

Code Library



Himemiya Nanao @ Perfect Freeze

August 17, 2013

Contents

1	data structure	1	4.15	Improved Shortest Augmenting Path Algorithm	25
1.1	atlantis	1	4.16	k Shortest Path	26
1.2	Binary Indexed tree	1	4.17	Kariv-Hakimi Algorithm	26
1.3	COT	1	4.18	Kuhn-Munkres algorithm	27
1.4	hose	2	4.19	LCA - DA	27
1.5	Leftist tree	2	4.20	LCA - tarjan - minmax	28
1.6	Network	2	4.21	Minimum Ratio Spanning Tree	28
1.7	OTOCI	4	4.22	Minimum Steiner Tree	28
1.8	picture	4	4.23	Minimum-cost flow problem	29
1.9	Size Blanced Tree	5	4.24	Second-best MST	30
1.10	Sparse Table - rectangle	6	4.25	Spanning tree	30
1.11	Sparse Table - square	6	4.26	Stable Marriage	30
1.12	Sparse Table	6	4.27	Stoer-Wagner Algorithm	30
1.13	Treap	6	4.28	Strongly Connected Component	31
2	geometry	7	4.29	ZKW's Minimum-cost flow	31
2.1	3D	7	5	math	31
2.2	3DCH	8	5.1	cantor	31
2.3	circle&ploy's area	9	5.2	Discrete logarithms - BSGS	32
2.4	circle's area	9	5.3	Divisor function	32
2.5	circle	9	5.4	Extended Euclidean Algorithm	32
2.6	closest point pair	10	5.5	Fast Fourier Transform	32
2.7	ellipse	11	5.6	Gaussian elimination	32
2.8	Graham's scan	11	5.7	inverse element	33
2.9	half-plane intersection	12	5.8	Linear programming	33
2.10	k-d tree	12	5.9	Lucas' theorem(2)	34
2.11	Manhattan MST	13	5.10	Lucas' theorem	34
2.12	others	13	5.11	Matrix	34
2.13	Pick's theorem	14	5.12	Miller-Rabin Algorithm	34
2.14	PointInPoly	14	5.13	Multiset	35
2.15	rotating caliper	14	5.14	Pell's equation	35
2.16	shit	15	5.15	Pollard's rho algorithm	35
2.17	sort - polar angle	15	5.16	Prime	36
2.18	triangle	15	5.17	Reduced Residue System	36
3	geometry/tmp	15	5.18	Simpson's rule	36
3.1	circle	15	5.19	System of linear congruences	36
3.2	circles	16	6	string	36
3.3	halfplane	17	6.1	Aho-Corasick Algorithm	36
3.4	line	17	6.2	Gusfield's Z Algorithm	37
3.5	line3d	18	6.3	Manacher's Algorithm	37
3.6	plane	18	6.4	Morris-Pratt Algorithm	38
3.7	point	18	6.5	smallest representation	38
3.8	point3d	19	6.6	Suffix Array - DC3 Algorithm	38
3.9	polygon	19	6.7	Suffix Array - Prefix-doubling Algorithm	38
3.10	polygons	21	6.8	Suffix Automaton	39
4	graph	21	7	dynamic programming	39
4.1	2SAT	21	7.1	knapsack problem	39
4.2	Articulation	21	7.2	LCIS	39
4.3	Augmenting Path Algorithm for Maximum Cardinality Bipartite Matching	21	8	search	39
4.4	Biconnected Component - Edge	21	8.1	dlx	39
4.5	Biconnected Component	22	8.2	dlx - exact cover	39
4.6	Blossom algorithm	22	8.3	dlx - repeat cover	40
4.7	Bridge	23	8.4	fibonacci knapsack	41
4.8	Chu-Liu:Edmonds' Algorithm	23	9	others	41
4.9	Covering problems	23	9.1	.vimrc	41
4.10	Difference constraints	24	9.2	bigint	41
4.11	Dinitz's algorithm	24	9.3	Binary Search	42
4.12	Flow network	24	9.4	Java	43
4.13	Hamiltonian circuit	25	9.5	others	43
4.14	Hopcroft-Karp algorithm	25			

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]=map[r]-map[l-1];
36     }
37     else 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 %lf\n",l,r,h,f);
68     }
69 }ln[inf];
70
71 int main()
72 {
73     make(1,1,inf);
74     while(scanf("%d",&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+1].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,rmap[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,rmap[ln[i].r],ln[i].f);
109         }
110         printf("Test_case_#%d\nTotal_explored_area: %.2lf\n",++cas,ans);
111     }
112     return 0;
113 }
```

1.2 Binary Indexed tree

```
1 int tree[MAXX];
2
```

```
3 inline int lowbit(const int &a)
4 {
5     return a&-a;
6 }
7
8 inline void update(int pos,const int &val)
9 {
10     while(pos<MAXX)
11     {
12         tree[pos]+=val;
13         pos+=lowbit(pos);
14     }
15 }
16
17 inline int read(int pos)
18 {
19     int re(0);
20     while(pos>0)
21     {
22         re+=tree[pos];
23         pos=lowbit(pos);
24     }
25     return re;
26 }
27
28 int find_Kth(int k)
29 {
30     int now=0;
31     for (char i=20;i>=0;--i)
32     {
33         now=(1<<i);
34         if (now<MAXX || tree[now]>=k)
35             now=(1<<i);
36         else k-=tree[now];
37     }
38     return now+1;
39 }
```

1.3 COT

```
1 #include<cstdio>
2 #include<algorithm>
3
4 #define MAXX 100111
5 #define MAX (MAXX*23)
6 #define N 18
7
8 int sz[MAX],lson[MAX],rson[MAX],cnt;
9 int head[MAXX];
10 int pre[MAXX][N];
11 int map[MAXX],m;
12
13 int edge[MAXX],nxt[MAXX<<1],to[MAXX<<1];
14 int n,i,j,k,q,l,r,mid;
15 int num[MAXX],dg[MAXX];
16
17 int make(int l,int r)
18 {
19     if(l==r)
20         return ++cnt;
21     int id(++cnt),mid((l+r)>>1);
22     lson[id]=make(l,mid);
23     rson[id]=make(mid+1,r);
24     return id;
25 }
26
27 inline int update(int id,int pos)
28 {
29     int re(++cnt);
30     l=1;
31     r=m;
32     int nid(re);
33     sz[nid]=sz[id]+1;
34     while(l<r)
35     {
36         mid=(l+r)>>1;
37         if(pos<=mid)
38         {
39             lson[nid]=++cnt;
40             rson[nid]=rson[id];
41             nid=lson[nid];
42             id=lson[id];
43             r=mid;
44         }
45         else
46         {
47             lson[nid]=lson[id];
48             rson[nid]=++cnt;
49             nid=rson[nid];
50             id=rson[id];
51             l=mid+1;
52         }
53         sz[nid]=sz[id]+1;
54     }
55     return re;
56 }
57
58 void rr(int now,int fa)
59 {
60     dg[now]=dg[fa]+1;
61     head[now]=update(head[fa],num[now]);
62     for(int i(edge[now]);i;i=nxt[i])
63     {
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
74 inline int query(int a,int b,int n,int k)
75 {
76     static int tmp,t;
77     l=1;
78     r=m;
79     a=head[a];
80     b=head[b];
81     t=num[n];
82     while(l<r)
83     {
84         mid=(l+r)>>1;
85         tmp=sz[lson[a]]+sz[lson[b]]-2*sz[lson[n]]+(1<=t && t<=mid);
86         if(tmp>=k)
```

```

87         a=son[a];
88         b=son[b];
89         n=son[n];
90         r=mid;
91     }
92     else
93     {
94         k=tmp;
95         a=son[a];
96         b=son[b];
97         n=son[n];
98         l=mid+1;
99     }
100 }
101 return l;
102 }
103
104 inline int lca(int a,int b)
105 {
106     static int i,j;
107     j=0;
108     if(dg[a]<dg[b])
109         std::swap(a,b);
110     for(i=dg[a]-dg[b];i;i>=&1++j)
111         if(i&1)
112             a=pre[a][j];
113     if(a==b)
114         return a;
115     for(i=N-1;i>=0;-i)
116         if(pre[a][i]!=pre[b][i])
117         {
118             a=pre[a][i];
119             b=pre[b][i];
120         }
121     return pre[a][0];
122 }
123
124 int main()
125 {
126     scanf("%d%d",&n,&q);
127     for(i=1;i<=n++i)
128     {
129         scanf("%d",&num[i]);
130         map[i]=num[i];
131     }
132     std::sort(map+1,map+n+1);
133     n=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+n+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(1,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:r<i.w;
14     }
15 }a[MAXX];
16
17 int c[MAXX];
18 long long col[MAXX],sz[MAXX],ans[MAXX];
19 int n,m,cnt,len;
20
21 long long gcd(long long a,long long b)
22 {
23     return a?gcd(b%a,a):b;
24 }
25
26 int i,j,k,now;
27 long long all,num;
28
29 int main()
30 {
31     scanf("%d%d",&n,&m);
32     for(i=1;i<=n++i)
33         scanf("%d",&c[i]);
34     len=sqrt(m);
35     for(i=1;i<=m++i)
36     {
37         scanf("%d%d",&a[i].l,&a[i].r);
38         if(a[i].l>a[i].r)
39             std::swap(a[i].l,a[i].r);
40         sz[i]=a[i].r-a[i].l+1;
41         a[i].wa=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<stdio>
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<stdlib.h>
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;i>=&1++j)
20         if(i&1)
21             a=fa[a][j];
22     if(a==b)

```

```

23     return a;
24     for (int 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 MAXI (MAXN*5)
68
69 namespace Treap
70 {
71     int cnt;
72     int son[MAXI][2],key[MAXI],val[MAXI],sz[MAXI];
73
74     inline void init()
75     {
76         key[0]=RAND_MAX;
77         val[0]=0xc0c0c0c0;
78         cnt=0;
79     }
80
81     inline void up(int id)
82     {
83         sz[id]=sz[son[id][0]]+sz[son[id][1]]+1;
84     }
85     inline void rot(int &id,int tp)
86     {
87         static int k;
88         k=son[id][tp];
89         son[id][tp]=son[k][tp^1];
90         son[k][tp^1]=id;
91         up(id);
92         up(k);
93         id=k;
94     }
95     void insert(int &id,int v)
96     {
97         if (id)
98         {
99             int k(v>val[id]);
100             insert(son[id][k],v);
101             if (key[son[id][k]]<key[id])
102                 rot(id,k);
103             else
104                 up(id);
105             return;
106         }
107         id=++cnt;
108         key[id]=rand()-1;
109         val[id]=v;
110         sz[id]=1;
111         son[id][0]=son[id][1]=0;
112     }
113     void del(int &id,int v)
114     {
115         if (!id)
116             return;
117         if (val[id]==v)
118         {
119             int k(key[son[id][1]]<key[son[id][0]]);
120             if (!son[id][k])
121             {
122                 id=0;
123                 return;
124             }
125             rot(id,k);
126             del(son[id][k^1],v);
127         }
128         else
129             del(son[id][v>val[id]],v);
130         up(id);
131     }
132     int rank(int id,int v)
133     {
134         if (!id)
135             return 0;
136         if (val[id]<=v)
137             return sz[son[id][0]]+1+rank(son[id][1],v);
138         return rank(son[id][0],v);
139     }
140     void print(int id)
141     {
142         if (!id)
143             return;
144         print(son[id][0]);
145         printf("%d ",val[id]);
146         print(son[id][1]);
147     }
148 }
149
150 int head[MAXX],root[MAXX],len[MAXX],pos[MAXX];

```

```

151
152 #define MAX (MAXN*6)
153 #define mid (l+r>>1)
154 #define lc son[id],l,mid
155 #define rc son[id],mid+1,r
156
157 int lson[MAX],rson[MAX];
158 int treap[MAX];
159
160 void make(int &id,int l,int r,int *the)
161 {
162     id=++cnt;
163     static int k;
164     for (k=l; k<=r; ++k)
165         Treap::insert(treap[id],the[k]);
166     if (l==r)
167     {
168         make(lc,the);
169         make(rc,the);
170     }
171 }
172
173 int query(int id,int l,int r,int a,int b,int q)
174 {
175     if (a<=l && r<=b)
176         return Treap::rank(treap[id],q);
177     int re(0);
178     if (a<=mid)
179         re=query(lc,a,b,q);
180     if (b>mid)
181         re=query(rc,a,b,q);
182     return re;
183 }
184
185 inline int query(int a,int b,int v)
186 {
187     static int re;
188     for (re=0; root[a]!=root[b]; a=fa[root[a]][0])
189         re=query(head[root[a]],1,len[root[a]],1,pos[a],v);
190     re=query(head[root[a]],1,len[root[a]],pos[b],pos[a],v);
191     return re;
192 }
193
194 inline void update(int id,int l,int r,int pos,int val,int n)
195 {
196     while (l<=r)
197     {
198         Treap::del(treap[id],val);
199         Treap::insert(treap[id],n);
200         if (l==r)
201             return;
202         if (pos<=mid)
203         {
204             id=lson[id];
205             r=mid;
206         }
207         else
208         {
209             id=rson[id];
210             l=mid+1;
211         }
212     }
213 }
214
215 int n,q,i,j,k;
216 int val[MAXX];
217
218 int main()
219 {
220     srand(1e9+7);
221     scanf("%d%d",&n,&q);
222     for (i=1; i<=n; ++i)
223         scanf("%d",&val[i]);
224     for (k=1; k<=n; ++k)
225     {
226         scanf("%d%d",&i,&j);
227         add(i,j);
228         add(j,i);
229     }
230     rr(rand()%n+1);
231     for (j=1; j<=n; ++j)
232         for (i=1; i<=n; ++i)
233             fa[i][j]=fa[fa[i][j-1]][j-1];
234
235     Treap::init();
236     cnt=0;
237     for (i=1; i<=n; ++i)
238         if (!pre[i])
239         {
240             static int tmp[MAXX];
241             for (k=1; i=j; j=next[j], ++k)
242             {
243                 pos[j]=k;
244                 root[j]=i;
245                 tmp[k]=val[j];
246             }
247             --k;
248             len[i]=k;
249             make(head[i],1,k,tmp);
250         }
251     while (q--)
252     {
253         scanf("%d",&k);
254         if (k)
255         {
256             static int a,b,c,d,l,r,ans,m;
257             scanf("%d%d",&a,&b);
258             c=ca(a,b);
259             if (dg[a]+dg[b]-2*dg[c]+1<k)
260             {
261                 puts("invalid request!");
262                 continue;
263             }
264             k=dg[a]+dg[b]-2*dg[c]+1-k+1;
265             if (dg[a]<dg[b])
266                 std::swap(a,b);
267             l=1e9;
268             r=le9;
269             if (bl==c)
270             {
271                 d=a;
272                 for (i=0; j=dg[a]-dg[c]-1; j;j>=l++i)
273                     if (j&1)
274                         d=fa[d][i];
275                 while (l<=r)
276                 {
277                     m=l+r>>1;
278                     if (query(a,d,m)+query(b,c,m)>=k)

```

