Code Library

Himemiya Nanao @ Perfect Freeze

September 13, 2013

Contents

1 Data Structure

1.1 atlantis

```cpp
#include<cstdio>
#include<algorithm>
#include<map>

#define MAXX 111
#define inf 333
#define MAX inf*5

int mid[MAX],cnt[MAX];
double len[MAX];

int n,i,cas;
double x1,x2,y1,y2;
double ans;
std::map<double,int>map;
std::map<double,int>::iterator it;
double rmap[inf];

void make(int id,int l,int r)
{
    mid[id]=(l+r)>>1;
    if(l!=r)
    {
        make(id<<1,l,mid[id]);
        make(id<<1|1,mid[id]+1,r);
    }
}

void update(int id,int ll,int rr,int l,int r,int val)
{
    if(ll==l && rr==r)
    {
        cnt[id]+=val;
        if(cnt[id])
            len[id]=rmap[r]-rmap[l-1];
        else
            if(l!=r)
                len[id]=len[id<<1]+len[id<<1|1];
            else
                len[id]=0;
        return;
    }
    if(mid[id]>=r)
        update(id<<1,ll,mid[id],l,r,val);
    else
        if(mid[id]<l)
            update(id<<1|1,mid[id]+1,rr,l,r,val);
        else
```

```
49          {
50              update(id<<1,ll,mid[id],l,mid[id],val);
51              update(id<<1|1,mid[id]+1,rr,mid[id]+1,r,val);
52          }
53      if(!cnt[id])
54          len[id]=len[id<<1]+len[id<<1|1];
55  }
56
57  struct node
58  {
59      double l,r,h;
60      char f;
61      inline bool operator<(const node &a)const
62      {
63          return h<a.h;
64      }
65      inline void print()
66      {
67          printf("%lf␣%lf␣%lf␣%d\n",l,r,h,f);
68      }
69  }ln[inf];
70
71  int main()
72  {
73      make(1,1,inf);
74      while(scanf("%d",&n),n)
75      {
76          n<<=1;
77          map.clear();
78          for(i=0;i<n;++i)
79          {
80              scanf("%lf%lf%lf%lf",&x1,&y1,&x2,&y2);
81              if(x1>x2)
82                  std::swap(x1,x2);
83              if(y1>y2)
84                  std::swap(y1,y2);
85              ln[i].l=x1;
86              ln[i].r=x2;
87              ln[i].h=y1;
88              ln[i].f=1;
89              ln[++i].l=x1;
90              ln[i].r=x2;
91              ln[i].h=y2;
92              ln[i].f=-1;
93              map[x1]=1;
94              map[x2]=1;
95          }
96          i=1;
97          for(it=map.begin();it!=map.end();++it,++i)
98          {
99              it->second=i;
```

```
100             rmap[i]=it->first;
101         }
102         std::sort(ln,ln+n);
103         ans=0;
104         update(1,1,inf,map[ln[0].l]+1,map[ln[0].r],ln[0].f);
105         for(i=1;i<n;++i)
106         {
107             ans+=len[1]*(ln[i].h-ln[i-1].h);
108             update(1,1,inf,map[ln[i].l]+1,map[ln[i].r],ln[i].f);
109         }
110         printf("Test␣case␣#%d\nTotal␣explored␣area:␣%.2lf\n\n",++
               cas,ans);
111     }
112     return 0;
113 }
```

1.2 binary indexed tree

```
 1 int tree[MAXX];
 2
 3 inline int lowbit(const int &a)
 4 {
 5     return a&-a;
 6 }
 7
 8 inline void update(int pos,const int &val)
 9 {
10     while(pos<MAXX)
11     {
12         tree[pos]+=val;
13         pos+=lowbit(pos);
14     }
15 }
16
17 inline int read(int pos)
18 {
19     int re(0);
20     while(pos>0)
21     {
22         re+=tree[pos];
23         pos-=lowbit(pos);
24     }
25     return re;
26 }
27
28 int find_Kth(int k)
29 {
30     int now=0;
31     for (char i=20;i>=0;--i)
32     {
33         now|=(1<<i);
34         if (now>MAXX || tree[now]>=k)
```

```
35              now^=(1<<i);
36          else k-=tree[now];
37      }
38      return now+1;
39  }
```

1.3 COT

```
 1  #include<cstdio>
 2  #include<algorithm>
 3
 4  #define MAXX 100111
 5  #define MAX (MAXX*23)
 6  #define N 18
 7
 8  int sz[MAX],lson[MAX],rson[MAX],cnt;
 9  int head[MAXX];
10  int pre[MAXX][N];
11  int map[MAXX],m;
12
13  int edge[MAXX],nxt[MAXX<<1],to[MAXX<<1];
14  int n,i,j,k,q,l,r,mid;
15  int num[MAXX],dg[MAXX];
16
17  int make(int l,int r)
18  {
19      if(l==r)
20          return ++cnt;
21      int id(++cnt),mid((l+r)>>1);
22      lson[id]=make(l,mid);
23      rson[id]=make(mid+1,r);
24      return id;
25  }
26
27  inline int update(int id,int pos)
28  {
29      int re(++cnt);
30      l=1;
31      r=m;
32      int nid(re);
33      sz[nid]=sz[id]+1;
34      while(l<r)
35      {
36          mid=(l+r)>>1;
37          if(pos<=mid)
38          {
39              lson[nid]=++cnt;
40              rson[nid]=rson[id];
41              nid=lson[nid];
42              id=lson[id];
43              r=mid;
44          }
```

```
45          else
46          {
47              lson[nid]=lson[id];
48              rson[nid]=++cnt;
49              nid=rson[nid];
50              id=rson[id];
51              l=mid+1;
52          }
53          sz[nid]=sz[id]+1;
54      }
55      return re;
56  }
57
58  void rr(int now,int fa)
59  {
60      dg[now]=dg[fa]+1;
61      head[now]=update(head[fa],num[now]);
62      for(int i(edge[now]);i;i=nxt[i])
63          if(to[i]!=fa)
64          {
65              j=1;
66              for(pre[to[i]][0]=now;j<N;++j)
67                  pre[to[i]][j]=pre[pre[to[i]][j-1]][j-1];
68              rr(to[i],now);
69          }
70  }
71
72  inline int query(int a,int b,int n,int k)
73  {
74      static int tmp,t;
75      l=1;
76      r=m;
77      a=head[a];
78      b=head[b];
79      t=num[n];
80      n=head[n];
81      while(l<r)
82      {
83          mid=(l+r)>>1;
84          tmp=sz[lson[a]]+sz[lson[b]]-2*sz[lson[n]]+(l<=t && t<=mid);
85          if(tmp>=k)
86          {
87              a=lson[a];
88              b=lson[b];
89              n=lson[n];
90              r=mid;
91          }
92          else
93          {
94              k-=tmp;
95              a=rson[a];
```

```
 96              b=rson[b];
 97              n=rson[n];
 98              l=mid+1;
 99          }
100      }
101      return l;
102  }
103
104  inline int lca(int a,int b)
105  {
106      static int i,j;
107      j=0;
108      if(dg[a]<dg[b])
109          std::swap(a,b);
110      for(i=dg[a]-dg[b];i;i>>=1,++j)
111          if(i&1)
112              a=pre[a][j];
113      if(a==b)
114          return a;
115      for(i=N-1;i>=0;--i)
116          if(pre[a][i]!=pre[b][i])
117          {
118              a=pre[a][i];
119              b=pre[b][i];
120          }
121      return pre[a][0];
122  }
123
124  int main()
125  {
126      scanf("%d␣%d",&n,&q);
127      for(i=1;i<=n;++i)
128      {
129          scanf("%d",num+i);
130          map[i]=num[i];
131      }
132      std::sort(map+1,map+n+1);
133      m=std::unique(map+1,map+n+1)-map-1;
134      for(i=1;i<=n;++i)
135          num[i]=std::lower_bound(map+1,map+m+1,num[i])-map;
136      for(i=1;i<n;++i)
137      {
138          scanf("%d␣%d",&j,&k);
139          nxt[++cnt]=edge[j];
140          edge[j]=cnt;
141          to[cnt]=k;
142
143          nxt[++cnt]=edge[k];
144          edge[k]=cnt;
145          to[cnt]=j;
146      }
```

```
147     cnt=0;
148     head[0]=make(1,m);
149     rr(1,0);
150     while(q--)
151     {
152         scanf("%d␣%d␣%d",&i,&j,&k);
153         printf("%d\n",map[query(i,j,lca(i,j),k)]);
154     }
155     return 0;
156 }
```

1.4 hose

```
1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  #include<cmath>
5
6  #define MAXX 50111
7
8  struct Q
9  {
10     int l,r,s,w;
11     bool operator<(const Q &i)const
12     {
13         return w==i.w?r<i.r:w<i.w;
14     }
15 }a[MAXX];
16
17 int c[MAXX];
18 long long col[MAXX],sz[MAXX],ans[MAXX];
19 int n,m,cnt,len;
20
21 long long gcd(long long a,long long b)
22 {
23     return a?gcd(b%a,a):b;
24 }
25
26 int i,j,k,now;
27 long long all,num;
28
29 int main()
30 {
31     scanf("%d␣%d",&n,&m);
32     for(i=1;i<=n;++i)
33         scanf("%d",c+i);
34     len=sqrt(m);
35     for(i=1;i<=m;++i)
36     {
37         scanf("%d␣%d",&a[i].l,&a[i].r);
38         if(a[i].l>a[i].r)
39             std::swap(a[i].l,a[i].r);
```

```
        sz[i]=a[i].r-a[i].l+1;
        a[i].w=a[i].l/len+1;
        a[i].s=i;
    }
    std::sort(a+1,a+m+1);
    i=1;
    while(i<=m)
    {
        now=a[i].w;
        memset(col,0,sizeof col);
        for(j=a[i].l;j<=a[i].r;++j)
            ans[a[i].s]+=2*(col[c[j]]++);
        for(++i;a[i].w==now;++i)
        {
            ans[a[i].s]=ans[a[i-1].s];
            for(j=a[i-1].r+1;j<=a[i].r;++j)
                ans[a[i].s]+=2*(col[c[j]]++);
            if(a[i-1].l<a[i].l)
                for(j=a[i-1].l;j<a[i].l;++j)
                    ans[a[i].s]-=2*(--col[c[j]]);
            else
                for(j=a[i].l;j<a[i-1].l;++j)
                    ans[a[i].s]+=2*(col[c[j]]++);
        }
    }
    for(i=1;i<=m;++i)
    {
        if(sz[i]==1)
            all=1ll;
        else
            all=sz[i]*(sz[i]-1);
        num=gcd(ans[i],all);
        printf("%lld/%lld\n",ans[i]/num,all/num);
    }
    return 0;
}
```

1.5 Leftist tree

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

#define MAXX 100111

int val[MAXX],l[MAXX],r[MAXX],d[MAXX];

int set[MAXX];

int merge(int a,int b)
{
    if(!a)
        return b;
```

```
14      if(!b)
15          return a;
16      if(val[a]<val[b]) // max-heap
17          std::swap(a,b);
18      r[a]=merge(r[a],b);
19      if(d[l[a]]<d[r[a]])
20          std::swap(l[a],r[a]);
21      d[a]=d[r[a]]+1;
22      set[l[a]]=set[r[a]]=a; // set a as father of its sons
23      return a;
24  }
25
26  inline int find(int &a)
27  {
28      while(set[a]) //brute-force to get the index of root
29          a=set[a];
30      return a;
31  }
32
33  inline void reset(int i)
34  {
35      l[i]=r[i]=d[i]=set[i]=0;
36  }
37
38  int n,i,j,k;
39
40  int main()
41  {
42      while(scanf("%d",&n)!=EOF)
43      {
44          for(i=1;i<=n;++i)
45          {
46              scanf("%d",val+i);
47              reset(i);
48          }
49          scanf("%d",&n);
50          while(n--)
51          {
52              scanf("%d␣%d",&i,&j);
53              if(find(i)==find(j))
54                  puts("-1");
55              else
56              {
57                  k=merge(l[i],r[i]);
58                  val[i]>>=1;
59                  reset(i);
60                  set[i=merge(i,k)]=0;
61
62                  k=merge(l[j],r[j]);
63                  val[j]>>=1;
64                  reset(j);
```

```
                set[j=merge(j,k)]=0;

                set[k=merge(i,j)]=0;
                printf("%d\n",val[k]);
            }
        }
    }
    return 0;
}
```

1.6 Network

```
//HLD……备忘……_(:3JZ)_
#include<cstdio>
#include<algorithm>
#include<cstdlib>

#define MAXX 80111
#define MAXE (MAXX<<1)
#define N 18

int edge[MAXX],nxt[MAXE],to[MAXE],cnt;
int fa[MAXX][N],dg[MAXX];

inline int lca(int a,int b)
{
    static int i,j;
    j=0;
    if(dg[a]<dg[b])
        std::swap(a,b);
    for(i=dg[a]-dg[b];i;i>>=1,++j)
        if(i&1)
            a=fa[a][j];
    if(a==b)
        return a;
    for(i=N-1;i>=0;--i)
        if(fa[a][i]!=fa[b][i])
        {
            a=fa[a][i];
            b=fa[b][i];
        }
    return fa[a][0];
}

inline void add(int a,int b)
{
    nxt[++cnt]=edge[a];
    edge[a]=cnt;
    to[cnt]=b;
}

int sz[MAXX],pre[MAXX],next[MAXX];
```

```
41
42  void rr(int now)
43  {
44      sz[now]=1;
45      int max,id;
46      max=0;
47      for(int i(edge[now]);i;i=nxt[i])
48          if(to[i]!=fa[now][0])
49          {
50              fa[to[i]][0]=now;
51              dg[to[i]]=dg[now]+1;
52              rr(to[i]);
53              sz[now]+=sz[to[i]];
54              if(sz[to[i]]>max)
55              {
56                  max=sz[to[i]];
57                  id=to[i];
58              }
59          }
60      if(max)
61      {
62          next[now]=id;
63          pre[id]=now;
64      }
65  }
66
67  #define MAXT (MAXX*N*5)
68
69  namespace Treap
70  {
71      int cnt;
72      int son[MAXT][2],key[MAXT],val[MAXT],sz[MAXT];
73
74      inline void init()
75      {
76          key[0]=RAND_MAX;
77          val[0]=0xc0c0c0c0;
78          cnt=0;
79      }
80
81      inline void up(int id)
82      {
83          sz[id]=sz[son[id][0]]+sz[son[id][1]]+1;
84      }
85      inline void rot(int &id,int tp)
86      {
87          static int k;
88          k=son[id][tp];
89          son[id][tp]=son[k][tp^1];
90          son[k][tp^1]=id;
91          up(id);
```

```
92          up(k);
93          id=k;
94      }
95      void insert(int &id,int v)
96      {
97          if(id)
98          {
99              int k(v>=val[id]);
100             insert(son[id][k],v);
101             if(key[son[id][k]]<key[id])
102                 rot(id,k);
103             else
104                 up(id);
105             return;
106         }
107         id=++cnt;
108         key[id]=rand()-1;
109         val[id]=v;
110         sz[id]=1;
111         son[id][0]=son[id][1]=0;
112     }
113     void del(int &id,int v)
114     {
115         if(!id)
116             return;
117         if(val[id]==v)
118         {
119             int k(key[son[id][1]]<key[son[id][0]]);
120             if(!son[id][k])
121             {
122                 id=0;
123                 return;
124             }
125             rot(id,k);
126             del(son[id][k^1],v);
127         }
128         else
129             del(son[id][v>val[id]],v);
130         up(id);
131     }
132     int rank(int id,int v)
133     {
134         if(!id)
135             return 0;
136         if(val[id]<=v)
137             return sz[son[id][0]]+1+rank(son[id][1],v);
138         return rank(son[id][0],v);
139     }
140     void print(int id)
141     {
142         if(!id)
```

```
143             return;
144         print(son[id][0]);
145         printf("%d␣",val[id]);
146         print(son[id][1]);
147     }
148 }
149 
150 int head[MAXX],root[MAXX],len[MAXX],pos[MAXX];
151 
152 #define MAX (MAXX*6)
153 #define mid (l+r>>1)
154 #define lc lson[id],l,mid
155 #define rc rson[id],mid+1,r
156 
157 int lson[MAX],rson[MAX];
158 int treap[MAX];
159 
160 void make(int &id,int l,int r,int *the)
161 {
162     id=++cnt;
163     static int k;
164     for(k=l;k<=r;++k)
165         Treap::insert(treap[id],the[k]);
166     if(l!=r)
167     {
168         make(lc,the);
169         make(rc,the);
170     }
171 }
172 
173 int query(int id,int l,int r,int a,int b,int q)
174 {
175     if(a<=l && r<=b)
176         return Treap::rank(treap[id],q);
177     int re(0);
178     if(a<=mid)
179         re=query(lc,a,b,q);
180     if(b>mid)
181         re+=query(rc,a,b,q);
182     return re;
183 }
184 
185 inline int query(int a,int b,int v)
186 {
187     static int re;
188     for(re=0;root[a]!=root[b];a=fa[root[a]][0])
189         re+=query(head[root[a]],1,len[root[a]],1,pos[a],v);
190     re+=query(head[root[a]],1,len[root[a]],pos[b],pos[a],v);
191     return re;
192 }
193 
```

```
194 inline void update(int id,int l,int r,int pos,int val,int n)
195 {
196     while(l<=r)
197     {
198         Treap::del(treap[id],val);
199         Treap::insert(treap[id],n);
200         if(l==r)
201             return;
202         if(pos<=mid)
203         {
204             id=lson[id];
205             r=mid;
206         }
207         else
208         {
209             id=rson[id];
210             l=mid+1;
211         }
212     }
213 }
214
215 int n,q,i,j,k;
216 int val[MAXX];
217
218 int main()
219 {
220     srand(1e9+7);
221     scanf("%d␣%d",&n,&q);
222     for(i=1;i<=n;++i)
223         scanf("%d",val+i);
224     for(k=1;k<n;++k)
225     {
226         scanf("%d␣%d",&i,&j);
227         add(i,j);
228         add(j,i);
229     }
230     rr(rand()%n+1);
231     for(j=1;j<N;++j)
232         for(i=1;i<=n;++i)
233             fa[i][j]=fa[fa[i][j-1]][j-1];
234
235     Treap::init();
236     cnt=0;
237     for(i=1;i<=n;++i)
238         if(!pre[i])
239         {
240             static int tmp[MAXX];
241             for(k=1,j=i;j;j=next[j],++k)
242             {
243                 pos[j]=k;
244                 root[j]=i;
```

```
245                 tmp[k]=val[j];
246             }
247             --k;
248             len[i]=k;
249             make(head[i],1,k,tmp);
250         }
251     while(q--)
252     {
253         scanf("%d",&k);
254         if(k)
255         {
256             static int a,b,c,d,l,r,ans,m;
257             scanf("%d %d",&a,&b);
258             c=lca(a,b);
259             if(dg[a]+dg[b]-2*dg[c]+1<k)
260             {
261                 puts("invalid request!");
262                 continue;
263             }
264             k=dg[a]+dg[b]-2*dg[c]+1-k+1;
265             if(dg[a]<dg[b])
266                 std::swap(a,b);
267             l=-1e9;
268             r=1e9;
269             if(b!=c)
270             {
271                 d=a;
272                 for(i=0,j=dg[a]-dg[c]-1;j;j>>=1,++i)
273                     if(j&1)
274                         d=fa[d][i];
275                 while(l<=r)
276                 {
277                     m=l+r>>1;
278                     if(query(a,d,m)+query(b,c,m)>=k)
279                     {
280                         ans=m;
281                         r=m-1;
282                     }
283                     else
284                         l=m+1;
285                 }
286             }
287             else
288             {
289                 while(l<=r)
290                 {
291                     m=l+r>>1;
292                     if(query(a,c,m)>=k)
293                     {
294                         ans=m;
295                         r=m-1;
```

```
                    }
                    else
                        l=m+1;
                }
            }
            printf("%d\n",ans);
        }
        else
        {
            scanf("%d␣%d",&i,&j);
            update(head[root[i]],1,len[root[i]],pos[i],val[i],j);
            val[i]=j;
        }
    }
    return 0;
}
```

1.7 OTOCI

```
//记得随手 down 啊……亲……
//debug 时记得优先检查 up/down/select
#include<cstdio>
#include<algorithm>

#define MAXX 30111
#define lson nxt[id][0]
#define rson nxt[id][1]

int nxt[MAXX][2],fa[MAXX],pre[MAXX],val[MAXX],sum[MAXX];
bool rev[MAXX];

inline void up(int id)
{
    static int i;
    sum[id]=val[id];
    for(i=0;i<2;++i)
        if(nxt[id][i])
            sum[id]+=sum[nxt[id][i]];
}

inline void rot(int id,int tp)
{
    static int k;
    k=pre[id];
    nxt[k][tp^1]=nxt[id][tp];
    if(nxt[id][tp])
        pre[nxt[id][tp]]=k;
    if(pre[k])
        nxt[pre[k]][k==nxt[pre[k]][1]]=id;
    pre[id]=pre[k];
    nxt[id][tp]=k;
    pre[k]=id;
```

```
34      up(k);
35      up(id);
36  }
37
38  inline void down(int id) //记得随手 down 啊……亲……
39  {
40      static int i;
41      if(rev[id])
42      {
43          rev[id]=false;
44          for(i=0;i<2;++i)
45              if(nxt[id][i])
46              {
47                  rev[nxt[id][i]]^=true;
48                  std::swap(nxt[nxt[id][i]][0],nxt[nxt[id][i]][1]);
49              }
50      }
51  }
52
53  inline void splay(int id)//记得随手 down 啊……亲……
54  {
55      down(id);
56      if(!pre[id])
57          return;
58      static int rt,k,st[MAXX];
59      for(rt=id,k=0;rt;rt=pre[rt])
60          st[k++]=rt;
61      rt=st[k-1];
62      while(k)
63          down(st[--k]);
64      for(std::swap(fa[id],fa[rt]);pre[id];rot(id,id==nxt[pre[id
           ]][0]));
65      /* another faster methond:
66      std::swap(fa[id],fa[rt]);
67      do
68      {
69          rt=pre[id];
70          if(pre[rt])
71          {
72              k=(nxt[pre[rt]][0]==rt);
73              if(nxt[rt][k]==id)
74                  rot(id,k^1);
75              else
76                  rot(rt,k);
77              rot(id,k);
78          }
79          else
80              rot(id,id==nxt[rt][0]);
81      }
82      while(pre[id]);
83      */
```

```
 84 }
 85
 86 inline int access(int id)
 87 {
 88     static int to;
 89     for(to=0;id;id=fa[id])
 90     {
 91         splay(id);
 92         if(rson)
 93         {
 94             pre[rson]=0;
 95             fa[rson]=id;
 96         }
 97         rson=to;
 98         if(to)
 99         {
100             pre[to]=id;
101             fa[to]=0;
102         }
103         up(to=id);
104     }
105     return to;
106 }
107
108 inline int getrt(int id)
109 {
110     access(id);
111     splay(id);
112     while(nxt[id][0])
113     {
114         id=nxt[id][0];
115         down(id);
116     }
117     return id;
118 }
119
120 inline void makert(int id)
121 {
122     access(id);
123     splay(id);
124     if(nxt[id][0])
125     {
126         rev[id]^=true;
127         std::swap(lson,rson);
128     }
129 }
130
131 int n,i,j,k,q;
132 char buf[11];
133
134 int main()
```

```
{
    scanf("%d",&n);
    for(i=1;i<=n;++i)
        scanf("%d",val+i);
    scanf("%d",&q);
    while(q--)
    {
        scanf("%s␣%d␣%d",buf,&i,&j);
        switch(buf[0])
        {
            case 'b':
                if(getrt(i)==getrt(j))
                    puts("no");
                else
                {
                    puts("yes");
                    makert(i);
                    fa[i]=j;
                }
                break;
            case 'p':
                access(i);
                splay(i);
                val[i]=j;
                up(i);
                break;
            case 'e':
                if(getrt(i)!=getrt(j))
                    puts("impossible");
                else
                {
                    makert(i);
                    access(j);
                    splay(j);
                    printf("%d\n",sum[j]);
                }
                break;
        }
    }
    return 0;
}
```

1.8 picture

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

#define MAXX 5555
#define MAX MAXX<<3
#define inf 10011

```

```
 9 int n,i;
10 int mid[MAX],cnt[MAX],len[MAX],seg[MAX];
11 bool rt[MAX],lf[MAX];
12
13 std::map<int,int>map;
14 std::map<int,int>::iterator it;
15 int rmap[inf];
16 long long sum;
17 int x1,x2,y1,y2,last;
18
19 void make(int id,int l,int r)
20 {
21     mid[id]=(l+r)>>1;
22     if(l!=r)
23     {
24         make(id<<1,l,mid[id]);
25         make(id<<1|1,mid[id]+1,r);
26     }
27 }
28
29 void update(int id,int ll,int rr,int l,int r,int val)
30 {
31     if(l==ll && rr==r)
32     {
33         cnt[id]+=val;
34         if(cnt[id])
35         {
36             rt[id]=lf[id]=true;
37             len[id]=rmap[r]-rmap[l-1];
38             seg[id]=1;
39         }
40         else
41             if(l!=r)
42             {
43                 len[id]=len[id<<1]+len[id<<1|1];
44                 seg[id]=seg[id<<1]+seg[id<<1|1];
45                 if(rt[id<<1] && lf[id<<1|1])
46                     --seg[id];
47                 rt[id]=rt[id<<1|1];
48                 lf[id]=lf[id<<1];
49             }
50             else
51             {
52                 len[id]=0;
53                 rt[id]=lf[id]=false;
54                 seg[id]=0;
55             }
56         return;
57     }
58     if(mid[id]>=r)
59         update(id<<1,ll,mid[id],l,r,val);
```

```
60      else
61          if(mid[id]<l)
62              update(id<<1|1,mid[id]+1,rr,l,r,val);
63          else
64          {
65              update(id<<1,ll,mid[id],l,mid[id],val);
66              update(id<<1|1,mid[id]+1,rr,mid[id]+1,r,val);
67          }
68      if(!cnt[id])
69      {
70          len[id]=len[id<<1]+len[id<<1|1];
71          seg[id]=seg[id<<1]+seg[id<<1|1];
72          if(rt[id<<1] && lf[id<<1|1])
73              --seg[id];
74          rt[id]=rt[id<<1|1];
75          lf[id]=lf[id<<1];
76      }
77  }
78
79  struct node
80  {
81      int l,r,h;
82      char val;
83      inline bool operator<(const node &a)const
84      {
85          return h==a.h?val<a.val:h<a.h;   // trick watch out. val<a.
                val? val>a.val?
86      }
87      inline void print()
88      {
89          printf("%d␣%d␣%d␣%d\n",l,r,h,val);
90      }
91  }ln[inf];
92
93  int main()
94  {
95      make(1,1,inf);
96      scanf("%d",&n);
97      n<<=1;
98      map.clear();
99      for(i=0;i<n;++i)
100     {
101         scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
102         ln[i].l=x1;
103         ln[i].r=x2;
104         ln[i].h=y1;
105         ln[i].val=1;
106         ln[++i].l=x1;
107         ln[i].r=x2;
108         ln[i].h=y2;
109         ln[i].val=-1;
```

```
110         map[x1]=1;
111         map[x2]=1;
112     }
113     i=1;
114     for(it=map.begin();it!=map.end();++it,++i)
115     {
116         it->second=i;
117         rmap[i]=it->first;
118     }
119     i=0;
120     std::sort(ln,ln+n);
121     update(1,1,inf,map[ln[0].l]+1,map[ln[0].r],ln[0].val);
122     sum+=len[1];
123     last=len[1];
124     for(i=1;i<n;++i)
125     {
126         sum+=2*seg[1]*(ln[i].h-ln[i-1].h);
127         update(1,1,inf,map[ln[i].l]+1,map[ln[i].r],ln[i].val);
128         sum+=abs(len[1]-last);
129         last=len[1];
130     }
131     printf("%lld\n",sum);
132     return 0;
133 }
```

1.9 Size Blanced Tree

```
1  template<class Tp>class sbt
2  {
3      public:
4          inline void init()
5          {
6              rt=cnt=l[0]=r[0]=sz[0]=0;
7          }
8          inline void ins(const Tp &a)
9          {
10             ins(rt,a);
11         }
12         inline void del(const Tp &a)
13         {
14             del(rt,a);
15         }
16         inline bool find(const Tp &a)
17         {
18             return find(rt,a);
19         }
20         inline Tp pred(const Tp &a)
21         {
22             return pred(rt,a);
23         }
24         inline Tp succ(const Tp &a)
25         {
```

```
26              return succ(rt,a);
27          }
28          inline bool empty()
29          {
30              return !sz[rt];
31          }
32          inline Tp min()
33          {
34              return min(rt);
35          }
36          inline Tp max()
37          {
38              return max(rt);
39          }
40          inline void delsmall(const Tp &a)
41          {
42              dels(rt,a);
43          }
44          inline int rank(const Tp &a)
45          {
46              return rank(rt,a);
47          }
48          inline Tp sel(const int &a)
49          {
50              return sel(rt,a);
51          }
52          inline Tp delsel(int a)
53          {
54              return delsel(rt,a);
55          }
56      private:
57          int cnt,rt,l[MAXX],r[MAXX],sz[MAXX];
58          Tp val[MAXX];
59          inline void rro(int &pos)
60          {
61              int k(l[pos]);
62              l[pos]=r[k];
63              r[k]=pos;
64              sz[k]=sz[pos];
65              sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
66              pos=k;
67          }
68          inline void lro(int &pos)
69          {
70              int k(r[pos]);
71              r[pos]=l[k];
72              l[k]=pos;
73              sz[k]=sz[pos];
74              sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
75              pos=k;
76          }
```

```
77          inline void mt(int &pos,bool flag)
78          {
79              if(!pos)
80                  return;
81              if(flag)
82                  if(sz[r[r[pos]]]>sz[l[pos]])
83                      lro(pos);
84                  else
85                      if(sz[l[r[pos]]]>sz[l[pos]])
86                      {
87                          rro(r[pos]);
88                          lro(pos);
89                      }
90                      else
91                          return;
92              else
93                  if(sz[l[l[pos]]]>sz[r[pos]])
94                      rro(pos);
95                  else
96                      if(sz[r[l[pos]]]>sz[r[pos]])
97                      {
98                          lro(l[pos]);
99                          rro(pos);
100                     }
101                     else
102                         return;
103             mt(l[pos],false);
104             mt(r[pos],true);
105             mt(pos,false);
106             mt(pos,true);
107         }
108         void ins(int &pos,const Tp &a)
109         {
110             if(pos)
111             {
112                 ++sz[pos];
113                 if(a<val[pos])
114                     ins(l[pos],a);
115                 else
116                     ins(r[pos],a);
117                 mt(pos,a>=val[pos]);
118                 return;
119             }
120             pos=++cnt;
121             l[pos]=r[pos]=0;
122             val[pos]=a;
123             sz[pos]=1;
124         }
125         Tp del(int &pos,const Tp &a)
126         {
127             --sz[pos];
```

```
128             if(val[pos]==a || (a<val[pos] && !l[pos]) || (a>val[pos
                    ] && !r[pos]))
129             {
130                 Tp ret(val[pos]);
131                 if(!l[pos] || !r[pos])
132                     pos=l[pos]+r[pos];
133                 else
134                     val[pos]=del(l[pos],val[pos]+1);
135                 return ret;
136             }
137             else
138                 if(a<val[pos])
139                     return del(l[pos],a);
140                 else
141                     return del(r[pos],a);
142         }
143         bool find(int &pos,const Tp &a)
144         {
145             if(!pos)
146                 return false;
147             if(a<val[pos])
148                 return find(l[pos],a);
149             else
150                 return (val[pos]==a || find(r[pos],a));
151         }
152         Tp pred(int &pos,const Tp &a)
153         {
154             if(!pos)
155                 return a;
156             if(a>val[pos])
157             {
158                 Tp ret(pred(r[pos],a));
159                 if(ret==a)
160                     return val[pos];
161                 else
162                     return ret;
163             }
164             return pred(l[pos],a);
165         }
166         Tp succ(int &pos,const Tp &a)
167         {
168             if(!pos)
169                 return a;
170             if(a<val[pos])
171             {
172                 Tp ret(succ(l[pos],a));
173                 if(ret==a)
174                     return val[pos];
175                 else
176                     return ret;
177             }
```

```
178             return succ(r[pos],a);
179         }
180         Tp min(int &pos)
181         {
182             if(l[pos])
183                 return min(l[pos]);
184             else
185                 return val[pos];
186         }
187         Tp max(int &pos)
188         {
189             if(r[pos])
190                 return max(r[pos]);
191             else
192                 return val[pos];
193         }
194         void dels(int &pos,const Tp &v)
195         {
196             if(!pos)
197                 return;
198             if(val[pos]<v)
199             {
200                 pos=r[pos];
201                 dels(pos,v);
202                 return;
203             }
204             dels(l[pos],v);
205             sz[pos]=1+sz[l[pos]]+sz[r[pos]];
206         }
207         int rank(const int &pos,const Tp &v)
208         {
209             if(val[pos]==v)
210                 return sz[l[pos]]+1;
211             if(v<val[pos])
212                 return rank(l[pos],v);
213             return rank(r[pos],v)+sz[l[pos]]+1;
214         }
215         Tp sel(const int &pos,const int &v)
216         {
217             if(sz[l[pos]]+1==v)
218                 return val[pos];
219             if(v>sz[l[pos]])
220                 return sel(r[pos],v-sz[l[pos]]-1);
221             return sel(l[pos],v);
222         }
223         Tp delsel(int &pos,int k)
224         {
225             --sz[pos];
226             if(sz[l[pos]]+1==k)
227             {
228                 Tp re(val[pos]);
```

```
229                 if(!l[pos] || !r[pos])
230                     pos=l[pos]+r[pos];
231                 else
232                     val[pos]=del(l[pos],val[pos]+1);
233                 return re;
234             }
235             if(k>sz[l[pos]])
236                 return delsel(r[pos],k-1-sz[l[pos]]);
237             return delsel(l[pos],k);
238         }
239 };
```

1.10 sparse table - rectangle

```
  1 #include<iostream>
  2 #include<cstdio>
  3 #include<algorithm>
  4
  5 #define MAXX 310
  6
  7 int mat[MAXX][MAXX];
  8 int table[9][9][MAXX][MAXX];
  9 int n;
 10 short lg[MAXX];
 11
 12 int main()
 13 {
 14     for(int i(2);i<MAXX;++i)
 15         lg[i]=lg[i>>1]+1;
 16     int T;
 17     std::cin >> T;
 18     while (T--)
 19     {
 20         std::cin >> n;
 21         for (int i = 0; i < n; ++i)
 22             for (int j = 0; j < n; ++j)
 23             {
 24                 std::cin >> mat[i][j];
 25                 table[0][0][i][j] = mat[i][j];
 26             }
 27
 28         // 从小到大计算，保证后来用到的都已经计算过
 29         for(int i=0;i<=lg[n];++i) // width
 30         {
 31             for(int j=0;j<=lg[n];++j) //height
 32             {
 33                 if(i==0 && j==0)
 34                     continue;
 35                 for(int ii=0;ii+(1<<j)<=n;++ii)
 36                     for(int jj=0;jj+(1<<i)<=n;++jj)
 37                         if(i==0)
 38                             table[i][j][ii][jj]=std::min(table[i][j
```

```
                                    -1][ii][jj],table[i][j-1][ii+(1<<(j
                                    -1))][jj]);
39                          else
40                              table[i][j][ii][jj]=std::min(table[i
                                    -1][j][ii][jj],table[i-1][j][ii][jj
                                    +(1<<(i-1))]);
41              }
42          }
43          long long N;
44          std::cin >> N;
45          int r1, c1, r2, c2;
46          for (int i = 0; i < N; ++i)
47          {
48              scanf("%d%d%d%d",&r1,&c1,&r2,&c2);
49              --r1;
50              --c1;
51              --r2;
52              --c2;
53              int w=lg[c2-c1+1];
54              int h=lg[r2-r1+1];
55              printf("%d\n",std::min(table[w][h][r1][c1],std::min(
                    table[w][h][r1][c2-(1<<w)+1],std::min(table[w][h][r2
                    -(1<<h)+1][c1],table[w][h][r2-(1<<h)+1][c2-(1<<w)
                    +1]))));
56          }
57      }
58      return 0;
59  }
```

1.11 sparse table - square

```
 1  int num[MAXX][MAXX],max[MAXX][MAXX][10];
 2  short lg[MAXX];
 3
 4  int main()
 5  {
 6      for(i=2;i<MAXX;++i)
 7          lg[i]=lg[i>>1]+1;
 8      scanf("%hd␣%d",&n,&q);
 9      for(i=0;i<n;++i)
10          for(j=0;j<n;++j)
11          {
12              scanf("%d",num[i]+j);
13              max[i][j][0]=num[i][j];
14          }
15      for(k=1;k<=lg[n];++k)
16      {
17          l=n+1-(1<<k);
18          for(i=0;i<l;++i)
19              for(j=0;j<l;++j)
20                  max[i][j][k]=std::max(std::max(max[i][j][k-1],max[i
                        +(1<<(k-1))][j][k-1]),std::max(max[i][j+(1<<(k
```

```
                    -1))][k-1],max[i+(1<<(k-1))][j+(1<<(k-1))][k-1])
                    );
    }
    printf("Case %hd:\n",t);
    while(q--)
    {
        scanf("%hd %hd %hd",&i,&j,&l);
        --i;
        --j;
        k=lg[l];
        printf("%d\n",std::max(std::max(max[i][j][k],max[i][j+l
           -(1<<k)][k]),std::max(max[i+l-(1<<k)][j][k],max[i+l-(1<<
           k)][j+l-(1<<k)][k])));
    }
}
```

1.12 sparse table

```
int num[MAXX],min[MAXX][20];
int lg[MAXX];


int main()
{
    for(i=2;i<MAXX;++i)
        lg[i]=lg[i>>1]+1;
    scanf("%d %d",&n,&q);
    for(i=1;i<=n;++i)
    {
        scanf("%d",num+i);
        min[i][0]=num[i];
    }
    for(j=1;j<=lg[n];++j)
    {
        l=n+1-(1<<j);
        j_=j-1;
        j__=(1<<j_);
        for(i=1;i<=l;++i)
            min[i][j]=std::min(min[i][j_],min[i+j__][j_]);
    }
    printf("Case %hd:\n",t);
    while(q--)
    {
        scanf("%d %d",&i,&j);
        k=lg[j-i+1];
        printf("%d\n",std::min(min[i][k],min[j-(1<<k)+1][k]));
    }
}
```

1.13 treap

```
#include<cstdlib>
#include<ctime>
```

```
 3 #include<cstring>
 4
 5 struct node
 6 {
 7     node *ch[2];
 8     int sz,val,key;
 9     node(){memset(this,0,sizeof(node));}
10     node(int a);
11 }*null;
12
13 node::node(int a):sz(1),val(a),key(rand()-1){ch[0]=ch[1]=null;}
14
15 class Treap
16 {
17     inline void up(node *pos)
18     {
19         pos->sz=pos->ch[0]->sz+pos->ch[1]->sz+1;
20     }
21     inline void rot(node *&pos,int tp)
22     {
23         node *k(pos->ch[tp]);
24         pos->ch[tp]=k->ch[tp^1];
25         k->ch[tp^1]=pos;
26         up(pos);
27         up(k);
28         pos=k;
29     }
30
31     void insert(node *&pos,int val)
32     {
33         if(pos!=null)
34         {
35             int t(val>=pos->val);
36             insert(pos->ch[t],val);
37             if(pos->ch[t]->key<pos->key)
38                 rot(pos,t);
39             else
40                 up(pos);
41             return;
42         }
43         pos=new node(val);
44     }
45     void rec(node *pos)
46     {
47         if(pos!=null)
48         {
49             rec(pos->ch[0]);
50             rec(pos->ch[1]);
51             delete pos;
52         }
53     }
```

```
54      inline int sel(node *pos,int k)
55      {
56          while(pos->ch[0]->sz+1!=k)
57              if(pos->ch[0]->sz>=k)
58                  pos=pos->ch[0];
59              else
60              {
61                  k-=pos->ch[0]->sz+1;
62                  pos=pos->ch[1];
63              }
64          return pos->val;
65      }
66      void del(node *&pos,int val)
67      {
68          if(pos!=null)
69          {
70              if(pos->val==val)
71              {
72                  int t(pos->ch[1]->key<pos->ch[0]->key);
73                  if(pos->ch[t]==null)
74                  {
75                      delete pos;
76                      pos=null;
77                      return;
78                  }
79                  rot(pos,t);
80                  del(pos->ch[t^1],val);
81              }
82              else
83                  del(pos->ch[val>pos->val],val);
84              up(pos);
85          }
86      }
87      public:
88      node *rt;
89
90      Treap():rt(null){}
91      inline void insert(int val)
92      {
93          insert(rt,val);
94      }
95      inline void reset()
96      {
97          rec(rt);
98          rt=null;
99      }
100     inline int sel(int k)
101     {
102         if(k<1 || k>rt->sz)
103             return 0;
104         return sel(rt,rt->sz+1-k);
```

```
105     }
106     inline void del(int val)
107     {
108         del(rt,val);
109     }
110     inline int size()
111     {
112         return rt->sz;
113     }
114 }treap[MAXX];
115 
116 init:
117 {
118     srand(time(0));
119     null=new node();
120     null->val=0xc0c0c0c0;
121     null->sz=0;
122     null->key=RAND_MAX;
123     null->ch[0]=null->ch[1]=null;
124     for(i=0;i<MAXX;++i)
125         treap[i].rt=null;
126 }
```

2 Geometry

2.1 3D

```
1  struct pv
2  {
3    double x,y,z;
4    pv() {}
5    pv(double xx,double yy,double zz):x(xx),y(yy),z(zz) {}
6    pv operator -(const pv& b)const
7    {
8      return pv(x-b.x,y-b.y,z-b.z);
9    }
10   pv operator *(const pv& b)const
11   {
12     return pv(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
13   }
14   double operator &(const pv& b)const
15   {
16     return x*b.x+y*b.y+z*b.z;
17   }
18 };
19 
20 //模
21 double Norm(pv p)
22 {
23   return sqrt(p&p);
24 }
25 
```

```
//绕单位向量 V 旋转 theta 角度
pv Trans(pv pa,pv V,double theta)
{
    double s = sin(theta);
    double c = cos(theta);
    double x,y,z;
    x = V.x;
    y = V.y;
    z = V.z;
    pv pp =
        pv(
                (x*x*(1-c)+c)*pa.x+(x*y*(1-c)-z*s)*pa.y+(x*z*(1-c)+
                   y*s)*pa.z,
                (y*x*(1-c)+z*s)*pa.x+(y*y*(1-c)+c)*pa.y+(y*z*(1-c)-
                   x*s)*pa.z,
                (x*z*(1-c)-y*s)*pa.x+(y*z*(1-c)+x*s)*pa.y+(z*z*(1-c
                   )+c)*pa.z
           );
    return pp;
}

//经纬度转换

x = r × sin(θ) × cos(α)
y = r × sin(θ) × sin(α)
z = r × cos(θ)

r = √(x × 2 + y × 2 + z × 2)
α=atan(y/x);
θ=acos(z/r);

r ∈ [0, ∞)
α ∈ [0, 2π]
θ ∈ [0, π]

lat ∈ [-π/2, π/2]
lng ∈ [-π, π]

pv getpv(double lat,double lng,double r)
{
  lat += pi/2;
  lng += pi;
  return
    pv(r*sin(lat)*cos(lng),r*sin(lat)*sin(lng),r*cos(lat));
}

//经纬度球面距离

#include<cstdio>
#include<cmath>

```

```
#define MAXX 1111

char buf[MAXX];
const double r=6875.0/2,pi=acos(-1.0);
double a,b,c,x1,x2,y2,ans;

int main()
{
    double y1;
    while(gets(buf)!=NULL)
    {
        gets(buf);
        gets(buf);

        scanf("%lf^%lf'%lf\" %s\n",&a,&b,&c,buf);
        x1=a+b/60+c/3600;
        x1=x1*pi/180;
        if(buf[0]=='S')
            x1=-x1;

        scanf("%s",buf);
        scanf("%lf^%lf'%lf\" %s\n",&a,&b,&c,buf);
        y1=a+b/60+c/3600;
        y1=y1*pi/180;
        if(buf[0]=='W')
            y1=-y1;

        gets(buf);

        scanf("%lf^%lf'%lf\" %s\n",&a,&b,&c,buf);
        x2=a+b/60+c/3600;
        x2=x2*pi/180;
        if(buf[0]=='S')
            x2=-x2;

        scanf("%s",buf);
        scanf("%lf^%lf'%lf\" %s\n",&a,&b,&c,buf);
        y2=a+b/60+c/3600;
        y2=y2*pi/180;
        if(buf[0]=='W')
            y2=-y2;

        ans=acos(cos(x1)*cos(x2)*cos(y1-y2)+sin(x1)*sin(x2))*r;
        printf("The distance to the iceberg: %.2lf miles.\n",ans);
        if(ans+0.005<100)
            puts("DANGER!");

        gets(buf);
    }
    return 0;
}
```

```
125
126 inline bool ZERO(const double &a)
127 {
128     return fabs(a)<eps;
129 }
130
131 //三维向量是否为零
132 inline bool ZERO(pv p)
133 {
134     return (ZERO(p.x) && ZERO(p.y) && ZERO(p.z));
135 }
136
137 //直线相交
138 bool LineIntersect(Line3D L1, Line3D L2)
139 {
140     pv s = L1.s-L1.e;
141     pv e = L2.s-L2.e;
142     pv p  = s*e;
143     if (ZERO(p))
144         return false;    //是否平行
145     p = (L2.s-L1.e)*(L1.s-L1.e);
146     return ZERO(p&L2.e);          //是否共面
147 }
148
149 //线段相交
150 bool inter(pv a,pv b,pv c,pv d)
151 {
152     pv ret = (a-b)*(c-d);
153     pv t1 = (b-a)*(c-a);
154     pv t2 = (b-a)*(d-a);
155     pv t3 = (d-c)*(a-c);
156     pv t4 = (d-c)*(b-c);
157     return sgn(t1&ret)*sgn(t2&ret) < 0 && sgn(t3&ret)*sgn(t4&ret) <
            0;
158 }
159
160 //点在直线上
161 bool OnLine(pv p, Line3D L)
162 {
163     return ZERO((p-L.s)*(L.e-L.s));
164 }
165
166 //点在线段上
167 bool OnSeg(pv p, Line3D L)
168 {
169     return (ZERO((L.s-p)*(L.e-p)) && EQ(Norm(p-L.s)+Norm(p-L.e),
           Norm(L.e-L.s)));
170 }
171
172 //点到直线距离
173 double Distance(pv p, Line3D L)
```

```
{
    return (Norm((p-L.s)*(L.e-L.s))/Norm(L.e-L.s));
}

//线段夹角
//范围值为[0,π 之间的弧度]
double Inclination(Line3D L1, Line3D L2)
{
    pv u = L1.e - L1.s;
    pv v = L2.e - L2.s;
    return acos( (u & v) / (Norm(u)*Norm(v)) );
}
```

