

Code Library



Himemiyu Nanao @ Perfect Freeze

August 18, 2013

Contents

1	Data Structure	1	4.16 k Shortest Path	35
1.1	atlantis	1	4.17 Kariv-Hakimi Algorithm	36
1.2	Binary Indexed tree	2	4.18 Kuhn-Munkres algorithm	37
1.3	COT	2	4.19 LCA - DA	37
1.4	hose	3	4.20 LCA - tarjan - minmax	38
1.5	Leftist tree	3	4.21 Minimum Ratio Spanning Tree	38
1.6	Network	4	4.22 Minimum Steiner Tree	39
1.7	OTOCI	5	4.23 Minimum-cost flow problem	40
1.8	picture	6	4.24 Second-best MST	40
1.9	Size Blanched Tree	7	4.25 Spanning tree	40
1.10	Sparse Table - rectangle	8	4.26 Stable Marriage	41
1.11	Sparse Table - square	8	4.27 Stoer-Wagner Algorithm	41
1.12	Sparse Table	9	4.28 Strongly Connected Component	41
1.13	Treap	9	4.29 ZKW's Minimum-cost flow	41
2	Geometry	10	5 Math	42
2.1	3D	10	5.1 cantor	42
2.2	3DCH	11	5.2 Discrete logarithms - BSGS	43
2.3	circle's area	12	5.3 Divisor function	43
2.4	circle	13	5.4 Extended Euclidean Algorithm	43
2.5	closest point pair	14	5.5 Fast Fourier Transform	43
2.6	ellipse	15	5.6 Gaussian elimination	44
2.7	Graham's scan	15	5.7 inverse element	45
2.8	half-plane intersection	16	5.8 Linear programming	45
2.9	intersection of circle and poly	16	5.9 Lucas' theorem(2)	45
2.10	k-d tree	17	5.10 Lucas' theorem	46
2.11	Manhattan MST	17	5.11 Matrix	46
2.12	others	18	5.12 Miller-Rabin Algorithm	46
2.13	Pick's theorem	19	5.13 Multiset	47
2.14	PointInPoly	19	5.14 Pell's equation	47
2.15	rotating caliper	19	5.15 Pollard's rho algorithm	47
2.16	shit	20	5.16 Prime	48
2.17	sort - polar angle	21	5.17 Reduced Residue System	48
2.18	triangle	21	5.18 Simpson's rule	49
			5.19 System of linear congruences	49
3	Geometry/tmp	21	6 String	49
3.1	circle	21	6.1 Aho-Corasick Algorithm	49
3.2	circles	23	6.2 Gusfield's Z Algorithm	50
3.3	halfplane	23	6.3 Manacher's Algorithm	50
3.4	line	24	6.4 Morris-Pratt Algorithm	50
3.5	line3d	24	6.5 smallest representation	50
3.6	plane	25	6.6 Suffix Array - DC3 Algorithm	51
3.7	point	25	6.7 Suffix Array - Prefix-doubling Algorithm	51
3.8	point3d	26	6.8 Suffix Automaton	52
3.9	polygon	26		
3.10	polygons	28	7 Dynamic Programming	52
			7.1 knapsack problem	52
			7.2 LCIS	52
4	Graph	29	8 Search	52
4.1	2SAT	29	8.1 dlx	52
4.2	Articulation	29	8.2 dlx - exact cover	53
4.3	Augmenting Path Algorithm for Maximum Cardinality Bipartite Matching	29	8.3 dlx - repeat cover	54
4.4	Biconnected Component - Edge	30	8.4 fibonacci knapsack	54
4.5	Biconnected Component	30		
4.6	Blossom algorithm	31	9 Others	55
4.7	Bridge	31	9.1 .vimrc	55
4.8	Chu-Liu:Edmonds' Algorithm	32	9.2 bigint	55
4.9	Covering problems	32	9.3 Binary Search	56
4.10	Difference constraints	32	9.4 java	57
4.11	Dinitz's algorithm	32	9.5 others	57
4.12	Flow network	33		
4.13	Hamiltonian circuit	34		
4.14	Hopcroft-Karp algorithm	34		

1 Data Structure

1.1 atlantis

```
1 #include<cstdio>
2 #include<algorithm>
3 #include<map>
4
5 #define MAXX 111
6 #define inf 333
7 #define MAX inf*5
8
9 int mid[MAX],cnt[MAX];
10 double len[MAX];
11
12 int n,i,cas;
13 double x1,x2,y1,y2;
14 double ans;
15 std::map<double,int>map;
16 std::map<double,int>::iterator it;
17 double rmap[inf];
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(ll==l && rr==r)
32     {
33         cnt[id]+=val;
34         if(cnt[id])
35             len[id]=rmap[r]-rmap[l-1];
36         else
37             if(l!=r)
38                 len[id]=len[id<<1]+len[id<<1|1];
39             else
40                 len[id]=0;
41         return;
42     }
43     if(mid[id]>=r)
44         update(id<<1,ll,mid[id],l,r,val);
45     else
46         if(mid[id]<l)
47             update(id<<1|1,mid[id]+1,rr,l,r,val);
48         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[i]*(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 inline int update(int id,int pos)
27 {
28     int re(++cnt);
29     l=1;
30     r=m;
31     int nid(re);
32     sz[nid]=sz[id]+1;
33     while(l<r)
34     {
35         mid=(l+r)>>1;
36         if(pos<=mid)
37         {
38             lson[nid]=++cnt;
39             rson[nid]=rson[id];
40             nid=lson[nid];
41             id=lson[id];
42             r=mid;
43         }
44         else
45         {
46             lson[nid]=lson[id];
47             rson[nid]=++cnt;
48             nid=rson[nid];
49             id=rson[id];
50             l=mid+1;
51         }
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     {
64         if(to[i]!=fa)
65         {
66             j=1;
67             for(pre[to[i]][0]=now;j<N;++j)
68                 pre[to[i]][j]=pre[pre[to[i]][j-1]][j-1];
69             rr(to[i],now);
70         }
71     }
72
73 inline int query(int a,int b,int n,int k)
74 {
75     static int tmp,t;
76     l=1;
77     r=m;
78     a=head[a];
79     b=head[b];
80     t=num[n];
81     n=head[n];
82     while(l<r)
83     {
84         mid=(l+r)>>1;
85         tmp=sz[lson[a]]+sz[lson[b]]-2*sz[lson[n]]+(l<=t && t<=mid);
86         if(tmp>=k)
87         {
88             a=lson[a];
89             b=lson[b];
90             n=lson[n];
91             r=mid;
92         }
93         else
94         {
95             k-=tmp;
96             a=rson[a];
97             b=rson[b];
98             n=rson[n];
99             l=mid+1;
100         }
101     }
102     return l;
103 }
104
105 inline int lca(int a,int b)
106 {
107     static int i,j;
108     j=0;
109     if(dg[a]<dg[b])
110         std::swap(a,b);
111     for(i=dg[a]-dg[b];i>=1;++i)
112         a=pre[a][i];
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",&a[i].l,&a[i].r);
38         if(a[i].l>a[i].r)
39             std::swap(a[i].l,a[i].r);
40         sz[i]=a[i].r-a[i].l+1;
41         a[i].w=a[i].l/len+1;
42         a[i].s=i;
43     }
44     std::sort(a+1,a+m+1);
45     i=1;
46     while(i<=m)
47     {
48         now=a[i].w;
49         memset(col,0,sizeof col);
50         for(j=a[i].l;j<=a[i].r;++j)
51             ans[a[i].s]+=2*(col[c[j]]++);
52         for(++i;a[i].w==now;++i)
53         {
54             ans[a[i].s]=ans[a[i-1].s];
55             for(j=a[i-1].r+1;j<=a[i].r;++j)
56                 ans[a[i].s]+=2*(col[c[j]]++);

```

```

57     if(a[i-1].l<a[i].l)
58         for(j=a[i-1].l;j<a[i].l;++j)
59             ans[a[i].s]-=2*(--col[c[j]]);
60     else
61         for(j=a[i].l;j<a[i-1].l;++j)
62             ans[a[i].s]+=2*(col[c[j]]++);
63     }
64 }
65 for(i=1;i<=m;++i)
66 {
67     if(sz[i]==1)
68         all=1ll;
69     else
70         all=sz[i]*(sz[i]-1);
71     num=gcd(ans[i],all);
72     printf("%lld/%lld\n",ans[i]/num,all/num);
73 }
74 return 0;
75 }

```

1.5 Leftist tree

```

1 #include<cstdio>
2 #include<algorithm>
3
4 #define MAXX 100111
5
6 int val[MAXX],l[MAXX],r[MAXX],d[MAXX];
7
8 int set[MAXX];
9
10 int merge(int a,int b)
11 {
12     if(!a)
13         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);
65                 set[j]=merge(j,k)=0;
66
67                 set[k=merge(i,j)]=0;
68                 printf("%d\n",val[k]);
69             }
70         }
71     }
72     return 0;
73 }

```

1.6 Network

```

1 //HLD.....备忘....._( :3JZ)_
2 #include<cstdio>
3 #include<algorithm>
4 #include<cstdlib>
5
6 #define MAXX 80111
7 #define MAXE (MAXX<<1)
8 #define N 18
9
10 int edge[MAXX],nxt[MAXE],to[MAXE],cnt;
11 int fa[MAXX][N],dg[MAXX];
12
13 inline int lca(int a,int b)
14 {
15     static int i,j;
16     j=0;
17     if(dg[a]<dg[b])
18         std::swap(a,b);
19     for(i=dg[a]-dg[b];i>=1,++j)
20         if(i&1)
21             a=fa[a][j];
22     if(a==b)
23         return a;
24     for(i=N-1;i>=0;--i)
25         if(fa[a][i]!=fa[b][i])
26         {
27             a=fa[a][i];
28             b=fa[b][i];
29         }
30     return fa[a][0];
31 }
32
33 inline void add(int a,int b)
34 {
35     nxt[++cnt]=edge[a];
36     edge[a]=cnt;
37     to[cnt]=b;
38 }
39
40 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 }

```

```

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 int head[MAXX],root[MAXX],len[MAXX],pos[MAXX];
150 #define MAX (MAXX*6)
151 #define mid (l+r>>1)
152 #define lc lson[id],l,mid
153 #define rc rson[id],mid+1,r
154 int lson[MAX],rson[MAX];
155 int treap[MAX];
156 void make(int &id,int l,int r,int *the)
157 {
158     id++;cnt++;
159     static int k;
160     for(k=l;k<=r;++k)
161         Treap::insert(treap[id],the[k]);
162     if(l==r)
163     {
164         make(lc,the);
165         make(rc,the);
166     }
167 }
168 int query(int id,int l,int r,int a,int b,int q)
169 {
170     if(a<=l && r<=b)
171         return Treap::rank(treap[id],q);
172     int re=0;
173     if(a<=mid)
174         re=query(lc,a,b,q);
175     if(b>mid)
176         re+=query(rc,a,b,q);
177     return re;
178 }
179 inline int query(int a,int b,int v)
180 {
181     static int re;
182     for(re=0;root[a]!=root[b];a=fa[root[a]][0])
183         re+=query(head[root[a]],1,len[root[a]],1,pos[a],v);
184 }
185 re+=query(head[root[a]],1,len[root[a]],pos[b],pos[a],v);
186 return re;
187 }
188 inline void update(int id,int l,int r,int pos,int val,int n)
189 {
190     while(l<=r)
191     {
192         Treap::del(treap[id],val);
193         Treap::insert(treap[id],n);
194         if(l==r)
195             return;
196         if(pos<=mid)
197         {
198             id=lson[id];
199             r=mid;
200         }
201         else
202         {
203             id=rson[id];
204             l=mid+1;
205         }
206     }
207 }
208 int n,q,i,j,k;
209 int val[MAXX];
210 int main()
211 {
212     srand(1e9+7);
213     scanf("%d",&n,&q);
214     for(i=1;i<=n;++i)
215         scanf("%d",val+i);
216     for(k=1;k<=n;++k)
217     {
218         scanf("%d",&i,&j);
219         add(i,j);
220         add(j,i);
221     }
222     rr(rand()%n+1);
223     for(j=1;j<=n;++j)
224         for(i=1;i<=n;++i)
225             fa[i][j]=fa[fa[i][j-1]][j-1];
226     Treap::init();
227     cnt=0;
228     for(i=1;i<=n;++i)
229         if(!pre[i])
230         {
231             static int tmp[MAXX];
232             for(k=1,j=i;j=jnext[j],++k)
233             {
234                 pos[j]=k;
235                 root[j]=i;
236                 tmp[k]=val[j];
237             }
238             k--;
239             len[i]=k;
240             make(head[i],1,k,tmp);
241         }
242     while(q--)
243     {
244         scanf("%d",&k);
245         if(k)
246         {
247             static int a,b,c,d,l,r,ans,m;
248             scanf("%d",&a,&b);
249             c=lca(a,b);
250             if(dg[a]+dg[b]-2*dg[c]+1<k)
251             {
252                 puts("invalid request!");
253                 continue;
254             }
255             k=dg[a]+dg[b]-2*dg[c]+1-k+1;
256             if(dg[a]<dg[b])
257                 std::swap(a,b);
258             l=-1e9;
259             r=1e9;
260             if(b!=c)
261             {
262                 d=a;
263                 for(i=0,j=dg[a]-dg[c]-1;j;j>=1,++i)
264                     if(j&1)
265                         d=fa[d][i];
266                 while(l<=r)
267                 {
268                     m=l+r>>1;
269                     if(query(a,d,m)+query(b,c,m)>=k)
270                     {
271                         ans=m;
272                         r=m-1;
273                     }
274                     else
275                         l=m+1;
276                 }
277             }
278         }
279     }

```

```

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;
296             }
297             else
298                 l=m+1;
299         }
300     }
301     printf("%d\n",ans);
302 }
303 else
304 {
305     scanf("%d%d",&i,&j);
306     update(head[root[i]],1,len[root[i]],pos[i],val[i],j);
307     val[i]=j;
308 }
309 }
310 return 0;
311 }

```

1.7 OTOCI

```

1 //记得随手 down 啊……亲……
2 //debug 时记得优先检查 up/down/select
3 #include<cstdio>
4 #include<algorithm>
5
6 #define MAXX 30111
7
8 int nxt[MAXX][2],fa[MAXX],pre[MAXX],val[MAXX],sum[MAXX];
9 bool rev[MAXX];
10
11 inline void up(int id)
12 {
13     static int i;
14     sum[id]=val[id];
15     for(i=0;i<2;++i)
16         if(nxt[id][i])
17             sum[id]+=sum[nxt[id][i]];
18 }
19
20 inline void rot(int id,int tp)
21 {
22     static int k;
23     k=pre[id];
24     nxt[k][tp^1]=nxt[id][tp];
25     if(nxt[id][tp])
26         pre[nxt[id][tp]]=k;
27     if(pre[k])
28         nxt[pre[k]][k==nxt[pre[k]][1]]=id;
29     pre[id]=pre[k];
30     nxt[id][tp]=k;
31     pre[k]=id;
32     up(k);
33     up(id);
34 }
35
36 inline void down(int id) //记得随手 down 啊……亲……
37 {
38     static int i;
39     if(rev[id])
40     {
41         rev[id]=false;
42         std::swap(nxt[id][0],nxt[id][1]);
43         for(i=0;i<2;++i)
44             if(nxt[id][i])
45                 rev[nxt[id][i]]^=true;
46     }
47 }
48
49 int freshen(int id)
50 {
51     int re(id);
52     if(pre[id])
53         re=freshen(pre[id]);
54     down(id);
55     return re;
56 }
57
58 inline void splay(int id)//记得随手 down 啊……亲……
59 {
60     static int rt;
61     if(id!=(rt=freshen(id)))
62         for(std::swap(fa[id],fa[rt]);pre[id];rot(id,id==nxt[pre[id]][0]));
63     /* another faster methond:
64     if(id!=rt)

```

```

65     {
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
87 inline void access(int id)
88 {
89     static int to;
90     for(to=0;id;id=fa[id])
91     {
92         splay(id);
93         if(nxt[id][1])
94         {
95             pre[nxt[id][1]]=0;
96             fa[nxt[id][1]]=id;
97         }
98         nxt[id][1]=to;
99         if(to)
100         {
101             pre[to]=id;
102             fa[to]=0;
103         }
104         up(to=id);
105     }
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         rev[id]^=true;
126 }
127
128 int n,i,j,k,q;
129 char buf[11];
130
131 int main()
132 {
133     scanf("%d",&n);
134     for(i=1;i<=n;++i)
135         scanf("%d",&val[i]);
136     scanf("%d",&q);
137     while(q--)
138     {
139         scanf("%s%d%d",buf,&i,&j);
140         switch(buf[0])
141         {
142             case 'b':
143                 if(getrt(i)==getrt(j))
144                     puts("no");
145                 else
146                 {
147                     puts("yes");
148                     makert(i);
149                     fa[i]=j;
150                 }
151                 break;
152             case 'p':
153                 access(i);
154                 splay(i);
155                 val[i]=j;
156                 up(i);
157                 break;
158             case 'e':
159                 if(getrt(i)!=getrt(j))
160                     puts("impossible");

```

```

161         else
162         {
163             makert(i);
164             access(j);
165             splay(j);
166             printf("%d\n",sum[j]);
167         }
168         break;
169     }
170 }
171 return 0;
172 }

```

1.8 picture

```

1 #include<cstdio>
2 #include<algorithm>
3 #include<map>
4
5 #define MAXX 5555
6 #define MAX MAXX<<3
7 #define inf 10011
8
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.
86         val<a.val? val>a.val?
87     }
88     inline void print()
89     {
90         printf("%d_%d_%d_%d\n",l,r,h,val);
91     }
92 }ln[inf];
93
94 int main()
95 {
96     make(1,1,inf);
97     scanf("%d",&n);
98     n<<=1;
99     map.clear();
100     for(i=0;i<n;++i)
101     {
102         scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
103         ln[i].l=x1;
104         ln[i].r=x2;
105         ln[i].h=y1;
106         ln[i].val=1;
107         ln[++i].l=x1;
108         ln[i].r=x2;
109         ln[i].h=y2;
110         ln[i].val=-1;
111         map[x1]=1;
112         map[x2]=1;
113     }
114     i=1;
115     for(it=map.begin();it!=map.end();++it,++i)
116     {
117         it->second=i;
118         rmap[i]=it->first;
119     }
120     i=0;
121     std::sort(ln,ln+n);
122     update(1,1,inf,map[ln[0].l]+1,map[ln[0].r],ln[0].val);
123     sum=len[1];
124     last=len[1];
125     for(i=1;i<n;++i)
126     {
127         sum+=2*seg[1]*(ln[i].h-ln[i-1].h);
128         update(1,1,inf,map[ln[i].l]+1,map[ln[i].r],ln[i].val);
129         sum+=abs(len[1]-last);
130         last=len[1];
131     }
132     printf("%lld\n",sum);
133     return 0;
134 }

```

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     };

```



```

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[
129             [pos] && !r[pos]))
130         {
131             Tp ret(val[pos]);
132             if(!l[pos] || !r[pos])
133                 pos=l[pos]+r[pos];
134             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
39                             i][j-1][ii][jj],table[i][j-1]
40                             ii+(1<<(j-1))][jj]);
41                         else
42                             table[i][j][ii][jj]=std::min(table
43                             i-1][j][ii][jj],table[i-1][j]
44                             ii+j+(1<<(i-1))));
45             }
46         }
47         long long N;
48         std::cin >> N;
49         int r1, c1, r2, c2;
50         for (int i = 0; i < N; ++i)
51         {
52             scanf("%d%d%d%d",&r1,&c1,&r2,&c2);
53             --r1;
54             --c1;
55             --r2;
56             --c2;
57             int w=lg[c2-c1+1];
58             int h=lg[r2-r1+1];
59             printf("%d\n",std::min(table[w][h][r1][c1],std::min
60             (table[w][h][r1][c2-(1<<w)+1],std::min(table[w]
61             ][h][r2-(1<<h)+1][c1],table[w][h][r2-(1<<h)
62             +1][c2-(1<<w)+1]))));
63         }
64     }
65     return 0;
66 }

```

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%hd",&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],
21                 max[i+(1<<(k-1))][j][k-1]),std::max(max[i
22                 ][j+(1<<(k-1))][k-1],max[i+(1<<(k-1))][j
23                 +(1<<(k-1))][k-1]));
24     }
25     printf("Case_%hd:\n",t);
26     while(q--)
27     {
28         scanf("%hd%hd%hd",&i,&j,&l);
29         --i;
30         --j;
31         k=lg[l];
32         printf("%d\n",std::max(std::max(max[i][j][k],max[i][j+l
33         -(1<<k)][k]),std::max(max[i+l-(1<<k)][j][k],max[i+
34         l-(1<<k)][j+l-(1<<k)][k])));
35     }
36 }

```

1.12 Sparse Table

```

1 int num[MAXX],min[MAXX][20];
2 int lg[MAXX];
3
4 int main()
5 {
6     for(i=2;i<MAXX;++i)
7         lg[i]=lg[i>>1]+1;
8     scanf("%d%d",&n,&q);
9     for(i=1;i<=n;++i)
10     {
11         scanf("%d",num+i);
12         min[i][0]=num[i];
13     }
14     for(j=1;j<=lg[n];++j)
15     {
16         l=n+1-(1<<j);
17         j_-=j-1;
18         j_+=(1<<j_);
19         for(i=1;i<=l;++i)
20             min[i][j]=std::min(min[i][j_],min[i+j_][j_]);
21     }
22     printf("Case_%hd:\n",t);
23     while(q--)
24     {
25         scanf("%d%d",&i,&j);
26         k=lg[j-i+1];
27         printf("%d\n",std::min(min[i][k],min[j-(1<<k)+1][k]));
28     }
29 }

```

1.13 Treap

```

1 #include<cstdlib>
2 #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
26 //绕单位向量 V 旋转 theta 角度
27 pv Trans(pv pa,pv V,double theta)
28 {
29     double s = sin(theta);
30     double c = cos(theta);
31     double x,y,z;
32     x = V.x;
33     y = V.y;
34     z = V.z;
35     pv pp =
36         pv(
37             (x*x*(1-c)+c)*pa.x+(x*y*(1-c)-z*s)*pa.y+(x*z
38                 *(1-c)+y*s)*pa.z,
39             (y*x*(1-c)+z*s)*pa.x+(y*y*(1-c)+c)*pa.y+(y*z
40                 *(1-c)-x*s)*pa.z,
41             (x*z*(1-c)-y*s)*pa.x+(y*z*(1-c)+x*s)*pa.y+(z*z
42                 *(1-c)+c)*pa.z
43         );
44     return pp;
45 }
46
47 //经纬度转换
48 x=r*sinθ()*cosθ();
49 y=r*sinθ()*sinθ();
50 z=r*cosθ();
51
52 r=sqrt(x*2+y*2+z*2);///??
53 r=sqrt(x^2+y^2+z^2);///??θ
54
55 =atan(y/x);θ
56 =acos(z/r);θ
57
58 r∞[0,]∞∞π
59 [0,2]∞∞π
60 [0,]∞
61
62 lat1π[-/2,/2]∞
63 lng1π[-,]
64
65 pv getpv(double lat,double lng,double r)
66 {
67     lat += pi/2;
68     lng += pi;
69     return
70     pv(r*sin(lat)*cos(lng),r*sin(lat)*sin(lng),r*cos(lat));
71 }
72
73 //经纬度球面距离
74
75 #include<cstdio>
76 #include<cmath>
77
78 #define MAXX 1111
79
80 char buf[MAXX];
81 const double r=6875.0/2,pi=acos(-1.0);
82 double a,b,c,x1,x2,y2,ans;
83
84 int main()
85 {
86     double y1;
87     while(gets(buf)!=NULL)
88     {
89         gets(buf);
90     }
91 }