```

279         {
280             ans=m;
281             r=m-1;
282         }
283         else
284             l=m+1;
285     }
286 }
287 else
288 {
289     while(l<=r)
290     {
291         m=l+r>>1;
292         if(query(a,c,m)>=k)
293         {
294             ans=m;
295             r=m-1;
296         }
297         else
298             l=m+1;
299     }
300     printf("%d\n",ans);
301 }
302 else
303 {
304     scanf("%d%d",&i,&j);
305     update(head[root[i]],1,len[root[i]],pos[i],val[i],j);
306     val[i]=j;
307 }
308 }
309 }
310 return 0;
311 }

```

1.7 OTOCI

```

1 //记得随手啊.....亲.....down
2 //时记得优先检查debugup/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) //记得随手啊.....亲.....down
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)//记得随手啊.....亲.....down
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 method:
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 MAXMAXX<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 mmap[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,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 && r==rr)
32     {
33         cnt[id]+=val;
34         if(cnt[id])
35         {
36             rt[id]=lf[id]=true;
37             len[id]=mmap[r]-mmap[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] && !f[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,lf,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,lf,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] && !f[id<<1|1])
73         --seg[id];
74     rt[id]=rt[id<<1|1];
75     lf[id]=lf[id<<1];
76 }
77 }
78
79 struct node
80 {
81     int l,r,h;
82     char val;
83     inline bool operator<(const node &a)const
84     {
85         return h==a.h?val<a.val:h<a.h;    // trick watch out. val<a.val? val>a.val?
86     }
87     inline void print()
88     {
89         printf("%d %d %d %d\n",l,r,h,val);
90     }
91 }ln[inf];
92
93 int main()
94 {
95     make(1,1,inf);
96     scanf("%d",&n);
97     n<=<=1;
98     map.clear();
99     for(i=0;i<n;++i)
100     {
101         scanf("%d%d%d", &x1,&y1,&x2,&y2);
102         ln[i].l=x1;
103         ln[i].r=x2;
104         ln[i].h=y1;
105         ln[i].val=1;
106         ln[i].l=x1;
107         ln[i].r=x2;
108         ln[i].h=y2;
109         ln[i].val=-1;
110         map[x1]=1;
111         map[x2]=1;
112     }
113     i=1;
114     for(it=map.begin();it!=map.end();++it,++i)
115     {
116         it->second=i;
117         rmap[i]=it->first;
118     }
119     i=0;
120     std::sort(ln,ln+n);
121     update(1,1,inf,map[ln[0].l]+1,map[ln[0].r],ln[0].val);
122     suml=len[1];
123     last=len[1];
124     for(i=1;i<n;++i)
125     {
126         suml-=2*seg[i]*(ln[i].h-ln[i-1].h);
127         update(1,1,inf,map[ln[i].l]+1,map[ln[i].r],ln[i].val);
128         suml+=abs(len[i]-last);
129         last=len[i];
130     }
131     printf("%d\n",sum);
132     return 0;
133 }

```

1.9 Size Blanced Tree

```

1  template<class Tp>class sbt
2  {
3      public:
4          inline void init()
5          {
6              rt=cnt=1[0]=r[0]=sz[0]=0;
7          }
8          inline void ins(const Tp &a)
9          {
10             ins(rt,a);
11         }
12         inline void del(const Tp &a)
13         {
14             del(rt,a);
15         }
16         inline bool find(const Tp &a)
17         {
18             return find(rt,a);
19         }
20         inline Tp pred(const Tp &a)
21         {
22             return pred(rt,a);
23         }
24         inline Tp succ(const Tp &a)
25         {
26             return succ(rt,a);
27         }
28         inline bool empty()
29         {
30             return !sz[rt];
31         }

```

```

32     inline Tp min()
33     {
34         return min(rt);
35     }
36     inline Tp max()
37     {
38         return max(rt);
39     }
40     inline void delsmall(const Tp &a)
41     {
42         dels(rt,a);
43     }
44     inline int rank(const Tp &a)
45     {
46         return rank(rt,a);
47     }
48     inline Tp sel(const int &a)
49     {
50         return sel(rt,a);
51     }
52     inline Tp dsel(int a)
53     {
54         return dsel(rt,a);
55     }
56 private:
57     int cnt,rt,l[MAXX],r[MAXX],sz[MAXX];
58     Tp val[MAXX];
59     inline void rro(int &pos)
60     {
61         int k(l[pos]);
62         l[pos]=r[k];
63         r[k]=pos;
64         sz[k]=sz[pos];
65         sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
66         pos=k;
67     }
68     inline void lro(int &pos)
69     {
70         int k(r[pos]);
71         r[pos]=l[k];
72         l[k]=pos;
73         sz[k]=sz[pos];
74         sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
75         pos=k;
76     }
77     inline void mt(int &pos,bool flag)
78     {
79         if(!pos)
80             return;
81         if(flag)
82             if(sz[r[r[pos]]]>sz[l[pos]])
83                 lro(pos);
84             else
85                 if(sz[l[r[pos]]]>sz[l[pos]])
86                 {
87                     rro(r[pos]);
88                     lro(pos);
89                 }
90             else
91                 return;
92         else
93             if(sz[l[l[pos]]]>sz[r[pos]])
94                 rro(pos);
95             else
96                 if(sz[r[l[pos]]]>sz[r[pos]])
97                 {
98                     lro(l[pos]);
99                     rro(pos);
100                 }
101             else
102                 return;
103         mt(l[pos],false);
104         mt(r[pos],true);
105         mt(pos,false);
106         mt(pos,true);
107     }
108     void ins(int &pos,const Tp &a)
109     {
110         if(pos)
111         {
112             ++sz[pos];
113             if(a<val[pos])
114                 ins(l[pos],a);
115             else
116                 ins(r[pos],a);
117             mt(pos,a>val[pos]);
118             return;
119         }
120         pos=++cnt;
121         l[pos]=r[pos]=0;
122         val[pos]=a;
123         sz[pos]=1;
124     }
125     Tp del(int &pos,const Tp &a)
126     {
127         --sz[pos];
128         if(val[pos]==a || (a<val[pos] && !l[pos]) || (a>val[pos] && !r[pos]))
129         {
130             Tp ret(val[pos]);
131             if(!l[pos] || !r[pos])
132                 pos=l[pos]+r[pos];
133             else
134                 val[pos]=del(l[pos],val[pos]+1);
135             return ret;
136         }
137         else
138             if(a<val[pos])
139                 return del(l[pos],a);
140             else
141                 return del(r[pos],a);
142     }
143     bool find(int &pos,const Tp &a)
144     {
145         if(!pos)
146             return false;
147         if(a<val[pos])
148             return find(l[pos],a);
149         else
150             return (val[pos]==a || find(r[pos],a));
151     }
152     Tp pred(int &pos,const Tp &a)
153     {
154         if(!pos)
155             return a;
156         if(a>val[pos])
157         {
158             Tp ret(pred(r[pos],a));
159             if(ret==a)

```

```

160         return val[pos];
161     else
162         return ret;
163     }
164     return pred(l[pos], a);
165 }
166 Tp succ(int &pos, const Tp &a)
167 {
168     if (!pos)
169         return a;
170     if (a < val[pos])
171     {
172         Tp ret(succ(l[pos], a));
173         if (ret == a)
174             return val[pos];
175         else
176             return ret;
177     }
178     return succ(r[pos], a);
179 }
180 Tp min(int &pos)
181 {
182     if (l[pos])
183         return min(l[pos]);
184     else
185         return val[pos];
186 }
187 Tp max(int &pos)
188 {
189     if (r[pos])
190         return max(r[pos]);
191     else
192         return val[pos];
193 }
194 void dels(int &pos, const Tp &v)
195 {
196     if (!pos)
197         return;
198     if (val[pos] < v)
199     {
200         pos = r[pos];
201         dels(pos, v);
202         return;
203     }
204     dels(l[pos], v);
205     sz[pos] = 1 + sz[l[pos]] + sz[r[pos]];
206 }
207 int rank(const int &pos, const Tp &v)
208 {
209     if (val[pos] == v)
210         return sz[l[pos]] + 1;
211     if (v < val[pos])
212         return rank(l[pos], v);
213     return rank(r[pos], v) + sz[l[pos]] + 1;
214 }
215 Tp sel(const int &pos, const int &v)
216 {
217     if (sz[l[pos]] + 1 == v)
218         return val[pos];
219     if (v > sz[l[pos]] + 1)
220         return sel(r[pos], v - sz[l[pos]] - 1);
221     return sel(l[pos], v);
222 }
223 Tp dsel(const int &pos, const 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]] + 1)
236         return dsel(r[pos], k - 1 - sz[l[pos]]);
237     return dsel(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                     {
38                         if (i == 0)
39                             table[ii][jj][ii][jj] = std::min(table[ii][j - 1][ii][jj], table[ii][j][ii - 1][jj]);
40                         else
41                             table[ii][jj][ii][jj] = std::min(table[ii - 1][jj][ii][jj], table[ii][j - 1][jj][ii - 1]);
42                     }
43             }
44         }
45     }
46 }

```

```

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

```

1.11 Sparse Table - square

```

1  int num[MAXX][MAXX], max[MAXX][MAXX][10];
2  short lg[MAXX];
3
4  int main()
5  {
6     for (i = 2; i <= MAXX; ++i)
7         lg[i] = lg[i >> 1] + 1;
8     scanf("%d", &n, &q);
9     for (i = 0; i <= n; ++i)
10         for (j = 0; j <= n; ++j)
11         {
12             scanf("%d", &num[i][j]);
13             max[i][j][0] = num[i][j];
14         }
15     for (k = 1; k <= lg[n]; ++k)
16     {
17         l = n + 1 - (1 << k);
18         for (i = 0; i < l; ++i)
19             for (j = 0; j < l; ++j)
20             {
21                 max[i][j][k] = std::max(std::max(max[i][j][k - 1], max[i + (1 << (k - 1))][j][k - 1]), std::max(max[i][j + (1 << (k - 1))][k - 1], max[i + (1 << (k - 1))][j + (1 << (k - 1))][k - 1]));
22             }
23     }
24     printf("Case %d: \n", t);
25     while (q--)
26     {
27         scanf("%d%d%d", &i, &j, &l);
28         --i;
29         --j;
30         k = lg[l];
31         printf("%d\n", std::max(std::max(max[i][j][k], max[i][j + (1 << k)][k]), std::max(max[i + (1 << k)][j][k], max[i + (1 << k)][j + (1 << k)][k])));
32     }
33 }

```

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", &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 %d: \n", t);
23     while (q--)
24     {
25         scanf("%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 }
30 }

```

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     }
29 }

```



```

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[1000];
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<1000;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*(1-c)+y*s)*pa.z,
38             (y*x*(1-c)+z*s)*pa.x+(y*y*(1-c)+c)*pa.y+(y*z*(1-c)-x*s)*pa.z,
39             (x*z*(1-c)-y*s)*pa.x+(y*z*(1-c)+x*s)*pa.y+(z*z*(1-c)+c)*pa.z
40         );
41     return pp;
42 }
43
44 //经纬度转换
45
46 x=sin()*cos();
47 y=sin()*sin();
48 z=cos();
49
50 r=sqrt(x^2+y^2+z^2);///?
51 r=sqrt(x^2+y^2+z^2);///?
52
53 =atan(y/x);
54 =acos(z/r);
55
56 r=0,]
57 [0,2]
58 [0,]
59
60 lat1 [-/2,/2]
61 lng1 [-,]
62
63 pv getpv(double lat,double lng,double r)
64 {
65     lat += pi/2;
66     lng += pi;
67     return
68         pv(r*sin(lat)*cos(lng),r*sin(lat)*sin(lng),r*cos(lat));
69 }
70
71 //经纬度球面距离
72
73 #include<cstdio>
74 #include<math>
75
76 #define MAXX 1111
77
78 char buf[MAXX];
79 const double r=6875.0/2,pi=acos(-1.0);
80 double a,b,c,x1,x2,y2,ans;
81
82 int main()
83 {
84     double y1;
85     while(gets(buf)!=NULL)
86     {
87         gets(buf);
88         gets(buf);
89
90         scanf("%lf%lf%lf%lf\n",%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%lf\n",%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%lf\n",%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%lf\n",%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",ans);
120         if(ans+0.005<100)
121             puts("DANGER!");
122
123         gets(buf);
124     }
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 //线段相交
151 bool inter(pv a,pv b,pv c,pv d)
152 {
153     pv ret = (a-b)*(c-d);
154     pv t1 = (b-a)*(c-a);
155     pv t2 = (b-a)*(d-a);
156     pv t3 = (d-c)*(a-c);
157     pv t4 = (d-c)*(b-c);
158     return sgn(t1&xret)*sgn(t2&xret) < 0 && sgn(t3&xret)*sgn(t4&xret) < 0;
159 }
160 //点在直线上
161 bool OnLine(pv p, Line3D L)
162 {
163     return ZERO((p-L.s)*(L.e-L.s));
164 }
165 //点在线段上
166 bool OnSeg(pv p, Line3D L)
167 {
168     return (ZERO((L.s-p)*(L.e-p)) && EQ(Norm(p-L.s)+Norm(p-L.e),Norm(L.e-L.s)));
169 }
170 //点到直线距离
171 double Distance(pv p, Line3D L)
172 {
173     return (Norm((p-L.s)*(L.e-L.s))/Norm(L.e-L.s));
174 }
175 //线段夹角
176 //范围值为 之间的弧度[0,]
177 double Inclination(Line3D L1, Line3D L2)
178 {
179     pv u = L1.e - L1.s;
180     pv v = L2.e - L2.s;
181     return acos( (u & v) / (Norm(u)*Norm(v)) );
182 }

```

2.2 3DCH