2.2 3DCH

```
#include<cstdio>
#include<cmath>
#include<vector>
#include<algorithm>

#define MAXX 1111
#define eps 1e-8
#define inf 1e20

struct pv
{
    double x,y,z;
    pv(){}
    pv(const double &xx,const double &yy,const double &zz):x(xx),y(
       yy),z(zz){}
    inline pv operator-(const pv &i)const
    {
        return pv(x-i.x,y-i.y,z-i.z);
    }
    inline pv operator+(const pv &i)const
    {
        return pv(x+i.x,y+i.y,z+i.z);
    }
    inline pv operator+=(const pv &i)
    {
        x+=i.x;
        y+=i.y;
        z+=i.z;
        return *this;
    }
    inline pv operator*(const pv &i)const //叉积
    {
        return pv(y*i.z-z*i.y,z*i.x-x*i.z,x*i.y-y*i.x);
    }
    inline pv operator*(const double a)const
    {
        return pv(x*a,y*a,z*a);
```

```
37      }
38      inline double operator^(const pv &i)const //点积
39      {
40          return x*i.x+y*i.y+z*i.z;
41      }
42      inline double len()
43      {
44          return sqrt(x*x+y*y+z*z);
45      }
46  };
47
48  struct pla
49  {
50      short a,b,c;
51      bool ok;
52      pla(){}
53      pla(const short &aa,const short &bb,const short &cc):a(aa),b(bb
            ),c(cc),ok(true){}
54      inline void set();
55      inline void print()
56      {
57          printf("%hd␣%hd␣%hd\n",a,b,c);
58      }
59  };
60
61  pv pnt[MAXX];
62  std::vector<pla>fac;
63  int to[MAXX][MAXX];
64
65  inline void pla::set()
66  {
67      to[a][b]=to[b][c]=to[c][a]=fac.size();
68  }
69
70  inline double ptof(const pv &p,const pla &f) //点面距离?
71  {
72      return (pnt[f.b]-pnt[f.a])*(pnt[f.c]-pnt[f.a])^(p-pnt[f.a]);
73  }
74
75  inline double vol(const pv &a,const pv &b,const pv &c,const pv &d)
        //有向体积，即六面体体
        积*6
76  {
77      return (b-a)*(c-a)^(d-a);
78  }
79
80  inline double ptof(const pv &p,const short &f) //点到号面的距离pf
81  {
82      return fabs(vol(pnt[fac[f].a],pnt[fac[f].b],pnt[fac[f].c],p)/((
            pnt[fac[f].b]-pnt[fac[f].a])*(pnt[fac[f].c]-pnt[fac[f].a])).
            len());
```

```
 83 }
 84
 85 void dfs(const short&,const short&);
 86
 87 void deal(const short &p,const short &a,const short &b)
 88 {
 89     if(fac[to[a][b]].ok)
 90         if(ptof(pnt[p],fac[to[a][b]])>eps)
 91             dfs(p,to[a][b]);
 92         else
 93         {
 94             pla add(b,a,p);
 95             add.set();
 96             fac.push_back(add);
 97         }
 98 }
 99
100 void dfs(const short &p,const short &now)
101 {
102     fac[now].ok=false;
103     deal(p,fac[now].b,fac[now].a);
104     deal(p,fac[now].c,fac[now].b);
105     deal(p,fac[now].a,fac[now].c);
106 }
107
108 inline void make(int n)
109 {
110     static int i,j;
111     fac.resize(0);
112     if(n<4)
113         return;
114
115     for(i=1;i<n;++i)
116         if((pnt[0]-pnt[i]).len()>eps)
117         {
118             std::swap(pnt[i],pnt[1]);
119             break;
120         }
121     if(i==n)
122         return;
123
124     for(i=2;i<n;++i)
125         if(((pnt[0]-pnt[1])*(pnt[1]-pnt[i])).len()>eps)
126         {
127             std::swap(pnt[i],pnt[2]);
128             break;
129         }
130     if(i==n)
131         return;
132
133     for(i=3;i<n;++i)
```

```
134         if(fabs((pnt[0]-pnt[1])*(pnt[1]-pnt[2])^(pnt[2]-pnt[i]))>
                eps)
135         {
136             std::swap(pnt[3],pnt[i]);
137             break;
138         }
139     if(i==n)
140         return;
141
142     for(i=0;i<4;++i)
143     {
144         pla add((i+1)%4,(i+2)%4,(i+3)%4);
145         if(ptof(pnt[i],add)>0)
146             std::swap(add.c,add.b);
147         add.set();
148         fac.push_back(add);
149     }
150     for(;i<n;++i)
151         for(j=0;j<fac.size();++j)
152             if(fac[j].ok && ptof(pnt[i],fac[j])>eps)
153             {
154                 dfs(i,j);
155                 break;
156             }
157
158     short tmp(fac.size());
159     fac.resize(0);
160     for(i=0;i<tmp;++i)
161         if(fac[i].ok)
162             fac.push_back(fac[i]);
163 }
164
165 inline pv gc() //重心
166 {
167     pv re(0,0,0),o(0,0,0);
168     double all(0),v;
169     for(int i=0;i<fac.size();++i)
170     {
171         v=vol(o,pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
172         re+=(pnt[fac[i].a]+pnt[fac[i].b]+pnt[fac[i].c])*0.25f*v;
173         all+=v;
174     }
175     return re*(1/all);
176 }
177
178 inline bool same(const short &s,const short &t) //两面是否相等
179 {
180     pv &a=pnt[fac[s].a],&b=pnt[fac[s].b],&c=pnt[fac[s].c];
181     return fabs(vol(a,b,c,pnt[fac[t].a]))<eps && fabs(vol(a,b,c,pnt
            [fac[t].b]))<eps && fabs(vol(a,b,c,pnt[fac[t].c]))<eps;
182 }
```

```
183 
184 //表面多边形数目
185 inline int facetcnt()
186 {
187     int ans=0;
188     static int i,j;
189     for(i=0;i<fac.size();++i)
190     {
191         for(j=0;j<i;++j)
192             if(same(i,j))
193                 break;
194         if(j==i)
195             ++ans;
196     }
197     return ans;
198 }
199 
200 //表面三角形数目
201 inline short trianglecnt()
202 {
203     return fac.size();
204 }
205 
206 //三点构成的三角形面积*2
207 inline double area(const pv &a,const pv &b,const pv &c)
208 {
209         return ((b-a)*(c-a)).len();
210 }
211 
212 //表面积
213 inline double area()
214 {
215     double ret(0);
216     static int i;
217     for(i=0;i<fac.size();++i)
218         ret+=area(pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
219     return ret/2;
220 }
221 
222 //体积
223 inline double volume()
224 {
225     pv o(0,0,0);
226     double ret(0);
227     for(short i(0);i<fac.size();++i)
228         ret+=vol(o,pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
229     return fabs(ret/6);
230 }
```

2.3 circle's area

```
1 //去重
```

```
{
    for (int i = 0; i < n; i++)
    {
        scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
        del[i] = false;
    }
    for (int i = 0; i < n; i++)
        if (del[i] == false)
        {
            if (c[i].r == 0.0)
                del[i] = true;
            for (int j = 0; j < n; j++)
                if (i != j)
                    if (del[j] == false)
                        if (cmp(Point(c[i].c,c[j].c).Len()+c[i].r,c
                           [j].r) <= 0)
                            del[i] = true;
        }
    tn = n;
    n = 0;
    for (int i = 0; i < tn; i++)
        if (del[i] == false)
            c[n++] = c[i];
}

//ans[i表示被覆盖]次的面积i
const double pi = acos(-1.0);
const double eps = 1e-8;
struct Point
{
    double x,y;
    Point(){}
    Point(double _x,double _y)
    {
        x = _x;
        y = _y;
    }
    double Length()
    {
        return sqrt(x*x+y*y);
    }
};
struct Circle
{
    Point c;
    double r;
};
struct Event
{
    double tim;
    int typ;
```

```
52      Event(){}
53      Event(double _tim,int _typ)
54      {
55          tim = _tim;
56          typ = _typ;
57      }
58  };
59
60  int cmp(const double& a,const double& b)
61  {
62      if (fabs(a-b) < eps)    return 0;
63      if (a < b)  return -1;
64      return 1;
65  }
66
67  bool Eventcmp(const Event& a,const Event& b)
68  {
69      return cmp(a.tim,b.tim) < 0;
70  }
71
72  double Area(double theta,double r)
73  {
74      return 0.5*r*r*(theta-sin(theta));
75  }
76
77  double xmult(Point a,Point b)
78  {
79      return a.x*b.y-a.y*b.x;
80  }
81
82  int n,cur,tote;
83  Circle c[1000];
84  double ans[1001],pre[1001],AB,AC,BC,theta,fai,a0,a1;
85  Event e[4000];
86  Point lab;
87
88  int main()
89  {
90      while (scanf("%d",&n) != EOF)
91      {
92          for (int i = 0;i < n;i++)
93              scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
94          for (int i = 1;i <= n;i++)
95              ans[i] = 0.0;
96          for (int i = 0;i < n;i++)
97          {
98              tote = 0;
99              e[tote++] = Event(-pi,1);
100             e[tote++] = Event(pi,-1);
101             for (int j = 0;j < n;j++)
102                 if (j != i)
```

```
103                 {
104                     lab = Point(c[j].c.x-c[i].c.x,c[j].c.y-c[i].c.y
                            );
105                     AB = lab.Length();
106                     AC = c[i].r;
107                     BC = c[j].r;
108                     if (cmp(AB+AC,BC) <= 0)
109                     {
110                         e[tote++] = Event(-pi,1);
111                         e[tote++] = Event(pi,-1);
112                         continue;
113                     }
114                     if (cmp(AB+BC,AC) <= 0) continue;
115                     if (cmp(AB,AC+BC) > 0)  continue;
116                     theta = atan2(lab.y,lab.x);
117                     fai = acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
118                     a0 = theta-fai;
119                     if (cmp(a0,-pi) < 0)    a0 += 2*pi;
120                     a1 = theta+fai;
121                     if (cmp(a1,pi) > 0)  a1 -= 2*pi;
122                     if (cmp(a0,a1) > 0)
123                     {
124                         e[tote++] = Event(a0,1);
125                         e[tote++] = Event(pi,-1);
126                         e[tote++] = Event(-pi,1);
127                         e[tote++] = Event(a1,-1);
128                     }
129                     else
130                     {
131                         e[tote++] = Event(a0,1);
132                         e[tote++] = Event(a1,-1);
133                     }
134                 }
135             sort(e,e+tote,Eventcmp);
136             cur = 0;
137             for (int j = 0;j < tote;j++)
138             {
139                 if (cur != 0 && cmp(e[j].tim,pre[cur]) != 0)
140                 {
141                     ans[cur] += Area(e[j].tim-pre[cur],c[i].r);
142                     ans[cur] += xmult(Point(c[i].c.x+c[i].r*cos(pre
                            [cur]),c[i].c.y+c[i].r*sin(pre[cur])),
143                                 Point(c[i].c.x+c[i].r*cos(e[j].tim),c[i
                                    ].c.y+c[i].r*sin(e[j].tim)))/2.0;
144                 }
145                 cur += e[j].typ;
146                 pre[cur] = e[j].tim;
147             }
148         }
149         for (int i = 1;i < n;i++)
150             ans[i] -= ans[i+1];
```

```
151         for (int i = 1;i <= n;i++)
152             printf("[%d] = %.3f\n",i,ans[i]);
153     }
154     return 0;
155 }
```

2.4 circle

```
 1 //单位圆覆盖
 2 #include<cstdio>
 3 #include<cmath>
 4 #include<algorithm>
 5 #include<vector>
 6
 7 #define eps 1e-8
 8 #define MAXX 211
 9 const double pi(acos(-1));
10 typedef std::pair<double,int> pdi;
11
12 struct pv
13 {
14     double x,y;
15     pv(double a=0,double b=0):x(a),y(b){}
16     pv operator-(const pv &i)const
17     {
18         return pv(x-i.x,y-i.y);
19     }
20     double len()
21     {
22         return hypot(x,y);
23     }
24 }pnt[MAXX];
25
26 std::vector<pdi>alpha(MAXX<<1);
27
28 inline int solve(double r) //radius
29 {
30     static int ans,sum,i,j;
31     sum=ans=0;
32     for(i=0;i<n;++i)
33     {
34         alpha.resize(0);
35         static double d,theta,phi;
36         static pv vec;
37         for(j=0;j<n;++j)
38         {
39             if(j==i || (d=(vec=pnt[i]-pnt[j]).len())>2*r+eps)
40                 continue;
41             if((theta=atan2(vec.y,vec.x))<-eps)
42                 theta+=2*pi;
43             phi=acos(d/(2*r));
44             alpha.push_back(pdi(theta-phi+2*pi,-1));
```

```
45              alpha.push_back(pdi(theta+phi+2*pi,1));
46          }
47          std::sort(alpha.begin(),alpha.end());
48          for(j=0;j<alpha.size();++j)
49          {
50              sum-=alpha[j].second;
51              if(sum>ans)
52                  ans=sum;
53          }
54      }
55      return ans+1;
56  }
```

2.5 closest point pair

```
1   //演算法笔记1
2
3   struct Point {double x, y;} p[10], t[10];
4   bool cmpx(const Point& i, const Point& j) {return i.x < j.x;}
5   bool cmpy(const Point& i, const Point& j) {return i.y < j.y;}
6
7   double DnC(int L, int R)
8   {
9       if (L >= R) return 1e9; // 沒有點、只有一個點。
10
11      /* : 把所有點分成左右兩側，點數盡量一樣多。Divide */
12
13      int M = (L + R) / 2;
14
15      /* : 左側、右側分別遞迴求解。Conquer */
16
17      double d = min(DnC(L,M), DnC(M+1,R));
18      //  if (d == 0.0) return d; // 提早結束
19
20      /* : 尋找靠近中線的點，並依座標排序。MergeYO(NlogN)。 */
21
22      int N = 0;  // 靠近中線的點數目
23      for (int i=M;   i>=L && p[M].x - p[i].x < d; --i) t[N++] = p[i
            ];
24      for (int i=M+1; i<=R && p[i].x - p[M].x < d; ++i) t[N++] = p[i
            ];
25      sort(t, t+N, cmpy); // Quicksort O(NlogN)
26
27      /* : 尋找橫跨兩側的最近點對。MergeO(N)。 */
28
29      for (int i=0; i<N-1; ++i)
30          for (int j=1; j<=2 && i+j<N; ++j)
31              d = min(d, distance(t[i], t[i+j]));
32
33      return d;
34  }
35
```

```
36 double closest_pair()
37 {
38     sort(p, p+10, cmpx);
39     return DnC(0, N-1);
40 }
41
42
43 //演算法笔记2
44
45 struct Point {double x, y;} p[10], t[10];
46 bool cmpx(const Point& i, const Point& j) {return i.x < j.x;}
47 bool cmpy(const Point& i, const Point& j) {return i.y < j.y;}
48
49 double DnC(int L, int R)
50 {
51     if (L >= R) return 1e9; // 沒有點、只有一個點。
52
53     /* : 把所有點分成左右兩側，點數盡量一樣多。Divide */
54
55     int M = (L + R) / 2;
56
57     // 先把中線的座標記起來，因為待會重新排序之後會跑掉。X
58     double x = p[M].x;
59
60     /* : 左側、右側分別遞迴求解。Conquer */
61
62     // 遞迴求解，並且依照座標重新排序。Y
63     double d = min(DnC(L,M), DnC(M+1,R));
64     //  if (d == 0.0) return d; // 提早結束
65
66     /* : 尋找靠近中線的點，並依座標排序。MergeYO(N)。 */
67
68     // 尋找靠近中線的點，先找左側。各點已照座標排序了。Y
69     int N = 0;  // 靠近中線的點數目
70     for (int i=0; i<=M; ++i)
71         if (x - p[i].x < d)
72             t[N++] = p[i];
73
74     // 尋找靠近中線的點，再找右側。各點已照座標排序了。Y
75     int P = N;  // 為分隔位置P
76     for (int i=M+1; i<=R; ++i)
77         if (p[i].x - x < d)
78             t[N++] = p[i];
79
80     // 以座標排序。使用YMerge 方式，合併已排序的兩陣列。Sort
81     inplace_merge(t, t+P, t+N, cmpy);
82
83     /* : 尋找橫跨兩側的最近點對。MergeO(N)。 */
84
85     for (int i=0; i<N; ++i)
86         for (int j=1; j<=2 && i+j<N; ++j)
```

```
87              d = min(d, distance(t[i], t[i+j]));
88
89      /* : 重新以座標排序所有點。MergeYO(N)。 */
90
91      // 如此一來，更大的子問題就可以直接使用Merge 。Sort
92      inplace_merge(p+L, p+M+1, p+R+1, cmpy);
93
94      return d;
95  }
96
97  double closest_pair()
98  {
99      sort(p, p+10, cmpx);
100     return DnC(0, N-1);
101 }
102
103 //mzry
104 //分治
105 double calc_dis(Point &a ,Point &b) {
106   return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
107 }
108 //別忘了排序
109 bool operator<(const Point &a ,const Point &b) {
110   if(a.y != b.y) return a.x < b.x;
111   return a.x < b.x;
112 }
113 double Gao(int l ,int r ,Point pnts[]) {
114   double ret = inf;
115   if(l == r) return ret;
116   if(l+1 ==r) {
117     ret = min(calc_dis(pnts[l],pnts[l+1]) ,ret);
118     return ret;
119   }
120   if(l+2 ==r) {
121     ret = min(calc_dis(pnts[l],pnts[l+1]) ,ret);
122     ret = min(calc_dis(pnts[l],pnts[l+2]) ,ret);
123     ret = min(calc_dis(pnts[l+1],pnts[l+2]) ,ret);
124     return ret;
125   }
126
127   int mid = l+r>>1;
128   ret = min (ret ,Gao(l ,mid,pnts));
129   ret = min (ret , Gao(mid+1, r,pnts));
130
131   for(int c = l ; c<=r; c++)
132     for(int d = c+1; d <=c+7 && d<=r; d++) {
133       ret = min(ret , calc_dis(pnts[c],pnts[d]));
134     }
135   return ret;
136 }
137
```

```
138 //增量
139 #include <iostream>
140 #include <cstdio>
141 #include <cstring>
142 #include <map>
143 #include <vector>
144 #include <cmath>
145 #include <algorithm>
146 #define Point pair<double,double>
147 using namespace std;
148
149 const int step[9][2] =
       {{-1,-1},{-1,0},{-1,1},{0,-1},{0,0},{0,1},{1,-1},{1,0},{1,1}};
150 int n,x,y,nx,ny;
151 map<pair<int,int>,vector<Point > > g;
152 vector<Point > tmp;
153 Point p[20000];
154 double tx,ty,ans,nowans;
155 vector<Point >::iterator it,op,ed;
156 pair<int,int> gird;
157 bool flag;
158
159 double Dis(Point p0,Point p1)
160 {
161   return sqrt((p0.first-p1.first)*(p0.first-p1.first)+
162         (p0.second-p1.second)*(p0.second-p1.second));
163 }
164
165 double CalcDis(Point p0,Point p1,Point p2)
166 {
167   return Dis(p0,p1)+Dis(p0,p2)+Dis(p1,p2);
168 }
169
170 void build(int n,double w)
171 {
172   g.clear();
173   for (int i = 0;i < n;i++)
174     g[make_pair((int)floor(p[i].first/w),(int)floor(p[i].second/w))
          ].push_back(p[i]);
175 }
176
177 int main()
178 {
179   int t;
180   scanf("%d",&t);
181   for (int ft = 1;ft <= t;ft++)
182   {
183     scanf("%d",&n);
184     for (int i = 0;i < n;i++)
185     {
186       scanf("%lf%lf",&tx,&ty);
```

```
187       p[i] = make_pair(tx,ty);
188     }
189     random_shuffle(p,p+n);
190     ans = CalcDis(p[0],p[1],p[2]);
191     build(3,ans/2.0);
192     for (int i = 3;i < n;i++)
193     {
194       x = (int)floor(2.0*p[i].first/ans);
195       y = (int)floor(2.0*p[i].second/ans);
196       tmp.clear();
197       for (int k = 0;k < 9;k++)
198       {
199         nx = x+step[k][0];
200         ny = y+step[k][1];
201         gird = make_pair(nx,ny);
202         if (g.find(gird) != g.end())
203         {
204           op = g[gird].begin();
205           ed = g[gird].end();
206           for (it = op;it != ed;it++)
207             tmp.push_back(*it);
208         }
209       }
210       flag = false;
211       for (int j = 0;j < tmp.size();j++)
212         for (int k = j+1;k < tmp.size();k++)
213         {
214           nowans = CalcDis(p[i],tmp[j],tmp[k]);
215           if (nowans < ans)
216           {
217             ans = nowans;
218             flag = true;
219           }
220         }
221       if (flag == true)
222         build(i+1,ans/2.0);
223       else
224         g[make_pair((int)floor(2.0*p[i].first/ans),(int)floor(2.0*p
              [i].second/ans))].push_back(p[i]);
225     }
226     printf("%.3f\n",ans);
227   }
228 }
```

2.6 half-plane intersection

```
1 //解析几何方式abc
2 inline pv ins(const pv &p1,const pv &p2)
3 {
4     u=fabs(a*p1.x+b*p1.y+c);
5     v=fabs(a*p2.x+b*p2.y+c);
6     return pv((p1.x*v+p2.x*u)/(u+v),(p1.y*v+p2.y*u)/(u+v));
```

```
 7 }
 8
 9 inline void get(const pv& p1,const pv& p2,double & a,double & b,
       double & c)
10 {
11     a=p2.y-p1.y;
12     b=p1.x-p2.x;
13     c=p2.x*p1.y-p2.y*p1.x;
14 }
15
16 inline pv ins(const pv &x,const pv &y)
17 {
18     get(x,y,d,e,f);
19     return pv((b*f-c*e)/(a*e-b*d),(a*f-c*d)/(b*d-a*e));
20 }
21
22 std::vector<pv>p[2];
23 inline bool go()
24 {
25     k=0;
26     p[k].resize(0);
27     p[k].push_back(pv(-inf,inf));
28     p[k].push_back(pv(-inf,-inf));
29     p[k].push_back(pv(inf,-inf));
30     p[k].push_back(pv(inf,inf));
31     for(i=0;i<n;++i)
32     {
33         get(pnt[i],pnt[(i+1)%n],a,b,c);
34         c+=the*sqrt(a*a+b*b);
35         p[!k].resize(0);
36         for(l=0;l<p[k].size();++l)
37             if(a*p[k][l].x+b*p[k][l].y+c<eps)
38                 p[!k].push_back(p[k][l]);
39             else
40             {
41                 m=(l+p[k].size()-1)%p[k].size();
42                 if(a*p[k][m].x+b*p[k][m].y+c<-eps)
43                     p[!k].push_back(ins(p[k][m],p[k][l]));
44                 m=(l+1)%p[k].size();
45                 if(a*p[k][m].x+b*p[k][m].y+c<-eps)
46                     p[!k].push_back(ins(p[k][m],p[k][l]));
47             }
48         k=!k;
49         if(p[k].empty())
50             break;
51     }
52     //结果在p[k中]
53     return p[k].empty();
54 }
55
56 //计算几何方式
```

```
57 //本例求多边形核
58
59 inline pv ins(const pv &a,const pv &b)
60 {
61     u=fabs(ln.cross(a-pnt[i]));
62     v=fabs(ln.cross(b-pnt[i]))+u;
63     tl=b-a;
64     return pv(u*tl.x/v+a.x,u*tl.y/v+a.y);
65 }
66
67 int main()
68 {
69     j=0;
70     for(i=0;i<n;++i)
71     {
72         ln=pnt[(i+1)%n]-pnt[i];
73         p[!j].resize(0);
74         for(k=0;k<p[j].size();++k)
75             if(ln.cross(p[j][k]-pnt[i])<=0)
76                 p[!j].push_back(p[j][k]);
77             else
78             {
79                 l=(k-1+p[j].size())%p[j].size();
80                 if(ln.cross(p[j][l]-pnt[i])<0)
81                     p[!j].push_back(ins(p[j][k],p[j][l]));
82                 l=(k+1)%p[j].size();
83                 if(ln.cross(p[j][l]-pnt[i])<0)
84                     p[!j].push_back(ins(p[j][k],p[j][l]));
85             }
86         j=!j;
87     }
88     //结果在p[j中]
89 }
90
91 //mrzy
92
93 bool HPIcmp(Line a, Line b)
94 {
95     if (fabs(a.k - b.k) > eps)
96         return a.k < b.k;
97     return ((a.s - b.s) * (b.e-b.s)) < 0;
98 }
99
100 Line Q[100];
101
102 void HPI(Line line[], int n, Point res[], int &resn)
103 {
104     int tot = n;
105     std::sort(line, line + n, HPIcmp);
106     tot = 1;
107     for (int i = 1; i < n; i++)
```

```
108         if (fabs(line[i].k - line[i - 1].k) > eps)
109             line[tot++] = line[i];
110     int head = 0, tail = 1;
111     Q[0] = line[0];
112     Q[1] = line[1];
113     resn = 0;
114     for (int i = 2; i < tot; i++)
115     {
116         if (fabs((Q[tail].e-Q[tail].s)*(Q[tail - 1].e-Q[tail - 1].s
                )) < eps || fabs((Q[head].e-Q[head].s)*(Q[head + 1].e-Q[
                head + 1].s)) < eps)
117             return;
118         while (head < tail && (((Q[tail]&Q[tail - 1]) - line[i].s)
                * (line[i].e-line[i].s)) > eps)
119             --tail;
120         while (head < tail && (((Q[head]&Q[head + 1]) - line[i].s)
                * (line[i].e-line[i].s)) > eps)
121             ++head;
122         Q[++tail] = line[i];
123     }
124     while (head < tail && (((Q[tail]&Q[tail - 1]) - Q[head].s) * (Q
            [head].e-Q[head].s)) > eps)
125         tail--;
126     while (head < tail && (((Q[head]&Q[head + 1]) - Q[tail].s) * (Q
            [tail].e-Q[tail].s)) > eps)
127         head++;
128     if (tail <= head + 1)
129         return;
130     for (int i = head; i < tail; i++)
131         res[resn++] = Q[i] & Q[i + 1];
132     if (head < tail + 1)
133         res[resn++] = Q[head] & Q[tail];
134 }
```

2.7 intersection of circle and poly

```
 1 pv c;
 2 double r;
 3
 4 inline double cal(const pv &a,const pv &b)
 5 {
 6     static double A,B,C,x,y,ts;
 7     A=(b-c).len();
 8     B=(a-c).len();
 9     C=(a-b).len();
10     if(A<r && B<r)
11         return (a-c).cross(b-c)/2;
12     x=((a-b).dot(c-b)+sqrt(r*r*C*C-sqr((a-b).cross(c-b))))/C;
13     y=((b-a).dot(c-a)+sqrt(r*r*C*C-sqr((b-a).cross(c-a))))/C;
14     ts=(a-c).cross(b-c)/2;
15
16     if(A<r && B>=r)
```

```
17          return asin(ts*(1-x/C)*2/r/B*(1-eps))*r*r/2+ts*x/C;
18      if(A>=r && B<r)
19          return asin(ts*(1-y/C)*2/r/A*(1-eps))*r*r/2+ts*y/C;
20
21      if(fabs((a-c).cross(b-c))>=r*C || (b-a).dot(c-a)<=0 || (a-b).
            dot(c-b)<=0)
22      {
23          if((a-c).dot(b-c)<0)
24          {
25              if((a-c).cross(b-c)<0)
26                  return (-pi-asin((a-c).cross(b-c)/A/B*(1-eps)))*r*r
                        /2;
27              return (pi-asin((a-c).cross(b-c)/A/B*(1-eps)))*r*r/2;
28          }
29          return asin((a-c).cross(b-c)/A/B*(1-eps))*r*r/2;
30      }
31
32      return (asin(ts*(1-x/C)*2/r/B*(1-eps))+asin(ts*(1-y/C)*2/r/A
            *(1-eps)))*r*r/2+ts*((y+x)/C-1);
33  }
34
35  inline double get(pv *the,int n)
36  {
37      double ans=0;
38      for(int i=0;i<n;++i)
39          ans+=cal(the[i],the[(i+1)%n]);
40      return ans;
41  }
```

2.8 k-d tree

```
1  /*
2  有个很关键的剪枝，在计算完与 mid 点的距离后，我们应该先进入左右哪个子树？我们应
       该先进入对于当前维度，查询点位于的那一边。显然，在查询点所在的子树，更容易查
       找出正确解。
3
4  那么当进入完左或右子树后，以查询点为圆心做圆，如果当前维度，查询点距离 mid 的距
       离（另一个子树中的点距离查询点的距离肯定大于这个距离）比堆里的最大值还大，那
       么就不再递归另一个子树。注意一下：如果堆里的元素个数不足 M，仍然还要进入另一
       棵子树。
5
6  说白了就是随便乱搞啦……
7  */
8  // hysbz 2626
9  #include<cstdio>
10 #include<algorithm>
11 #include<queue>
12
13 inline long long sqr(long long a){ return a*a;}
14 typedef std::pair<long long,int> pli;
15
16 #define MAXX 100111
```

```
17 #define MAX (MAXX<<2)
18 #define inf 0x3f3f3f3fll
19 int idx;
20
21 struct PNT
22 {
23     long long x[2];
24     int lb;
25     bool operator<(const PNT &i)const
26     {
27         return x[idx]<i.x[idx];
28     }
29     pli dist(const PNT &i)const
30     {
31         return pli(-(sqr(x[0]-i.x[0])+sqr(x[1]-i.x[1])),lb);
32     }
33 }a[MAXX],the[MAX],p;
34
35 #define mid (l+r>>1)
36 #define lson (id<<1)
37 #define rson (id<<1|1)
38 #define lc lson,l,mid-1
39 #define rc rson,mid+1,r
40 int n,m;
41
42 long long rg[MAX][2][2];
43
44 void make(int id=1,int l=1,int r=n,int d=0)
45 {
46     the[id].lb=-1;
47     rg[id][0][0]=rg[id][1][0]=inf;
48     rg[id][0][1]=rg[id][1][1]=-inf;
49     if(l>r)
50         return;
51     idx=d;
52     std::nth_element(a+l,a+mid,a+r+1);
53     the[id]=a[mid];
54     rg[id][0][0]=rg[id][0][1]=the[id].x[0];
55     rg[id][1][0]=rg[id][1][1]=the[id].x[1];
56     make(lc,d^1);
57     make(rc,d^1);
58
59     rg[id][0][0]=std::min(rg[id][0][0],std::min(rg[lson][0][0],rg[
          rson][0][0]));
60     rg[id][1][0]=std::min(rg[id][1][0],std::min(rg[lson][1][0],rg[
          rson][1][0]));
61
62     rg[id][0][1]=std::max(rg[id][0][1],std::max(rg[lson][0][1],rg[
          rson][0][1]));
63     rg[id][1][1]=std::max(rg[id][1][1],std::max(rg[lson][1][1],rg[
          rson][1][1]));
```

```
64 }
65
66 inline long long cal(int id)
67 {
68     static long long a[2];
69     static int i;
70     for(i=0;i<2;++i)
71         a[i]=std::max(abs(p.x[i]-rg[id][i][0]),abs(p.x[i]-rg[id][i
           ][1]));
72     return sqr(a[0])+sqr(a[1]);
73 }
74
75 std::priority_queue<pli>ans;
76
77 void query(const int id=1,const int d=0)
78 {
79     if(the[id].lb<0)
80         return;
81     pli tmp(the[id].dist(p));
82     int a(lson),b(rson);
83     if(p.x[d]<=the[id].x[d])
84         std::swap(a,b);
85     if(ans.size()<m)
86         ans.push(tmp);
87     else
88         if(tmp<ans.top())
89         {
90             ans.push(tmp);
91             ans.pop();
92         }
93     if(ans.size()<m || cal(a)>=-ans.top().first)
94         query(a,d^1);
95     if(ans.size()<m || cal(b)>=-ans.top().first)
96         query(b,d^1);
97 }
98
99 int q,i,j,k;
100
101 int main()
102 {
103     scanf("%d",&n);
104     for(i=1;i<=n;++i)
105     {
106         scanf("%lld␣%lld",&a[i].x[0],&a[i].x[1]);
107         a[i].lb=i;
108     }
109     make();
110     scanf("%d",&q);
111     while(q--)
112     {
113         scanf("%lld␣%lld",&p.x[0],&p.x[1]);
```

```
114         scanf("%d",&m);
115         while(!ans.empty())
116             ans.pop();
117         query();
118         printf("%d\n",ans.top().second);
119     }
120     return 0;
121 }
```

2.9 Manhattan MST

```
  1 #include<iostream>
  2 #include<cstdio>
  3 #include<cstring>
  4 #include<queue>
  5 #include<cmath>
  6 using namespace std;
  7 const int srange = 10000000;       //坐标范围
  8 const int ra = 131072;     //线段树常量
  9 int c[ ra * 2 ], d[ ra * 2 ];      //线段树
 10 int a[ 100000 ], b[ 100000 ];   //排序临时变量
 11 int order[ 400000 ], torder[ 100000 ]; //排序结果
 12 int Index[ 100000 ];     //排序结果取反（为了在常数时间内取得某数的位置）
 13 int road[ 100000 ][ 8 ];    //每个点连接出去的条边8
 14 int y[ 100000 ], x[ 100000 ];      //点坐标
 15 int n;          //点个数
 16
 17 int swap( int &a, int &b )    //交换两个数
 18 {
 19     int t = a; a = b; b = t;
 20 }
 21
 22 int insert( int a, int b, int i ) //向线段树中插入一个数
 23 {
 24     a += ra;
 25     while ( a != 0 )
 26     {
 27         if ( c[ a ] > b )
 28         {
 29             c[ a ] = b;
 30             d[ a ] = i;
 31         }
 32         else break;
 33         a >>= 1;
 34     }
 35 }
 36
 37 int find( int a )      //从c[0..a中找最小的数，线段树查询]
 38 {
 39     a += ra;
 40     int ret = d[ a ], max = c[ a ];
 41     while ( a > 1 )
```

```
42      {
43          if ( ( a & 1 ) == 1 )
44              if ( c[ --a ] < max )
45              {
46                  max = c[ a ];
47                  ret = d[ a ];
48              }
49          a >>= 1;
50      }
51      return ret;
52  }
53
54  int ta[ 65536 ], tb[ 100000 ];    //基数排序临时变量
55
56  int radixsort( int *p )     //基数排序，以为基准p
57  {
58      memset( ta, 0, sizeof( ta ) );
59      for (int i = 0; i < n; i++ ) ta[ p[ i ] & 0xffff ]++;
60      for (int i = 0; i < 65535; i++ ) ta[ i + 1 ] += ta[ i ];
61      for (int i = n - 1; i >= 0; i-- ) tb[ --ta[ p[ order[ i ] ] & 0
          xffff ] ] = order[ i ];
62      memmove( order, tb, n * sizeof( int ) );
63      memset( ta, 0, sizeof( ta ) );
64      for (int i = 0; i < n; i++ ) ta[ p[ i ] >> 16 ]++;
65      for (int i = 0; i < 65535; i++ ) ta[ i + 1 ] += ta[ i ];
66      for (int i = n - 1; i >= 0; i-- ) tb[ --ta[ p[ order[ i ] ] >>
          16 ] ] = order[ i ];
67      memmove( order, tb, n * sizeof( int ) );
68  }
69
70  int work( int ii )                  //求每个点在一个方向上最近的点
71  {
72      for (int i = 0; i < n; i++ ) //排序前的准备工作
73      {
74          a[ i ] = y[ i ] - x[ i ] + srange;
75          b[ i ] = srange - y[ i ];
76          order[ i ] = i;
77      }
78      radixsort( b );       //排序
79      radixsort( a );
80      for (int i = 0; i < n; i++ )
81      {
82          torder[ i ] = order[ i ];
83          order[ i ] = i;
84      }
85      radixsort( a );       //为线段树而做的排序
86      radixsort( b );
87      for (int i = 0; i < n; i++ )
88      {
89          Index[ order[ i ] ] = i; //取反，求orderIndex
90      }
```

```
91      for (int i = 1; i < ra + n; i++ ) c[ i ] = 0x7fffffff; //线段树
            初始化
92      memset( d, 0xff, sizeof( d ) );
93      for (int i = 0; i < n; i++ ) //线段树插入删除调用
94      {
95          int tt = torder[ i ];
96          road[ tt ][ ii ] = find( Index[ tt ] );
97          insert( Index[ tt ], y[ tt ] + x[ tt ], tt );
98      }
99  }
100
101 int distanc( int a, int b )        //求两点的距离，之所以少一个是因为编译
        器不让使用作为函数名edistance
102 {
103     return abs( x[ a ] - x[ b ] ) + abs( y[ a ] - y[ b ] );
104 }
105
106 int ttb[ 400000 ];       //边排序的临时变量
107 int rx[ 400000 ], ry[ 400000 ], rd[ 400000 ]; //边的存储
108 int rr = 0;
109
110 int radixsort_2( int *p )    //还是基数排序，copy+的产物paste
111 {
112     memset( ta, 0, sizeof( ta ) );
113     for (int i = 0; i < rr; i++ ) ta[ p[ i ] & 0xffff ]++;
114     for (int i = 0; i < 65535; i++ ) ta[ i + 1 ] += ta[ i ];
115     for (int i = rr - 1; i >= 0; i-- ) ttb[ --ta[ p[ order[ i ] ] &
            0xffff ] ] = order[ i ];
116     memmove( order, ttb, rr * sizeof( int ) );
117     memset( ta, 0, sizeof( ta ) );
118     for (int i = 0; i < rr; i++ ) ta[ p[ i ] >> 16 ]++;
119     for (int i = 0; i < 65535; i++ ) ta[ i + 1 ] += ta[ i ];
120     for (int i = rr - 1; i >= 0; i-- ) ttb[ --ta[ p[ order[ i ] ]
           >> 16 ] ] = order[ i ];
121     memmove( order, ttb, rr * sizeof( int ) );
122 }
123
124 int father[ 100000 ], rank[ 100000 ];    //并查集
125 int findfather( int x )                  //并查集寻找代表元
126 {
127     if ( father[ x ] != -1 )
128         return ( father[ x ] = findfather( father[ x ] ) );
129     else return x;
130 }
131
132 long long kruskal()                      //最小生成树
133 {
134     rr = 0;
135     int tot = 0;
136     long long ans = 0;
137     for (int i = 0; i < n; i++ )         //得到边表
```

```
138     {
139         for (int j = 0; j < 4; j++ )
140         {
141             if ( road[ i ][ j ] != -1 )
142             {
143                 rx[ rr ] = i;
144                 ry[ rr ] = road[ i ][ j ];
145                 rd[ rr++ ] = distanc( i, road[ i ][ j ] );
146             }
147         }
148     }
149     for (int i = 0; i < rr; i++ ) order[ i ] = i; //排序
150     radixsort_2( rd );
151     memset( father, 0xff, sizeof( father ) ); //并查集初始化
152     memset( rank, 0, sizeof( rank ) );
153     for (int i = 0; i < rr; i++ )     //最小生成树标准算法kruskal
154     {
155         if ( tot == n - 1 ) break;
156         int t = order[ i ];
157         int x = findfather( rx[ t ] ), y = findfather( ry[ t ] );
158         if ( x != y )
159         {
160             ans += rd[ t ];
161             tot++;
162             int &rkx = rank[ x ], &rky = rank[ y ];
163             if ( rkx > rky ) father[ y ] = x;
164             else
165             {
166                 father[ x ] = y;
167                 if ( rkx == rky ) rky++;
168             }
169         }
170     }
171     return ans;
172 }
173
174 int casenum = 0;
175
176 int main()
177 {
178     while ( cin >> n )
179     {
180         if ( n == 0 ) break;
181         for (int i = 0; i < n; i++ )
182             scanf( "%d␣%d", &x[ i ], &y[ i ] );
183         memset( road, 0xff, sizeof( road ) );
184         for (int i = 0; i < 4; i++ )              //为了减少编程复杂度，
              work()函数只写了一种，其他情况用转换坐标的方式类似处理
185         {           //为了降低算法复杂度，只求出个方向的边4
186             if ( i == 2 )
187             {
```

```
                for (int j = 0; j < n; j++ ) swap( x[ j ], y[ j ] )
                    ;
            }
            if ( ( i & 1 ) == 1 )
            {
                for (int j = 0; j < n; j++ ) x[ j ] = srange - x[ j
                    ];
            }
            work( i );
        }
        printf( "Case %d: Total Weight = ", ++casenum );
        cout << kruskal() << endl;
    }
    return 0;
}
```

2.10 rotating caliper

```
//最远点对

inline double go()
{
    l=ans=0;
    for(i=0;i<n;++i)
    {
        tl=pnt[(i+1)%n]-pnt[i];
        while(abs(tl.cross(pnt[(l+1)%n]-pnt[i]))>=abs(tl.cross(pnt[
            l]-pnt[i])))
            l=(l+1)%n;
        ans=std::max(ans,std::max(dist(pnt[l],pnt[i]),dist(pnt[l],
            pnt[(i+1)%n])));
    }
    return ans;
}

//两凸包最近距离
double go()
{
    sq=sp=0;
    for(i=1;i<ch[1].size();++i)
        if(ch[1][sq]<ch[1][i])
            sq=i;
    tp=sp;
    tq=sq;
    ans=(ch[0][sp]-ch[1][sq]).len();
    do
    {
        a1=ch[0][sp];
        a2=ch[0][(sp+1)%ch[0].size()];
        b1=ch[1][sq];
        b2=ch[1][(sq+1)%ch[1].size()];
        tpv=b1-(b2-a1);
```

```
33          tpv.x = b1.x - (b2.x - a1.x);
34          tpv.y = b1.y - (b2.y - a1.y);
35          len=(tpv-a1).cross(a2-a1);
36          if(fabs(len)<eps)
37          {
38              ans=std::min(ans,p2l(a1,b1,b2));
39              ans=std::min(ans,p2l(a2,b1,b2));
40              ans=std::min(ans,p2l(b1,a1,a2));
41              ans=std::min(ans,p2l(b2,a1,a2));
42              sp=(sp+1)%ch[0].size();
43              sq=(sq+1)%ch[1].size();
44          }
45          else
46              if(len<-eps)
47              {
48                  ans=std::min(ans,p2l(b1,a1,a2));
49                  sp=(sp+1)%ch[0].size();
50              }
51              else
52              {
53                  ans=std::min(ans,p2l(a1,b1,b2));
54                  sq=(sq+1)%ch[1].size();
55              }
56      }while(tp!=sp || tq!=sq);
57      return ans;
58  }
59
60  //外接矩形 by mzry
61  inline void solve()
62  {
63      resa = resb = 1e100;
64      double dis1,dis2;
65      Point xp[4];
66      Line l[4];
67      int a,b,c,d;
68      int sa,sb,sc,sd;
69      a = b = c = d = 0;
70      sa = sb = sc = sd = 0;
71      Point va,vb,vc,vd;
72      for (a = 0; a < n; a++)
73      {
74          va = Point(p[a],p[(a+1)%n]);
75          vc = Point(-va.x,-va.y);
76          vb = Point(-va.y,va.x);
77          vd = Point(-vb.x,-vb.y);
78          if (sb < sa)
79          {
80              b = a;
81              sb = sa;
82          }
83          while (xmult(vb,Point(p[b],p[(b+1)%n])) < 0)
```

```
84          {
85              b = (b+1)%n;
86              sb++;
87          }
88          if (sc < sb)
89          {
90              c = b;
91              sc = sb;
92          }
93          while (xmult(vc,Point(p[c],p[(c+1)%n])) < 0)
94          {
95              c = (c+1)%n;
96              sc++;
97          }
98          if (sd < sc)
99          {
100             d = c;
101             sd = sc;
102         }
103         while (xmult(vd,Point(p[d],p[(d+1)%n])) < 0)
104         {
105             d = (d+1)%n;
106             sd++;
107         }
108
109         //卡在 p[a],p[b],p[c],p[d] 上
110         sa++;
111     }
112 }
113
114 //合并凸包给定凸多边形
115  P = { p(1) , ... , p(m) } 和 Q = { q(1) , ... , q(n) ，一个点
        对} (p(i), q(j)) 形成 P 和 Q 之间的桥当且仅当：
116
117 (p(i), q(j)) 形成一个并踵点对。
118 p(i−1), p(i+1), q(j−1), q(j+1) 都位于由 (p(i), q(j)) 组成的线的同一侧。
    假设多边形以标准形式给出并且顶点是以顺时针序排列，算法如下：、分别计算
119
120
121
122 1 P 和 Q 拥有最大 y 坐标的顶点。如果存在不止一个这样的点，取    x 坐标最大的。、
    构造这些点的遂平切线，
123 2 以多边形处于其右侧为正方向（因此他们指向 x 轴正方向）。、同时顺时针旋转两条切
    线直到其中一条与边相交。
124 3 得到一个新的并踵点对 (p(i), q(j)) 。对于平行边的情况，得到三个并踵点对。、
    对于所有有效的并踵点对
125 4 (p(i), q(j)): 判定 p(i−1), p(i+1), q(j−1), q(j+1) 是否都位于连接
    点 (p(i), q(j)) 形成的线的同一侧。如果是，这个并踵点对就形成了一个桥，并标
    记他。、重复执行步骤和步骤直到切线回到他们原来的位置。
126 534、所有可能的桥此时都已经确定了。
127 6 通过连续连接桥间对应的凸包链来构造合并凸包。上述的结论确定了算法的正确性。运行
```

时间受步骤，，约束。

156 他们都为 O(N) 运行时间（N 是顶点总数）。因此算法拥有现行的时间复杂度。一个凸多边形间的桥实际上确定了另一个有用的概念：多边形间公切线。同时，桥也是计算凸多边形交的算法核心。

//临界切线、计算
1 P 上 y 坐标值最小的顶点（称为 yminP ）和 Q 上 y 坐标值最大的顶点（称为）。ymaxQ、为多边形在
2 yminP 和 ymaxQ 处构造两条切线 LP 和 LQ 使得他们对应的多边形位于他们的右侧。此时 LP 和 LQ 拥有不同的方向，并且 yminP 和 ymaxQ 成为了多边形间的一个对踵点对。、令
3 p(i)= ，yminP q(j)= 。ymaxQ (p(i)，q(j)) 构成了多边形间的一个对踵点对。检测是否有 p(i−1),p(i+1) 在线 (p(i)，q(j)) 的一侧，并且 q(j−1),q(j+1) 在另一侧。如果成立， (p(i)，q(j)) 确定了一条线。CS、旋转这两条线，
4 直到其中一条和其对应的多边形的边重合。、一个新的对踵点对确定了。
5 如果两条线都与边重合，总共三对对踵点对（原先的顶点和新的顶点的组合）需要考虑。对于所有的对踵点对，执行上面的测试。、重复执行步骤和步骤，
645 直到新的点对为(yminP,ymaxQ)。、输出
7线。CS

//最小最大周长面积外接矩形//、计算全部四个多边形的端点，
1 称之为， xminP ，xmaxP ，yminP 。ymaxP、通过四个点构造
2 P 的四条切线。他们确定了两个“卡壳”集合。、如果一条（或两条）线与一条边重合，
3 那么计算由四条线决定的矩形的面积，并且保存为当前最小值。否则将当前最小值定义为无穷大。、顺时针旋转线直到其中一条和多边形的一条边重合。
4、计算新矩形的周长面积，
5/ 并且和当前最小值比较。如果小于当前最小值则更新，并保存确定最小值的矩形信息。、重复步骤和步骤，
645 直到线旋转过的角度大于度。90、输出外接矩形的最小周长。
7

2.11 shit

```
struct pv
{
    double x,y;
    pv(double a=0,double b=0):x(a),y(b){}
    inline pv operator+(const pv &i)const
    {
        return pv(x+i.x,y+i.y);
    }
    inline pv operator−(const pv &i)const
    {
        return pv(x−i.x,y−i.y);
    }
    inline bool operator ==(const pv &i)const
    {
        return fabs(x−i.x)<eps && fabs(y−i.y)<eps;
```

```
16      }
17      inline bool operator<(const pv &i)const
18      {
19          return y==i.y?x<i.x:y<i.y;
20      }
21      inline double cross(const pv &i)const
22      {
23          return x*i.y-y*i.x;
24      }
25      inline double dot(const pv &i)const
26      {
27          return x*i.x+y*i.y;
28      }
29      inline double len()
30      {
31          return hypot(x,y);
32      }
33      inline pv rotate(pv p,double theta)
34      {
35          static pv v;
36          v=*this-p;
37          static double c,s;
38          c=cos(theta);
39          s=sin(theta);
40          return pv(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
41      }
42  };
43
44  pv rotate(pv v,pv p,double theta,double sc=1) // rotate vector v,
        theta in [0,2pi]
45  {
46      static pv re;
47      re=p;
48      v=v-p;
49      p.x=sc*cos(theta);
50      p.y=sc*sin(theta);
51      re.x+=v.x*p.x-v.y*p.y;
52      re.y+=v.x*p.y+v.y*p.x;
53      return re;
54  }
55
56  struct line
57  {
58      pv pnt[2];
59      line(double a,double b,double c) // a*x + b*y + c = 0
60      {
61  #define maxl 1e2 //preciseness should not be too high ( compare
        with eps )
62          if(fabs(b)>eps)
63          {
64              pnt[0]=pv(maxl,(c+a*maxl)/(-b));
```

```
65              pnt[1]=pv(-maxl,(c-a*maxl)/(-b));
66          }
67          else
68          {
69              pnt[0]=pv(-c/a,maxl);
70              pnt[1]=pv(-c/a,-maxl);
71          }
72  #undef maxl
73      }
74      pv cross(const line &v)const
75      {
76          double a=(v.pnt[1]-v.pnt[0]).cross(pnt[0]-v.pnt[0]);
77          double b=(v.pnt[1]-v.pnt[0]).cross(pnt[1]-v.pnt[0]);
78          return pv((pnt[0].x*b-pnt[1].x*a)/(b-a),(pnt[0].y*b-pnt[1].
                y*a)/(b-a));
79      }
80  };
81
82  inline std::pair<pv,double> getcircle(const pv &a,const pv &b,const
        pv &c)
83  {
84      static pv ct;
85      ct=line(2*(b.x-a.x),2*(b.y-a.y),a.len()-b.len()).cross(line(2*(
           c.x-b.x),2*(c.y-b.y),b.len()-c.len()));
86      return std::make_pair(ct,sqrt((ct-a).len()));
87  }
88
89  //sort with polar angle
90  inline bool cmp(const Point& a,const Point& b)
91  {
92      if (a.y*b.y <= 0)
93      {
94          if (a.y > 0 || b.y > 0)
95              return a.y < b.y;
96          if (a.y == 0 && b.y == 0)
97              return a.x < b.x;
98      }
99      return a.cross(b) > 0;
100 }
101
102 //graham
103 inline bool com(const pv &a,const pv &b)
104 {
105     if(fabs(t=(a-pnt[0]).cross(b-pnt[0]))>eps)
106         return t>0;
107     return (a-pnt[0]).len()<(b-pnt[0]).len();
108 }
109
110 inline void graham(std::vector<pv> &ch,const int n)
111 {
112     std::nth_element(pnt,pnt,pnt+n);
```

```
113     std::sort(pnt+1,pnt+n,com);
114     ch.resize(0);
115     ch.push_back(pnt[0]);
116     ch.push_back(pnt[1]);
117     static int i;
118     for(i=2;i<n;++i)
119         if(fabs((pnt[i]-ch[0]).cross(ch[1]-ch[0]))>eps)
120         {
121             ch.push_back(pnt[i++]);
122             break;
123         }
124         else
125             ch.back()=pnt[i];
126     for(;i<n;++i)
127     {
128         while((ch.back()-ch[ch.size()-2]).cross(pnt[i]-ch[ch.size()
               -2])<eps)
129             ch.pop_back();
130         ch.push_back(pnt[i]);
131     }
132 }
```

2.12 other

2.12.1 Pick's theorem

给定顶点座标均是整点（或正方形格点）的简单多边形
A: 面积
i: 内部格点数目
b: 边上格点数目
$A = i + \frac{b}{2} - 1$

取格点的组成图形的面积为一单位。在平行四边形格点，皮克定理依然成立。套用于任意三角形格点，皮克定理则是
$A = 2 \times i + b - 2$

2.12.2 Triangle

Area:
$p = \frac{a+b+c}{2}$
$area = \sqrt{p \times (p-a) \times (p-b) \times (p-c)}$
$area = \frac{a \times b \times \sin(\angle C)}{2}$
$area = \frac{a^2 \times \sin(\angle B) \times \sin(\angle C)}{2 \times \sin(\angle B + \angle C)}$
$area = \frac{a^2}{2 \times (\cot(\angle B) + \cot(\angle C))}$

centroid:
center of mass
intersection of triangle's three triangle medians

Trigonometric conditions:
$\tan\frac{\alpha}{2}\tan\frac{\beta}{2} + \tan\frac{\beta}{2}\tan\frac{\gamma}{2} + \tan\frac{\gamma}{2}\tan\frac{\alpha}{2} = 1$

$\sin^2\frac{\alpha}{2}+\sin^2\frac{\beta}{2}+\sin^2\frac{\gamma}{2}+2\sin\frac{\alpha}{2}\sin\frac{\beta}{2}\sin\frac{\gamma}{2}=1$

Circumscribed circle:
$diameter=\frac{abc}{2\cdot\text{area}}=\frac{|AB||BC||CA|}{2|\Delta ABC|}$

$=\frac{abc}{2\sqrt{s(s-a)(s-b)(s-c)}}$
$=\frac{2abc}{\sqrt{(a+b+c)(-a+b+c)(a-b+c)(a+b-c)}}$
$diameter=\sqrt{\frac{2\cdot\text{area}}{\sin A\sin B\sin C}}$
$diameter=\frac{a}{\sin A}=\frac{b}{\sin B}=\frac{c}{\sin C}$

Incircle:
$inradius=\frac{2\times area}{a+b+c}$
coordinates(x,y)=$\left(\frac{ax_a+bx_b+cx_c}{a+b+c},\frac{ay_a+by_b+cy_c}{a+b+c}\right)=\frac{a}{a+b+c}(x_a,y_a)+\frac{b}{a+b+c}(x_b,y_b)+\frac{c}{a+b+c}(x_c,y_c)$

Excircles:
radius[a]=$\frac{2\times area}{b+c-a}$
radius[b]=$\frac{2\times area}{a+c-b}$
radius[c]=$\frac{2\times area}{a+b-c}$

Steiner circumellipse (least area circumscribed ellipse)
area=$\Delta\times\frac{4\pi}{3\sqrt{3}}$
center is the triangle's centroid.