```

```

88     gets(buf);
89
90     scanf("%lf^%lf'%lf'\\"_s\\n",&a,&b,&c,buf);
91     x1=a+b/60+c/3600;
92     x1=x1*pi/180;
93     if(buf[0]=='S')
94         x1=-x1;
95
96     scanf("%s",buf);
97     scanf("%lf^%lf'%lf'\\"_s\\n",&a,&b,&c,buf);
98     y1=a+b/60+c/3600;
99     y1=y1*pi/180;
100    if(buf[0]=='W')
101        y1=-y1;
102
103    gets(buf);
104
105    scanf("%lf^%lf'%lf'\\"_s\\n",&a,&b,&c,buf);
106    x2=a+b/60+c/3600;
107    x2=x2*pi/180;
108    if(buf[0]=='S')
109        x2=-x2;
110
111    scanf("%s",buf);
112    scanf("%lf^%lf'%lf'\\"_s\\n",&a,&b,&c,buf);
113    y2=a+b/60+c/3600;
114    y2=y2*pi/180;
115    if(buf[0]=='W')
116        y2=-y2;
117
118    ans=acos(cos(x1)*cos(x2)*cos(y1-y2)+sin(x1)*sin(x2))*r;
119    printf("The distance to the iceberg: %.2lf miles.\\n",
120        ans);
121    if(ans+0.005<100)
122        puts("DANGER!");
123
124    gets(buf);
125    return 0;
126 }
127
128 inline bool ZERO(const double &a)
129 {
130     return fabs(a)<eps;
131 }
132
133 //三维向量是否为零
134 inline bool ZERO(pv p)
135 {
136     return (ZERO(p.x) && ZERO(p.y) && ZERO(p.z));
137 }
138
139 //直线相交
140 bool LineIntersect(Line3D L1, Line3D L2)
141 {
142     pv s = L1.s-L1.e;
143     pv e = L2.s-L2.e;
144     pv p = s*e;
145     if (ZERO(p))
146         return false; //是否平行
147     p = (L2.s-L1.e)*(L1.s-L1.e);
148     return ZERO(p&L2.e); //是否共面
149 }
150
151 //线段相交
152 bool inter(pv a,pv b,pv c,pv d)
153 {
154     pv ret = (a-b)*(c-d);
155     pv t1 = (b-a)*(c-a);
156     pv t2 = (b-a)*(d-a);
157     pv t3 = (d-c)*(a-c);
158     pv t4 = (d-c)*(b-c);
159     return sgn(t1&ret)*sgn(t2&ret) < 0 && sgn(t3&ret)*sgn(t4&ret) < 0;
160 }
161
162 //点在直线上
163 bool OnLine(pv p, Line3D L)
164 {
165     return ZERO((p-L.s)*(L.e-L.s));
166 }
167
168 //点在线段上
169 bool OnSeg(pv p, Line3D L)
170 {
171     return (ZERO((L.s-p)*(L.e-p)) && EQ(Norm(p-L.s)+Norm(p-L.e),Norm(L.e-L.s)));
172 }
173
174 //点到直线距离
175 double Distance(pv p, Line3D L)
176 {
177     return (Norm((p-L.s)*(L.e-L.s))/Norm(L.e-L.s));
178 }
179

```

```

180 //线段夹角
181 //范围值为 π 之间的弧度[0,]
182 double Inclination(Line3D L1, Line3D L2)
183 {
184     pv u = L1.e - L1.s;
185     pv v = L2.e - L2.s;
186     return acos( (u & v) / (Norm(u)*Norm(v)) );
187 }

```

2.2 3DCH

```

1 #include<cstdio>
2 #include<cmath>
3 #include<vector>
4 #include<algorithm>
5
6 #define MAXX 1111
7 #define eps 1e-8
8 #define inf 1e20
9
10 struct pv
11 {
12     double x,y,z;
13     pv(){}
14     pv(const double &xx,const double &yy,const double &zz):x(xx),y(yy),z(zz){}
15     inline pv operator-(const pv &i)const
16     {
17         return pv(x-i.x,y-i.y,z-i.z);
18     }
19     inline pv operator*(const pv &i)const //叉积
20     {
21         return pv(y*i.z-z*i.y,z*i.x-x*i.z,x*i.y-y*i.x);
22     }
23     inline double operator^(const pv &i)const //点积
24     {
25         return x*i.x+y*i.y+z*i.z;
26     }
27     inline double len()
28     {
29         return sqrt(x*x+y*y+z*z);
30     }
31 };
32
33 struct pla
34 {
35     short a,b,c;
36     bool ok;
37     pla(){}
38     pla(const short &aa,const short &bb,const short &cc):a(aa),b(bb),c(cc),ok(true){}
39     inline void set();
40     inline void print()
41     {
42         printf("%hd\\%hd\\%hd\\n",a,b,c);
43     }
44 };
45
46 pv pnt[MAXX];
47 std::vector<pla> fac;
48 short to[MAXX][MAXX];
49
50 inline void pla::set()
51 {
52     to[a][b]=to[b][c]=to[c][a]=fac.size();
53 }
54
55 inline double ptof(const pv &p,const pla &f) //点面距离?
56 {
57     return (pnt[f.b]-pnt[f.a])*(pnt[f.c]-pnt[f.a])^(p-pnt[f.a])
58         ;
59 }
60 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
61 {
62     return (b-a)*(c-a)^(d-a);
63 }
64
65 inline double ptof(const pv &p,const short &f) //点到号面的距离pf
66 {
67     return fabs(vol(pnt[fac[f].a],pnt[fac[f].b],pnt[fac[f].c],p
68         )/((pnt[fac[f].b]-pnt[fac[f].a])*(pnt[fac[f].c]-pnt[
69         fac[f].a])).len());
70 }
71
72 void dfs(const short&,const short&);
73
74 void deal(const short &p,const short &a,const short &b)
75 {
76     if(fac[to[a][b]].ok)
77         if(ptof(pnt[p],fac[to[a][b]])>eps)
78             dfs(p,to[a][b]);
79     else

```

```

78     {
79         pla add(b,a,p);
80         add.set();
81         fac.push_back(add);
82     }
83 }
84
85 void dfs(const short &p,const short &now)
86 {
87     fac[now].ok=false;
88     deal(p,fac[now].b,fac[now].a);
89     deal(p,fac[now].c,fac[now].b);
90     deal(p,fac[now].a,fac[now].c);
91 }
92
93 inline void make()
94 {
95     fac.resize(0);
96     if(n<4)
97         return;
98
99     for(i=1;i<n;++i)
100         if((pnt[0]-pnt[i]).len()>eps)
101         {
102             std::swap(pnt[i],pnt[1]);
103             break;
104         }
105     if(i==n)
106         return;
107
108     for(i=2;i<n;++i)
109         if(((pnt[0]-pnt[1])*(pnt[1]-pnt[i])).len()>eps)
110         {
111             std::swap(pnt[i],pnt[2]);
112             break;
113         }
114     if(i==n)
115         return;
116
117     for(i=3;i<n;++i)
118         if(fabs((pnt[0]-pnt[1])*(pnt[1]-pnt[2])^(pnt[2]-pnt[i]))>eps)
119         {
120             std::swap(pnt[3],pnt[i]);
121             break;
122         }
123     if(i==n)
124         return;
125
126     for(i=0;i<4;++i)
127     {
128         pla add((i+1)%4,(i+2)%4,(i+3)%4);
129         if(ptof(pnt[i],add)>0)
130             std::swap(add.c,add.b);
131         add.set();
132         fac.push_back(add);
133     }
134     for(;i<n;++i)
135         for(j=0;j<fac.size();++j)
136             if(fac[j].ok && ptof(pnt[i],fac[j])>eps)
137             {
138                 dfs(i,j);
139                 break;
140             }
141
142     short tmp(fac.size());
143     fac.resize(0);
144     for(i=0;i<tmp;++i)
145         if(fac[i].ok)
146             fac.push_back(fac[i]);
147 }
148
149 inline pv gc() //重心
150 {
151     pv re(0,0,0),o(0,0,0);
152     double all(0),v;
153     for(i=0;i<fac.size();++i)
154     {
155         v=vol(o,pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
156         re+=(pnt[fac[i].a]+pnt[fac[i].b]+pnt[fac[i].c])*0.25*v;
157         all+=v;
158     }
159     return re*(1/all);
160 }
161
162 inline bool same(const short &s,const short &t) //两面是否相等
163 {
164     pv &a=pnt[fac[s].a],&b=pnt[fac[s].b],&c=pnt[fac[s].c];
165     return fabs(vol(a,b,c,pnt[fac[t].a]))<eps && fabs(vol(a,b,c,
166         pnt[fac[t].b]))<eps && fabs(vol(a,b,c,pnt[fac[t].c]))<eps;
167 }
168 //表面多边形数目
169 inline short facetcnt()
170 {
171     short ans=0;
172     for(short i=0;i<fac.size();++i)
173     {
174         for(j=0;j<i;++j)
175             if(same(i,j))
176                 break;
177         if(j==i)
178             ++ans;
179     }
180     return ans;
181 }
182
183 //表面三角形数目
184 inline short trianglecnt()
185 {
186     return fac.size();
187 }
188
189 //三点构成的三角形面积*2
190 inline double area(const pv &a,const pv &b,const pv &c)
191 {
192     return (b-a)*(c-a).len();
193 }
194
195 //表面积
196 inline double area()
197 {
198     double ret(0);
199     for(i=0;i<fac.size();++i)
200         ret+=area(pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
201     return ret/2;
202 }
203
204 //体积
205 inline double volume()
206 {
207     pv o(0,0,0);
208     double ret(0);
209     for(short i(0);i<fac.size();++i)
210         ret+=vol(o,pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
211     return fabs(ret/6);
212 }
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

50 double tim;
51 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
144
145
146
147
148
149
150
151
152
153
154
155 }
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
```

```

76     for(j=0;j<alpha.size();++j)
77     {
78         if(alpha[j].flag)
79             ++sum;
80         else
81             --sum;
82         ans=std::max(ans,sum);
83     }
84 }
85 printf("%hd\n",ans+1);
86 }
87 return 0;
88 }
89
90 //最小覆蓋圓
91
92 #include<cstdio>
93 #include<cmath>
94
95 #define MAXX 511
96 #define eps 1e-8
97
98 struct pv
99 {
100     double x,y;
101     pv(){}
102     pv(const double &xx,const double &yy):x(xx),y(yy){}
103     inline pv operator-(const pv &i) const
104     {
105         return pv(x-i.x,y-i.y);
106     }
107     inline pv operator+(const pv &i) const
108     {
109         return pv(x+i.x,y+i.y);
110     }
111     inline double cross(const pv &i) const
112     {
113         return x*i.y-y*i.x;
114     }
115     inline double len()
116     {
117         return sqrt(x*x+y*y);
118     }
119     inline pv operator/(const double &a) const
120     {
121         return pv(x/a,y/a);
122     }
123     inline pv operator*(const double &a) const
124     {
125         return pv(x*a,y*a);
126     }
127 }pnt[MAXX],o,tl,lt,aa,bb,cc,dd;
128
129 short n,i,j,k,l;
130 double r,u;
131
132 inline pv ins(const pv &a1,const pv &a2,const pv &b1,const pv &b2)
133 {
134     tl=a2-a1;
135     lt=b2-b1;
136     u=(b1-a1).cross(lt)/(tl.cross(lt));
137     return a1+tl*u;
138 }
139
140 inline pv get(const pv &a,const pv &b,const pv &c)
141 {
142     aa=(a+b)/2;
143     bb.x=aa.x-a.y+b.y;
144     bb.y=aa.y+a.x-b.x;
145     cc=(a+c)/2;
146     dd.x=cc.x-a.y+c.y;
147     dd.y=cc.y+a.x-c.x;
148     return ins(aa,bb,cc,dd);
149 }
150
151 int main()
152 {
153     while(scanf("%hd",&n),n)
154     {
155         for(i=0;i<n;++i)
156             scanf("%lf%lf",&pnt[i].x,&pnt[i].y);
157         o=pnt[0];
158         r=0;
159         for(i=1;i<n;++i)
160             if((pnt[i]-o).len()>r+eps)
161             {
162                 o=pnt[i];
163                 r=0;
164                 for(j=0;j<i;++j)
165                     if((pnt[j]-o).len()>r+eps)
166                     {
167                         o=(pnt[i]+pnt[j])/2;
168                         r=(o-pnt[j]).len();
169                         for(k=0;k<j;++k)
170                             if((o-pnt[k]).len()>r+eps)
171                             {
172                                 o=get(pnt[i],pnt[j],pnt[k]);
173                                 r=(o-pnt[i]).len();
174                             }
175                         }
176             }
177         printf("%.2lf_%.2lf_%.2lf\n",o.x,o.y,r);
178     }
179     return 0;
180 }
181
182 //兩原面積交
183 double dis(int x,int y)
184 {
185     return sqrt((double)(x*x+y*y));
186 }
187
188 double area(int x1,int y1,int x2,int y2,double r1,double r2)
189 {
190     double s=dis(x2-x1,y2-y1);
191     if(r1+r2<s) return 0;
192     else if(r2-r1>s) return PI*r1*r1;
193     else if(r1-r2>s) return PI*r2*r2;
194     double q1=acos((r1*r1+s*s-r2*r2)/(2*r1*s));
195     double q2=acos((r2*r2+s*s-r1*r1)/(2*r2*s));
196     return (r1*r1*q1+r2*r2*q2-r1*s*sin(q1));
197 }
198
199 //三角形外接圓
200 {
201     for (int i = 0; i < 3; i++)
202         scanf("%lf%lf",&p[i].x,&p[i].y);
203     tp = pv((p[0].x+p[1].x)/2,(p[0].y+p[1].y)/2);
204     l[0] = Line(tp,pv(tp.x-(p[1].y-p[0].y),tp.y+(p[1].x-p[0].x)));
205     tp = pv((p[0].x+p[2].x)/2,(p[0].y+p[2].y)/2);
206     l[1] = Line(tp,pv(tp.x-(p[2].y-p[0].y),tp.y+(p[2].x-p[0].x)));
207     tp = LineToLine(l[0],l[1]);
208     r = pv(tp,p[0]).Length();
209     printf("%.6f,%.6f,%.6f\n",tp.x,tp.y,r);
210 }
211
212 //三角形內切圓
213 {
214     for (int i = 0; i < 3; i++)
215         scanf("%lf%lf",&p[i].x,&p[i].y);
216     if (xmult(pv(p[0],p[1]),pv(p[0],p[2])) < 0)
217         swap(p[1],p[2]);
218     for (int i = 0; i < 3; i++)
219         len[i] = pv(p[i],p[(i+1)%3]).Length();
220     tr = (len[0]+len[1]+len[2])/2;
221     r = sqrt((tr-len[0])*(tr-len[1])*(tr-len[2])/tr);
222     for (int i = 0; i < 2; i++)
223     {
224         v = pv(p[i],p[i+1]);
225         tv = pv(-v.y,v.x);
226         tr = tv.Length();
227         tv = pv(tv.x*r/tr,tv.y*r/tr);
228         tp = pv(p[i].x+tv.x,p[i].y+tv.y);
229         l[i].s = tp;
230         tp = pv(p[i+1].x+tv.x,p[i+1].y+tv.y);
231         l[i].e = tp;
232     }
233     tp = LineToLine(l[0],l[1]);
234     printf("%.6f,%.6f,%.6f\n",tp.x,tp.y,r);
235 }

```

2.5 closest point pair

```

1 //演算法筆記
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。MergeO(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++] =116
        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 //演算法筆記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
117     if(l+1 ==r) {
118         ret = min(calc_dis(pnts[l],pnts[l+1]) ,ret);
119         return ret;
120     }
121     if(l+2 ==r) {
122         ret = min(calc_dis(pnts[l],pnts[l+1]) ,ret);
123         ret = min(calc_dis(pnts[l],pnts[l+2]) ,ret);
124         ret = min(calc_dis(pnts[l+1],pnts[l+2]) ,ret);
125         return ret;
126     }
127
128     int mid = l+r>>1;
129     ret = min (ret ,Gao(l ,mid,pnts));
130     ret = min (ret , Gao(mid+1, r,pnts));
131
132     for(int c = l ; c<=r; c++)
133         for(int d = c+1; d <=c+7 && d<=r; d++) {
134             ret = min(ret , calc_dis(pnts[c],pnts[d]));
135         }
136     return ret;
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] =
150     {{-1,-1},{-1,0},{-1,1},{0,-1},{0,0},{0,1},{1,-1},{1,0},{1,1}};
151
152 int n,x,y,nx,ny;
153 map<pair<int,int>,vector<Point > > g;
154 vector<Point > tmp;
155 Point p[20000];
156 double tx,ty,ans,nowans;
157 vector<Point >::iterator it,op,ed;
158 pair<int,int> gird;
159 bool flag;
160
161 double Dis(Point p0,Point p1)
162 {
163     return sqrt((p0.first-p1.first)*(p0.first-p1.first)+
164         (p0.second-p1.second)*(p0.second-p1.second));
165 }
166
167 double CalcDis(Point p0,Point p1,Point p2)
168 {
169     return Dis(p0,p1)+Dis(p0,p2)+Dis(p1,p2);
170 }
171
172 void build(int n,double w)
173 {
174     g.clear();
175     for (int i = 0;i < n;i++)
176         g[make_pair((int)floor(p[i].first/w),(int)floor(p[i].second
177             /w))].push_back(p[i]);
178 }
179
180 int main()
181 {
182     int t;
183     scanf("%d",&t);
184     for (int ft = 1;ft <= t;ft++)
185     {
186         scanf("%d",&n);
187         for (int i = 0;i < n;i++)
188         {
189             scanf("%lf%lf",&tx,&ty);
190             p[i] = make_pair(tx,ty);
191         }
192         random_shuffle(p,p+n);
193         ans = CalcDis(p[0],p[1],p[2]);
194         build(3,ans/2.0);
195         for (int i = 3;i < n;i++)
196         {
197             x = (int)floor(2.0*p[i].first/ans);
198             y = (int)floor(2.0*p[i].second/ans);
199             tmp.clear();
200             for (int k = 0;k < 9;k++)
201             {
202                 nx = x+step[k][0];
203                 ny = y+step[k][1];
204                 gird = make_pair(nx,ny);
205                 if (g.find(gird) != g.end())
206                 {
207                     op = g[gird].begin();
208                     ed = g[gird].end();
209                     for (it = op;it != ed;it++)
210                         tmp.push_back(*it);
211                 }
212             }
213         }
214     }
215 }

```



```

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(
225         2.0*p[i].second/ans))] .push_back(p[i]);
226 }
227 printf("%.3f\n", ans);
228 }

```

2.6 ellipse

```

1 |  $\frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2} = 1$ 
2 |
3 |  $x = h + a \times \cos(t)$ 
4 |  $y = k + b \times \sin(t)$ 
5 |
6 |  $area = \pi \times a \times b$ 
7 | distance from center to focus:  $f = \sqrt{a^2 - b^2}$ 
8 | eccentricity:  $e = \sqrt{a - \frac{b^2}{a}} = \frac{f}{a}$ 
9 | focal parameter:  $\frac{b^2}{\sqrt{a^2 - b^2}} = \frac{b^2}{f}$ 
10 |
11 | double circumference(double a, double b) // accuracy: pow
    | (0.5, 53);
12 | {
13 |     double x=a;
14 |     double y=b;
15 |     if(x<y)
16 |         std::swap(x,y);
17 |     double digits=53, tol=sqrt(pow(0.5, digits));
18 |     if(digits<y<tol*x)
19 |         return 4*x;
20 |     double s=0, m=1;
21 |     while(x>(tol+1)*y)
22 |     {
23 |         double tx=x;
24 |         double ty=y;
25 |         x=0.5f*(tx+ty);
26 |         y=sqrt(tx*ty);
27 |         m*=2;
28 |         s+=m*pow(x-y, 2);
29 |     }
30 |     return pi*(pow(a+b, 2)-s)/(x+y);
31 | }

```

2.7 Graham's scan

```

1 | pv pnt[MAXX];
2 |
3 | inline bool com(const pv &a, const pv &b)
4 | {
5 |     if (fabs(t=(a-pnt[0]).cross(b-pnt[0]))>eps)
6 |         return t>0;
7 |     return (a-pnt[0]).len()<(b-pnt[0]).len();
8 | }
9 |
10 | inline void graham(std::vector<pv> &ch, const int n)
11 | {
12 |     std::nth_element(pnt, pnt, pnt+n);
13 |     std::sort(pnt+1, pnt+n, com);
14 |     ch.resize(0);
15 |     ch.push_back(pnt[0]);
16 |     ch.push_back(pnt[1]);
17 |     static int i;
18 |     for(i=2; i<n; ++i)
19 |         if (fabs((pnt[i]-ch[0]).cross(ch[1]-ch[0]))>eps)
20 |         {
21 |             ch.push_back(pnt[i]);
22 |             break;
23 |         }
24 |     else
25 |         ch.back()=pnt[i];
26 |     for(; i<n; ++i)
27 |     {
28 |         while((ch.back()-ch[ch.size()-2]).cross(pnt[i]-ch[ch.
29 |             size()-2])<eps)
30 |             ch.pop_back();
31 |         ch.push_back(pnt[i]);
32 |     }

```

2.8 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 }

```

2.9 intersection of circle and poly

```

1 bool InCircle(Point a, double r)
2 {
3     return cmp(a.x*a.x+a.y*a.y, r*r) <= 0;
4     //这里判断的时候 EPS 一定不要太小!!
5 }
6
7 double CalcArea(Point a, Point b, double r)
8 {
9     Point p[4];
10    int tot = 0;
11    p[tot++] = a;
12
13    Point tv = Point(a, b);
14    Line tmp = Line(Point(0, 0), Point(tv.y, -tv.x));
15    Point near = LineToLine(Line(a, b), tmp);
16    if (cmp(near.x*near.x+near.y*near.y, r*r) <= 0)
17    {
18        double A, B, C;
19        A = near.x*near.x+near.y*near.y;
20        C = r;
21        B = C*C-A;
22        double tvl = tv.x*tv.x+tv.y*tv.y;
23        double tmp = sqrt(B/tvl); //这样做只用一次开根
24        p[tot] = Point(near.x+tmp*tv.x, near.y+tmp*tv.y);
25        if (OnSeg(Line(a, b), p[tot]) == true) tot++;
26        p[tot] = Point(near.x-tmp*tv.x, near.y-tmp*tv.y);
27        if (OnSeg(Line(a, b), p[tot]) == true) tot++;
28    }
29    if (tot == 3)
30    {
31        if (cmp(Point(p[0], p[1]).Length(), Point(p[0], p[2]).Length()) > 0)
32            swap(p[1], p[2]);
33    }
34    p[tot++] = b;
35
36    double res = 0.0, theta, a0, a1, sgn;
37    for (int i = 0; i < tot-1; i++)
38    {
39        if (InCircle(p[i], r) == true && InCircle(p[i+1], r) == true)
40        {
41            res += 0.5*xmult(p[i], p[i+1]);
42        }
43        else

```