```

1  #include<stdio>
2  #include<math>
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,x*i.z-x*i.y,y*i.x-x*i.y);
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 struct pla
33 {
34     short a,b,c;
35     bool ok;
36     pla(){}
37     pla(const short &aa,const short &bb,const short &cc):a(aa),b(bb),c(cc),ok(true){}
38     inline void set();
39     inline void print()
40     {
41         printf("%hd %hd %hd\n",a,b,c);
42     }
43 };
44
45 pv pnt[MAXX];
46 std::vector<pla>fac;
47 short to[MAXX][MAXX];
48
49 inline void pla::set()
50 {
51     to[a][b]=to[b][c]=to[c][a]=fac.size();
52 }
53
54 inline double ptof(const pv &p,const pla &f) //点面距离?
55 {
56     return (pnt[f.b]-pnt[f.a])*(pnt[f.c]-pnt[f.a])^(p-pnt[f.a]);
57 }
58
59 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
60 {
61     return (b-a)*(c-a)^(d-a);
62 }
63
64 inline double ptof(const pv &p,const short &f) //点到号面的距离pf
65 {
66     return fabs(vol(pnt[fac[f].a],pnt[fac[f].b],pnt[fac[f].c],p)/((pnt[fac[f].b]-pnt[fac[f].a])*(pnt[fac[f].c]-pnt[fac[f].a])).len());
67 }
68
69 void dfs(const short&,const short&);
70
71 void deal(const short &p,const short &a,const short &b)
72 {
73     if(fac[to[a][b]].ok)
74         if(ptof(pnt[p],fac[to[a][b]])>eps)
75             dfs(p,to[a][b]);
76     else
77     {
78         pla add(b,a,p);
79         add.set();
80         fac.push_back(add);
81     }
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<x++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<x++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<x++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<x++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,pnt[fac[t].b])<eps &&
166         fabs(vol(a,b,c,pnt[fac[t].c])<eps);
167 }
168
169 //表面多边形数目
170 inline short facetcnt()
171 {
172     short ans=0;
173     for(short i=0;i<fac.size();++i)
174     {
175         for(j=0;j<i;++j)
176             if(same(i,j))
177                 break;
178         if(j==i)
179             ++ans;
180     }
181     return ans;
182 }
183
184 //表面三角形数目
185 inline short trianglecnt()
186 {
187     return fac.size();
188 }
189
190 //三点构成的三角形面积*2
191 inline double area(const pv &a,const pv &b,const pv &c)
192 {
193     return (b-a)*(c-a).len();
194 }
195
196 //表面积
197 inline double area()
198 {
199     double ret(0);
200     for(i=0;i<fac.size();++i)
201         ret+=area(pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
202     return ret/2;
203 }
204
205 //体积
206 inline double volume()
207 {
208     pv o(0,0,0);
209     double ret(0);
210     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 }
```

2.3 circle&ploy’s area

```
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 tv1 = tv.x*tv.x+tv.y*tv.y;
23        double tmp = sqrt(B/tv1); //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);
```

2.4 circle’s area

```
1 //去重
2 {
3     for (int i = 0; i < n; i++)
4     {
5         scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
6         del[i] = false;
7     }
8     for (int i = 0; i < n; i++)
9         if (del[i] == false)
10         {
11             if (c[i].r == 0.0)
12                 del[i] = true;
13             for (int j = 0; j < n; j++)
14                 if (i != j)
15                     if (del[j] == false)
16                         if (cmp(Point(c[i].c,c[j].c).Len()+c[i].r,c[j].r) <= 0)
17                             del[i] = true;
18         }
19         tn = n;
20         n = 0;
21         for (int i = 0; i < tn; i++)
22             if (del[i] == false)
23                 c[n++] = c[i];
24     }
25
26     //ans[i表示被覆盖]次的面积i
27     const double pi = acos(-1.0);
28     const double eps = 1e-8;
29     struct Point
30     {
31         double x,y;
32         Point(){ }
33         Point(double _x,double _y)
34         {
35             x = _x;
36             y = _y;
37         }
38         double Length()
39         {
40             return sqrt(x*x+y*y);
41         }
42     };
43     struct Circle
44     {
45         Point c;
46         double r;
47     };
48     struct Event
49     {
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*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                    }
130                else
131                {
132                    e[tote++] = Event(a0,1);
133                    e[tote++] = Event(a1,-1);
134                }
135            }
136            sort(e,e+tote,Eventcmp);
137            cur = 0;
138            for (int j = 0;j < tote;j++)
139            {
140                if (cur != 0 && cmp(e[j].tim,pre[cur]) != 0)
141                {
142                    ans[cur] += Area(e[j].tim-pre[cur],c[i].r);
143                    ans[cur] += xmult(Point(c[i].c.x+c[i].r*cos(pre[cur]),c[i].c.y+c[i].r*sin(pre[cur])),
144                        Point(c[i].c.x+c[i].r*cos(e[j].tim),c[i].c.y+c[i].r*sin(e[j].tim)))/2.0;
145                }
146                cur += e[j].typ;
147                pre[cur] = e[j].tim;
148            }
149            for (int i = 1;i < n;i++)
150                ans[i] -= ans[i+1];
151            for (int i = 1;i <= n;i++)
152                printf("%d]_=%.3f\n",i,ans[i]);
153        }
154        return 0;
155    }
```

2.5 circle

```
1 //单位圆覆盖
2 #include<stdio>
3 #include<math>
4 #include<vector>
5 #include<algorithm>
6
7 #define MAXX 333
8 #define eps 1e-8
9
10 struct pv
11 {
12     double x,y;
13     pv(){ }
14     pv(const double &xx,const double &yy):x(xx),y(yy){ }
15     inline pv operator-(const pv &i)const
16     {
17         return pv(x-i.x,y-i.y);
18     }
```

```

19 inline double cross(const pv &i)const
20 {
21     return x*i.y-y*i.x;
22 }
23 inline void print()
24 {
25     printf("%lf, %lf\n", x, y);
26 }
27 inline double len()
28 {
29     return sqrt(x*x+y*y);
30 }
31 }pnt[1000];
32
33 struct node
34 {
35     double k;
36     bool flag;
37     node(){}
38     node(const double &kk, const bool &ff):k(kk), flag(ff){}
39     inline bool operator<(const node &i)const
40     {
41         return k<i.k;
42     }
43 };
44
45 std::vector<node>alpha;
46
47 short n, i, j, k, l;
48 short ans, sum;
49 double R=2;
50 double theta, phi, d;
51 const double pi(acos(-1.0));
52
53 int main()
54 {
55     alpha.reserve(1000);
56     while(scanf("%d", &n), n)
57     {
58         for(i=0; i<n; ++i)
59             scanf("%lf, %lf", &pnt[i].x, &pnt[i].y);
60         ans=0;
61         for(i=0; i<n; ++i)
62         {
63             alpha.resize(0);
64             for(j=0; j<n; ++j)
65                 if(i!=j)
66                 {
67                     if((d=pnt[i].pnt[j]).len()>R)
68                         continue;
69                     if(((theta=atan2(pnt[j].y-pnt[i].y, pnt[j].x-pnt[i].x)<0)
70                        theta+=2*pi;
71                        phi=acos(d/R);
72                        alpha.push_back(node(theta-phi, true));
73                        alpha.push_back(node(theta+phi, false));
74                     }
75                     std::sort(alpha.begin(), alpha.end());
76                     for(j=0; j<alpha.size(); ++j)
77                     {
78                         if(alpha[j].flag)
79                             ++sum;
80                         else
81                             --sum;
82                     }
83                     ans+=std::max(ans, sum);
84                 }
85             printf("%d\n", ans+1);
86         }
87         return 0;
88     }
89 }
90 //最小覆盖圆
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[1000], o, t1, 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     t1=a2-a1;
135     lt=b2-b1;
136     u=(b1-a1).cross(lt)/(t1.cross(lt));
137     return a1+t1*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+aa.x-c.x;
148     return ins(aa, bb, cc, dd);
149 }
150
151 int main()
152 {
153     while(scanf("%d", &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<n; ++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<n; ++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("%.21f, %.21f, %.21f\n", o.x, o.y, r);
178     }
179     return 0;
180 }
181
182 //两原面积交
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 //三角形外接圆
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 //三角形内切圆
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 }

```

2.6 closest point pair

```

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

```

```

35 double closest_pair()
36 {
37     sort(p, p+10, cmpx);
38     return DnC(0, N-1);
39 }
40
41 //演算法笔记2
42
43 struct Point {double x, y;}; p[10], t[10];
44 bool cmpx(const Point& i, const Point& j) {return i.x < j.x;};
45 bool cmpy(const Point& i, const Point& j) {return i.y < j.y;};
46
47 double DnC(int L, int R)
48 {
49     if (L>=R) return 1e9; // 沒有點、只有一個點。
50
51     /* : 把所有點分成左右兩側，點數盡量一樣多。Divide */
52
53     int M = (L + R) / 2;
54
55     // 先把中線的座標記起來，因為待會重新排序之後會跑掉。X
56     double x = p[M].x;
57
58     /* : 左側、右側分別遞迴求解。Conquer */
59
60     // 遞迴求解，並且依照座標重新排序。Y
61     double d = min(DnC(L, M), DnC(M+1, R));
62     // if (d == 0.0) return d; // 提早結束
63
64     /* : 尋找靠近中線的點，並依座標排序。MergeYO(N)。 */
65
66     // 尋找靠近中線的點，先找左側。各點已照座標排序了。Y
67     int N = 0; // 靠近中線的點數目
68     for (int i=0; i<=M; ++i)
69         if (x - p[i].x < d)
70             t[N++] = p[i];
71
72     // 尋找靠近中線的點，再找右側。各點已照座標排序了。Y
73     int P = N; // 為分隔位置P
74     for (int i=M+1; i<=R; ++i)
75         if (p[i].x - x < d)
76             t[P++] = p[i];
77
78     // 以座標排序。使用YMerge 方式，合併已排序的兩陣列。Sort
79     inplace_merge(t, t+P, t+N, cmpy);
80
81     /* : 尋找橫跨兩側的最近點對。MergeO(N)。 */
82
83     for (int i=0; i<N; ++i)
84         for (int j=1; j<=2&& i+j<N; ++j)
85             d = min(d, distance(t[i], t[i+j]));
86
87     /* : 重新以座標排序所有點。MergeYO(N)。 */
88
89     // 如此一來，更大的子問題就可以直接使用Merge 。Sort
90     inplace_merge(p+L, p+M+1, p+R+1, cmpy);
91
92     return d;
93 }
94
95 double closest_pair()
96 {
97     sort(p, p+10, cmpx);
98     return DnC(0, N-1);
99 }
100
101 //mzzy
102 //分治
103 double calc_dis(Point &a ,Point &b) {
104     return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
105 }
106
107 //別忘了排序
108 bool operator<(const Point &a ,const Point &b) {
109     if(a.y != b.y) return a.x < b.x;
110     return a.x < b.x;
111 }
112
113 double Gao(int l ,int r ,Point pnts[]) {
114     double ret = inf;
115     if(l == r) return ret;
116     if(l+1 == r) {
117         ret = min(calc_dis(pnts[l], pnts[l+1]), ret);
118         return ret;
119     }
120     if(l+2 == r) {
121         ret = min(calc_dis(pnts[l], pnts[l+1]), ret);
122         ret = min(calc_dis(pnts[l], pnts[l+2]), ret);
123         ret = min(calc_dis(pnts[l+1], pnts[l+2]), ret);
124         return ret;
125     }
126
127     int mid = l+r>>1;
128     ret = min (ret ,Gao(l ,mid,pnts));
129     ret = min (ret , Gao(mid+1, r,pnts));
130
131     for(int c = l ; c<=r; c++)
132         for(int d = c+1; d<=c+7 && d<=r; d++) {
133             ret = min(ret , calc_dis(pnts[c], pnts[d]));
134         }
135     return ret;
136 }
137
138 //增量
139 #include <iostream>
140 #include <cstdio>
141 #include <cstring>
142 #include <map>
143 #include <vector>
144 #include <cmath>
145 #include <algorithm>
146 #define Point pair<double,double>
147 using namespace std;
148
149 const int step[9][2] = {{-1,-1},{-1,0},{-1,1},{0,-1},{0,0},{0,1},{1,-1},{1,0},{1,1}};
150 int n,x,y,nx,ny;
151 map<pair<int,int>,vector<Point>>> g;
152 vector<Point> tmp;
153 Point p[20000];
154 double tx,ty,ans,nowans;
155 vector<Point> >::iterator it,op,ed;
156 pair<int,int> gird;
157 bool flag;
158
159 double Dis(Point p0,Point p1)
160 {
161     return sqrt((p0.first-p1.first)*(p0.first-p1.first)+
162         (p0.second-p1.second)*(p0.second-p1.second));
163
164
165 double CalcDis(Point p0,Point p1,Point p2)
166 {
167     return Dis(p0,p1)+Dis(p0,p2)+Dis(p1,p2);
168 }
169
170 void build(int n,double w)
171 {
172     g.clear();
173     for (int i = 0; i < n; i++)
174         g[make_pair((int)floor(p[i].first/w),(int)floor(p[i].second/w))].push_back(p[i]);
175 }
176
177 int main()
178 {
179     int t;
180     scanf("%d",&t);
181     for (int ft = 1; ft <= t; ft++)
182     {
183         scanf("%d",&n);
184         for (int i = 0; i < n; i++)
185         {
186             scanf("%lf%lf",&tx,&ty);
187             p[i] = make_pair(tx,ty);
188         }
189         random_shuffle(p,p+n);
190         ans = CalcDis(p[0],p[1],p[2]);
191         build(3,ans/2.0);
192         for (int i = 3; i < n; i++)
193         {
194             x = (int)floor(2.0*p[i].first/ans);
195             y = (int)floor(2.0*p[i].second/ans);
196             tmp.clear();
197             for (int k = 0; k < 9; k++)
198             {
199                 nx = x+step[k][0];
200                 ny = y+step[k][1];
201                 gird = make_pair(nx,ny);
202                 if (g.find(gird) != g.end())
203                 {
204                     op = g[gird].begin();
205                     ed = g[gird].end();
206                     for (it = op; it != ed; it++)
207                         tmp.push_back(*it);
208                 }
209             }
210             flag = false;
211             for (int j = 0; j < tmp.size(); j++)
212                 for (int k = j+1; k < tmp.size(); k++)
213                 {
214                     nowans = CalcDis(p[i],tmp[j],tmp[k]);
215                     if (nowans < ans)
216                     {
217                         ans = nowans;
218                         flag = true;
219                     }
220                 }
221             if (flag == true)
222                 build(i+1,ans/2.0);
223             else
224                 g[make_pair((int)floor(2.0*p[i].first/ans),(int)floor(2.0*p[i].second/ans))].push_back(p[i]);
225         }
226         printf("%.3f\n",ans);
227     }
228 }
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

```

2.7 ellipse