Steiner inellipse (maximum area inellipse)
area=$\Delta\times\frac{\pi}{3\sqrt{3}}$
center is the triangle's centroid.

Fermat Point:

1. 当有一个内角不小于 120° 时，费马点为此角对应顶点。
2. 当三角形的内角都小于 120°
   (a) 以三角形的每一边为底边，向外做三个正三角形 ΔABC'，ΔBCA'，ΔCAB'。
   (b) 连接 CC'、BB'、AA'，则三条线段的交点就是所求的点。

2.12.3 Ellipse

$\frac{(x-h)^2}{a^2}+\frac{(y-k)^2}{b^2}=1$

$x=h+a\times\cos(t)$
$y=k+b\times\sin(t)$

area=$\pi\times a\times b$
distance from center to focus: $f=\sqrt{a^2-b^2}$
eccentricity: $e=\sqrt{a-\frac{b}{a}^2}=\frac{f}{a}$
focal parameter: $\frac{b^2}{\sqrt{a^2-b^2}}=\frac{b^2}{f}$

```cpp
inline double circumference(double a,double b) // accuracy: pow
   (0.5,53);
{
    static double digits=53;
    static double tol=sqrt(pow(0.5,digits));
    double x=a;
    double y=b;
    if(x<y)
        std::swap(x,y);
    if(digits*y<tol*x)
        return 4*x;
    double s=0,m=1;
    while(x>(tol+1)*y)
    {
        double tx=x;
        double ty=y;
        x=0.5f*(tx+ty);
        y=sqrt(tx*ty);
        m*=2;
        s+=m*pow(x-y,2);
    }
    return pi*(pow(a+b,2)-s)/(x+y);
}
```

2.12.4 about double

如果 sqrt(a), asin(a), acos(a) 中的 a 是你自己算出来并传进来的，那就得小心了。如果 a 本来应该是 0 的，由于浮点误差，可能实际是一个绝对值很小的负数（比如 -1^{-12}），这样 sqrt(a) 应得 0 的，直接因 a 不在定义域而出错。类似地，如果 a 本来应该是 ± 1, 则 asin(a)、acos(a) 也有可能出错。因此，对于此种函数，必需事先对 a 进行校正。

现在考虑一种情况，题目要求输出保留两位小数。有个 case 的正确答案的精确值是 0.005, 按理应该输出 0.01，但你的结果可能是 0.005000000001(恭喜)，也有可能是 0.004999999999(悲剧)，如果按照 printf("%.2lf", a) 输出，那你的遭遇将和括号里的字相同。
如果 a 为正，则输出 a + eps, 否则输出 a - eps。

不要输出 -0.000

注意 double 的数据范围

$a = b$	fabs(a-b)<eps
$a \neq b$	fabs(a-b)>eps
$a < b$	a+eps<b
$a \leq b$	a<b+eps
$a > b$	a>b+eps
$a \geq b$	a+eps>b

2.12.5 trigonometric functions

	input	output
sin	radian	$[-1,+1]$
cos	radian	$[-1,+1]$
tan	radian	$(-\infty,+\infty)$
asin	$[-1,+1]$	$[-\frac{\pi}{2},+\frac{\pi}{2}]$
acos	$[-1,+1]$	[0,π]
atan	$(-\infty,+\infty)$	$[-\frac{\pi}{2},+\frac{\pi}{2}]$
atan2	(y,x)	$\tan(\frac{y}{x}) \in [-\pi,+\pi]$ (watch out if x=y=0)

exp	x^e
log	ln
log10	log_{10}
ceil	smallest interger $\geq$ x (watch out x<0
floor	greatest interger $\leq$ x (watch out x<0
trunc	nearest integral value close to 0
nearybyint	round to intergral, up to fegetround
round	round with halfway cases rounded away from zero

2.12.6 round

1. cpp: 四舍六入五留双
   (a) 当尾数小于或等于 4 时，直接将尾数舍去
   (b) 当尾数大于或等于 6 时，将尾数舍去并向前一位进位
   (c) 当尾数为 5，而尾数后面的数字均为 0 时，应看尾数“5”的前一位：若前一位数字此时为奇数，就应向前进一位；若前一位数字此时为偶数，则应将尾数舍去。数字“0”在此时应被视为偶数
   (d) 当尾数为 5，而尾数“5”的后面还有任何不是 0 的数字时，无论前一位在此时为奇数还是偶数，也无论“5”后面不为 0 的数字在哪一位上，都应向前进一位
2. java: add 0.5,then floor

2.12.7 rotation matrix

original matrix:

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

3-dimension:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

rotation by unit vector $v = (x, y, z)$:
$$\begin{bmatrix} \cos\theta + (1-\cos\theta)x^2 & (1-\cos\theta)xy - (\sin\theta)z & (1-\cos\theta)xz + (\sin\theta)y \\ (1-\cos\theta)yx + (\sin\theta)z & \cos\theta + (1-\cos\theta)y^2 & (1-\cos\theta)yz - (\sin\theta)x \\ (1-\cos\theta)zx - (\sin\theta)y & (1-\cos\theta)zy + (\sin\theta)x & \cos\theta + (1-\cos\theta)z^2 \end{bmatrix}$$
we use transform matrix muliply our original matrix

and we can presetation a transformation as a 4×4 matrix:
$$\begin{bmatrix} a_{11} & a_{12} & a_{12} & a_{14} \\ a_{21} & a_{22} & a_{22} & a_{24} \\ a_{31} & a_{32} & a_{32} & a_{34} \\ a_{41} & a_{42} & a_{42} & a_{44} \end{bmatrix}$$
Matrix $\begin{bmatrix} a_{11} & a_{12} & a_{12} \\ a_{21} & a_{22} & a_{22} \\ a_{31} & a_{32} & a_{32} \end{bmatrix}$ presetation the transformation as same as 3×3 matrx.
Matrix $\begin{bmatrix} a_{14} \\ a_{24} \\ a_{34} \end{bmatrix}$ as translation.
Matrix $\begin{bmatrix} a_{41} & a_{42} & a_{43} \end{bmatrix}$ as projection.
Matrix $\begin{bmatrix} a_{44} \end{bmatrix}$ as scale.