```

44    {
45        a0 = atan2(p[i+1].y, p[i+1].x);
46        a1 = atan2(p[i].y, p[i].x);
47        if (a0 < a1) a0 += 2*pi;
48        theta = a0-a1;
49        if (cmp(theta, pi) >= 0) theta = 2*pi-theta;
50        sgn = xmult(p[i], p[i+1])/2.0;
51        if (cmp(sgn, 0) < 0) theta = -theta;
52        res += 0.5*r*r*theta;
53    }
54 }
55 return res;
56 }
57
58 //调用
59
60 area2 = 0.0;
61 for (int i = 0; i < resn; i++) //遍历每条边, 按照逆时针
62     area2 += CalcArea(p[i], p[(i+1)%resn], r);

```

2.10 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]
72             ][i][1]));
73     return sqr(a[0])+sqr(a[1]);
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 }

```

2.11 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 ] ]
62         & 0xffff ] ] = order[ i ];
63     memmove( order, tb, n * sizeof( int ) );
64     memset( ta, 0, sizeof( ta ) );
65     for ( int i = 0; i < n; i++ ) ta[ p[ i ] >> 16 ]++;
66     for ( int i = 0; i < 65535; i++ ) ta[ i + 1 ] += ta[ i ];
67     for ( int i = n - 1; i >= 0; i— ) tb[ —ta[ p[ order[ i ] ]
68         >> 16 ] ] = order[ i ];
69     memmove( order, tb, n * sizeof( int ) );
70 }
71
72 int work( int ii ) //求每个点在一个方向上最近的点
73 {
74     for ( int i = 0; i < n; i++ ) //排序前的准备工作
75     {
76         a[ i ] = y[ i ] - x[ i ] + srange;
77         b[ i ] = srange - y[ i ];
78         order[ i ] = i;
79     }
80     radixsort( b ); //排序
81     radixsort( a );
82     for ( int i = 0; i < n; i++ )
83     {
84         torder[ i ] = order[ i ];
85         order[ i ] = i;
86     }
87     radixsort( a ); //为线段树而做的排序
88     radixsort( b );
89     for ( int i = 0; i < n; i++ )
90     {
91         Index[ order[ i ] ] = i; //取反, 求orderIndex
92     }
93     for ( int i = 1; i < ra + n; i++ ) c[ i ] = 0x7fffffff; //线段树初始化
94     memset( d, 0xff, sizeof( d ) );
95     for ( int i = 0; i < n; i++ ) //线段树插入删除调用
96     {
97         int tt = torder[ i ];
98         road[ tt ][ ii ] = find( Index[ tt ] );
99         insert( Index[ tt ], y[ tt ] + x[ tt ], tt );
100     }
101 }
102
103 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
104 {
105     return abs( x[ a ] - x[ b ] ) + abs( y[ a ] - y[ b ] );
106 }
107
108 int ttb[ 400000 ]; //边排序的临时变量
109 int rx[ 400000 ], ry[ 400000 ], rd[ 400000 ]; //边的存储
110 int rr = 0;
111
112 int radixsort_2( int *p ) //还是基数排序, copy+的产物paste
113 {
114     memset( ta, 0, sizeof( ta ) );
115     for ( int i = 0; i < rr; i++ ) ta[ p[ i ] & 0xffff ]++;
116     for ( int i = 0; i < 65535; i++ ) ta[ i + 1 ] += ta[ i ];
117     for ( int i = rr - 1; i >= 0; i— ) ttb[ —ta[ p[ order[ i ] ]
118         & 0xffff ] ] = order[ i ];
119     memmove( order, ttb, rr * sizeof( int ) );
120     memset( ta, 0, sizeof( ta ) );
121     for ( int i = 0; i < rr; i++ ) ta[ p[ i ] >> 16 ]++;
122     for ( int i = 0; i < 65535; i++ ) ta[ i + 1 ] += ta[ i ];
123     for ( int i = rr - 1; i >= 0; i— ) ttb[ —ta[ p[ order[ i ] ]

```

```

121     ] >> 16 ] ] = order[ i ];
122     memmove( order, ttb, rr * sizeof( int ) );
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         {
186             //为了降低算法复杂度, 只求出一个方向的边4
187             if ( i == 2 )
188             {
189                 for ( int j = 0; j < n; j++ ) swap( x[ j ], y[ j
190                 ] );
189             }
190             if ( ( i & 1 ) == 1 )
191             {
192                 for ( int j = 0; j < n; j++ ) x[ j ] = srange -
193                 x[ j ];
194             }
195             work( i );
196             printf( "Case_%d: %Total_Weight_=", ++casenum );
197             cout << kruskal() << endl;
198         }
199     }
200     return 0;

```

2.12 others

```

1| eps
2|
3| 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。

```

4|
5| 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。
6| 如果 a 为正, 则输出 a + eps, 否则输出 a - eps。
7|
8| 不要输出 -0.000
9|
10| 注意 double 的数据范围
11|
12| a==b fabs(a-b)<eps
13| a!=b fabs(a-b)>eps
14| a<b a+eps<b
15| a<=b a<b+eps
16| a>b a>b+eps
17| a>=b a+eps>b
18|
19| 三角函数
20|
21| cos/sin/tan 输入弧度
22| acos 输入 [-1,+1], 输出 [0,π]
23| asin 输入 [-1,+1], 输出  $[-\frac{\pi}{2}, +\frac{\pi}{2}]$ 
24| atan 输出  $[-\frac{\pi}{2}, +\frac{\pi}{2}]$ 
25| atan2 输入 (y,x)(注意顺序), 返回  $\tan(\frac{y}{x}) \in [-\pi, +\pi]$ 。xy 都是零的时候会发
    生除零错误
26|
27| other
28|
29| log 自然对数(ln)
30| log10 你猜……
31| ceil 向上
32| floor 向下
33|
34| round
35|
36| cpp: 四舍六入五留双
37| java: add 0.5, then floor
38| cpp:
39| (一) 当尾数小于或等于 4 时, 直接将尾数舍去。
40| (二) 当尾数大于或等于 6 时, 将尾数舍去并向前一位进位。
41| (三) 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。
42| (四) 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。
43|
44| rotate mat:
45| 
$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$


```

2.13 Pick's theorem

```

1| 给定顶点座标均是整点（或正方形格点）的简单多边形
2|
3| A: 面积
4| i: 内部格点数目
5| b: 边上格点数目
6|  $A = i + \frac{b}{2} - 1$  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
7|
8|
9|  $A = 2 \times i + b - 2$ 

```

2.14 PointInPoly

```

1| /*射线法
2| , 多边形可以是凸的或凹的的顶点数目要大于等于
3| poly3返回值为:
4|
5| 0 — 点在内poly
6| 1 — 点在边界上poly
7| 2 — 点在外poly
8| */
9|
10| int inPoly(pv p, pv poly[], int n)
11| {
12|     int i, count;
13|     Line ray, side;
14|
15|     count = 0;
16|     ray.s = p;
17|     ray.e.y = p.y;
18|     ray.e.x = -1; //-, 注意取值防止越界! INF
19|
20|     for ( i = 0; i < n; i++ )

```

```

21 {
22     side.s = poly[i];
23     side.e = poly[(i+1)%n];
24
25     if(OnSeg(p, side))
26         return 1;
27
28     // 如果平行轴则不考虑side
29     if (side.s.y == side.e.y)
30         continue;
31
32     if (OnSeg(side.s, ray))
33     {
34         if (side.s.y > side.e.y)
35             count++;
36     }
37     else
38     {
39         if (OnSeg(side.e, ray))
40         {
41             if (side.e.y > side.s.y)
42                 count++;
43         }
44         else if (inter(ray, side))
45             count++;
46     }
47     return ((count % 2 == 1) ? 0 : 2);
48 }

```

2.15 rotating caliper

```

1 //最远点对
2
3 inline double go()
4 {
5     l=ans=0;
6     for(i=0;i<n;++i)
7     {
8         tl=pnt[(i+1)%n]-pnt[i];
9         while(abs(tl.cross(pnt[(l+1)%n]-pnt[l]))>=abs(tl.cross(
10             pnt[l]-pnt[i])))
11             l=(l+1)%n;
12         ans=std::max(ans,std::max(dist(pnt[l],pnt[i]),dist(pnt[
13             l],pnt[(i+1)%n])));
14     }
15     return ans;
16 }
17 //两凸包最近距离
18 double go()
19 {
20     sq=sp=0;
21     for(i=1;i<ch[1].size();++i)
22         if(ch[1][sq]<ch[1][i])
23             sq=i;
24     tp=sp;
25     tq=sq;
26     ans=(ch[0][sp]-ch[1][sq]).len();
27     do
28     {
29         a1=ch[0][sp];
30         a2=ch[0][(sp+1)%ch[0].size()];
31         b1=ch[1][sq];
32         b2=ch[1][(sq+1)%ch[1].size()];
33         tpv=b1-(b2-a1);
34         tpv.x = b1.x - (b2.x - a1.x);
35         tpv.y = b1.y - (b2.y - a1.y);
36         len=(tpv-a1).cross(a2-a1);
37         if(fabs(len)<eps)
38         {
39             ans=std::min(ans,p2l(a1,b1,b2));
40             ans=std::min(ans,p2l(a2,b1,b2));
41             ans=std::min(ans,p2l(b1,a1,a2));
42             ans=std::min(ans,p2l(b2,a1,a2));
43             sp=(sp+1)%ch[0].size();
44             sq=(sq+1)%ch[1].size();
45         }
46         else 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。、构造这些点的逐平切线,
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。
128
129 156 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。
130
131
132
133 //临界切线、计算
134 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
135 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
136 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,
137 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。
138 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，
139| 645 直到新的点为 (yminP,ymaxQ)。、输出
140| 7线。CS
141|
142| //最小最大周长面积外接矩形//、计算全部四个多边形的端点，
143| 1 称之为， xminP , xmaxP , yminP 。ymaxP、通过四个点构造
144| 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，
145| 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。
146| 4、计算新矩形的周长面积，
147| 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，
148| 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。
149| 7

```

2.16 shit

```

1| struct pv
2| {
3|     double x,y;
4|     pv():x(0),y(0){}
5|     pv(double xx,double yy):x(xx),y(yy){}
6|     inline pv operator+(const pv &i) const
7|     {
8|         return pv(x+i.x,y+i.y);
9|     }
10|    inline pv operator-(const pv &i) const
11|    {
12|        return pv(x-i.x,y-i.y);
13|    }
14|    inline bool operator==(const pv &i) const
15|    {
16|        return fabs(x-i.x)<eps && fabs(y-i.y)<eps;
17|    }
18|    inline bool operator<(const pv &i) const
19|    {
20|        return y==i.y?x<i.x:y<i.y;
21|    }
22|    inline double cross(const pv &i) const
23|    {
24|        return x*i.y-y*i.x;
25|    }
26|    inline double dot(const pv &i) const
27|    {
28|        return x*i.x+y*i.y;
29|    }
30|    inline double len()
31|    {
32|        return sqrt(x*x+y*y);
33|    }
34|    inline pv rotate(pv p,double theta)
35|    {
36|        static pv v;
37|        v=*this-p;
38|        static double c,s;
39|        c=cos(theta);
40|        s=sin(theta);
41|        return pv(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
42|    }
43| };
44|
45| inline int dblcmp(double d)
46| {
47|     if(fabs(d)<eps)
48|         return 0;
49|     return d>eps?1:-1;
50| }
51|
52| inline int cross(pv *a,pv *b) // 不相交0 不规范1 规范2
53| {
54|     int d1=dblcmp((a[1]-a[0]).cross(b[0]-a[0]));
55|     int d2=dblcmp((a[1]-a[0]).cross(b[1]-a[0]));
56|     int d3=dblcmp((b[1]-b[0]).cross(a[0]-b[0]));
57|     int d4=dblcmp((b[1]-b[0]).cross(a[1]-b[0]));
58|     if((d1^d2)==-2 && (d3^d4)==-2)
59|         return 2;
60|     return ((d1==0 && dblcmp((b[0]-a[0]).dot(b[0]-a[1]))<=0) ||
61|            (d2==0 && dblcmp((b[1]-a[0]).dot(b[1]-a[1]))<=0) ||
62|            (d3==0 && dblcmp((a[0]-b[0]).dot(a[0]-b[1]))<=0) ||
63|            (d4==0 && dblcmp((a[1]-b[0]).dot(a[1]-b[1]))<=0));
64| }
65|
66| inline bool pntonseg(const pv &p,const pv *a)
67| {
68|     return fabs((p-a[0]).cross(p-a[1]))<eps && (p-a[0]).dot(p-a[1])<eps;
69| }
70|
71| pv rotate(pv v,pv p,double theta,double sc=1) // rotate vector
    v, theta 弧度 [0,2]
72| {
73|     static pv re;
74|     re=p;

```

```

75|     v=v-p;
76|     p.x=sc*cos(theta);
77|     p.y=sc*sin(theta);
78|     re.x+=v.x*p.x-v.y*p.y;
79|     re.y+=v.x*p.y+v.y*p.x;
80|     return re;
81| }
82|
83| struct line
84| {
85|     pv pnt[2];
86|     line(double a,double b,double c) // a*x + b*y + c = 0
87|     {
88|         #define maxl 1e2 //preciseness should not be too high ( compare
            with eps )
89|         if(fabs(b)>eps)
90|         {
91|             pnt[0]=pv(maxl,(c+a*maxl)/(-b));
92|             pnt[1]=pv(-maxl,(c-a*maxl)/(-b));
93|         }
94|         else
95|         {
96|             pnt[0]=pv(-c/a,maxl);
97|             pnt[1]=pv(-c/a,-maxl);
98|         }
99|         #undef maxl
100|     }
101|     pv cross(const line &v) const
102|     {
103|         double a=(v.pnt[1]-v.pnt[0]).cross(pnt[0]-v.pnt[0]);
104|         double b=(v.pnt[1]-v.pnt[0]).cross(pnt[1]-v.pnt[0]);
105|         return pv((pnt[0].x*b-pnt[1].x*a)/(b-a),(pnt[0].y*b-pnt[1].y*a)/(b-a));
106|     }
107| };
108|
109| inline std::pair<pv,double> getcircle(const pv &a,const pv &b,
    const pv &c)
110| {
111|     static pv ct;
112|     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()));
113|     return std::make_pair(ct,sqrt((ct-a).len()));
114| }

```

2.17 sort - polar angle

```

1| inline bool cmp(const Point& a,const Point& b)
2| {
3|     if (a.y*b.y <= 0)
4|     {
5|         if (a.y > 0 || b.y > 0)
6|             return a.y < b.y;
7|         if (a.y == 0 && b.y == 0)
8|             return a.x < b.x;
9|     }
10|    return a.cross(b) > 0;
11| }

```

2.18 triangle

```

1| Area:
2|  $p = \frac{a+b+c}{2}$ 
3|  $area = \sqrt{p \times (p-a) \times (p-b) \times (p-c)}$ 
4|  $area = \frac{a \times b \times \sin(\angle C)}{2}$ 
5|  $area = \frac{a^2 \times \sin(\angle B) \times \sin(\angle C)}{2 \times \sin(\angle B + \angle C)}$ 
6|  $area = \frac{a^2}{2 \times (\cot(\angle B) + \cot(\angle C))}$ 
7|
8| centroid:
9| center of mass
10| intersection of triangle's three triangle medians
11|
12| Trigonometric conditions:
13|  $\tan \frac{\alpha}{2} \tan \frac{\beta}{2} + \tan \frac{\gamma}{2} \tan \frac{\alpha}{2} + \tan \frac{\gamma}{2} \tan \frac{\beta}{2} = 1$ 
14|  $\sin^2 \frac{\alpha}{2} + \sin^2 \frac{\beta}{2} + \sin^2 \frac{\gamma}{2} + 2 \sin \frac{\alpha}{2} \sin \frac{\beta}{2} \sin \frac{\gamma}{2} = 1$ 
15|
16| Circumscribed circle:
17|  $diameter = \frac{abc}{2 \cdot area} = \frac{|AB||BC||CA|}{2|\Delta ABC|}$ 
    =  $\frac{abc}{2\sqrt{s(s-a)(s-b)(s-c)}}$ 
    =  $\frac{2 \cdot area}{\sqrt{(a+b+c)(-a+b+c)(a-b+c)(a+b-c)}}$ 
18|  $diameter = \sqrt{\frac{2 \cdot area}{\sin A \sin B \sin C}}$ 
19|  $diameter = \frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$ 
20|
21| Incircle:
22|  $inradius = \frac{2 \times area}{a+b+c}$ 
23| coordinates(x,y)= $\left(\frac{ax_a+bx_b+cx_c}{a+b+c}, \frac{ay_a+by_b+cy_c}{a+b+c}\right) =$ 
     $\frac{a}{a+b+c}(x_a,y_a) + \frac{b}{a+b+c}(x_b,y_b) + \frac{c}{a+b+c}(x_c,y_c)$ 

```

```

24|
25| Excircles:
26| radius[a]= $\frac{2 \times \text{area}}{b+c-a}$ 
27| radius[b]= $\frac{2 \times \text{area}}{a+c-b}$ 
28| radius[c]= $\frac{2 \times \text{area}}{a+b-c}$ 
29|
30| Steiner circumellipse (least area circumscribed ellipse)
31| area= $\Delta \times \frac{4\pi}{3\sqrt{3}}$ 
32| center is the triangle's centroid.
33|
34| Steiner inellipse ( maximum area inellipse )
35| area= $\Delta \times \frac{\pi}{3\sqrt{3}}$ 
36| center is the triangle's centroid.
37|
38| Fermat Point:
39| 当有一个内角不小于 120° 时, 费马点为此角对应顶点。
40|
41| 当三角形的内角都小于 120° 时
42|
43| 以三角形的每一边为底边, 向外做三个正三角形  $\triangle ABC'$ ,  $\triangle BCA'$ ,  $\triangle CAB'$ 。
44| 连接  $CC'$ 、 $BB'$ 、 $AA'$ , 则三条线段的交点就是所求的点。

3 Geometry/tmp

3.1 circle

1| struct circle
2| {
3|     point p;
4|     double r;
5|     circle(){}
6|     circle(point _p,double _r):
7|     p(_p),r(_r){};
8|     circle(double x,double y,double _r):
9|     p(point(x,y)),r(_r){};
10|    circle(point a,point b,point c)//三角形的外接圆
11|    {
12|        p=line(a.add(b).div(2),a.add(b).div(2).add(b.sub(a).
13|            rotleft()).crosspoint(line(c.add(b).div(2),c.add(b).
14|            div(2).add(b.sub(c).rotleft())));
15|        r=p.distance(a);
16|    }
17|    circle(point a,point b,point c,bool t)//三角形的内切圆
18|    {
19|        line u,v;
20|        double m=atan2(b.y-a.y,b.x-a.x),n=atan2(c.y-a.y,c.x-a.x);
21|        u.a=a;
22|        u.b=u.a.add(point(cos((n+m)/2),sin((n+m)/2)));
23|        v.a=b;
24|        m=atan2(a.y-b.y,a.x-b.x),n=atan2(c.y-b.y,c.x-b.x);
25|        v.b=v.a.add(point(cos((n+m)/2),sin((n+m)/2)));
26|        p=u.crosspoint(v);
27|        r=line(a,b).dispointtoseg(p);
28|    }
29|    void input()
30|    {
31|        p.input();
32|        scanf("%lf",&r);
33|    }
34|    void output()
35|    {
36|        printf("%.2lf,%.2lf,%.2lf\n",p.x,p.y,r);
37|    }
38|    bool operator==(circle v)
39|    {
40|        return ((p==v.p)&&dblcmp(r-v.r)==0);
41|    }
42|    bool operator<(circle v)const
43|    {
44|        return ((p<v.p)|| (p==v.p)&&dblcmp(r-v.r)<0);
45|    }
46|    double area()
47|    {
48|        return pi*sqr(r);
49|    }
50|    double circumference()
51|    {
52|        return 2*pi*r;
53|    }
54|    //0 圆外
55|    //1 圆上
56|    //2 圆内
57|    int relation(point b)
58|    {
59|        double dst=b.distance(p);
60|        if (dblcmp(dst-r)<0)return 2;
61|        if (dblcmp(dst-r)==0)return 1;
62|        return 0;
63|    }
64|    int relationseg(line v)
65|    {
66|        double dst=v.dispointtoseg(p);
67|        if (dblcmp(dst-r)<0)return 2;
68|        if (dblcmp(dst-r)==0)return 1;
69|        return 0;
70|    }
71|    int relationline(line v)
72|    {
73|        double dst=v.dispointtoline(p);
74|        if (dblcmp(dst-r)<0)return 2;
75|        if (dblcmp(dst-r)==0)return 1;
76|        return 0;
77|    }
78|    //过a 两点b 半径的两个圆r
79|    int getcircle(point a,point b,double r,circle&c1,circle&c2)
80|    {
81|        circle x(a,r),y(b,r);
82|        int t=x.pointcrosscircle(y,c1.p,c2.p);
83|        if (!t)return 0;
84|        c1.r=c2.r=r;
85|        return t;
86|    }
87|    //与直线相切u 过点q 半径的圆r1
88|    int getcircle(line u,point q,double r1,circle &c1,circle &
89|        c2)
90|    {
91|        double dis=u.dispointtoline(q);
92|        if (dblcmp(dis-r1*2)>0)return 0;
93|        if (dblcmp(dis)==0)
94|        {
95|            c1.p=q.add(u.b.sub(u.a).rotleft().trunc(r1));
96|            c2.p=q.add(u.b.sub(u.a).rotright().trunc(r1));
97|            c1.r=c2.r=r1;
98|            return 2;
99|        }
100|        line u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),u.
101|            b.add(u.b.sub(u.a).rotleft().trunc(r1)));
102|        line u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),u
103|            .b.add(u.b.sub(u.a).rotright().trunc(r1)));
104|        circle cc=circle(q,r1);
105|        point p1,p2;
106|        if (!cc.pointcrossline(u1,p1,p2))cc.pointcrossline(u2,p1,
107|            p2);
108|        c1=circle(p1,r1);
109|        if (p1==p2)
110|        {
111|            c2=c1;return 1;
112|        }
113|        c2=circle(p2,r1);
114|        return 2;
115|    }
116|    //同时与直线u,相切v 半径的圆r1
117|    int getcircle(line u,line v,double r1,circle &c1,circle &c2
118|        ,circle &c3,circle &c4)
119|    {
120|        if (u.parallel(v))return 0;
121|        line u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),u.
122|            b.add(u.b.sub(u.a).rotleft().trunc(r1)));
123|        line u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),u
124|            .b.add(u.b.sub(u.a).rotright().trunc(r1)));
125|        line v1=line(v.a.add(v.b.sub(v.a).rotleft().trunc(r1)),v.
126|            b.add(v.b.sub(v.a).rotleft().trunc(r1)));
127|        line v2=line(v.a.add(v.b.sub(v.a).rotright().trunc(r1)),v
128|            .b.add(v.b.sub(v.a).rotright().trunc(r1)));
129|        c1.r=c2.r=c3.r=c4.r=r1;
130|        c1.p=u1.crosspoint(v1);
131|        c2.p=u1.crosspoint(v2);
132|        c3.p=u2.crosspoint(v1);
133|        c4.p=u2.crosspoint(v2);
134|        return 4;
135|    }
136|    //同时与不相交圆cx,相切cy 半径为的圆r1
137|    int getcircle(circle cx,circle cy,double r1,circle&c1,circle&
138|        c2)
139|    {
140|        circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
141|        int t=x.pointcrosscircle(y,c1.p,c2.p);
142|        if (!t)return 0;
143|        c1.r=c2.r=r1;
144|        return t;
145|    }
146|    int pointcrossline(line v,point &p1,point &p2)//求与线段交要
147|        先判断relationseg
148|    {
149|        if (!(*this).relationline(v))return 0;
150|        point a=v.lineprog(p);
151|        double d=v.dispointtoline(p);
152|        d=sqrt(r*r-d*d);
153|        if (dblcmp(d)==0)
154|        {
155|            p1=a;
156|            p2=a;
157|            return 1;
158|        }
159|        p1=a.sub(v.b.sub(v.a).trunc(d));
160|        p2=a.add(v.b.sub(v.a).trunc(d));
161|        return 2;