```

1  sq(x-h)/sq(q) + sq(y-k)/sq(b) = 1
2
3  x=h+a*cos(t);
4  y=k+b*sin(t);
5
6  area: pi*a*b;
7  distance from center to focus: f=sqrt(sq(a)-sq(b));
8  eccentricity: e=sqrt(a-sq(b/a))=f/a;
9  focal parameter: sq(b)/sqrt(sq(a)-sq(b))=sq(b)/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+ty);
31 }

```

2.8 Graham's scan

```

1  pv pnt[MXN];
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-1)
27     {
28         while((ch.back()-ch[ch.size()-2]).cross(pnt[i]-ch[ch.size()-2])<eps)
29             ch.pop_back();
30         ch.push_back(pnt[i]);
31     }
32 }

```

2.9 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     t=l-b-a;
64     return pv(u*t.l.x/v+a.x,u*t.l.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 //mry
92 bool HPImp(Line a, Line b)
93 {
94     if (fabs(a.k - b.k) > eps)
95         return a.k < b.k;
96     return ((a.s - b.s) * (b.e - b.s)) < 0;
97 }
98
99 Line Q[100];
100
101 void HPI(Line line[], int n, Point res[], int &resn)
102 {
103     int tot = n;
104     std::sort(line, line + n, HPImp);
105     tot = 1;
106     for (int i = 1; i < n; i++)
107         if (fabs(line[i].k - line[i - 1].k) > eps)
108             line[tot++] = line[i];
109     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 }

```

2.10 k-d tree

```

1 //有个很关键的剪枝，在计算完与点的距离后，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，更容易找出正确解。
2 mid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，查询点距离圆的距离（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，仍然还要进入另一棵子树。
3
4 midM说白了就是随便乱搞啦.....
5
6
7 */
8 // hysbz 2626
9 #include<cstdio>
10 #include<algorithm>
11 #include<queue>
12
13 inline long long sqr(long long a){ return a*a;}
14 typedef std::pair<long long,int> pli;
15
16 #define MAXX 100111
17 #define MAX (MAXX<2)
18 #define inf 0x3f3f3f3fll
19 int idx;
20
21 struct PNT
22 {
23     long long x[2];
24     int lb;
25     bool operator<(const PNT &i)const
26     {
27         return x[idx]<i.x[idx];
28     }
29     pli dist(const PNT &i)const
30     {
31         return pli(-(sqr(x[0]-i.x[0])+sqr(x[1]-i.x[1])),lb);
32     }
33 }a[MAXX],the[MAX],p;
34
35 #define mid (l+r>>1)
36 #define lson (id<<1)
37 #define rson (id<<1|1)
38 #define lc lson,l,mid-1
39 #define rc rson,mid+1,r
40 int n,m;
41
42 long long rg[MAX][2][2];
43
44 void make(int id=1,int l=1,int r=n,int d=0)
45 {
46     the[id].lb=-1;
47     rg[id][0][0]=rg[id][1][0]=inf;
48     rg[id][0][1]=rg[id][1][1]=-inf;
49     if(l>r)
50         return;
51     idx=d;
52     std::nth_element(a+l,a+mid,a+r+1);
53     the[id]=a[mid];
54     rg[id][0][0]=rg[id][0][1]=the[id].x[0];
55     rg[id][1][0]=rg[id][1][1]=the[id].x[1];
56     make(lc,d*1);
57     make(rc,d*1);
58
59     rg[id][0][0]=std::min(rg[id][0][0],std::min(rg[lson][0][0],rg[rson][0][0]));
60     rg[id][1][0]=std::min(rg[id][1][0],std::min(rg[lson][1][0],rg[rson][1][0]));
61
62     rg[id][0][1]=std::max(rg[id][0][1],std::max(rg[lson][0][1],rg[rson][0][1]));
63     rg[id][1][1]=std::max(rg[id][1][1],std::max(rg[lson][1][1],rg[rson][1][1]));
64 }
65
66 inline long long cal(int id)
67 {
68     static long long a[2];
69     static int i;
70     for(i=0;i<2;i++)
71         a[i]=std::max(abs(p.x[i]-rg[id][i][0]),abs(p.x[i]-rg[id][i][1]));
72     return sqr(a[0])+sqr(a[1]);
73 }
74
75 std::priority_queue<pli>ans;
76
77 void query(const int id=1,const int d=0)
78 {
79     if(the[id].lb<0)
80         return;
81     pli tmp(the[id].dist(p));
82     int a(lson),b(rson);
83     if(p.x[d]<=the[id].x[d])
84         std::swap(a,b);
85     if(ans.size()<n)
86         ans.push(tmp);
87     else
88         if(tmp<ans.top())
89         {
90             ans.push(tmp);
91             ans.pop();

```

```

92     }
93     if(ans.size()<n || cal(a)>=ans.top().first)
94         query(a,d^1);
95     if(ans.size()<n || 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-->0)
112     {
113         scanf("%lld%lld",&p.x[0],&p.x[1]);
114         scanf("%d",&n);
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
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 - 1] < max)
45             {
46                 max = c[a - 1];
47                 ret = d[a - 1];
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base 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]]] & 0xffff] = order[i];
62     memmove(order, tb, n * sizeof(int));
63     memset(ta, 0, sizeof(ta));
64     for(int i = 0; i < n; i++) ta[p[i] >> 16]++;
65     for(int i = 0; i < 65535; i++) ta[i + 1] += ta[i];
66     for(int i = n - 1; i >= 0; i--) tb[--ta[p[order[i]] >> 16]] = order[i];
67     memmove(order, tb, n * sizeof(int));
68 }
69
70 int work(int ii) //求每个点在一个方向上最近的点
71 {
72     for(int i = 0; i < n; i++) //排序前的准备工作
73     {
74         a[i] = y[i] - x[i] + srange;
75         b[i] = srange - y[i];
76         order[i] = i;
77     }
78     radixsort(b); //排序
79     radixsort(a);
80     for(int i = 0; i < n; i++)
81     {
82         torder[i] = order[i];
83         order[i] = i;
84     }
85     radixsort(a); //为线段树而做的排序
86     radixsort(b);
87     for(int i = 0; i < n; i++)
88     {
89         Index[order[i]] = i; //取反, 求orderIndex
90     }
91     for(int i = 1; i < ra + n; i++) c[i] = 0x7fffffff; //线段树初始化
92     memset(d, 0xff, sizeof(d));

```

```

93     for(int i = 0; i < n; i++) //线段树插入删除调用
94     {
95         int tt = torder[i];
96         road[tt][ii] = find(Index[tt]);
97         insert(Index[tt], y[tt] + x[tt], tt);
98     }
99 }
100
101 int distanc(int a, int b) //求两点的距离, 之所以少一个是因为编译器不让使用作为函数名
102 {
103     return abs(x[a] - x[b]) + abs(y[a] - y[b]);
104 }
105
106 int ttb[400000]; //边排序的临时变量
107 int rx[400000], ry[400000], rd[400000]; //边的存储
108 int rr = 0;
109
110 int radixsort_2(int *p) //还是基数排序, copy+的产物paste
111 {
112     memset(ta, 0, sizeof(ta));
113     for(int i = 0; i < rr; i++) ta[p[i] & 0xffff]++;
114     for(int i = 0; i < 65535; i++) ta[i + 1] += ta[i];
115     for(int i = rr - 1; i >= 0; i--) ttb[--ta[p[order[i]]] & 0xffff] = order[i];
116     memmove(order, ttb, rr * sizeof(int));
117     memset(ta, 0, sizeof(ta));
118     for(int i = 0; i < rr; i++) ta[p[i] >> 16]++;
119     for(int i = 0; i < 65535; i++) ta[i + 1] += ta[i];
120     for(int i = rr - 1; i >= 0; i--) ttb[--ta[p[order[i]] >> 16]] = order[i];
121     memmove(order, ttb, rr * sizeof(int));
122 }
123
124 int father[100000], rank[100000]; //并查集
125 int findfather(int x) //并查集寻找代表元
126 {
127     if(father[x] != -1)
128         return (father[x] = findfather(father[x]));
129     else return x;
130 }
131
132 long long kruskal() //最小生成树
133 {
134     rr = 0;
135     int tot = 0;
136     long long ans = 0;
137     for(int i = 0; i < n; i++) //得到边表
138     {
139         for(int j = 0; j < 4; j++)
140         {
141             if(road[i][j] != -1)
142             {
143                 rx[rr] = i;
144                 ry[rr] = road[i][j];
145                 rd[rr++] = distanc(i, road[i][j]);
146             }
147         }
148     }
149     for(int i = 0; i < rr; i++) order[i] = i; //排序
150     radixsort_2(rd);
151     memset(father, 0xff, sizeof(father)); //并查集初始化
152     memset(rank, 0, sizeof(rank));
153     for(int i = 0; i < rr; i++) //最小生成树标准算法kruskal
154     {
155         if(tot == n - 1) break;
156         int t = order[i];
157         int x = findfather(rx[t]), y = findfather(ry[t]);
158         if(x != y)
159         {
160             ans += rd[t];
161             tot++;
162             int &rkx = rank[x], &rky = rank[y];
163             if(rkx > rky) father[y] = x;
164             else
165             {
166                 father[x] = y;
167                 if(rkx == rky) rky++;
168             }
169         }
170     }
171     return ans;
172 }
173
174 int casenum = 0;
175
176 int main()
177 {
178     while(cin >> n)
179     {
180         if(n == 0) break;
181         for(int i = 0; i < n; i++)
182             scanf("%d%d", &x[i], &y[i]);
183         memset(road, 0xff, sizeof(road));
184         for(int i = 0; i < 4; i++) //为了减少编程复杂度, work()函数只写了一种, 其他情况用转换坐标的方式类似处理
185         {
186             //为了降低算法复杂度, 只求出个方向的边4
187             if(i == 2)
188             {
189                 for(int j = 0; j < n; j++) swap(x[j], y[j]);
190             }
191             if((i & 1) == 1)
192             {
193                 for(int j = 0; j < n; j++) x[j] = srange - x[j];
194             }
195             work(i);
196             printf("Case %d: Total Weight = %d\n", ++casenum);
197             cout << kruskal() << endl;
198         }
199         return 0;
200     }

```

2.12 others

```

1  eps如果
2
3  sqrt(a), asin(a), acos(a) 中的是你自己算出来并传进来的, 那就得小心了。如果本来应该是的, 由于浮点误差,
   可能实际是一个绝对值很小的负数(比如aa0-1e), 这样-12sqrt(a)应得的, 直接因0不在定义域而出错。类似地, 如果本来应该是±aa1则, asin(a), acos(a)也有可能出错。因此, 对于此种函数, 必需事先对进行校正。a现在考虑一种情况, 题目要求输出保留两位小数。有个的正确答案的精确值是
4
5  case0按理应该输出, 但你的结果可能是恭喜:005.0:010:005000000001(), 也有可能是悲剧0:004999999999(), 如果按照printf("%.2lf", a)输出, 那你的遭遇将和括号里的字相同。如果为正, 则输出
6  aa + eps, 否则输出a - eps不要输出
7

```

```
8 -0.000注意的数据范围
9
10 double
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三角函数
18
19
20
21 cos/sin/tan 输入弧度
22 acos 输入, 输出 [-1,+1][0,]
23 asin 输入, 输出 [-1,+1][-/2,+/2]
24 atan 输出 [-/2,+/2]
25 atan2 输入(y,x)注意顺序()返回,tan(y/x) 。 ,[,+]都是零的时候会发发生除零错误xy
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: (—) 当尾数小于或等于时, 直接将尾数舍去。
39 4 (二) 当尾数大于或等于时, 将尾数舍去并向前一位进位。
40 6 (三) 当尾数为, 而尾数后面的数字均为时, 应看尾数 “” 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 “” 在此时应被视为偶数。
41 5050 (四) 当尾数为 “” 的后面还有任何不是的数字时, 无论前一位在此时为奇数还是偶数, 也无论 “” 后面不为的数字在哪一位上, 都应向前进一位。
42
43 55050
44
45 rotate mat:
46 [ cos(theta) -sin(theta) ]
[ sin(theta) cos(theta) ]
```

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 + b/2 - . 1取格点的组成图形的面积为—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 = 2i + b - . 2
```

2.14 PointInPoly

```
1 /*射线法
2 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inpoly
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sidex
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             if (OnSeg(side.e, ray))
39             {
40                 if (side.e.y > side.s.y)
41                     count++;
42             }
43         else
44             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         t1=pnt[(i+1)%n]-pnt[i];
9         while(abs(t1.cross(pnt[(i+1)%n]-pnt[i]))>=abs(t1.cross(pnt[1]-pnt[i])))
10             l=(i+1)%n;
11         ans=std::max(ans,std::max(dist(pnt[1],pnt[i]),dist(pnt[1],pnt[(i+1)%n])));
12     }
13     return ans;
```