original Matrix:
$$\begin{bmatrix} x \\ y \\ z \\ Scale \end{bmatrix}$$

3 Geometry/tmp

3.1 test

```
//三角形:
//1. 半周长 P = (a+b+c)/2
//2. 面积 S = aH/2 = ab*sin(C)/2 = sqrt(P*(P-a)*(P-b)*(P-c))
//3. 中线 Ma = sqrt(2(b^2+c^2)-a^2)/2 = sqrt(b^2+c^2+2bc*cos(A))/2
//4. 角平分线 Ta = sqrt(bc((b+c)^2-a^2))/(b+c) = 2bc*cos(A/2)/(b+c)
//5. 高线 Ha = b*sin(C) = c*sin(B) = sqrt(b^2 - ((a^2+b^2-c^2)/2a)^2)
//6. 内切圆半径 r = S/P = arcsin(B/2)*sin(C/2)/sin((B+C)/2) = 4R*sin(A/2)*sin(B/2)*sin(C/2) = sqrt((P-a)(P-b)(P-c)/P) =
    P*tan(A/2)*tan(B/2)*tan(C/2)
//7. 外接圆半径 R = abc/4S = a/2sin(A) = b/2sin(B) = c/2sin(C)
//四边形:
//D1,D2 为对角线,M 对角线中点连线,A 为对角线夹角
//1. a^2+b^2+c^2+d^2 = D1^2+D2^2+4M^2
//2. S = D1*D2*sin(A)/2
//(以下对圆的内接四边形)
//3. ac+bd = D1*D2
//4. S = sqrt((P-a)(P-b)(P-c)(P-d)),P 为半周长
//正 n 边形:
```

```
//R 为外接圆半径,r 为内切圆半径
//1. 中心角 A = 2π/n
//2. 内角 C = (n-2)π/n
//3. 边长 a = 2√(R^2 - r^2) = 2R sin(A/2) = 2r tan(A/2)
//4. 面积 S = nar/2 = nr^2 tan(A/2) = nR^2 sin(A)/2 = na^2/(4 tan(A/2))
//圆:
//1. 弧长 l = rA
//2. 弦长 a = 2√(2hr - h^2) = 2r sin(A/2)
//3. 弓形高 h = r - √(r^2 - a^2/4) = r(1 - cos(A/2)) = arctan(A/4)/2
//4. 扇形面积 S1 = rl/2 = r^2 A/2
//5. 弓形面积 S2 = rl-a(r-h)/2 = r^2(A-sin(A))/2
//棱柱:
//1. 体积 V = Ah,A 为底面积,h 为高
//2. 侧面积 S = lp,l 为棱长,p 为直截面周长
//3. 全面积 T = S + 2A
//棱锥:
//1. 体积 V = Ah/3,A 为底面积,h 为高
//(以下对正棱锥)
//2. 侧面积 S = lp/2,l 为斜高,p 为底面周长
//3. 全面积 T = S + A
//棱台:
//1. 体积 V = (A1 + A2 + √(A1A2))h/3,A1.A2 为上下底面积,h 为高
//(以下为正棱台)
//2. 侧面积 S = (p1+p2)l/2,p1.p2 为上下底面周长,l 为斜高
//3. 全面积 T = S + A1 + A2
//圆柱:
//1. 侧面积 S = 2πrh
//2. 全面积 T = 2πr(h + r)
//3. 体积 V = πr^2 h
//圆锥:
//1. 斜高 l = √(h^2 + r^2)
//2. 侧面积 S = πrl
//3. 全面积 T = πr(l + r)
//4. 体积 V = πr^2 h/3
//圆台:
//1. 母线 l = √(h^2 + (r1 - r2)^2)
//2. 侧面积 S = π(r1 + r2)l
//3. 全面积 T = πr1(l + r1) + πr2(l + r2)
//4. 体积 V = π(r1^2 + r2^2 + r1r2)h/3
//球:
//1. 全面积 T = 4πr^2
//2. 体积 V = πr^3 4/3
//球台:
//1. 侧面积 S = 2πrh
//2. 全面积 T = π(2rh + r1^2 + r2^2)
//3. 体积 V=1/6πh(3(r1^2 + r2^2) + h^2)
//球扇形:
```

```
//1. 全面积 T = πr(2h + r0),h 为球冠高,r0 为球冠底面半径
//2. 体积 V = (2/3)πr^2h

//polygon
#include <stdlib.h>
#include <math.h>
#define MAXN 1000
#define offset 10000
#define eps 1e-8
#define zero(x) (((x)>0?(x):-(x))<eps)
#define _sign(x) ((x)>eps?1:((x)<-eps?2:0))
struct point{double x,y;};
struct line{point a,b;};
double xmult(point p1,point p2,point p0)
{
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
//判定凸多边形，顶点按顺时针或逆时针给出，允许相邻边共线
int is_convex(int n,point* p)
{
    int i,s[3]={1,1,1};
    for (i=0;i<n&&s[1]|s[2];i++)
        s[_sign(xmult(p[(i+1)%n],p[(i+2)%n],p[i]))]=0;
    return s[1]|s[2];
}
//判定凸多边形，顶点按顺时针或逆时针给出，不允许相邻边共线
int is_convex_v2(int n,point* p)
{
    int i,s[3]={1,1,1};
    for (i=0;i<n&&s[0]&&s[1]|s[2];i++)
        s[_sign(xmult(p[(i+1)%n],p[(i+2)%n],p[i]))]=0;
    return s[0]&&s[1]|s[2];
}
//判点在凸多边形内或多边形边上，顶点按顺时针或逆时针给出
int inside_convex(point q,int n,point* p)
{
    int i,s[3]={1,1,1};
    for (i=0;i<n&&s[1]|s[2];i++)
        s[_sign(xmult(p[(i+1)%n],q,p[i]))]=0;
    return s[1]|s[2];
}
//判点在凸多边形内，顶点按顺时针或逆时针给出，在多边形边上返回 0
int inside_convex_v2(point q,int n,point* p)
{
    int i,s[3]={1,1,1};
    for (i=0;i<n&&s[0]&&s[1]|s[2];i++)
        s[_sign(xmult(p[(i+1)%n],q,p[i]))]=0;
    return s[0]&&s[1]|s[2];
}
//判点在任意多边形内，顶点按顺时针或逆时针给出
//on_edge 表示点在多边形边上时的返回值,offset 为多边形坐标上限
```

```
115 int inside_polygon(point q,int n,point* p,int on_edge=1)
116 {
117     point q2;
118     int i=0,count;
119     while (i<n)
120         for (count=i=0,q2.x=rand()+offset,q2.y=rand()+offset;i<n;i
               ++)
121             if
122                 (zero(xmult(q,p[i],p[(i+1)%n]))&&(p[i].x-q.x)*(p[(i
                    +1)%n].x-q.x)<eps&&(p[i].y-q.y)*(p[(i+1)%n].y-q.
                    y)<eps)
123                     return on_edge;
124             else if (zero(xmult(q,q2,p[i])))
125                 break;
126             else if
127                 (xmult(q,p[i],q2)*xmult(q,p[(i+1)%n],q2)<-eps&&
                    xmult(p[i],q,p[(i+1)%n])*xmult(p[i],q2,p[(i+1)%n
                    ])<-eps)
128                     count++;
129     return count&1;
130 }
131 inline int opposite_side(point p1,point p2,point l1,point l2)
132 {
133     return xmult(l1,p1,l2)*xmult(l1,p2,l2)<-eps;
134 }
135 inline int dot_online_in(point p,point l1,point l2)
136 {
137     return zero(xmult(p,l1,l2))&&(l1.x-p.x)*(l2.x-p.x)<eps&&(l1.y-p
           .y)*(l2.y-p.y)<eps;
138 }
139 //判线段在任意多边形内，顶点按顺时针或逆时针给出，与边界相交返回 1
140 int inside_polygon(point l1,point l2,int n,point* p)
141 {
142     point t[MAXN],tt;
143     int i,j,k=0;
144     if (!inside_polygon(l1,n,p)||!inside_polygon(l2,n,p))
145         return 0;
146     for (i=0;i<n;i++)
147         if (opposite_side(l1,l2,p[i],p[(i+1)%n])&&opposite_side(p[i
               ],p[(i+1)%n],l1,l2))
148             return 0;
149         else if (dot_online_in(l1,p[i],p[(i+1)%n]))
150             t[k++]=l1;
151         else if (dot_online_in(l2,p[i],p[(i+1)%n]))
152             t[k++]=l2;
153         else if (dot_online_in(p[i],l1,l2))
154             t[k++]=p[i];
155     for (i=0;i<k;i++)
156         for (j=i+1;j<k;j++)
157         {
158             tt.x=(t[i].x+t[j].x)/2;
```

```
159             tt.y=(t[i].y+t[j].y)/2;
160             if (!inside_polygon(tt,n,p))
161                 return 0;
162         }
163     return 1;
164 }
165 point intersection(line u,line v)
166 {
167     point ret=u.a;
168     double t=((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.
        x))
169         /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
170     ret.x+=(u.b.x-u.a.x)*t;
171     ret.y+=(u.b.y-u.a.y)*t;
172     return ret;
173 }
174 point barycenter(point a,point b,point c)
175 {
176     line u,v;
177     u.a.x=(a.x+b.x)/2;
178     u.a.y=(a.y+b.y)/2;
179     u.b=c;
180     v.a.x=(a.x+c.x)/2;
181     v.a.y=(a.y+c.y)/2;
182     v.b=b;
183     return intersection(u,v);
184 }
185 //多边形重心
186 point barycenter(int n,point* p)
187 {
188     point ret,t;
189     double t1=0,t2;
190     int i;
191     ret.x=ret.y=0;
192     for (i=1;i<n-1;i++)
193         if (fabs(t2=xmult(p[0],p[i],p[i+1]))>eps)
194         {
195             t=barycenter(p[0],p[i],p[i+1]);
196             ret.x+=t.x*t2;
197             ret.y+=t.y*t2;
198             t1+=t2;
199         }
200     if (fabs(t1)>eps)
201         ret.x/=t1,ret.y/=t1;
202     return ret;
203 }
204
205
206 //cut polygon
207 //多边形切割
208 //可用于半平面交
```

```
#define MAXN 100
#define eps 1e-8
#define zero(x) (((x)>0?(x):-(x))<eps)
struct point{double x,y;};
double xmult(point p1,point p2,point p0)
{
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
int same_side(point p1,point p2,point l1,point l2)
{
    return xmult(l1,p1,l2)*xmult(l1,p2,l2)>eps;
}
point intersection(point u1,point u2,point v1,point v2)
{
    point ret=u1;
    double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
        /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
    ret.x+=(u2.x-u1.x)*t;
    ret.y+=(u2.y-u1.y)*t;
    return ret;
}
//将多边形沿 l1,l2 确定的直线切割在 side 侧切割, 保证 l1,l2,side 不共线
void polygon_cut(int& n,point* p,point l1,point l2,point side)
{
    point pp[100];
    int m=0,i;
    for (i=0;i<n;i++)
    {
        if (same_side(p[i],side,l1,l2))
            pp[m++]=p[i];
        if
            (!same_side(p[i],p[(i+1)%n],l1,l2)&&!(zero(xmult(p[i],
               l1,l2))&&zero(xmult(p[(i+1)%n],l1,l2))))
                pp[m++]=intersection(p[i],p[(i+1)%n],l1,l2);
    }
    for (n=i=0;i<m;i++)
        if (!i||!zero(pp[i].x-pp[i-1].x)||!zero(pp[i].y-pp[i-1].y))
            p[n++]=pp[i];
    if (zero(p[n-1].x-p[0].x)&&zero(p[n-1].y-p[0].y))
        n--;
    if (n<3)
        n=0;
}

//float
//浮点几何函数库
#include <math.h>
#define eps 1e-8
#define zero(x) (((x)>0?(x):-(x))<eps)
struct point{double x,y;};
struct line{point a,b;};
```

```
259 //计算 cross product (P1-P0)x(P2-P0)
260 double xmult(point p1,point p2,point p0)
261 {
262     return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
263 }
264 double xmult(double x1,double y1,double x2,double y2,double x0,
       double y0)
265 {
266     return (x1-x0)*(y2-y0)-(x2-x0)*(y1-y0);
267 }
268 //计算 dot product (P1-P0).(P2-P0)
269 double dmult(point p1,point p2,point p0)
270 {
271     return (p1.x-p0.x)*(p2.x-p0.x)+(p1.y-p0.y)*(p2.y-p0.y);
272 }
273 double dmult(double x1,double y1,double x2,double y2,double x0,
       double y0)
274 {
275     return (x1-x0)*(x2-x0)+(y1-y0)*(y2-y0);
276 }
277 //两点距离
278 double distance(point p1,point p2)
279 {
280     return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
281 }
282 double distance(double x1,double y1,double x2,double y2)
283 {
284     return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
285 }
286 //判三点共线
287 int dots_inline(point p1,point p2,point p3)
288 {
289     return zero(xmult(p1,p2,p3));
290 }
291 int dots_inline(double x1,double y1,double x2,double y2,double x3,
       double y3)
292 {
293     return zero(xmult(x1,y1,x2,y2,x3,y3));
294 }
295 //判点是否在线段上, 包括端点
296 int dot_online_in(point p,line l)
297 {
298     return zero(xmult(p,l.a,l.b))&&(l.a.x-p.x)*(l.b.x-p.x)<eps&&(l.
          a.y-p.y)*(l.b.y-p.y)<eps;
299 }
300 int dot_online_in(point p,point l1,point l2)
301 {
302     return zero(xmult(p,l1,l2))&&(l1.x-p.x)*(l2.x-p.x)<eps&&(l1.y-p
          .y)*(l2.y-p.y)<eps;
303 }
304 int dot_online_in(double x,double y,double x1,double y1,double x2,
```

```
    double y2)
{
    return zero(xmult(x,y,x1,y1,x2,y2))&&(x1-x)*(x2-x)<eps&&(y1-y)
        *(y2-y)<eps;
}
//判点是否在线段上，不包括端点
int dot_online_ex(point p,line l)
{
    return
        dot_online_in(p,l)&&(!zero(p.x-l.a.x)||!zero(p.y-l.a.y))
            &&(!zero(p.x-l.b.x)||!zero(p.y-l.b.y));
}
int dot_online_ex(point p,point l1,point l2)
{
    return
        dot_online_in(p,l1,l2)&&(!zero(p.x-l1.x)||!zero(p.y-l1.y))
            &&(!zero(p.x-l2.x)||!zero(p.y-l2.y));
}
int dot_online_ex(double x,double y,double x1,double y1,double x2,
    double y2)
{
    return
        dot_online_in(x,y,x1,y1,x2,y2)&&(!zero(x-x1)||!zero(y-y1))
            &&(!zero(x-x2)||!zero(y-y2));
}
//判两点在线段同侧，点在线段上返回 0
int same_side(point p1,point p2,line l)
{
    return xmult(l.a,p1,l.b)*xmult(l.a,p2,l.b)>eps;
}
int same_side(point p1,point p2,point l1,point l2)
{
    return xmult(l1,p1,l2)*xmult(l1,p2,l2)>eps;
}
//判两点在线段异侧，点在线段上返回 0
int opposite_side(point p1,point p2,line l)
{
    return xmult(l.a,p1,l.b)*xmult(l.a,p2,l.b)<-eps;
}
int opposite_side(point p1,point p2,point l1,point l2)
{
    return xmult(l1,p1,l2)*xmult(l1,p2,l2)<-eps;
}
//判两直线平行
int parallel(line u,line v)
{
    return zero((u.a.x-u.b.x)*(v.a.y-v.b.y)-(v.a.x-v.b.x)*(u.a.y-u.
        b.y));
}
int parallel(point u1,point u2,point v1,point v2)
{
```

```
349     return zero((u1.x-u2.x)*(v1.y-v2.y)-(v1.x-v2.x)*(u1.y-u2.y));
350 }
351 //判两直线垂直
352 int perpendicular(line u,line v)
353 {
354     return zero((u.a.x-u.b.x)*(v.a.x-v.b.x)+(u.a.y-u.b.y)*(v.a.y-v.
        b.y));
355 }
356 int perpendicular(point u1,point u2,point v1,point v2)
357 {
358     return zero((u1.x-u2.x)*(v1.x-v2.x)+(u1.y-u2.y)*(v1.y-v2.y));
359 }
360 //判两线段相交，包括端点和部分重合
361 int intersect_in(line u,line v)
362 {
363     if (!dots_inline(u.a,u.b,v.a)||!dots_inline(u.a,u.b,v.b))
364         return !same_side(u.a,u.b,v)&&!same_side(v.a,v.b,u);
365     return dot_online_in(u.a,v)||dot_online_in(u.b,v)||
        dot_online_in(v.a,u)||dot_online_in(v.b,u);
366 }
367 int intersect_in(point u1,point u2,point v1,point v2)
368 {
369     if (!dots_inline(u1,u2,v1)||!dots_inline(u1,u2,v2))
370         return !same_side(u1,u2,v1,v2)&&!same_side(v1,v2,u1,u2);
371     return
372         dot_online_in(u1,v1,v2)||dot_online_in(u2,v1,v2)||
            dot_online_in(v1,u1,u2)||dot_online_in(v2,u1,u
373                 2);
374 }
375 //判两线段相交，不包括端点和部分重合
376 int intersect_ex(line u,line v)
377 {
378     return opposite_side(u.a,u.b,v)&&opposite_side(v.a,v.b,u);
379 }
380 int intersect_ex(point u1,point u2,point v1,point v2)
381 {
382     return opposite_side(u1,u2,v1,v2)&&opposite_side(v1,v2,u1,u2);
383 }
384 //计算两直线交点，注意事先判断直线是否平行!
385 //线段交点请另外判线段相交（同时还是要判断是否平行!）
386 point intersection(line u,line v)
387 {
388     point ret=u.a;
389     double t=((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.
        x))
390         /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
391     ret.x+=(u.b.x-u.a.x)*t;
392     ret.y+=(u.b.y-u.a.y)*t;
393     return ret;
394 }
395 point intersection(point u1,point u2,point v1,point v2)
```

```
{
    point ret=u1;
    double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
        /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
    ret.x+=(u2.x-u1.x)*t;
    ret.y+=(u2.y-u1.y)*t;
    return ret;
}
//点到直线上的最近点
point ptoline(point p,line l)
{
    point t=p;
    t.x+=l.a.y-l.b.y,t.y+=l.b.x-l.a.x;
    return intersection(p,t,l.a,l.b);
}
point ptoline(point p,point l1,point l2)
{
    point t=p;
    t.x+=l1.y-l2.y,t.y+=l2.x-l1.x;
    return intersection(p,t,l1,l2);
}
//点到直线距离
double disptoline(point p,line l)
{
    return fabs(xmult(p,l.a,l.b))/distance(l.a,l.b);
}
double disptoline(point p,point l1,point l2)
{
    return fabs(xmult(p,l1,l2))/distance(l1,l2);
}
double disptoline(double x,double y,double x1,double y1,double x2,
   double y2)
{
    return fabs(xmult(x,y,x1,y1,x2,y2))/distance(x1,y1,x2,y2);
}
//点到线段上的最近点
point ptoseg(point p,line l)
{
    point t=p;
    t.x+=l.a.y-l.b.y,t.y+=l.b.x-l.a.x;
    if (xmult(l.a,t,p)*xmult(l.b,t,p)>eps)
        return distance(p,l.a)<distance(p,l.b)?l.a:l.b;
    return intersection(p,t,l.a,l.b);
}
point ptoseg(point p,point l1,point l2)
{
    point t=p;
    t.x+=l1.y-l2.y,t.y+=l2.x-l1.x;
    if (xmult(l1,t,p)*xmult(l2,t,p)>eps)
        return distance(p,l1)<distance(p,l2)?l1:l2;
    return intersection(p,t,l1,l2);
```

```
446 }
447 //点到线段距离
448 double disptoseg(point p,line l)
449 {
450     point t=p;
451     t.x+=l.a.y-l.b.y,t.y+=l.b.x-l.a.x;
452     if (xmult(l.a,t,p)*xmult(l.b,t,p)>eps)
453         return distance(p,l.a)<distance(p,l.b)?distance(p,l.a):
              distance(p,l.b);
454     return fabs(xmult(p,l.a,l.b))/distance(l.a,l.b);
455 }
456 double disptoseg(point p,point l1,point l2)
457 {
458     point t=p;
459     t.x+=l1.y-l2.y,t.y+=l2.x-l1.x;
460     if (xmult(l1,t,p)*xmult(l2,t,p)>eps)
461         return distance(p,l1)<distance(p,l2)?distance(p,l1):
              distance(p,l2);
462     return fabs(xmult(p,l1,l2))/distance(l1,l2);
463 }
464 //矢量 V 以 P 为顶点逆时针旋转 angle 并放大 scale 倍
465 point rotate(point v,point p,double angle,double scale)
466 {
467     point ret=p;
468     v.x-=p.x,v.y-=p.y;
469     p.x=scale*cos(angle);
470     p.y=scale*sin(angle);
471     ret.x+=v.x*p.x-v.y*p.y;
472     ret.y+=v.x*p.y+v.y*p.x;
473     return ret;
474 }
475
476 //area
477 #include <math.h>
478 struct point{double x,y;};
479 //计算 cross product (P1-P0)x(P2-P0)
480 double xmult(point p1,point p2,point p0)
481 {
482     return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
483 }
484 double xmult(double x1,double y1,double x2,double y2,double x0,
      double y0)
485 {
486     return (x1-x0)*(y2-y0)-(x2-x0)*(y1-y0);
487 }
488 //计算三角形面积，输入三顶点
489 double area_triangle(point p1,point p2,point p3)
490 {
491     return fabs(xmult(p1,p2,p3))/2;
492 }
493 double area_triangle(double x1,double y1,double x2,double y2,double
```

```
    x3,double y3)
{
    return fabs(xmult(x1,y1,x2,y2,x3,y3))/2;
}
37
//计算三角形面积, 输入三边长
double area_triangle(double a,double b,double c)
{
    double s=(a+b+c)/2;
    return sqrt(s*(s-a)*(s-b)*(s-c));
}
//计算多边形面积, 顶点按顺时针或逆时针给出
double area_polygon(int n,point* p)
{
    double s1=0,s2=0;
    int i;
    for (i=0;i<n;i++)
        s1+=p[(i+1)%n].y*p[i].x,s2+=p[(i+1)%n].y*p[(i+2)%n].x;
    return fabs(s1-s2)/2;
}

//surface of ball
#include <math.h>
const double pi=acos(-1);
//计算圆心角 lat 表示纬度,-90<=w<=90,lng 表示经度
//返回两点所在大圆劣弧对应圆心角,0<=angle<=pi
double angle(double lng1,double lat1,double lng2,double lat2)
{
    double dlng=fabs(lng1-lng2)*pi/180;
    while (dlng>=pi+pi)
        dlng-=pi+pi;
    if (dlng>pi)
        dlng=pi+pi-dlng;
    lat1*=pi/180,lat2*=pi/180;
    return acos(cos(lat1)*cos(lat2)*cos(dlng)+sin(lat1)*sin(lat2));
}
//计算距离,r 为球半径
double line_dist(double r,double lng1,double lat1,double lng2,
   double lat2)
{
    double dlng=fabs(lng1-lng2)*pi/180;
    while (dlng>=pi+pi)
        dlng-=pi+pi;
    if (dlng>pi)
        dlng=pi+pi-dlng;
    lat1*=pi/180,lat2*=pi/180;
    return r*sqrt(2-2*(cos(lat1)*cos(lat2)*cos(dlng)+sin(lat1)*sin(
       lat2)));
}
//计算球面距离,r 为球半径
inline double sphere_dist(double r,double lng1,double lat1,double
```

```
    lng2,double lat2)
{
    return r*angle(lng1,lat1,lng2,lat2);
}

//triangle
#include <math.h>
struct point{double x,y;};
struct line{point a,b;};
double distance(point p1,point p2)
{
    return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}
point intersection(line u,line v)
{
    point ret=u.a;
    double t=((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.
       x))
         /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
    ret.x+=(u.b.x-u.a.x)*t;
    ret.y+=(u.b.y-u.a.y)*t;
    return ret;
}
//外心
point circumcenter(point a,point b,point c)
{
    line u,v;
    u.a.x=(a.x+b.x)/2;
    u.a.y=(a.y+b.y)/2;
    u.b.x=u.a.x-a.y+b.y;
    u.b.y=u.a.y+a.x-b.x;
    v.a.x=(a.x+c.x)/2;
    v.a.y=(a.y+c.y)/2;
    v.b.x=v.a.x-a.y+c.y;
    v.b.y=v.a.y+a.x-c.x;
    return intersection(u,v);
}
//内心
point incenter(point a,point b,point c)
{
    line u,v;
    double m,n;
    u.a=a;
    m=atan2(b.y-a.y,b.x-a.x);
    n=atan2(c.y-a.y,c.x-a.x);
    u.b.x=u.a.x+cos((m+n)/2);
    u.b.y=u.a.y+sin((m+n)/2);
    v.a=b;
    m=atan2(a.y-b.y,a.x-b.x);
    n=atan2(c.y-b.y,c.x-b.x);
    v.b.x=v.a.x+cos((m+n)/2);
```

```
591     v.b.y=v.a.y+sin((m+n)/2);
592     return intersection(u,v);
593 }
594 //垂心
595 point perpencenter(point a,point b,point c)
596 {
597     line u,v;
598     u.a=c;
599     u.b.x=u.a.x-a.y+b.y;
600     u.b.y=u.a.y+a.x-b.x;
601     v.a=b;
602     v.b.x=v.a.x-a.y+c.y;
603     v.b.y=v.a.y+a.x-c.x;
604     return intersection(u,v);
605 }
606 //重心
607 //到三角形三顶点距离的平方和最小的点
608 //三角形内到三边距离之积最大的点
609 point barycenter(point a,point b,point c)
610 {
611     line u,v;
612     u.a.x=(a.x+b.x)/2;
613     u.a.y=(a.y+b.y)/2;
614     u.b=c;
615     v.a.x=(a.x+c.x)/2;
616     v.a.y=(a.y+c.y)/2;
617     v.b=b;
618     return intersection(u,v);
619 }
620 //费马点
621 //到三角形三顶点距离之和最小的点
622 point fermentpoint(point a,point b,point c)
623 {
624     point u,v;
625     double step=fabs(a.x)+fabs(a.y)+fabs(b.x)+fabs(b.y)+fabs(c.x)+
        fabs(c.y);
626     int i,j,k;
627     u.x=(a.x+b.x+c.x)/3;
628     u.y=(a.y+b.y+c.y)/3;
629     while (step>1e-10)
630         for (k=0;k<10;step/=2,k++)
631             for (i=-1;i<=1;i++)
632                 for (j=-1;j<=1;j++)
633                 {
634                     v.x=u.x+step*i;
635                     v.y=u.y+step*j;
636                     if
637                         (distance(u,a)+distance(u,b)+distance(u,c)>
                            distance(v,a)+distance(v,b)+distance(v,c
                            ))
638                          u=v;
```

```
                }
    return u;
}

//3-d
//三维几何函数库
#include <math.h>
#define eps 1e-8
#define zero(x) (((x)>0?(x):-(x))<eps)
struct point3{double x,y,z;};
struct line3{point3 a,b;};
struct plane3{point3 a,b,c;};
//计算 cross product U x V
point3 xmult(point3 u,point3 v)
{
    point3 ret;
    ret.x=u.y*v.z-v.y*u.z;
    ret.y=u.z*v.x-u.x*v.z;
    ret.z=u.x*v.y-u.y*v.x;
    return ret;
}
//计算 dot product U . V
double dmult(point3 u,point3 v)
{
    return u.x*v.x+u.y*v.y+u.z*v.z;
}
//矢量差 U - V
point3 subt(point3 u,point3 v)
{
    point3 ret;
    ret.x=u.x-v.x;
    ret.y=u.y-v.y;
    ret.z=u.z-v.z;
    return ret;
}
//取平面法向量
point3 pvec(plane3 s)
{
    return xmult(subt(s.a,s.b),subt(s.b,s.c));
}
point3 pvec(point3 s1,point3 s2,point3 s3)
{
    return xmult(subt(s1,s2),subt(s2,s3));
}
//两点距离，单参数取向量大小
double distance(point3 p1,point3 p2)
{
    return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y)+(p1
       .z-p2.z)*(p1.z-p2.z));
}
//向量大小
```

```
689 double vlen(point3 p)
690 {
691     return sqrt(p.x*p.x+p.y*p.y+p.z*p.z);
692 }
693 //判三点共线
694 int dots_inline(point3 p1,point3 p2,point3 p3)
695 {
696     return vlen(xmult(subt(p1,p2),subt(p2,p3)))<eps;
697 }
698 //判四点共面
699 int dots_onplane(point3 a,point3 b,point3 c,point3 d)
700 {
701     return zero(dmult(pvec(a,b,c),subt(d,a)));
702 }
703 //判点是否在线段上，包括端点和共线
704 int dot_online_in(point3 p,line3 l)
705 {
706     return zero(vlen(xmult(subt(p,l.a),subt(p,l.b))))&&(l.a.x-p.x)
        *(l.b.x-p.x)<eps&&
707         (l.a.y-p.y)*(l.b.y-p.y)<eps&&(l.a.z-p.z)*(l.b.z-p.z)<eps;
708 }
709 int dot_online_in(point3 p,point3 l1,point3 l2)
710 {
711     return zero(vlen(xmult(subt(p,l1),subt(p,l2))))&&(l1.x-p.x)*(l2
        .x-p.x)<eps&&
712         (l1.y-p.y)*(l2.y-p.y)<eps&&(l1.z-p.z)*(l2.z-p.z)<eps;
713 }
714 //判点是否在线段上，不包括端点
715 int dot_online_ex(point3 p,line3 l)
716 {
717     return dot_online_in(p,l)&&(!zero(p.x-l.a.x)||!zero(p.y-l.a.y)
        ||!zero(p.z-l.a.z))&&
718         (!zero(p.x-l.b.x)||!zero(p.y-l.b.y)||!zero(p.z-l.b.z));
719 }
720 int dot_online_ex(point3 p,point3 l1,point3 l2)
721 {
722     return dot_online_in(p,l1,l2)&&(!zero(p.x-l1.x)||!zero(p.y-l1.y
        )||!zero(p.z-l1.z))&&
723         (!zero(p.x-l2.x)||!zero(p.y-l2.y)||!zero(p.z-l2.z));
724 }
725 //判点是否在空间三角形上，包括边界，三点共线无意义
726 int dot_inplane_in(point3 p,plane3 s)
727 {
728     return zero(vlen(xmult(subt(s.a,s.b),subt(s.a,s.c)))-vlen(xmult
        (subt(p,s.a),subt(p,s.b)))-
729             vlen(xmult(subt(p,s.b),subt(p,s.c)))-vlen(xmult(subt(p,
                s.c),subt(p,s.a))));
730 }
731 int dot_inplane_in(point3 p,point3 s1,point3 s2,point3 s3)
732 {
733     return zero(vlen(xmult(subt(s1,s2),subt(s1,s3)))-vlen(xmult(
```

```
       subt(p,s1),subt(p,s2)))-
734          vlen(xmult(subt(p,s2),subt(p,s3)))-vlen(xmult(subt(p,s3
                ),subt(p,s1))));
735 }
736 //判点是否在空间三角形上，不包括边界，三点共线无意义
737 int dot_inplane_ex(point3 p,plane3 s)
738 {
739     return dot_inplane_in(p,s)&&vlen(xmult(subt(p,s.a),subt(p,s.b))
           )>eps&&
740         vlen(xmult(subt(p,s.b),subt(p,s.c)))>eps&&vlen(xmult(subt(p
              ,s.c),subt(p,s.a)))>eps;
741 }
742 int dot_inplane_ex(point3 p,point3 s1,point3 s2,point3 s3)
743 {
744     return dot_inplane_in(p,s1,s2,s3)&&vlen(xmult(subt(p,s1),subt(p
           ,s2)))>eps&&
745         vlen(xmult(subt(p,s2),subt(p,s3)))>eps&&vlen(xmult(subt(p,
              s3),subt(p,s1)))>eps;
746 }
747 //判两点在线段同侧，点在线段上返回 0，不共面无意义
748 int same_side(point3 p1,point3 p2,line3 l)
749 {
750     return dmult(xmult(subt(l.a,l.b),subt(p1,l.b)),xmult(subt(l.a,l
           .b),subt(p2,l.b)))>eps;
751 }
752 int same_side(point3 p1,point3 p2,point3 l1,point3 l2)
753 {
754     return dmult(xmult(subt(l1,l2),subt(p1,l2)),xmult(subt(l1,l2),
           subt(p2,l2)))>eps;
755 }
756 //判两点在线段异侧，点在线段上返回 0，不共面无意义
757 int opposite_side(point3 p1,point3 p2,line3 l)
758 {
759     return dmult(xmult(subt(l.a,l.b),subt(p1,l.b)),xmult(subt(l.a,l
           .b),subt(p2,l.b)))<-eps;
760 }
761 int opposite_side(point3 p1,point3 p2,point3 l1,point3 l2)
762 {
763     return dmult(xmult(subt(l1,l2),subt(p1,l2)),xmult(subt(l1,l2),
           subt(p2,l2)))<-eps;
764 }
765 //判两点在平面同侧，点在平面上返回 0
766 int same_side(point3 p1,point3 p2,plane3 s)
767 {
768     return dmult(pvec(s),subt(p1,s.a))*dmult(pvec(s),subt(p2,s.a))>
           eps;
769 }
770 int same_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3)
771 {
772     return dmult(pvec(s1,s2,s3),subt(p1,s1))*dmult(pvec(s1,s2,s3),
           subt(p2,s1))>eps;
```

```
773 }
774 //判两点在平面异侧, 点在平面上返回 0
775 int opposite_side(point3 p1,point3 p2,plane3 s)
776 {
777     return dmult(pvec(s),subt(p1,s.a))*dmult(pvec(s),subt(p2,s.a))
        <-eps;
778 }
779 int opposite_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3
    )
780 {
781     return dmult(pvec(s1,s2,s3),subt(p1,s1))*dmult(pvec(s1,s2,s3),
        subt(p2,s1))<-eps;
782 }
783 //判两直线平行
784 int parallel(line3 u,line3 v)
785 {
786     return vlen(xmult(subt(u.a,u.b),subt(v.a,v.b)))<eps;
787 }
788 int parallel(point3 u1,point3 u2,point3 v1,point3 v2)
789 {
790     return vlen(xmult(subt(u1,u2),subt(v1,v2)))<eps;
791 }
792 //判两平面平行
793 int parallel(plane3 u,plane3 v)
794 {
795     return vlen(xmult(pvec(u),pvec(v)))<eps;
796 }
797 int parallel(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,
    point3 v3)
798 {
799     return vlen(xmult(pvec(u1,u2,u3),pvec(v1,v2,v3)))<eps;
800 }
801 //判直线与平面平行
802 int parallel(line3 l,plane3 s)
803 {
804     return zero(dmult(subt(l.a,l.b),pvec(s)));
805 }
806 int parallel(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
807 {
808     return zero(dmult(subt(l1,l2),pvec(s1,s2,s3)));
809 }
810 //判两直线垂直
811 int perpendicular(line3 u,line3 v)
812 {
813     return zero(dmult(subt(u.a,u.b),subt(v.a,v.b)));
814 }
815 int perpendicular(point3 u1,point3 u2,point3 v1,point3 v2)
816 {
817     return zero(dmult(subt(u1,u2),subt(v1,v2)));
818 }
819 //判两平面垂直
```

```
820 int perpendicular(plane3 u,plane3 v)
821 {
822     return zero(dmult(pvec(u),pvec(v)));
823 }
824 int perpendicular(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2
       ,point3 v3)
825 {
826     return zero(dmult(pvec(u1,u2,u3),pvec(v1,v2,v3)));
827 }
828 //判直线与平面平行
829 int perpendicular(line3 l,plane3 s)
830 {
831     return vlen(xmult(subt(l.a,l.b),pvec(s)))<eps;
832 }
833 int perpendicular(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3
       )
834 {
835     return vlen(xmult(subt(l1,l2),pvec(s1,s2,s3)))<eps;
836 }
837 //判两线段相交, 包括端点和部分重合
838 int intersect_in(line3 u,line3 v)
839 {
840     if (!dots_onplane(u.a,u.b,v.a,v.b))
841         return 0;
842     if (!dots_inline(u.a,u.b,v.a)||!dots_inline(u.a,u.b,v.b))
843         return !same_side(u.a,u.b,v)&&!same_side(v.a,v.b,u);
844     return dot_online_in(u.a,v)||dot_online_in(u.b,v)||
          dot_online_in(v.a,u)||dot_online_in(v.b,u);
845 }
846 int intersect_in(point3 u1,point3 u2,point3 v1,point3 v2)
847 {
848     if (!dots_onplane(u1,u2,v1,v2))
849         return 0;
850     if (!dots_inline(u1,u2,v1)||!dots_inline(u1,u2,v2))
851         return !same_side(u1,u2,v1,v2)&&!same_side(v1,v2,u1,u2);
852     return
853         dot_online_in(u1,v1,v2)||dot_online_in(u2,v1,v2)||
              dot_online_in(v1,u1,u2)||dot_online_in(v2,u1,u
854                   2);
855 }
856 //判两线段相交, 不包括端点和部分重合
857 int intersect_ex(line3 u,line3 v)
858 {
859     return dots_onplane(u.a,u.b,v.a,v.b)&&opposite_side(u.a,u.b,v)
          &&opposite_side(v.a,v.b,u);
860 }
861 int intersect_ex(point3 u1,point3 u2,point3 v1,point3 v2)
862 {
863     return
864         dots_onplane(u1,u2,v1,v2)&&opposite_side(u1,u2,v1,v2)&&
              opposite_side(v1,v2,u1,u2);
```

```
865 }
866 //判线段与空间三角形相交，包括交于边界和（部分）包含
867 int intersect_in(line3 l,plane3 s)
868 {
869     return !same_side(l.a,l.b,s)&&!same_side(s.a,s.b,l.a,l.b,s.c)&&
870         !same_side(s.b,s.c,l.a,l.b,s.a)&&!same_side(s.c,s.a,l.a,l.b
            ,s.b);
871 }
872 int intersect_in(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
873 {
874     return !same_side(l1,l2,s1,s2,s3)&&!same_side(s1,s2,l1,l2,s3)&&
875         !same_side(s2,s3,l1,l2,s1)&&!same_side(s3,s1,l1,l2,s2);
876 }
877 //判线段与空间三角形相交，不包括交于边界和（部分）包含
878 int intersect_ex(line3 l,plane3 s)
879 {
880     return opposite_side(l.a,l.b,s)&&opposite_side(s.a,s.b,l.a,l.b,
        s.c)&&
881         opposite_side(s.b,s.c,l.a,l.b,s.a)&&opposite_side(s.c,s.a,l
            .a,l.b,s.b);
882 }
883 int intersect_ex(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
884 {
885     return opposite_side(l1,l2,s1,s2,s3)&&opposite_side(s1,s2,l1,l2
        ,s3)&&
886         opposite_side(s2,s3,l1,l2,s1)&&opposite_side(s3,s1,l1,l2,s2
            );
887 }
888 //计算两直线交点，注意事先判断直线是否共面和平行!
889 //线段交点请另外判线段相交（同时还是要判断是否平行!）
890 point3 intersection(line3 u,line3 v)
891 {
892     point3 ret=u.a;
893     double t=((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.
        x))
894         /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
895     ret.x+=(u.b.x-u.a.x)*t;
896     ret.y+=(u.b.y-u.a.y)*t;
897     ret.z+=(u.b.z-u.a.z)*t;
898     return ret;
899 }
900 point3 intersection(point3 u1,point3 u2,point3 v1,point3 v2)
901 {
902     point3 ret=u1;
903     double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
904         /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
905     ret.x+=(u2.x-u1.x)*t;
906     ret.y+=(u2.y-u1.y)*t;
907     ret.z+=(u2.z-u1.z)*t;
908     return ret;
909 }
```

```
910 //计算直线与平面交点，注意事先判断是否平行，并保证三点不共线!
911 //线段和空间三角形交点请另外判断
912 point3 intersection(line3 l,plane3 s)
913 {
914     point3 ret=pvec(s);
915     double t=(ret.x*(s.a.x-l.a.x)+ret.y*(s.a.y-l.a.y)+ret.z*(s.a.z-
        l.a.z))/
916          (ret.x*(l.b.x-l.a.x)+ret.y*(l.b.y-l.a.y)+ret.z*(l.b.z-l.a.z
             ));
917     ret.x=l.a.x+(l.b.x-l.a.x)*t;
918     ret.y=l.a.y+(l.b.y-l.a.y)*t;
919     ret.z=l.a.z+(l.b.z-l.a.z)*t;
920     return ret;
921 }
922 point3 intersection(point3 l1,point3 l2,point3 s1,point3 s2,point3
    s3)
923 {
924     point3 ret=pvec(s1,s2,s3);
925     double t=(ret.x*(s1.x-l1.x)+ret.y*(s1.y-l1.y)+ret.z*(s1.z-l1.z)
        )/
926          (ret.x*(l2.x-l1.x)+ret.y*(l2.y-l1.y)+ret.z*(l2.z-l1.z));
927     ret.x=l1.x+(l2.x-l1.x)*t;
928     ret.y=l1.y+(l2.y-l1.y)*t;
929     ret.z=l1.z+(l2.z-l1.z)*t;
930     return ret;
931 }
932 //计算两平面交线，注意事先判断是否平行，并保证三点不共线!
933 line3 intersection(plane3 u,plane3 v)
934 {
935     line3 ret;
936     ret.a=parallel(v.a,v.b,u.a,u.b,u.c)?intersection(v.b,v.c,u.a,u.
        b,u.c):intersection(v.a,v.b,u.a,u.b,u.
937               c);
938     ret.b=parallel(v.c,v.a,u.a,u.b,u.c)?intersection(v.b,v.c,u.a,u.
        b,u.c):intersection(v.c,v.a,u.a,u.b,u.
939               c);
940     return ret;
941 }
942 line3 intersection(point3 u1,point3 u2,point3 u3,point3 v1,point3
    v2,point3 v3)
943 {
944     line3 ret;
945     ret.a=parallel(v1,v2,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):
        intersection(v1,v2,u1,u2,u3);
946     ret.b=parallel(v3,v1,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):
        intersection(v3,v1,u1,u2,u3);
947     return ret;
948 }
949 //点到直线距离
950 double ptoline(point3 p,line3 l)
951 {
```

```
952     return vlen(xmult(subt(p,l.a),subt(l.b,l.a)))/distance(l.a,l.b)
            ;
953 }
954 double ptoline(point3 p,point3 l1,point3 l2)
955 {
956     return vlen(xmult(subt(p,l1),subt(l2,l1)))/distance(l1,l2);
957 }
958 //点到平面距离
959 double ptoplane(point3 p,plane3 s)
960 {
961     return fabs(dmult(pvec(s),subt(p,s.a)))/vlen(pvec(s));
962 }
963 double ptoplane(point3 p,point3 s1,point3 s2,point3 s3)
964 {
965     return fabs(dmult(pvec(s1,s2,s3),subt(p,s1)))/vlen(pvec(s1,s2,
            s3));
966 }
967 //直线到直线距离
968 double linetoline(line3 u,line3 v)
969 {
970     point3 n=xmult(subt(u.a,u.b),subt(v.a,v.b));
971     return fabs(dmult(subt(u.a,v.a),n))/vlen(n);
972 }
973 double linetoline(point3 u1,point3 u2,point3 v1,point3 v2)
974 {
975     point3 n=xmult(subt(u1,u2),subt(v1,v2));
976     return fabs(dmult(subt(u1,v1),n))/vlen(n);
977 }
978 //两直线夹角 cos 值
979 double angle_cos(line3 u,line3 v)
980 {
981     return dmult(subt(u.a,u.b),subt(v.a,v.b))/vlen(subt(u.a,u.b))/
            vlen(subt(v.a,v.b));
982 }
983 double angle_cos(point3 u1,point3 u2,point3 v1,point3 v2)
984 {
985     return dmult(subt(u1,u2),subt(v1,v2))/vlen(subt(u1,u2))/vlen(
            subt(v1,v2));
986 }
987 //两平面夹角 cos 值
988 double angle_cos(plane3 u,plane3 v)
989 {
990     return dmult(pvec(u),pvec(v))/vlen(pvec(u))/vlen(pvec(v));
991 }
992 double angle_cos(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,
       point3 v3)
993 {
994     return dmult(pvec(u1,u2,u3),pvec(v1,v2,v3))/vlen(pvec(u1,u2,u3)
            )/vlen(pvec(v1,v2,v3));
995 }
996 //直线平面夹角 sin 值
```

```
 997 double angle_sin(line3 l,plane3 s)
 998 {
 999     return dmult(subt(l.a,l.b),pvec(s))/vlen(subt(l.a,l.b))/vlen(
         pvec(s));
1000 }
1001 double angle_sin(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3)
1002 {
1003     return dmult(subt(l1,l2),pvec(s1,s2,s3))/vlen(subt(l1,l2))/vlen
         (pvec(s1,s2,s3));
1004 }
1005
1006 //CH
1007 #include <stdlib.h>
1008 #define eps 1e-8
1009 #define zero(x) (((x)>0?(x):-(x))<eps)
1010 struct point{double x,y;};
1011 //计算 cross product (P1-P0)x(P2-P0)
1012 double xmult(point p1,point p2,point p0)
1013 {
1014     return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
1015 }
1016 //graham 算法顺时针构造包含所有共线点的凸包,O(nlogn)
1017 point p1,p2;
1018 int graham_cp(const void* a,const void* b)
1019 {
1020     double ret=xmult(*((point*)a),*((point*)b),p1);
1021     return zero(ret)?(xmult(*((point*)a),*((point*)b),p2)>0?1:-1):(
         ret>0?1:-1);
1022 }
1023 void _graham(int n,point* p,int& s,point* ch)
1024 {
1025     int i,k=0;
1026     for (p1=p2=p[0],i=1;i<n;p2.x+=p[i].x,p2.y+=p[i].y,i++)
1027         if (p1.y-p[i].y>eps||(zero(p1.y-p[i].y)&&p1.x>p[i].x))
1028             p1=p[k=i];
1029     p2.x/=n,p2.y/=n;
1030     p[k]=p[0],p[0]=p1;
1031     qsort(p+1,n-1,sizeof(point),graham_cp);
1032     for (ch[0]=p[0],ch[1]=p[1],ch[2]=p[2],s=i=3;i<n;ch[s++]=p[i++])
1033         for (;s>2&&xmult(ch[s-2],p[i],ch[s-1])<-eps;s--);
1034 }
1035 //构造凸包接口函数，传入原始点集大小 n，点集 p(p 原有顺序被打乱!)
1036 //返回凸包大小，凸包的点在 convex 中
1037 //参数 maxsize 为 1 包含共线点，为 0 不包含共线点，缺省为 1
1038 //参数 clockwise 为 1 顺时针构造，为 0 逆时针构造，缺省为 1
1039 //在输入仅有若干共线点时算法不稳定，可能有此类情况请另行处理!
1040 //不能去掉点集中重合的点
1041 int graham(int n,point* p,point* convex,int maxsize=1,int dir=1)
1042 {
1043     point* temp=new point[n];
1044     int s,i;
```

```
1045        _graham(n,p,s,temp);
1046        for (convex[0]=temp[0],n=1,i=(dir?1:(s-1));dir?(i<s):i;i+=(dir
                ?1:-1))
1047            if (maxsize||!zero(xmult(temp[i-1],temp[i],temp[(i+1)%s])))
1048                convex[n++]=temp[i];
1049        delete []temp;
1050        return n;
1051    }
1052
1053    //Pick's
1054    #define abs(x) ((x)>0?(x):-(x))
1055    struct point{int x,y;};
1056    int gcd(int a,int b)
1057    {
1058        return b?gcd(b,a%b):a;
1059    }
1060    //多边形上的网格点个数
1061    int grid_onedge(int n,point* p)
1062    {
1063        int i,ret=0;
1064        for (i=0;i<n;i++)
1065            ret+=gcd(abs(p[i].x-p[(i+1)%n].x),abs(p[i].y-p[(i+1)%n].y))
                    ;
1066        return ret;
1067    }
1068    //多边形内的网格点个数
1069    int grid_inside(int n,point* p)
1070    {
1071        int i,ret=0;
1072        for (i=0;i<n;i++)
1073            ret+=p[(i+1)%n].y*(p[i].x-p[(i+2)%n].x);
1074        return (abs(ret)-grid_onedge(n,p))/2+1;
1075    }
1076
1077    //circle
1078    #include <math.h>
1079    #define eps 1e-8
1080    struct point{double x,y;};
1081    double xmult(point p1,point p2,point p0)
1082    {
1083        return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
1084    }
1085    double distance(point p1,point p2)
1086    {
1087        return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
1088    }
1089    double disptoline(point p,point l1,point l2)
1090    {
1091        return fabs(xmult(p,l1,l2))/distance(l1,l2);
1092    }
1093    point intersection(point u1,point u2,point v1,point v2)
```

```
{
    point ret=u1;
    double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
        /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
    ret.x+=(u2.x-u1.x)*t;
    ret.y+=(u2.y-u1.y)*t;
    return ret;
}
//判直线和圆相交，包括相切
int intersect_line_circle(point c,double r,point l1,point l2)
{
    return disptoline(c,l1,l2)<r+eps;
}
//判线段和圆相交，包括端点和相切
int intersect_seg_circle(point c,double r,point l1,point l2)
{
    double t1=distance(c,l1)-r,t2=distance(c,l2)-r;
    point t=c;
    if (t1<eps||t2<eps)
        return t1>-eps||t2>-eps;
    t.x+=l1.y-l2.y;
    t.y+=l2.x-l1.x;
    return xmult(l1,c,t)*xmult(l2,c,t)<eps&&disptoline(c,l1,l2)-r<
       eps;
}
//判圆和圆相交，包括相切
int intersect_circle_circle(point c1,double r1,point c2,double r2)
{
    return distance(c1,c2)<r1+r2+eps&&distance(c1,c2)>fabs(r1-r2)-
       eps;
}
//计算圆上到点 p 最近点，如 p 与圆心重合，返回 p 本身
point dot_to_circle(point c,double r,point p)
{
    point u,v;
    if (distance(p,c)<eps)
        return p;
    u.x=c.x+r*fabs(c.x-p.x)/distance(c,p);
    u.y=c.y+r*fabs(c.y-p.y)/distance(c,p)*((c.x-p.x)*(c.y-p.y)
       <0?-1:1);
    v.x=c.x-r*fabs(c.x-p.x)/distance(c,p);
    v.y=c.y-r*fabs(c.y-p.y)/distance(c,p)*((c.x-p.x)*(c.y-p.y)
       <0?-1:1);
    return distance(u,p)<distance(v,p)?u:v;
}
//计算直线与圆的交点，保证直线与圆有交点
//计算线段与圆的交点可用这个函数后判点是否在线段上
void intersection_line_circle(point c,double r,point l1,point l2,
   point& p1,point& p2)
{
    point p=c;
```

```
    double t;
    p.x+=l1.y-l2.y;
    p.y+=l2.x-l1.x;
    p=intersection(p,c,l1,l2);
    t=sqrt(r*r-distance(p,c)*distance(p,c))/distance(l1,l2);
    p1.x=p.x+(l2.x-l1.x)*t;
    p1.y=p.y+(l2.y-l1.y)*t;
    p2.x=p.x-(l2.x-l1.x)*t;
    p2.y=p.y-(l2.y-l1.y)*t;
}
//计算圆与圆的交点，保证圆与圆有交点，圆心不重合
void intersection_circle_circle(point c1,double r1,point c2,double
   r2,point& p1,point& p2)
{
    point u,v;
    double t;
    t=(1+(r1*r1-r2*r2)/distance(c1,c2)/distance(c1,c2))/2;
    u.x=c1.x+(c2.x-c1.x)*t;
    u.y=c1.y+(c2.y-c1.y)*t;
    v.x=u.x+c1.y-c2.y;
    v.y=u.y-c1.x+c2.x;
    intersection_line_circle(c1,r1,u,v,p1,p2);
}

//integer
//整数几何函数库
//注意某些情况下整数运算会出界!
#define sign(a) ((a)>0?1:(((a)<0?-1:0)))
struct point{int x,y;};
struct line{point a,b;};
//计算 cross product (P1-P0)x(P2-P0)
int xmult(point p1,point p2,point p0)
{
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
int xmult(int x1,int y1,int x2,int y2,int x0,int y0)
{
    return (x1-x0)*(y2-y0)-(x2-x0)*(y1-y0);
}
//计算 dot product (P1-P0).(P2-P0)
int dmult(point p1,point p2,point p0)
{
    return (p1.x-p0.x)*(p2.x-p0.x)+(p1.y-p0.y)*(p2.y-p0.y);
}
int dmult(int x1,int y1,int x2,int y2,int x0,int y0)
{
    return (x1-x0)*(x2-x0)+(y1-y0)*(y2-y0);
}
//判三点共线
int dots_inline(point p1,point p2,point p3)
{
```

```
1190        return !xmult(p1,p2,p3);
1191    }
1192    int dots_inline(int x1,int y1,int x2,int y2,int x3,int y3)
1193    {
1194        return !xmult(x1,y1,x2,y2,x3,y3);
1195    }
1196    //判点是否在线段上，包括端点和部分重合
1197    int dot_online_in(point p,line l)
1198    {
1199        return !xmult(p,l.a,l.b)&&(l.a.x-p.x)*(l.b.x-p.x)<=0&&(l.a.y-p.
              y)*(l.b.y-p.y)<=0;
1200    }
1201    int dot_online_in(point p,point l1,point l2)
1202    {
1203        return !xmult(p,l1,l2)&&(l1.x-p.x)*(l2.x-p.x)<=0&&(l1.y-p.y)*(
              l2.y-p.y)<=0;
1204    }
1205    int dot_online_in(int x,int y,int x1,int y1,int x2,int y2)
1206    {
1207        return !xmult(x,y,x1,y1,x2,y2)&&(x1-x)*(x2-x)<=0&&(y1-y)*(y2-y)
              <=0;
1208    }
1209    //判点是否在线段上，不包括端点
1210    int dot_online_ex(point p,line l)
1211    {
1212        return dot_online_in(p,l)&&(p.x!=l.a.x||p.y!=l.a.y)&&(p.x!=l.b.
              x||p.y!=l.b.y);
1213    }
1214    int dot_online_ex(point p,point l1,point l2)
1215    {
1216        return dot_online_in(p,l1,l2)&&(p.x!=l1.x||p.y!=l1.y)&&(p.x!=l2
              .x||p.y!=l2.y);
1217    }
1218    int dot_online_ex(int x,int y,int x1,int y1,int x2,int y2)
1219    {
1220        return dot_online_in(x,y,x1,y1,x2,y2)&&(x!=x1||y!=y1)&&(x!=x2||
              y!=y2);
1221    }
1222    //判两点在直线同侧，点在直线上返回 0
1223    int same_side(point p1,point p2,line l)
1224    {
1225        return sign(xmult(l.a,p1,l.b))*xmult(l.a,p2,l.b)>0;
1226    }
1227    int same_side(point p1,point p2,point l1,point l2)
1228    {
1229        return sign(xmult(l1,p1,l2))*xmult(l1,p2,l2)>0;
1230    }
1231    //判两点在直线异侧，点在直线上返回 0
1232    int opposite_side(point p1,point p2,line l)
1233    {
1234        return sign(xmult(l.a,p1,l.b))*xmult(l.a,p2,l.b)<0;
```

```
1235 }
1236 int opposite_side(point p1,point p2,point l1,point l2)
1237 {
1238     return sign(xmult(l1,p1,l2))*xmult(l1,p2,l2)<0;
1239 }
1240 //判两直线平行
1241 int parallel(line u,line v)
1242 {
1243     return (u.a.x-u.b.x)*(v.a.y-v.b.y)==(v.a.x-v.b.x)*(u.a.y-u.b.y)
        ;
1244 }
1245 int parallel(point u1,point u2,point v1,point v2)
1246 {
1247     return (u1.x-u2.x)*(v1.y-v2.y)==(v1.x-v2.x)*(u1.y-u2.y);
1248 }
1249 //判两直线垂直
1250 int perpendicular(line u,line v)
1251 {
1252     return (u.a.x-u.b.x)*(v.a.x-v.b.x)==-(u.a.y-u.b.y)*(v.a.y-v.b.y
        );
1253 }
1254 int perpendicular(point u1,point u2,point v1,point v2)
1255 {
1256     return (u1.x-u2.x)*(v1.x-v2.x)==-(u1.y-u2.y)*(v1.y-v2.y);
1257 }
1258 //判两线段相交，包括端点和部分重合
1259 int intersect_in(line u,line v)
1260 {
1261     if (!dots_inline(u.a,u.b,v.a)||!dots_inline(u.a,u.b,v.b))
1262         return !same_side(u.a,u.b,v)&&!same_side(v.a,v.b,u);
1263     return dot_online_in(u.a,v)||dot_online_in(u.b,v)||
        dot_online_in(v.a,u)||dot_online_in(v.b,u);
1264 }
1265 int intersect_in(point u1,point u2,point v1,point v2)
1266 {
1267     if (!dots_inline(u1,u2,v1)||!dots_inline(u1,u2,v2))
1268         return !same_side(u1,u2,v1,v2)&&!same_side(v1,v2,u1,u2);
1269     return
1270         dot_online_in(u1,v1,v2)||dot_online_in(u2,v1,v2)||
            dot_online_in(v1,u1,u2)||dot_online_in(v2,u1,u
1271                 2);
1272 }
1273 //判两线段相交，不包括端点和部分重合
1274 int intersect_ex(line u,line v)
1275 {
1276     return opposite_side(u.a,u.b,v)&&opposite_side(v.a,v.b,u);
1277 }
1278 int intersect_ex(point u1,point u2,point v1,point v2)
1279 {
1280     return opposite_side(u1,u2,v1,v2)&&opposite_side(v1,v2,u1,u2);
1281 }
```

```
#include<vector>
#include<list>
#include<map>
#include<set>
#include<deque>
#include<queue>
#include<stack>
#include<bitset>
#include<algorithm>
#include<functional>
#include<numeric>
#include<utility>
#include<iostream>
#include<sstream>
#include<iomanip>
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<cctype>
#include<string>
#include<cstring>
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<ctime>
#include<climits>
#include<complex>
#define mp make_pair
#define pb push_back
using namespace std;
const double eps=1e-8;
const double pi=acos(-1.0);
const double inf=1e20;
const int maxp=8;
int dblcmp(double d)
{
    if (fabs(d)<eps)return 0;
    return d>eps?1:-1;
}
inline double sqr(double x){return x*x;}
struct point
{
    double x,y;
    point(){}
    point(double _x,double _y):
        x(_x),y(_y){};
    void input()
    {
        scanf("%lf%lf",&x,&y);
```

```
50      }
51      void output()
52      {
53          printf("%.2f␣%.2f\n",x,y);
54      }
55      bool operator==(point a)const
56      {
57          return dblcmp(a.x-x)==0&&dblcmp(a.y-y)==0;
58      }
59      bool operator<(point a)const
60      {
61          return dblcmp(a.x-x)==0?dblcmp(y-a.y)<0:x<a.x;
62      }
63      double len()
64      {
65          return hypot(x,y);
66      }
67      double len2()
68      {
69          return x*x+y*y;
70      }
71      double distance(point p)
72      {
73          return hypot(x-p.x,y-p.y);
74      }
75      point add(point p)
76      {
77          return point(x+p.x,y+p.y);
78      }
79      point sub(point p)
80      {
81          return point(x-p.x,y-p.y);
82      }
83      point mul(double b)
84      {
85          return point(x*b,y*b);
86      }
87      point div(double b)
88      {
89          return point(x/b,y/b);
90      }
91      double dot(point p)
92      {
93          return x*p.x+y*p.y;
94      }
95      double det(point p)
96      {
97          return x*p.y-y*p.x;
98      }
99      double rad(point a,point b)
100     {
```

```
101         point p=*this;
102         return fabs(atan2(fabs(a.sub(p).det(b.sub(p))),a.sub(p).dot
                (b.sub(p))));
103     }
104     point trunc(double r)
105     {
106         double l=len();
107         if (!dblcmp(l))return *this;
108         r/=l;
109         return point(x*r,y*r);
110     }
111     point rotleft()
112     {
113         return point(-y,x);
114     }
115     point rotright()
116     {
117         return point(y,-x);
118     }
119     point rotate(point p,double angle)//绕点逆时针旋转角度pangle
120     {
121         point v=this->sub(p);
122         double c=cos(angle),s=sin(angle);
123         return point(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
124     }
125 };
126 struct line
127 {
128     point a,b;
129     line(){}
130     line(point _a,point _b)
131     {
132         a=_a;
133         b=_b;
134     }
135     bool operator==(line v)
136     {
137         return (a==v.a)&&(b==v.b);
138     }
139     //倾斜角angle
140     line(point p,double angle)
141     {
142         a=p;
143         if (dblcmp(angle-pi/2)==0)
144         {
145             b=a.add(point(0,1));
146         }
147         else
148         {
149             b=a.add(point(1,tan(angle)));
150         }
```

```
151     }
152     //ax+by+c=0
153     line(double _a,double _b,double _c)
154     {
155         if (dblcmp(_a)==0)
156         {
157             a=point(0,-_c/_b);
158             b=point(1,-_c/_b);
159         }
160         else if (dblcmp(_b)==0)
161         {
162             a=point(-_c/_a,0);
163             b=point(-_c/_a,1);
164         }
165         else
166         {
167             a=point(0,-_c/_b);
168             b=point(1,(-_c-_a)/_b);
169         }
170     }
171     void input()
172     {
173         a.input();
174         b.input();
175     }
176     void adjust()
177     {
178         if (b<a)swap(a,b);
179     }
180     double length()
181     {
182         return a.distance(b);
183     }
184     double angle()//直线倾斜角 0<=angle<180
185     {
186         double k=atan2(b.y-a.y,b.x-a.x);
187         if (dblcmp(k)<0)k+=pi;
188         if (dblcmp(k-pi)==0)k-=pi;
189         return k;
190     }
191     //点和线段关系
192     //1 在逆时针
193     //2 在顺时针
194     //3 平行
195     int relation(point p)
196     {
197         int c=dblcmp(p.sub(a).det(b.sub(a)));
198         if (c<0)return 1;
199         if (c>0)return 2;
200         return 3;
201     }
```

```
202     bool pointonseg(point p)
203     {
204         return dblcmp(p.sub(a).det(b.sub(a)))==0&&dblcmp(p.sub(a).
                dot(p.sub(b)))<=0;
205     }
206     bool parallel(line v)
207     {
208         return dblcmp(b.sub(a).det(v.b.sub(v.a)))==0;
209     }
210     //2 规范相交
211     //1 非规范相交
212     //0 不相交
213     int segcrossseg(line v)
214     {
215         int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
216         int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
217         int d3=dblcmp(v.b.sub(v.a).det(a.sub(v.a)));
218         int d4=dblcmp(v.b.sub(v.a).det(b.sub(v.a)));
219         if ((d1^d2)==-2&&(d3^d4)==-2)return 2;
220         return (d1==0&&dblcmp(v.a.sub(a).dot(v.a.sub(b)))<=0||
221                 d2==0&&dblcmp(v.b.sub(a).dot(v.b.sub(b)))<=0||
222                 d3==0&&dblcmp(a.sub(v.a).dot(a.sub(v.b)))<=0||
223                 d4==0&&dblcmp(b.sub(v.a).dot(b.sub(v.b)))<=0);
224     }
225     int linecrossseg(line v)//*this seg v line
226     {
227         int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
228         int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
229         if ((d1^d2)==-2)return 2;
230         return (d1==0||d2==0);
231     }
232     //0 平行
233     //1 重合
234     //2 相交
235     int linecrossline(line v)
236     {
237         if ((*this).parallel(v))
238         {
239             return v.relation(a)==3;
240         }
241         return 2;
242     }
243     point crosspoint(line v)
244     {
245         double a1=v.b.sub(v.a).det(a.sub(v.a));
246         double a2=v.b.sub(v.a).det(b.sub(v.a));
247         return point((a.x*a2-b.x*a1)/(a2-a1),(a.y*a2-b.y*a1)/(a2-a1
                ));
248     }
249     double dispointtoline(point p)
250     {
```

```
251         return fabs(p.sub(a).det(b.sub(a)))/length();
252     }
253     double dispointtoseg(point p)
254     {
255         if (dblcmp(p.sub(b).dot(a.sub(b)))<0||dblcmp(p.sub(a).dot(b
               .sub(a)))<0)
256         {
257             return min(p.distance(a),p.distance(b));
258         }
259         return dispointtoline(p);
260     }
261     point lineprog(point p)
262     {
263         return a.add(b.sub(a).mul(b.sub(a).dot(p.sub(a))/b.sub(a).
               len2()));
264     }
265     point symmetrypoint(point p)
266     {
267         point q=lineprog(p);
268         return point(2*q.x-p.x,2*q.y-p.y);
269     }
270 };
271 struct circle
272 {
273     point p;
274     double r;
275     circle(){}
276     circle(point _p,double _r):
277         p(_p),r(_r){};
278     circle(double x,double y,double _r):
279         p(point(x,y)),r(_r){};
280     circle(point a,point b,point c)//三角形的外接圆
281     {
282         p=line(a.add(b).div(2),a.add(b).div(2).add(b.sub(a).rotleft
               ())).crosspoint(line(c.add(b).div(2),c.add(b).div(2).add
               (b.sub(c).rotleft()));
283         r=p.distance(a);
284     }
285     circle(point a,point b,point c,bool t)//三角形的内切圆
286     {
287         line u,v;
288         double m=atan2(b.y-a.y,b.x-a.x),n=atan2(c.y-a.y,c.x-a.x);
289         u.a=a;
290         u.b=u.a.add(point(cos((n+m)/2),sin((n+m)/2)));
291         v.a=b;
292         m=atan2(a.y-b.y,a.x-b.x),n=atan2(c.y-b.y,c.x-b.x);
293         v.b=v.a.add(point(cos((n+m)/2),sin((n+m)/2)));
294         p=u.crosspoint(v);
295         r=line(a,b).dispointtoseg(p);
296     }
297     void input()
```

```
298     {
299         p.input();
300         scanf("%lf",&r);
301     }
302     void output()
303     {
304         printf("%.2lf␣%.2lf␣%.2lf\n",p.x,p.y,r);
305     }
306     bool operator==(circle v)
307     {
308         return ((p==v.p)&&dblcmp(r-v.r)==0);
309     }
310     bool operator<(circle v)const
311     {
312         return ((p<v.p)||(p==v.p)&&dblcmp(r-v.r)<0);
313     }
314     double area()
315     {
316         return pi*sqr(r);
317     }
318     double circumference()
319     {
320         return 2*pi*r;
321     }
322     //0 圆外
323     //1 圆上
324     //2 圆内
325     int relation(point b)
326     {
327         double dst=b.distance(p);
328         if (dblcmp(dst-r)<0)return 2;
329         if (dblcmp(dst-r)==0)return 1;
330         return 0;
331     }
332     int relationseg(line v)
333     {
334         double dst=v.dispointtoseg(p);
335         if (dblcmp(dst-r)<0)return 2;
336         if (dblcmp(dst-r)==0)return 1;
337         return 0;
338     }
339     int relationline(line v)
340     {
341         double dst=v.dispointtoline(p);
342         if (dblcmp(dst-r)<0)return 2;
343         if (dblcmp(dst-r)==0)return 1;
344         return 0;
345     }
346     //过a 两点b 半径的两个圆r
347     int getcircle(point a,point b,double r,circle&c1,circle&c2)
348     {
```

```
349         circle x(a,r),y(b,r);
350         int t=x.pointcrosscircle(y,c1.p,c2.p);
351         if (!t)return 0;
352         c1.r=c2.r=r;
353         return t;
354     }
355     //与直线相切u 过点q 半径的圆r1
356     int getcircle(line u,point q,double r1,circle &c1,circle &c2)
357     {
358         double dis=u.dispointtoline(q);
359         if (dblcmp(dis-r1*2)>0)return 0;
360         if (dblcmp(dis)==0)
361         {
362             c1.p=q.add(u.b.sub(u.a).rotleft().trunc(r1));
363             c2.p=q.add(u.b.sub(u.a).rotright().trunc(r1));
364             c1.r=c2.r=r1;
365             return 2;
366         }
367         line u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),u.b.
               add(u.b.sub(u.a).rotleft().trunc(r1)));
368         line u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),u.b
               .add(u.b.sub(u.a).rotright().trunc(r1)));
369         circle cc=circle(q,r1);
370         point p1,p2;
371         if (!cc.pointcrossline(u1,p1,p2))cc.pointcrossline(u2,p1,p2
               );
372         c1=circle(p1,r1);
373         if (p1==p2)
374         {
375             c2=c1;return 1;
376         }
377         c2=circle(p2,r1);
378         return 2;
379     }
380     //同时与直线u,相切v 半径的圆r1
381     int getcircle(line u,line v,double r1,circle &c1,circle &c2,
           circle &c3,circle &c4)
382     {
383         if (u.parallel(v))return 0;
384         line u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),u.b.
               add(u.b.sub(u.a).rotleft().trunc(r1)));
385         line u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),u.b
               .add(u.b.sub(u.a).rotright().trunc(r1)));
386         line v1=line(v.a.add(v.b.sub(v.a).rotleft().trunc(r1)),v.b.
               add(v.b.sub(v.a).rotleft().trunc(r1)));
387         line v2=line(v.a.add(v.b.sub(v.a).rotright().trunc(r1)),v.b
               .add(v.b.sub(v.a).rotright().trunc(r1)));
388         c1.r=c2.r=c3.r=c4.r=r1;
389         c1.p=u1.crosspoint(v1);
390         c2.p=u1.crosspoint(v2);
391         c3.p=u2.crosspoint(v1);
```

```
392         c4.p=u2.crosspoint(v2);
393         return 4;
394     }
395     //同时与不相交圆cx,相切cy 半径为的圆r1
396     int getcircle(circle cx,circle cy,double r1,circle&c1,circle&c2
        )
397     {
398         circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
399         int t=x.pointcrosscircle(y,c1.p,c2.p);
400         if (!t)return 0;
401         c1.r=c2.r=r1;
402         return t;
403     }
404     int pointcrossline(line v,point &p1,point &p2)//求与线段交要先判
        断relationseg
405     {
406         if (!(*this).relationline(v))return 0;
407         point a=v.lineprog(p);
408         double d=v.dispointtoline(p);
409         d=sqrt(r*r-d*d);
410         if (dblcmp(d)==0)
411         {
412             p1=a;
413             p2=a;
414             return 1;
415         }
416         p1=a.sub(v.b.sub(v.a).trunc(d));
417         p2=a.add(v.b.sub(v.a).trunc(d));
418         return 2;
419     }
420     //5 相离
421     //4 外切
422     //3 相交
423     //2 内切
424     //1 内含
425     int relationcircle(circle v)
426     {
427         double d=p.distance(v.p);
428         if (dblcmp(d-r-v.r)>0)return 5;
429         if (dblcmp(d-r-v.r)==0)return 4;
430         double l=fabs(r-v.r);
431         if (dblcmp(d-r-v.r)<0&&dblcmp(d-l)>0)return 3;
432         if (dblcmp(d-l)==0)return 2;
433         if (dblcmp(d-l)<0)return 1;
434     }
435     int pointcrosscircle(circle v,point &p1,point &p2)
436     {
437         int rel=relationcircle(v);
438         if (rel==1||rel==5)return 0;
439         double d=p.distance(v.p);
440         double l=(d+(sqr(r)-sqr(v.r))/d)/2;
```

```
441         double h=sqrt(sqr(r)-sqr(l));
442         p1=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotleft().trunc
               (h)));
443         p2=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotright().
               trunc(h)));
444         if (rel==2||rel==4)
445         {
446             return 1;
447         }
448         return 2;
449     }
450     //过一点做圆的切线 先判断点和圆关系()
451     int tangentline(point q,line &u,line &v)
452     {
453         int x=relation(q);
454         if (x==2)return 0;
455         if (x==1)
456         {
457             u=line(q,q.add(q.sub(p).rotleft()));
458             v=u;
459             return 1;
460         }
461         double d=p.distance(q);
462         double l=sqr(r)/d;
463         double h=sqrt(sqr(r)-sqr(l));
464         u=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotleft().
               trunc(h))));
465         v=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotright().
               trunc(h))));
466         return 2;
467     }
468     double areacircle(circle v)
469     {
470         int rel=relationcircle(v);
471         if (rel>=4)return 0.0;
472         if (rel<=2)return min(area(),v.area());
473         double d=p.distance(v.p);
474         double hf=(r+v.r+d)/2.0;
475         double ss=2*sqrt(hf*(hf-r)*(hf-v.r)*(hf-d));
476         double a1=acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
477         a1=a1*r*r;
478         double a2=acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
479         a2=a2*v.r*v.r;
480         return a1+a2-ss;
481     }
482     double areatriangle(point a,point b)
483     {
484         if (dblcmp(p.sub(a).det(p.sub(b))==0))return 0.0;
485         point q[5];
486         int len=0;
487         q[len++]=a;
```

```
488         line l(a,b);
489         point p1,p2;
490         if (pointcrossline(l,q[1],q[2])==2)
491         {
492             if (dblcmp(a.sub(q[1]).dot(b.sub(q[1])))<0)q[len++]=q
                    [1];
493             if (dblcmp(a.sub(q[2]).dot(b.sub(q[2])))<0)q[len++]=q
                    [2];
494         }
495         q[len++]=b;
496         if (len==4&&(dblcmp(q[0].sub(q[1]).dot(q[2].sub(q[1])))>0))
               swap(q[1],q[2]);
497         double res=0;
498         int i;
499         for (i=0;i<len-1;i++)
500         {
501             if (relation(q[i])==0||relation(q[i+1])==0)
502             {
503                 double arg=p.rad(q[i],q[i+1]);
504                 res+=r*r*arg/2.0;
505             }
506             else
507             {
508                 res+=fabs(q[i].sub(p).det(q[i+1].sub(p))/2.0);
509             }
510         }
511         return res;
512     }
513 };
514 struct polygon
515 {
516     int n;
517     point p[maxp];
518     line l[maxp];
519     void input()
520     {
521         n=4;
522         p[0].input();
523         p[2].input();
524         double dis=p[0].distance(p[2]);
525         p[1]=p[2].rotate(p[0],pi/4);
526         p[1]=p[0].add((p[1].sub(p[0])).trunc(dis/sqrt(2.0)));
527         p[3]=p[2].rotate(p[0],2*pi-pi/4);
528         p[3]=p[0].add((p[3].sub(p[0])).trunc(dis/sqrt(2.0)));
529     }
530     void add(point q)
531     {
532         p[n++]=q;
533     }
534     void getline()
535     {
```

```
536         for (int i=0;i<n;i++)
537         {
538             l[i]=line(p[i],p[(i+1)%n]);
539         }
540     }
541     struct cmp
542     {
543         point p;
544         cmp(const point &p0){p=p0;}
545         bool operator()(const point &aa,const point &bb)
546         {
547             point a=aa,b=bb;
548             int d=dblcmp(a.sub(p).det(b.sub(p)));
549             if (d==0)
550             {
551                 return dblcmp(a.distance(p)-b.distance(p))<0;
552             }
553             return d>0;
554         }
555     };
556     void norm()
557     {
558         point mi=p[0];
559         for (int i=1;i<n;i++)mi=min(mi,p[i]);
560         sort(p,p+n,cmp(mi));
561     }
562     void getconvex(polygon &convex)
563     {
564         int i,j,k;
565         sort(p,p+n);
566         convex.n=n;
567         for (i=0;i<min(n,2);i++)
568         {
569             convex.p[i]=p[i];
570         }
571         if (n<=2)return;
572         int &top=convex.n;
573         top=1;
574         for (i=2;i<n;i++)
575         {
576             while (top&&convex.p[top].sub(p[i]).det(convex.p[top
                    -1].sub(p[i]))<=0)
577                 top--;
578             convex.p[++top]=p[i];
579         }
580         int temp=top;
581         convex.p[++top]=p[n-2];
582         for (i=n-3;i>=0;i--)
583         {
584             while (top!=temp&&convex.p[top].sub(p[i]).det(convex.p[
                    top-1].sub(p[i]))<=0)
```

```
585                 top--;
586             convex.p[++top]=p[i];
587         }
588     }
589     bool isconvex()
590     {
591         bool s[3];
592         memset(s,0,sizeof(s));
593         int i,j,k;
594         for (i=0;i<n;i++)
595         {
596             j=(i+1)%n;
597             k=(j+1)%n;
598             s[dblcmp(p[j].sub(p[i]).det(p[k].sub(p[i])))+1]=1;
599             if (s[0]&&s[2])return 0;
600         }
601         return 1;
602     }
603     //3 点上
604     //2 边上
605     //1 内部
606     //0 外部
607     int relationpoint(point q)
608     {
609         int i,j;
610         for (i=0;i<n;i++)
611         {
612             if (p[i]==q)return 3;
613         }
614         getline();
615         for (i=0;i<n;i++)
616         {
617             if (l[i].pointonseg(q))return 2;
618         }
619         int cnt=0;
620         for (i=0;i<n;i++)
621         {
622             j=(i+1)%n;
623             int k=dblcmp(q.sub(p[j]).det(p[i].sub(p[j])));
624             int u=dblcmp(p[i].y-q.y);
625             int v=dblcmp(p[j].y-q.y);
626             if (k>0&&u<0&&v>=0)cnt++;
627             if (k<0&&v<0&&u>=0)cnt--;
628         }
629         return cnt!=0;
630     }
631     //1 在多边形内长度为正
632     //2 相交或与边平行
633     //0 无任何交点
634     int relationline(line u)
635     {
```

```
636         int i,j,k=0;
637         getline();
638         for (i=0;i<n;i++)
639         {
640             if (l[i].segcrossseg(u)==2)return 1;
641             if (l[i].segcrossseg(u)==1)k=1;
642         }
643         if (!k)return 0;
644         vector<point>vp;
645         for (i=0;i<n;i++)
646         {
647             if (l[i].segcrossseg(u))
648             {
649                 if (l[i].parallel(u))
650                 {
651                     vp.pb(u.a);
652                     vp.pb(u.b);
653                     vp.pb(l[i].a);
654                     vp.pb(l[i].b);
655                     continue;
656                 }
657                 vp.pb(l[i].crosspoint(u));
658             }
659         }
660         sort(vp.begin(),vp.end());
661         int sz=vp.size();
662         for (i=0;i<sz-1;i++)
663         {
664             point mid=vp[i].add(vp[i+1]).div(2);
665             if (relationpoint(mid)==1)return 1;
666         }
667         return 2;
668     }
669     //直线切割凸多边形左侧u
670     //注意直线方向
671     void convexcut(line u,polygon &po)
672     {
673         int i,j,k;
674         int &top=po.n;
675         top=0;
676         for (i=0;i<n;i++)
677         {
678             int d1=dblcmp(p[i].sub(u.a).det(u.b.sub(u.a)));
679             int d2=dblcmp(p[(i+1)%n].sub(u.a).det(u.b.sub(u.a)));
680             if (d1>=0)po.p[top++]=p[i];
681             if (d1*d2<0)po.p[top++]=u.crosspoint(line(p[i],p[(i+1)%
                  n]));
682         }
683     }
684     double getcircumference()
685     {
```

```
686         double sum=0;
687         int i;
688         for (i=0;i<n;i++)
689         {
690             sum+=p[i].distance(p[(i+1)%n]);
691         }
692         return sum;
693     }
694     double getarea()
695     {
696         double sum=0;
697         int i;
698         for (i=0;i<n;i++)
699         {
700             sum+=p[i].det(p[(i+1)%n]);
701         }
702         return fabs(sum)/2;
703     }
704     bool getdir()//代表逆时针1 代表顺时针0
705     {
706         double sum=0;
707         int i;
708         for (i=0;i<n;i++)
709         {
710             sum+=p[i].det(p[(i+1)%n]);
711         }
712         if (dblcmp(sum)>0)return 1;
713         return 0;
714     }
715     point getbarycentre()
716     {
717         point ret(0,0);
718         double area=0;
719         int i;
720         for (i=1;i<n-1;i++)
721         {
722             double tmp=p[i].sub(p[0]).det(p[i+1].sub(p[0]));
723             if (dblcmp(tmp)==0)continue;
724             area+=tmp;
725             ret.x+=(p[0].x+p[i].x+p[i+1].x)/3*tmp;
726             ret.y+=(p[0].y+p[i].y+p[i+1].y)/3*tmp;
727         }
728         if (dblcmp(area))ret=ret.div(area);
729         return ret;
730     }
731     double areaintersection(polygon po)
732     {
733     }
734     double areaunion(polygon po)
735     {
736         return getarea()+po.getarea()-areaintersection(po);
```

```
737     }
738     double areacircle(circle c)
739     {
740         int i,j,k,l,m;
741         double ans=0;
742         for (i=0;i<n;i++)
743         {
744             int j=(i+1)%n;
745             if (dblcmp(p[j].sub(c.p).det(p[i].sub(c.p)))>=0)
746             {
747                 ans+=c.areatriangle(p[i],p[j]);
748             }
749             else
750             {
751                 ans-=c.areatriangle(p[i],p[j]);
752             }
753         }
754         return fabs(ans);
755     }
756     //多边形和圆关系
757     //0 一部分在圆外
758     //1 与圆某条边相切
759     //2 完全在圆内
760     int relationcircle(circle c)
761     {
762         getline();
763         int i,x=2;
764         if (relationpoint(c.p)!=1)return 0;
765         for (i=0;i<n;i++)
766         {
767             if (c.relationseg(l[i])==2)return 0;
768             if (c.relationseg(l[i])==1)x=1;
769         }
770         return x;
771     }
772     void find(int st,point tri[],circle &c)
773     {
774         if (!st)
775         {
776             c=circle(point(0,0),-2);
777         }
778         if (st==1)
779         {
780             c=circle(tri[0],0);
781         }
782         if (st==2)
783         {
784             c=circle(tri[0].add(tri[1]).div(2),tri[0].distance(tri
                    [1])/2.0);
785         }
786         if (st==3)
```

```
787         {
788             c=circle(tri[0],tri[1],tri[2]);
789         }
790     }
791     void solve(int cur,int st,point tri[],circle &c)
792     {
793         find(st,tri,c);
794         if (st==3)return;
795         int i;
796         for (i=0;i<cur;i++)
797         {
798             if (dblcmp(p[i].distance(c.p)-c.r)>0)
799             {
800                 tri[st]=p[i];
801                 solve(i,st+1,tri,c);
802             }
803         }
804     }
805     circle mincircle()//点集最小圆覆盖
806     {
807         random_shuffle(p,p+n);
808         point tri[4];
809         circle c;
810         solve(n,0,tri,c);
811         return c;
812     }
813     int circlecover(double r)//单位圆覆盖
814     {
815         int ans=0,i,j;
816         vector<pair<double,int> >v;
817         for (i=0;i<n;i++)
818         {
819             v.clear();
820             for (j=0;j<n;j++)if (i!=j)
821             {
822                 point q=p[i].sub(p[j]);
823                 double d=q.len();
824                 if (dblcmp(d-2*r)<=0)
825                 {
826                     double arg=atan2(q.y,q.x);
827                     if (dblcmp(arg)<0)arg+=2*pi;
828                     double t=acos(d/(2*r));
829                     v.push_back(make_pair(arg-t+2*pi,-1));
830                     v.push_back(make_pair(arg+t+2*pi,1));
831                 }
832             }
833             sort(v.begin(),v.end());
834             int cur=0;
835             for (j=0;j<v.size();j++)
836             {
837                 if (v[j].second==-1)++cur;
```

```
838                 else --cur;
839                 ans=max(ans,cur);
840             }
841         }
842         return ans+1;
843     }
844     int pointinpolygon(point q)//点在凸多边形内部的判定
845     {
846         if (getdir())reverse(p,p+n);
847         if (dblcmp(q.sub(p[0]).det(p[n-1].sub(p[0])))==0)
848         {
849             if (line(p[n-1],p[0]).pointonseg(q))return n-1;
850             return -1;
851         }
852         int low=1,high=n-2,mid;
853         while (low<=high)
854         {
855             mid=(low+high)>>1;
856             if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0])))>=0&&
                    dblcmp(q.sub(p[0]).det(p[mid+1].sub(p[0])))<0)
857             {
858                 polygon c;
859                 c.p[0]=p[mid];
860                 c.p[1]=p[mid+1];
861                 c.p[2]=p[0];
862                 c.n=3;
863                 if (c.relationpoint(q))return mid;
864                 return -1;
865             }
866             if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0])))>0)
867             {
868                 low=mid+1;
869             }
870             else
871             {
872                 high=mid-1;
873             }
874         }
875         return -1;
876     }
877 };
878 struct polygons
879 {
880     vector<polygon>p;
881     polygons()
882     {
883         p.clear();
884     }
885     void clear()
886     {
887         p.clear();
```

```
888     }
889     void push(polygon q)
890     {
891         if (dblcmp(q.getarea()))p.pb(q);
892     }
893     vector<pair<double,int> >e;
894     void ins(point s,point t,point X,int i)
895     {
896         double r=fabs(t.x-s.x)>eps?(X.x-s.x)/(t.x-s.x):(X.y-s.y)/(t
                .y-s.y);
897         r=min(r,1.0);r=max(r,0.0);
898         e.pb(mp(r,i));
899     }
900     double polyareaunion()
901     {
902         double ans=0.0;
903         int c0,c1,c2,i,j,k,w;
904         for (i=0;i<p.size();i++)
905         {
906             if (p[i].getdir()==0)reverse(p[i].p,p[i].p+p[i].n);
907         }
908         for (i=0;i<p.size();i++)
909         {
910             for (k=0;k<p[i].n;k++)
911             {
912                 point &s=p[i].p[k],&t=p[i].p[(k+1)%p[i].n];
913                 if (!dblcmp(s.det(t)))continue;
914                 e.clear();
915                 e.pb(mp(0.0,1));
916                 e.pb(mp(1.0,-1));
917                 for (j=0;j<p.size();j++)if (i!=j)
918                 {
919                     for (w=0;w<p[j].n;w++)
920                     {
921                         point a=p[j].p[w],b=p[j].p[(w+1)%p[j].n],c=
                            p[j].p[(w-1+p[j].n)%p[j].n];
922                         c0=dblcmp(t.sub(s).det(c.sub(s)));
923                         c1=dblcmp(t.sub(s).det(a.sub(s)));
924                         c2=dblcmp(t.sub(s).det(b.sub(s)));
925                         if (c1*c2<0)ins(s,t,line(s,t).crosspoint(
                            line(a,b)),-c2);
926                         else if (!c1&&c0*c2<0)ins(s,t,a,-c2);
927                         else if (!c1&&!c2)
928                         {
929                             int c3=dblcmp(t.sub(s).det(p[j].p[(w+2)
                                %p[j].n].sub(s)));
930                             int dp=dblcmp(t.sub(s).dot(b.sub(a)));
931                             if (dp&&c0)ins(s,t,a,dp>0?c0*((j>i)^(c0
                                <0)):-(c0<0));
932                             if (dp&&c3)ins(s,t,b,dp>0?-c3*((j>i)^(
                                c3<0)):c3<0);
```

```
                        }
                    }
                }
                sort(e.begin(),e.end());
                int ct=0;
                double tot=0.0,last;
                for (j=0;j<e.size();j++)
                {
                    if (ct==p.size())tot+=e[j].first-last;
                    ct+=e[j].second;
                    last=e[j].first;
                }
                ans+=s.det(t)*tot;
            }
        }
        return fabs(ans)*0.5;
    }
};
const int maxn=500;
struct circles
{
    circle c[maxn];
    double ans[maxn];//ans[i表示被覆盖了]次的面积i
    double pre[maxn];
    int n;
    circles(){}
    void add(circle cc)
    {
        c[n++]=cc;
    }
    bool inner(circle x,circle y)
    {
        if (x.relationcircle(y)!=1)return 0;
        return dblcmp(x.r-y.r)<=0?1:0;
    }
    void init_or()//圆的面积并去掉内含的圆
    {
        int i,j,k=0;
        bool mark[maxn]={0};
        for (i=0;i<n;i++)
        {
            for (j=0;j<n;j++)if (i!=j&&!mark[j])
            {
                if ((c[i]==c[j])||inner(c[i],c[j]))break;
            }
            if (j<n)mark[i]=1;
        }
        for (i=0;i<n;i++)if (!mark[i])c[k++]=c[i];
        n=k;
    }
    void init_and()//圆的面积交去掉内含的圆
```

```
    {
        int i,j,k=0;
        bool mark[maxn]={0};
        for (i=0;i<n;i++)
        {
            for (j=0;j<n;j++)if (i!=j&&!mark[j])
            {
                if ((c[i]==c[j])||inner(c[j],c[i]))break;
            }
            if (j<n)mark[i]=1;
        }
        for (i=0;i<n;i++)if (!mark[i])c[k++]=c[i];
        n=k;
    }
    double areaarc(double th,double r)
    {
        return 0.5*sqr(r)*(th-sin(th));
    }
    void getarea()
    {
        int i,j,k;
        memset(ans,0,sizeof(ans));
        vector<pair<double,int> >v;
        for (i=0;i<n;i++)
        {
            v.clear();
            v.push_back(make_pair(-pi,1));
            v.push_back(make_pair(pi,-1));
            for (j=0;j<n;j++)if (i!=j)
            {
                point q=c[j].p.sub(c[i].p);
                double ab=q.len(),ac=c[i].r,bc=c[j].r;
                if (dblcmp(ab+ac-bc)<=0)
                {
                    v.push_back(make_pair(-pi,1));
                    v.push_back(make_pair(pi,-1));
                    continue;
                }
                if (dblcmp(ab+bc-ac)<=0)continue;
                if (dblcmp(ab-ac-bc)>0) continue;
                double th=atan2(q.y,q.x),fai=acos((ac*ac+ab*ab-bc*
                   bc)/(2.0*ac*ab));
                double a0=th-fai;
                if (dblcmp(a0+pi)<0)a0+=2*pi;
                double a1=th+fai;
                if (dblcmp(a1-pi)>0)a1-=2*pi;
                if (dblcmp(a0-a1)>0)
                {
                    v.push_back(make_pair(a0,1));
                    v.push_back(make_pair(pi,-1));
                    v.push_back(make_pair(-pi,1));
```

```
                    v.push_back(make_pair(a1,-1));
                }
                else
                {
                    v.push_back(make_pair(a0,1));
                    v.push_back(make_pair(a1,-1));
                }
            }
            sort(v.begin(),v.end());
            int cur=0;
            for (j=0;j<v.size();j++)
            {
                if (cur&&dblcmp(v[j].first-pre[cur]))
                {
                    ans[cur]+=areaarc(v[j].first-pre[cur],c[i].r);
                    ans[cur]+=0.5*point(c[i].p.x+c[i].r*cos(pre[cur
                       ]),c[i].p.y+c[i].r*sin(pre[cur])).det(point(
                       c[i].p.x+c[i].r*cos(v[j].first),c[i].p.y+c[i
                       ].r*sin(v[j].first)));
                }
                cur+=v[j].second;
                pre[cur]=v[j].first;
            }
        }
        for (i=1;i<=n;i++)
        {
            ans[i]-=ans[i+1];
        }
    }
};
struct halfplane:public line
{
    double angle;
    halfplane(){}
    //表示向量 a->逆时针b左侧()的半平面
    halfplane(point _a,point _b)
    {
        a=_a;
        b=_b;
    }
    halfplane(line v)
    {
        a=v.a;
        b=v.b;
    }
    void calcangle()
    {
        angle=atan2(b.y-a.y,b.x-a.x);
    }
    bool operator<(const halfplane &b)const
    {
```

```
1082            return angle<b.angle;
1083        }
1084    };
1085    struct halfplanes
1086    {
1087        int n;
1088        halfplane hp[maxp];
1089        point p[maxp];
1090        int que[maxp];
1091        int st,ed;
1092        void push(halfplane tmp)
1093        {
1094            hp[n++]=tmp;
1095        }
1096        void unique()
1097        {
1098            int m=1,i;
1099            for (i=1;i<n;i++)
1100            {
1101                if (dblcmp(hp[i].angle-hp[i-1].angle))hp[m++]=hp[i];
1102                else if (dblcmp(hp[m-1].b.sub(hp[m-1].a).det(hp[i].a.
                       sub(hp[m-1].a))>0))hp[m-1]=hp[i];
1103            }
1104            n=m;
1105        }
1106        bool halfplaneinsert()
1107        {
1108            int i;
1109            for (i=0;i<n;i++)hp[i].calcangle();
1110            sort(hp,hp+n);
1111            unique();
1112            que[st=0]=0;
1113            que[ed=1]=1;
1114            p[1]=hp[0].crosspoint(hp[1]);
1115            for (i=2;i<n;i++)
1116            {
1117                while (st<ed&&dblcmp((hp[i].b.sub(hp[i].a).det(p[ed].
                       sub(hp[i].a))))<0)ed--;
1118                while (st<ed&&dblcmp((hp[i].b.sub(hp[i].a).det(p[st+1].
                       sub(hp[i].a))))<0)st++;
1119                que[++ed]=i;
1120                if (hp[i].parallel(hp[que[ed-1]]))return false;
1121                p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
1122            }
1123            while (st<ed&&dblcmp(hp[que[st]].b.sub(hp[que[st]].a).det(p
                   [ed].sub(hp[que[st]].a)))<0)ed--;
1124            while (st<ed&&dblcmp(hp[que[ed]].b.sub(hp[que[ed]].a).det(p
                   [st+1].sub(hp[que[ed]].a)))<0)st++;
1125            if (st+1>=ed)return false;
1126            return true;
1127        }
```

```
1128        void getconvex(polygon &con)
1129        {
1130            p[st]=hp[que[st]].crosspoint(hp[que[ed]]);
1131            con.n=ed-st+1;
1132            int j=st,i=0;
1133            for (;j<=ed;i++,j++)
1134            {
1135                con.p[i]=p[j];
1136            }
1137        }
1138    };
1139    struct point3
1140    {
1141        double x,y,z;
1142        point3(){}
1143        point3(double _x,double _y,double _z):
1144            x(_x),y(_y),z(_z){};
1145        void input()
1146        {
1147            scanf("%lf%lf%lf",&x,&y,&z);
1148        }
1149        void output()
1150        {
1151            printf("%.2lf␣%.2lf␣%.2lf\n",x,y,z);
1152        }
1153        bool operator==(point3 a)
1154        {
1155            return dblcmp(a.x-x)==0&&dblcmp(a.y-y)==0&&dblcmp(a.z-z)
                    ==0;
1156        }
1157        bool operator<(point3 a)const
1158        {
1159            return dblcmp(a.x-x)==0?dblcmp(y-a.y)==0?dblcmp(z-a.z)<0:y<
                    a.y:x<a.x;
1160        }
1161        double len()
1162        {
1163            return sqrt(len2());
1164        }
1165        double len2()
1166        {
1167            return x*x+y*y+z*z;
1168        }
1169        double distance(point3 p)
1170        {
1171            return sqrt((p.x-x)*(p.x-x)+(p.y-y)*(p.y-y)+(p.z-z)*(p.z-z)
                    );
1172        }
1173        point3 add(point3 p)
1174        {
1175            return point3(x+p.x,y+p.y,z+p.z);
```

```
1176     }
1177     point3 sub(point3 p)
1178     {
1179         return point3(x-p.x,y-p.y,z-p.z);
1180     }
1181     point3 mul(double d)
1182     {
1183         return point3(x*d,y*d,z*d);
1184     }
1185     point3 div(double d)
1186     {
1187         return point3(x/d,y/d,z/d);
1188     }
1189     double dot(point3 p)
1190     {
1191         return x*p.x+y*p.y+z*p.z;
1192     }
1193     point3 det(point3 p)
1194     {
1195         return point3(y*p.z-p.y*z,p.x*z-x*p.z,x*p.y-p.x*y);
1196     }
1197     double rad(point3 a,point3 b)
1198     {
1199         point3 p=(*this);
1200         return acos(a.sub(p).dot(b.sub(p))/(a.distance(p)*b.
               distance(p)));
1201     }
1202     point3 trunc(double r)
1203     {
1204         r/=len();
1205         return point3(x*r,y*r,z*r);
1206     }
1207     point3 rotate(point3 o,double r)
1208     {
1209     }
1210 };
1211 struct line3
1212 {
1213     point3 a,b;
1214     line3(){}
1215     line3(point3 _a,point3 _b)
1216     {
1217         a=_a;
1218         b=_b;
1219     }
1220     bool operator==(line3 v)
1221     {
1222         return (a==v.a)&&(b==v.b);
1223     }
1224     void input()
1225     {
```

```
1226        a.input();
1227        b.input();
1228    }
1229    double length()
1230    {
1231        return a.distance(b);
1232    }
1233    bool pointonseg(point3 p)
1234    {
1235        return dblcmp(p.sub(a).det(p.sub(b)).len())==0&&dblcmp(a.
               sub(p).dot(b.sub(p)))<=0;
1236    }
1237    double dispointtoline(point3 p)
1238    {
1239        return b.sub(a).det(p.sub(a)).len()/a.distance(b);
1240    }
1241    double dispointtoseg(point3 p)
1242    {
1243        if (dblcmp(p.sub(b).dot(a.sub(b)))<0||dblcmp(p.sub(a).dot(b
               .sub(a)))<0)
1244        {
1245            return min(p.distance(a),p.distance(b));
1246        }
1247        return dispointtoline(p);
1248    }
1249    point3 lineprog(point3 p)
1250    {
1251        return a.add(b.sub(a).trunc(b.sub(a).dot(p.sub(a))/b.
               distance(a)));
1252    }
1253    point3 rotate(point3 p,double ang)//绕此向量逆时针角度parg
1254    {
1255        if (dblcmp((p.sub(a).det(p.sub(b)).len()))==0)return p;
1256        point3 f1=b.sub(a).det(p.sub(a));
1257        point3 f2=b.sub(a).det(f1);
1258        double len=fabs(a.sub(p).det(b.sub(p)).len()/a.distance(b))
               ;
1259        f1=f1.trunc(len);f2=f2.trunc(len);
1260        point3 h=p.add(f2);
1261        point3 pp=h.add(f1);
1262        return h.add((p.sub(h)).mul(cos(ang*1.0))).add((pp.sub(h)).
               mul(sin(ang*1.0)));
1263    }
1264 };
1265 struct plane
1266 {
1267    point3 a,b,c,o;
1268    plane(){}
1269    plane(point3 _a,point3 _b,point3 _c)
1270    {
1271        a=_a;
```

```
1272        b=_b;
1273        c=_c;
1274        o=pvec();
1275    }
1276    plane(double _a,double _b,double _c,double _d)
1277    {
1278        //ax+by+cz+d=0
1279        o=point3(_a,_b,_c);
1280        if (dblcmp(_a)!=0)
1281        {
1282            a=point3((-_d-_c-_b)/_a,1,1);
1283        }
1284        else if (dblcmp(_b)!=0)
1285        {
1286            a=point3(1,(-_d-_c-_a)/_b,1);
1287        }
1288        else if (dblcmp(_c)!=0)
1289        {
1290            a=point3(1,1,(-_d-_a-_b)/_c);
1291        }
1292    }
1293    void input()
1294    {
1295        a.input();
1296        b.input();
1297        c.input();
1298        o=pvec();
1299    }
1300    point3 pvec()
1301    {
1302        return b.sub(a).det(c.sub(a));
1303    }
1304    bool pointonplane(point3 p)//点是否在平面上
1305    {
1306        return dblcmp(p.sub(a).dot(o))==0;
1307    }
1308    //0 不在
1309    //1 在边界上
1310    //2 在内部
1311    int pointontriangle(point3 p)//点是否在空间三角形上abc
1312    {
1313        if (!pointonplane(p))return 0;
1314        double s=a.sub(b).det(c.sub(b)).len();
1315        double s1=p.sub(a).det(p.sub(b)).len();
1316        double s2=p.sub(a).det(p.sub(c)).len();
1317        double s3=p.sub(b).det(p.sub(c)).len();
1318        if (dblcmp(s-s1-s2-s3))return 0;
1319        if (dblcmp(s1)&&dblcmp(s2)&&dblcmp(s3))return 2;
1320        return 1;
1321    }
1322    //判断两平面关系
```

```
1323        //0 相交
1324        //1 平行但不重合
1325        //2 重合
1326        bool relationplane(plane f)
1327        {
1328            if (dblcmp(o.det(f.o).len()))return 0;
1329            if (pointonplane(f.a))return 2;
1330            return 1;
1331        }
1332        double angleplane(plane f)//两平面夹角
1333        {
1334            return acos(o.dot(f.o)/(o.len()*f.o.len()));
1335        }
1336        double dispoint(point3 p)//点到平面距离
1337        {
1338            return fabs(p.sub(a).dot(o)/o.len());
1339        }
1340        point3 pttoplane(point3 p)//点到平面最近点
1341        {
1342            line3 u=line3(p,p.add(o));
1343            crossline(u,p);
1344            return p;
1345        }
1346        int crossline(line3 u,point3 &p)//平面和直线的交点
1347        {
1348            double x=o.dot(u.b.sub(a));
1349            double y=o.dot(u.a.sub(a));
1350            double d=x-y;
1351            if (dblcmp(fabs(d))==0)return 0;
1352            p=u.a.mul(x).sub(u.b.mul(y)).div(d);
1353            return 1;
1354        }
1355        int crossplane(plane f,line3 &u)//平面和平面的交线
1356        {
1357            point3 oo=o.det(f.o);
1358            point3 v=o.det(oo);
1359            double d=fabs(f.o.dot(v));
1360            if (dblcmp(d)==0)return 0;
1361            point3 q=a.add(v.mul(f.o.dot(f.a.sub(a))/d));
1362            u=line3(q,q.add(oo));
1363            return 1;
1364        }
1365    };
```