```

```

149 |     }
150 |     //5 相离
151 | //4 外切
152 | //3 相交
153 | //2 内切
154 | //1 内含
155 | int relationcircle(circle v)
156 | {
157 |     double d=p.distance(v.p);
158 |     if (dblcmp(d-r-v.r)>0)return 5;
159 |     if (dblcmp(d-r-v.r)==0)return 4;
160 |     double l=fabs(r-v.r);
161 |     if (dblcmp(d-r-v.r)<0&&dblcmp(d-l)>0)return 3;
162 |     if (dblcmp(d-l)==0)return 2;
163 |     if (dblcmp(d-l)<0)return 1;
164 | }
165 | int pointcrosscircle(circle v,point &p1,point &p2)
166 | {
167 |     int rel=relationcircle(v);
168 |     if (rel==1||rel==5)return 0;
169 |     double d=p.distance(v.p);
170 |     double l=(d+(sqr(r)-sqr(v.r))/d)/2;
171 |     double h=sqrt(sqr(r)-sqr(l));
172 |     p1=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotleft().
173 |         trunc(h)));
174 |     p2=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotright().
175 |         trunc(h)));
176 |     if (rel==2||rel==4)
177 |     {
178 |         return 1;
179 |     }
180 |     return 2;
181 | }
182 | //过一点做圆的切线 先判断点和圆关系()
183 | int tangentline(point q,line &u,line &v)
184 | {
185 |     int x=relation(q);
186 |     if (x==2)return 0;
187 |     if (x==1)
188 |     {
189 |         u=line(q,q.add(q.sub(p).rotleft()));
190 |         v=u;
191 |         return 1;
192 |     }
193 |     double d=p.distance(q);
194 |     double l=sqr(r)/d;
195 |     double h=sqrt(sqr(r)-sqr(l));
196 |     u=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotleft().
197 |         trunc(h)));
198 |     v=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotright().
199 |         trunc(h)));
200 |     return 2;
201 | }
202 | double areacircle(circle v)
203 | {
204 |     int rel=relationcircle(v);
205 |     if (rel>=4)return 0.0;
206 |     if (rel<=2)return min(area(),v.area());
207 |     double d=p.distance(v.p);
208 |     double hf=(r+v.r+d)/2.0;
209 |     double ss=2*sqr(hf*(hf-r)*(hf-v.r)*(hf-d));
210 |     double a1=acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
211 |     a1=a1*r*r;
212 |     double a2=acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
213 |     a2=a2*v.r*v.r;
214 |     return a1+a2-ss;
215 | }
216 | double areatriangle(point a,point b)
217 | {
218 |     if (dblcmp(p.sub(a).det(p.sub(b))==0))return 0.0;
219 |     point q[5];
220 |     int len=0;
221 |     q[len++]=a;
222 |     line l(a,b);
223 |     point p1,p2;
224 |     if (pointcrossline(l,q[1],q[2])==2)
225 |     {
226 |         if (dblcmp(a.sub(q[1]).dot(b.sub(q[1])))<0)q[len
227 |             ++]=q[1];
228 |         if (dblcmp(a.sub(q[2]).dot(b.sub(q[2])))<0)q[len
229 |             ++]=q[2];
230 |     }
231 |     q[len++]=b;
232 |     if (len==4&&(dblcmp(q[0].sub(q[1]).dot(q[2].sub(q[1]))
233 |         >0))swap(q[1],q[2]));
234 |     double res=0;
235 |     int i;
236 |     for (i=0;i<len-1;i++)
237 |     {
238 |         if (relation(q[i])==0||relation(q[i+1])==0)
239 |         {
240 |             double arg=p.rad(q[i],q[i+1]);
241 |             res+=r*r*arg/2.0;
242 |         }
243 |         else

```

```

237 |         {
238 |             res+=fabs(q[i].sub(p).det(q[i+1].sub(p))/2.0);
239 |         }
240 |     }
241 |     return res;
242 | }
243 | };

```

3.2 circles

```

1 | const int maxn=500;
2 | struct circles
3 | {
4 |     circle c[maxn];
5 |     double ans[maxn]; //ans[i]表示被覆盖了i次的面积
6 |     double pre[maxn];
7 |     int n;
8 |     circles(){}
9 |     void add(circle cc)
10 |     {
11 |         c[n++]=cc;
12 |     }
13 |     bool inner(circle x,circle y)
14 |     {
15 |         if (x.relationcircle(y)!=1)return 0;
16 |         return dblcmp(x.r-y.r)<=0?1:0;
17 |     }
18 |     void init_or() //圆的面积并去掉内含的圆
19 |     {
20 |         int i,j,k=0;
21 |         bool mark[maxn]={0};
22 |         for (i=0;i<n;i++)
23 |         {
24 |             for (j=0;j<n;j++)if (i!=j&&!mark[j])
25 |             {
26 |                 if ((c[i]==c[j])||inner(c[i],c[j]))break;
27 |             }
28 |             if (j<n)mark[i]=1;
29 |         }
30 |         for (i=0;i<n;i++)if (!mark[i])c[k++]=c[i];
31 |         n=k;
32 |     }
33 |     void init_and() //圆的面积交去掉内含的圆
34 |     {
35 |         int i,j,k=0;
36 |         bool mark[maxn]={0};
37 |         for (i=0;i<n;i++)
38 |         {
39 |             for (j=0;j<n;j++)if (i!=j&&!mark[j])
40 |             {
41 |                 if ((c[i]==c[j])||inner(c[j],c[i]))break;
42 |             }
43 |             if (j<n)mark[i]=1;
44 |         }
45 |         for (i=0;i<n;i++)if (!mark[i])c[k++]=c[i];
46 |         n=k;
47 |     }
48 |     double areaarc(double th,double r)
49 |     {
50 |         return 0.5*sqr(r)*(th-sin(th));
51 |     }
52 |     void getarea()
53 |     {
54 |         int i,j,k;
55 |         memset(ans,0,sizeof(ans));
56 |         vector<pair<double,int> >v;
57 |         for (i=0;i<n;i++)
58 |         {
59 |             v.clear();
60 |             v.push_back(make_pair(-pi,1));
61 |             v.push_back(make_pair(pi,-1));
62 |             for (j=0;j<n;j++)if (i!=j)
63 |             {
64 |                 point q=c[j].p.sub(c[i].p);
65 |                 double ab=q.len(),ac=c[i].r,bc=c[j].r;
66 |                 if (dblcmp(ab+ac-bc)<=0)
67 |                 {
68 |                     v.push_back(make_pair(-pi,1));
69 |                     v.push_back(make_pair(pi,-1));
70 |                     continue;
71 |                 }
72 |                 if (dblcmp(ab+bc-ac)<=0)continue;
73 |                 if (dblcmp(ab-ac-bc)>0) continue;
74 |                 double th=atan2(q.y,q.x),fai=acos((ac*ac+ab*ab-bc*bc)
75 |                     /(2.0*ac*ab));
76 |                 double a0=th-fai;
77 |                 if (dblcmp(a0+pi)<0)a0+=2*pi;
78 |                 double a1=th+fai;
79 |                 if (dblcmp(a1-pi)>0)a1-=2*pi;
80 |                 if (dblcmp(a0-a1)>0)
81 |                 {
82 |                     v.push_back(make_pair(a0,1));
83 |                     v.push_back(make_pair(pi,-1));
84 |                     v.push_back(make_pair(-pi,1));
85 |                     v.push_back(make_pair(a1,-1));

```



```

85     }
86     else
87     {
88         v.push_back(make_pair(a0,1));
89         v.push_back(make_pair(a1,-1));
90     }
91 }
92 sort(v.begin(),v.end());
93 int cur=0;
94 for (j=0;j<v.size();j++)
95 {
96     if (cur&&dblcmp(v[j].first-pre[cur]))
97     {
98         ans[cur]+=areaarc(v[j].first-pre[cur],c[i].r);
99         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)));
100     }
101     cur+=v[j].second;
102     pre[cur]=v[j].first;
103 }
104 }
105 for (i=1;i<=n;i++)
106 {
107     ans[i]-=ans[i+1];
108 }
109 }
110 };

```

3.3 halfplane

```

1 struct halfplane:public line
2 {
3     double angle;
4     halfplane(){}
5     //表示向量 a->逆时针b左侧()的半平面
6     halfplane(point _a,point _b)
7     {
8         a=_a;
9         b=_b;
10    }
11    halfplane(line v)
12    {
13        a=v.a;
14        b=v.b;
15    }
16    void calcangle()
17    {
18        angle=atan2(b.y-a.y,b.x-a.x);
19    }
20    bool operator<(const halfplane &b)const
21    {
22        return angle<b.angle;
23    }
24 };
25 struct halfplanes
26 {
27     int n;
28     halfplane hp[maxp];
29     point p[maxp];
30     int que[maxp];
31     int st,ed;
32     void push(halfplane tmp)
33     {
34         hp[n++]=tmp;
35     }
36     void unique()
37     {
38         int m=1,i;
39         for (i=1;i<n;i++)
40         {
41             if (dblcmp(hp[i].angle-hp[i-1].angle))hp[m++]=hp[i];
42             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];
43         }
44         n=m;
45     }
46     bool halfplaneinsert()
47     {
48         int i;
49         for (i=0;i<n;i++)hp[i].calcangle();
50         sort(hp,hp+n);
51         unique();
52         que[st=0]=0;
53         que[ed=1]=1;
54         p[1]=hp[0].crosspoint(hp[1]);
55         for (i=2;i<n;i++)
56         {
57             while (st<ed&&dblcmp((hp[i].b.sub(hp[i].a).det(p[ed].sub(hp[i].a)))<0)ed--;
58             while (st<ed&&dblcmp((hp[i].b.sub(hp[i].a).det(p[st+1].sub(hp[i].a)))<0)st++;
59             que[++ed]=i;
60             if (hp[i].parallel(hp[que[ed-1]]))return false;

```

```

61         p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
62     }
63     while (st<ed&&dblcmp(hp[que[st]].b.sub(hp[que[st]].a).det(p[ed].sub(hp[que[st]].a)))<0)ed--;
64     while (st<ed&&dblcmp(hp[que[ed]].b.sub(hp[que[ed]].a).det(p[st+1].sub(hp[que[ed]].a)))<0)st++;
65     if (st+1==ed)return false;
66     return true;
67 }
68 void getconvex(polygon &con)
69 {
70     p[st]=hp[que[st]].crosspoint(hp[que[ed]]);
71     con.n=ed-st+1;
72     int j=st,i=0;
73     for (;j<=ed;i++,j++)
74     {
75         con.p[i]=p[j];
76     }
77 }
78 };

```

3.4 line

```

1 struct line
2 {
3     point a,b;
4     line(){}
5     line(point _a,point _b)
6     {
7         a=_a;
8         b=_b;
9     }
10    bool operator==(line v)
11    {
12        return (a==v.a)&&(b==v.b);
13    }
14    //倾斜角angle
15    line(point p,double angle)
16    {
17        a=p;
18        if (dblcmp(angle-pi/2)==0)
19        {
20            b=a.add(point(0,1));
21        }
22        else
23        {
24            b=a.add(point(1,tan(angle)));
25        }
26    }
27    //ax+by+c=0
28    line(double _a,double _b,double _c)
29    {
30        if (dblcmp(_a)==0)
31        {
32            a=point(0,-_c/_b);
33            b=point(1,-_c/_b);
34        }
35        else if (dblcmp(_b)==0)
36        {
37            a=point(-_c/_a,0);
38            b=point(-_c/_a,1);
39        }
40        else
41        {
42            a=point(0,-_c/_b);
43            b=point(1,(-_c-a)/_b);
44        }
45    }
46    void input()
47    {
48        a.input();
49        b.input();
50    }
51    void adjust()
52    {
53        if (b<a)swap(a,b);
54    }
55    double length()
56    {
57        return a.distance(b);
58    }
59    double angle()//直线倾斜角 0<=angle<180
60    {
61        double k=atan2(b.y-a.y,b.x-a.x);
62        if (dblcmp(k)<0)k+=pi;
63        if (dblcmp(k-pi)==0)k=-pi;
64        return k;
65    }
66    //点和线段关系
67    //1 在逆时针
68    //2 在顺时针
69    //3 平行
70    int relation(point p)
71    {

```

```

72     int c=dblcmp(p.sub(a).det(b.sub(a)));
73     if (c<0)return 1;
74     if (c>0)return 2;
75     return 3;
76 }
77 bool pointonseg(point p)
78 {
79     return dblcmp(p.sub(a).det(b.sub(a)))==0&&dblcmp(p.sub(
80         a).dot(p.sub(b)))<=0;
81 }
82 bool parallel(line v)
83 {
84     return dblcmp(b.sub(a).det(v.b.sub(v.a)))==0;
85 }
86 //2 规范相交
87 //1 非规范相交
88 //0 不相交
89 int segcrossseg(line v)
90 {
91     int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
92     int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
93     int d3=dblcmp(v.b.sub(v.a).det(a.sub(v.a)));
94     int d4=dblcmp(v.b.sub(v.a).det(b.sub(v.a)));
95     if ((d1^d2)==-2&&(d3^d4)==-2)return 2;
96     return (d1==0&&dblcmp(v.a.sub(a).dot(v.a.sub(b)))<=0||
97         d2==0&&dblcmp(v.b.sub(a).dot(v.b.sub(b)))<=0||
98         d3==0&&dblcmp(a.sub(v.a).dot(a.sub(v.b)))<=0||
99         d4==0&&dblcmp(b.sub(v.a).dot(b.sub(v.b)))<=0);
100 }
101 int linecrossseg(line v)/*this seg v line
102 {
103     int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
104     int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
105     if ((d1^d2)==-2)return 2;
106     return (d1==0||d2==0);
107 }
108 //0 平行
109 //1 重合
110 //2 相交
111 int linecrossline(line v)
112 {
113     if ((*this).parallel(v))
114     {
115         return v.relation(a)==3;
116     }
117     return 2;
118 }
119 point crosspoint(line v)
120 {
121     double a1=v.b.sub(v.a).det(a.sub(v.a));
122     double a2=v.b.sub(v.a).det(b.sub(v.a));
123     return point((a.x*a2-b.x*a1)/(a2-a1),(a.y*a2-b.y*a1)/(
124         a2-a1));
125 }
126 double dispointtoline(point p)
127 {
128     return fabs(p.sub(a).det(b.sub(a)))/length();
129 }
130 double dispointtoseg(point p)
131 {
132     if (dblcmp(p.sub(b).dot(a.sub(b)))<0||dblcmp(p.sub(a).
133         dot(b.sub(a)))<0)
134     {
135         return min(p.distance(a),p.distance(b));
136     }
137     return dispointtoline(p);
138 }
139 point lineprog(point p)
140 {
141     return a.add(b.sub(a).mul(b.sub(a).dot(p.sub(a))/b.sub(
142         a).len(2)));
143 }
144 point symmetrypoint(point p)
145 {
146     point q=lineprog(p);
147     return point(2*q.x-p.x,2*q.y-p.y);
148 }
149 };

```

3.5 line3d

```

1 struct line3
2 {
3     point3 a,b;
4     line3(){}
5     line3(point3 _a,point3 _b)
6     {
7         a=_a;
8         b=_b;
9     }
10    bool operator==(line3 v)
11    {
12        return (a==v.a)&&(b==v.b);
13    }

```

```

14 void input()
15 {
16     a.input();
17     b.input();
18 }
19 double length()
20 {
21     return a.distance(b);
22 }
23 bool pointonseg(point3 p)
24 {
25     return dblcmp(p.sub(a).det(p.sub(b)).len())==0&&dblcmp(a.
26         sub(p).dot(b.sub(p)))<=0;
27 }
28 double dispointtoline(point3 p)
29 {
30     return b.sub(a).det(p.sub(a)).len()/a.distance(b);
31 }
32 double dispointtoseg(point3 p)
33 {
34     if (dblcmp(p.sub(b).dot(a.sub(b)))<0||dblcmp(p.sub(a).
35         dot(b.sub(a)))<0)
36     {
37         return min(p.distance(a),p.distance(b));
38     }
39     return dispointtoline(p);
40 }
41 point3 lineprog(point3 p)
42 {
43     return a.add(b.sub(a).trunc(b.sub(a).dot(p.sub(a))/b.
44         distance(a)));
45 }
46 point3 rotate(point3 p,double ang)//绕此向量逆时针角度pang
47 {
48     if (dblcmp((p.sub(a).det(p.sub(b)).len()))==0)return p;
49     point3 f1=b.sub(a).det(p.sub(a));
50     point3 f2=b.sub(a).det(f1);
51     double len=fabs(a.sub(p).det(b.sub(p)).len()/a.distance(b))
52     ;
53     f1=f1.trunc(len);f2=f2.trunc(len);
54     point3 h=p.add(f2);
55     point3 pp=h.add(f1);
56     return h.add((p.sub(h)).mul(cos(ang*1.0)).add((pp.sub(h)).
57         mul(sin(ang*1.0)));
58 }
59 };

```

3.6 plane

```

1 struct plane
2 {
3     point3 a,b,c,o;
4     plane(){}
5     plane(point3 _a,point3 _b,point3 _c)
6     {
7         a=_a;
8         b=_b;
9         c=_c;
10        o=pvec();
11    }
12    plane(double _a,double _b,double _c,double _d)
13    {
14        //ax+by+cz+d=0
15        o=point3(_a,_b,_c);
16        if (dblcmp(_a)!=0)
17        {
18            a=point3((-_d-_c-_b)/_a,1,1);
19        }
20        else if (dblcmp(_b)!=0)
21        {
22            a=point3(1,(-_d-_c-_a)/_b,1);
23        }
24        else if (dblcmp(_c)!=0)
25        {
26            a=point3(1,1,(-_d-_a-_b)/_c);
27        }
28    }
29    void input()
30    {
31        a.input();
32        b.input();
33        c.input();
34        o=pvec();
35    }
36    point3 pvec()
37    {
38        return b.sub(a).det(c.sub(a));
39    }
40    bool pointonplane(point3 p)//点是否在平面上
41    {
42        return dblcmp(p.sub(a).dot(o))==0;
43    }
44    //0 不在
45    //1 在边界上
46    //2 在内部

```

```

47 | int pointontriangle(point3 p)//点是否在空间三角形上abc
48 | {
49 |     if (!pointonplane(p))return 0;
50 |     double s=a.sub(b).det(c.sub(b)).len();
51 |     double s1=p.sub(a).det(p.sub(b)).len();
52 |     double s2=p.sub(a).det(p.sub(c)).len();
53 |     double s3=p.sub(b).det(p.sub(c)).len();
54 |     if (dblcmp(s-s1-s2-s3))return 0;
55 |     if (dblcmp(s1)&&dblcmp(s2)&&dblcmp(s3))return 2;
56 |     return 1;
57 | }
58 | //判断两平面关系
59 | //0 相交
60 | //1 平行但不重合
61 | //2 重合
62 | bool relationplane(plane f)
63 | {
64 |     if (dblcmp(o.det(f.o).len()))return 0;
65 |     if (pointonplane(f.a))return 2;
66 |     return 1;
67 | }
68 | double angleplane(plane f)//两平面夹角
69 | {
70 |     return acos(o.dot(f.o)/(o.len()*f.o.len()));
71 | }
72 | double dispoint(point3 p)//点到平面距离
73 | {
74 |     return fabs(p.sub(a).dot(o)/o.len());
75 | }
76 | point3 pttoplane(point3 p)//点到平面最近点
77 | {
78 |     line3 u=line3(p,p.add(o));
79 |     crossline(u,p);
80 |     return p;
81 | }
82 | int crossline(line3 u,point3 &p)//平面和直线的交点
83 | {
84 |     double x=o.dot(u.b.sub(a));
85 |     double y=o.dot(u.a.sub(a));
86 |     double d=x-y;
87 |     if (dblcmp(fabs(d))==0)return 0;
88 |     p=u.a.mul(x).sub(u.b.mul(y)).div(d);
89 |     return 1;
90 | }
91 | int crossplane(plane f,line3 &u)//平面和平面的交线
92 | {
93 |     point3 oo=o.det(f.o);
94 |     point3 v=o.det(oo);
95 |     double d=fabs(f.o.dot(v));
96 |     if (dblcmp(d)==0)return 0;
97 |     point3 q=a.add(v.mul(f.o.dot(f.a.sub(a))/d));
98 |     u=line3(q,q.add(oo));
99 |     return 1;
100 | }
101 | };

```

3.7 point

```

1 | using namespace std;
2 |
3 | #define mp make_pair
4 | #define pb push_back
5 |
6 | const double eps=1e-8;
7 | const double pi=acos(-1.0);
8 | const double inf=1e20;
9 | const int maxp=8;
10 |
11 | int dblcmp(double d)
12 | {
13 |     if (fabs(d)<eps)return 0;
14 |     return d>eps?1:-1;
15 | }
16 |
17 | inline double sqr(double x)
18 | {
19 |     return x*x;
20 | }
21 |
22 | struct point
23 | {
24 |     double x,y;
25 |     point(){ }
26 |     point(double _x,double _y):
27 |     x(_x),y(_y){ };
28 |     void input()
29 |     {
30 |         scanf("%lf%lf",&x,&y);
31 |     }
32 |     void output()
33 |     {
34 |         printf("%.2f %.2f\n",x,y);
35 |     }
36 |     bool operator==(point a)const

```

```

37 | {
38 |     return dblcmp(a.x-x)==0&&dblcmp(a.y-y)==0;
39 | }
40 | bool operator<(point a)const
41 | {
42 |     return dblcmp(a.x-x)==0?dblcmp(y-a.y)<0:x<a.x;
43 | }
44 | double len()
45 | {
46 |     return hypot(x,y);
47 | }
48 | double len2()
49 | {
50 |     return x*x+y*y;
51 | }
52 | double distance(point p)
53 | {
54 |     return hypot(x-p.x,y-p.y);
55 | }
56 | point add(point p)
57 | {
58 |     return point(x+p.x,y+p.y);
59 | }
60 | point sub(point p)
61 | {
62 |     return point(x-p.x,y-p.y);
63 | }
64 | point mul(double b)
65 | {
66 |     return point(x*b,y*b);
67 | }
68 | point div(double b)
69 | {
70 |     return point(x/b,y/b);
71 | }
72 | double dot(point p)
73 | {
74 |     return x*p.x+y*p.y;
75 | }
76 | double det(point p)
77 | {
78 |     return x*p.y-y*p.x;
79 | }
80 | double rad(point a,point b)
81 | {
82 |     point p=*this;
83 |     return fabs(atan2(fabs(a.sub(p).det(b.sub(p))),a.sub(p).
84 |         dot(b.sub(p))));
85 | }
86 | point trunc(double r)
87 | {
88 |     double l=len();
89 |     if (!dblcmp(l))return *this;
90 |     r/=l;
91 |     return point(x*r,y*r);
92 | }
93 | point rotleft()
94 | {
95 |     return point(-y,x);
96 | }
97 | point rotright()
98 | {
99 |     return point(y,-x);
100 | }
101 | point rotate(point p,double angle)//绕点逆时针旋转角度pangle
102 | {
103 |     point v=*this->sub(p);
104 |     double c=cos(angle),s=sin(angle);
105 |     return point(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
106 | };