```
14 }
15
16 //两凸包最近距离
17 double go()
18 {
19     sq=sp=0;
20     for(i=1;i<ch[1].size();i++)
21         if(ch[1][sq]<ch[1][i])
22             sq=i;
23
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
47         {
48             if(len<eps)
49             {
50                 ans=std::min(ans,p2l(b1,a1,a2));
51                 sp=(sp+1)%ch[0].size();
52             }
53             else
54             {
55                 ans=std::min(ans,p2l(a1,b1,b2));
56                 sq=(sq+1)%ch[1].size();
57             }
58         }
59     }while(tp!=sp || tq!=sq);
60     return ans;
61 }
62
63 //外接矩形 by mzry
64 inline void solve()
65 {
66     resa = resb = 1e100;
67     double dis1,dis2;
68     Point xp[4];
69     Line l[4];
70     int a,b,c,d;
71     int sa,sb,sc,sd;
72     a = b = c = d = 0;
73     sa = sb = sc = sd = 0;
74     Point va,vb,vc,vd;
75     for (a = 0; a < n; a++)
76     {
77         va = Point(p[a],p[(a+1)%n]);
78         vb = Point(-va.x,-va.y);
79         vc = Point(-va.y,va.x);
80         vd = Point(-vb.x,-vb.y);
81         if (sb < sa)
82         {
83             b = a;
84             sb = sa;
85         }
86         while (xmult(vb,Point(p[b],p[(b+1)%n])) < 0)
87         {
88             b = (b+1)%n;
89             sb++;
90         }
91         if (sc < sb)
92         {
93             c = b;
94             sc = sb;
95         }
96         while (xmult(vc,Point(p[c],p[(c+1)%n])) < 0)
97         {
98             c = (c+1)%n;
99             sc++;
100         }
101         if (sd < sc)
102         {
103             d = c;
104             sd = sc;
105         }
106         while (xmult(vd,Point(p[d],p[(d+1)%n])) < 0)
107         {
108             d = (d+1)%n;
109             sd++;
110         }
111
112         //卡在 p[a],p[b],p[c],p[d] 上
113         sa++;
114     }
115
116 //合并凸包给定凸多边形
117 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:
118
119 (p(i) , q(j)) 形成一个并踵点对。
120 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画出并且顶点是以顺时针序排列, 算法如下: 、 分别计算
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。、 构造这些点的逐平行线,
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
127 534、 所有可能的桥此时都已经确定了。
128 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。
129
130 156 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。
131
132 //临界切线、计算
133 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)。yminxQ、 为多边形在
134 2 yminP 和 yminxQ 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 yminxQ 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, 执行以下
    | 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。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。
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 [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 //precisioness 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 &x)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=(a+b+c)/2
3 | area=sqrt(p*(p-a)*(p-b)*(p-c));
4 | area=a*b*sin(C)/2;
5 | area=sq(a)*sin(B)*sin(C)/2/sin(B+C);
6 | area=sq(a)/2/(cot(B)+cot(C));
7 |
8 | centroid:
9 |     center of mass
10 |     intersection of triangle's three triangle medians
11 |
12 | Trigonometric conditions:
13 | tan(A/2)*tan(B/2)+tan(B/2)*tan(C/2)+tan(A/2)*tan(C/2)=1
14 | sq(sin(A/2))+sq(sin(B/2))+sq(sin(C/2))+2*sin(A/2)*sin(B/2)*sin(C/2)=1
15 |
16 | Circumscribed circle:
17 | diameter=a*b*c/(2*area);
18 | diameter=sqrt(2*area/sin(A)/sin(B)/sin(c));
19 | diameter=a/sin(A)=b/sin(B)=c/sin(C);
20 |
21 | Incircle:
22 | inradius=2*area/(a+b+c);
23 | coordinates(x,y)=a*{xa,ya}/(a+b+c)+b*{xb,yb}/(a+b+c)+c*{xc,yc}/(a+b+c);
24 |
25 | Excircles:
26 | radius[a]=2*area/(b+c-a);
27 | radius[b]=2*area/(a+c-b);
28 | radius[c]=2*area/(a+b-c);
29 |
30 | Steiner circumellipse (least area circumscribed ellipse)
31 | area= area * 4*pi/3/sqrt(3);
32 | center is the triangle's centroid.
33 |
34 | Steiner inellipse ( maximum area inellipse )
35 | area= area * pi/3/sqrt(3);
36 | center is the triangle's centroid.
37 |
38 | Fermat Point: 当有一个内角不小于 ° 时, 费马点为此角对应顶点。
39 | 120当三角形的内角都小于 ° 时
40 |
41 | 120以三角形的每一边为底边, 向外做三个正三角形
42 |
43 | ABC 'BCA' 'CAB' 连接
44 | 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).rotleft())).crosspoint(line(c.
    |             add(b).div(2),c.add(b).div(2).add(b.sub(c).rotleft())));
13 |         r=p.distance(a);
14 |     }
15 |     circle(point a,point b,point c,bool t)//三角形的内切圆
16 |     {
17 |         line u,v;
18 |         double n=atan2(b.y-a.y,b.x-a.x),n=atan2(c.y-a.y,c.x-a.x);
19 |         u.a=a;
20 |         u.b=a.add(point(cos((n+n)/2),sin((n+n)/2)));
21 |         v.a=b;
22 |         n=atan2(a.y-b.y,a.x-b.x),n=atan2(c.y-b.y,c.x-b.x);
23 |         v.b=v.a.add(point(cos((n+n)/2),sin((n+n)/2)));
24 |         p=a.crosspoint(v);
25 |         r=line(a,b).dispointtoseg(p);
26 |     }
27 |     void input()
28 |     {
29 |         p.input();
30 |         scanf("%lf",&r);
31 |     }
32 |     void output()
33 |     {
34 |         printf("%.2lf_%.2lf_%.2lf\n",p.x,p.y,r);
35 |     }
36 |     bool operator==(circle v)
37 |     {
38 |         return ((p==v.p)&&dblcmp(r-v.r)==0);
39 |     }
40 |     bool operator<(circle v)const
41 |     {
```

```

42     return ((p<v.p)||((p==v.p)&&dblcmp(r-v.r)<0);
43 }
44 double area()
45 {
46     return pi*sqr(r);
47 }
48 double circumference()
49 {
50     return 2*pi*r;
51 }
52 //0 圆外
53 //1 圆上
54 //2 圆内
55 int relation(point b)
56 {
57     double dst=b.distance(p);
58     if (dblcmp(dst-r)<0)return 2;
59     if (dblcmp(dst-r)==0)return 1;
60     return 0;
61 }
62 int relationseg(line v)
63 {
64     double dst=v.dispointtoseg(p);
65     if (dblcmp(dst-r)<0)return 2;
66     if (dblcmp(dst-r)==0)return 1;
67     return 0;
68 }
69 int relationline(line v)
70 {
71     double dst=v.dispointtoline(p);
72     if (dblcmp(dst-r)<0)return 2;
73     if (dblcmp(dst-r)==0)return 1;
74     return 0;
75 }
76 //过a 两点b 半径的两个圆r
77 int getcircle(point a,point b,double r,circle &c1,circle &c2)
78 {
79     circle x(a,r),y(b,r);
80     int t=x.pointcrosscircle(y,c1.p,c2.p);
81     if (!t)return 0;
82     c1.r=c2.r=t;
83     return t;
84 }
85 //与直线相切u 过点q 半径的圆r1
86 int getcircle(line u,point q,double r1,circle &c1,circle &c2)
87 {
88     double dis=u.dispointtoline(q);
89     if (dblcmp(dis-r1*2)>0)return 0;
90     if (dblcmp(dis)==0)
91     {
92         c1.p=q.add(u.b.sub(u.a).rotleft().trunc(r1));
93         c2.p=q.add(u.b.sub(u.a).rotright().trunc(r1));
94         c1.r=c2.r=r1;
95         return 2;
96     }
97     line u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),u.b.add(u.b.sub(u.a).
98         rotright().trunc(r1)));
99     line u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),u.b.add(u.b.sub(u.a).
100         rotright().trunc(r1)));
101     circle c=circle(q,r1);
102     point p1,p2;
103     if (!cc.pointcrossline(u1,p1,p2))cc.pointcrossline(u2,p1,p2);
104     c1=circle(p1,r1);
105     if (p1==p2)
106     {
107         c2=c1;return 1;
108     }
109     c2=circle(p2,r1);
110     return 2;
111 }
112 //同时与直线u,相切v 半径的圆r1
113 int getcircle(line u,line v,double r1,circle &c1,circle &c2,circle &c3,circle &c4)
114 {
115     if (u.parallel(v))return 0;
116     line u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),u.b.add(u.b.sub(u.a).
117         rotright().trunc(r1)));
118     line u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),u.b.add(u.b.sub(u.a).
119         rotright().trunc(r1)));
120     line v1=line(v.a.add(v.b.sub(v.a).rotleft().trunc(r1)),v.b.add(v.b.sub(v.a).
121         rotright().trunc(r1)));
122     line v2=line(v.a.add(v.b.sub(v.a).rotright().trunc(r1)),v.b.add(v.b.sub(v.a).
123         rotright().trunc(r1)));
124     c1.r=c2.r=c3.r=c4.r=r1;
125     c1.p=u1.crosspoint(v1);
126     c2.p=u1.crosspoint(v2);
127     c3.p=u2.crosspoint(v1);
128     c4.p=u2.crosspoint(v2);
129     return 4;
130 }
131 //同时与不相交圆cx,相切cy 半径为的圆r1
132 int getcircle(circle cx,circle cy,double r1,circle &c1,circle &c2)
133 {
134     circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
135     int t=x.pointcrosscircle(y,c1.p,c2.p);
136     if (!t)return 0;
137     c1.r=c2.r=t;
138     return t;
139 }
140 int pointcrossline(line v,point &p1,point &p2)//求与线段交要先判断relationseg
141 {
142     if (!(*this).relationline(v))return 0;
143     point a=v.lineprog(p);
144     double d=v.dispointtoline(p);
145     d=sqrt(r*r-d*d);
146     if (dblcmp(d)==0)
147     {
148         p1=a;
149         p2=a;
150         return 1;
151     }
152     p1=a.sub(v.b.sub(v.a).trunc(d));
153     p2=a.add(v.b.sub(v.a).trunc(d));
154     return 2;
155 }
156 //5 相离
157 //4 外切
158 //3 相交
159 //2 内切
160 //1 内含
161 int relationcircle(circle v)
162 {
163     double d=p.distance(v.p);
164     if (dblcmp(d-r-v.r)>0)return 5;
165     if (dblcmp(d-r-v.r)==0)return 4;
166     double l=fabs(r-v.r);
167     if (dblcmp(d-r-v.r)<0&&dblcmp(d-l)>0)return 3;
168     if (dblcmp(d-l)==0)return 2;
169     if (dblcmp(d-l)<0)return 1;
170 }
171 int pointcrosscircle(circle v,point &p1,point &p2)
172 {
173     int rel=relationcircle(v);
174     if (rel==1||rel==5)return 0;
175     double d=p.distance(v.p);
176     double l=(d+(sqr(r)-sqr(v.r))/d)/2;
177     double h=sqrt(sqr(r)-sqr(l));
178     p1=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotleft().trunc(h)));
179     p2=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotright().trunc(h)));
180     if (rel==2||rel==4)
181     {
182         return 1;
183     }
184     return 2;
185 }
186 //过一点做圆的切线 先判断点和圆关系()
187 int tangentline(point q,line &a,line &b)
188 {
189     int x=relation(q);
190     if (x==2)return 0;
191     if (x==1)
192     {
193         v=line(q,q.add(q.sub(p).rotleft()));
194         v=a;
195         return 1;
196     }
197     double d=p.distance(q);
198     double l=sqr(r)/d;
199     double h=sqrt(sqr(r)-sqr(l));
200     v=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotleft().trunc(h))));
201     v=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotright().trunc(h))));
202     return 2;
203 }
204 double areacircle(circle v)
205 {
206     int rel=relationcircle(v);
207     if (rel==4)return 0.0;
208     if (rel<=2)return min(area(),v.area());
209     double d=p.distance(v.p);
210     double hf=(r+v.r+d)/2.0;
211     double ss=2*sqr(hf*(hf-r)*(hf-v.r)*(hf-d));
212     double a1=acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
213     a1=a1*r*r;
214     double a2=acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
215     a2=a2*v.r*v.r;
216     return a1+a2-ss;
217 }
218 double areatriangle(point a,point b)
219 {
220     if (dblcmp(p.sub(a).det(p.sub(b))==0))return 0.0;
221     point q[5];
222     int len=0;
223     q[len++]=a;
224     line l(a,b);
225     point p1,p2;
226     if (pointcrossline(l,q[1],q[2])==2)
227     {
228         if (dblcmp(a.sub(q[1]).dot(b.sub(q[1]))<0)q[len++]=q[1];
229         if (dblcmp(a.sub(q[2]).dot(b.sub(q[2]))<0)q[len++]=q[2];
230     }
231     q[len++] = b;
232     if (len==4&&dblcmp(q[0].sub(q[1]).dot(q[2].sub(q[1]))>0)swap(q[1],q[2]);
233     double res=0;
234     int i;
235     for (i=0;i<len-1;i++)
236     {
237         if (relation(q[i])==0||relation(q[i+1])==0)
238         {
239             double arg=p.rad(q[i],q[i+1]);
240             res+=r*r*arg/2.0;
241         }
242         else
243         {
244             res+=fabs(q[i].sub(p).det(q[i+1].sub(p)))/2.0;
245         }
246     }
247     return res;
248 }
249 };

```

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]次的面积i
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),fa=acos((ac*ac+ab*ab-bc*bc)/(2.0*ac*ab));
75             double a0=th-fa;
76             if (dblcmp(a0+pi)<0)a0+=2*pi;
77             double a1=th+fa;
78             if (dblcmp(a1-pi)>0)a1-=2*pi;
79             if (dblcmp(a0-a1)>0)
80             {
81                 v.push_back(make_pair(a0,1));
82                 v.push_back(make_pair(pi,-1));
83                 v.push_back(make_pair(-pi,1));
84                 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]))
100                    .det(point(c[i].p.x+c[i].r*cos(v[j].first),c[i].p.y+c[i].r*sin(v[j].first)));
101             }
102             cur=v[j].second;
103             pre[cur]=v[j].first;
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++]=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)))
64        <0)ed--;
65     while (st<ed&&dblcmp(hp[que[ed]].b.sub(hp[que[ed]].a).det(p[st+1].sub(hp[que[ed]].a)))
66        <0)st++;
67     if (st+1==ed)return false;
68     return true;
69 }
70 void getconvex(polygon &con)
71 {
72     p[st]=hp[que[st]].crosspoint(hp[que[ed]]);
73     con.m=st+1;
74     int j=st,i=0;
75     for (;i<=ed;i++,j++)
76     {
77         con.p[i]=p[j];
78     }
79 };

```

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(a).dot(p.sub(b)))<=0;
80    }
81    bool parallel(line v)
82    {
83        return dblcmp(b.sub(a).det(v.b.sub(v.a)))==0;
84    }
85    //2 规范相交
86    //1 非规范相交
87    //0 不相交
88    int segcrossseg(line v)
89    {
90        int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
91        int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
92        int d3=dblcmp(v.b.sub(v.a).det(a.sub(v.a)));
93        int d4=dblcmp(v.b.sub(v.a).det(b.sub(v.a)));
94        if ((d1*d2)==2&&(d3*d4)==2)return 2;
95        return (d1==0&&dblcmp(v.a.sub(a).dot(v.a.sub(b)))<=0||
96            d2==0&&dblcmp(v.b.sub(a).dot(v.b.sub(b)))<=0||

```