4 Graph

4.1 2SAT

```
1  /*
2  x & y == true:
3  ~x -> x
4  ~y -> y
```

```
 5
 6 x & y == false:
 7 x -> ~y
 8 y -> ~x
 9
10 x | y == true:
11 ~x -> y
12 ~y -> x
13
14 x | y == false:
15 x -> ~x
16 y -> ~y
17
18 x ^ y == true:
19 ~x -> y
20 y -> ~x
21 x -> ~y
22 ~y -> x
23
24 x ^ y == false:
25 x -> y
26 y -> x
27 ~x -> ~y
28 ~y -> ~x
29 */
30 #include<cstdio>
31 #include<cstring>
32
33 #define MAXX 16111
34 #define MAXE 200111
35 #define v to[i]
36
37 int edge[MAXX],to[MAXE],nxt[MAXE],cnt;
38 inline void add(int a,int b)
39 {
40     nxt[++cnt]=edge[a];
41     edge[a]=cnt;
42     to[cnt]=b;
43 }
44
45 bool done[MAXX];
46 int st[MAXX];
47
48 bool dfs(const int now)
49 {
50     if(done[now^1])
51         return false;
52     if(done[now])
53         return true;
54     done[now]=true;
55     st[cnt++]=now;
```

```
56      for(int i(edge[now]);i;i=nxt[i])
57          if(!dfs(v))
58              return false;
59      return true;
60  }
61
62  int n,m;
63  int i,j,k;
64
65  inline bool go()
66  {
67      memset(done,0,sizeof done);
68      for(i=0;i<n;i+=2)
69          if(!done[i] && !done[i^1])
70          {
71              cnt=0;
72              if(!dfs(i))
73              {
74                  while(cnt)
75                      done[st[--cnt]]=false;
76                  if(!dfs(i^1))
77                      return false;
78              }
79          }
80      return true;
81  }
82  //done array will be a solution with minimal lexicographical order
83  // or maybe we can solve it with dual SCC method, and get a
       solution by reverse the edges of DAG then product a topsort
```

4.2 Articulation

```
1   void dfs(int now,int fa)  // now 从 1 开始
2   {
3       int p(0);
4       dfn[now]=low[now]=cnt++;
5       for(std::list<int>::const_iterator it(edge[now].begin());it!=
           edge[now].end();++it)
6           if(dfn[*it]==-1)
7           {
8               dfs(*it,now);
9               ++p;
10              low[now]=std::min(low[now],low[*it]);
11              if((now==1 && p>1) || (now!=1 && low[*it]>=dfn[now]))
                    // 如果从出发点出发的子节点不能由兄弟节点到达，那么出发点为割
                    点。如果现节点不是出发点，但是其子孙节点不能达到祖先节点，那么
                    该节点为割点
12                      ans.insert(now);
13          }
14          else
15              if(*it!=fa)
16                  low[now]=std::min(low[now],dfn[*it]);
```

```
}
```

4.3 Augmenting Path Algorithm for Maximum Cardinality Bipartite Matching

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

#define MAXX 111

bool Map[MAXX][MAXX],visit[MAXX];
int link[MAXX],n,m;
bool dfs(int t)
{
    for (int i=0; i<m; i++)
        if (!visit[i] && Map[t][i]){
            visit[i] = true;
            if (link[i]==-1 || dfs(link[i])){
                link[i] = t;
                return true;
            }
        }
    return false;
}
int main()
{
    int k,a,b,c;
    while (scanf("%d",&n),n){
        memset(Map,false,sizeof(Map));
        scanf("%d%d",&m,&k);
        while (k--){
            scanf("%d%d%d",&a,&b,&c);
            if (b && c)
                Map[b][c] = true;
        }
        memset(link,-1,sizeof(link));
        int ans = 0;
        for (int i=0; i<n; i++){
            memset(visit,false,sizeof(visit));
            if (dfs(i))
                ans++;
        }
        printf("%d\n",ans);
    }
}
```

4.4 Biconnected Component - Edge

```
// hdu 4612
#include<cstdio>
#include<algorithm>
#include<set>
#include<cstring>
```

```cpp
#include<stack>
#include<queue>

#define MAXX 200111
#define MAXE (1000111*2)
#pragma comment(linker, "/STACK:16777216")

int edge[MAXX],to[MAXE],nxt[MAXE],cnt;
#define v to[i]
inline void add(int a,int b)
{
    nxt[++cnt]=edge[a];
    edge[a]=cnt;
    to[cnt]=b;
}

int dfn[MAXX],low[MAXX],col[MAXX],belong[MAXX];
int idx,bcnt;
std::stack<int>st;

void tarjan(int now,int last)
{
    col[now]=1;
    st.push(now);
    dfn[now]=low[now]=++idx;
    bool flag(false);
    for(int i(edge[now]);i;i=nxt[i])
    {
        if(v==last && !flag)
        {
            flag=true;
            continue;
        }
        if(!col[v])
        {
            tarjan(v,now);
            low[now]=std::min(low[now],low[v]);
            /*
            if(low[v]>dfn[now])
            then this is a bridge
            */
        }
        else
            if(col[v]==1)
                low[now]=std::min(low[now],dfn[v]);
    }
    col[now]=2;
    if(dfn[now]==low[now])
    {
        ++bcnt;
        static int x;
```

```
57          do
58          {
59              x=st.top();
60              st.pop();
61              belong[x]=bcnt;
62          }while(x!=now);
63      }
64  }
65
66  std::set<int>set[MAXX];
67
68  int dist[MAXX];
69  std::queue<int>q;
70  int n,m,i,j,k;
71
72  inline int go(int s)
73  {
74      static std::set<int>::const_iterator it;
75      memset(dist,0x3f,sizeof dist);
76      dist[s]=0;
77      q.push(s);
78      while(!q.empty())
79      {
80          s=q.front();
81          q.pop();
82          for(it=set[s].begin();it!=set[s].end();++it)
83              if(dist[*it]>dist[s]+1)
84              {
85                  dist[*it]=dist[s]+1;
86                  q.push(*it);
87              }
88      }
89      return std::max_element(dist+1,dist+1+bcnt)-dist;
90  }
91
92  int main()
93  {
94      while(scanf("%d␣%d",&n,&m),(n||m))
95      {
96          cnt=0;
97          memset(edge,0,sizeof edge);
98          while(m--)
99          {
100             scanf("%d␣%d",&i,&j);
101             add(i,j);
102             add(j,i);
103         }
104
105         memset(dfn,0,sizeof dfn);
106         memset(belong,0,sizeof belong);
107         memset(low,0,sizeof low);
```

```
108         memset(col,0,sizeof col);
109         bcnt=idx=0;
110         while(!st.empty())
111             st.pop();
112 
113         tarjan(1,-1);
114         for(i=1;i<=bcnt;++i)
115             set[i].clear();
116         for(i=1;i<=n;++i)
117             for(j=edge[i];j;j=nxt[j])
118                 set[belong[i]].insert(belong[to[j]]);
119         for(i=1;i<=bcnt;++i)
120             set[i].erase(i);
121         /*
122         printf("%d\n",dist[go(go(1))]);
123         for(i=1;i<=bcnt;++i)
124             printf("%d\n",dist[i]);
125         puts("");
126         */
127         printf("%d\n",bcnt-1-dist[go(go(1))]);
128     }
129     return 0;
130 }
```

4.5 Biconnected Component

```
 1 #include<cstdio>
 2 #include<cstring>
 3 #include<stack>
 4 #include<queue>
 5 #include<algorithm>
 6 
 7 const int MAXN=100000*2;
 8 const int MAXM=200000;
 9 
10 //0-based
11 
12 struct edges
13 {
14     int to,next;
15     bool cut,visit;
16 } edge[MAXM<<1];
17 
18 int head[MAXN],low[MAXN],dpt[MAXN],L;
19 bool visit[MAXN],cut[MAXN];
20 int idx;
21 std::stack<int> st;
22 int bcc[MAXM];
23 
24 void init(int n)
25 {
26     L=0;
```

```
27      memset(head,-1,4*n);
28      memset(visit,0,n);
29  }
30
31  void add_edge(int u,int v)
32  {
33      edge[L].cut=edge[L].visit=false;
34      edge[L].to=v;
35      edge[L].next=head[u];
36      head[u]=L++;
37  }
38
39  void dfs(int u,int fu,int deg)
40  {
41      cut[u]=false;
42      visit[u]=true;
43      low[u]=dpt[u]=deg;
44      int tot=0;
45      for (int i=head[u]; i!=-1; i=edge[i].next)
46      {
47          int v=edge[i].to;
48          if (edge[i].visit)
49              continue;
50          st.push(i/2);
51          edge[i].visit=edge[i^1].visit=true;
52          if (visit[v])
53          {
54              low[u]=dpt[v]>low[u]?low[u]:dpt[v];
55              continue;
56          }
57          dfs(v,u,deg+1);
58          edge[i].cut=edge[i^1].cut=(low[v]>dpt[u] || edge[i].cut);
59          if (u!=fu) cut[u]=low[v]>=dpt[u]?1:cut[u];
60          if (low[v]>=dpt[u] || u==fu)
61          {
62              while (st.top()!=i/2)
63              {
64                  int x=st.top()*2,y=st.top()*2+1;
65                  bcc[st.top()]=idx;
66                  st.pop();
67              }
68              bcc[i/2]=idx++;
69              st.pop();
70          }
71          low[u]=low[v]>low[u]?low[u]:low[v];
72          tot++;
73      }
74      if (u==fu && tot>1)
75          cut[u]=true;
76  }
77
```

```
78 int main()
79 {
80     int n,m;
81     while (scanf("%d%d",&n,&m)!=EOF)
82     {
83         init(n);
84         for (int i=0; i<m; i++)
85         {
86             int u,v;
87             scanf("%d%d",&u,&v);
88             add_edge(u,v);
89             add_edge(v,u);
90         }
91         idx=0;
92         for (int i=0; i<n; i++)
93             if (!visit[i])
94                 dfs(i,i,0);
95     }
96     return 0;
97 }
```

4.6 Blossom algorithm

```
 1 #include<cstdio>
 2 #include<vector>
 3 #include<cstring>
 4 #include<algorithm>
 5
 6 #define MAXX 233
 7
 8 bool map[MAXX][MAXX];
 9 std::vector<int>p[MAXX];
10 int m[MAXX];
11 int vis[MAXX];
12 int q[MAXX],*qf,*qb;
13
14 int n;
15
16 inline void label(int x,int y,int b)
17 {
18     static int i,z;
19     for(i=b+1;i<p[x].size();++i)
20         if(vis[z=p[x][i]]==1)
21         {
22             p[z]=p[y];
23             p[z].insert(p[z].end(),p[x].rbegin(),p[x].rend()-i);
24             vis[z]=0;
25             *qb++=z;
26         }
27 }
28
29 inline bool bfs(int now)
```

```
30 {
31     static int i,x,y,z,b;
32     for(i=0;i<n;++i)
33         p[i].resize(0);
34     p[now].push_back(now);
35     memset(vis,-1,sizeof vis);
36     vis[now]=0;
37     qf=qb=q;
38     *qb++=now;
39 
40     while(qf<qb)
41         for(x=*qf++,y=0;y<n;++y)
42             if(map[x][y] && m[y]!=y && vis[y]!=1)
43             {
44                 if(vis[y]==-1)
45                     if(m[y]==-1)
46                     {
47                         for(i=0;i+1<p[x].size();i+=2)
48                         {
49                             m[p[x][i]]=p[x][i+1];
50                             m[p[x][i+1]]=p[x][i];
51                         }
52                         m[x]=y;
53                         m[y]=x;
54                         return true;
55                     }
56                     else
57                     {
58                         p[z=m[y]]=p[x];
59                         p[z].push_back(y);
60                         p[z].push_back(z);
61                         vis[y]=1;
62                         vis[z]=0;
63                         *qb++=z;
64                     }
65                 else
66                 {
67                     for(b=0;b<p[x].size() && b<p[y].size() && p[x][
                           b]==p[y][b];++b);
68                     --b;
69                     label(x,y,b);
70                     label(y,x,b);
71                 }
72             }
73     return false;
74 }
75 
76 int i,j,k;
77 int ans;
78 
79 int main()
```

```
{
    scanf("%d",&n);
    for(i=0;i<n;++i)
        p[i].reserve(n);
    while(scanf("%d␣%d",&i,&j)!=EOF)
    {
        --i;
        --j;
        map[i][j]=map[j][i]=true;
    }
    memset(m,-1,sizeof m);
    for(i=0;i<n;++i)
        if(m[i]==-1)
        {
            if(bfs(i))
                ++ans;
            else
                m[i]=i;
        }
    printf("%d\n",ans<<1);
    for(i=0;i<n;++i)
        if(i<m[i])
            printf("%d␣%d\n",i+1,m[i]+1);
    return 0;
}
```

4.7 Bridge

```
void dfs(const short &now,const short &fa)
{
    dfn[now]=low[now]=cnt++;
    for(int i(0);i<edge[now].size();++i)
        if(dfn[edge[now][i]]==-1)
        {
            dfs(edge[now][i],now);
            low[now]=std::min(low[now],low[edge[now][i]]);
            if(low[edge[now][i]]>dfn[now]) //如果子节点不能够走到父节点之前去，那么该边为桥
            {
                if(edge[now][i]<now)
                {
                    j=edge[now][i];
                    k=now;
                }
                else
                {
                    j=now;
                    k=edge[now][i];
                }
                ans.push_back(node(j,k));
            }
        }
```

```
        else
            if(edge[now][i]!=fa)
                low[now]=std::min(low[now],low[edge[now][i]]);
}
```

4.8 Chu-Liu:Edmonds' Algorithm

```
#include<cstdio>
#include<cstring>
#include<vector>

#define MAXX 1111
#define MAXE 10111
#define inf 0x3f3f3f3f

int n,m,i,j,k,ans,u,v,tn,rt,sum,on,om;
int pre[MAXX],id[MAXX],in[MAXX],vis[MAXX];

struct edge
{
    int a,b,c;
    edge(){}
    edge(int aa,int bb,int cc):a(aa),b(bb),c(cc){}
};
std::vector<edge>ed(MAXE);

int main()
{
    while(scanf("%d␣%d",&n,&m)!=EOF)
    {
        on=n;
        om=m;
        ed.resize(0);
        sum=1;
        while(m--)
        {
            scanf("%d␣%d␣%d",&i,&j,&k);
            if(i!=j)
            {
                ed.push_back(edge(i,j,k));
                sum+=k;
            }
        }
        ans=0;
        rt=n;
        for(i=0;i<n;++i)
            ed.push_back(edge(n,i,sum));
        ++n;
        while(true)
        {
            memset(in,0x3f,sizeof in);
            for(i=0;i<ed.size();++i)
```

```
46                  if(ed[i].a!=ed[i].b && in[ed[i].b]>ed[i].c)
47                  {
48                      in[ed[i].b]=ed[i].c;
49                      pre[ed[i].b]=ed[i].a;
50                      if(ed[i].a==rt)
51                          j=i;
52                  }
53              for(i=0;i<n;++i)
54                  if(i!=rt && in[i]==inf)
55                      goto ot;
56              memset(id,-1,sizeof id);
57              memset(vis,-1,sizeof vis);
58              tn=in[rt]=0;
59              for(i=0;i<n;++i)
60              {
61                  ans+=in[i];
62                  for(v=i;vis[v]!=i && id[v]==-1 && v!=rt;v=pre[v])
63                      vis[v]=i;
64                  if(v!=rt && id[v]==-1)
65                  {
66                      for(u=pre[v];u!=v;u=pre[u])
67                          id[u]=tn;
68                      id[v]=tn++;
69                  }
70              }
71              if(!tn)
72                  break;
73              for(i=0;i<n;++i)
74                  if(id[i]==-1)
75                      id[i]=tn++;
76              for(i=0;i<ed.size();++i)
77              {
78                  v=ed[i].b;
79                  ed[i].a=id[ed[i].a];
80                  ed[i].b=id[ed[i].b];
81                  if(ed[i].a!=ed[i].b)
82                      ed[i].c-=in[v];
83              }
84              n=tn;
85              rt=id[rt];
86          }
87          if(ans>=2*sum)
88  ot:             puts("impossible");
89          else
90              printf("%d␣%d\n",ans-sum,j-om);
91          puts("");
92      }
93      return 0;
94  }
```

4.9 Count MST

```cpp
//hdu 4408
#include<cstdio>
#include<cstring>
#include<algorithm>

#define MAXX 111

long long mod;
long long a[MAXX][MAXX];

inline long long det(int n)
{
    static int i,j,k;
    static long long re,t;
    for(i=0;i<n;++i)
        for(j=0;j<n;++j)
            a[i][j]%=mod;
    re=1ll;
    for(i=0;i<n;++i)
    {
        for(j=i+1;j<n;++j)
            while(a[j][i])
            {
                t=a[i][i]/a[j][i];
                for(k=i;k<n;++k)
                    a[i][k]=(a[i][k]-a[j][k]*t)%mod;
                for(k=i;k<n;++k)
                    std::swap(a[i][k],a[j][k]);
                re=-re;
            }
        if(!a[i][i])
            return 0ll;
        re=re*a[i][i]%mod;
    }
    return (re+mod)%mod;
}

struct E
{
    int a,b,c;
    bool operator<(const E &i)const
    {
        return c<i.c;
    }
}edge[1111];

int set[2][MAXX];
int find(int a,int t)
{
    return set[t][a]?set[t][a]=find(set[t][a],t):a;
}
```

```
52
53  int id[MAXX],dg[MAXX];
54  int map[MAXX][MAXX];
55  int n,m,i,j,k;
56  long long ans;
57  int cnt;
58
59  int main()
60  {
61      while(scanf("%d␣%d␣%lld",&n,&m,&mod),(n||m||mod))
62      {
63          for(i=0;i<m;++i)
64              scanf("%d␣%d␣%d",&edge[i].a,&edge[i].b,&edge[i].c);
65          std::sort(edge,edge+m);
66          memset(set[0],0,sizeof set[0]);
67          ans=cnt=1;
68          for(i=0;i<m;i=j)
69          {
70              for(j=i;j<m;++j)
71                  if(edge[i].c!=edge[j].c)
72                      break;
73              memset(dg,0,sizeof dg);
74              memset(map,0,sizeof map);
75              memset(set[1],0,sizeof set[0]);
76              static int t,x,y;
77              t=0;
78              for(k=i;k<j;++k)
79              {
80                  x=find(edge[k].a,0);
81                  y=find(edge[k].b,0);
82                  if(x!=y)
83                  {
84                      ++map[x][y];
85                      ++map[y][x];
86                      ++dg[x];
87                      ++dg[y];
88                      x=find(x,1);
89                      y=find(y,1);
90                      if(x!=y)
91                          set[1][x]=y;
92                      ++t;
93                  }
94              }
95              for(k=i;k<j;++k)
96              {
97                  x=find(edge[k].a,0);
98                  y=find(edge[k].b,0);
99                  if(x!=y)
100                 {
101                     ++cnt;
102                     set[0][x]=y;
```

```
                }
            }
            if(t)
            {
                for(k=1;k<=n;++k)
                    if(dg[k] && find(k,1)==k)
                    {
                        memset(a,0,sizeof a);
                        t=0;
                        static int ii,jj;
                        for(ii=1;ii<=n;++ii)
                            if(dg[ii] && find(ii,1)==k)
                                id[ii]=t++;
                        for(ii=1;ii<=n;++ii)
                            if(dg[ii] && find(ii,1)==k)
                            {
                                a[id[ii]][id[ii]]=dg[ii];
                                for(jj=1;jj<=n;++jj)
                                {
                                    if(!dg[jj] || ii==jj || find(jj
                                       ,1)!=k)
                                        continue;
                                    if(map[ii][jj])
                                    {
                                        static long long cnt;
                                        cnt=-map[ii][jj];
                                        a[id[ii]][id[jj]]=(cnt%mod+
                                           mod)%mod;
                                    }
                                }
                            }
                        ans=(ans*det(t-1))%mod;
                    }
            }
        }
        if(cnt!=n)
            puts("0");
        else
            printf("%lld\n",(ans%mod+mod)%mod);
    }
    return 0;
}
```

4.10 Covering problems

```
最大团以及相关知识

独立集：独立集是指图的顶点集的一个子集，该子集的导出子图的点互不相邻．如果一个独立集不是任何一个独立集的子集，那么称这个独立集是一个极大独立集．一个图中包含顶点数目最多的独立集称为最大独立集。最大独立集一定是极大独立集，但是极大独立集不一定是最大的独立集。

```

支配集：与独立集相对应的就是支配集，支配集也是图顶点集的一个子集，设 S 是图 G 的一个支配集，则对于图中的任意一个顶点 u，要么属于集合 s，要么与 s 中的顶点相邻。在 s 中除去任何元素后 s 不再是支配集，则支配集 s 是极小支配集。称 G 的所有支配集中顶点个数最少的支配集为最小支配集，最小支配集中的顶点个数成为支配数。

最小点（对边）的覆盖：最小点的覆盖也是图的顶点集的一个子集，如果我们选中一个点，则称这个点将以他为端点的所有边都覆盖了。将图中所有的边都覆盖所用顶点数最少，这个集合就是最小的点的覆盖。

最大团：图 G 的顶点的子集，设 D 是最大团，则 D 中任意两点相邻。若 u，v 是最大团，则 u,v 有边相连，其补图 u,v 没有边相连，所以图 G 的最大团 = 其补图的最大独立集。给定无向图 G = (V;E)，如果 U 属于 V，并且对于任意 u,v 包含于 U 有 < u; v > 包含于 E，则称 U 是 G 的完全子图，G 的完全子图 U 是 G 的团，当且仅当 U 不包含在 G 的更大的完全子图中，G 的最大团是指 G 中所含顶点数目最多的团。如果 U 属于 V，并且对于任意 u; v 包含于 U 有 < u; v > 不包含于 E，则称 U 是 G 的空子图，G 的空子图 U 是 G 的独立集，当且仅当 U 不包含在 G 的更大的独立集，G 的最大团是指 G 中所含顶点数目最多的独立集。

性质：
最大独立集 + 最小覆盖集 = V
最大团 = 补图的最大独立集
最小覆盖集 = 最大匹配

minimum cover:
vertex cover vertex bipartite graph = maximum cardinality bipartite matching
找完最大二分匹配後，有三種情況要分別處理：
甲、X 側未匹配點的交錯樹們。
乙、Y 側未匹配點的交錯樹們。
丙、層層疊疊的交錯環們（包含單獨的匹配邊）。
這三個情況互不干涉。用 Graph Traversal 建立甲、乙的交錯樹們，剩下部分就是丙。
要找點覆蓋，甲、乙是取盡奇數距離的點，丙是取盡偶數距離的點、或者是取盡奇數距離的點，每塊連通分量可以各自為政。另外，小心處理的話，是可以印出字典順序最小的點覆蓋的。
已經有最大匹配時，求點覆蓋的時間複雜度等同於一次 Graph Traversal 的時間。

vertex cover edge

edge cover vertex
首先在圖上求得一個 Maximum Matching 之後，對於那些單身的點，都由匹配點連過去。如此便形成了 Minimum Edge Cover 。

edge cover edge

path cover vertex
general graph: NP—H
tree: DP
DAG: 将每个节点拆分为入点和出点,ans= 节点数 -匹配数

path cover edge
minimize the count of euler path (greedy is ok?)

```
dg[i] 表示每个点的 id-od,ans = ∑dg[i],∀dg[i] > 0

cycle cover vertex
general: NP—H
weighted: do like path cover vertex, with KM algorithm

cycle cover edge
NP—H
```

4.11 difference constraints

```
for a − b <= c
    add(b,a,c);

最短路得最远解
最长路得最近解
//根据情况反转边?(反转方向及边权)

全 0 点得普通解
```

4.12 Dinitz's algorithm

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

#define MAXX 111
#define MAXM (MAXX*MAXX*4)
#define inf 0x3f3f3f3f

int n;
int w[MAXX],h[MAXX],q[MAXX];
int edge[MAXX],to[MAXM],cap[MAXM],nxt[MAXM],cnt;
int source,sink;

inline void add(int a,int b,int c)
{
    nxt[cnt]=edge[a];
    edge[a]=cnt;
    to[cnt]=b;
    cap[cnt]=c;
    ++cnt;
}

inline bool bfs()
{
    static int *qf,*qb;
    static int i;
    memset(h,-1,sizeof h);
    qf=qb=q;
    h[*qb++=source]=0;
    for(;qf!=qb;++qf)
        for(i=edge[*qf];i!=-1;i=nxt[i])
```

```
32              if(cap[i] && h[to[i]]==-1)
33                  h[*qb++=to[i]]=h[*qf]+1;
34      return h[sink]!=-1;
35  }
36
37  int dfs(int now,int maxcap)
38  {
39      if(now==sink)
40          return maxcap;
41      int flow(maxcap),d;
42      for(int &i(w[now]);i!=-1;i=nxt[i])
43          if(cap[i] && h[to[i]]==h[now]+1)// && (flow=dfs(to[i],std::
                min(maxcap,cap[i])))
44          {
45              d=dfs(to[i],std::min(flow,cap[i]));
46              cap[i]-=d;
47              cap[i^1]+=d;
48              flow-=d;
49              if(!flow)
50                  return maxcap;
51          }
52      return maxcap-flow;
53  }
54
55  int nc,np,m,i,j,k;
56  int ans;
57
58  int main()
59  {
60      while(scanf("%d␣%d␣%d␣%d",&n,&np,&nc,&m)!=EOF)
61      {
62          cnt=0;
63          memset(edge,-1,sizeof edge);
64          while(m--)
65          {
66              while(getchar()!='(');
67              scanf("%d",&i);
68              while(getchar()!=',');
69              scanf("%d",&j);
70              while(getchar()!=')');
71              scanf("%d",&k);
72              if(i!=j)
73              {
74                  ++i;
75                  ++j;
76                  add(i,j,k);
77                  add(j,i,0);
78              }
79          }
80          source=++n;
81          while(np--)
```

```
82          {
83              while(getchar()!='(');
84              scanf("%d",&i);
85              while(getchar()!=')');
86              scanf("%d",&j);
87              ++i;
88              add(source,i,j);
89              add(i,source,0);
90          }
91          sink=++n;
92          while(nc--)
93          {
94              while(getchar()!='(');
95              scanf("%d",&i);
96              while(getchar()!=')');
97              scanf("%d",&j);
98              ++i;
99              add(i,sink,j);
100             add(sink,i,0);
101         }
102         ans=0;
103         while(bfs())
104         {
105             memcpy(w,edge,sizeof edge);
106             ans+=dfs(source,inf);
107             /*
108             while((k=dfs(source,inf)))
109                 ans+=k;
110                 */
111         }
112         printf("%d\n",ans);
113     }
114     return 0;
115 }
```

4.13 Flow network

```
1  Maximum weighted closure of a graph:
2
3  所有由这个子图中的点出发的边都指向这个子图，那么这个子图为原图的一个 closure
       （闭合子图）
4
5  每个节点向其所有依赖节点连边，容量 inf
6  源点向所有正权值节点连边，容量为该权值
7  所有负权值节点向汇点连边，容量为该权值绝对值
8  以上均为有向边
9  最大权为 sum{正权值}-{新图的最小割}
10 残量图中所有由源点可达的点即为所选子图
11
12
13
14 Eulerian circuit:
```

计入度和出度之差
无向边任意定向
出入度之差为奇数则无解
然后构图:
原图有向边不变，容量 1 // 好像需要在新图中忽略有向边?
无向边按之前认定方向，容量 1
源点向所有度数为正的点连边，容量 abs(度数/2)
所有度数为负的点向汇点连边，容量 abs(度数/2)
两侧均满流则有解
相当于规约为可行流问题
注意连通性的 trick

终点到起点加一条有向边即可将 path 问题转为 circuit 问题

Feasible flow problem:
由超级源点出发的边全部满流则有解
有源汇时，由汇点向源点连边，下界 0 上界 inf 即可转化为无源无汇上下界流

对于每条边 <a->b cap{u,d}>，建边 <ss->b cap(u)>、<a->st cap(u)>、<a->b cap(d-u)>

Maximum flow: //好像也可以二分
//将流量还原至原图后，在残量网络上继续完成最大流
直接把 source 和 sink 设为原来的 st，此时输出的最大流即是答案
不需要删除或者调整 t->s 弧
Minimum flow: //好像也可以二分
建图时先不连汇点到源点的边，新图中完成最大流之后再连原汇至原源的边完成第二次最大流，此时 t->s 这条弧的流量即为最小流
判断可行流存在还是必须连原汇 -> 原源的边之后查看满流
所以可以使用跑流 -> 加 ts 弧 -> 跑流，最后检查超级源点满流情况来一步搞定
tips:
合并流量、减少边数来加速

Minimum cost feasible flow problem:
TODO
看起来像是在上面那样跑费用流就行了……

Minimum weighted vertex cover edge for bipartite graph:
for all vertex in X:
edge < s—>x cap(weight(x)) >
for all vertex in Y:
edge < y—>t cap(weight(y)) >
for original edges
edge < x—>y cap(inf) >

```
ans={maximum flow}={minimum cut}
残量网络中的所有简单割 ( (源点可达 && 汇点不可达) || (源点不可达 && 汇点可达) ) 对应着解



Maximum weighted vertex independent set for bipartite graph:
ans=Sum 点权 -valueMinimum weighted vertex cover edge
解应该就是最小覆盖集的补图吧……



方格取数: // refer: hdu 3820 golden eggs
取方格获得收益
当取了相邻方格时付出边的代价

必取的方格到源/汇的边的容量 inf
相邻方格之间的边的容量为 {代价}*2
ans=sum{方格收益}-{最大流}



最小割的唯一性: // refer: 关键边。有向边起点为 s 集，终点为 t 集
从源和汇分别能够到的点集是所有点时，最小割唯一
也就是每一条增广路径都仅有一条边满流
注意查看的是实际的网络，不是残量网络

具体来说

void rr(int now)
{
    done[now]=true;
    ++cnt;
    for(int i(edge[now]);i!=-1;i=nxt[i])
        if(cap[i] && !done[v])
            rr(v);
}

void dfs(int now)
{
    done[now]=true;
    ++cnt;
    for(int i(edge[now]);i!=-1;i=nxt[i])
        if(cap[i^1] && !done[v])
            dfs(v);
}

memset(done,0,sizeof done);
cnt=0;
rr(source);
dfs(sink);
```

```
puts(cnt==n?"UNIQUE":"AMBIGUOUS");



Tips:
两点间可以不止有一种边，也可以不止有一条边，无论有向无向;
两点间容量 inf 则可以设法化简为一个点;
点权始终要转化为边权;
不参与决策的边权设为 inf 来排除掉;
贪心一个初始不合法情况，然后通过可行流调整; // refer: 混合图欧拉回路存在性、有向/无向图中国邮差问题 (遍历所有边至少一次后回到原点)
按时间拆点 (时间层……? );
```

4.14 Hamiltonian circuit

```cpp
//if every point connect with not less than [(N+1)/2] points
#include<cstdio>
#include<algorithm>
#include<cstring>

#define MAXX 177
#define MAX (MAXX*MAXX)

int edge[MAXX],nxt[MAX],to[MAX],cnt;

inline void add(int a,int b)
{
    nxt[++cnt]=edge[a];
    edge[a]=cnt;
    to[cnt]=b;
}

bool done[MAXX];
int n,m,i,j,k;

inline int find(int a)
{
    static int i;
    for(i=edge[a];i;i=nxt[i])
        if(!done[to[i]])
        {
            edge[a]=nxt[i];
            return to[i];
        }
    return 0;
}

int a,b;
int next[MAXX],pre[MAXX];
bool mat[MAXX][MAXX];

int main()
```

```
{
    while(scanf("%d␣%d",&n,&m)!=EOF)
    {
        for(i=1;i<=n;++i)
            next[i]=done[i]=edge[i]=0;
        memset(mat,0,sizeof mat);
        cnt=0;
        while(m--)
        {
            scanf("%d␣%d",&i,&j);
            add(i,j);
            add(j,i);
            mat[i][j]=mat[j][i]=true;
        }
        a=1;
        b=to[edge[a]];
        cnt=2;
        done[a]=done[b]=true;
        next[a]=b;
        while(cnt<n)
        {
            while(i=find(a))
            {
                next[i]=a;
                done[a=i]=true;
                ++cnt;
            }
            while(i=find(b))
            {
                next[b]=i;
                done[b=i]=true;
                ++cnt;
            }
            if(!mat[a][b])
                for(i=next[a];next[i]!=b;i=next[i])
                    if(mat[a][next[i]] && mat[i][b])
                    {
                        for(j=next[i];j!=b;j=next[j])
                            pre[next[j]]=j;
                        for(j=b;j!=next[i];j=pre[j])
                            next[j]=pre[j];
                        std::swap(next[i],b);
                        break;
                    }
            next[b]=a;
            for(i=a;i!=b;i=next[i])
                if(find(i))
                {
                    a=next[b=i];
                    break;
                }
```

```
89          }
90          while(a!=b)
91          {
92              printf("%d␣",a);
93              a=next[a];
94          }
95          printf("%d\n",b);
96      }
97      return 0;
98  }
```