```

3.8 point3d

```

1 | struct point3
2 | {
3 |     double x,y,z;
4 |     point3(){ }
5 |     point3(double _x,double _y,double _z):
6 |     x(_x),y(_y),z(_z){ };
7 |     void input()
8 |     {
9 |         scanf("%lf%lf%lf",&x,&y,&z);
10 |     }
11 |     void output()
12 |     {
13 |         printf("%.2lf %.2lf %.2lf\n",x,y,z);
14 |     }
15 |     bool operator==(point3 a)
16 |     {
17 |         return dblcmp(a.x-x)==0&&dblcmp(a.y-y)==0&&dblcmp(a.z-z)
18 |             ==0;
19 |     }
20 |     bool operator<(point3 a)const

```

```

21     return dblcmp(a.x-x)==0?dblcmp(y-a.y)==0?dblcmp(z-a.z)
    <0:y<a.y:x<a.x;
22 }
23 double len()
24 {
25     return sqrt(len2());
26 }
27 double len2()
28 {
29     return x*x+y*y+z*z;
30 }
31 double distance(point3 p)
32 {
33     return sqrt((p.x-x)*(p.x-x)+(p.y-y)*(p.y-y)+(p.z-z)*(p.z-z));
34 }
35 point3 add(point3 p)
36 {
37     return point3(x+p.x,y+p.y,z+p.z);
38 }
39 point3 sub(point3 p)
40 {
41     return point3(x-p.x,y-p.y,z-p.z);
42 }
43 point3 mul(double d)
44 {
45     return point3(x*d,y*d,z*d);
46 }
47 point3 div(double d)
48 {
49     return point3(x/d,y/d,z/d);
50 }
51 double dot(point3 p)
52 {
53     return x*p.x+y*p.y+z*p.z;
54 }
55 point3 det(point3 p)
56 {
57     return point3(y*p.z-p.y*z,p.x*z-x*p.z,x*p.y-p.x*y);
58 }
59 double rad(point3 a,point3 b)
60 {
61     point3 p=(*this);
62     return acos(a.sub(p).dot(b.sub(p))/(a.distance(p)*b.distance(p)));
63 }
64 point3 trunc(double r)
65 {
66     r/=len();
67     return point3(x*r,y*r,z*r);
68 }
69 point3 rotate(point3 o,double r) // building?
70 {
71 }
72 };

```

3.9 polygon

```

1 struct polygon
2 {
3     int n;
4     point p[maxp];
5     line l[maxp];
6     void input()
7     {
8         n=4;
9         p[0].input();
10        p[2].input();
11        double dis=p[0].distance(p[2]);
12        p[1]=p[2].rotate(p[0],pi/4);
13        p[1]=p[0].add((p[1].sub(p[0])).trunc(dis/sqrt(2.0)));
14        p[3]=p[2].rotate(p[0],2*pi-pi/4);
15        p[3]=p[0].add((p[3].sub(p[0])).trunc(dis/sqrt(2.0)));
16    }
17    void add(point q)
18    {
19        p[n++]=q;
20    }
21    void getline()
22    {
23        for (int i=0;i<n;i++)
24        {
25            l[i]=line(p[i],p[(i+1)%n]);
26        }
27    }
28    struct cmp
29    {
30        point p;
31        cmp(const point &p0){p=p0;}
32        bool operator()(const point &aa,const point &bb)
33        {
34            point a=aa,b=bb;
35            int d=dblcmp(a.sub(p).det(b.sub(p)));
36            if (d==0)
37            {
38                return dblcmp(a.distance(p)-b.distance(p))<0;
39            }
40            return d>0;
41        }
42    };
43    void norm()
44    {
45        point mi=p[0];
46        for (int i=1;i<n;i++)mi=min(mi,p[i]);
47        sort(p,p+n,cmp(mi));
48    }
49    void getconvex(polygon &convex)
50    {
51        int i,j,k;
52        sort(p,p+n);
53        convex.n=n;
54        for (i=0;i<min(n,2);i++)
55        {
56            convex.p[i]=p[i];
57        }
58        if (n<=2)return;
59        int &top=convex.n;
60        top=1;
61        for (i=2;i<n;i++)
62        {
63            while (top&&convex.p[top].sub(p[i]).det(convex.p[top-1].sub(p[i]))<=0)
64                top--;
65            convex.p[++top]=p[i];
66        }
67        int temp=top;
68        convex.p[++top]=p[n-2];
69        for (i=n-3;i>=0;i--)
70        {
71            while (top!=temp&&convex.p[top].sub(p[i]).det(convex.p[top-1].sub(p[i]))<=0)
72                top--;
73            convex.p[++top]=p[i];
74        }
75    }
76    bool isconvex()
77    {
78        bool s[3];
79        memset(s,0,sizeof(s));
80        int i,j,k;
81        for (i=0;i<n;i++)
82        {
83            j=(i+1)%n;
84            k=(j+1)%n;
85            s[dblcmp(p[j].sub(p[i]).det(p[k].sub(p[i])))+1]=1;
86            if (s[0]&&s[2])return 0;
87        }
88        return 1;
89    }
90    //3 点上
91    //2 边上
92    //1 内部
93    //0 外部
94    int relationpoint(point q)
95    {
96        int i,j;
97        for (i=0;i<n;i++)
98        {
99            if (p[i]==q)return 3;
100        }
101        getline();
102        for (i=0;i<n;i++)
103        {
104            if (l[i].pointonseg(q))return 2;
105        }
106        int cnt=0;
107        for (i=0;i<n;i++)
108        {
109            j=(i+1)%n;
110            int k=dblcmp(q.sub(p[j]).det(p[i].sub(p[j])));
111            int u=dblcmp(p[i].y-q.y);
112            int v=dblcmp(p[j].y-q.y);
113            if (k>0&&u<0&&v>=0)cnt++;
114            if (k<0&&v<0&&u>=0)cnt--;
115        }
116        return cnt!=0;
117    }
118    //1 在多边形内长度为正
119    //2 相交或与边平行
120    //0 无任何交点
121    int relationline(line u)
122    {
123        int i,j,k=0;
124        getline();
125        for (i=0;i<n;i++)
126        {
127            if (l[i].segcrossseg(u)==2)return 1;
128            if (l[i].segcrossseg(u)==1)k=1;
129        }
130        if (!k)return 0;

```

```

131 vector<point>vp;
132 for (i=0;i<n;i++)
133 {
134     if (l[i].segcrossseg(u))
135     {
136         if (l[i].parallel(u))
137         {
138             vp.pb(u.a);
139             vp.pb(u.b);
140             vp.pb(l[i].a);
141             vp.pb(l[i].b);
142             continue;
143         }
144         vp.pb(l[i].crosspoint(u));
145     }
146 }
147 sort(vp.begin(),vp.end());
148 int sz=vp.size();
149 for (i=0;i<sz-1;i++)
150 {
151     point mid=vp[i].add(vp[i+1]).div(2);
152     if (relationpoint(mid)==1)return 1;
153 }
154 return 2;
155 }
156 //直线切割凸多边形左侧u
157 //注意直线方向
158 void convexcute(line u,polygon &po)
159 {
160     int i,j,k;
161     int &top=po.n;
162     top=0;
163     for (i=0;i<n;i++)
164     {
165         int d1=dblcmp(p[i].sub(u.a).det(u.b.sub(u.a)));
166         int d2=dblcmp(p[(i+1)%n].sub(u.a).det(u.b.sub(u.a)));
167         if (d1>=0)po.p[top++]=p[i];
168         if (d1*d2<0)po.p[top++]=u.crosspoint(line(p[i],p[(i+1)%n]));
169     }
170 }
171 double getcircumference()
172 {
173     double sum=0;
174     int i;
175     for (i=0;i<n;i++)
176     {
177         sum+=p[i].distance(p[(i+1)%n]);
178     }
179     return sum;
180 }
181 double getarea()
182 {
183     double sum=0;
184     int i;
185     for (i=0;i<n;i++)
186     {
187         sum+=p[i].det(p[(i+1)%n]);
188     }
189     return fabs(sum)/2;
190 }
191 bool getdir()//代表逆时针 1 代表顺时针0
192 {
193     double sum=0;
194     int i;
195     for (i=0;i<n;i++)
196     {
197         sum+=p[i].det(p[(i+1)%n]);
198     }
199     if (dblcmp(sum)>0)return 1;
200     return 0;
201 }
202 point getbarycentre() // centroid
203 {
204     point ret(0,0);
205     double area=0;
206     int i;
207     for (i=1;i<n-1;i++)
208     {
209         double tmp=p[i].sub(p[0]).det(p[i+1].sub(p[0]));
210         if (dblcmp(tmp)==0)continue;
211         area+=tmp;
212         ret.x+=(p[0].x+p[i].x+p[i+1].x)/3*tmp;
213         ret.y+=(p[0].y+p[i].y+p[i+1].y)/3*tmp;
214     }
215     if (dblcmp(area))ret=ret.div(area);
216     return ret;
217 }
218 double areaintersection(polygon po) // refer: HPI
219 {
220 }
221 double areaunion(polygon po)
222 {
223     return getarea()+po.getarea()-areaintersection(po);
224 }
225 double areacircle(circle c)
226 {
227     int i,j,k,l,m;
228     double ans=0;
229     for (i=0;i<n;i++)
230     {
231         int j=(i+1)%n;
232         if (dblcmp(p[j].sub(c.p).det(p[i].sub(c.p)))>=0)
233         {
234             ans+=c.reatriangle(p[i],p[j]);
235         }
236         else
237         {
238             ans-=c.reatriangle(p[i],p[j]);
239         }
240     }
241     return fabs(ans);
242 }
243 //多边形和圆关系
244 //0 一部分在圆外
245 //1 与圆某条边相切
246 //2 完全在圆内
247 int relationcircle(circle c)
248 {
249     getline();
250     int i,x=2;
251     if (relationpoint(c.p)!=1)return 0;
252     for (i=0;i<n;i++)
253     {
254         if (c.relationseg(l[i])==2)return 0;
255         if (c.relationseg(l[i])==1)x=1;
256     }
257     return x;
258 }
259 void find(int st,point tri[],circle &c)
260 {
261     if (!st)
262     {
263         c=circle(point(0,0),-2);
264     }
265     if (st==1)
266     {
267         c=circle(tri[0],0);
268     }
269     if (st==2)
270     {
271         c=circle(tri[0].add(tri[1]).div(2),tri[0].distance(tri[1])/2.0);
272     }
273     if (st==3)
274     {
275         c=circle(tri[0],tri[1],tri[2]);
276     }
277 }
278 void solve(int cur,int st,point tri[],circle &c)
279 {
280     find(st,tri,c);
281     if (st==3)return;
282     int i;
283     for (i=0;i<cur;i++)
284     {
285         if (dblcmp(p[i].distance(c.p)-c.r)>0)
286         {
287             tri[st]=p[i];
288             solve(i,st+1,tri,c);
289         }
290     }
291 }
292 circle mincircle()//点集最小圆覆盖
293 {
294     random_shuffle(p,p+n);
295     point tri[4];
296     circle c;
297     solve(n,0,tri,c);
298     return c;
299 }
300 int circlecover(double r)//单位圆覆盖
301 {
302     int ans=0,i,j;
303     vector<pair<double,int> >v;
304     for (i=0;i<n;i++)
305     {
306         v.clear();
307         for (j=0;j<n;j++)if (i!=j)
308         {
309             point q=p[i].sub(p[j]);
310             double d=q.len();
311             if (dblcmp(d-2*r)<=0)
312             {
313                 double arg=atan2(q.y,q.x);
314                 if (dblcmp(arg)<0)arg+=2*pi;
315                 double t=acos(d/(2*r));
316                 v.push_back(make_pair(arg-t+2*pi,-1));
317                 v.push_back(make_pair(arg+t+2*pi,1));
318             }
319         }
320     }
321 }

```

```

319     }
320     sort(v.begin(),v.end());
321     int cur=0;
322     for (j=0;j<v.size();j++)
323     {
324         if (v[j].second==-1)+cur;
325         else —cur;
326         ans=max(ans,cur);
327     }
328 }
329 return ans+1;
330 }
331 int pointinpolygon(point q)//点在凸多边形内部的判定
332 {
333     if (getdir())reverse(p,p+n);
334     if (dblcmp(q.sub(p[0]).det(p[n-1].sub(p[0])))==0)
335     {
336         if (line(p[n-1],p[0]).pointonseg(q))return n-1;
337         return -1;
338     }
339     int low=1,high=n-2,mid;
340     while (low<=high)
341     {
342         mid=(low+high)>>1;
343         if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0])))>=0&&dblcmp(
344             q.sub(p[0]).det(p[mid+1].sub(p[0]))<0)
345         {
346             polygon c;
347             c.p[0]=p[mid];
348             c.p[1]=p[mid+1];
349             c.p[2]=p[0];
350             c.n=3;
351             if (c.relationpoint(q))return mid;
352             return -1;
353         }
354         if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0])))>0)
355         {
356             low=mid+1;
357         }
358         else
359         {
360             high=mid-1;
361         }
362     }
363     return -1;
364 }
};

```

3.10 polygons

```

1 struct polygons
2 {
3     vector<polygon>p;
4     polygons()
5     {
6         p.clear();
7     }
8     void clear()
9     {
10        p.clear();
11    }
12    void push(polygon q)
13    {
14        if (dblcmp(q.getarea()))p.pb(q);
15    }
16    vector<pair<double,int> >e;
17    void ins(point s,point t,point X,int i)
18    {
19        double r=fabs(t.x-s.x)>eps?(X.x-s.x)/(t.x-s.x):(X.y-s.y)/(t
20            .y-s.y);
21        r=min(r,1.0);r=max(r,0.0);
22        e.pb(mp(r,i));
23    }
24    double polyareaunion()
25    {
26        double ans=0.0;
27        int c0,c1,c2,i,j,k,w;
28        for (i=0;i<p.size();i++)
29        {
30            if (p[i].getdir()==0)reverse(p[i].p,p[i].p+p[i].n);
31        }
32        for (i=0;i<p.size();i++)
33        {
34            for (k=0;k<p[i].n;k++)
35            {
36                point &s=p[i].p[k],&t=p[i].p[(k+1)%p[i].n];
37                if (!dblcmp(s.det(t)))continue;
38                e.clear();
39                e.pb(mp(0.0,1));
40                e.pb(mp(1.0,-1));
41                for (j=0;j<p.size();j++)if (i!=j)
42                {
43                    for (w=0;w<p[j].n;w++)
44                    {

```

```

45                        w+1+p[j].n)%p[j].n);
46                        c0=dblcmp(t.sub(s).det(c.sub(s)));
47                        c1=dblcmp(t.sub(s).det(a.sub(s)));
48                        c2=dblcmp(t.sub(s).det(b.sub(s)));
49                        if (c1*c2<0)ins(s,t,line(s,t).crosspoint(line(a,b)
50                            ,-c2);
51                        else if (!c1&&c0*c2<0)ins(s,t,a,-c2);
52                        else if (!c1&&!c2)
53                        {
54                            int c3=dblcmp(t.sub(s).det(p[j].p[(w+2)%p[j].n].
55                                sub(s)));
56                            int dp=dblcmp(t.sub(s).dot(b.sub(a)));
57                            if (dp&&c0)ins(s,t,a,dp>0?c0*((j>i)^(c0<0)):-c0
58                                <0));
59                            if (dp&&c3)ins(s,t,b,dp>0?-c3*((j>i)^(c3<0)):c3
60                                <0);
61                        }
62                    }
63                }
64                sort(e.begin(),e.end());
65                int ct=0;
66                double tot=0.0,last;
67                for (j=0;j<e.size();j++)
68                {
69                    if (ct==p.size())tot+=e[j].first-last;
70                    ct+=e[j].second;
71                    last=e[j].first;
72                }
73                ans+=s.det(t)*tot;
74            }
75        }
76        return fabs(ans)*0.5;
77    }
};

```

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
83 // order
84 // or maybe we can solve it with dual SCC method, and get a
85 // solution by reverse the edges of DAG then product a
86 // 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<
6         !=edge[now].end();++it)
7         if(dfn[*it]==-1)
8         {
9             dfs(*it,now);
10            ++p;
11            low[now]=std::min(low[now],low[*it]);
12            if((now==1 && p>1) || (now!=1 && low[*it]>=dfn[now]
13                )) // 如果从出发点出发的子节点不能由兄弟节点到达, 那
14                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
15                能达到祖先节点, 那么该节点为割点
16            ans.insert(now);
17        }
18    }
19    else
20        if(*it!=fa)
21            low[now]=std::min(low[now],dfn[*it]);
22 }

```

4.3 Augmenting Path Algorithm for Maximum Cardinality Bipartite Matching

```

1 #include<cstdio>
2 #include<cstring>
3
4 #define MAXX 111
5
6 bool Map[MAXX][MAXX],visit[MAXX];
7 int link[MAXX],n,m;
8 bool dfs(int t)
9 {
10     for (int i=0; i<m; i++)
11         if (!visit[i] && Map[t][i]){
12             visit[i] = true;
13             if (link[i]==-1 || dfs(link[i])){
14                 link[i] = t;
15                 return true;
16             }
17         }
18     return false;
19 }
20 int main()
21 {
22     int k,a,b,c;
23     while (scanf("%d",&n),n){
24         memset(Map,false,sizeof(Map));
25         scanf("%d",&m,&k);
26         while (k--){
27             scanf("%d%d",&a,&b,&c);
28             if (b && c)
29                 Map[b][c] = true;
30         }
31         memset(link,-1,sizeof(link));
32         int ans = 0;
33         for (int i=0; i<n; i++){
34             memset(visit,false,sizeof(visit));
35             if (dfs(i))

```

```

36         ans++;
37     }
38     printf("%d\n",ans);
39 }
40 }

```

4.4 Biconnected Component - Edge

```

1 // hdu 4612
2 #include<cstdio>
3 #include<algorithm>
4 #include<set>
5 #include<cstring>
6 #include<stack>
7 #include<queue>
8
9 #define MAXX 200111
10 #define MAXE (1000111*2)
11 #pragma comment(linker, "/STACK:16777216")
12
13 int edge[MAXX],to[MAXE],nxt[MAXE],cnt;
14 #define v to[i]
15 inline void add(int a,int b)
16 {
17     nxt[++cnt]=edge[a];
18     edge[a]=cnt;
19     to[cnt]=b;
20 }
21
22 int dfn[MAXX],low[MAXX],col[MAXX],belong[MAXX];
23 int idx,bcnt;
24 std::stack<int>st;
25
26 void tarjan(int now,int last)
27 {
28     col[now]=1;
29     st.push(now);
30     dfn[now]=low[now]=++idx;
31     bool flag(false);
32     for(int i(edge[now]);i;i=nxt[i])
33     {
34         if(v==last && !flag)
35         {
36             flag=true;
37             continue;
38         }
39         if(!col[v])
40         {
41             tarjan(v,now);
42             low[now]=std::min(low[now],low[v]);
43             /*
44              if(low[v]>dfn[now])
45              then this is a bridge
46              */
47         }
48     }
49     else
50         if(col[v]==1)
51             low[now]=std::min(low[now],dfn[v]);
52 }
53 col[now]=2;
54 if(dfn[now]==low[now])
55 {
56     ++bcnt;
57     static int x;
58     do
59     {
60         x=st.top();
61         st.pop();
62         belong[x]=bcnt;
63     }while(x!=now);
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 }

```

```

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[MAXN<<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[MAXN];
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     {
46         if(m[y]==-1)
47         {
48             for(i=0;i+1<p[x].size();i+=2)
49             {
50                 m[p[x][i]]=p[x][i+1];
51                 m[p[x][i+1]]=p[x][i];
52             }
53             m[x]=y;
54             m[y]=x;
55             return true;
56         }
57         else
58         {
59             p[z=m[y]]=p[x];
60             p[z].push_back(y);
61             p[z].push_back(z);
62             vis[y]=1;
63             vis[z]=0;
64             *qb++=z;
65         }
66     }
67     else
68     {
69         for(b=0;b<p[x].size() && b<p[y].size() &&
70             [x][b]=p[y][b];++b;
71         --b;
72         label(x,y,b);
73         label(y,x,b);
74     }
75 }
76 return false;
77 }
78 int i,j,k;
79 int ans;
80 int main()
81 {
82     scanf("%d",&n);
83     for(i=0;i<n;++i)
84     p[i].reserve(n);
85     while(scanf("%d%d",&i,&j)!=EOF)
86     {
87         --i;
88         --j;
89         map[i][j]=map[j][i]=true;
90     }
91     memset(m,-1,sizeof m);
92     for(i=0;i<n;++i)
93     {
94         if(m[i]==-1)
95         {
96             if(bfs(i))
97                 ++ans;
98             else
99                 m[i]=i;
100         }
101         printf("%d\n",ans<<1);
102     }
103     for(i=0;i<n;++i)
104     {
105         if(i<m[i])
106             printf("%d%d\n",i+1,m[i]+1);
107     }
108     return 0;
109 }

```

4.7 Bridge

```

1 void dfs(const short &now,const short &fa)
2 {
3     dfn[now]=low[now]=cnt++;
4     for(int i(0);i<edge[now].size();++i)
5     {
6         if(dfn[edge[now][i]]==0)
7         {
8             dfs(edge[now][i],now);
9             low[now]=std::min(low[now],low[edge[now][i]]);
10            if(low[edge[now][i]]>dfn[now]) //如果子节点不能够走到
11                父节点之前去, 那么该边为桥
12            {
13                if(edge[now][i]<now)
14                {
15                    j=edge[now][i];
16                    k=now;
17                }
18                else
19                {
20                    j=now;
21                    k=edge[now][i];
22                }
23                ans.push_back(node(j,k));
24            }
25        }
26        else
27        {
28            if(edge[now][i]!=fa)
29                low[now]=std::min(low[now],low[edge[now][i]]);
30        }
31    }
32 }

```

4.8 Chu-Liu:Edmonds' Algorithm

```

1 #include<cstdio>
2 #include<cstring>
3 #include<vector>
4
5 #define MAXX 1111
6 #define MAXE 10111
7 #define inf 0x3f3f3f3f
8
9 int n,m,i,j,k,ans,u,v,tn,rt,sum,on,om;
10 int pre[MAXX],id[MAXX],in[MAXX],vis[MAXX];
11
12 struct edge
13 {
14     int a,b,c;
15     edge(){}
16     edge(int aa,int bb,int cc):a(aa),b(bb),c(cc){}
17 };
18 std::vector<edge>ed(MAXE);
19
20 int main()
21 {
22     while(scanf("%d%d",&n,&m)!=EOF)
23     {
24         on=n;
25         om=m;
26         ed.resize(0);
27         sum=1;
28         while(m--)
29         {
30             scanf("%d%d%d",&i,&j,&k);
31             if(i!=j)
32             {
33                 ed.push_back(edge(i,j,k));
34                 sum+=k;
35             }
36         }
37         ans=0;
38         rt=n;
39         for(i=0;i<n;++i)
40             ed.push_back(edge(n,i,sum));
41         ++n;
42         while(true)
43         {
44             memset(in,0x3f,sizeof in);
45             for(i=0;i<ed.size();++i)
46             {
47                 if(ed[i].a!=ed[i].b && in[ed[i].b]>ed[i].c)
48                 {
49                     in[ed[i].b]=ed[i].c;
50                     pre[ed[i].b]=ed[i].a;
51                     if(ed[i].a==rt)
52                         j=i;
53                 }
54             }
55             for(i=0;i<n;++i)
56             {
57                 if(i!=rt && in[i]==inf)
58                     goto ot;
59             }
60             memset(id,-1,sizeof id);
61             memset(vis,-1,sizeof vis);
62             tn=in[rt]=0;
63             for(i=0;i<n;++i)
64             {
65                 ans+=in[i];
66                 for(v=i;vis[v]!=i && id[v]==-1 && v!=rt;v=pre[v])
67                     vis[v]=i;
68                 if(v!=rt && id[v]==-1)
69                 {
70                     for(u=pre[v];u!=v;u=pre[u])
71                         id[u]=tn;
72                     id[v]=tn++;
73                 }
74             }
75             if(!tn)
76                 break;
77             for(i=0;i<n;++i)
78             {
79                 if(id[i]==-1)
80                     id[i]=tn++;
81             }
82             for(i=0;i<ed.size();++i)
83             {
84                 v=ed[i].b;
85                 ed[i].a=id[ed[i].a];
86                 ed[i].b=id[ed[i].b];
87                 if(ed[i].a!=ed[i].b)
88                     ed[i].c-=in[v];
89             }
90             n=tn;
91             rt=id[rt];
92         }
93     }
94     if(ans>=2*sum)
95         puts("impossible");
96     else
97         printf("%d\n",ans-sum,j-on);
98     puts("");
99 }

```