```

97         d3==0&&dblcmp(a.sub(v.a).dot(a.sub(v.b)))<=0||
98         d4==0&&dblcmp(b.sub(v.a).dot(b.sub(v.b)))<=0);
99     }
100     int linecrossseg(line v)/*this seg v line
101     {
102         int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
103         int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
104         if ((d1^d2)==2)return 2;
105         return (d1==0|d2==0);
106     }
107     //0 平行
108     //1 重合
109     //2 相交
110     int linecrossline(line v)
111     {
112         if ((*this).parallel(v))
113         {
114             return v.relation(a)==3;
115         }
116         return 2;
117     }
118     point crosspoint(line v)
119     {
120         double a1=v.b.sub(v.a).det(a.sub(v.a));
121         double a2=v.b.sub(v.a).det(b.sub(v.a));
122         return point((a.x*a2-b.x*a1)/(a2-a1),(a.y*a2-b.y*a1)/(a2-a1));
123     }
124     double dispointtoline(point p)
125     {
126         return fabs(p.sub(a).det(b.sub(a)))/length();
127     }
128     double dispointtoseg(point p)
129     {
130         if (dblcmp(p.sub(b).dot(a.sub(b)))<0|dblcmp(p.sub(a).dot(b.sub(a)))<0)
131         {
132             return min(p.distance(a),p.distance(b));
133         }
134         return dispointtoline(p);
135     }
136     point lineprog(point p)
137     {
138         return a.add(b.sub(a).mul(b.sub(a).dot(p.sub(a))/b.sub(a).len2()));
139     }
140     point symmetrpoint(point p)
141     {
142         point q=lineprog(p);
143         return point(2*q.x-p.x,2*q.y-p.y);
144     }
145 };

```

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.sub(p).dot(b.sub(p)))<=0;
26     }
27     double dispointtoline(point3 p)
28     {
29         return b.sub(a).det(p.sub(a)).len()/a.distance(b);
30     }
31     double dispointtoseg(point3 p)
32     {
33         if (dblcmp(p.sub(b).dot(a.sub(b)))<0|dblcmp(p.sub(a).dot(b.sub(a)))<0)
34         {
35             return min(p.distance(a),p.distance(b));
36         }
37         return dispointtoline(p);
38     }
39     point3 lineprog(point3 p)
40     {
41         return a.add(b.sub(a).trunc(b.sub(a).dot(p.sub(a))/b.distance(a)));
42     }
43     point3 rotate(point3 p,double ang)/*绕此向量逆时针角度pang
44     {
45         if (dblcmp((p.sub(a).det(p.sub(b)).len())==0)return p;
46         point3 f1=b.sub(a).det(p.sub(a));
47         point3 f2=b.sub(a).det(f1);
48         double len=fabs(a.sub(p).det(b.sub(p)).len()/a.distance(b));
49         f1=f1.trunc(len);f2=f2.trunc(len);
50         point3 h=p.add(f2);
51         point3 pp=h.add(f1);
52         return h.add((p.sub(h)).mul(cos(ang*1.0)).add((pp.sub(h)).mul(sin(ang*1.0)));
53     }
54 };

```

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 &a)/*平面和平面的交线
92    {
93        point3 oo=p.det(f.o);
94        point3 vo=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 maxq=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).dot(b.sub(p))));
84 }
85 point trunc(double r)
86 {
87     double l=len();
88     if (!dblcmp(l))return *this;
89     r/=l;
90     return point(x*r,y*r);
91 }
92 point rotleft()
93 {
94     return point(-y,x);
95 }
96 point rotright()
97 {
98     return point(y,-x);
99 }
100 point rotate(point p,double angle)//绕点逆时针旋转角度pangle
101 {
102     point v=this->sub(p);
103     double c=cos(angle),s=sin(angle);
104     return point(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
105 }
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("%d%d%d",&x,&y,&z);
10    }
11    void output()
12    {
13        printf("%21f_%21f_%21f\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)==0;
18    }
19    bool operator<(point3 a)const
20    {
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 &a,const point &b)
33        {
34            point a=a,b=b;
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&&u<0&&v>=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 convexcut(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     {
235         ans+=c.areatriangle(p[i],p[j]);
236     }
237     {
238         ans=c.areatriangle(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     sort(v.begin(),v.end());
322     int cur=0;
323     for (j=0;j<v.size();j++)
324     {
325         if (v[j].second==1)+cur;
326         else -cur;
327         ans=max(ans,cur);
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(q.sub(p[0]).det(p[mid+1].sub(p[0]))<0)
344         {
345             polygon c;
346             c.p[0]=p[mid];
347             c.p[1]=p[mid+1];
348             c.p[2]=p[0];
349             c.n=3;
350             if (c.relationpoint(q))return mid;
351             return -1;
352         }
353         if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0]))>0)
354         {
355             low=mid+1;
356         }
357         else
358         {
359             high=mid-1;
360         }
361     }

```

```

360     }
361     }
362     return -1;
363 }
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.y-s.y);
20        r=min(r,1.0);r=max(r,0.0);
21        e.pb(mp(r,i));
22    }
23    double polyareaunion()
24    {
25        double ans=0.0;
26        int c0,c1,c2,i,j,k,w;
27        for (i=0;i<p.size();i++)
28        {
29            if (p[i].getdir()==0)reverse(p[i].p,p[i].p+p[i].n);
30        }
31        for (i=0;i<p.size();i++)
32        {
33            for (k=0;k<p[i].n;k++)
34            {
35                point &s=p[i].p[k],&t=p[i].p[(k+1)%p[i].n];
36                if (!dblcmp(s.det(t)))continue;
37                e.clear();
38                e.pb(mp(0.0,1));
39                e.pb(mp(1.0,-1));
40                for (j=0;j<p.size();j++)if (i!=j)
41                {
42                    for (w=0;w<p[j].n;w++)
43                    {
44                        point &p=p[j].p[w],&b=p[j].p[(w+1)%p[j].n],&c=p[j].p[(w+1+p[j].n)%p[j].n];
45                        c0=dblcmp(t.sub(s).det(c.sub(s)));
46                        c1=dblcmp(t.sub(s).det(a.sub(s)));
47                        c2=dblcmp(t.sub(s).det(b.sub(s)));
48                        if (c1*c2<0)ins(s,t,line(s,t).crosspoint(line(a,b)),-c2);
49                        else if (!c1&&c0*c2<0)ins(s,t,a,-c2);
50                        else if (!c1&&c2)
51                        {
52                            int c3=dblcmp(t.sub(s).det(p[j].p[(w+2)%p[j].n].sub(s)));
53                            int dp=dblcmp(t.sub(s).dot(b.sub(a)));
54                            if (dp&&c0)ins(s,t,a,dp>0?c0*((j>i)^(c0<0)):(-c0<0));
55                            if (dp&&c3)ins(s,t,b,dp>0?c3*((j>i)^(c3<0)):c3<0);
56                        }
57                    }
58                }
59                sort(e.begin(),e.end());
60                int ct=0;
61                double tot=0.0,last;
62                for (j=0;j<e.size();j++)
63                {
64                    if (ct==p.size())tot+=e[j].first-last;
65                    ct+=e[j].second;
66                    last=e[j].first;
67                }
68                ans+=s.det(t)*tot;
69            }
70        }
71        return fabs(ans)*0.5;
72    }
73 };

```

4 graph

4.1 2SAT

```

1 /*
2  x & y == true:
3  -x -> x
4  -y -> y
5
6  x & y == false:
7  x -> -y
8  y -> -x
9
10 x | y == true:
11 -x -> y
12 -y -> x
13
14 x | y == false:
15 x -> -x
16 y -> -y
17
18 x ^ y == true:
19 -x -> y
20 y -> -x
21 x -> -y
22 -y -> x
23
24 x ^ y == false:
25 x -> y
26 y -> x
27 -x -> -y
28 -y -> -x
29 */
30 #include<cstdio>
31 #include<cstring>
32
33 #define MAXX 16111
34 #define MAXE 200111

```

```

35 #define v to[i]
36
37 int edge[MAXX],to[MAXE],nxt[MAXE],cnt;
38 inline void add(int a,int b)
39 {
40     nxt[++cnt]=edge[a];
41     edge[a]=cnt;
42     to[cnt]=b;
43 }
44
45 bool done[MAXX];
46 int st[MAXX];
47
48 bool dfs(const int now)
49 {
50     if(done[now^1])
51         return false;
52     if(done[now])
53         return true;
54     done[now]=true;
55     st[cnt++]=now;
56     for(int i=edge[now];i;i=nxt[i])
57         if(!dfs(v))
58             return false;
59     return true;
60 }
61
62 int n,m;
63 int i,j,k;
64
65 inline bool go()
66 {
67     memset(done,0,sizeof done);
68     for(i=0;i<n;i+=2)
69         if(!done[i] && !done[i^1])
70         {
71             cnt=0;
72             if(!dfs(i))
73             {
74                 while(cnt)
75                     done[st[--cnt]]=false;
76                 if(!dfs(i^1))
77                     return false;
78             }
79         }
80     return true;
81 }
82 //done array will be a solution with minimal lexicographical order
83 // or maybe we can solve it with dual SCC method, and get a solution by reverse the
    edges of DAG then product a topsort

```

4.2 Articulation

```

1 void dfs(int now,int fa) // 从开始now1
2 {
3     int p(0);
4     dfn[now]=low[now]=cnt++;
5     for(std::list<int>::const_iterator it(edge[now].begin());it!=edge[now].end();++it)
6         if(dfn[*it]==-1)
7         {
8             dfs(*it,now);
9             ++p;
10            low[now]=std::min(low[now],low[*it]);
11            if((now==1 && p>1) || (now!=1 && low[*it]>=dfn[now])) // 如果从出发点出发的子节
                点不能由兄弟节点到达, 那么出发点为割点。如果现节点不是出发点, 但是其子孙节点不能达到
                祖先节点, 那么该节点为割点
                ans.insert(now);
        }
        else
            if(*it!=fa)
                low[now]=std::min(low[now],dfn[*it]);
    }
}

```

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<n;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",&n&&k);
26         while (k--){
27             scanf("%d",&t);
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<stdio>
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         else
49             if(col[v]==1)
50                 low[now]=std::min(low[now], dfn[v]);
51     }
52     col[now]=2;
53     if(dfn[now]==low[now])
54     {
55         ++bcnt;
56         static int x;
57         do
58         {
59             x=st.top();
60             st.pop();
61             belong[x]=bcnt;
62         } while(x!=now);
63     }
64 }
65
66 std::set<int> set[MAXX];
67
68 int dist[MAXX];
69 std::queue<int> q;
70 int n, m, i, j, k;
71
72 inline int go(int s)
73 {
74     static std::set<int>::const_iterator it;
75     memset(dist, 0x3f, sizeof dist);
76     dist[s]=0;
77     q.push(s);
78     while(!q.empty())
79     {
80         s=q.front();
81         q.pop();
82         for(it=set[s].begin(); it!=set[s].end(); ++it)
83             if(dist[*it]>dist[s]+1)
84             {
85                 dist[*it]=dist[s]+1;
86                 q.push(*it);
87             }
88     }
89     return std::max_element(dist+1, dist+1+bcnt)-dist;
90 }
91
92 int main()
93 {
94     while(scanf("%d%d", &n, &m), (n|m))
95     {
96         cnt=0;
97         memset(edge, 0, sizeof edge);
98         while(m--)
99         {
100             scanf("%d%d", &i, &j);
101             add(i, j);
102             add(j, i);
103         }
104
105         memset(dfn, 0, sizeof dfn);
106         memset(belong, 0, sizeof belong);
107         memset(low, 0, sizeof low);
108         memset(col, 0, sizeof col);
109         bcnt=idx=0;
110         while(!st.empty())
111             st.pop();
112
113         tarjan(1, -1);
114         for(i=1; i<=bcnt; ++i)
115             set[i].clear();
116         for(i=1; i<=n; ++i)
117             for(j=edge[i]; j; j=nxt[j])
118                 set[belong[i]].insert(belong[to[j]]);
119         for(i=1; i<=bcnt; ++i)
120             set[i].erase(i);
121
122         /*
123          printf("%d\n", dist[go(1)]);
124          for(i=1; i<=bcnt; ++i)
125              printf("%d\n", dist[i]);
126          puts("");
127          */
128         printf("%d\n", bcnt-1-dist[go(1)]);
129     }
130 }

```

```

129     return 0;
130 }

```

4.5 Biconnected Component

```

1 #include<stdio>
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<stdio>
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<x+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<x+i+y)
42                 if (map[x][y] && m[y]!=y && vis[y]!=1)
43                     {
44                         if (vis[y]==1)
45                             if (m[y]==1)
46                                 {
47                                     for (i=0;i+k<p[x].size();i+=2)
48                                         {
49                                             m[p[x][i]]=p[x][i+1];
50                                             m[p[x][i+1]]=p[x][i];
51                                         }
52                                             m[x]=y;
53                                             m[y]=x;
54                                             return true;
55                                         }
56                                     else
57                                     {
58                                         p[z=m[y]]=p[x];
59                                         p[z].push_back(y);
60                                         p[z].push_back(z);
61                                         vis[y]=1;
62                                         vis[z]=0;
63                                         *qb+=z;
64                                     }
65                                     else
66                                     {
67                                         for (b=0;b<p[x].size() && b<p[y].size() && p[x][b]==p[y][b];b++)
68                                             --b;
69                                         label(x,y,b);
70                                         label(y,x,b);
71                                     }
72                                 }
73         return false;
74     }
75
76     int i,j,k;
77     int ans;
78
79     int main()
80     {
81         scanf("%d",&n);
82         for (i=0;i<x+i)
83             p[i].reserve(n);
84         while (scanf("%d%d",&i,&j)!=EOF)
85             {
86                 --i;
87                 --j;
88                 map[i][j]=map[j][i]=true;
89             }
90         memset(m,-1,sizeof m);
91         for (i=0;i<x+i)
92             if (m[i]==1)
93                 {
94                     if (bfs(i))
95                         ++ans;
96                     else
97                         m[i]=i;
98                 }
99         printf("%d\n",ans<<1);
100         for (i=0;i<x+i)
101             if (i<n[i])
102                 printf("%d%d\n",i+1,m[i]+1);
103         return 0;
104     }

```

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         if (dfn[edge[now][i]]==1)
6             {
7                 dfs(edge[now][i],now);
8                 low[now]=std::min(low[now],low[edge[now][i]]);
9                 if (low[edge[now][i]]>dfn[now]) //如果子节点不能够走到父节点之前去那么该边为桥,
10                     {
11                         if (edge[now][i]<now)
12                             {
13                                 j=edge[now][i];
14                                 k=now;
15                             }
16                         else
17                         {
18                             j=now;
19                             k=edge[now][i];
20                         }
21                         ans.push_back(node(j,k));
22                     }
23             }
24         else
25             if (edge[now][i]!=fa)
26                 low[now]=std::min(low[now],low[edge[now][i]]);
27 }

```

4.8 Chu–Liu–Edmonds’ Algorithm

