
18     int a[55];
19
20     memcpy(a,rubik,sizeof(a));
21     rubik[x1]=a[x7]; rubik[x2]=a[x4]; rubik[x3]=a[x1];
22     rubik[x4]=a[x8]; rubik[x5]=a[x5]; rubik[x6]=a[x2];
23     rubik[x7]=a[x9]; rubik[x8]=a[x6]; rubik[x9]=a[x3];
24 }
25
26 //侧边顺时针选装度90
27 void trans(int x1,int x2,int x3,int x4,int x5,int x6,int x7,int x8,int x9,
28            int x10,int x11,int x12)
29 {
30     int a[55];
31
32     memcpy(a,rubik,sizeof(a));
33     rubik[x1]=a[x10]; rubik[x2]=a[x11]; rubik[x3]=a[x12];
34     rubik[x4]=a[x1]; rubik[x5]=a[x2]; rubik[x6]=a[x3];
35     rubik[x7]=a[x4]; rubik[x8]=a[x5]; rubik[x9]=a[x6];
36     rubik[x10]=a[x7]; rubik[x11]=a[x8]; rubik[x12]=a[x9];
37 }
38 //从上面看去，顺时针旋转第一层
39 void turn_U()
40 {
41     rotate(1,2,3,4,5,6,7,8,9);
42     trans(30,29,28,21,20,19,12,11,10,39,38,37);
43 }
44 //从上面看去，顺时针旋转第二层
45 void turn_X()
46 {
47     trans(33,32,31,24,23,22,15,14,13,42,41,40);
48 }
49 //从上面看去，顺时针旋转第三层
50 void turn_D()
51 {
52     rotate(48,47,46,51,50,49,54,53,52);
53     trans(36,35,34,27,26,25,18,17,16,45,44,43);
54 }
55 //从右面看去，顺治针旋转第一层
56 void turn_R()
57 {
58     rotate(19,20,21,22,23,24,25,26,27);
59     trans(9,6,3,28,31,34,46,49,52,18,15,12);
```

```

60 }
61 //从右面看去, 顺治针旋转第二层
62 void turn_Y()
63 {
64     trans(8,5,2,29,32,35,47,50,53,17,14,11);
65 }
66 //从右面看去, 顺治针旋转第三层
67 void turn_L()
68 {
69     rotate(39,38,37,42,41,40,45,44,43);
70     trans(7,4,1,30,33,36,48,51,54,16,13,10);
71 }
72 //从前面看去, 顺治针旋转第一层
73 void turn_F()
74 {
75     rotate(10,11,12,13,14,15,16,17,18);
76     trans(7,8,9,19,22,25,52,53,54,45,42,39);
77 }
78 //从前面看去, 顺治针旋转第二层
79 void turn_Z()
80 {
81     trans(4,5,6,20,23,26,49,50,51,44,41,38);
82 }
83 //从前面看去, 顺治针旋转第三层
84 void turn_B()
85 {
86     rotate(30,29,28,33,32,31,36,35,34);
87     trans(1,2,3,21,24,27,46,47,48,43,40,37);
88 }
89
90 //hdu4397: 询问操作后是否与初始状态完全相同 (每个面的每个格子都完全一样)
91 int main()
92 {
93     int i,j,ans,ys;
94     char str[1000];
95
96     ys=0;
97     freopen("data.in","r",stdin);
98     while (scanf("%s",str)!=EOF)
99     {
100         init();
101         for (i=0;str[i];i++)
102             switch (str[i])
103             {
104                 case 'U':
105                     turn_U();
106                     break;
107                 case 'u':
108                     turn_U();turn_U();turn_U();
109                     break;
110                 case 'X':
111                     turn_X();
112                     break;
113                 case 'x':
114                     turn_X();turn_X();turn_X();
115                     break;
116                 case 'D':
117                     turn_D();turn_D();turn_D();
118                     break;
119                 case 'd':
120                     turn_D();
121                     break;

```

```

122         case 'R':
123             turn_R();
124         break;
125         case 'r':
126             turn_R();turn_R();turn_R();
127         break;
128         case 'Y':
129             turn_Y();
130         break;
131         case 'y':
132             turn_Y();turn_Y();turn_Y();
133         break;
134         case 'L':
135             turn_L();turn_L();turn_L();
136         break;
137         case 'l':
138             turn_L();
139         break;
140         case 'F':
141             turn_F();
142         break;
143         case 'f':
144             turn_F();turn_F();turn_F();
145         break;
146         case 'Z':
147             turn_Z();turn_Z();turn_Z();
148         break;
149         case 'z':
150             turn_Z();
151         break;
152         case 'B':
153             turn_B();turn_B();turn_B();
154         break;
155         case 'b':
156             turn_B();
157         break;
158     }
159
160     ans=1;
161     for (i=1;i<=54;i++)
162         if (rubik[i]!=i)
163             {
164                 ans=0;
165                 break;
166             }
167
168     ys++;
169     if (ys>1)
170         printf("\n");
171
172     if (ans)
173         printf("Yes\n");
174     else
175         printf("No\n");
176 }
177
178 return 0;
179 }

```


8.6 Hashmap

GCC中的hash_map定义在<ext/hash_map>文件，namespace __gnu_cxx中。要定义一个hash_map<int, int>非常简单：

```
1 #include <ext/hash_map>
2 using namespace __gnu_cxx;
3 hash_map<int, int> hm;
```

在使用map时，如果我们想要改变元素顺序，或以自定义的struct/class作为key的时候，可以设定map第三个模板参数（默认是less<Key>，即operator<）。对于hash_map，我们需要设定其第三个(hash<Key>)和第四个模板参数(equal_to<Key>, operator==)。

```
1 typedef long long my_type;
2 typedef int any_type;
3 struct my_hash {
4     size_t operator()(const my_type& key) const {
5         return (key >> 32) ^ key;
6     }
7 };
8 struct my_equal_to {
9     bool operator()(const my_type& lhs, const my_type& rhs) const {
10         return lhs == rhs;
11     }
12 };
13 hash_map<my_type, any_type, my_hash, my_equal_to> my_hash_map;
```