4.15 Hopcroft-Karp algorithm

```
1   #include<cstdio>
2   #include<cstring>
3
4   #define MAXX 50111
5   #define MAX 150111
6
7   int nx,p;
8   int i,j,k;
9   int x,y;
10  int ans;
11  bool flag;
12
13  int edge[MAXX],nxt[MAX],to[MAX],cnt;
14
15  int cx[MAXX],cy[MAXX];
16  int px[MAXX],py[MAXX];
17
18  int q[MAXX],*qf,*qb;
19
20  bool ag(int i)
21  {
22      int j,k;
23      for(k=edge[i];k;k=nxt[k])
24          if(py[j=to[k]]==px[i]+1)
25          {
26              py[j]=0;
27              if(cy[j]==-1 || ag(cy[j]))
28              {
29                  cx[i]=j;
30                  cy[j]=i;
31                  return true;
32              }
33          }
34      return false;
35  }
36
37  int main()
38  {
39      scanf("%d␣%*d␣%d",&nx,&p);
```

```
    while(p--)
    {
        scanf("%d %d",&i,&j);
        nxt[++cnt]=edge[i];
        edge[i]=cnt;
        to[cnt]=j;
    }
    memset(cx,-1,sizeof cx);
    memset(cy,-1,sizeof cy);
    while(true)
    {
        memset(px,0,sizeof(px));
        memset(py,0,sizeof(py));
        qf=qb=q;
        flag=false;

        for(i=1;i<=nx;++i)
            if(cx[i]==-1)
                *qb++=i;
        while(qf!=qb)
            for(k=edge[i=*qf++];k;k=nxt[k])
                if(!py[j=to[k]])
                {
                    py[j]=px[i]+1;
                    if(cy[j]==-1)
                        flag=true;
                    else
                    {
                        px[cy[j]]=py[j]+1;
                        *qb++=cy[j];
                    }
                }
        if(!flag)
            break;
        for(i=1;i<=nx;++i)
            if(cx[i]==-1 && ag(i))
                ++ans;
    }
    printf("%d\n",ans);
    return 0;
}
```

4.16 Improved Shortest Augmenting Path Algorithm

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

#define MAXX 5111
#define MAXM (30111*4)
#define inf 0x3f3f3f3f3f3f3f3fll

```

```
 9 int edge[MAXX],to[MAXM],nxt[MAXM],cnt;
10 #define v to[i]
11 long long cap[MAXM];
12
13 int n;
14 int h[MAXX],gap[MAXX],pre[MAXX],w[MAXX];
15
16 inline void add(int a,int b,long long c)
17 {
18     nxt[++cnt]=edge[a];
19     edge[a]=cnt;
20     to[cnt]=b;
21     cap[cnt]=c;
22 }
23
24 int source,sink;
25
26 inline long long go(const int N=sink)
27 {
28     static int now,i;
29     static long long min,mf;
30     memset(gap,0,sizeof gap);
31     memset(h,0,sizeof h);
32     memcpy(w,edge,sizeof w);
33     gap[0]=N;
34     mf=0;
35
36     pre[now=source]=-1;
37     while(h[source]<N)
38     {
39 rep:
40         if(now==sink)
41         {
42             min=inf;
43             for(i=pre[sink];i!=-1;i=pre[to[i^1]])
44                 if(min>=cap[i])
45                 {
46                     min=cap[i];
47                     now=to[i^1];
48                 }
49             for(i=pre[sink];i!=-1;i=pre[to[i^1]])
50             {
51                 cap[i]-=min;
52                 cap[i^1]+=min;
53             }
54             mf+=min;
55         }
56         for(int &i(w[now]);i!=-1;i=nxt[i])
57             if(cap[i] && h[v]+1==h[now])
58             {
59                 pre[now=v]=i;
```

```
60                  goto rep;
61              }
62          if(!--gap[h[now]])
63              return mf;
64          min=N;
65          for(i=w[now]=edge[now];i!=-1;i=nxt[i])
66              if(cap[i])
67                  min=std::min(min,(long long)h[v]);
68          ++gap[h[now]=min+1];
69          if(now!=source)
70              now=to[pre[now]^1];
71      }
72      return mf;
73  }
74
75  int m,i,j,k;
76  long long ans;
77
78  int main()
79  {
80      scanf("%d %d",&n,&m);
81      source=1;
82      sink=n;
83      cnt=-1;
84      memset(edge,-1,sizeof edge);
85      while(m--)
86      {
87          scanf("%d %d %lld",&i,&j,&ans);
88          add(i,j,ans);
89          add(j,i,ans);
90      }
91      printf("%lld\n",go());
92      return 0;
93  }
```

4.17 k Shortest Path

```
1   #include<cstdio>
2   #include<cstring>
3   #include<queue>
4   #include<vector>
5
6   int K;
7
8   class states
9   {
10      public:
11          int cost,id;
12  };
13
14  int dist[1000];
15
```

```
16 class cmp
17 {
18     public:
19         bool operator ()(const states &i,const states &j)
20         {
21             return i.cost>j.cost;
22         }
23 };
24
25 class cmp2
26 {
27     public:
28         bool operator ()(const states &i,const states &j)
29         {
30             return i.cost+dist[i.id]>j.cost+dist[j.id];
31         }
32 };
33
34 struct edges
35 {
36     int to,next,cost;
37 } edger[100000],edge[100000];
38
39 int headr[1000],head[1000],Lr,L;
40
41 void dijkstra(int s)
42 {
43     states u;
44     u.id=s;
45     u.cost=0;
46     dist[s]=0;
47     std::priority_queue<states,std::vector<states>,cmp> q;
48     q.push(u);
49     while (!q.empty())
50     {
51         u=q.top();
52         q.pop();
53         if (u.cost!=dist[u.id])
54             continue;
55         for (int i=headr[u.id]; i!=-1; i=edger[i].next)
56         {
57             states v=u;
58             v.id=edger[i].to;
59             if (dist[v.id]>dist[u.id]+edger[i].cost)
60             {
61                 v.cost=dist[v.id]=dist[u.id]+edger[i].cost;
62                 q.push(v);
63             }
64         }
65     }
66 }
```

```
67
68 int num[1000];
69
70 inline void init(int n)
71 {
72     Lr=L=0;
73     memset(head,-1,4*n);
74     memset(headr,-1,4*n);
75     memset(dist,63,4*n);
76     memset(num,0,4*n);
77 }
78
79 void add_edge(int u,int v,int x)
80 {
81     edge[L].to=v;
82     edge[L].cost=x;
83     edge[L].next=head[u];
84     head[u]=L++;
85     edger[Lr].to=u;
86     edger[Lr].cost=x;
87     edger[Lr].next=headr[v];
88     headr[v]=Lr++;
89 }
90
91 inline int a_star(int s,int t)
92 {
93     if (dist[s]==0x3f3f3f3f)
94         return -1;
95     std::priority_queue<states,std::vector<states>,cmp2> q;
96     states tmp;
97     tmp.id=s;
98     tmp.cost=0;
99     q.push(tmp);
100    while (!q.empty())
101    {
102        states u=q.top();
103        q.pop();
104        num[u.id]++;
105        if (num[t]==K)
106            return u.cost;
107        for (int i=head[u.id]; i!=-1; i=edge[i].next)
108        {
109            int v=edge[i].to;
110            tmp.id=v;
111            tmp.cost=u.cost+edge[i].cost;
112            q.push(tmp);
113        }
114    }
115    return -1;
116 }
117
```

```
118 int main()
119 {
120     int n,m;
121     scanf("%d%d",&n,&m);
122     init(n);
123     for (int i=0; i<m; i++)
124     {
125         int u,v,x;
126         scanf("%d%d%d",&u,&v,&x);
127         add_edge(u-1,v-1,x);
128     }
129     int s,t;
130     scanf("%d%d%d",&s,&t,&K);
131     if (s==t)
132         ++K;
133     dijkstra(t-1);
134     printf("%d\n",a_star(s-1,t-1));
135     return 0;
136 }
```

4.18 Kariv-Hakimi Algorithm

```
 1 //Absolute Center of a graph, not only a tree
 2 #include<cstdio>
 3 #include<algorithm>
 4 #include<vector>
 5 #include<cstring>
 6 #include<set>
 7
 8 #define MAXX 211
 9 #define inf 0x3f3f3f3f
10
11 int e[MAXX][MAXX],dist[MAXX][MAXX];
12 double dp[MAXX],ta;
13 int ans,d;
14 int n,m,a,b;
15 int i,j,k;
16 typedef std::pair<int,int> pii;
17 std::vector<pii>vt[2];
18 bool done[MAXX];
19 typedef std::pair<double,int> pdi;
20 std::multiset<pdi>q;
21 int pre[MAXX];
22
23 int main()
24 {
25     vt[0].reserve(MAXX);
26     vt[1].reserve(MAXX);
27     scanf("%d␣%d",&n,&m);
28     memset(e,0x3f,sizeof(e));
29     while(m--)
30     {
```

```
31          scanf("%d␣%d␣%d",&i,&j,&k);
32          e[i][j]=e[j][i]=std::min(e[i][j],k);
33      }
34      for(i=1;i<=n;++i)
35          e[i][i]=0;
36      memcpy(dist,e,sizeof(dist));
37      for(k=1;k<=n;++k)
38          for(i=1;i<=n;++i)
39              for(j=1;j<=n;++j)
40                  dist[i][j]=std::min(dist[i][j],dist[i][k]+dist[k][j
                        ]);
41      ans=inf;
42      for(i=1;i<=n;++i)
43          for(j=i;j<=n;++j)
44              if(e[i][j]!=inf)
45              {
46                  vt[0].resize(0);
47                  vt[1].resize(0);
48                  static int i;
49                  for(i=1;i<=n;++i)
50                      vt[0].push_back(pii(dist[::i][i],dist[j][i]));
51                  std::sort(vt[0].begin(),vt[0].end());
52                  for(i=0;i<vt[0].size();++i)
53                  {
54                      while(!vt[1].empty() && vt[1].back().second<=vt
                            [0][i].second)
55                          vt[1].pop_back();
56                      vt[1].push_back(vt[0][i]);
57                  }
58                  d=inf;
59                  if(vt[1].size()==1)
60                      if(vt[1][0].first<vt[1][0].second)
61                      {
62                          ta=0;
63                          d=(vt[1][0].first<<1);
64                      }
65                      else
66                      {
67                          ta=e[::i][j];
68                          d=(vt[1][0].second<<1);
69                      }
70                  else
71                      for(i=1;i<vt[1].size();++i)
72                          if(d>e[::i][j]+vt[1][i-1].first+vt[1][i].
                                second)
73                          {
74                              ta=(e[::i][j]+vt[1][i].second-vt[1][i
                                    -1].first)/(double)2.0f;
75                              d=e[::i][j]+vt[1][i-1].first+vt[1][i].
                                    second;
76                          }
```

```
77                  if(d<ans)
78                  {
79                      ans=d;
80                      a=::i;
81                      b=j;
82                      dp[::i]=ta;
83                      dp[j]=e[::i][j]-ta;
84                  }
85              }
86      printf("%d\n",ans);
87      for(i=1;i<=n;++i)
88          if(i!=a && i!=b)
89              dp[i]=1e20;
90      q.insert(pdi(dp[a],a));
91      if(a!=b)
92          q.insert(pdi(dp[b],b));
93      if(a!=b)
94          pre[b]=a;
95      while(!q.empty())
96      {
97          k=q.begin()->second;
98          q.erase(q.begin());
99          if(done[k])
100             continue;
101         done[k]=true;
102         for(i=1;i<=n;++i)
103             if(e[k][i]!=inf && dp[k]+e[k][i]<dp[i])
104             {
105                 dp[i]=dp[k]+e[k][i];
106                 q.insert(pdi(dp[i],i));
107                 pre[i]=k;
108             }
109     }
110     vt[0].resize(0);
111     for(i=1;i<=n;++i)
112         if(pre[i])
113             if(i<pre[i])
114                 printf("%d␣%d\n",i,pre[i]);
115             else
116                 printf("%d␣%d\n",pre[i],i);
117     return 0;
118 }
```

4.19 Kuhn-Munkres algorithm

```
1  bool match(int u)//匈牙利
2  {
3      vx[u]=true;
4      for(int i=1;i<=n;++i)
5          if(lx[u]+ly[i]==g[u][i]&&!vy[i])
6          {
7              vy[i]=true;
```

```
 8            if(!d[i]||match(d[i]))
 9            {
10                d[i]=u;
11                return true;
12            }
13        }
14    return false;
15}
16inline void update()//
17{
18    int i,j;
19    int a=1<<30;
20    for(i=1;i<=n;++i)if(vx[i])
21        for(j=1;j<=n;++j)if(!vy[j])
22            a=min(a,lx[i]+ly[j]-g[i][j]);
23    for(i=1;i<=n;++i)
24    {
25        if(vx[i])lx[i]-=a;
26        if(vy[i])ly[i]+=a;
27    }
28}
29void km()
30{
31    int i,j;
32    for(i=1;i<=n;++i)
33    {
34        lx[i]=ly[i]=d[i]=0;
35        for(j=1;j<=n;++j)
36            lx[i]=max(lx[i],g[i][j]);
37    }
38    for(i=1;i<=n;++i)
39    {
40        while(true)
41        {
42            memset(vx,0,sizeof(vx));
43            memset(vy,0,sizeof(vy));
44            if(match(i))
45                break;
46            update();
47        }
48    }
49    int ans=0;
50    for(i=1;i<=n;++i)
51        if(d[i]!=0)
52            ans+=g[d[i]][i];
53    printf("%d\n",ans);
54}
55int main()
56{
57    while(scanf("%d\n",&n)!=EOF)
58    {
```

```
59          for(int i=1;i<=n;++i)gets(s[i]);
60          memset(g,0,sizeof(g));
61          for(int i=1;i<=n;++i)
62              for(int j=1;j<=n;++j)
63                  if(i!=j) g[i][j]=cal(s[i],s[j]);
64          km();
65      }
66      return 0;
67  }
68
69
70  //bupt
71
72  //算法：求二分图最佳匹配km n复杂度^3
73  int dfs(int u)//匈牙利求增广路
74  {
75      int v;
76      sx[u]=1;
77      for ( v=1; v<=n; v++)
78          if (!sy[v] && lx[u]+ly[v]==map[u][v])
79          {
80              sy[v]=1;
81              if (match[v]==-1 || dfs(match[v]))
82              {
83                  match[v]=u;
84                  return 1;
85              }
86          }
87      return 0;
88  }
89
90  int bestmatch(void)//求最佳匹配km
91  {
92      int i,j,u;
93      for (i=1; i<=n; i++)//初始化顶标
94      {
95          lx[i]=-1;
96          ly[i]=0;
97          for (j=1; j<=n; j++)
98              if (lx[i]<map[i][j])
99                  lx[i]=map[i][j];
100     }
101     memset(match, -1, sizeof(match));
102     for (u=1; u<=n; u++)
103     {
104         while (true)
105         {
106             memset(sx,0,sizeof(sx));
107             memset(sy,0,sizeof(sy));
108             if (dfs(u))
109                 break;
```

```
            int dx=Inf;//若找不到增广路，则修改顶标~~
            for (i=1; i<=n; i++)
            {
                if (sx[i])
                    for (j=1; j<=n; j++)
                        if(!sy[j] && dx>lx[i]+ly[j]-map[i][j])
                            dx=lx[i]+ly[j]-map[i][j];
            }
            for (i=1; i<=n; i++)
            {
                if (sx[i])
                    lx[i]-=dx;
                if (sy[i])
                    ly[i]+=dx;
            }
        }
    }
    int sum=0;
    for (i=1; i<=n; i++)
        sum+=map[match[i]][i];
    return sum;
}
```

4.20 LCA - DA

```
int edge[MAXX],nxt[MAXX<<1],to[MAXX<<1],cnt;
int pre[MAXX][N],dg[MAXX];

inline void add(int j,int k)
{
    nxt[++cnt]=edge[j];
    edge[j]=cnt;
    to[cnt]=k;
}

void rr(int now,int fa)
{
    dg[now]=dg[fa]+1;
    for(int i(edge[now]);i;i=nxt[i])
        if(to[i]!=fa)
        {
            static int j;
            j=1;
            for(pre[to[i]][0]=now;j<N;++j)
                pre[to[i]][j]=pre[pre[to[i]][j-1]][j-1];
            rr(to[i],now);
        }
}

inline int lca(int a,int b)
{
    static int i,j;
```

```
28      j=0;
29      if(dg[a]<dg[b])
30          std::swap(a,b);
31      for(i=dg[a]-dg[b];i;i>>=1,++j)
32          if(i&1)
33              a=pre[a][j];
34      if(a==b)
35          return a;
36      for(i=N-1;i>=0;--i)
37          if(pre[a][i]!=pre[b][i])
38          {
39              a=pre[a][i];
40              b=pre[b][i];
41          }
42      return pre[a][0];
43
44  // looks like above is a wrong version
45
46      static int i,log;
47      for(log=0;(1<<(log+1))<=dg[a];++log);
48      for(i=log;i>=0;--i)
49          if(dg[a]-(1<<i)>=dg[b])
50              a=pre[a][i];
51      if(a==b)
52          return a;
53      for(i=log;i>=0;--i)
54          if(pre[a][i]!=-1 && pre[a][i]!=pre[b][i])
55              a=pre[a][i],b=pre[b][i];
56      return pre[a][0];
57  }
```

4.21 LCA - tarjan - minmax

```
1   #include<cstdio>
2   #include<list>
3   #include<algorithm>
4   #include<cstring>
5
6   #define MAXX 100111
7   #define inf 0x5fffffff
8
9   short T,t;
10  int set[MAXX],min[MAXX],max[MAXX],ans[2][MAXX];
11  bool done[MAXX];
12  std::list<std::pair<int,int> >edge[MAXX];
13  std::list<std::pair<int,int> >q[MAXX];
14  int n,i,j,k,l,m;
15
16  struct node
17  {
18      int a,b,id;
19      node() {}
```

```
20      node(const int &aa,const int &bb,const int &idd): a(aa),b(bb),
            id(idd){}
21  };
22
23  std::list<node>to[MAXX];
24
25  int find(const int &a)
26  {
27      if(set[a]==a)
28          return a;
29      int b(set[a]);
30      set[a]=find(set[a]);
31      max[a]=std::max(max[a],max[b]);
32      min[a]=std::min(min[a],min[b]);
33      return set[a];
34  }
35
36  void tarjan(const int &now)
37  {
38      done[now]=true;
39      for(std::list<std::pair<int,int> >::const_iterator it(q[now].
            begin());it!=q[now].end();++it)
40          if(done[it->first])
41              if(it->second>0)
42                  to[find(it->first)].push_back(node(now,it->first,it
                        ->second));
43              else
44                  to[find(it->first)].push_back(node(it->first,now,-
                        it->second));
45      for(std::list<std::pair<int,int> >::const_iterator it(edge[now
            ].begin());it!=edge[now].end();++it)
46          if(!done[it->first])
47          {
48              tarjan(it->first);
49              set[it->first]=now;
50              min[it->first]=it->second;
51              max[it->first]=it->second;
52          }
53      for(std::list<node>::const_iterator it(to[now].begin());it!=to[
            now].end();++it)
54      {
55          find(it->a);
56          find(it->b);
57          ans[0][it->id]=std::min(min[it->b],min[it->a]);
58          ans[1][it->id]=std::max(max[it->a],max[it->b]);
59      }
60  }
61
62  int main()
63  {
64      scanf("%hd",&T);
```

```
65      for(t=1;t<=T;++t)
66      {
67          scanf("%d",&n);
68          for(i=1;i<=n;++i)
69          {
70              edge[i].clear();
71              q[i].clear();
72              to[i].clear();
73              done[i]=false;
74              set[i]=i;
75              min[i]=inf;
76              max[i]=0;
77          }
78          for(i=1;i<n;++i)
79          {
80              scanf("%d%d%d",&j,&k,&l);
81              edge[j].push_back(std::make_pair(k,l));
82              edge[k].push_back(std::make_pair(j,l));
83          }
84          scanf("%d",&m);
85          for(i=0;i<m;++i)
86          {
87              scanf("%d␣%d",&j,&k);
88              q[j].push_back(std::make_pair(k,i));
89              q[k].push_back(std::make_pair(j,-i));
90          }
91          tarjan(1);
92          printf("Case␣%hd:\n",t);
93          for(i=0;i<m;++i)
94              printf("%d␣%d\n",ans[0][i],ans[1][i]);
95      }
96      return 0;
97  }
```

4.22 Minimum Ratio Spanning Tree

```
1   #include<cstdio>
2   #include<cstring>
3   #include<cmath>
4
5   #define MAXX 1111
6
7   struct
8   {
9       int x,y;
10      double z;
11  } node[MAXX];
12
13  struct
14  {
15      double l,c;
16  } map[MAXX][MAXX];
```

```
17
18 int n,l,f[MAXX],pre[MAXX];
19 double dis[MAXX];
20
21 double mst(double x)
22 {
23     int i,j,tmp;
24     double min,s=0,t=0;
25     memset(f,0,sizeof(f));
26     f[1]=1;
27     for (i=2; i<=n; i++)
28     {
29         dis[i]=map[1][i].c-map[1][i].l*x;
30         pre[i]=1;
31     }
32     for (i=1; i<n; i++)
33     {
34         min=1e10;
35         for (j=1; j<=n; j++)
36             if (!f[j] && min>dis[j])
37             {
38                 min=dis[j];
39                 tmp=j;
40             }
41         f[tmp]=1;
42         t+=map[pre[tmp]][tmp].l;
43         s+=map[pre[tmp]][tmp].c;
44         for (j=1; j<=n; j++)
45             if (!f[j] && map[tmp][j].c-map[tmp][j].l*x<dis[j])
46             {
47                 dis[j]=map[tmp][j].c-map[tmp][j].l*x;
48                 pre[j]=tmp;
49             }
50     }
51     return s/t;
52 }
53
54 int main()
55 {
56     int i,j;
57     double a,b;
58     while (scanf("%d",&n),n);
59     {
60         for (i=1; i<=n; i++)
61             scanf("%d%d%lf",&node[i].x,&node[i].y,&node[i].z);
62         for (i=1; i<=n; i++)
63             for (j=i+1; j<=n; j++)
64             {
65                 map[j][i].l=map[i][j].l=sqrt(1.0*(node[i].x-node[j
                       ].x)*(node[i].x-node[j].x)+(node[i].y-node[j].y)
                       *(node[i].y-node[j].y));
```

```
66                  map[j][i].c=map[i][j].c=fabs(node[i].z-node[j].z);
67              }
68          a=0,b=mst(a);
69          while (fabs(b-a)>1e-8)
70          {
71              a=b;
72              b=mst(a);
73          }
74          printf("%.3lf\n",b);
75      }
76      return 0;
77
78  }
```

4.23 Minimum Steiner Tree

```
1   #include<cstdio>
2   #include<cstring>
3   #include<algorithm>
4   #include<queue>
5
6   #define MAXX 211
7   #define MAXE 10111
8   #define inf 0x3f3f3f3f
9
10  int edge[MAXX],nxt[MAXE],to[MAXE],wg[MAXE],cnt;
11  inline void add(int a,int b,int c)
12  {
13      nxt[++cnt]=edge[a];
14      edge[a]=cnt;
15      to[cnt]=b;
16      wg[cnt]=c;
17  }
18
19  int dp[1<<8];
20  int s[MAXX];
21  int d[1<<8][MAXX];
22  int S[MAXX],P[MAXX];
23  int fac[8];
24
25  struct node
26  {
27      int a,b,dist;
28      node(){}
29      node(int i,int j,int k):a(i),b(j),dist(k){}
30      bool operator<(const node &i)const
31      {
32          return dist>i.dist;
33      }
34      int &get()
35      {
36          return d[b][a];
```

```
37      }
38  }now;
39
40  std::priority_queue<node>q;
41
42  int n,m,nn,i,j,k;
43  int cs,cf,x,y;
44  int ans,cst;
45
46  inline bool check(int x)
47  {
48      static int re,i;
49      for(i=re=0;x;x>>=1,++i)
50          re+=(x&1)*(i<cf?fac[i]:-1);
51      return re>=0;
52  }
53
54  inline int count(int x)
55  {
56      static int i,re;
57      x>>=cf;
58      for(re=0;x;x>>=1)
59          re+=(x&1);
60      return re;
61  }
62
63  int main()
64  {
65      while(scanf("%d",&n)!=EOF)
66      {
67          memset(s,0,sizeof s);
68          memset(d,0x3f,sizeof d);
69          memset(dp,0x3f,sizeof dp);
70          ans=cnt=cf=cs=0;
71          memset(edge,0,sizeof edge);
72          for(i=1;i<=n;++i)
73          {
74              scanf("%d␣%d",P+i,S+i);
75              if(S[i] && P[i])
76              {
77                  ++ans;
78                  --P[i];
79                  S[i]=0;
80              }
81              if(P[i])
82              {
83                  s[i]=1<<cf;
84                  fac[cf]=P[i];
85                  d[s[i]][i]=0;
86                  ++cf;
87              }
```

```
88          }
89          for(i=1;i<=n;++i)
90              if(S[i])
91              {
92                  s[i]=1<<(cf+cs);
93                  d[s[i]][i]=0;
94                  ++cs;
95              }
96          nn=1<<(cf+cs);
97          scanf("%d",&m);
98          while(m--)
99          {
100             scanf("%d␣%d␣%d",&i,&j,&k);
101             add(i,j,k);
102             add(j,i,k);
103         }
104         for(y=1;y<nn;++y)
105         {
106             for(x=1;x<=n;++x)
107             {
108                 if(s[x] && !(s[x]&y))
109                     continue;
110                 for(i=(y-1)&y;i;i=(i-1)&y)
111                     d[y][x]=std::min(d[y][x],d[i|s[x]][x]+d[(y^i)|s
                            [x]][x]);
112                 if(d[y][x]!=inf)
113                     q.push(node(x,y,d[y][x]));
114             }
115             while(!q.empty())
116             {
117                 now=q.top();
118                 q.pop();
119                 if(now.dist!=now.get())
120                     continue;
121                 static int x,y,a,b;
122                 x=now.a;
123                 y=now.b;
124                 for(i=edge[x];i;i=nxt[i])
125                 {
126                     a=to[i];
127                     b=y|s[a];
128                     if(d[b][a]>now.get()+wg[i])
129                     {
130                         d[b][a]=now.get()+wg[i];
131                         if(b==y)
132                             q.push(node(a,b,d[b][a]));
133                     }
134                 }
135             }
136         }
137         for(j=0;j<nn;++j)
```

```
138             dp[j]=*std::min_element(d[j]+1,d[j]+1+n);
139         cnt=cst=0;
140         for(i=1;i<nn;++i)
141             if(check(i))
142             {
143                 for(j=(i-1)&i;j;j=(j-1)&i)
144                     if(check(j) && check(i^j))
145                         dp[i]=std::min(dp[i],dp[j]+dp[i^j]);
146                 k=count(i);
147                 if(dp[i]!=inf && (k>cnt || (k==cnt && dp[i]<cst)))
148                 {
149                     cnt=k;
150                     cst=dp[i];
151                 }
152             }
153         printf("%d %d\n",ans+cnt,cst);
154     }
155     return 0;
156 }
```

4.24 Minimum-cost flow problem

```
1  // like Edmonds-Karp Algorithm
2  #include<cstdio>
3  #include<cstring>
4  #include<algorithm>
5  #include<queue>
6
7  #define MAXX 5011
8  #define MAXE (MAXX*10*2)
9  #define inf 0x3f3f3f3f
10
11 int edge[MAXX],nxt[MAXE],to[MAXE],cap[MAXE],cst[MAXE],cnt;
12 #define v to[i]
13 inline void adde(int a,int b,int c,int d)
14 {
15     nxt[++cnt]=edge[a];
16     edge[a]=cnt;
17     to[cnt]=b;
18     cap[cnt]=c;
19     cst[cnt]=d;
20 }
21 inline void add(int a,int b,int c,int d)
22 { adde(a,b,c,d);adde(b,a,0,-d);}
23
24 int dist[MAXX],pre[MAXX];
25 int source,sink;
26 std::queue<int>q;
27 bool in[MAXX];
28
29 inline bool go()
30 {
```

```
    static int now,i;
    memset(dist,0x3f,sizeof dist);
    dist[source]=0;
    pre[source]=-1;
    q.push(source);
    in[source]=true;
    while(!q.empty())
    {
        in[now=q.front()]=false;
        q.pop();
        for(i=edge[now];i!=-1;i=nxt[i])
            if(cap[i] && dist[v]>dist[now]+cst[i])
            {
                dist[v]=dist[now]+cst[i];
                pre[v]=i;
                if(!in[v])
                {
                    q.push(v);
                    in[v]=true;
                }
            }
    }
    return dist[sink]!=inf;
}

inline int mcmf(int &flow)
{
    static int ans,i;
    flow=ans=0;
    while(go())
    {
        static int min;
        min=inf;
        for(i=pre[sink];i!=-1;i=pre[to[i^1]])
            min=std::min(min,cap[i]);
        flow+=min;
        ans+=min*dist[sink];
        for(i=pre[sink];i!=-1;i=pre[to[i^1]])
        {
            cap[i]-=min;
            cap[i^1]+=min;
        }
    }
    return ans;
}
```

4.25 Second-best MST

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

```

```
 5 #define MAXN 511
 6 #define MAXM 2500111
 7 #define v to[i]
 8
 9 int set[MAXN];
10 int find(int a)
11 {
12     return set[a]?set[a]=find(set[a]):a;
13 }
14
15 int n,m,i,j,k,ans;
16
17 struct edge
18 {
19     int a,b,c;
20     bool in;
21     bool operator<(const edge &i)const
22     {
23         return c<i.c;
24     }
25 }ed[MAXM];
26
27 int map[MAXN][MAXN];
28 bool done[MAXN];
29
30 int head[MAXN],to[MAXN<<1],nxt[MAXN<<1],wg[MAXN<<1],cnt;
31 inline void add(int a,int b,int c)
32 {
33     nxt[++cnt]=head[a];
34     head[a]=cnt;
35     to[cnt]=b;
36     wg[cnt]=c;
37 }
38
39 void dfs(const int now,const int fa)
40 {
41     done[now]=true;
42     for(int i(head[now]);i;i=nxt[i])
43         if(v!=fa)
44         {
45             for(int j(1);j<=n;++j)
46                 if(done[j])
47                     map[v][j]=map[j][v]=std::max(map[j][now],wg[i])
                        ;
48             dfs(v,now);
49         }
50 }
51
52 int main()
53 {
54     scanf("%d␣%d",&n,&m);
```

```
    for(i=0;i<m;++i)
        scanf("%d␣%d␣%d",&ed[i].a,&ed[i].b,&ed[i].c);
    std::sort(ed,ed+m);
    for(i=0;i<m;++i)
        if(find(ed[i].a)!=find(ed[i].b))
        {
            j+=ed[i].c;
            ++k;
            set[find(ed[i].a)]=find(ed[i].b);
            ed[i].in=true;
            add(ed[i].a,ed[i].b,ed[i].c);
            add(ed[i].b,ed[i].a,ed[i].c);
        }
    if(k+1!=n)
        puts("Cost:␣-1\nCost:␣-1");
    else
    {
        printf("Cost:␣%d\n",j);
        if(m==n-1)
        {
            puts("Cost:␣-1");
            return 0;
        }
        ans=0x3f3f3f3f;
        memset(map,0x3f,sizeof map);
        for(i=1;i<=n;++i)
            map[i][i]=0;
        dfs(1,0);
        for(i=0;i<m;++i)
            if(!ed[i].in)
                ans=std::min(ans,j+ed[i].c-map[ed[i].a][ed[i].b]);
        printf("Cost:␣%d\n",ans);
    }
    return 0;
}
```

4.26 Spanning tree

```
Minimum Bottleneck Spanning Tree:
Kruscal

All-pairs vertexes' Minimum Bottleneck Path:
DP in the Kruscal's MST
O(n^2)*O(1)

Minimum Diameter Spanning Tree:
Kariv-Hakimi Algorithm

Directed MST:-
ChuLiu/Edmonds' Algorithm

Second-best MST:
```

```
get All-pairs vertexes' Minimum Bottleneck Path, then enumerate all no-tree-edges to replace the longest edge between two vertexes to get a worse MST

Degree-constrained MST:
remove the vertex from the whole graph,then add edges to increase degrees and connect different connected components together ( O(mlogm + n) with kruscal )
if we can't connect all connected components together, there exists no any spanning tree
next step is add edges to root vertex greedily, increase degrees, and decrease our answer ( O(k*n) )
need all vertexes' minimum bottleneck path to root vertex

Minimum Ratio Spanning Tree:
Binary search

Manhattan MST:
combining line sweep with divide-and-conquer algorithm

Minimum Steiner Tree:
the MST contain all k vertexes
bit-mask with dijkstra O( (1<<k)*( {dijkstra} ) )
then run a bit-mask DP( O( n*(1<<k) ) )

Count Spanning Trees:
Kirchhoff's theorem
simply calculate the minor of (degree Matrix - edge Matrix)

k-best MST:
do like second-best MST for k times
```

4.27 Stable Marriage

```
//对于每个预备队列中的对象，及被匹配对象，先按照喜好程度排列匹配对象

while(!g.empty())  // 预备匹配队列
{
    if(dfn[edge[g.front()].front()]==-1)
        dfn[edge[g.front()].front()]=g.front(); // 如果目前还没尝试匹配过的对象没有被任何别的对象占据
    else
    {
        for(it=edge[edge[g.front()].front()].begin();it!=edge[edge[g.front()].front()].end();++it)
            if(*it==dfn[edge[g.front()].front()] || *it==g.front())
                //如果被匹配对象更喜欢正在被匹配的人或现在准备匹配的对象
                break;
        if(*it==g.front()) //如果更喜欢新的
        {
            g.push_back(dfn[edge[g.front()].front()]);
```

```
            dfn[edge[g.front()].front()]=g.front();
        }
        else
            g.push_back(g.front()); //否则放到队尾，重新等待匹配
    }
    edge[g.front()].pop_front(); //每组匹配最多只考虑一次
    g.pop_front();
}
```

4.28 Stoer-Wagner Algorithm

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

const int maxn=510;

int map[maxn][maxn];
int n;

void contract(int x,int y)//合并两个点
{
    int i,j;
    for (i=0; i<n; i++)
        if (i!=x)
        {
            map[x][i]+=map[y][i];
            map[i][x]+=map[i][y];
        }
    for (i=y+1; i<n; i++)
        for (j=0; j<n; j++)
        {
            map[i-1][j]=map[i][j];
            map[j][i-1]=map[j][i];
        }
    n--;
}

int w[maxn],c[maxn];
int sx,tx;

int mincut() //求最大生成树，计算最后一个点的割，并保存最后一条边的两个顶点
{
    static int i,j,k,t;
    memset(c,0,sizeof(c));
    c[0]=1;
    for (i=0; i<n; i++)
        w[i]=map[0][i];
    for (i=1; i+1<n; i++)
    {
        t=k=-1;
        for (j=0; j<n; j++)
            if (c[j]==0&&w[j]>k)
```

```
42                  k=w[t=j];
43          c[sx=t]=1;
44          for (j=0; j<n; j++)
45              w[j]+=map[t][j];
46      }
47      for (i=0; i<n; i++)
48          if (c[i]==0)
49              return w[tx=i];
50  }
51  int main()
52  {
53      int i,j,k,m;
54      while (scanf("%d%d",&n,&m)!=EOF)
55      {
56          memset(map,0,sizeof(map));
57          while (m--)
58          {
59              scanf("%d%d%d",&i,&j,&k);
60              map[i][j]+=k;
61              map[j][i]+=k;
62          }
63          int mint=999999999;
64          while (n>1)
65          {
66              k=mincut();
67              if (k<mint) mint=k;
68              contract(sx,tx);
69          }
70          printf("%d\n",mint);
71      }
72      return 0;
73  }
```

4.29 Strongly Connected Component

```
1  //缩点后注意自环
2  void dfs(const short &now)
3  {
4      dfn[now]=low[now]=cnt++;
5      st.push(now);
6      for(std::list<short>::const_iterator it(edge[now].begin());it!=
          edge[now].end();++it)
7          if(dfn[*it]==-1)
8          {
9              dfs(*it);
10             low[now]=std::min(low[now],low[*it]);
11         }
12         else
13             if(sc[*it]==-1)
14                 low[now]=std::min(low[now],dfn[*it]);
15     if(dfn[now]==low[now])
16     {
```

```
17          while(sc[now]==-1)
18          {
19              sc[st.top()]=p;
20              st.pop();
21          }
22          ++p;
23      }
24  }
```

4.30 ZKW's Minimum-cost flow

```
1   #include<cstdio>
2   #include<algorithm>
3   #include<cstring>
4   #include<vector>
5   #include<deque>
6
7   #define MAXX 111
8   #define MAXN 211
9   #define MAXE (MAXN*MAXN*3)
10  #define inf 0x3f3f3f3f
11
12  char buf[MAXX];
13
14  int edge[MAXN],nxt[MAXE],to[MAXE],cap[MAXE],cst[MAXE],cnt;
15
16  inline void adde(int a,int b,int c,int k)
17  {
18      nxt[cnt]=edge[a];
19      edge[a]=cnt;
20      to[cnt]=b;
21      cap[cnt]=c;
22      cst[cnt]=k;
23      ++cnt;
24  }
25
26  inline void add(int a,int b,int c,int k)
27  {
28      adde(a,b,c,k);
29      adde(b,a,0,-k);
30  }
31
32  int n,mf,cost,pi1;
33  int source,sink;
34  bool done[MAXN];
35
36  int aug(int now,int maxcap)
37  {
38      if(now==sink)
39      {
40          mf+=maxcap;
41          cost+=maxcap*pi1;
```

```
42          return maxcap;
43      }
44      done[now]=true;
45      int l=maxcap;
46      for(int i(edge[now]);i!=-1;i=nxt[i])
47          if(cap[i] && !cst[i] && !done[to[i]])
48          {
49              int d(aug(to[i],std::min(l,cap[i])));
50              cap[i]-=d;
51              cap[i^1]+=d;
52              l-=d;
53              if(!l)
54                  return maxcap;
55          }
56      return maxcap-l;
57  }
58
59  inline bool label()
60  {
61      static int d,i,j;
62      d=inf;
63      for(i=1;i<=n;++i)
64          if(done[i])
65              for(j=edge[i];j!=-1;j=nxt[j])
66                  if(cap[j] && !done[to[j]] && cst[j]<d)
67                      d=cst[j];
68      if(d==inf)
69          return false;
70      for(i=1;i<=n;++i)
71          if(done[i])
72              for(j=edge[i];j!=-1;j=nxt[j])
73              {
74                  cst[j]-=d;
75                  cst[j^1]+=d;
76              }
77      pi1+=d;
78      return true;
79      /* primal-dual approach
80      static int d[MAXN],i,j;
81      static std::deque<int>q;
82      memset(d,0x3f,sizeof d);
83      d[sink]=0;
84      q.push_back(sink);
85      while(!q.empty())
86      {
87          static int dt,now;
88          now=q.front();
89          q.pop_front();
90          for(i=edge[now];i!=-1;i=nxt[i])
91              if(cap[i^1] && (dt=d[now]-cst[i])<d[to[i]])
92                  if((d[to[i]]=dt)<=d[q.empty()?0:q.front()])
```

```
 93                      q.push_front(to[i]);
 94                  else
 95                      q.push_back(to[i]);
 96      }
 97      for(i=1;i<=n;++i)
 98          for(j=edge[i];j!=-1;j=nxt[j])
 99              cst[j]+=d[to[j]]-d[i];
100      pi1+=d[source];
101      return d[source]!=inf;
102      */
103  }
104
105  int m,i,j,k;
106  typedef std::pair<int,int> pii;
107  std::vector<pii>M(MAXN),H(MAXN);
108
109  int main()
110  {
111      while(scanf("%d %d",&n,&m),(n||m))
112      {
113          M.resize(0);
114          H.resize(0);
115          for(i=0;i<n;++i)
116          {
117              scanf("%s",buf);
118              for(j=0;j<m;++j)
119                  if(buf[j]=='m')
120                      M.push_back(pii(i,j));
121                  else
122                      if(buf[j]=='H')
123                          H.push_back(pii(i,j));
124          }
125          n=M.size()+H.size();
126          source=++n;
127          sink=++n;
128          memset(edge,-1,sizeof edge);
129          cnt=0;
130          for(i=0;i<M.size();++i)
131              for(j=0;j<H.size();++j)
132                  add(i+1,j+1+M.size(),1,abs(M[i].first-H[j].first)+
                        abs(M[i].second-H[j].second));
133          for(i=0;i<M.size();++i)
134              add(source,i+1,1,0);
135          for(i=0;i<H.size();++i)
136              add(i+1+M.size(),sink,1,0);
137          mf=cost=pi1=0;
138          do
139              do
140                  memset(done,0,sizeof done);
141              while(aug(source,inf));
142          while(label());
```

```
143         /* primal-dual approach
144         while(label())
145             do
146                 memset(done,0,sizeof done);
147             while(aug(source,inf));
148         */
149         printf("%d\n",cost);
150     }
151     return 0;
152 }
```

5 Math

5.1 cantor

```
 1 const int PermSize = 12;
 2 int fac[PermSize] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880,
       3628800, 39916800};
 3
 4 inline int Cantor(int a[])
 5 {
 6     int i, j, cnt;
 7     int res = 0;
 8     for (i = 0; i < PermSize; ++i)
 9     {
10         cnt = 0;
11         for (j = i + 1; j < PermSize; ++j)
12             if (a[i] > a[j])
13                 ++cnt;
14         res = res + cnt * fac[PermSize - i - 1];
15     }
16     return res;
17 }
18
19 bool h[13];
20
21 inline void UnCantor(int x, int res[])
22 {
23     int i,j,l,t;
24     for (i = 1;i <= 12;i++)
25         h[i] = false;
26     for (i = 1; i <= 12; i++)
27     {
28         t = x / fac[12 - i];
29         x -= t * fac[12 - i];
30         for (j = 1, l = 0; l <= t; j++)
31             if (!h[j])
32                 l++;
33         j--;
34         h[j] = true;
35         res[i - 1] = j;
36     }
```

```
}
```

5.2 discrete logarithms - BSGS

```
//The running time of BSGS and the space complexity is O(√n)
//Pollard's rho algorithm for logarithms' running time is approximately O(√p) where p is n's largest prime factor.
#include<cstdio>
#include<cmath>
#include<cstring>

struct Hash // std::map is bad. clear() 时会付出巨大的代价
{
    static const int mod=100003; // prime is good
    static const int MAXX=47111; // bigger than √c
    int hd[mod],nxt[MAXX],cnt;
    long long v[MAXX],k[MAXX]; // a^k ≡ v (mod c)
    inline void init()
    {
        memset(hd,0,sizeof hd);
        cnt=0;
    }
    inline long long find(long long v)
    {
        static int now;
        for(now=hd[v%mod];now;now=nxt[now])
            if(this->v[now]==v)
                return k[now];
        return -1ll;
    }
    inline void insert(long long k,long long v)
    {
        if(find(v)!=-1ll)
            return;
        nxt[++cnt]=hd[v%mod];
        hd[v%mod]=cnt;
        this->v[cnt]=v;
        this->k[cnt]=k;
    }
}hash;

long long gcd(long long a,long long b)
{
    return b?gcd(b,a%b):a;
}

long long exgcd(long long a,long long b,long long &x,long long &y)
{
    if(b)
    {
        long long re(exgcd(b,a%b,x,y)),tmp(x);
        x=y;
```

```
48          y=tmp-(a/b)*y;
49          return re;
50      }
51      x=1ll;
52      y=0ll;
53      return a;
54  }
55
56  inline long long bsgs(long long a,long long b,long long c) // a^x ≡ b (mod c)
57  {
58      static long long x,y,d,g,m,am,k;
59      static int i,cnt;
60      a%=c;
61      b%=c;
62      x=1ll%c; // if c==1....
63      for(i=0;i<100;++i)
64      {
65          if(x==b)
66              return i;
67          x=(x*a)%c;
68      }
69      d=1ll%c;
70      cnt=0;
71      while((g=gcd(a,c))!=1ll)
72      {
73          if(b%g)
74              return -1ll;
75          ++cnt;
76          c/=g;
77          b/=g;
78          d=a/g*d%c;
79      }
80      hash.init();
81      m=sqrt((double)c); // maybe need a ceil
82      am=1ll%c;
83      hash.insert(0,am);
84      for(i=1;i<=m;++i)
85      {
86          am=am*a%c;
87          hash.insert(i,am);
88      }
89      for(i=0;i<=m;++i)
90      {
91          g=exgcd(d,c,x,y);
92          x=(x*b/g%c+c)%c;
93          k=hash.find(x);
94          if(k!=-1ll)
95              return i*m+k+cnt;
96          d=d*am%c;
97      }
```

```
    return -1ll;
}

long long k,p,n;

int main()
{
    while(scanf("%lld %lld %lld",&k,&p,&n)!=EOF)
    {
        if(n>p || (k=bsgs(k,n,p))==-1ll)
            puts("Orz,I' cant find D!");
        else
            printf("%lld\n",k);
    }
    return 0;
}
```

5.3 extended euclidean algorithm

```
//返回ax+by=gcd(a,b)的一组解
long long ex_gcd(long long a,long long b,long long &x,long long &y)
{
    if (b)
    {
        long long ret = ex_gcd(b,a%b,x,y),tmp = x;
        x = y;
        y = tmp-(a/b)*y;
        return ret;
    }
    else
    {
        x = 1;
        y = 0;
        return a;
    }
}
```

5.4 Fast Fourier Transform

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

#define MAXX 100111
#define MAXN (MAXX<<2)

int T;
int n,i,j,k;

typedef std::complex<long double> com;
std::vector<com>x(MAXN);
```

```
15 int a[MAXX];
16 long long pre[MAXN],cnt[MAXN];
17 long long ans;
18
19 inline void fft(std::vector<com> &y,int sign)
20 {
21     static int i,j,k,h;
22     static com u,t,w,wn;
23     for(i=1,j=y.size()/2;i+1<y.size();++i)
24     {
25         if(i<j)
26             std::swap(y[i],y[j]);
27         k=y.size()/2;
28         while(j>=k)
29         {
30             j-=k;
31             k/=2;
32         }
33         if(j<k)
34             j+=k;
35     }
36     for(h=2;h<=y.size();h<<=1)
37     {
38         wn=com(cos(-sign*2*M_PI/h),sin(-sign*2*M_PI/h));
39         for(j=0;j<y.size();j+=h)
40         {
41             w=com(1,0);
42             for(k=j;k<j+h/2;++k)
43             {
44                 u=y[k];
45                 t=w*y[k+h/2];
46                 y[k]=u+t;
47                 y[k+h/2]=u-t;
48                 w*=wn;
49             }
50         }
51     }
52     if(sign==-1)
53         for(i=0;i<y.size();++i)
54             y[i]=com(y[i].real()/y.size(),y[i].imag());
55 }
56
57 int main()
58 {
59     scanf("%d",&T);
60     while(T--)
61     {
62         memset(cnt,0,sizeof cnt);
63         scanf("%d",&n);
64         for(i=0;i<n;++i)
65         {
```

```
66              scanf("%d",a+i);
67              ++cnt[a[i]];
68          }
69          std::sort(a,a+n);
70          k=a[n-1]+1;
71          for(j=1;j<(k<<1);j<<=1);// size must be such many
72          x.resize(0);
73          for(i=0;i<k;++i)
74              x.push_back(com(cnt[i],0));
75          x.insert(x.end(),j-k,com(0,0));
76
77          fft(x,1);
78          for(i=0;i<x.size();++i)
79              x[i]=x[i]*x[i];
80          fft(x,-1);
81          /*
82          if we need to combine 2 arrays
83          fft(x,1);
84          fft(y,1);
85          for(i=0;i<x.size();++i)
86              x[i]=x[i]*y[i];
87          fft(x,-1);
88          */
89
90          for(i=0;i<x.size();++i)
91              cnt[i]=ceil(x[i].real()); //  maybe we need (x[i].real
                    ()+0.5f) or nearbyint(x[i].real())
92          x.resize(2*a[n-1]); // result here
93      }
94      return 0;
95  }
```