```

93|     return 0;
94| }

```

4.9 Covering problems

1| 最大团以及相关知识

2|

3| 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。

4|

5| 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。

6|

7| 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。

8|

9| 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 \rightarrow 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 $\langle u, v \rangle$ 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 $\langle u, v \rangle$ 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。

10|

11| 性质：

12| 最大独立集 + 最小覆盖集 = V

13| 最大团 = 补图的最大独立集

14| 最小覆盖集 = 最大匹配

15|

16| minimum cover:

17| vertex cover vertex bipartite graph = maximum cardinality bipartite matching

18| 找完最大二分匹配後，有三種情况要分別處理：

19| 甲、 X 側未匹配點的交錯樹們。

20| 乙、 Y 側未匹配點的交錯樹們。

21| 丙、層層疊疊的交錯環們（包含單獨的匹配邊）。

22| 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。

23| 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。

24| 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。

25|

26| vertex cover edge

27|

28| edge cover vertex

29| 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。

30|

31| edge cover edge

32|

33| path cover vertex

34| general graph: NP-H

35| tree: DP

36| DAG: 将每个节点拆分为入点和出点, ans = 节点数 - 匹配数

37|

38| path cover edge

39| minimize the count of euler path (greedy is ok?)

40|

41| cycle cover vertex

42| general: NP-H

43| weighted: do like path cover vertex, with KM algorithm

44|

45| cycle cover edge

46| NP-H

4.10 Difference constraints

```

1| for a - b <= c
2|     add(b,a,c);
3|

```

4| 最短路得最远解

5| 最长路得最近解

6| // 根据情况反转边? (反转方向及边权)

7|

8| 全 0 点得普通解

4.11 Dinitz's algorithm

```

1| #include<stdio>
2| #include<algorithm>
3| #include<cstring>

```

```

4|
5| #define MAXX 111
6| #define MAXM (MAXX*MAXX*4)
7| #define inf 0x3f3f3f3f
8|
9| int n;
10| int w[MAXX],h[MAXX],q[MAXX];
11| int edge[MAXX],to[MAXM],cap[MAXM],nxt[MAXM],cnt;
12| int source,sink;
13|
14| inline void add(int a,int b,int c)
15| {
16|     nxt[cnt]=edge[a];
17|     edge[a]=cnt;
18|     to[cnt]=b;
19|     cap[cnt]=c;
20|     ++cnt;
21| }
22|
23| inline bool bfs()
24| {
25|     static int *qf,*qb;
26|     static int i;
27|     memset(h,-1,sizeof h);
28|     qf=qb=q;
29|     h[*qb++=source]=0;
30|     for(;qf!=qb;++qf)
31|         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],
44|             std::min(maxcap,cap[i])))
45|         {
46|             d=dfs(to[i],std::min(flow,cap[i]));
47|             cap[i]-=d;
48|             cap[i^1]+=d;
49|             flow-=d;
50|             if(!flow)
51|                 return maxcap;
52|         }
53|     return maxcap-flow;
54| }
55|
56| int nc,np,m,i,j,k;
57| int ans;
58|
59| int main()
60| {
61|     while(scanf("%d%d%d%d",&n,&np,&nc,&m)!=EOF)
62|     {
63|         cnt=0;
64|         memset(edge,-1,sizeof edge);
65|         while(m--)
66|         {
67|             while(getchar()!='(');
68|             scanf("%d",&i);
69|             while(getchar()!='(');
70|             scanf("%d",&j);
71|             while(getchar()!='(');
72|             scanf("%d",&k);
73|             if(i!=j)
74|             {
75|                 ++i;
76|                 ++j;
77|                 add(i,j,k);
78|                 add(j,i,0);
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 }

```

4.12 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:
15 计入度和出度之差
16 无向边任意定向
17 出入度之差为奇数则无解
18 然后构图:
19 原图有向边不变，容量 1 // 好像需要在新图中忽略有向边？
20 无向边按之前认定方向，容量 1
21 源点向所有度数为正的点连边，容量 abs(度数/2)
22 所有度数为负的点向汇点连边，容量 abs(度数/2)
23 两侧均满流则有解
24 相当于规约为可行流问题
25 注意连通性的 trick
26
27 终点到起点加一条有向边即可将 path 问题转为 circuit 问题
28
29
30
31 Feasible flow problem:
32 由超级源点出发的边全部满流则有解
33 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
34
35 对于每条边 <a->b capu,d>, 建边 <ss->b cap(u)>、<a->st cap(u)>、
   <a->b cap(d-u)>
36
37 Maximum flow: //好像也可以二分
38 //将流量还原至原图后，在残量网络上继续完成最大流
39 直接把 source 和 sink 设为原来的 st，此时输出的最大流即是答案
40 不需要删除或者调整 t->s 弧
41 Minimum flow: //好像也可以二分
42 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
43 判断可行流存在还是必须连原汇 -> 原源的边之后查看满流
44 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
45 tips:
46 合并流量、减少边数来加速
47
48
49
50 Minimum cost feasible flow problem:
51 TODO
52 看起来像是在上面那样跑费用流就行了……
53
54
55
56 Minimum weighted vertex cover edge for bipartite graph:
57 for all vertex in X:
58 edge < s->x cap(weight(x)) >
59 for all vertex in Y:
60 edge < y->t cap(weight(y)) >
61 for original edges
62 edge < x->y cap(inf) >
63
64 ans={maximum flow}={minimum cut}
65 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
66

```

```

67
68
69 Maximum weighted vertex independent set for bipartite graph:
70 ans=Sum 点权 -valueMinimum weighted vertex cover edge
71 解应该就是最小覆盖集的补图吧……
72
73
74
75 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
76
77
78
79 必取的方格到源/汇的边的容量 inf
80 相邻方格之间的边的容量为 {代价}*2
81 ans=sum{方格收益}-{最大流}
82
83
84
85 最小割的唯一性: // refer: 关键边。有向边起点为 s 集，终点为 t 集
86 从源和汇分别能够到的点集是所有点时，最小割唯一
87 也就是每一条增广路径都仅有一条边满流
88 注意查看的是实际的网络，不是残量网络
89
90 具体来说
91
92 void rr(int now)
93 {
94     done[now]=true;
95     ++cnt;
96     for(int i=edge[now];i!=-1;i=nxt[i])
97         if(cap[i] && !done[v])
98             rr(v);
99 }
100
101 void dfs(int now)
102 {
103     done[now]=true;
104     ++cnt;
105     for(int i=edge[now];i!=-1;i=nxt[i])
106         if(cap[i^1] && !done[v])
107             dfs(v);
108 }
109
110 memset(done,0,sizeof done);
111 cnt=0;
112 rr(source);
113 dfs(sink);
114 puts(cnt==n?"UNIQUE":"AMBIGUOUS");
115
116
117
118 Tips:
119 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；
120 两点间容量 inf 则可以设法化简为一个点；
121 点权始终要转化为边权；
122 不参与决策的边权设为 inf 来排除掉；
123 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）
124 按时间拆点（时间层……？）；

```

4.13 Hamiltonian circuit

```

1 //if every point connect with not less than [(N+1)/2] points
2 #include<cstdio>
3 #include<algorithm>
4 #include<cstring>
5
6 #define MAXX 177
7 #define MAX (MAXX*MAXX)
8
9 int edge[MAXX],nxt[MAX],to[MAX],cnt;
10
11 inline void add(int a,int b)
12 {
13     nxt[++cnt]=edge[a];
14     edge[a]=cnt;
15     to[cnt]=b;
16 }
17
18 bool done[MAXX];
19 int n,m,i,j,k;
20
21 inline int find(int a)
22 {
23     static int i;
24     for(i=edge[a];i;i=nxt[i])
25         if(!done[to[i]])
26         {
27             edge[a]=nxt[i];
28             return to[i];
29         }
30     return 0;

```

```

31 }
32
33 int a,b;
34 int next[MAXX],pre[MAXX];
35 bool mat[MAXX][MAXX];
36
37 int main()
38 {
39     while(scanf("%d%d",&n,&m)!=EOF)
40     {
41         for(i=1;i<=n;++i)
42             next[i]=done[i]=edge[i]=0;
43         memset(mat,0,sizeof mat);
44         cnt=0;
45         while(m--)
46         {
47             scanf("%d%d",&i,&j);
48             add(i,j);
49             add(j,i);
50             mat[i][j]=mat[j][i]=true;
51         }
52         a=1;
53         b=to[edge[a]];
54         cnt=2;
55         done[a]=done[b]=true;
56         next[a]=b;
57         while(cnt<n)
58         {
59             while(i=find(a))
60             {
61                 next[i]=a;
62                 done[a=i]=true;
63                 ++cnt;
64             }
65             while(i=find(b))
66             {
67                 next[b]=i;
68                 done[b=i]=true;
69                 ++cnt;
70             }
71             if(!mat[a][b])
72                 for(i=next[a];next[i]!=b;i=next[i])
73                     if(mat[a][next[i]] && mat[i][b])
74                     {
75                         for(j=next[i];j!=b;j=next[j])
76                             pre[next[j]]=j;
77                         for(j=b;j!=next[i];j=pre[j])
78                             next[j]=pre[j];
79                         std::swap(next[i],b);
80                         break;
81                     }
82             next[b]=a;
83             for(i=a;i!=b;i=next[i])
84                 if(find(i))
85                 {
86                     a=next[b=i];
87                     break;
88                 }
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 }

```

4.14 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",&nx,&p);
40     while(p--)
41     {
42         scanf("%d",&i,&j);
43         nxt[++cnt]=edge[i];
44         edge[i]=cnt;
45         to[cnt]=j;
46     }
47     memset(cx,-1,sizeof cx);
48     memset(cy,-1,sizeof cy);
49     while(true)
50     {
51         memset(px,0,sizeof(px));
52         memset(py,0,sizeof(py));
53         qf=qb=q;
54         flag=false;
55
56         for(i=1;i<=nx;++i)
57             if(cx[i]==-1)
58                 *qb++=i;
59         while(qf!=qb)
60             for(k=edge[i=*qf++];k;k=nxt[k])
61                 if(!py[j=to[k]])
62                 {
63                     py[j]=px[i]+1;
64                     if(cy[j]==-1)
65                         flag=true;
66                     else
67                     {
68                         px[cy[j]]=py[j]+1;
69                         *qb++=cy[j];
70                     }
71                 }
72         if(!flag)
73             break;
74         for(i=1;i<=nx;++i)
75             if(cx[i]==-1 && ag(i))
76                 ++ans;
77     }
78     printf("%d\n",ans);
79     return 0;
80 }

```

4.15 Improved Shortest Augmenting Path Algorithm

```

1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4
5 #define MAXX 5111
6 #define MAXM (30111*4)
7 #define inf 0x3f3f3f3f3f3f3f3f
8
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,N,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%d",&i,&j,&ans);
88         add(i,j,ans);
89         add(j,i,ans);
90     }
91     printf("%lld\n",go());
92     return 0;
93 }

```

4.16 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 }

```

4.17 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-1].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::i]=ta;
83                     dp[j]=e[i::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 }

```

4.18 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 }
16 inline 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 }
29 void 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 }

```

```

55 int 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 //bupt
70
71 //算法：求二分图最佳匹配km n复杂度^3
72 int dfs(int u)//匈牙利求增广路
73 {
74     int v;
75     sx[u]=1;
76     for (v=1; v<=n; v++)
77         if (!sy[v] && lx[u]+ly[v]==map[u][v])
78         {
79             sy[v]=1;
80             if (match[v]==-1 || dfs(match[v]))
81             {
82                 match[v]=u;
83                 return 1;
84             }
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;
110             int dx=Inf;//若找不到增广路，则修改顶标~~
111             for (i=1; i<=n; i++)
112             {
113                 if (sx[i])
114                     for (j=1; j<=n; j++)
115                         if(!sy[j] && dx>lx[i]+ly[j]-map[i][j])
116                             dx=lx[i]+ly[j]-map[i][j];
117             }
118             for (i=1; i<=n; i++)
119             {
120                 if (sx[i])
121                     lx[i]-=dx;
122                 if (sy[i])
123                     ly[i]+=dx;
124             }
125         }
126     }
127     int sum=0;
128     for (i=1; i<=n; i++)
129         sum+=map[match[i]][i];
130     return sum;
131 }

```

4.19 LCA - DA

```

1 int edge[MAXX],nxt[MAXX<<1],to[MAXX<<1],cnt;
2 int pre[MAXX][N],dg[MAXX];
3
4 inline void add(int j,int k)
5 {
6     nxt[++cnt]=edge[j];
7     edge[j]=cnt;
8     to[cnt]=k;
9 }
10
11 void rr(int now,int fa)
12 {
13     dg[now]=dg[fa]+1;
14     for(int i=edge[now];i;i=nxt[i])
15         if(to[i]!=fa)

```

```

16         {
17             static int j;
18             j=1;
19             for(pre[to[i]][0]=now;j<N;++j)
20                 pre[to[i]][j]=pre[pre[to[i]][j-1]][j-1];
21             rr(to[i],now);
22         }
23 }
24
25 inline int lca(int a,int b)
26 {
27     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>=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 }

```

4.20 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(
21         bb),id(idd){}
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]
40         ).begin());it!=q[now].end();++it)
41         if(done[it->first])
42             if(it->second>0)
43                 to[find(it->first)].push_back(node(now,it->
44                     first,it->second));
45             else
46                 to[find(it->first)].push_back(node(it->first,
47                     now,-it->second));
48     for(std::list<std::pair<int,int> >::const_iterator it(edge[
49         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
54         !=to[now].end();++it)
55     {
56         find(it->a);
57         find(it->b);
58         ans[0][it->id]=std::min(min[it->b],min[it->a]);
59         ans[1][it->id]=std::max(max[it->a],max[it->b]);
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",&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 }

```

4.21 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-
66                     node[j].x)*(node[i].x-node[j].x)+(node[i].
67                     y-node[j].y)*(node[i].y-node[j].y));
68                 map[j][i].c=map[i][j].c=fabs(node[i].z-node[j].
69                     z);
70             }
71         a=0,b=mst(a);
72         while (fabs(b-a)>1e-8)
73         {
74             a=b;
75             b=mst(a);
76         }
77         printf("%.3lf\n",b);
78     }
79     return 0;
80 }

```

4.22 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>=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         {
91             if(S[i])
92             {
93                 s[i]=1<<(cf+cs);
94                 d[s[i]][i]=0;
95                 ++cs;
96             }
97             nn=1<<(cf+cs);
98             scanf("%d",&m);
99             while(m—)
100             {
101                 scanf("%d%d%d",&i,&j,&k);
102                 add(i,j,k);
103                 add(j,i,k);
104             }
105             for(y=1;y<nn;++y)
106             {
107                 for(x=1;x<=n;++x)
108                 {
109                     if(s[x] && !(s[x]&y))
110                         continue;
111                     for(i=(y-1)&y;i=(i-1)&y)
112                         d[y][x]=std::min(d[y][x],d[i|s[x]][x]+d[y^i|s[x]][x]);
113                     if(d[y][x]!=inf)
114                         q.push(node(x,y,d[y][x]));
115                 }
116                 while(!q.empty())
117                 {
118                     now=q.top();
119                     q.pop();
120                     if(now.dist!=now.get())
121                         continue;
122                     static int x,y,a,b;
123                     x=now.a;
124                     y=now.b;
125                     for(i=edge[x];i=nxt[i])
126                     {
127                         a=to[i];
128                         b=y|s[a];
129                         if(d[b][a]>now.get()+wg[i])
130                         {
131                             d[b][a]=now.get()+wg[i];
132                             if(b==y)
133                                 q.push(node(a,b,d[b][a]));
134                         }
135                     }
136                 }
137             }
138             for(j=0;j<nn;++j)
139                 dp[j]=*std::min_element(d[j]+1,d[j]+1+n);
140             cnt=cst=0;
141             for(i=1;i<nn;++i)
142                 if(check(i))
143                 {
144                     for(j=(i-1)&i;j;j=(j-1)&i)
145                         if(check(j) && check(i^j))
146                             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         printf("%d%d\n",ans+cnt,cst);
153     }
154     return 0;
155 }
156 }

```

4.23 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 {
31     static int now,i;
32     memset(dist,0x3f,sizeof dist);
33     dist[source]=0;
34     pre[source]=-1;
35     q.push(source);
36     in[source]=true;
37     while(!q.empty())
38     {
39         in[now=q.front()]=false;
40         q.pop();
41         for(i=edge[now];i!=-1;i=nxt[i])
42             if(cap[i] && dist[v]>dist[now]+cst[i])
43             {
44                 dist[v]=dist[now]+cst[i];
45                 pre[v]=i;
46                 if(!in[v])
47                 {
48                     q.push(v);
49                     in[v]=true;
50                 }
51             }
52     }
53     return dist[sink]!=inf;
54 }
55
56 inline int mcmf(int &flow)
57 {
58     static int ans,i;
59     flow=ans=0;
60     while(go())
61     {
62         static int min;
63         min=inf;
64         for(i=pre[sink];i!=-1;i=pre[to[i^1]])
65             min=std::min(min,cap[i]);
66         flow+=min;
67         ans+=min*dist[sink];
68         for(i=pre[sink];i!=-1;i=pre[to[i^1]])
69         {
70             cap[i]-=min;
71             cap[i^1]+=min;
72         }
73     }
74     return ans;
75 }

```

4.24 Second-best MST

```

1 #include<cstdio>

```

```

2 #include<cstring>
3 #include<algorithm>
4
5 #define MAXN 511
6 #define MAXM 250011
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[MAXN];
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);
55     for(i=0;i<m;++i)
56         scanf("%d%d%d",&ed[i].a,&ed[i].b,&ed[i].c);
57     std::sort(ed,ed+m);
58     for(i=0;i<m;++i)
59         if(find(ed[i].a)!=find(ed[i].b))
60         {
61             j+=ed[i].c;
62             ++k;
63             set[find(ed[i].a)]=find(ed[i].b);
64             ed[i].in=true;
65             add(ed[i].a,ed[i].b,ed[i].c);
66             add(ed[i].b,ed[i].a,ed[i].c);
67         }
68     if(k+1!=n)
69         puts("Cost: -1\nCost: -1");
70     else
71     {
72         printf("Cost: %d\n",j);
73         if(m==n-1)
74         {
75             puts("Cost: -1");
76             return 0;
77         }
78         ans=0x3f3f3f3f;
79         memset(map,0x3f,sizeof map);
80         for(i=1;i<=n;++i)
81             map[i][i]=0;
82         dfs(1,0);
83         for(i=0;i<m;++i)
84             if(!ed[i].in)
85                 ans=std::min(ans,j+ed[i].c-map[ed[i].a][ed[i].b]);
86         printf("Cost: %d\n",ans);
87     }
88     return 0;
89 }

```

4.25 Spanning tree

```

1 Minimum Bottleneck Spanning Tree:
2 Kruscal

```

```

3 All-pairs vertexes' Minimum Bottleneck Path:
4 DP in the Kruscal's MST
5  $O(n^2)*O(1)$ 
6
7 Minimum Diameter Spanning Tree:
8 Kariv-Hakimi Algorithm
9
10 Directed MST:-
11 ChuLiu/Edmonds' Algorithm
12
13 Second-best MST:
14 get All-pairs vertexes' Minimum Bottleneck Path, then enumerate
15     all no-tree-edges to replace the longest edge between two
16     vertexes to get a worse MST
17
18 Degree-constrained MST:
19 remove the vertex from the whole graph, then add edges to
20     increase degrees and connect different connected
21     components together (  $O(m\log m + n)$  with kruscal )
22 if we can't connect all connected components together, there
23     exists no any spanning tree
24 next step is add edges to root vertex greedily, increase
25     degrees, and decrease our answer (  $O(k*n)$  )
26 need all vertexes' minimum bottleneck path to root vertex
27
28 Minimum Ratio Spanning Tree:
29 Binary search
30
31 Manhattan MST:
32 combining line sweep with divide-and-conquer algorithm
33
34 Minimum Steiner Tree:
35 the MST contain all k vertexes
36 bit-mask with dijkstra  $O( (1<k)* ( \{dijkstra\} ) )$ 
37 then run a bit-mask DP(  $O( n*(1<k) )$  )
38
39 Count Spanning Trees:
40 TODO
41 Kirchhoff's theorem
42
43 k-best MST:
44 do like second-best MST for k times

```

4.26 Stable Marriage

```

1 //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
2
3 while(!g.empty()) // 预备匹配队列
4 {
5     if(dfn[edge[g.front()].front()]!=-1)
6         dfn[edge[g.front()].front()]=g.front(); // 如果目前还没尝
7         试匹配过的对象没有被任何别的对象占据
8     else
9     {
10         for(it=edge[edge[g.front()].front()].begin();it!=edge[
11             edge[g.front()].front()].end();++it)
12             if(*it==dfn[edge[g.front()].front()] || *it==g.
13                 front()) //如果被匹配对象更喜欢正在被匹配的人或现在准
14                 备匹配的对象
15                 break;
16         if(*it==g.front()) //如果更喜欢新的
17         {
18             g.push_back(dfn[edge[g.front()].front()]);
19             dfn[edge[g.front()].front()]=g.front();
20         }
21         else
22             g.push_back(g.front()); //否则放到队尾，重新等待匹配
23     }
24     edge[g.front()].pop_front(); //每组匹配最多只考虑一次
25     g.pop_front();
26 }

```

4.27 Stoer-Wagner Algorithm

```

1 #include<cstdio>
2 #include<cstring>
3
4 const int maxn=510;
5
6 int map[maxn][maxn];
7 int n;
8
9 void contract(int x,int y)//合并两个点
10 {
11     int i,j;
12     for (i=0; i<n; i++)
13         if (i!=x)
14         {
15             map[x][i]+=map[y][i];
16             map[i][x]+=map[i][y];
17         }
18     for (i=y+1; i<n; i++)

```

```

19     for (j=0; j<n; j++)
20     {
21         map[i-1][j]=map[i][j];
22         map[j][i-1]=map[j][i];
23     }
24     n--;
25 }
26
27 int w[maxn],c[maxn];
28 int sx,tx;
29
30 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
31 {
32     static int i,j,k,t;
33     memset(c,0,sizeof(c));
34     c[0]=1;
35     for (i=0; i<n; i++)
36         w[i]=map[0][i];
37     for (i=1; i+1<n; i++)
38     {
39         t=k=-1;
40         for (j=0; j<n; j++)
41             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 }