```

1 #include<cstdio>
2 #include<string>
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<x+i)
40             ed.push_back(edge(n,i,sum));
41         ++m;
42         while (true)
43             {
44                 memset(in,0,sizeof in);
45                 for (i=0;i<ed.size();i++)
46                     if (ed[i].a!=ed[i].b && in[ed[i].b]>ed[i].c)
47                         {
48                             in[ed[i].b]=ed[i].c;
49                             pre[ed[i].b]=ed[i].a;
50                             if (ed[i].a==rt)
51                                 j=i;
52                         }
53                 for (i=0;i<x+i)
54                     if (i!=rt && in[i]==inf)
55                         goto ot;
56                 memset(id,-1,sizeof id);
57                 memset(vis,-1,sizeof vis);
58                 tn=in[rt]=0;
59                 for (i=0;i<x+i)
60                     {
61                         ans+=in[i];
62                         for (v=i;vis[v]!=i && id[v]==1 && v!=rt;v=pre[v])
63                             vis[v]=1;
64                         if (v!=rt && id[v]==1)
65                             {
66                                 for (u=pre[v];u!=v;u=pre[u])
67                                     id[u]=tn;
68                                 id[v]=tn++;
69                             }
70             }
71             if (!tn)
72                 break;
73             for (i=0;i<x+i)
74                 if (id[i]==1)
75                     id[i]=tn++;
76             for (i=0;i<ed.size();i++)
77                 {
78                     v=ed[i].b;
79                     ed[i].a=id[ed[i].a];
80                     ed[i].b=id[ed[i].b];
81                     if (ed[i].a!=ed[i].b)
82                         ed[i].c=in[v];
83                 }
84             m=tn;
85             rt=id[rt];
86         }
87         if (ans>=2*sum)
88             puts("impossible");
89         else
90             printf("%d%d\n",ans-sum,j-om);
91         puts("");
92     }
93     return 0;
94 }

```

4.9 Covering problems

1 最大团以及相关知识独立集：独立集是指图的顶点集的一个子集该子集的导出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

2

3

4 。。那么称这个独立集是一个极大独立集一个图中包含顶点数目最多的独立集称为最大独立集。最大独立集一定是极大独立集，但是极大独立集不一定是最大的独立集。支配集：与独立集相对应的就是支配集，支配集也是图顶点集的一个子集，设是图的一个支配集，则对于图中的任意一个顶点，要么属于集合

5

6 SGus，要么与中的顶点相邻。在中除去任何元素后不再是支配集，则支配集是极小支配集。称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。ssssG最小点对边

7

8 ()的覆盖：最小点的覆盖也是图的顶点集的一个子集，如果我们选中一个点，则称这个点将以他为端点的所有边都覆盖了。将图中所有的边都覆盖所用顶点数最少，这个集合就是最小的点的覆盖。最大团：图的顶点的子集，设是最大团，则中任意两点相邻。若，是最大团，则

9

10 GDDuvu有边相连，其补图vu,没有边相连，所以图的最大团G其补图的最大独立集。给定无向图-G=(V;E)，如果属于，并且对于任意UVu包含于V 有<u;v 包含于>，则称是图的完全子图，的完全子图是图的团，当且仅当不包含在的更大的完全子图中，的最大团是指中所含顶点数目最多的团。如果属于，并且对于任意UVu包含于V 有vU<u;v 不包含于>，则称是图的空子图，的空子图是图的独立集，当且仅当不包含在的更大的独立集，的最大团是指中所含顶点数目最多的独立集。EUGUGUGUG性质：最大独立集最小覆盖集

11

12

13 +=V最大团补图的最大独立集

14 =最小覆盖集最大匹配

15 =

16

17 minimum cover:

18 vertex cover vertex bipartite graph = maximum cardinality bipartite matching找完最大二分匹配后，有三种情况要分别处理：甲、

19

20 X 侧未匹配点的交错树们。乙、

21 Y 侧未匹配点的交错树们。丙、层层叠叠的交错环们（包含单独的匹配边）。这三个情况互不干涉。用

```
22
23 Graph Traversal 建立甲、乙的交錯樹們，剩下部分就是丙。要找點覆蓋，甲、乙是取盡奇數距離的點，丙是取盡偶
    數距離的點、或者是取盡奇數距離的點，每塊連通分量可以各自為政。另外，小心處理的話，是可以印出字典
    順序最小的點覆蓋的。已經有最大匹配時，求點覆蓋的時間複雜度等同於一次
24
25 Graph Traversal 的時間。
26
27 vertex cover edge
28
29 edge cover vertex首先在圖上求得一個
30 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 。
31
32 edge cover edge
33
34 path cover vertex
35 general graph: NP-H
36 tree: DP
37 DAG: 将每个节点拆分为入点和出点,ans节点数匹配数=
38
39 path cover edge
40 minimize the count of euler path ( greedy is ok? )
41
42 cycle cover vertex
43 general: NP-H
44 weighted: do like path cover vertex, with KM algorithm
45
46 cycle cover edge
47 NP-H
```

4.10 Difference constraints

```
1 for a - b <= c
2     add(b,a,c); 最短路得最近解最长路得最近解根据情况反转边反转方向及边权
3
4
5
6 //?() 全点得普通解
7
8 0
```

4.11 Dinitz's algorithm

```
1 #include<cstdio>
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],std::min(maxcap,cap[i])))>0
44             {
45                 d=dfs(to[i],std::min(flow,cap[i]));
46                 cap[i]-=d;
47                 cap[i^1]+=d;
48                 flow-=d;
49                 if(!flow)
50                     return maxcap;
51             }
52     return maxcap-flow;
53 }
54
55 int nc,np,m,i,j,k;
56 int ans;
57
58 int main()
59 {
60     while(scanf("%d%d%d%d",&n,&np,&nc,&m)!=EOF)
61     {
62         cnt=0;
63         memset(edge,-1,sizeof edge);
64         while(m--)
65         {
66             while(getchar()!=' ');
67             scanf("%d",&i);
68             while(getchar()!=' ');
69             scanf("%d",&j);
70             while(getchar()!=' ');
71             scanf("%d",&k);
72             if(i!=j)
73             {
74                 ++i;
75                 ++j;
76                 add(i,j,k);
77                 add(j,i,0);
78             }
79         }
80         source=np+n;
81         while(np--)
```

```
{
    while(getchar()!=' ');
    scanf("%d",&i);
    while(getchar()!=' ');
    scanf("%d",&j);
    ++i;
    add(source,i,j);
    add(i,source,0);
}
sink=np+n;
while(nc--)
{
    while(getchar()!=' ');
    scanf("%d",&i);
    while(getchar()!=' ');
    scanf("%d",&j);
    ++i;
    add(i,sink,j);
    add(sink,i,0);
}
ans=0;
while(bfs())
{
    memcpy(w,edge,sizeof edge);
    ans+=dfs(source,inf);
    /*
    while((k=dfs(source,inf)))
        ans+=k;
    */
}
printf("%d\n",ans);
}
return 0;
}
```

4.12 Flow network

```
1 Maximum weighted closure of a graph: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（闭合子图）
2
3 closure每个节点向其所有依赖节点连边，容量
4
5 inf源点向所有正权值节点连边，容量为该权值所有负权值节点向汇点连边，容量为该权值绝对值以上均为有向边最大权
    为
6
7
8
9 sum正权值新图的最小割{}-{}残量图中所有由源点可达的点即为所选子图
10
11
12
13
14 Eulerian circuit:计入度和出度之差无向边任意定向出入度之差为奇数则无解然后构图
15
16
17
18 :原图有向边不变，容量
19 1 // 好像需要在新图中忽略有向边？无向边按之前认定方向，容量
20 1源点向所有度数为正的点连边，容量
21 abs度数/2所有度数为负的点向汇点连边，容量
22 abs度数/2两侧均满流则有解相当于规约为可行流问题注意连通性的
23
24
25 trick终点到起点加一条有向边即可将问题转为问题
26
27 pathcircuit
28
29
30
31 Feasible flow problem:
32 refer Feasible flow problem.cpp由超级源点出发的边全部满流则有解有源汇时，由汇点向源点连边，下界上界
33
34 0即可转化为无源无汇上下界流inf对于每条边
35
36 <a>b cap{u,d, 建边}<ss>b cap(u),<a>st cap(u),<a>b cap(d-u)>
37
38 Maximum flow: 好像也可以二分//将流量还原至原图后，在残量网络上继续完成最大流
39 //直接把和设为原来的，此时输出的最大流即是答案
40 sourcesink不需要删除或者调整
41 t->弧s
42 Minimum flow: 好像也可以二分//建图时先不连汇点到源点的边，新图中完成最大流之后再连原汇至原源的边完成第
    二次最大流，此时
43 t->这条弧的流量即为最小流s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
45 ->弧ts跑流，-> 最后检查超级源点满流情况来一步步确定
46 tips :合并流量、减少边数来加速
47
48
49
50
51 Minimum cost feasible flow problem:
52 TODO看起来像是在上面那样跑费用流就行了……
53
54
55
56
57 Minimum weighted vertex cover edge for bipartite graph:
58 for all vertex in X:
59     edge < s->x cap(weight(x)) >
60 for all vertex in Y:
61     edge < y->t cap(weight(y)) >
62 for original edges
63     edge < x->y cap(inf) >
64
65 ans={maximum flow}={minimum cut}残量网络中的所有简单割
66 ( 源点可达( && 汇点不可达) || 源点不可达( && 汇点可达) )对应着解
67
68
69
70 Maximum weighted vertex independent set for bipartite graph:
71 ans=Sum点权{}-value{Minimum weighted vertex cover edge}解应该就是最小覆盖集的补图吧……方格取数
72
73
74
75
76 : // refer: hdu 3820 golden eggs取方格获得收益当取了相邻方格时付出出边的代价必取的方格到源汇的边的容
    量
77
78
79
80 /inf相邻方格之间的边的容量为代价
81 *2
82 ans=sum方格收益最大流{}-{}最小割的唯一性
83
84
85
```

```

86 : // refer关键边。有向边起点为:集, 终点为集st从源和汇分别能够到的点集是所有点时, 最小割唯一也就是每一
87 增广路径都仅有一条边满流注意查看的是实际的网络, 不是残量网络具体来说
88
89
90
91
92
93 void rr(int now)
94 {
95     done[now]=true;
96     ++cnt;
97     for(int i=edge[now]; i!=-1; i=nxt[i])
98         if(cap[i]&& !done[v])
99             rr(v);
100 }
101
102 void dfs(int now)
103 {
104     done[now]=true;
105     ++cnt;
106     for(int i=edge[now]; i!=-1; i=nxt[i])
107         if(cap[i]&& !done[v])
108             dfs(v);
109 }
110
111 memset(done,0,sizeof done);
112 cnt=0;
113 rr(source);
114 dfs(sink);
115 puts(cnt==n?"UNIQUE":"AMBIGUOUS");
116
117
118
119 Tips:两点间可以不止有一种边, 也可以不止有一条边, 无论有向无向
120 ;两点间容量则可以设法简化为一个点
121 inf;点权始终要转化为边权
122 ;不参与决策的边权设为来排除掉
123 inf;贪心一个初始不合法情况, 然后通过可行流调整
124 ; // refer: 混合图欧拉回路存在性、有向无向图中国邮差问题遍历所有边至少一次后回到原点/()按时间拆点时间
125 |();

```

4.13 Hamiltonian circuit

```

1 //if every point connect with not less than [(N+1)/2] points
2 #include<stdio>
3 #include<algorithm>
4 #include<cstring>
5
6 #define MAXX 177
7 #define MAX (MAXX*MAXX)
8
9 int edge[MAXX],nxt[MAXX],to[MAXX],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]=true;
63                     ++cnt;
64                 }
65                 while(i=find(b))
66                 {
67                     next[b]=i;
68                     done[b]=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                             }

```

```

83     next[b]=a;
84     for(i=a; i!=b; i=next[i])
85         if(find(i))
86             {
87                 a=next[b];
88                 break;
89             }
90     while(a!=b)
91     {
92         printf("%d_",a);
93         a=next[a];
94     }
95     printf("%d\n",b);
96 }
97 return 0;
98 }

```

4.14 Hopcroft-Karp algorithm

```

1 #include<stdio>
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[MAXX],to[MAXX],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==1 || ag(cy[j])])
25             {
26                 py[j]=0;
27                 if(cy[j]==1 || ag(cy[j]))
28                     {
29                         cx[i]=j;
30                         cy[j]=i;
31                         return true;
32                     }
33             }
34     return false;
35 }
36
37 int main()
38 {
39     scanf("%d_%d_%d",&n,&p);
40     while(p--)
41     {
42         scanf("%d%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=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==1 || ag(cy[j])])
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<stdio>
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,&k);
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 } edge[100000],edge2[100000];
38
39 int head[1000],head2[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=head[u.id]; i!=-1; i=edge[i].next)
56         {
57             states v=u;
58             v.id=edge[i].to;
59             if (dist[v.id]>dist[u.id]+edge[i].cost)
60             {
61                 v.cost=dist[v.id]=dist[u.id]+edge[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[u.id]==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<n; 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",&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[MX][MX],dist[MX][MX];
12 double dp[MX],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[MX];
19 typedef std::pair<double,int> pdi;
20 std::multiset<pdi>q;
21 int pre[MX];
22
23 int main()
24 {
25     vt[0].reserve(MX);
26     vt[1].reserve(MX);
27     scanf("%d",&n);
28     memset(e,0x3f,sizeof(e));
29     while(m--)
30     {
31         scanf("%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 mempy(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
42 ans=inf;
43 for(i=1;i<=n;i++)
44     for(j=1;j<=n;j++)
45         if(e[i][j]!=inf)
46         {
47             vt[0].resize(0);
48             vt[1].resize(0);
49             static int i;
50             for(i=1;i<=n;i++)
51                 vt[0].push_back(pii(dist[::i][i],dist[j][i]));
52             std::sort(vt[0].begin(),vt[0].end());
53             for(i=0;i<vt[0].size();i++)
54             {
55                 while(!vt[1].empty() && vt[1].back().second<=vt[0][i].second)
56                     vt[1].pop_back();
57                 vt[1].push_back(vt[0][i]);
58             }
59             d=inf;
60             if(vt[1].size()==1)
61                 if(vt[1][0].first<vt[1][0].second)
62                 {
63                     ta=0;
64                     d=(vt[1][0].first<<1);
65                 }
66             else
67             {
68                 ta=e[::i][j];
69                 d=(vt[1][0].second<<1);
70             }
71             else
72                 for(i=1;i<vt[1].size();i++)
73                     if(d>e[::i][j]+vt[1][i-1].first+vt[1][i].second)
74                     {
75                         ta=(e[::i][j]+vt[1][i].second-vt[1][i-1].first)/(double)2.718;
76                         d=e[::i][j]+vt[1][i-1].first+vt[1][i].second;
77                     }
78             if(d<ans)
79             {
80                 ans=d;
81                 a=::i;
82                 b=j;
83                 dp[::i]=ta;
84                 dp[j]=e[::i][j]-ta;
85             }
86             printf("%d\n",ans);
87             for(i=1;i<=n;i++)
88                 if(i!=a && i!=b)
89                     dp[i]=le20;
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 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     return 0;
87 }
88
89 int bestmatch(void)//求最佳匹配
90 {
91     int i,j,u;
92     for (i=1; i<=n; i++)//初始化顶标
93     {
94         lx[i]=-1;
95         ly[i]=0;
96         for (j=1; j<=n; j++)
97             if (lx[i]<map[i][j])
98                 lx[i]=map[i][j];
99     }
100     memset(match,-1,sizeof(match));
101     for (u=1; u<=n; u++)
102     {
103         while (true)
104         {
105             memset(sx,0,sizeof(sx));
106             memset(sy,0,sizeof(sy));
107             if (dfs(u))
108                 break;
109             int dx=Inf;//若找不到增广路，则修改顶标--
110             for (i=1; i<=n; i++)
111             {
112                 if (sx[i])
113                     for (j=1; j<=n; j++)
114                         if (!sy[j] && dx>lx[i]+ly[j]-map[i][j])
115                             dx=lx[i]+ly[j]-map[i][j];
116             }
117             for (i=1; i<=n; i++)
118             {
119                 if (sx[i])
120                     lx[i]-=dx;
121                 if (sy[i])
122                     ly[i]+=dx;
123             }
124         }
125     }
126     int sum=0;
127     for (i=1; i<=n; i++)
128         sum+=map[match[i]][i];
129     return sum;
130 }

```

4.19 LCA - DA