5.5 Gaussian elimination

```
1   #define N
2
3   inline int ge(int a[N][N],int n) // 返回系数矩阵的秩
4   {
5       static int i,j,k,l;
6       for(j=i=0;j<n;++j) //第 i 行, 第 j 列
7       {
8           for(k=i;k<n;++k)
9               if(a[k][j])
10                  break;
11          if(k==n)
12              continue;
13          for(l=0;l<=n;++l)
14              std::swap(a[i][l],a[k][l]);
15          for(l=0;l<=n;++l)
16              if(l!=i && a[l][j])
17                  for(k=0;k<=n;++k)
18                      a[l][k]^=a[i][k];
```

```
19          ++i;
20      }
21      for(j=i;j<n;++j)
22          if(a[j][n])
23              return -1; //无解
24      return i;
25  }
26  /*
27   */
28
29  void dfs(int v)
30  {
31      if(v==n)
32      {
33          static int x[MAXX],ta[MAXX][MAXX];
34          static int tmp;
35          memcpy(x,ans,sizeof(x));
36          memcpy(ta,a,sizeof(ta));
37          for(i=l-1;i>=0;--i)
38          {
39              for(j=i+1;j<n;++j)
40                  ta[i][n]^=(x[j]&&ta[i][j]); //迭代消元求解
41              x[i]=ta[i][n];
42          }
43          for(tmp=i=0;i<n;++i)
44              if(x[i])
45                  ++tmp;
46          cnt=std::min(cnt,tmp);
47          return;
48      }
49      ans[v]=0;
50      dfs(v+1);
51      ans[v]=1;
52      dfs(v+1);
53  }
54
55  inline int ge(int a[N][N],int n)
56  {
57      static int i,j,k,l;
58      for(i=j=0;j<n;++j)
59      {
60          for(k=i;k<n;++k)
61              if(a[k][i])
62                  break;
63          if(k<n)
64          {
65              for(l=0;l<=n;++l)
66                  std::swap(a[i][l],a[k][l]);
67              for(k=0;k<n;++k)
68                  if(k!=i && a[k][i])
69                      for(l=0;l<=n;++l)
```

```
70                          a[k][l]^=a[i][l];
71              ++i;
72          }
73          else //将不定元交换到后面去
74          {
75              l=n-1-j+i;
76              for(k=0;k<n;++k)
77                  std::swap(a[k][l],a[k][i]);
78          }
79      }
80      if(i==n)
81      {
82          for(i=cnt=0;i<n;++i)
83              if(a[i][n])
84                  ++cnt;
85          printf("%d\n",cnt);
86          continue;
87      }
88      for(j=i;j<n;++j)
89          if(a[j][n])
90              break;
91      if(j<n)
92          puts("impossible");
93      else
94      {
95          memset(ans,0,sizeof(ans));
96          cnt=111;
97          dfs(l=i);
98          printf("%d\n",cnt);
99      }
100 }
101
102 /*
103  */
104 inline int ge(int n,int m)
105 {
106     static int i,j,r,c;
107     static double mv;
108     for(r=c=0;r<n && c<m;++r,++c)
109     {
110         for(mv=0,i=r;i<n;++i)
111             if(fabs(mv)<fabs(a[i][c]))
112                 mv=a[j=i][c];
113         if(fabs(mv)<eps) // important
114         {
115             --r;
116             continue;
117         }
118         for(i=0;i<=m;++i)
119             std::swap(a[r][i],a[j][i]);
120         for(j=c+1;j<=m;++j)
```

```
121         {
122             a[r][j]/=mv;
123             for(i=r+1;i<n;++i)
124                 a[i][j]-=a[i][c]*a[r][j];
125         }
126     }
127     for(i=r;i<n;++i)
128         if(fabs(a[i][m])>eps)
129             return -1;
130     if(r<m) // rank
131         return m-r;
132     for(i=m-1;i>=0;--i)
133         for(j=i+1;j<m;++j)
134             a[i][m]-=a[i][j]*a[j][m];  // answer will be a[i][m]
135     return 0;
136 }
```

5.6 Integration

```
1  // simpson 公式用到的函数
2  double F(double x) {
3    return sqrt(1 + 4*a*a*x*x);
4  }
5
6  // 三点 simpson 法。这里要求 F 是一个全局函数
7  double simpson(double a, double b) {
8    double c = a + (b-a)/2;
9    return (F(a)+4*F(c)+F(b))*(b-a)/6;
10 }
11
12 // 自适应 Simpson 公式（递归过程）。已知整个区间 [a,b] 上的三点 simpson 值 A
13 double asr(double a, double b, double eps, double A) {
14   double c = a + (b-a)/2;
15   double L = simpson(a, c), R = simpson(c, b);
16   if(fabs(L+R-A) <= 15*eps)
17       return L+R+(L+R-A)/15.0;
18   return asr(a, c, eps/2, L) + asr(c, b, eps/2, R);
19 }
20
21 // 自适应 Simpson 公式（主过程）
22 double asr(double a, double b, double eps)
23 {
24   return asr(a, b, eps, simpson(a, b));
25 }
26
27 // 用自适应 Simpson 公式计算宽度为 w，高度为 h 的抛物线长
28 double parabola_arc_length(double w, double h)
29 {
30   a = 4.0*h/(w*w); // 修改全局变量 a，从而改变全局函数 F 的行为
31   return asr(0, w/2, 1e-5)*2;
32 }
33
```

```
34 // thx for mzry
35 inline double f(double)
36 {
37     /*
38     define the function
39     */
40 }
41
42 inline double simp(double l,double r)
43 {
44     double h = (r-l)/2.0;
45     return h*(f(l)+4*f((l+r)/2.0)+f(r))/3.0;
46 }
47
48 inline double rsimp(double l,double r) // call here
49 {
50     double mid = (l+r)/2.0;
51     if(fabs((simp(l,r)-simp(l,mid)-simp(mid,r)))/15 < eps)
52         return simp(l,r);
53     else
54         return rsimp(l,mid)+rsimp(mid,r);
55 }
56
57 //Romberg
58
59 /* Romberg 求定积分
60  * 输入：积分区间 [a,b]，被积函数 f(x,y,z)
61  * 输出：积分结果
62  * f(x,y,z) 示例：
63  * double f0( double x, double l, double t )
64  * {
65  * return sqrt(1.0+l*l*t*t*cos(t*x)*cos(t*x));
66  * }
67  */
68 double Integral(double a, double b, double (*f)(double x, double y,
        double z), double eps, double l, double t);
69
70 inline double Romberg (double a, double b, double (*f)(double x,
       double y, double z), double eps, double l, double t)
71 {
72 #define MAX_N 1000
73     int i, j, temp2, min;
74     double h, R[2][MAX_N], temp4;
75     for (i=0; i<MAX_N; i++)
76     {
77         R[0][i] = 0.0;
78         R[1][i] = 0.0;
79     }
80     h = b-a;
81     min = (int)(log(h*10.0)/log(2.0)); //h should be at most 0.1
82     R[0][0] = ((*f)(a, l, t)+(*f)(b, l, t))*h*0.50;
```

```
 83      i = 1;
 84      temp2 = 1;
 85      while (i<MAX_N)
 86      {
 87          i++;
 88          R[1][0] = 0.0;
 89          for (j=1; j<=temp2; j++)
 90              R[1][0] += (*f)(a+h*((double)j-0.50), l, t);
 91          R[1][0] = (R[0][0] + h*R[1][0])*0.50;
 92          temp4 = 4.0;
 93          for (j=1; j<i; j++)
 94          {
 95              R[1][j] = R[1][j-1] + (R[1][j-1]-R[0][j-1])/(temp4-1.0)
                    ;
 96              temp4 *= 4.0;
 97          }
 98          if ((fabs(R[1][i-1]-R[0][i-2])<eps) && (i>min))
 99              return R[1][i-1];
100          h *= 0.50;
101          temp2 *= 2;
102          for (j=0; j<i; j++)
103              R[0][j] = R[1][j];
104      }
105      return R[1][MAX_N-1];
106  }
107
108  inline double Integral(double a, double b, double (*f)(double x,
        double y, double z), double eps, double l, double t)
109  {
110      const double pi(acos(-1.0f));
111      int n;
112      double R, p, res;
113      n = (int)(floor)(b * t * 0.50 / pi);
114      p = 2.0 * pi / t;
115      res = b - (double)n * p;
116      if (n)
117          R = Romberg (a, p, f0, eps/(double)n, l, t);
118      R = R * (double)n + Romberg( 0.0, res, f0, eps, l, t );
119      return R/100.0;
120  }
121
122  //
123  inline double romberg(double a,double b)
124  {
125  #define MAXN 111
126      double t[MAXN][MAXN];
127      int n,k,i,m;
128      double h,g,p;
129      h=(double)(b-a)/2;
130      t[0][0]=h*(func(a)+func(b));
131      k=n=1;
```

```
132     do
133     {
134         g=0;
135         for(i=1;i<=n;i++)
136             g+=func((a+((2*i-1)*h)));
137         t[k][0]=(t[k-1][0]/2)+(h*g);
138         p = 1.0;
139         for(m=1;m<=k;m++)
140         {
141             p=p*4.0f;
142             t[k-m][m]=(p*t[k-m+1][m-1]-t[k-m][m-1])/(p-1);
143         }
144         m-=1;
145         h/=2;
146         n*=2;
147         k+=1;
148 
149     }
150     while (fabs(t[0][m]-t[0][m-1])>eps);
151     return t[0][m];
152 }
```

5.7 inverse element

```
1  inline void getInv2(int x,int mod)
2  {
3      inv[1]=1;
4      for (int i=2; i<=x; i++)
5          inv[i]=(mod-(mod/i)*inv[mod%i]%mod)%mod;
6  }
7  
8  long long inv(long long x)// likes above one
9  {
10     return x <= 1ll ? x : (mod - mod / x) * inv(mod % x) % mod;
11 }
12 
13 inline long long power(long long x,long long y,int mod)
14 {
15     long long ret=1;
16     for (long long a=x%mod; y; y>>=1,a=a*a%mod)
17         if (y&1)
18             ret=ret*a%mod;
19     return ret;
20 }
21 
22 inline int getInv(int x,int mod)//mod 为素数
23 {
24     return power(x,mod-2,mod);
25 }
26 
27 //谨慎来说，用 exgcd 更靠谱
28 void gcd(int n,int k,int &x,int &y)
```

```
{
    if(k)
    {
        gcd(k,n%k,x,y);
        int t=x;
        x=y;
        y=t-(n/k)*y;
        return;
    }
    x=1;
    y=0;
}

inline int inv(int b,int mod)
{
    static int x,y;
    gcd(b,mod,x,y);
    if(x<0)
        x+=mod;
    return x;
}
```

5.8 Linear programming

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

#define MAXN 33
#define MAXM 33
#define eps 1e-8

double a[MAXN][MAXM],b[MAXN],c[MAXM];
double x[MAXM],d[MAXN][MAXM];
int ix[MAXN+MAXM];
double ans;
int n,m;
int i,j,k,r,s;
double D;

inline bool simplex()
{
    r=n;
    s=m++;
    for(i=0;i<n+m;++i)
        ix[i]=i;
    memset(d,0,sizeof d);
    for(i=0;i<n;++i)
    {
        for(j=0;j+1<m;++j)
            d[i][j]=-a[i][j];
```

```
29          d[i][m-1]=1;
30          d[i][m]=b[i];
31          if(d[r][m]>d[i][m])
32              r=i;
33      }
34      for(j=0;j+1<m;++j)
35          d[n][j]=c[j];
36      d[n+1][m-1]=-1;
37      while(true)
38      {
39          if(r<n)
40          {
41              std::swap(ix[s],ix[r+m]);
42              d[r][s]=1./d[r][s];
43              for(j=0;j<=m;++j)
44                  if(j!=s)
45                      d[r][j]*=-d[r][s];
46              for(i=0;i<=n+1;++i)
47                  if(i!=r)
48                  {
49                      for(j=0;j<=m;++j)
50                          if(j!=s)
51                              d[i][j]+=d[r][j]*d[i][s];
52                      d[i][s]*=d[r][s];
53                  }
54          }
55          r=-1;
56          s=-1;
57          for(j=0;j<m;++j)
58              if((s<0 || ix[s]>ix[j]) && (d[n+1][j]>eps || (d[n+1][j
                    ]>-eps && d[n][j]>eps)))
59                  s=j;
60          if(s<0)
61              break;
62          for(i=0;i<n;++i)
63              if(d[i][s]<-eps && (r<0 || (D=(d[r][m]/d[r][s]-d[i][m]/
                    d[i][s]))<-eps || (D<eps && ix[r+m]>ix[i+m])))
64                  r=i;
65          if(r<0)
66              return false;
67      }
68      if(d[n+1][m]<-eps)
69          return false;
70      for(i=m;i<n+m;++i)
71          if(ix[i]+1<m)
72              x[ix[i]]=d[i-m][m]; // answer
73      ans=d[n][m]; // maxium value
74      return true;
75  }
76
77  int main()
```

```
{
    while(scanf("%d␣%d",&m,&n)!=EOF)
    {
        for(i=0;i<m;++i)
            scanf("%lf",c+i);  // max{ sum{c[i]*x[i]} }
        for(i=0;i<n;++i)
        {
            for(j=0;j<m;++j)
                scanf("%lf",a[i]+j); // sum{ a[i]*x[i] } <= b
            scanf("%lf",b+i);
            b[i]*=n;
        }
        simplex();
        printf("Nasa␣can␣spend␣%.0lf␣taka.\n",ceil(ans));
    }
    return 0;
}

/*
Simplex C(n+m)(n)
maximize:
    \sum_{i=1}^{n} (c[i] \times x[i])
subject to
    \forall i \in [1,m]
    \sum_{j=1}^{n} (a[i][j] \times x[j]) \leq rhs[i]
限制:
    传入的矩阵必须是标准形式的.
sample:
3 3
15 17 20
0 1 -1 2
3 3 5 15
3 2 1 8
out:
OPTIMAL
76.00000
x[ 1 ] = 0.333333
x[ 2 ] = 3.000000
x[ 3 ] = 1.000000
*/

#include <cstdio>
#include <cstring>
#include <cmath>

#define eps 1e-8
#define inf 1e15
#define OPTIMAL -1 //最优解
#define UNBOUNDED -2 //无边界的
```

```
127 #define FEASIBLE -3 //可行的
128 #define INFEASIBLE -4 //无解
129 #define PIVOT_OK 1 //还可以松弛
130
131 #define N 45 //变量个数
132 #define M 45 //约束个数
133
134 int basic[N],row[M],col[N];
135 double c0[N];
136
137 inline double dcmp(double x)
138 {
139     if(x>eps)
140         return 1;
141     if(x<-eps)
142         return -1;
143     return 0;
144 }
145
146 inline int Pivot(int n,int m,double *c,double a[M][N],double *rhs,
       int &i,int &j)
147 {
148     double min=inf;
149     int k=-1;
150     for(j=0;j<=n;j++)
151         if(!basic[j] && dcmp(c[j])>0)
152             if(k<0 || dcmp(c[j]-c[k])>0)
153                 k=j;
154     j=k;
155     if(k<0)
156         return OPTIMAL;
157     for(k=-1,i=1;i<=m;i++)
158         if(dcmp(a[i][j])>0 && dcmp(rhs[i]/a[i][j]-min)<0)
159         {
160             min=rhs[i]/a[i][j];
161             k=i;
162         }
163     i=k;
164     if(k<0)
165         return UNBOUNDED;
166     return PIVOT_OK;
167 }
168
169 inline int PhaseII(int n,int m,double *c,double a[M][N],double *rhs
       ,double &ans,int PivotIndex)
170 {
171     static int i,j,k,l;
172     static double tmp;
173     while((k=Pivot(n,m,c,a,rhs,i,j))==PIVOT_OK || PivotIndex)
174     {
175         if(PivotIndex)
```

```
176         {
177             i=PivotIndex;
178             j=PivotIndex=0;
179         }
180         basic[row[i]]=0;
181         col[row[i]]=0;
182         basic[j]=1;
183         col[j]=i;
184         row[i]=j;
185         tmp=a[i][j];
186         for(k=0;k<=n;k++)
187             a[i][k]/=tmp;
188         rhs[i]/=tmp;
189         for(k=1;k<=m;k++)
190             if(k!=i && dcmp(a[k][j]))
191             {
192                 tmp=-a[k][j];
193                 for(l=0;l<=n;l++)
194                     a[k][l]+=tmp*a[i][l];
195                 rhs[k]+=tmp*rhs[i];
196             }
197         tmp=-c[j];
198         for(l=0;l<=n;l++)
199             c[l]+=a[i][l]*tmp;
200         ans-=tmp*rhs[i];
201     }
202     return k;
203 }
204
205 inline int PhaseI(int n,int m,double *c,double a[M][N],double *rhs,
    double &ans)
206 {
207     int i,j,k=-1;
208     double tmp,min=0,ans0=0;
209     for(i=1;i<=m;i++)
210         if(dcmp(rhs[i]-min)<0)
211         {
212             min=rhs[i];
213             k=i;
214         }
215     if(k<0)
216         return FEASIBLE;
217     for(i=1;i<=m;i++)
218         a[i][0]=-1;
219     for(j=1;j<=n;j++)
220         c0[j]=0;
221     c0[0]=-1;
222     PhaseII(n,m,c0,a,rhs,ans0,k);
223     if(dcmp(ans0)<0)
224         return INFEASIBLE;
225     for(i=1;i<=m;i++)
```

```
226         a[i][0]=0;
227     for(j=1;j<=n;j++)
228         if(dcmp(c[j]) && basic[j])
229         {
230             tmp=c[j];
231             ans+=rhs[col[j]]*tmp;
232             for(i=0;i<=n;i++)
233                 c[i]-=tmp*a[col[j]][i];
234         }
235     return FEASIBLE;
236 }
237 inline int simplex(int n,int m,double *c,double a[M][N],double *rhs
        ,double &ans,double *x)
238 {
239     int i,j,k;
240     for(i=1;i<=m;i++)
241     {
242         for(j=n+1;j<=n+m;j++)
243             a[i][j]=0;
244         a[i][n+i]=1;
245         a[i][0]=0;
246         row[i]=n+i;
247         col[n+i]=i;
248     }
249     k=PhaseI(n+m,m,c,a,rhs,ans);
250     if(k==INFEASIBLE)
251         return k; //无解
252     k=PhaseII(n+m,m,c,a,rhs,ans,0);
253     for(j=0;j<=n+m;j++)
254         x[j] = 0;
255     for(i=1;i<=m;i++)
256         x[row[i]] = rhs[i];
257     return k;
258 }
259
260 double c[M],ans,a[M][N],rhs[M],x[N];
261
262 int main()
263 {
264     int i,j,n,m;
265     while(scanf("%d%d",&n,&m)!=EOF)
266     {
267         for(int i=0;i<=n+m;i++)
268         {
269             for(int j=0;j<=n+m;j++)
270                 a[i][j]=0;
271             basic[i]=0;
272             row[i]=0;
273             col[i]=0;
274             c[i]=0;
275             rhs[i]=0;
```

```
276         }
277         ans=0;
278 
279         for(j=1;j<=n;++j)
280             scanf("%lf",c+j);
281         for(i=1;i<=m;++i)
282         {
283             for(j=1;j<=n;++j)
284                 scanf("%lf",a[i]+j);
285             scanf("%lf",rhs+i);
286         }
287 
288         switch(simplex(n,m,c,a,rhs,ans,x))
289         {
290             case OPTIMAL:
291                 printf("Nasa␣can␣spend␣%.0f␣taka.\n",ceil(m*ans));
292                 //for(j=1;j<=n;j++)
293                 //    printf("x[ %2d ] = %10lf\n",j,x[j]);
294                 break;
295             case UNBOUNDED:
296                 puts("UNBOUNDED");
297                 break;
298             case INFEASIBLE:
299                 puts("INFEASIBLE");
300                 break;
301         }
302     }
303     return 0;
304 }
```

5.9 Lucas' theorem(2)

```
 1 #include<cstdio>
 2 #include<cstring>
 3 #include<iostream>
 4 
 5 int mod;
 6 long long num[100000];
 7 int ni[100],mi[100];
 8 int len;
 9 
10 void init(int p)
11 {
12     mod=p;
13     num[0]=1;
14     for (int i=1; i<p; i++)
15         num[i]=i*num[i-1]%p;
16 }
17 
18 void get(int n,int ni[],int p)
19 {
20     for (int i = 0; i < 100; i++)
```

```
21          ni[i] = 0;
22      int tlen = 0;
23      while (n != 0)
24      {
25          ni[tlen++] = n%p;
26          n /= p;
27      }
28      len = tlen;
29  }
30
31  long long power(long long x,long long y)
32  {
33      long long ret=1;
34      for (long long a=x%mod; y; y>>=1,a=a*a%mod)
35          if (y&1)
36              ret=ret*a%mod;
37      return ret;
38  }
39
40  long long getInv(long long x)//mod 为素数
41  {
42      return power(x,mod-2);
43  }
44
45  long long calc(int n,int m,int p)//C(n,m)%p
46  {
47      init(p);
48      long long ans=1;
49      for (; n && m && ans; n/=p,m/=p)
50      {
51          if (n%p>=m%p)
52              ans = ans*num[n%p]%p *getInv(num[m%p]%p)%p *getInv(num[
                    n%p-m%p])%p;
53          else
54              ans=0;
55      }
56      return ans;
57  }
58
59  int main()
60  {
61      int t;
62      scanf("%d",&t);
63      while (t--)
64      {
65          int n,m,p;
66          scanf("%d%d%d",&n,&m,&p);
67          printf("%lld\n",calc(n+m,m,p));
68      }
69      return 0;
70  }
```

5.10 Lucas' theorem

```cpp
#include <cstdio>
/*
   Lucas 快速求解C(n,m)%p
   */
void gcd(int n,int k,int &x,int &y)
{
    if(k)
    {
        gcd(k,n%k,x,y);
        int t=x;
        x=y;
        y=t-(n/k)*y;
        return;
    }
    x=1;
    y=0;
}

int CmodP(int n,int k,int p)
{
    if(k>n)
        return 0;
    int a,b,flag=0,x,y;
    a=b=1;
    for(int i=1;i<=k;i++)
    {
        x=n-i+1;
        y=i;
        while(x%p==0)
        {
            x/=p;
            ++flag;
        }
        while(y%p==0)
        {
            y/=p;
            --flag;
        }
        x%=p;
        y%=p;

        a*=x;
        b*=y;

        b%=p;
        a%=p;
    }
    if(flag)
        return 0;
```

```
50      gcd(b,p,x,y);
51      if(x<0)
52          x+=p;
53      a*=x;
54      a%=p;
55      return a;
56  }
57
58  //用Lucas 定理求解 C(n,m) % p ,p 是素数
59  long long Lucas(long long n, long long m, long long p)
60  {
61      long long ans=1;
62      while(m && n && ans)
63      {
64          ans*=(CmodP(n%p,m%p,p));
65          ans=ans%p;
66          n=n/p;
67          m=m/p;
68      }
69      return ans;
70  }
71  int main()
72  {
73      long long n,k,p,ans;
74      int cas=0;
75      while(scanf("%I64d%I64d%I64d",&n,&k,&p)!=EOF)
76      {
77          if(k>n-k)
78              k=n-k;
79          ans=Lucas(n+1,k,p)+n-k;
80          printf("Case␣#%d:␣%I64d\n",++cas,ans%p);
81      }
82      return 0;
83  }
```

5.11 matrix

```
1   template<int n>class Matrix
2   {
3       long long a[n][n];
4       inline Matrix<n> operator*(const Matrix<n> &b)const //比照着公式
            来会快一点常数……nmlgb 的 zoj3289……
5       {
6           //别忘了矩阵乘法虽然满足结合律但是不满足交换律……
7           static Matrix<n> re;
8           static int i,j,k;
9           for(i=0;i<n;++i)
10              for(j=0;j<n;++j)
11                  re.a[i][j]=0;
12          for(k=0;k<n;++k)
13              for(i=0;i<n;++i)
14                  if(a[i][k])
```

```
15                      for(j=0;j<n;++j)
16                          if(b.a[k][j])
17                              re.a[i][j]=(re.a[i][j]+a[i][k]*b.a[k][j
                                    ])%mod;
18          return re;
19      }
20      inline Matrix<n> operator^(int y)const
21      {
22          static Matrix<n> re,x;
23          static int i,j;
24          for(i=0;i<n;++i)
25          {
26              for(j=0;j<n;++j)
27              {
28                  re.a[i][j]=0;
29                  x.a[i][j]=a[i][j];
30              }
31              re.a[i][i]=1;
32          }
33          for(;y;y>>=1,x=x*x)
34              if(y&1)
35                  re=re*x;
36          return re;
37      }
38      long long det()
39      {
40          static int i,j,k;
41          static long long ret,t;
42          ret=1ll;
43          for(i=0;i<n;++i)
44              for(j=0;j<n;++j)
45                  a[i][j]%=mod;
46          for(i=0;i<n;++i)
47          {
48              for(j=i+1;j<n;++j)
49                  while(a[j][i])
50                  {
51                      t=a[i][i]/a[j][i];
52                      for(k=i;k<n;++k)
53                          a[i][k]=(a[i][k]-a[j][k]*t)%mod;
54                      for(k=i;k<n;++k)
55                          std::swap(a[i][k],a[j][k]);
56                      ret=-ret;
57                  }
58              if(!a[i][i])
59                  return 0ll;
60              ret=ret*a[i][i]%mod;
61          }
62          return (ret+mod)%mod;
63      }
64  };
```

```
65 |
66 |/*
67 |Fibonacci Matrix
68 |1  1
   |1  0
69 |
70 |org[0][j], trans[i][j]
71 | means
72 |transform(org,1 times) -> org[0][j]=∑_{i=0}^{n} org[0][i] × trans[i][j]
73 |      */
```

5.12 Pell's equation

```
 1 |/*
 2 |find the (x,y)pair that x^2 − n × y^2 = 1
 3 |these is not solution if and only if n is a square number.
 4 |
 5 |solution:
 6 |simply brute−force search the integer y, get (x1,y1). ( toooo slow
   |   in some situation )
 7 |or we can enumerate the continued fraction of √n, as x/y, it will be
   |   much more faster
 8 |
 9 |other solution pairs' matrix:
10 |x1  n × y1
   |y1   x1
11 |k-th solution is {matrix}^k
12 |*/
13 |
14 |import java.util.*;
15 |import java.math.*;
16 |
17 |public class Main
18 |{
19 |    static BigInteger p,q,p1,p2,p3,q1,q2,q3,a1,a2,a0,h1,h2,g1,g2,n0
   |       ;
20 |    static int n,t;
21 |    static void solve()
22 |    {
23 |        p2=BigInteger.ONE;
24 |        p1=BigInteger.ZERO;
25 |        q2=BigInteger.ZERO;
26 |        q1=BigInteger.ONE;
27 |        a0=a1=BigInteger.valueOf((long)Math.sqrt(n));
28 |        g1=BigInteger.ZERO;
29 |        h1=BigInteger.ONE;
30 |        n0=BigInteger.valueOf(n);
31 |        while(true)
32 |        {
33 |            g2=a1.multiply(h1).subtract(g1);
```

```
34              h2=(n0.subtract(g2.multiply(g2))).divide(h1);
35              a2=(g2.add(a0)).divide(h2);
36              p=p2.multiply(a1).add(p1);
37              q=q2.multiply(a1).add(q1);
38              if(p.multiply(p).subtract(n0.multiply(q.multiply(q))).
                   equals(BigInteger.ONE))
39                  return ;
40              a1=a2;
41              g1=g2;
42              h1=h2;
43              p1=p2;
44              p2=p;
45              q1=q2;
46              q2=q;
47          }
48      }
49      public static void main(String[] args)
50      {
51          Scanner in=new Scanner(System.in);
52          t=in.nextInt();
53          for(int i=0;i<t;++i)
54          {
55              n=in.nextInt();
56              solve();
57              System.out.println(p+"␣"+q);
58          }
59      }
60  }
```

5.13 Pollard's rho algorithm

```
1   #include<cstdio>
2   #include<cstdlib>
3   #include<list>
4
5   short T;
6   unsigned long long a;
7   std::list<unsigned long long>fac;
8
9   inline unsigned long long multi_mod(const unsigned long long &a,
       unsigned long long b,const unsigned long long &n)
10  {
11      unsigned long long exp(a%n),tmp(0);
12      while(b)
13      {
14          if(b&1)
15          {
16              tmp+=exp;
17              if(tmp>n)
18                  tmp-=n;
19          }
20          exp<<=1;
```

```
21          if(exp>n)
22              exp-=n;
23          b>>=1;
24      }
25      return tmp;
26  }
27
28  inline unsigned long long exp_mod(unsigned long long a,unsigned
       long long b,const unsigned long long &c)
29  {
30      unsigned long long tmp(1);
31      while(b)
32      {
33          if(b&1)
34              tmp=multi_mod(tmp,a,c);
35          a=multi_mod(a,a,c);
36          b>>=1;
37      }
38      return tmp;
39  }
40
41  inline bool miller_rabbin(const unsigned long long &n,short T)
42  {
43      if(n==2)
44          return true;
45      if(n<2 || !(n&1))
46          return false;
47      unsigned long long a,u(n-1),x,y;
48      short t(0),i;
49      while(!(u&1))
50      {
51          ++t;
52          u>>=1;
53      }
54      while(T--)
55      {
56          a=rand()%(n-1)+1;
57          x=exp_mod(a,u,n);
58          for(i=0;i<t;++i)
59          {
60              y=multi_mod(x,x,n);
61              if(y==1 && x!=1 && x!=n-1)
62                  return false;
63              x=y;
64          }
65          if(y!=1)
66              return false;
67      }
68      return true;
69  }
70
```

```
71 unsigned long long gcd(const unsigned long long &a,const unsigned
      long long &b)
72 {
73     return b?gcd(b,a%b):a;
74 }
75
76 inline unsigned long long pollar_rho(const unsigned long long n,
      const unsigned long long &c)
77 {
78     unsigned long long x(rand()%(n-1)+1),y,d,i(1),k(2);
79     y=x;
80     while(true)
81     {
82         ++i;
83         x=(multi_mod(x,x,n)+c)%n;
84         d=gcd((x-y+n)%n,n);
85         if(d>1 && d<n)
86             return d;
87         if(x==y)
88             return n;
89         if(i==k)
90         {
91             k<<=1;
92             y=x;
93         }
94     }
95 }
96
97 void find(const unsigned long long &n,short c)
98 {
99     if(n==1)
100         return;
101     if(miller_rabbin(n,6))
102     {
103         fac.push_back(n);
104         return;
105     }
106     unsigned long long p(n);
107     short k(c);
108     while(p>=n)
109         p=pollar_rho(p,c--);
110     find(p,k);
111     find(n/p,k);
112 }
113
114 int main()
115 {
116     scanf("%hd",&T);
117     while(T--)
118     {
119         scanf("%llu",&a);
```

```
120         fac.clear();
121         find(a,120);
122         if(fac.size()==1)
123             puts("Prime");
124         else
125         {
126             fac.sort();
127             printf("%llu\n",fac.front());
128         }
129     }
130     return 0;
131 }
```

5.14 Combinatorics

5.14.1 Subfactorial

$!n =$number of permutations of n elements with no fixed points

from !0:
1, 0, 1, 2, 9, 44, 265, 1854, 14833, 133496, 1334961, 14684570
$!n = (n-1)(!(n-1)+!(n-2))$
PS:$n! = (n-1)((n-1)! + (n-2)!)$
$!n = n \times n! + (-1)^n$

Rencontres numbers:
$D_{n,k}$ is the number of permutations of { 1, ..., n } that have exactly k fixed points.
$D_{n,0} = !n$
$D_{n,k} = \binom{n}{k} \times !(n-k)$

5.14.2 Ménage numbers

Ménage numbers:
number of permutations s of [0, ..., n-1] such that.
$\forall i, s(i) \neq i$ and $s(i) \not\equiv i+1 \pmod{n}$.

from A(0):
1, 0, 0, 1, 2, 13, 80, 579, 4738, 43387, 439792, 4890741

$A_n = \sum_{k=0}^{n} (-1)^k \frac{2n}{2n-k} \binom{2n-k}{k} (n-k)!$
$A_n = nA_{n-1} + \frac{n}{n-2} A_{n-2} + \frac{4(-1)^{n-1}}{n-2}$
$A_n = nA_{n-1} + 2A_{n-2} - (n-4)A_{n-3} - A_{n-4}$

5.14.3 Multiset

Permutation:
MultiSet S={1 m,4 s,4 i,2 p}
$P(S) = \frac{(1+4+4+2)!}{1!4!4!2!}$

Combination:
MultiSet S=$\{\infty a1, \infty a2, ...\infty ak\}$
$\binom{S}{r} = \frac{(r+k-1)!}{r!(k-1)!} = \binom{r+k-1}{r}$

if(r>min{count(element[i])})
you have to resolve this problem with inclusion-exclusion principle.

MS T={3 a,4 b,5 c}
MS $T_* = \{\infty a, \infty b, \infty c\}$
$A1 = \{\binom{T_*}{10} | count(a) > 3\} // \binom{8}{6}$
$A2 = \{\binom{T_*}{10} | count(b) > 4\} // \binom{7}{5}$
$A3 = \{\binom{T_*}{10} | count(c) > 5\} // \binom{6}{4}$

$\binom{T}{10} = \binom{T_*}{10} - (|A_1| + |A_2| + |A_3|) + (|A_1 \cap A_2| + |A_1 \cap A_3| + |A_2 \cap A_3|) - |A_1 \cap A_2 \cap A_3|$
ans=C(10,12)-(C(6,8)+C(5,7)+C(4,6))+(C(1,3)+C(0,2)+0)-0=6

5.14.4 Distributing Balls into Boxes

Distributing m Balls into n Boxes.

balls	boxes	empty	counts
diff	diff	empty	n^m
diff	diff	full	$n! \times S(m,n) = \sum_{i=0}^{n} (-1)^n \binom{n}{i} (n-i)^m$(inclusion-exclusion principle)
diff	same	empty	$\sum_{k=1}^{min\{n,m\}} s(m,k) = \frac{1}{n!} \sum_{k=1}^{min\{n,m\}} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^m$
diff	same	full	S(m,n) (Stirling numbers of the second kind)
same	diff	empty	$\binom{n+m-1}{n-1}$
same	diff	full	$\binom{m-1}{n-1}$

same	same	empty	dp[0][0..n]=dp[1..m][1]=1;
			if(m≥n)
			dp[m][n]=dp[m][n-1]+dp[m-n][n];
			else
			dp[m][n]=dp[m][n-1];

same	same	full	g[m][n]=dp[m-n][n];

5.14.5 Combinatorial Game Theory

Wythoff's game:

- There are two piles of counters.
- Players take turns removing counters (at least 1 counter) from one or both piles; in the latter case, the numbers of counters removed from each pile must be equal.
- The player who removes the last counter wins.

consider the counters of status as pair (a,b) ($a \leq b$)
{first player loses} $\iff a = \lfloor (b-a) \times \phi \rfloor, \phi = \frac{\sqrt{5}+1}{2}$

Fibonacci Nim:

- There is one pile of n counters.
- The first player may remove any positive number of counters, but not the whole pile.

- Thereafter, each player may remove at most twice the number of counters his opponent took on the previous move.
- The player who removes the last counter wins.

{first player wins} $\Longleftrightarrow$ $n \notin$ {Fibonacci number}

poj 1740:

- There are n piles of stones.
- At each step of the game,the player choose a pile,remove at least one stones,then freely move stones from this pile to any other pile that still has stones.
- The player who removes the last counter wins.

{first player lose} $\Longleftrightarrow$ n is even && $(a_1, a_2, ..., a_k)(a_1 \leq a_2 \leq ... \leq a_{2k})$ satisfy $a_{2i-1} = a_{2i}\{\forall i \in [1, k]\}$

Staircase Nim:

- A staircase of n steps contains coins on some of the steps.
- A move of staircase nim consists of moving any positive number of coins from any step j , to the next lower step, j − 1.
- Coins reaching the ground (step 0) are removed from play.
- The player who removes the last counter wins.

Even steps are unusefull.
$SG = x_1 \oplus x_3 \oplus x_5 ...$

Anti-SG:

- Everything is likes SG.
- The player who removes the last counter loses.

{first player wins} $\Longleftrightarrow$
SGsum=0,&& {all piles is 1}
SGsum$\neq$0,&& {some piles ars larger than 1}

Every-SG:

- Everything is likes SG.
- For each turns, player have to move all of sub-games if the sub-game was not ended yet.

{first player wins} $\Longleftrightarrow$ max(steps of all sub-games) is odd.

Coin Game:

- Given a horizontal line of N coins with some coins showing heads and some tails.
- Each turn, a player have to follow some rules, flip some coins. But the most right coin he fliped has to be fliped from head to tail.

- The player who can not flip coin loses.

game{THHTTH} = game{TH}⊕game{TTH}⊕game{TTTTTH}

Tree Game:

- There is a rooted tree.
- Each turn, a player has to remove a edge from the tree. The parts can not connect with root with also are removed.
- The player who removes the last edge wins.

$\forall node(x)$,
$SG(x) = (SG(i_1)+1) \oplus (SG(i_2)+1) \oplus ...$(∀ i are childnodes of x)

Undirectional Graph Game:

- There is a rooted undirectional graph.
- Other rules are likes Tree Game.

Odd Circle's SG value is 1.
Even Circel's SG value is 0.
turn the graph to a tree.

5.14.6 Catalan number

from C_0
1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420
$C_0 = 1$
$C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i}$
$C_{n+1} = \frac{2(2n+1)}{n+1} C_n$

$C_n = \binom{2n}{n} - \binom{2n}{n+1} = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$
$C_n \sim \frac{4^n}{n^{3/2}\sqrt{\pi}}$
Applications:

1. C_n counts the number of expressions containing n pairs of parentheses which are correctly matched.
2. C_n is the number of full binary trees with n + 1 leaves.
3. C_n is the number of non-isomorphic ordered trees with n+1 vertices. (An ordered tree is a rooted tree in which the children of each vertex are given a fixed left-to-right order.)
4. C_n is the number of monotonic paths along the edges of a grid with n × n square cells, which do not pass above the diagonal.($x \leq y$ for C_n, $x < y$ for $C_n - 1$)
   (a) for the rectangle (p,q),($x < y$), $ans = \binom{p+q-1}{p} - \binom{p+q-1}{p-1} = \frac{q-p}{q+p}\binom{p+q}{q}$
   (b) for the rectangle (p,q),($x \leq y$),$ans = \binom{p+q}{p} - \binom{p+q}{p-1} = \frac{q-p+1}{q+1}\binom{p+q}{q}$

5. C_n is the number of different ways a convex polygon with n + 2 sides can be cut into triangles by connecting vertices with straight lines.
6. C_n is the number of permutations of {1, ..., n} that avoid the pattern 123.
7. C_n is the number of ways to tile a stairstep shape of height n with n rectangles.

5.14.7 Stirling number

First kind:
Stirling numbers of the first kind is signed.
The unsigned Stirling numbers of the first kind are denoted by s(n,k).
s(4,2)=11
s(n,k) count the number of permutations of n elements with k disjoint cycles.
s(n,0)=s(1,1)=1
s(n+1,k)=s(n,k-1)+n s(n,k)

Second kind:
S(n,k) count the number of ways to partition a set of n labelled objects into k nonempty unlabelled subsets.
S(4,2)=7
S(n,n)=S(n,1)=1
S(n,k)=S(n-1,k-1)+k S(n-1,k)
$S(n, n-1) = \binom{n}{2} = \frac{n(n-1)}{2}$
$S(n, 2) = 2^{n-1} - 1$

5.14.8 Delannoy number

Delannoy number D describes the number of paths from (0, 0) to (m, n), using only single steps north, northeast, or east.
D(0,0)=1
D(m,n)=D(m-1,n)+D(m-1,n-1)+D(m,n-1)

central Delannoy numbers D(n) = D(n,n)
D(n) from 0:
1, 3, 13, 63, 321, 1683, 8989, 48639, 265729
$nD(n) = 3(2n-1)D(n-1) - (n-1)D(n-2)$

5.14.9 Schröder number

Large:
Describes the number of paths from (0, 0) to (m, n), using only single steps north, northeast, or east, for all (x,y), ($x \leq y$).
for(n==m),from 0:
1, 2, 6, 22, 90, 394, 1806, 8558, 41586, 206098
$S(n) = S(n-1) + \sum_{k=0}^{n-1} S(k)S(n-1-k)$

Little: (aka. super-Catalan numbers, Hipparchus numbers)

1. the number of different trees with n leaves and with all internal vertices having two or more children.
2. the number of ways of inserting brackets into a sequence.
3. the number of ways of dissecting a convex polygon into smaller polygons by inserting diagonals.

from 0:
1, 1, 3, 11, 45, 197, 903, 4279, 20793, 103049
s(n)=S(n)/2
s(0)=s(1)=1
ns(n)=(6n-9)s(n-1)-(n-3)s(n-2)
$a(n+1) = -a(n) + 2\sum_{k=1}^{n} a(k) \times a(n+1-k)$
$a(n+1) = \sum_{k=0}^{(n-1)/2} 2^k \times 3^{n-1-2k} \binom{n-1}{2k}$

5.14.10 Bell number

Number of partitions of a set of n labeled elements.
from 0:
1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975
$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$
$B_{p+n} \equiv B_n + B_{n+1} \pmod{p}$ (p for prime)
$B_{p^m+n} \equiv mB_n + B_{n+1} \pmod{p}$ (p for prime)
$B_n = \sum_{k=1}^{n} S(n,k)$(S for Stirling second kind)

5.14.11 Eulerian number

First kind:
the number of permutations of the numbers 1 to n in which exactly m elements are greater than the previous element
A(n,0)=1
A(n,m)=(n-m)A(n-1,m-1)+(m+1)A(n-1,m)
A(n,m)=(n-m+1)A(n-1,m-1)+mA(n-1,m)
A(n,m)=A(n,n-1-m)

Second kind:
count the permutations of the multiset {1,1,2,2,...,n,n} with k ascents with the restriction that for all m
T(n,0)=1
T(n,m)=(2n-m-1)T(n-1,m-1)+(m+1)T(n-1,m)

5.14.12 Motzkin number

1. the number of different ways of drawing non-intersecting chords on a circle between n points

2. Number of sequences of length n-1 consisting of positive integers such that the opening and ending elements are 1 or 2 and the absolute difference between any 2 consecutive elements is 0 or 1

3. paths from (0,0) to (n,0) in an n X n grid using only steps U = (1,1), F = (1,0) and D = (1,-1)

from 0:
1, 1, 2, 4, 9, 21, 51, 127, 323, 835, 2188, 5798, 15511, 41835, 113634, 310572, 853467
$M_{n+1} = M_n + \sum_{i=0}^{n-1} M_i M_{n-1-i} = \frac{2n+3}{n+3} M_n + \frac{3n}{n+3} M_{n-1}$
$M_n = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{2k} C_k$(C for catalan)

5.14.13 Narayana number

1. the number of expressions containing n pairs of brackets which are correctly matched and which contain k pairs of ().

2. the number of paths from (0, 0) to (2n, 0), with steps only northeast and southeast, not straying below the x-axis, with k peaks.

N(n,0)=0
$N(n,k) = \frac{1}{n}\binom{n}{k}\binom{n}{k-1}$
$N(n,k) = \frac{1}{k}\binom{n-1}{k-1}\binom{n}{k-1}$
$\sum_{k=1}^{n} N(n,k) = C_n$(C for catalan)

5.15 Number theory

5.15.1 Divisor Fuction

$n = p_1^{a_1} \times p_2^{a_2} \times ... \times p_s^{a_s}$
sum of positive divisors function
$\sigma(n) = \prod_{j=1}^{s} \frac{p_j^{a_j+1}-1}{p_j-1}$
number of postive diversors function
$\tau(n) = \prod_{j=1}^{s} (a_j + 1)$

5.15.2 Reduced Residue System

Euler's totient function:

对正整数 n，欧拉函数 φ 是小于或等于 n 的数中与 n 互质的数的数目，也就是对 n 的简化剩余系的大小。
$\varphi(2)=1$（唯一和 1 互质的数就是 1 本身）。
若 m,n 互质，$\varphi(m \times n) = \varphi(m) \times \varphi(n)$。
对于 n 来说，所有这样的数的和为 $\frac{n \times \varphi(n)}{2}$ 。
$gcd(k,n) = d, k \in [1,n]$，这样的 k 有 $\varphi(\frac{n}{d})$

```
inline int phi(int n)
{
    static int i;
    static int re;
    re=n;
    for(i=0;prm[i]*prm[i]<=n;++i)
        if(n%prm[i]==0)
        {
            re-=re/prm[i];
            do
                n/=prm[i];
            while(n%prm[i]==0);
        }
    if(n!=1)
        re-=re/n;
    return re;
}

inline void Euler()
{
    static int i,j;
    phi[1]=1;
    for(i=2;i<MAXX;++i)
        if(!phi[i])
            for(j=i;j<MAXX;j+=i)
            {
                if(!phi[j])
                    phi[j]=j;
                phi[j]=phi[j]/i*(i-1);
            }
}
```

Multiplicative order:

the multiplicative order of a modulo n is the smallest positive integer k with
$a^k \equiv 1 \pmod{n}$

对 m 的简化剩余系中的所有 x,ord(x) 都一定是 φ(m) 的一个约数 (aka. Euler's totient theorem)

求:
method 1、根据定义，对 φ(m) 分解素因子之后暴力寻找最小的一个 $d\{d|\varphi(m)\}$，满足 $x^d \equiv 1 \pmod{m}$;
method 2、

```
inline long long ord(long long x,long long m)
{
    static long long ans;
    static int i,j;
    ans=phi(m);
    for(i=0;i<fac.size();++i)
```

```
        for(j=0;j<fac[i].second && pow(x,ans/fac[i].first,m)==1ll
           ;++j)
            ans/=fac[i].first;
    return ans;
}
```

Primitive root:

若 ord(x)==φ(m)，则 x 为 m 的一个原根
因此只需检查所有 x^d $\{d|\varphi(m)\}$ 找到使 $x^d \equiv 1 \pmod m$ 的所有 d，当且仅当这样的 d 只有一个，并且为 φ(m) 的时候，x 是 m 的一个原根

当且仅当 m= 1,2,4,p^n,$2 \times p^n$ {p 为奇质数,n 为正整数} 时， m 存在原根 // 应该是指存在对于完全剩余系的原根……？

当 m 存在原根时，原根数目为 $\varphi(\varphi$(m))

求：
枚举每一个简化剩余系中的数 i，若对于 i 的每一个质因子 p[j],$i^{\frac{\varphi(m)}{p[j]}} \not\equiv 1 \pmod m$，那么 i 为 m 的一个原根。也就是说，ord(i)==φ(m)。
最小原根通常极小。

Carmichael function:

λ(n) is defined as the smallest positive integer m such that
$a^m \equiv 1 \pmod n \{\forall a! = 1 \&\& gcd(a,n) == 1\}$
也就是简化剩余系 (完全剩余系中存在乘法群中无法得到 1 的数) 中所有 x 的 lcm{ord(x)}

if n=$p[0]^{a[0]} \times p[1]^{a[1]} \times ... \times p[m-1]^{a[m-1]}$
then λ(n)=lcm($\lambda(p[0]^{a[0]})$,$\lambda(p[1]^{a[1]})$,...,$\lambda(p[m-1]^{a[m-1]})$);

if n=$2^c \times p[0]^{a[0]} \times p[1]^{a[1]} \times ... \times p[m-1]^{a[m-1]}$
then λ(n)=lcm(2^c,$\varphi(p[0]^{a[0]})$,$\varphi(p[1]^{a[1]})$,...,$\varphi(p[m-1]^{a[m-1]})$);
c=0 if a<2; c=1 if a==2; c=a-2 if a>3;

Carmichael's theorem:
if gcd(a,n)==1
then $\lambda(n) \equiv 1 \pmod n$

5.15.3 Prime

Prime number theorem:
Let π(x) be the prime-counting function that gives the number of primes less than or equal to x, for any real number x.
$\lim\limits_{x\to\infty} \frac{\pi(x)}{x/\ln(x)} = 1$
known as the asymptotic law of distribution of prime numbers.
$\pi(x) \sim \frac{x}{\ln x}$.