```

4.28 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());
7         it!=edge[now].end();++it)
8         if(dfn[*it]==-1)
9         {
10             dfs(*it);
11             low[now]=std::min(low[now],low[*it]);
12         }
13         else
14             if(sc[*it]==-1)
15                 low[now]=std::min(low[now],dfn[*it]);
16     if(dfn[now]==low[now])
17     {
18         while(sc[now]==-1)
19         {
20             sc[st.top()]=p;
21             st.pop();
22         }
23         ++p;
24     }
25 }

```

4.29 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,pil;
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*pil;
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     pil+=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)&& !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     pil+=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=pil=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,
3 362880, 3628800, 39916800};
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     }
37 }

```

5.2 Discrete logarithms - BSGS

```

1 //The running time of BSGS and the space complexity is  $O(\sqrt{n})$ 
2 //Pollard's rho algorithm for logarithms' running time is
   approximately  $O(\sqrt{p})$  where p is n's largest prime factor.
3 #include<cstdio>
4 #include<cmath>
5 #include<cstring>
6
7 struct Hash // std::map is bad. clear() 时会付出巨大的代价
8 {
9     static const int mod=1000003; // prime is good
10    static const int MAXX=47111; // bigger than  $\sqrt{c}$ 
11    int hd[mod],nxt[MAXX],cnt;
12    long long v[MAXX],k[MAXX]; //  $a^k \equiv v \pmod{c}$ 
13    inline void init()
14    {
15        memset(hd,0,sizeof hd);
16        cnt=0;
17    }
18    inline long long find(long long v)
19    {
20        static int now;
21        for(now=hd[v%mod];now;now=nxt[now])
22            if(this->v[now]==v)
23                return k[now];
24        return -1ll;
25    }
26    inline void insert(long long k,long long v)
27    {
28        if(find(v)!=-1ll)
29            return;
30        nxt[++cnt]=hd[v%mod];
31        hd[v%mod]=cnt;
32        this->v[cnt]=v;
33        this->k[cnt]=k;
34    }
35 }hash;
36
37 long long gcd(long long a,long long b)
38 {
39     return b?gcd(b,a%b):a;
40 }
41
42 long long exgcd(long long a,long long b,long long &x,long long &y)
43 {
44     if(b)
45     {
46         long long re(exgcd(b,a%b,x,y)),tmp(x);
47         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) //
    $a^x \equiv b \pmod{c}$ 
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*a*m%c;
97 | }
98 | return -1ll;
99 | }
100 |
101 | long long k,p,n;
102 |
103 | int main()
104 | {
105 |     while(scanf("%lld_%lld_%lld",&k,&p,&n)!=EOF)
106 |     {
107 |         if(n>p || (k=bsgs(k,n,p))!=-1ll)
108 |             puts("Orz,I cant find D!");
109 |         else
110 |             printf("%lld\n",k);
111 |     }
112 |     return 0;
113 | }

```

5.3 Divisor function

- 1 $n = p_1^{a_1} \times p_2^{a_2} \times \dots \times p_s^{a_s}$
- 2 sum of positive divisors function
- 3
$$\sigma(n) = \prod_{j=1}^s \frac{p_j^{a_j+1}-1}{p_j-1}$$
- 4 number of positive divisors function
- 5
$$\tau(n) = \prod_{j=1}^s (a_j + 1)$$

5.4 Extended Euclidean Algorithm

```

1 | //返回ax+by=gcd(a,b)的一组解
2 | long long ex_gcd(long long a,long long b,long long &x,long long &y)
3 | {
4 |     if (b)
5 |     {
6 |         long long ret = ex_gcd(b,a%b,x,y),tmp = x;
7 |         x = y;
8 |         y = tmp-(a/b)*y;
9 |         return ret;
10 |    }
11 |    else
12 |    {
13 |        x = 1;
14 |        y = 0;
15 |        return a;
16 |    }
17 | }

```

5.5 Fast Fourier Transform

```

1 | #include<cstdio>
2 | #include<cstring>
3 | #include<complex>
4 | #include<vector>
5 | #include<algorithm>
6 |
7 | #define MAXX 100111
8 | #define MAXN (MAXX<2)
9 |
10 | int T;
11 | int n,i,j,k;
12 |
13 | typedef std::complex<long double> com;
14 | 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].
92 |                 real()+0.5f) or nearbyint(x[i].real())
93 |             x.resize(2*a[n-1]); // result here
94 |         }
95 |     }
96 |     return 0;
97 | }

```

5.6 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 | void dfs(int v)
29 | {
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
105 inline void ge(int a[N][N],int m,int n) // m*n
106 {
107     static int i,j,k,l,b,c;
108     for(i=j=0;i<m && j<n;++j)
109     {
110         for(k=i;k<m;++k)
111             if(a[k][j])
112                 break;
113         if(k==m)
114             continue;
115         for(l=0;l<n;++l)
116             std::swap(a[i][l],a[k][l]);
117         for(k=0;k<m;++k)
118             if(k!=i && a[k][j])
119             {
120                 b=a[k][j];
121                 c=a[i][j];
122                 for(l=0;l<n;++l)
123                     a[k][l]=((a[k][l]*c-a[i][l]*b)%7+7)%7;
124             }
125         ++i;
126     }

```

```

127     for(j=i;j<m;++j)
128         if(a[j][n])
129             break;
130     if(j<m)
131     {
132         puts("Inconsistent_data.");
133         return;
134     }
135     if(i<n)
136         puts("Multiple_solutions.");
137     else
138     {
139         memset(ans,0,sizeof(ans));
140         for(i=n-1;i>=0;--i)
141         {
142             k=a[i][n];
143             for(j=i+1;j<n;++j)
144                 k=((k-a[i][j]*ans[j])%7+7)%7;
145             while(k%a[i][i])
146                 k+=7;
147             ans[i]=(k/a[i][i])%7;
148         }
149         for(i=0;i<n;++i)
150             printf("%d%c",ans[i],i+1==n?'\\n':' ');
151     }
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 power(long long x,long long y,int mod)
9 {
10     long long ret=1;
11     for (long long a=x%mod; y;y>>=1,a=a*a%mod)
12         if (y&1)
13             ret=ret*a%mod;
14     return ret;
15 }
16
17 inline int getInv(int x,int mod)//mod 为素数
18 {
19     return power(x,mod-2);
20 }

```

5.8 Linear programming

```

1 #include<cstdio>
2 #include<cstring>
3 #include<cmath>
4 #include<algorithm>
5
6 #define MAXN 33
7 #define MAXM 33
8 #define eps 1e-8
9
10 double a[MAXN][MAXM],b[MAXN],c[MAXM];
11 double x[MAXM],d[MAXN][MAXM];
12 int ix[MAXN+MAXM];
13 double ans;
14 int n,m;
15 int i,j,k,r,s;
16 double D;
17
18 inline bool simplex()
19 {
20     r=n;
21     s=m++;
22     for(i=0;i<n+m;++i)
23         ix[i]=i;
24     memset(d,0,sizeof d);
25     for(i=0;i<n;++i)
26     {
27         for(j=0;j+1<m;++j)
28             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()
78 {
79     while(scanf("%d%d",&m,&n)!=EOF)
80     {
81         for(i=0;i<=m;++i)
82             scanf("%lf",c+i); // max{ sum{c[i]*x[i]} }
83         for(i=0;i<=n;++i)
84         {
85             for(j=0;j<=m;++j)
86                 scanf("%lf",a[i+j]); // sum{ a[i]*x[i] } <= b
87             scanf("%lf",b+i);
88             b[i]*=n;
89         }
90         simplex();
91         printf("Nasa_u can_u spend_u %.0lf_u taka.\n",ceil(ans));
92     }
93     return 0;
94 }

```

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

```

1 #include <cstdio>
2 /*
3 Lucas 快速求解C(n,m)%p
4 */
5 void gcd(int n,int k,int &x,int &y)
6 {
7     if(k)
8     {
9         gcd(k,n%k,x,y);
10        int t=x;
11        x=y;
12        y=t-(n/k)*y;
13        return;
14    }
15    x=1;
16    y=0;
17 }
18
19 int CmodP(int n,int k,int p)
20 {
21     if(k>n)
22         return 0;
23     int a,b,flag=0,x,y;
24     a=b=1;
25     for(int i=1;i<=k;i++)
26     {
27         x=n-i+1;
28         y=i;
29         while(x%p==0)
30         {
31             x/=p;
32             ++flag;
33         }
34         while(y%p==0)
35         {
36             y/=p;
37             --flag;
38         }
39         x%=p;
40         y%=p;
41
42         a*=x;
43         b*=y;
44
45         b%=p;
46         a%=p;
47     }
48     if(flag)
49         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 struct Matrix
2 {
3     const int N(52);
4     int a[N][N];
5     inline Matrix operator*(const Matrix &b)const
6     {
7         static Matrix res;
8         static int i,j,k;
9         for(i=0;i<N;++i)
10             for(j=0;j<N;++j)
11             {
12                 res.a[i][j]=0;
13                 for(k=0;k<N;++k)
14                     res.a[i][j]+=a[i][k]*b.a[k][j];
15             }
16         return res;
17     }
18     inline Matrix operator^(int y)const
19     {
20         static Matrix res,x;
21         static int i,j;
22         for(i=0;i<N;++i)
23         {
24             for(j=0;j<N;++j)
25             {
26                 res.a[i][j]=0;
27                 x.a[i][j]=a[i][j];
28             }
29             res.a[i][i]=1;
30         }
31         for(;y>=1,x=x*x)
32             if(y&1)
33                 res=res*x;
34         return res;
35     }
36 };
37
38 Fibonacci Matrix
39 1 1
40 1 0

```

5.12 Miller-Rabin Algorithm

```

1 inline unsigned long long multi_mod(const unsigned long long &a,
2     ,unsigned long long b,const unsigned long long &n)
3 {
4     unsigned long long exp(a%n),tmp(0);
5     while(b)
6     {
7         if(b&1)
8         {
9             tmp+=exp;
10            if(tmp>n)
11                tmp-=n;
12        }
13        exp<<=1;
14        if(exp>n)
15            exp-=n;
16        b>>=1;
17    }
18    return tmp;
19 }
20 inline unsigned long long exp_mod(unsigned long long a,unsigned
21     long long b,const unsigned long long &c)
22 {

```

```

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

```

5.13 Multiset

```

1 Permutation:
2 MultiSet S={1 m,4 s,4 i,2 p}
3  $P(S) = \frac{(1+4+4+2)!}{1!4!4!2!}$ 
4
5 Combination:
6 MultiSet S={∞a1,∞a2,...∞ak}
7  $\binom{S}{r} = \frac{(r+k-1)!}{r!(k-1)!} = \binom{r+k-1}{r}$ 
8
9 if(r>min{count(element[i])})
10     you have to resolve this problem with inclusion-exclusion
11     principle.
12
13 MS T={3 a,4 b,5 c}
14 MS  $T_s = \{\infty a, \infty b, \infty c\}$ 
15  $A1 = \{\binom{T_s}{10} | \text{count}(a) > 3\} // \binom{8}{6}$ 
16  $A2 = \{\binom{T_s}{10} | \text{count}(b) > 4\} // \binom{6}{5}$ 
17  $A3 = \{\binom{T_s}{10} | \text{count}(c) > 5\} // \binom{6}{4}$ 
18  $\binom{T}{10} = \binom{T_s}{10} - (|A1| + |A2| + |A3|) + (|A1 \cap A2| + |A1 \cap A3| + |A2 \cap A3|) - |A1 \cap A2 \cap A3|$ 
19 ans=C(10,12)+C(1,3)+C(0,2)+0+0=6

```

5.14 Pell's equation

```

1 /*
2 find the (x,y)pair that  $x^2 - n \times y^2 = 1$ 
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
7 slow in some situation )
8 or we can enumerate the continued fraction of  $\sqrt{n}$ , as  $\frac{x}{y}$ , it will
9 be much more faster
10
11 other solution pairs' matrix:
12
13  $\begin{matrix} x1 & n \times y1 \\ y1 & x1 \end{matrix}$ 
14 k-th solution is {matrix}k
15 */
16
17 import java.util.*;
18 import java.math.*;
19
20 public class Main
21 {
22     static BigInteger p,q,p1,p2,p3,q1,q2,q3,a1,a2,a0,h1,h2,g1,
23         g2,n0;
24     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)
39             )),equals(BigInteger.ONE))
40             return ;
41         a1=a2;
42         g1=g2;
43         h1=h2;
44         p1=p2;
45         p2=p;
46         q1=q2;
47         q2=q;
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 }

```

5.15 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,
10 unsigned long long b,const unsigned long long &n)
11 {
12     unsigned long long exp(a%n),tmp(0);
13     while(b)
14     {
15         if(b&1)
16         {
17             tmp+=exp;
18             if(tmp>n)
19                 tmp-=n;
20         }
21         exp<<=1;
22         if(exp>n)
23             exp-=n;
24         b>>=1;
25     }
26     return tmp;
27 }
28 inline unsigned long long exp_mod(unsigned long long a,unsigned
29 long long b,const unsigned long long &c)
30 {
31     unsigned long long tmp(1);
32     while(b)
33     {
34         if(b&1)
35             tmp=multi_mod(tmp,a,c);
36         a=multi_mod(a,a,c);
37         b>>=1;
38     }
39     return tmp;
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
72 unsigned long long &b)
73 {
74     return b?gcd(b,a%b):a;
75 }
76 inline unsigned long long pollar_rho(const unsigned long long n
77 ,const unsigned long long &c)
78 {
79     unsigned long long x(rand()%(n-1)+1),y,d,i(1),k(2);
80     y=x;
81     while(true)
82     {
83         ++i;
84         x=(multi_mod(x,x,n)+c)%n;
85         d=gcd((x-y+n)%n,n);
86         if(d>1 && d<n)
87             return d;
88         if(x==y)
89             return n;
90         if(i==k)
91         {
92             k<<=1;
93             y=x;
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 }

```

5.16 Prime

```

1 #include<vector>
2
3 std::vector<int> prm;
4 bool flag[MAXX];
5
6 int main()
7 {
8     prm.reserve(MAXX); // pi(x)=x/ln(x);

```

```

9|   for(i=2;i<MAXX;++i)
10|   {
11|       if(!flag[i])
12|           prm.push_back(i);
13|       for(j=0;j<prm.size() && i*prm[j]<MAXX;++j)
14|       {
15|           flag[i*prm[j]]=true;
16|           if(i%prm[j]==0)
17|               break;
18|       }
19|   }
20|   return 0;
21| }

```

5.17 Reduced Residue System

```

1| Euler's totient function:
2|
3| 对正整数 n, 欧拉函数  $\varphi$  是少于或等于 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。
4|  $\varphi(2)=1$  (唯一和 1 互质的数就是 1 本身)。
5| 若 m,n 互质,  $\varphi(m \times n) = \varphi(m) \times \varphi(n)$ 。
6| 对于 n 来说, 所有这样的数的和为  $\frac{n \times \varphi(n)}{2}$ 。
7|
8| inline long long phi(int n)
9| {
10|     static int i;
11|     static int re;
12|     re=n;
13|     for(i=0;prm[i]*prm[i]<=n;++i)
14|         if(n%prm[i]==0)
15|         {
16|             re-=re/prm[i];
17|             do
18|                 n/=prm[i];
19|             while(n%prm[i]==0);
20|         }
21|     if(n!=1)
22|         re-=re/n;
23|     return re;
24| }
25|
26| inline void Euler()
27| {
28|     static int i,j;
29|     phi[1]=1;
30|     for(i=2;i<MAXX;++i)
31|         if(!phi[i])
32|             for(j=i;j<MAXX;j+=i)
33|             {
34|                 if(!phi[j])
35|                     phi[j]=j;
36|                 phi[j]=phi[j]/i*(i-1);
37|             }
38| }
39|
40| Multiplicative order:
41|
42| the multiplicative order of a modulo n is the smallest positive
   integer k with
43|  $a^k \equiv 1 \pmod{n}$ 
44|
45| 对 m 的简化剩余系中的所有 x, ord(x) 都一定是  $\varphi(m)$  的一个约数 (aka.
   Euler's totient theorem)
46|
47| 求:
48| method 1、根据定义, 对  $\varphi(m)$  分解素因子之后暴力枚举所有  $\varphi(m)$  的约数, 找到4
   最小的一个 d, 满足  $x^d \equiv 1 \pmod{m}$ ;
49| method 2
50| inline long long ord(long long x, long long m)
51| {
52|     static long long ans;
53|     static int i,j;
54|     ans=phi(m);
55|     for(i=0;i<fac.size();++i)
56|         for(j=0;j<fac[i].second && pow(x,ans/fac[i].first,m)==1;
57|             ++j)
58|             ans/=fac[i].first;
59|     return ans;
60| }
61|
62| Primitive root:
63|
64| 若 ord(x)== $\varphi(m)$ , 则 x 为 m 的一个原根
65| 因此只需检查所有  $x^d$  {d 为  $\varphi(m)$  的约数} 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  $\varphi(m)$  的时候, x 是 m 的一个原根
66|
67| 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……?
68|
69| 当 m 存在原根时, 原根数目为  $\varphi(\varphi(m))$ 
70|

```

```

71| 求:
72| 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],  $i^{\frac{\varphi(m)}{p[j]}} \not\equiv 1$ 
   (mod m), 那么 i 为 m 的一个原根。也就是说, ord(i)== $\varphi(m)$ 。
73| 最小原根通常极小。
74|
75| Carmichael function:
76|
77|  $\lambda(n)$  is defined as the smallest positive integer m such that
78|  $a^m \equiv 1 \pmod{n}$  { forall a!=1 && gcd(a,n)==1 }
79| 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)}
80|
81| if n=p[0]a[0] × p[1]a[1] × ... × p[m-1]a[m-1]
82| then  $\lambda(n) = \text{lcm}(\lambda(p[0]^{a[0]}), \lambda(p[1]^{a[1]}), \dots, \lambda(p[m-1]^{a[m-1]}))$ ;
83|
84| if n=2c × p[0]a[0] × p[1]a[1] × ... × p[m-1]a[m-1]
85| then  $\lambda(n) = \text{lcm}(2^c, \varphi(p[0]^{a[0]}), \varphi(p[1]^{a[1]}), \dots, \varphi(p[m-1]^{a[m-1]}))$ ;
86| { c=0 if a<2; c=1 if a==2; c=a-2 if a>3; }
87|
88| Carmichael's theorem:
89| if gcd(a,n)==1
90| then  $\lambda(n) \equiv 1 \pmod{n}$ 
91|

```

5.18 Simpson's rule

```

1| // thx for mzry
2| inline double f(double)
3| {
4|     /*
5|     define the function
6|     */
7| }
8|
9| inline double simp(double l, double r)
10| {
11|     double h = (r-l)/2.0;
12|     return h*(f(l)+4*f((l+r)/2.0)+f(r))/3.0;
13| }
14|
15| inline double rsimp(double l, double r) // call here
16| {
17|     double mid = (l+r)/2.0;
18|     if(fabs((simp(l,r)-simp(l,mid)-simp(mid,r)))/15 < eps)
19|         return simp(l,r);
20|     else
21|         return rsimp(l,mid)+rsimp(mid,r);
22| }

```

5.19 System of linear congruences

```

1| // minimal val that for all (m,a), val%m == a
2| #include<cstdio>
3|
4| #define MAXX 11
5|
6| int T,t;
7| int m[MAXX],a[MAXX];
8| int n,i,j,k;
9| int x,y,c,d;
10| int lcm;
11|
12| int exgcd(int a,int b,int &x,int &y)
13| {
14|     if(b)
15|     {
16|         int re(exgcd(b,a%b,x,y)),tmp(x);
17|         x=y;
18|         y=tmp-(a/b)*y;
19|         return re;
20|     }
21|     x=1;
22|     y=0;
23|     return a;
24| }
25|
26| int main()
27| {
28|     scanf("%d",&T);
29|     for(t=1;t<=T;++t)
30|     {
31|         scanf("%d",&n);
32|         lcm=1;
33|         for(i=0;i<n;++i)
34|         {
35|             scanf("%d",m+i);
36|             lcm*=m[i]/exgcd(lcm,m[i],x,y);
37|         }
38|         for(i=0;i<n;++i)
39|             scanf("%d",a+i);
40|         for(i=1;i<n;++i)
41|         {

```

```

42     c=a[i]-a[0];
43     d=exgcd(m[0],m[i],x,y);
44     if(c%d)
45         break;
46     y=m[i]/d;
47     c/=d;
48     x=(x*c%y+y)%y;
49     a[0]+=m[0]*x;
50     m[0]*=y;
51 }
52 printf("Case_%d: %d\n",t,i<n?-1:(a[0]?a[0]:lcm));
53 }
54 return 0;
55 }

```

6 String

6.1 Aho-Corasick Algorithm

```

1 //trie graph
2 #include<cstring>
3 #include<queue>
4
5 #define MAX 1000111
6 #define N 26
7
8 int nxt[MAX][N],fal[MAX],cnt;
9 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

```

```

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
128             like not necessary at all, I delete it in an
129             other solution
130             ++cnt[q->idx];
131     }
132 }
133 //可以考虑 dfs 一下,拉直 fal 指针来跳过无效的匹配
134 //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
135 //大量内容匹配并且需要记录关键字出现次数的时候,可以考虑记录每个节点被覆盖
    的次数,然后沿着 fal 指针构成的 DAG 往上传递覆盖次数

```

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
        ++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;
46 }

```

6.4 Morris-Pratt Algorithm

```

1 inline void make(char *buf,int *fal)
2 {
3     static int i,j;
4     fal[0]=-1;
5     for(i=1,j=-1;buf[i];++i)
6     {
7         while(j>=0 && buf[j+1]!=buf[i])
8             j=fal[j];
9         if(buf[j+1]==buf[i])
10             ++j;
11         fal[i]=j;
12     }
13 }
14
15 inline int match(char *p,char *t,int* fal)
16 {
17     static int i,j,re;
18     re=0;
19     for(i=0,j=-1;t[i];++i)
20     {
21         while(j>=0 && p[j+1]!=t[i])
22             j=fal[j];
23         if(p[j+1]==t[i])
24             ++j;
25         if(!p[j+1])
26         {
27             ++re;
28             j=fal[j];
29         }
30     }
31     return re;
32 }
33 }

```

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)

```

```

        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 for(i=1;i<n;++i)
102     sptb[0][i]=i;
103 for(i=1;i<=lg[n];++i) //sparse table RMQ
104 {
105     k=n+1-(1<<i);
106     for(j=1;j<=k;++j)
107     {
108         a=sptb[i-1][j];
109         b=sptb[i-1][j+(1<<(i-1))];
110         sptb[i][j]=lcpa[a]<lcpa[b]?a:b;
111     }
112 }
113 }
114
115 inline int ask(int l,int r)
116 {
117     a=lg[r-l+1];
118     r-=(1<<a)-1;
119     l=sptb[a][l];
120     r=sptb[a][r];
121     return lcpa[l]<lcpa[r]?l:r;
122 }
123
124 inline int lcp(int l,int r) // 字符串上 [l,r] 区间的 rmq
125 {
126     l=rk[l];
127     r=rk[r];
128     if(l>r)
129         std::swap(l,r);
130     return lcpa[ask(l+1,r)];
131 }
132 }