```
#include<vector>

```

```
std::vector<int>prm;
bool flag[MAXX];

int main()
{
    prm.reserve(MAXX); // pi(x)=x/ln(x);
    for(i=2;i<MAXX;++i)
    {
        if(!flag[i])
            prm.push_back(i);
        for(j=0;j<prm.size() && i*prm[j]<MAXX;++j)
        {
            flag[i*prm[j]]=true;
            if(i%pmr[j]==0)
                break;
        }
    }
    return 0;
}
```

5.15.4 Euler–Mascheroni constant

$$\gamma = \lim_{n\to\infty}\left(\sum_{k=1}^{n}\frac{1}{k} - \ln(n)\right) = \int_1^{\infty}\left(\frac{1}{\lfloor x\rfloor} - \frac{1}{x}\right)dx$$

0.57721566490153286060651209008240243104215933593992...

5.15.5 Fibonacci

gcd(fib[i],fib[j])=fib[gcd(i,j)]

5.16 System of linear congruences

```
// minimal val that for all (m,a) , val%m == a
#include<cstdio>

#define MAXX 11

int T,t;
int m[MAXX],a[MAXX];
int n,i,j,k;
int x,y,c,d;
int lcm;

int exgcd(int a,int b,int &x,int &y)
{
    if(b)
    {
        int re(exgcd(b,a%b,x,y)),tmp(x);
        x=y;
        y=tmp-(a/b)*y;
```

```
        return re;
    }
    x=1;
    y=0;
    return a;
}

int main()
{
    scanf("%d",&T);
    for(t=1;t<=T;++t)
    {
        scanf("%d",&n);
        lcm=1;
        for(i=0;i<n;++i)
        {
            scanf("%d",m+i);
            lcm*=m[i]/exgcd(lcm,m[i],x,y);
        }
        for(i=0;i<n;++i)
            scanf("%d",a+i);
        for(i=1;i<n;++i)
        {
            c=a[i]-a[0];
            d=exgcd(m[0],m[i],x,y);
            if(c%d)
                break;
            y=m[i]/d;
            c/=d;
            x=(x*c%y+y)%y;
            a[0]+=m[0]*x;
            m[0]*=y;
        }
        //标程用的步长可能是最终的 m[0] 而不是 lcm。枚举一下标程
        printf("Case␣%d:␣%d\n",t,i<n?-1:(a[0]?a[0]:lcm));
    }
    return 0;
}
```

6 String

6.1 Aho-Corasick Algorithm

```
//trie graph
#include<cstring>
#include<queue>

#define MAX 1000111
#define N 26

int nxt[MAX][N],fal[MAX],cnt;
bool ed[MAX];
```

```
10 char buf[MAX];
11
12 inline void init(int a)
13 {
14     memset(nxt[a],0,sizeof(nxt[0]));
15     fal[a]=0;
16     ed[a]=false;
17 }
18
19 inline void insert()
20 {
21     static int i,p;
22     for(i=p=0;buf[i];++i)
23     {
24         if(!nxt[p][map[buf[i]]])
25             init(nxt[p][map[buf[i]]]=++cnt);
26         p=nxt[p][map[buf[i]]];
27     }
28     ed[p]=true;
29 }
30
31 inline void make()
32 {
33     static std::queue<int>q;
34     int i,now,p;
35     q.push(0);
36     while(!q.empty())
37     {
38         now=q.front();
39         q.pop();
40         for(i=0;i<N;++i)
41             if(nxt[now][i])
42             {
43                 q.push(p=nxt[now][i]);
44                 if(now)
45                     fal[p]=nxt[fal[now]][i];
46                 ed[p]|=ed[fal[p]];
47             }
48             else
49                 nxt[now][i]=nxt[fal[now]][i]; // 使用本身的 trie 存串
                    的时候注意 nxt 已被重载
50     }
51 }
52
53 // normal version
54
55 #define N 128
56
57 char buf[MAXX];
58 int cnt[1111];
59
```

```
60 struct node
61 {
62     node *fal,*nxt[N];
63     int idx;
64     node() { memset(this,0,sizeof node); }
65 }*rt;
66 std::queue<node*>Q;
67 
68 void free(node *p)
69 {
70     for(int i(0);i<N;++i)
71         if(p->nxt[i])
72             free(p->nxt[i]);
73     delete p;
74 }
75 
76 inline void add(char *s,int idx)
77 {
78     static node *p;
79     for(p=rt;*s;++s)
80     {
81         if(!p->nxt[*s])
82             p->nxt[*s]=new node();
83         p=p->nxt[*s];
84     }
85     p->idx=idx;
86 }
87 
88 inline void make()
89 {
90     Q.push(rt);
91     static node *p,*q;
92     static int i;
93     while(!Q.empty())
94     {
95         p=Q.front();
96         Q.pop();
97         for(i=0;i<N;++i)
98             if(p->nxt[i])
99             {
100                 q=p->fal;
101                 while(q)
102                 {
103                     if(q->nxt[i])
104                     {
105                         p->nxt[i]->fal=q->nxt[i];
106                         break;
107                     }
108                     q=q->fal;
109                 }
110                 if(!q)
```

```
111                     p->nxt[i]->fal=rt;
112                 Q.push(p->nxt[i]);
113             }
114     }
115 }
116
117 inline void match(const char *s)
118 {
119     static node *p,*q;
120     for(p=rt;*s;++s)
121     {
122         while(p!=rt && !p->nxt[*s])
123             p=p->fal;
124         p=p->nxt[*s];
125         if(!p)
126             p=rt;
127         for(q=p;q!=rt && q->idx;q=q->fal) // why q->idx ? looks
               like not necessary at all, I delete it in an other
               solution
128             ++cnt[q->idx];
129     }
130 }
131
132 //可以考虑 dfs 一下，拉直 fal 指针来跳过无效的匹配
133 //在线调整关键字存在性的时候，可以考虑欧拉序压扁之后使用 BIT 或者线段树进行区间
    修改
134 //fal 指针构成的是一颗树，从匹配到的节点到树根都数一次
```

6.2 Gusfield's Z Algorithm

```
 1 inline void make(int *z,char *buf)
 2 {
 3     int i,j,l,r;
 4     l=0;
 5     r=1;
 6     z[0]=strlen(buf);
 7     for(i=1;i<z[0];++i)
 8         if(r<=i || z[i-l]>=r-i)
 9         {
10             j=std::max(i,r);
11             while(j<z[0] && buf[j]==buf[j-i])
12                 ++j;
13             z[i]=j-i;
14             if(i<j)
15             {
16                 l=i;
17                 r=j;
18             }
19         }
20         else
21             z[i]=z[i-l];
22 }
```

```
23
24 for(i=1;i<len && i+z[i]<len;++i); //i= 可能最小循环节长度
```

6.3 Manacher's Algorithm

```
1  inline int match(const int a,const int b,const std::vector<int> &
       str)
2  {
3      static int i;
4      i=0;
5      while(a-i>=0 && b+i<str.size() && str[a-i]==str[b+i])//注意是 i
           不是 1，打错过很多次了
6          ++i;
7      return i;
8  }
9
10 inline void go(int *z,const std::vector<int> &str)
11 {
12     static int c,l,r,i,ii,n;
13     z[0]=1;
14     c=l=r=0;
15     for(i=1;i<str.size();++i)
16     {
17         ii=(l<<1)-i;
18         n=r+1-i;
19
20         if(i>r)
21         {
22             z[i]=match(i,i,str);
23             l=i;
24             r=i+z[i]-1;
25         }
26         else
27             if(z[ii]==n)
28             {
29                 z[i]=n+match(i-n,i+n,str);
30                 l=i;
31                 r=i+z[i]-1;
32             }
33             else
34                 z[i]=std::min(z[ii],n);
35         if(z[i]>z[c])
36             c=i;
37     }
38 }
39
40 inline bool check(int *z,int a,int b) //检查子串 [a,b] 是否回文
41 {
42     a=a*2-1;
43     b=b*2-1;
44     int m=(a+b)/2;
45     return z[m]>=b-m+1;
```

```
}
```

6.4 Morris-Pratt Algorithm

```
inline void make(char *buf,int *fal)
{
    static int i,j;
    fal[0]=-1;
    for(i=1,j=-1;buf[i];++i)
    {
        while(j>=0 && buf[j+1]!=buf[i])
            j=fal[j];
        if(buf[j+1]==buf[i])
            ++j;
        fal[i]=j;
    }
}

inline int match(char *p,char *t,int* fal)
{
    static int i,j,re;
    re=0;
    for(i=0,j=-1;t[i];++i)
    {
        while(j>=0 && p[j+1]!=t[i])
            j=fal[j];
        if(p[j+1]==t[i])
            ++j;
        if(!p[j+1])
        {
            ++re;
            j=fal[j];
        }
    }
    return re;
}

inline void make(char *buf,int *fal) // knuth-morris-pratt, not tested yet
{
    static int i,j;
    fal[0]=-1;
    for(i=1,j=-1;buf[i];++i)
    {
        while(j>=0 && buf[j+1]!=buf[i])
            j=fal[j];
        if(buf[j+1]==buf[i])
            ++j;
        fal[i]=j;
    }
    for(i-=2;i>=0;--i)
    {
```

```
48          for(j=fal[i];j!=-1 && buf[j+1]!=buf[i+1];j=fal[j]);
49          fal[i]=j;
50      }
51  }
```

6.5 smallest representation

```
1  int min(char a[],int len)
2  {
3      int i = 0,j = 1,k = 0;
4      while (i < len && j < len && k < len)
5      {
6          int cmp = a[(j+k)%len]-a[(i+k)%len];
7          if (cmp == 0)
8              k++;
9          else
10         {
11             if (cmp > 0)
12                 j += k+1;
13             else
14                 i += k+1;
15             if (i == j) j++;
16             k = 0;
17         }
18     }
19     return std::min(i,j);
20 }
```

6.6 Suffix Array - DC3 Algorithm

```
1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4
5  #define MAXX 1111
6  #define F(x) ((x)/3+((x)%3==1?0:tb))
7  #define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
8
9  int wa[MAXX],wb[MAXX],wv[MAXX],ws[MAXX];
10
11 inline bool c0(const int *str,const int &a,const int &b)
12 {
13     return str[a]==str[b] && str[a+1]==str[b+1] && str[a+2]==str[b
       +2];
14 }
15
16 inline bool c12(const int *str,const int &k,const int &a,const int
    &b)
17 {
18     if(k==2)
19         return str[a]<str[b] || str[a]==str[b] && c12(str,1,a+1,b
           +1);
20     else
```

```
21          return str[a]<str[b] || str[a]==str[b] && wv[a+1]<wv[b+1];
22  }
23
24  inline void sort(int *str,int *a,int *b,const int &n,const int &m)
25  {
26      memset(ws,0,sizeof(ws));
27      int i;
28      for(i=0;i<n;++i)
29          ++ws[wv[i]=str[a[i]]];
30      for(i=1;i<m;++i)
31          ws[i]+=ws[i-1];
32      for(i=n-1;i>=0;--i)
33          b[--ws[wv[i]]]=a[i];
34  }
35
36  inline void dc3(int *str,int *sa,const int &n,const int &m)
37  {
38      int *strn(str+n);
39      int *san(sa+n),tb((n+1)/3),ta(0),tbc(0),i,j,k;
40      str[n]=str[n+1]=0;
41      for(i=0;i<n;++i)
42          if(i%3)
43              wa[tbc++]=i;
44      sort(str+2,wa,wb,tbc,m);
45      sort(str+1,wb,wa,tbc,m);
46      sort(str,wa,wb,tbc,m);
47      for(i=j=1,strn[F(wb[0])]=0;i<tbc;++i)
48          strn[F(wb[i])]=c0(str,wb[i-1],wb[i])?j-1:j++;
49      if(j<tbc)
50          dc3(strn,san,tbc,j);
51      else
52          for(i=0;i<tbc;++i)
53              san[strn[i]]=i;
54      for(i=0;i<tbc;++i)
55          if(san[i]<tb)
56              wb[ta++]=san[i]*3;
57      if(n%3==1)
58          wb[ta++]=n-1;
59      sort(str,wb,wa,ta,m);
60      for(i=0;i<tbc;++i)
61          wv[wb[i]=G(san[i])]=i;
62      for(i=j=k=0;i<ta && j<tbc;)
63          sa[k++]=c12(str,wb[j]%3,wa[i],wb[j])?wa[i++]:wb[j++];
64      while(i<ta)
65          sa[k++]=wa[i++];
66      while(j<tbc)
67          sa[k++]=wb[j++];
68  }
69
70  int rk[MAXX],lcpa[MAXX],sa[MAXX*3];
71  int str[MAXX*3]; //必须int
```

```
 72
 73 int main()
 74 {
 75     scanf("%d␣%d",&n,&j);
 76     for(i=0;i<n;++i)
 77     {
 78         scanf("%d",&k);
 79         num[i]=k-j+100;
 80         j=k;
 81     }
 82     num[n]=0;
 83
 84     dc3(num,sa,n+1,191); //191: str 中取值范围，桶排序
 85
 86     for(i=1;i<=n;++i) // rank 数组
 87         rk[sa[i]]=i;
 88     for(i=k=0;i<n;++i) // lcp 数组
 89         if(!rk[i])
 90             lcpa[0]=0;
 91         else
 92         {
 93             j=sa[rk[i]-1];
 94             if(k>0)
 95                 --k;
 96             while(num[i+k]==num[j+k])
 97                 ++k;
 98             lcpa[rk[i]]=k;
 99         }
100
101
102     for(i=1;i<=n;++i)
103         sptb[0][i]=i;
104     for(i=1;i<=lg[n];++i)  //sparse table RMQ
105     {
106         k=n+1-(1<<i);
107         for(j=1;j<=k;++j)
108         {
109             a=sptb[i-1][j];
110             b=sptb[i-1][j+(1<<(i-1))];
111             sptb[i][j]=lcpa[a]<lcpa[b]?a:b;
112         }
113     }
114 }
115
116 inline int ask(int l,int r)
117 {
118     a=lg[r-l+1];
119     r-=(1<<a)-1;
120     l=sptb[a][l];
121     r=sptb[a][r];
122     return lcpa[l]<lcpa[r]?l:r;
```

```
}

inline int lcp(int l,int r) // 字符串上 [l,r] 区间的 rmq
{
    l=rk[l];
    r=rk[r];
    if(l>r)
        std::swap(l,r);
    return lcpa[ask(l+1,r)];
}
```

6.7 Suffix Array - Prefix-doubling Algorithm

```
int wx[maxn],wy[maxn],*x,*y,wss[maxn],wv[maxn];

bool cmp(int *r,int n,int a,int b,int l)
{
    return a+l<n && b+l<n && r[a]==r[b]&&r[a+l]==r[b+l];
}
void da(int str[],int sa[],int rank[],int height[],int n,int m)
{
    int *s = str;
    int *x=wx,*y=wy,*t,p;
    int i,j;
    for(i=0; i<m; i++)
        wss[i]=0;
    for(i=0; i<n; i++)
        wss[x[i]=s[i]]++;
    for(i=1; i<m; i++)
        wss[i]+=wss[i-1];
    for(i=n-1; i>=0; i--)
        sa[--wss[x[i]]]=i;
    for(j=1,p=1; p<n && j<n; j*=2,m=p)
    {
        for(i=n-j,p=0; i<n; i++)
            y[p++]=i;
        for(i=0; i<n; i++)
            if(sa[i]-j>=0)
                y[p++]=sa[i]-j;
        for(i=0; i<n; i++)
            wv[i]=x[y[i]];
        for(i=0; i<m; i++)
            wss[i]=0;
        for(i=0; i<n; i++)
            wss[wv[i]]++;
        for(i=1; i<m; i++)
            wss[i]+=wss[i-1];
        for(i=n-1; i>=0; i--)
            sa[--wss[wv[i]]]=y[i];
        for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0; i<n; i++)
            x[sa[i]]=cmp(y,n,sa[i-1],sa[i],j)?p-1:p++;
    }
```

```
    for(int i=0; i<n; i++)
        rank[sa[i]]=i;
    for(int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
        if(rank[i]>0)
            for(k?k--:0,j=sa[rank[i]-1]; i+k < n && j+k < n && str[
               i+k]==str[j+k]; ++k);
}
```

6.8 Suffix Automaton

```
/*
length(s) ∈ [ min(s), max(s) ] = [ val[fal[s]]+1, val[s] ]
 */
#define MAXX 90111
#define MAXN (MAXX<<1)

int fal[MAXN],nxt[MAXN][26],val[MAXN],cnt,rt,last;

inline int neww(int v=0)
{
    val[++cnt]=v;
    fal[cnt]=0;
    memset(nxt[cnt],0,sizeof nxt[0]);
    return cnt;
}

inline void add(int w)
{
    static int p,np,q,nq;
    p=last;
    last=np=neww(val[p]+1);
    while(p && !nxt[p][w])
    {
        nxt[p][w]=np;
        p=fal[p];
    }
    if(!p)
        fal[np]=rt;
    else
    {
        q=nxt[p][w];
        if(val[p]+1==val[q])
            fal[np]=q;
        else
        {
            nq=neww(val[p]+1);
            memcpy(nxt[nq],nxt[q],sizeof nxt[0]);
            fal[nq]=fal[q];

            fal[q]=fal[np]=nq;
            while(p && nxt[p][w]==q)
            {
```

```
                nxt[p][w]=nq;
                p=fal[p];
            }
        }
    }
}

int v[MAXN],the[MAXN];

inline void make(char *str)
{
    cnt=0;
    rt=last=neww();
    static int i,len,now;
    for(i=0;str[i];++i)
        add(str[i]-'a');
    len=i;
    memset(v,0,sizeof v);
    for(i=1;i<=cnt;++i)
        ++v[val[i]];
    for(i=1;i<=len;++i)
        v[i]+=v[i-1];
    for(i=1;i<=cnt;++i)
        the[v[val[i]]--]=i;
    for(i=cnt;i;--i)
    {
        now=the[i];
        // topsort already
    }
}
/*
sizeof right(s):
    init:
        for all np:
            count[np]=1;
    process:
        for all status s:
            count[fal[s]]+=count[s];
*/
```

7 Dynamic Programming

7.1 knapsack problem

```
multiple-choice knapsack problem:

for 所有的组k
    for v=V..0
 for 所有的 i 属于组 k
            f[v]=max{f[v],f[v-c[i]]+w[i]}
```

7.2 LCIS

```cpp
#include<cstdio>
#include<cstring>
#include<vector>

#define MAXX 1111

int T;
int n,m,p,i,j,k;
std::vector<int>the[2];
int dp[MAXX],path[MAXX];
int ans[MAXX];

int main()
{
    the[0].reserve(MAXX);
    the[1].reserve(MAXX);
    {
        scanf("%d",&n);
        the[0].resize(n);
        for(i=0;i<n;++i)
            scanf("%d",&the[0][i]);
        scanf("%d",&m);
        the[1].resize(m);
        for(i=0;i<m;++i)
            scanf("%d",&the[1][i]);
        memset(dp,0,sizeof dp);
        for(i=0;i<the[0].size();++i)
        {
            n=0;
            p=-1;
            for(j=0;j<the[1].size();++j)
            {
                if(the[0][i]==the[1][j] && n+1>dp[j])
                {
                    dp[j]=n+1;
                    path[j]=p;
                }
                if(the[1][j]<the[0][i] && n<dp[j])
                {
                    n=dp[j];
                    p=j;
                }
            }
        }
        n=0;
        p=-1;
        for(i=0;i<the[1].size();++i)
            if(dp[i]>n)
                n=dp[p=i];
        printf("%d\n",n);
        for(i=n-1;i>=0;--i)
```

```
52          {
53              ans[i]=the[1][p];
54              p=path[p];
55          }
56          for(i=0;i<n;++i)
57              printf("%d␣",ans[i]);
58          puts("");
59      }
60      return 0;
61  }
```

7.3 LCS

```
1   #include<cstdio>
2   #include<algorithm>
3   #include<vector>
4
5   #define MAXX 111
6   #define N 128
7
8   std::vector<char>the[2];
9   std::vector<int>dp(MAXX),p[N];
10
11  int i,j,k;
12  char buf[MAXX];
13  int t;
14
15  int main()
16  {
17      the[0].reserve(MAXX);
18      the[1].reserve(MAXX);
19      while(gets(buf),buf[0]!='#')
20      {
21          the[0].resize(0);
22          for(i=0;buf[i];++i)
23              the[0].push_back(buf[i]);
24          the[1].resize(0);
25          gets(buf);
26          for(i=0;buf[i];++i)
27              the[1].push_back(buf[i]);
28          for(i=0;i<N;++i)
29              p[i].resize(0);
30          for(i=0;i<the[1].size();++i)
31              p[the[1][i]].push_back(i);
32          dp.resize(1);
33          dp[0]=-1;
34          for(i=0;i<the[0].size();++i)
35              for(j=p[the[0][i]].size()-1;j>=0;--j)
36              {
37                  k=p[the[0][i]][j];
38                  if(k>dp.back())
39                      dp.push_back(k);
```

```
                else
                    *std::lower_bound(dp.begin(),dp.end(),k)=k;
            }
        printf("Case␣#%d:␣you␣can␣visit␣at␣most␣%ld␣cities.\n",++t,
           dp.size()-1);
    }
    return 0;
}
```

7.4 sequence partitioning

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

#define MAXX 40111

int a[MAXX],b[MAXX];
int n,R;
std::multiset<int>set;

inline bool check(const int g)
{
    static int i,j,k;
    static long long sum;
    static int l,r,q[MAXX],dp[MAXX];
    set.clear();
    q[0]=dp[0]=l=r=sum=0;
    for(j=i=1;i<=n;++i)
    {
        sum+=b[i];
        while(sum>g)
            sum-=b[j++];
        if(j>i)
            return false;
        while(l<r && q[l]<j)
        {
            ++l;
            if(l<r && set.count(dp[q[l-1]]+a[q[l]]))
                set.erase(set.find(dp[q[l-1]]+a[q[l]]));
        }
        while(l<r && a[q[r-1]]<=a[i])
        {
            --r;
            if(l<r && set.count(dp[q[r-1]]+a[q[r]]))
                set.erase(set.find(dp[q[r-1]]+a[q[r]]));
        }
        if(l<r)
            set.insert(dp[q[r-1]]+a[i]);
        q[r++]=i;
        dp[i]=dp[j-1]+a[q[l]];
```

```
42          if(r-l>1)
43              dp[i]=std::min(dp[i],*set.begin());
44      }
45      return dp[n]<=R;
46  }
47
48  int i,j,k;
49  long long l,r,mid,ans;
50
51  int main()
52  {
53      while(scanf("%d␣%d",&n,&R)!=EOF)
54      {
55          l=r=0;
56          for(i=1;i<=n;++i)
57          {
58              scanf("%d␣%d",a+i,b+i);
59              r+=b[i];
60          }
61          ans=-1;
62          while(l<=r)
63          {
64              mid=l+r>>1;
65              if(check(mid))
66              {
67                  ans=mid;
68                  r=mid-1;
69              }
70              else
71                  l=mid+1;
72          }
73          printf("%lld\n",ans);
74      }
75      return 0;
76  }
```

8 Search

8.1 dlx

```
1  精确覆盖：给定一个 01 矩阵，现在要选择一些行，使得每一列有且仅有一个 1。
2  每次选定一个元素个数最少的列，从该列中选择一行加入答案，删除该行所有的列以及与该
       行冲突的行。
3
4  重复覆盖：给定一个 01 矩阵，现在要选择一些行，使得每一列至少有一个 1。
5  每次选定一个元素个数最少的列，从该列中选择一行加入答案，删除该行所有的列。与该行
       冲突的行可能满足重复覆盖。
```

8.2 dlx - exact cover

```
1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
```

```
 4 #include<vector>
 5
 6 #define N 256
 7 #define MAXN N*22
 8 #define MAXM N*5
 9 #define inf 0x3f3f3f3f
10 const int MAXX(MAXN*MAXM);
11
12 bool mat[MAXN][MAXM];
13
14 int u[MAXX],d[MAXX],l[MAXX],r[MAXX],ch[MAXX],rh[MAXX];
15 int sz[MAXM];
16 std::vector<int>ans(MAXX);
17 int hd,cnt;
18
19 inline int node(int up,int down,int left,int right)
20 {
21     u[cnt]=up;
22     d[cnt]=down;
23     l[cnt]=left;
24     r[cnt]=right;
25     u[down]=d[up]=l[right]=r[left]=cnt;
26     return cnt++;
27 }
28
29 inline void init(int n,int m)
30 {
31     cnt=0;
32     hd=node(0,0,0,0);
33     static int i,j,k,r;
34     for(j=1;j<=m;++j)
35     {
36         ch[j]=node(cnt,cnt,l[hd],hd);
37         sz[j]=0;
38     }
39     for(i=1;i<=n;++i)
40     {
41         r=-1;
42         for(j=1;j<=m;++j)
43             if(mat[i][j])
44             {
45                 if(r==-1)
46                 {
47                     r=node(u[ch[j]],ch[j],cnt,cnt);
48                     rh[r]=i;
49                     ch[r]=ch[j];
50                 }
51                 else
52                 {
53                     k=node(u[ch[j]],ch[j],l[r],r);
54                     rh[k]=i;
```

```
                    ch[k]=ch[j];
                }
                ++sz[j];
            }
    }
}

inline void rm(int c)
{
    l[r[c]]=l[c];
    r[l[c]]=r[c];
    static int i,j;
    for(i=d[c];i!=c;i=d[i])
        for(j=r[i];j!=i;j=r[j])
        {
            u[d[j]]=u[j];
            d[u[j]]=d[j];
            --sz[ch[j]];
        }
}

inline void add(int c)
{
    static int i,j;
    for(i=u[c];i!=c;i=u[i])
        for(j=l[i];j!=i;j=l[j])
        {
            ++sz[ch[j]];
            u[d[j]]=d[u[j]]=j;
        }
    l[r[c]]=r[l[c]]=c;
}

bool dlx(int k)
{
    if(hd==r[hd])
    {
        ans.resize(k);
        return true;
    }
    int s=inf,c;
    int i,j;
    for(i=r[hd];i!=hd;i=r[i])
        if(sz[i]<s)
        {
            s=sz[i];
            c=i;
        }
    rm(c);
    for(i=d[c];i!=c;i=d[i])
    {
```

```
106         ans[k]=rh[i];
107         for(j=r[i];j!=i;j=r[j])
108             rm(ch[j]);
109         if(dlx(k+1))
110             return true;
111         for(j=l[i];j!=i;j=l[j])
112             add(ch[j]);
113     }
114     add(c);
115     return false;
116 }
117 
118 #include <cstdio>
119 #include <cstring>
120 
121 #define N 1024
122 #define M 1024*110
123 using namespace std;
124 
125 int l[M], r[M], d[M], u[M], col[M], row[M], h[M], res[N], cntcol[N
       ];
126 int dcnt = 0;
127 //初始化一个节点
128 inline void addnode(int &x)
129 {
130     ++x;
131     r[x] = l[x] = u[x] = d[x] = x;
132 }
133 //将加入到后xrowx
134 inline void insert_row(int rowx, int x)
135 {
136     r[l[rowx]] = x;
137     l[x] = l[rowx];
138     r[x] = rowx;
139     l[rowx] = x;
140 }
141 //将加入到后xcolx
142 inline void insert_col(int colx, int x)
143 {
144     d[u[colx]] = x;
145     u[x] = u[colx];
146     d[x] = colx;
147     u[colx] = x;
148 }
149 //全局初始化
150 inline void dlx_init(int cols)
151 {
152     memset(h, -1, sizeof(h));
153     memset(cntcol, 0, sizeof(cntcol));
154     dcnt = -1;
155     addnode(dcnt);
```

```
156     for (int i = 1; i <= cols; ++i)
157     {
158         addnode(dcnt);
159         insert_row(0, dcnt);
160     }
161 }
162 //删除一列以及相关的所有行
163 inline void remove(int c)
164 {
165     l[r[c]] = l[c];
166     r[l[c]] = r[c];
167     for (int i = d[c]; i != c; i = d[i])
168         for (int j = r[i]; j != i; j = r[j])
169         {
170             u[d[j]] = u[j];
171             d[u[j]] = d[j];
172             cntcol[col[j]]--;
173         }
174 }
175 //恢复一列以及相关的所有行
176 inline void resume(int c)
177 {
178     for (int i = u[c]; i != c; i = u[i])
179         for (int j = l[i]; j != i; j = l[j])
180         {
181             u[d[j]] = j;
182             d[u[j]] = j;
183             cntcol[col[j]]++;
184         }
185     l[r[c]] = c;
186     r[l[c]] = c;
187 }
188 //搜索部分
189 bool DLX(int deep)
190 {
191     if (r[0] == 0)
192     {
193 //Do anything you want to do here
194         printf("%d", deep);
195         for (int i = 0; i < deep; ++i) printf(" %d", res[i]);
196         puts("");
197         return true;
198     }
199     int min = INT_MAX, tempc;
200     for (int i = r[0]; i != 0; i = r[i])
201         if (cntcol[i] < min)
202         {
203             min = cntcol[i];
204             tempc = i;
205         }
206     remove(tempc);
```

```
207     for (int i = d[tempc]; i != tempc; i = d[i])
208     {
209         res[deep] = row[i];
210         for (int j = r[i]; j != i; j = r[j]) remove(col[j]);
211         if (DLX(deep + 1)) return true;
212         for (int j = l[i]; j != i; j = l[j]) resume(col[j]);
213     }
214     resume(tempc);
215     return false;
216 }
217 //插入矩阵中的节点"1"
218 inline void insert_node(int x, int y)
219 {
220     cntcol[y]++;
221     addnode(dcnt);
222     row[dcnt] = x;
223     col[dcnt] = y;
224     insert_col(y, dcnt);
225     if (h[x] == -1) h[x] = dcnt;
226     else insert_row(h[x], dcnt);
227 }
228 int main()
229 {
230     int n, m;
231     while (~scanf("%d%d", &n, &m))
232     {
233         dlx_init(m);
234         for (int i = 1; i <= n; ++i)
235         {
236             int k, x;
237             scanf("%d", &k);
238             while (k--)
239             {
240                 scanf("%d", &x);
241                 insert_node(i, x);
242             }
243         }
244         if (!DLX(0))
245             puts("NO");
246     }
247     return 0;
248 }
```

8.3 dlx - repeat cover

```
1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4
5 #define MAXN 110
6 #define MAXM 1000000
7 #define INF 0x7FFFFFFF
```

```
 8
 9 using namespace std;
10
11 int G[MAXN][MAXN];
12 int L[MAXM], R[MAXM], U[MAXM], D[MAXM];
13 int size, ans, S[MAXM], H[MAXM], C[MAXM];
14 bool vis[MAXN * 100];
15 void Link(int r, int c)
16 {
17     U[size] = c;
18     D[size] = D[c];
19     U[D[c]] = size;
20     D[c] = size;
21     if (H[r] < 0)
22         H[r] = L[size] = R[size] = size;
23     else
24     {
25         L[size] = H[r];
26         R[size] = R[H[r]];
27         L[R[H[r]]] = size;
28         R[H[r]] = size;
29     }
30     S[c]++;
31     C[size++] = c;
32 }
33 void Remove(int c)
34 {
35     int i;
36     for (i = D[c]; i != c; i = D[i])
37     {
38         L[R[i]] = L[i];
39         R[L[i]] = R[i];
40     }
41 }
42 void Resume(int c)
43 {
44     int i;
45     for (i = D[c]; i != c; i = D[i])
46         L[R[i]] = R[L[i]] = i;
47 }
48 int A()
49 {
50     int i, j, k, res;
51     memset(vis, false, sizeof(vis));
52     for (res = 0, i = R[0]; i; i = R[i])
53     {
54         if (!vis[i])
55         {
56             res++;
57             for (j = D[i]; j != i; j = D[j])
58             {
```

```
                for (k = R[j]; k != j; k = R[k])
                    vis[C[k]] = true;
            }
        }
    }
    return res;
}
void Dance(int now)
{
    if (R[0] == 0)
        ans = min(ans, now);
    else if (now + A() < ans)
    {
        int i, j, temp, c;
        for (temp = INF,i = R[0]; i; i = R[i])
        {
            if (temp > S[i])
            {
                temp = S[i];
                c = i;
            }
        }
        for (i = D[c]; i != c; i = D[i])
        {
            Remove(i);
            for (j = R[i]; j != i; j = R[j])
                Remove(j);
            Dance(now + 1);
            for (j = L[i]; j != i; j = L[j])
                Resume(j);
            Resume(i);
        }
    }
}
void Init(int m)
{
    int i;
    for (i = 0; i <= m; i++)
    {
        R[i] = i + 1;
        L[i + 1] = i;
        U[i] = D[i] = i;
        S[i] = 0;
    }
    R[m] = 0;
    size = m + 1;
}
```

8.4 fibonacci knapsack

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

```
 3 #include<algorithm>
 4
 5 #define MAXX 71
 6
 7 struct mono
 8 {
 9     long long weig,cost;
10 }goods[MAXX];
11
12 int n,T,t,i;
13 long long carry,sumw,sumc;
14 long long ans,las[MAXX];
15
16 bool comp(const struct mono a,const struct mono b)
17 {
18     if(a.weig!=b.weig)
19         return a.weig<b.weig;
20     return b.cost<a.cost;
21 }
22
23 void dfs(int i,long long cost_n,long long carry_n,int last)
24 {
25     if(ans<cost_n)
26         ans=cost_n;
27     if(i==n || goods[i].weig>carry_n || cost_n+las[i]<=ans)
28         return;
29     if(last || (goods[i].weig!=goods[i-1].weig && goods[i].cost>
       goods[i-1].cost))
30         dfs(i+1,cost_n+goods[i].cost,carry_n-goods[i].weig,1);
31     dfs(i+1,cost_n,carry_n,0);
32 }
33
34 int main()
35 {
36     scanf("%d",&T);
37     for(t=1;t<=T;++t)
38     {
39         scanf("%d␣%lld",&n,&carry);
40         sumw=0;
41         sumc=0;
42         ans=0;
43         for(i=0;i<n;++i)
44         {
45             scanf("%lld␣%lld",&goods[i].weig,&goods[i].cost);
46             sumw+=goods[i].weig;
47             sumc+=goods[i].cost;
48         }
49         if(sumw<=carry)
50         {
51             printf("Case␣%d:␣%lld\n",t,sumc);
52             continue;
```

```
53          }
54          std::sort(goods,goods+n,comp);
55          for(i=0;i<n;++i)
56          {
57              las[i]=sumc;
58              sumc-=goods[i].cost;
59          }
60          dfs(0,0,carry,1);
61          printf("Case␣%d:␣%lld\n",t,ans);
62      }
63      return 0;
64  }
```

9 Others

9.1 .vimrc

```
1  set number
2  set history=1000000
3  set autoindent
4  set smartindent
5  set tabstop=4
6  set shiftwidth=4
7  set expandtab
8  set showmatch
9
10 set nocp
11 filetype plugin indent on
12
13 filetype on
14 syntax on
```

9.2 bigint

```
1  // header files
2  #include <cstdio>
3  #include <string>
4  #include <algorithm>
5  #include <iostream>
6
7  struct Bigint
8  {
9      // representations and structures
10     std::string a; // to store the digits
11     int sign; // sign = -1 for negative numbers, sign = 1 otherwise
12     // constructors
13     Bigint() {} // default constructor
14     Bigint( std::string b ) { (*this) = b; } // constructor for std
           ::string
15     // some helpful methods
16     int size() // returns number of digits
17     {
18         return a.size();
```

```
19      }
20      Bigint inverseSign() // changes the sign
21      {
22          sign *= -1;
23          return (*this);
24      }
25      Bigint normalize( int newSign ) // removes leading 0, fixes
           sign
26      {
27          for( int i = a.size() - 1; i > 0 && a[i] == '0'; i-- )
28              a.erase(a.begin() + i);
29          sign = ( a.size() == 1 && a[0] == '0' ) ? 1 : newSign;
30          return (*this);
31      }
32      // assignment operator
33      void operator = ( std::string b ) // assigns a std::string to
           Bigint
34      {
35          a = b[0] == '-' ? b.substr(1) : b;
36          reverse( a.begin(), a.end() );
37          this->normalize( b[0] == '-' ? -1 : 1 );
38      }
39      // conditional operators
40      bool operator < ( const Bigint &b ) const // less than operator
41      {
42          if( sign != b.sign )
43              return sign < b.sign;
44          if( a.size() != b.a.size() )
45              return sign == 1 ? a.size() < b.a.size() : a.size() > b
                   .a.size();
46          for( int i = a.size() - 1; i >= 0; i-- )
47              if( a[i] != b.a[i] )
48                  return sign == 1 ? a[i] < b.a[i] : a[i] > b.a[i];
49          return false;
50      }
51      bool operator == ( const Bigint &b ) const // operator for
           equality
52      {
53          return a == b.a && sign == b.sign;
54      }
55
56      // mathematical operators
57      Bigint operator + ( Bigint b ) // addition operator overloading
58      {
59          if( sign != b.sign )
60              return (*this) - b.inverseSign();
61          Bigint c;
62          for(int i = 0, carry = 0; i<a.size() || i<b.size() || carry
               ; i++ )
63          {
```

```
64              carry+=(i<a.size() ? a[i]-48 : 0)+(i<b.a.size() ? b.a[i
                    ]-48 : 0);
65              c.a += (carry % 10 + 48);
66              carry /= 10;
67          }
68          return c.normalize(sign);
69      }
70
71      Bigint operator - ( Bigint b ) // subtraction operator
            overloading
72      {
73          if( sign != b.sign )
74              return (*this) + b.inverseSign();
75          int s = sign; sign = b.sign = 1;
76          if( (*this) < b )
77              return ((b - (*this)).inverseSign()).normalize(-s);
78          Bigint c;
79          for( int i = 0, borrow = 0; i < a.size(); i++ )
80          {
81              borrow = a[i] - borrow - (i < b.size() ? b.a[i] : 48);
82              c.a += borrow >= 0 ? borrow + 48 : borrow + 58;
83              borrow = borrow >= 0 ? 0 : 1;
84          }
85          return c.normalize(s);
86      }
87      Bigint operator * ( Bigint b ) // multiplication operator
            overloading
88      {
89          Bigint c("0");
90          for( int i = 0, k = a[i] - 48; i < a.size(); i++, k = a[i]
                - 48 )
91          {
92              while(k--)
93                  c = c + b; // ith digit is k, so, we add k times
94              b.a.insert(b.a.begin(), '0'); // multiplied by 10
95          }
96          return c.normalize(sign * b.sign);
97      }
98      Bigint operator / ( Bigint b ) // division operator overloading
99      {
100         if( b.size() == 1 && b.a[0] == '0' )
101             b.a[0] /= ( b.a[0] - 48 );
102         Bigint c("0"), d;
103         for( int j = 0; j < a.size(); j++ )
104             d.a += "0";
105         int dSign = sign * b.sign;
106         b.sign = 1;
107         for( int i = a.size() - 1; i >= 0; i-- )
108         {
109             c.a.insert( c.a.begin(), '0');
110             c = c + a.substr( i, 1 );
```

```
111             while( !( c < b ) )
112             {
113                 c = c - b;
114                 d.a[i]++;
115             }
116         }
117         return d.normalize(dSign);
118     }
119     Bigint operator % ( Bigint b ) // modulo operator overloading
120     {
121         if( b.size() == 1 && b.a[0] == '0' )
122             b.a[0] /= ( b.a[0] - 48 );
123         Bigint c("0");
124         b.sign = 1;
125         for( int i = a.size() - 1; i >= 0; i-- )
126         {
127             c.a.insert( c.a.begin(), '0');
128             c = c + a.substr( i, 1 );
129             while( !( c < b ) )
130                 c = c - b;
131         }
132         return c.normalize(sign);
133     }
134 
135     // output method
136     void print()
137     {
138         if( sign == -1 )
139             putchar('-');
140         for( int i = a.size() - 1; i >= 0; i-- )
141             putchar(a[i]);
142     }
143 };
144 
145 
146 
147 int main()
148 {
149     Bigint a, b, c; // declared some Bigint variables
150     //////////////////////////
151     // taking Bigint input //
152     //////////////////////////
153 
154     std::string input; // std::string to take input
155     std::cin >> input; // take the Big integer as std::string
156     a = input; // assign the std::string to Bigint a
157 
158     std::cin >> input; // take the Big integer as std::string
159     b = input; // assign the std::string to Bigint b
160 
161     ////////////////////////////////
```

```
162     // Using mathematical operators //
163     ////////////////////////////////
164
165     c = a + b; // adding a and b
166     c.print(); // printing the Bigint
167     puts(""); // newline
168
169     c = a - b; // subtracting b from a
170     c.print(); // printing the Bigint
171     puts(""); // newline
172
173     c = a * b; // multiplying a and b
174     c.print(); // printing the Bigint
175     puts(""); // newline
176
177     c = a / b; // dividing a by b
178     c.print(); // printing the Bigint
179     puts(""); // newline
180
181     c = a % b; // a modulo b
182     c.print(); // printing the Bigint
183     puts(""); // newline
184
185     ////////////////////////////////
186     // Using conditional operators //
187     ////////////////////////////////
188
189     if( a == b )
190         puts("equal"); // checking equality
191     else
192         puts("not␣equal");
193
194     if( a < b )
195         puts("a␣is␣smaller␣than␣b"); // checking less than operator
196
197     return 0;
198 }
```

9.3 Binary Search

```
1  //[0,n)
2  inline int go(int A[],int n,int x) // return the least i that make
       A[i]==x;
3  {
4      static int l,r,mid,re;
5      l=0;
6      r=n-1;
7      re=-1;
8      while(l<=r)
9      {
10         mid=l+r>>1;
11         if(A[mid]<x)
```

```
12              l=mid+1;
13          else
14          {
15              r=mid-1;
16              if(A[mid]==x)
17                  re=mid;
18          }
19      }
20      return re;
21  }
22
23  inline int go(int A[],int n,int x) // return the largest i that
        make A[i]==x;
24  {
25      static int l,r,mid,re;
26      l=0;
27      r=n-1;
28      re=-1;
29      while(l<=r)
30      {
31          mid=l+r>>1;
32          if(A[mid]<=x)
33          {
34              l=mid+1;
35              if(A[mid]==x)
36                  re=mid;
37          }
38          else
39              r=mid-1;
40      }
41      return re;
42  }
43
44  inline int go(int A[],int n,int x) // retrun the largest i that
        make A[i]<x;
45  {
46      static int l,r,mid,re;
47      l=0;
48      r=n-1;
49      re=-1;
50      while(l<=r)
51      {
52          mid=l+r>>1;
53          if(A[mid]<x)
54          {
55              l=mid+1;
56              re=mid;
57          }
58          else
59              r=mid-1;
60      }
```

```
61      return re;
62  }
63
64  inline int go(int A[],int n,int x)// return the largest i that make
        A[i]<=x;
65  {
66      static int l,r,mid,re;
67      l=0;
68      r=n-1;
69      re=-1;
70      while(l<=r)
71      {
72          mid=l+r>>1;
73          if(A[mid]<=x)
74          {
75              l=mid+1;
76              re=mid;
77          }
78          else
79              r=mid-1;
80      }
81      return re;
82  }
83
84  inline int go(int A[],int n,int x)// return the least i that make A
       [i]>x;
85  {
86      static int l,r,mid,re;
87      l=0;
88      r=n-1;
89      re=-1;
90      while(l<=r)
91      {
92          mid=l+r>>1;
93          if(A[mid]<=x)
94              l=mid+1;
95          else
96          {
97              r=mid-1;
98              re=mid;
99          }
100     }
101     return re;
102 }
103
104 inline int go(int A[],int n,int x)// upper_bound();
105 {
106     static int l,r,mid;
107     l=0;
108     r=n-1;
109     while(l<r)
```

```
110     {
111         mid=l+r>>1;
112         if(A[mid]<=x)
113             l=mid+1;
114         else
115             r=mid;
116     }
117     return r;
118 }
119
120 inline int go(int A[],int n,int x)// lower_bound();
121 {
122     static int l,r,mid,;
123     l=0;
124     r=n-1;
125     while(l<r)
126     {
127         mid=l+r>>1;
128         if(A[mid]<x)
129             l=mid+1;
130         else
131             r=mid;
132     }
133     return r;
134 }
```

9.4 java

```
1  //Scanner
2
3  Scanner in=new Scanner(new FileReader("asdf"));
4  PrintWriter pw=new PrintWriter(new Filewriter("out"));
5  boolean       in.hasNext();
6  String        in.next();
7  BigDecimal    in.nextBigDecimal();
8  BigInteger    in.nextBigInteger();
9  BigInteger    in.nextBigInteger(int radix);
10 double        in.nextDouble();
11 int           in.nextInt();
12 int           in.nextInt(int radix);
13 String        in.nextLine();
14 long          in.nextLong();
15 long          in.nextLong(int radix);
16 short         in.nextShort();
17 short         in.nextShort(int radix);
18 int           in.radix(); //Returns this scanner's default radix.
19 Scanner       in.useRadix(int radix);// Sets this scanner's default
       radix to the specified radix.
20 void          in.close();//Closes this scanner.
21
22 //String
23
```

```
char          str.charAt(int index);
int           str.compareTo(String anotherString); // <0 if less. ==0 if equal. >0 if greater.
int           str.compareToIgnoreCase(String str);
String        str.concat(String str);
boolean       str.contains(CharSequence s);
boolean       str.endsWith(String suffix);
boolean       str.startsWith(String preffix);
boolean       str.startsWith(String preffix,int toffset);
int           str.hashCode();
int           str.indexOf(int ch);
int           str.indexOf(int ch,int fromIndex);
int           str.indexOf(String str);
int           str.indexOf(String str,int fromIndex);
int           str.lastIndexOf(int ch);
int           str.lastIndexOf(int ch,int fromIndex);
//(ry
int           str.length();
String        str.substring(int beginIndex);
String        str.substring(int beginIndex,int endIndex);
String        str.toLowerCase();
String        str.toUpperCase();
String        str.trim();// Returns a copy of the string, with leading and trailing whitespace omitted.

//StringBuilder
StringBuilder str.insert(int offset,...);
StringBuilder str.reverse();
void          str.setCharAt(int index,int ch);

//BigInteger
compareTo(); equals(); doubleValue(); longValue(); hashCode(); toString(); toString(int radix); max(); min(); mod(); modPow(BigInteger exp,BigInteger m); nextProbablePrime(); pow();
andNot(); and(); xor(); not(); or(); getLowestSetBit(); bitCount(); bitLength(); setBig(int n); shiftLeft(int n); shiftRight(int n);
add(); divide(); divideAndRemainder(); remainder(); multiply(); subtract(); gcd(); abs(); signum(); negate();

//BigDecimal
movePointLeft(); movePointRight(); precision(); stripTrailingZeros(); toBigInteger(); toPlainString();

import java.util.*;

//sort
class pii implements Comparable
{
    public int a,b;
    public int compareTo(Object i)
```

```
67      {
68          pii c=(pii)i;
69          return a==c.a?c.b-b:c.a-a;
70      }
71  }
72
73  class Main
74  {
75      public static void main(String[] args)
76      {
77          pii[] the=new pii[2];
78          the[0]=new pii();
79          the[1]=new pii();
80          the[0].a=1;
81          the[0].b=1;
82          the[1].a=1;
83          the[1].b=2;
84          Arrays.sort(the);
85          for(int i=0;i<2;++i)
86              System.out.printf("%d␣%d\n",the[i].a,the[i].b);
87      }
88  }
89
90  //fraction
91  class frac
92  {
93      public BigInteger a,b;
94      public frac(long aa,long bb)
95      {
96          a=BigInteger.valueOf(aa);
97          b=BigInteger.valueOf(bb);
98          BigInteger c=a.gcd(b);
99          a=a.divide(c);
100         b=b.divide(c);
101     }
102     public frac(BigInteger aa,BigInteger bb)
103     {
104         BigInteger c=aa.gcd(bb);
105         a=aa.divide(c);
106         b=bb.divide(c);
107     }
108     public frac mul(frac i)
109     {
110         return new frac(a.multiply(i.a),b.multiply(i.b));
111     }
112     public frac mul(long i)
113     {
114         return new frac(a.multiply(BigInteger.valueOf(i)),b);
115     }
116     public frac div(long i)
117     {
```

```
118         return new frac(a,b.multiply(BigInteger.valueOf(i)));
119     }
120     public frac add(frac i)
121     {
122         return new frac((a.multiply(i.b)).add(i.a.multiply(b)),b.
               multiply(i.b));
123     }
124     public void print()
125     {
126         System.out.println(a+"/"+b); //printf 会 PE 啊尼玛死……
127     }
128 }
```

9.5 others

```
1  god damn it windows:
2  #pragma comment(linker, "/STACK:16777216")
3  #pragma comment(linker,"/STACK:102400000,102400000")
4
5
6  chmod +x [filename]
7
8  while true; do
9  ./gen > input
10 ./sol < input > output.sol
11 ./bf < input > output.bf
12
13 diff output.sol output.bf
14 if [ $? -ne 0 ]; then break; fi
15 done
16
17
18
```

1. nothing to be afraid of, 'cause you love it. isn't it?
2. calm_down();calm_down();calm_down();
3. 读完题目读完题目读完题目
   (a) 认真读题、认真读题、认真读题、认真读题、
   (b) 不盲目跟版
   (c) 换题/换想法
4. 对数/离线/hash/观察问题本身/点 $\leftrightarrow$ 区间互转
   (a) 对数调整精度 or 将乘法转换成加法
   (b) 点化区间，区间化点
5. 数组大小……
6. 写解释器/编译器的时候别忘了负数
   (a) 还有 istringstream in <sstream>

(b) 指令/函数名也可能是变量名

7. vector 比 array 慢很多

8. modPow 比手写快速幂慢很多

9. 对于 bool 数组，memset 快 8 倍