```

6.7 Suffix Array - Prefix-doubling Algorithm

```

1 int wx[maxn],wy[maxn],*x,*y,wss[maxn],wv[maxn];
2
3 bool cmp(int *r,int n,int a,int b,int l)
4 {
5     return a+l<n && b+l<n && r[a]==r[b]&&r[a+l]==r[b+l];
6 }
7 void da(int str[],int sa[],int rank[],int height[],int n,int m)
8 {
9     int *s = str;
10    int *x=wx,*y=wy,*t,p;
11    int i,j;
12    for(i=0; i<m; i++)
13        wss[i]=0;
14    for(i=0; i<n; i++)
15        wss[x[i]=s[i]]++;
16    for(i=1; i<m; i++)
17        wss[i]+=wss[i-1];
18    for(i=n-1; i>=0; i--)
19        sa[--wss[x[i]]]=i;
20    for(j=1,p=1; p<n && j<n; j*=2,m=p)
21    {
22        for(i=n-j,p=0; i<n; i++)
23            y[p++]=i;
24        for(i=0; i<n; i++)
25            if(sa[i]-j>=0)
26                y[p++]=sa[i]-j;
27        for(i=0; i<n; i++)
28            wv[i]=x[y[i]];
29        for(i=0; i<m; i++)
30            wss[i]=0;
31        for(i=0; i<n; i++)
32            wss[wv[i]]++;
33        for(i=1; i<m; i++)
34            wss[i]+=wss[i-1];

```

```

35        for(i=n-1; i>=0; i--)
36            sa[--wss[wv[i]]]=y[i];
37        for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0; i<n; i++)
38            x[sa[i]]=cmp(y,n,sa[i-1],sa[i],j)?p-1:p++;
39    }
40    for(int i=0; i<n; i++)
41        rank[sa[i]]=i;
42    for(int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
43        if(rank[i]>0)
44            for(k?k--:0,j=sa[rank[i]-1]; i+k < n && j+k < n &&
45                str[i+k]==str[j+k]; ++k);

```

6.8 Suffix Automaton

```

1 /*
2 length(s) ∈ [ min(s), max(s) ] = [ val[fal[s]]+1, val[s] ]
3 */
4 #define MAXX 90111
5 #define MAXN (MAXX<<1)
6
7 int fal[MAXN],nxt[MAXN][26],val[MAXN],cnt,rt,last;
8
9 inline int neww(int v=0)
10 {
11     val[++cnt]=v;
12     fal[cnt]=0;
13     memset(nxt[cnt],0,sizeof nxt[0]);
14     return cnt;
15 }
16
17 inline void add(int w)
18 {
19     static int p,np,q,nq;
20     p=last;
21     np=neww(val[p]+1);
22     while(p && !nxt[p][w])
23     {
24         nxt[p][w]=np;
25         p=fal[p];
26     }
27     if(!p)
28         fal[np]=rt;
29     else
30     {
31         q=nxt[p][w];
32         if(val[p]+1==val[q])
33             fal[np]=q;
34         else
35         {
36             nq=neww(val[p]+1);
37             memcpy(nxt[nq],nxt[q],sizeof nxt[0]);
38             fal[nq]=fal[q];
39
40             fal[q]=fal[np]=nq;
41             while(p && nxt[p][w]==q)
42             {
43                 nxt[p][w]=nq;
44                 p=fal[p];
45             }
46         }
47     }
48     last=np;
49 }
50
51 int v[MAXN],the[MAXN];
52
53 inline void make(char *str)
54 {
55     cnt=0;
56     rt=last=neww();
57     static int i,len,now;
58     for(i=0;str[i];++i)
59         add(str[i]-'a');
60     len=i;
61     memset(v,0,sizeof v);
62     for(i=1; i<=cnt; ++i)
63         ++v[val[i]];
64     for(i=1; i<=len; ++i)
65         v[i]+=v[i-1];
66     for(i=1; i<=cnt; ++i)
67         the[v[val[i]]--]=i;
68     for(i=cnt; i--;)
69     {
70         now=the[i];
71         // topsort already
72     }
73 }
74 /*
75 sizeof right(s):
76     init:
77         for all np:
78             count[np]=1;
79     process:
80         for all status s:

```

```

81 |         count[fal[s]]+=count[s];
82 | */

```

7 Dynamic Programming

7.1 knapsack problem

```

1 | multiple-choice knapsack problem:
2 |
3 | for 所有的组k
4 |     for v=V..0
5 |         for 所有的属于组ik
6 |             f[v]=max{f[v],f[v-c[i]]+w[i]}

```

7.2 LCIS

```

1 | #include<cstdio>
2 | #include<cstring>
3 | #include<vector>
4 |
5 | #define MAXX 1111
6 |
7 | int T;
8 | int n,m,p,i,j,k;
9 | std::vector<int>the[2];
10 | int dp[MAXX],path[MAXX];
11 | int ans[MAXX];
12 |
13 | int main()
14 | {
15 |     the[0].reserve(MAXX);
16 |     the[1].reserve(MAXX);
17 |     {
18 |         scanf("%d",&n);
19 |         the[0].resize(n);
20 |         for(i=0;i<n;++i)
21 |             scanf("%d",&the[0][i]);
22 |         scanf("%d",&m);
23 |         the[1].resize(m);
24 |         for(i=0;i<m;++i)
25 |             scanf("%d",&the[1][i]);
26 |         memset(dp,0,sizeof dp);
27 |         for(i=0;i<the[0].size();++i)
28 |         {
29 |             n=0;
30 |             p=-1;
31 |             for(j=0;j<the[1].size();++j)
32 |             {
33 |                 if(the[0][i]==the[1][j] && n+1>dp[j])
34 |                 {
35 |                     dp[j]=n+1;
36 |                     path[j]=p;
37 |                 }
38 |                 if(the[1][j]<the[0][i] && n<dp[j])
39 |                 {
40 |                     n=dp[j];
41 |                     p=j;
42 |                 }
43 |             }
44 |             n=0;
45 |             p=-1;
46 |             for(i=0;i<the[1].size();++i)
47 |                 if(dp[i]>n)
48 |                     n=dp[p=i];
49 |             printf("%d\n",n);
50 |             for(i=n-1;i>=0;--i)
51 |             {
52 |                 ans[i]=the[1][p];
53 |                 p=path[p];
54 |             }
55 |             for(i=0;i<n;++i)
56 |                 printf("%d_",ans[i]);
57 |             puts("");
58 |         }
59 |     }
60 |     return 0;
61 | }

```

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;
55 |                     ch[k]=ch[j];
56 |                 }
57 |                 ++sz[j];
58 |             }
59 |     }
60 | }
61 |
62 | inline void rm(int c)
63 | {
64 |     l[r[c]]=l[c];
65 |     r[l[c]]=r[c];
66 |     static int i,j;
67 |     for(i=d[c];i!=c;i=d[i])
68 |         for(j=l[i];j!=i;j=r[j])
69 |         {
70 |             u[d[j]]=u[j];
71 |             d[u[j]]=d[j];
72 |             --sz[ch[j]];
73 |         }
74 | }
75 |
76 | inline void add(int c)
77 | {
78 |     static int i,j;
79 |     for(i=u[c];i!=c;i=u[i])
80 |         for(j=l[i];j!=i;j=r[j])
81 |         {
82 |             ++sz[ch[j]];
83 |             u[d[j]]=d[u[j]]=j;
84 |         }
85 |     l[r[c]]=r[l[c]]=c;
86 | }
87 |
88 | bool dlx(int k)
89 | {
90 |     if(hd==r[hd])
91 |     {
92 |         ans.resize(k);
93 |         return true;

```

```

94     }
95     int s=inf,c;
96     int i,j;
97     for(i=r[hd];i!=hd;i=r[i])
98         if(sz[i]<s)
99             {
100                 s=sz[i];
101                 c=i;
102             }
103     rm(c);
104     for(i=d[c];i!=c;i=d[i])
105     {
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],
126     cntcol[N];
127 int dcnt = 0;
128 //初始化一个节点
129 inline void addnode(int &x)
130 {
131     ++x;
132     r[x] = l[x] = u[x] = d[x] = x;
133 }
134 //将加入到后xrowx
135 inline void insert_row(int rowx, int x)
136 {
137     r[l[rowx]] = x;
138     l[x] = l[rowx];
139     r[x] = rowx;
140     l[rowx] = x;
141 }
142 //将加入到后xcolx
143 inline void insert_col(int colx, int x)
144 {
145     d[u[colx]] = x;
146     u[x] = u[colx];
147     d[x] = colx;
148     u[colx] = x;
149 }
150 //全局初始化
151 inline void dlx_init(int cols)
152 {
153     memset(h, -1, sizeof(h));
154     memset(cntcol, 0, sizeof(cntcol));
155     dcnt = -1;
156     addnode(dcnt);
157     for (int i = 1; i <= cols; ++i)
158     {
159         addnode(dcnt);
160         insert_row(0, dcnt);
161     }
162 }
163 //删除一列以及相关的所有行
164 inline void remove(int c)
165 {
166     l[r[c]] = l[c];
167     r[l[c]] = r[c];
168     for (int i = d[c]; i != c; i = d[i])
169         for (int j = r[i]; j != i; j = r[j])
170             {
171                 u[d[j]] = u[j];
172                 d[u[j]] = d[j];
173                 cntcol[col[j]]--;
174             }
175 }
176 //恢复一列以及相关的所有行
177 inline void resume(int c)
178 {
179     for (int i = u[c]; i != c; i = u[i])
180         for (int j = l[i]; j != i; j = l[j])
181             {
182                 u[d[j]] = j;
183                 d[u[j]] = j;
184                 cntcol[col[j]]++;
185             }
186     l[r[c]] = c;
187     r[l[c]] = c;
188 }

```

```

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             {
59                 for (k = R[j]; k != j; k = R[k])
60                     vis[C[k]] = true;
61             }
62         }
63     }
64     return res;
65 }
66 void Dance(int now)
67 {
68     if (R[0] == 0)
69         ans = min(ans, now);
70     else if (now + A() < ans)
71     {
72         int i, j, temp, c;
73         for (temp = INF, i = R[0]; i; i = R[i])
74         {
75             if (temp > S[i])
76             {
77                 temp = S[i];
78                 c = i;
79             }
80         }
81         for (i = D[c]; i != c; i = D[i])
82         {
83             Remove(i);
84             for (j = R[i]; j != i; j = R[j])
85                 Remove(j);
86             Dance(now + 1);
87             for (j = L[i]; j != i; j = L[j])
88                 Resume(j);
89             Resume(i);
90         }
91     }
92 }
93 void Init(int m)
94 {
95     int i;
96     for (i = 0; i <= m; i++)
97     {
98         R[i] = i + 1;
99         L[i + 1] = i;
100         U[i] = D[i] = i;
101         S[i] = 0;
102     }
103     R[m] = 0;
104     size = m + 1;
105 }

```

8.4 fibonacci knapsack

```

1 #include<stdio.h>
2 #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 short n, T, t, i;
13 long long carry, sumw, sumc;
14 long long ans, las[MAXX];
15
16 int com(const void *n, const void *m)
17 {
18     struct mono *a = (struct mono *)n, *b = (struct mono *)m;

```

```

19     if(a->weig != b->weig)
20         return a->weig - b->weig;
21     else
22         return b->cost - a->cost;
23 }
24
25 bool comp(const struct mono a, const struct mono b)
26 {
27     if(a.weig != b.weig)
28         return a.weig < b.weig;
29     else
30         return b.cost < a.cost;
31 }
32
33 void dfs(short i, long long cost_n, long long carry_n, short last)
34 {
35     if(ans < cost_n)
36         ans = cost_n;
37     if(i == n || goods[i].weig > carry_n || cost_n + las[i] <= ans)
38         return;
39     if(last || (goods[i].weig != goods[i-1].weig && goods[i].cost
40         > goods[i-1].cost))
41         dfs(i+1, cost_n + goods[i].cost, carry_n - goods[i].weig, 1);
42     dfs(i+1, cost_n, carry_n, 0);
43 }
44
45 int main()
46 {
47     // freopen("asdf", "r", stdin);
48     scanf("%hd", &T);
49     for(t=1; t<=T; ++t)
50     {
51         scanf("%hd%lld", &n, &carry);
52         sumw=0;
53         sumc=0;
54         ans=0;
55         for(i=0; i<n; ++i)
56         {
57             scanf("%lld%lld", &goods[i].weig, &goods[i].cost);
58             sumw += goods[i].weig;
59             sumc += goods[i].cost;
60         }
61         if(sumw <= carry)
62         {
63             printf("Case_%hd: %lld\n", t, sumc);
64             continue;
65         }
66         // qsort(goods, n, sizeof(struct mono), com);
67         std::sort(goods, goods+n, comp);
68         for(i=0; i<n; ++i)
69         {
70             printf("%lld %lld\n", goods[i].weig, goods[i].cost);
71             ;
72             las[i] = sumc;
73             sumw -= goods[i].weig;
74             dfs(0, 0, carry, 1);
75             printf("Case_%hd: %lld\n", t, ans);
76         }
77     }
78     return 0;
79 }

```

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 <stdio>
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
12      otherwise
13  // constructors
14  Bigint() {} // default constructor
15  Bigint( std::string b ) { (*this) = b; } // constructor for
16      std::string
17  // some helpful methods
18  int size() // returns number of digits
19  {
20      return a.size();
21  }
22  Bigint inverseSign() // changes the sign
23  {
24      sign *= -1;
25      return (*this);
26  }
27  Bigint normalize( int newSign ) // removes leading 0, fixes
28      sign
29  {
30      for( int i = a.size() - 1; i > 0 && a[i] == '0'; i-- )
31          a.erase(a.begin() + i);
32      sign = ( a.size() == 1 && a[0] == '0' ) ? 1 : newSign;
33      return (*this);
34  }
35  // assignment operator
36  void operator = ( std::string b ) // assigns a std::string
37      to Bigint
38  {
39      a = b[0] == '-' ? b.substr(1) : b;
40      reverse( a.begin(), a.end() );
41      this->normalize( b[0] == '-' ? -1 : 1 );
42  }
43  // conditional operators
44  bool operator < ( const Bigint &b ) const // less than
45      operator
46  {
47      if( sign != b.sign )
48          return sign < b.sign;
49      if( a.size() != b.a.size() )
50          return sign == 1 ? a.size() < b.a.size() : a.size()
51              > b.a.size();
52      for( int i = a.size() - 1; i >= 0; i-- )
53          if( a[i] != b.a[i] )
54              return sign == 1 ? a[i] < b.a[i] : a[i] > b.a[i];
55      return false;
56  }
57  bool operator == ( const Bigint &b ) const // operator for
58      equality
59  {
60      return a == b.a && sign == b.sign;
61  }
62  // mathematical operators
63  Bigint operator + ( Bigint b ) // addition operator
64      overloading
65  {
66      if( sign != b.sign )
67          return (*this) - b.inverseSign();
68      Bigint c;
69      for( int i = 0, carry = 0; i < a.size() || i < b.size() ||
70          carry; i++ )
71      {
72          carry += (i < a.size() ? a[i] - 48 : 0) + (i < b.size() ?
73              b.a[i] - 48 : 0);
74          c.a += (carry % 10 + 48);
75          carry /= 10;
76      }
77      return c.normalize(sign);
78  }
79  Bigint operator - ( Bigint b ) // subtraction operator
80      overloading
81  {
82      if( sign != b.sign )
83          return (*this) + b.inverseSign();
84      int s = sign; sign = b.sign = 1;
85      if( (*this) < b )
86          return ((b - (*this)).inverseSign()).normalize(-s);
87      Bigint c;
88      for( int i = 0, borrow = 0; i < a.size(); i++ )
89      {
90          borrow = a[i] - borrow - (i < b.size() ? b.a[i] :
91              48);
92          c.a += borrow >= 0 ? borrow + 48 : borrow + 58;
93          borrow = borrow >= 0 ? 0 : 1;
94      }
95      return c.normalize(s);
96  }
97  Bigint operator * ( Bigint b ) // multiplication operator
98      overloading
99  {
100      Bigint c("0");
101      for( int i = 0, k = a[i] - 48; i < a.size(); i++, k =
102          [i] - 48 )
103      {
104          while(k-->0)
105              c = c + b; // ith digit is k, so, we add k
106                          times
107          b.a.insert(b.a.begin(), '0'); // multiplied by 10
108      }
109      return c.normalize(sign * b.sign);
110  }
111  Bigint operator / ( Bigint b ) // division operator
112      overloading
113  {
114      if( b.size() == 1 && b.a[0] == '0' )
115          b.a[0] /= ( b.a[0] - 48 );
116      Bigint c("0"), d;
117      for( int j = 0; j < a.size(); j++ )
118          d.a += "0";
119      int dSign = sign * b.sign;
120      b.sign = 1;
121      for( int i = a.size() - 1; i >= 0; i-- )
122      {
123          c.a.insert( c.a.begin(), '0' );
124          c = c + a.substr( i, 1 );
125          while( !( c < b ) )
126          {
127              c = c - b;
128              d.a[i]++;
129          }
130      }
131      return d.normalize(dSign);
132  }
133  Bigint operator % ( Bigint b ) // modulo operator
134      overloading
135  {
136      if( b.size() == 1 && b.a[0] == '0' )
137          b.a[0] /= ( b.a[0] - 48 );
138      Bigint c("0");
139      b.sign = 1;
140      for( int i = a.size() - 1; i >= 0; i-- )
141      {
142          c.a.insert( c.a.begin(), '0' );
143          c = c + a.substr( i, 1 );
144          while( !( c < b ) )
145              c = c - b;
146      }
147      return c.normalize(sign);
148  }
149  // output method
150  void print()
151  {
152      if( sign == -1 )
153          putchar('-');
154      for( int i = a.size() - 1; i >= 0; i-- )
155          putchar(a[i]);
156  }
157  }
158  int main()
159  {
160      Bigint a, b, c; // declared some Bigint variables
161      // taking Bigint input //
162      // using mathematical operators //
163      std::string input; // std::string to take input
164      std::cin >> input; // take the Big integer as std::string
165      a = input; // assign the std::string to Bigint a
166      std::cin >> input; // take the Big integer as std::string
167      b = input; // assign the std::string to Bigint b
168      // Using mathematical operators //
169      c = a + b; // adding a and b
170      c.print(); // printing the Bigint
171      puts(""); // newline
172      c = a - b; // subtracting b from a
173      c.print(); // printing the Bigint
174      puts(""); // newline
175      c = a * b; // multiplying a and b
176      c.print(); // printing the Bigint
177      puts(""); // newline
178      c = a / b; // dividing a by b
179      c.print(); // printing the Bigint
180      puts(""); // newline
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
196                                     operator
197
198 return 0;
199 }

```

9.3 Binary Search

```

1 // [0,n)
2 inline int go(int A[],int n,int x) // return the least i that
3   make A[i]==x;
4 {
5     static int l,r,mid,re;
6     l=0;
7     r=n-1;
8     re=-1;
9     while(l<=r)
10    {
11        mid=l+r>>1;
12        if(A[mid]<x)
13            l=mid+1;
14        else
15        {
16            r=mid-1;
17            if(A[mid]==x)
18                re=mid;
19        }
20    }
21    return re;
22 }
23 inline int go(int A[],int n,int x) // return the largest i that
24   make A[i]==x;
25 {
26     static int l,r,mid,re;
27     l=0;
28     r=n-1;
29     re=-1;
30     while(l<=r)
31    {
32        mid=l+r>>1;
33        if(A[mid]<=x)
34        {
35            l=mid+1;
36            if(A[mid]==x)
37                re=mid;
38        }
39        else
40            r=mid-1;
41    }
42    return re;
43 }
44 inline int go(int A[],int n,int x) // retrun the largest i that
45   make A[i]<x;
46 {
47     static int l,r,mid,re;
48     l=0;
49     r=n-1;
50     re=-1;
51     while(l<=r)
52    {
53        mid=l+r>>1;
54        if(A[mid]<x)
55        {
56            l=mid+1;
57            re=mid;
58        }
59        else
60            r=mid-1;
61    }
62    return re;
63 }
64 inline int go(int A[],int n,int x) // return the largest i that
65   make A[i]<=x;
66 {
67     static int l,r,mid,re;
68     l=0;
69     r=n-1;
70     re=-1;
71     while(l<=r)
72    {
73        mid=l+r>>1;
74        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
85   make A[i]>x;
86 {
87     static int l,r,mid,re;
88     l=0;
89     r=n-1;
90     re=-1;
91     while(l<=r)
92    {
93        mid=l+r>>1;
94        if(A[mid]<=x)
95            l=mid+1;
96        else
97        {
98            r=mid-1;
99            re=mid;
100        }
101    }
102    return re;
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.nextInt(); //Returns this scanner's default
19 Scanner in.useRadix(int radix); // Sets this scanner's
20 default radix to the specified radix.
21 void in.close(); //Closes this scanner.
22
23 //String
24 char str.charAt(int index);
25 int str.compareTo(String anotherString); // <0 if
26 less. ==0 if equal. >0 if greater.
27 int str.compareToIgnoreCase(String str);
28 String str.concat(String str);

```

28	boolean	str.contains(CharSequence s);	24	6.1、对数调整精度 or 将乘法转换成加法
29	boolean	str.endsWith(String suffix);	25	6.2、点化区间, 区间化点
30	boolean	str.startsWith(String preffix);	26	7、数组大小……
31	boolean	str.startsWith(String preffix,int toffset);		
32	int	str.hashCode();		
33	int	str.indexOf(int ch);		
34	int	str.indexOf(int ch, int fromIndex);		
35	int	str.indexOf(String str);		
36	int	str.indexOf(String str, int fromIndex);		
37	int	str.lastIndexOf(int ch);		
38	int	str.lastIndexOf(int ch, int fromIndex);		
39	//(ry			
40	int	str.length();		
41	String	str.substring(int beginIndex);		
42	String	str.substring(int beginIndex, int endIndex);		
43	String	str.toLowerCase();		
44	String	str.toUpperCase();		
45	String	str.trim();// Returns a copy of the string, with leading and trailing whitespace omitted.		
46				
47	//StringBuilder			
48	StringBuilder	str.insert(int offset,...);		
49	StringBuilder	str.reverse();		
50	void	str.setCharAt(int index, int ch);		
51				
52	//BigInteger			
53	compareTo(); equals(); doubleValue(); longValue(); hashCode();			
	toString(); toString(int radix); max(); min(); mod();			
	modPow(BigInteger exp, BigInteger m); nextProbablePrime();			
	pow();			
54	andNot(); and(); xor(); not(); or(); getLowestSetBit();			
	bitCount(); bitLength(); setBig(int n); shiftLeft(int n);			
	shiftRight(int n);			
55	add(); divide(); divideAndRemainder(); remainder(); multiply();			
	subtract(); gcd(); abs(); signum(); negate();			
56				
57	//BigDecimal			
58	movePointLeft(); movePointRight(); precision();			
	stripTrailingZeros(); toBigInteger(); toPlainString();			
59				
60				
61	//sort			
62	class pii implements Comparable			
63	{			
64	public int a,b;			
65	public int compareTo(Object i)			
66	{			
67	pii c=(pii)i;			
68	return a==c.a?c.b-b:c.a-a;			
69	}			
70	}			
71				
72	class Main			
73	{			
74	public static void main(String[] args)			
75	{			
76	pii[] the=new pii[2];			
77	the[0]=new pii();			
78	the[1]=new pii();			
79	the[0].a=1;			
80	the[0].b=1;			
81	the[1].a=1;			
82	the[1].b=2;			
83	Arrays.sort(the);			
84	for (int i=0;i<2;++i)			
85	System.out.printf("%d_%d\n",the[i].a,the[i].b);			
86	}			
87	}			

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、状态状态状态状态状态状态状态状态状态
19 2、calm_down();calm_down();calm_down();
20 3、读完题目读完题目读完题目
21 4、不盲目跟版
22 5、考虑换题/换想法
23 6、对数/离线/hash/观察问题本身/点 ↔ 区间互转

```