```

1 int edge[MX][MX],nxt[MX][MX],to[MX][MX],cnt;
2 int pre[MX][N],dg[MX][MX];
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;i>=1+j)

```

```

32         if(i&l)
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 &a,const int &b,const int &d): a(a),b(b),id(d){}
21 };
22
23 std::list<node> to[MAXX];
24
25 int find(const int &a)
26 {
27     if(set[a]==a)
28         return a;
29     int b(set[a]);
30     set[a]=find(set[a]);
31     max[a]=std::max(max[a],max[b]);
32     min[a]=std::min(min[a],min[b]);
33     return set[a];
34 }
35
36 void tarjan(const int &now)
37 {
38     done[now]=true;
39     for(std::list<std::pair<int,int>> >::const_iterator it(q[now].begin()); it!=q[now].end(); ++it)
40         if(done[it->first])
41             if(it->second>0)
42                 to[find(it->first)].push_back(node(now,it->first,it->second));
43         else
44             to[find(it->first)].push_back(node(it->first,now,-it->second));
45     for(std::list<std::pair<int,int>> >::const_iterator it(edge[now].begin()); it!=edge[now].end(); ++it)
46         if(!done[it->first])
47         {
48             tarjan(it->first);
49             set[it->first]=now;
50             min[it->first]=it->second;
51             max[it->first]=it->second;
52         }
53     for(std::list<node> >::const_iterator it(to[now].begin()); it!=to[now].end(); ++it)
54     {
55         find(it->a);
56         find(it->b);
57         ans[0][it->id]=std::min(min[it->b],min[it->a]);
58         ans[1][it->id]=std::max(max[it->a],max[it->b]);
59     }
60 }
61
62 int main()
63 {
64     scanf("%d",&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]=1;
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 %d:\n",t);
93         for(i=0;i<=m;++i)
94             printf("%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=le10;
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))
59     {
60         for(i=1;i<=n;i++)
61             scanf("%d%d%d",&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=sqrt(1.0*(node[i].x-node[j].x)*(node[i].x-node[j].x)+
66                     (node[i].y-node[j].y)*(node[i].y-node[j].y));
67                 map[j][i].c=map[i][j].c=fabs(node[i].z-node[j].z);
68             }
69         a=0,b=mst(a);
70         while(fabs(b-a)>1e-8)
71         {
72             a=b;
73             b=mst(a);
74         }
75         printf("%.31f\n",b);
76     }
77     return 0;
78 }

```

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     }

```

```

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;>=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=cst=cf=cs=0;
71         memset(edge,0,sizeof edge);
72         for(i=1;i<=n;++i)
73         {
74             scanf("%d%d",&Pi,&Si);
75             if(S[i] && P[i])
76             {
77                 ++ans;
78                 --P[i];
79                 S[i]=0;
80             }
81             if(P[i])
82             {
83                 s[i]=1<<cf;
84                 fac[cf]=P[i];
85                 d[s[i]][i]=0;
86                 ++cf;
87             }
88         }
89         for(i=1;i<=n;++i)
90             if(S[i])
91             {
92                 s[i]=1<<(cf+cs);
93                 d[s[i]][i]=0;
94                 ++cs;
95             }
96         m=1<<(cf+cs);
97         scanf("%d",&m);
98         while(m--)
99         {
100             scanf("%d%d%d",&i,&j,&k);
101             add(i,j,k);
102             add(j,i,k);
103         }
104         for(y=1;y<=m;++y)
105         {
106             for(x=1;x<=n;++x)
107             {
108                 if(s[x] && !(s[x]&y))
109                     continue;
110                 for(i=(y-1)&y; i; i=(i-1)&y)
111                     d[y][x]=std::min(d[y][x],d[i|s[x]][x]+d[(y^i)|s[x]][x]);
112                 if(d[y][x]==inf)
113                     q.push(node(x,y,d[y][x]));
114             }
115             while(!q.empty())
116             {
117                 now=q.top();
118                 q.pop();
119                 if(now.dist!=now.get())
120                     continue;
121                 static int x,y,a,b;
122                 x=now.a;
123                 y=now.b;
124                 for(i=edge[x]; i; i=nxt[i])
125                 {
126                     a=to[i];
127                     b=y|s[a];
128                     if(d[b][a]>now.get()+wg[i])
129                     {
130                         d[b][a]=now.get()+wg[i];
131                         if(b==y)
132                             q.push(node(a,b,d[b][a]));
133                     }
134                 }
135             }
136         }
137         for(j=0;j<=n;++j)
138             dp[j]=*std::min_element(d[j]+1,d[j]+1+n);
139         cnt=cst=0;
140         for(i=1;i<=n;++i)
141             if(check(i))
142             {
143                 for(j=(i-1)&i; j; j=(j-1)&i)
144                     if(check(j) && check(i^j))
145                         dp[i]=std::min(dp[i],dp[j]+dp[i^j]);
146                 k=count(i);
147                 if(dp[i]!=inf && (k<cnt || (k==cnt && dp[i]<cst)))
148                 {
149                     cnt=k;
150                     cst=dp[i];
151                 }
152             }
153         printf("%d%d\n",ans+cnt,cst);
154     }
155     return 0;
156 }

```

4.23 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 }
76
77 // TQ's version
78 struct mcmf
79 {
80     struct Edge
81     {
82         int from,to,cap,flow,cost;
83     };
84     int n,m,s,t;
85     std::vector<Edge>edges;
86     std::vector<int>G[maxn];
87     int inq[maxn],d[maxn],p[maxn],a[maxn];
88
89     void init(int n)
90     {
91         this->n=n;
92         for(int i=0;i<=n;++i)
93             G[i].clear();
94         edges.clear();
95     }
96
97     void addedge(int from,int to,int cap,int cost)
98     {
99         Edge x={from,to,cap,0,cost};
100         edges.push_back(x);
101         Edge y={to,from,0,0,-cost};
102         edges.push_back(y);
103         m=edges.size();
104         G[from].push_back(m-2);
105         G[to].push_back(m-1);
106     }
107     int mincost(int s,int t)
108     {
109         int flow=0,cost=0;
110         while(BellmanFord(s,t,flow,cost));
111         if(flow!=(n-1)/2)return -1;
112         return cost;
113     }
114 private:
115     bool BellmanFord(int s,int t,int& flow,int& cost)
116     {
117         for(int i=0;i<=n;++i)
118             d[i]=INF;
119         memset(inq,0,sizeof(inq));
120         d[s]=0; inq[s]=1; p[s]=0; a[s]=INF;
121         std::queue<int>Q;
122         Q.push(s);
123         while(!Q.empty())
124         {
125             int u=Q.front();
126             Q.pop();
127             inq[u]=0;
128             for(int i=0;i<G[u].size();++i)
129             {
130                 Edge& e=edges[G[u][i]];
131                 if(e.cap>e.flow && d[e.to]>d[u]+e.cost)

```

```

132         {
133             d[e.to]=d[u]+e.cost;
134             p[e.to]=G[u][i];
135             a[e.to]=min(a[u],e.cap-e.flow);
136             if(!inq[e.to])
137             {
138                 Q.push(e.to);
139                 inq[e.to]=1;
140             }
141         }
142     }
143 }
144 if(d[t]==INF)
145     return false;
146 flow+=a[t];
147 cost+=d[t]*a[t];
148 int u=t;
149 while(u!=s)
150 {
151     edges[p[u]].flow+=a[t];
152     edges[p[u]^1].flow-=a[t];
153     u=edges[p[u]].from;
154 }
155 return true;
156 }
157
158 }G;

```

4.24 Second-best MST

```

1 #include<stdio>
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<=x++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<n;++i)
56         scanf("%d%d",&ed[i].a,&ed[i].b,&ed[i].c);
57     std::sort(ed,ed+m);
58     for(i=0;i<n;++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(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<=x;++i)
81             map[i][i]=0;
82         dfs(1,0);
83         for(i=0;i<n;++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
4 All-pairs vertexes' Minimum Bottleneck Path:
5 DP in the Kruscal's MST
6  $O(n^2) \cdot O(1)$ 
7
8 Minimum Diameter Spanning Tree:
9 Kariv-Hakimi Algorithm
10
11 Directed MST:
12 ChuLiu/Edmonds' Algorithm
13
14 Second-best MST:
15 get All-pairs vertexes' Minimum Bottleneck Path, then enumerate all no-tree-edges to
    replace the longest edge between two vertexes to get a worse MST
16
17 Degree-constrained MST:
18 remove the vertex from the whole graph, then add edges to increase degrees and connect
    different connected components together ( $O(m \log n + n)$  with kruscal)
19 if we can't connect all connected components together, there exists no any spanning tree
20 next step is add edges to root vertex greedily, increase degrees, and decrease our
    answer ( $O(k \cdot n)$ )
21 need all vertexes' minimum bottleneck path to root vertex
22
23 Minimum Ratio Spanning Tree:
24 Binary search
25
26 Manhattan MST:
27 combining line sweep with divide-and-conquer algorithm
28
29 Minimum Steiner Tree:
30 the MST contain all k vertexes
31 bit-mask with dijkstra  $O((l < k) \cdot (\{dijkstra\}))$ 
32 then run a bit-mask DP ( $O(n^{l < k})$ )
33
34 Count Spanning Trees:
35 TOTO
36 Matrix multiplication
37
38 k-best MST:
39 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(); // 如果目前还没尝试匹配过的对象没有被任何别的对象占据
7     else
8     {
9         for(it=edge[edge[g.front()].front()].begin();it!=edge[edge[g.front()].front()].end();++it)
10             if(*it==dfn[edge[g.front()].front()] || *it==g.front()) //如果被匹配对象更喜欢正在被匹配的人或现在准备匹配的对象
11                 break;
12             if(*it==g.front()) //如果更喜欢新的
13             {
14                 g.push_back(dfn[edge[g.front()].front()]);
15                 dfn[edge[g.front()].front()]=g.front();
16             }
17             else
18                 g.push_back(g.front()); //否则放到队尾，重新等待匹配
19         }
20         edge[g.front()].pop_front(); //每组匹配最多只考虑一次
21         g.pop_front();
22     }

```

4.27 Stoer-Wagner Algorithm

```

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

```

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, pi1;
33 int source, sink;
34 bool done[MAXN];
35
36 int aug(int now, int maxcap)
37 {
38     if(now == sink)
39     {
40         mf += maxcap;
41         cost += maxcap * pi1;
42         return maxcap;
43     }
44     done[now] = true;
45     int l = maxcap;
46     for(int i(edge[now]); i != -1; i = nxt[i])
47     {
48         if(cap[i] && !cst[i] && !done[to[i]])
49         {
50             int d(aug(to[i], std::min(l, cap[i])));
51             cap[i] -= d;
52             cap[i ^ 1] += d;
53             l -= d;
54             if(!l)
55                 return maxcap;
56         }
57     }
58     return maxcap - l;
59 }
60
61 inline bool label()
62 {
63     static int d, i, j;
64     d = inf;
65     for(i = 1; i <= n; ++i)
66     {
67         if(done[i])
68             continue;
69         for(j = 1; j <= n; ++j)
70         {
71             if(d == inf)
72                 return false;
73             if(i < j)
74                 if(d >= cst[j])
75                     d = cst[j];
76             else
77                 if(d >= -cst[j])
78                     d = -cst[j];
79             if(d < cst[j])
80                 d = cst[j];
81             if(d < -cst[j])
82                 d = -cst[j];
83             if(d < cst[j])
84                 d = cst[j];
85             if(d < -cst[j])
86                 d = -cst[j];
87             if(d < cst[j])
88                 d = cst[j];
89             if(d < -cst[j])
90                 d = -cst[j];
91             if(d < cst[j])
92                 d = cst[j];
93             if(d < -cst[j])
94                 d = -cst[j];
95             if(d < cst[j])
96                 d = cst[j];
97             if(d < -cst[j])
98                 d = -cst[j];
99             if(d < cst[j])
100                 d = cst[j];
101             if(d < -cst[j])
102                 d = -cst[j];
103             if(d < cst[j])
104                 d = cst[j];
105             if(d < -cst[j])
106                 d = -cst[j];
107             if(d < cst[j])
108                 d = cst[j];
109             if(d < -cst[j])
110                 d = -cst[j];
111             if(d < cst[j])
112                 d = cst[j];
113             if(d < -cst[j])
114                 d = -cst[j];
115             if(d < cst[j])
116                 d = cst[j];
117             if(d < -cst[j])
118                 d = -cst[j];
119             if(d < cst[j])
120                 d = cst[j];
121             if(d < -cst[j])
122                 d = -cst[j];
123             if(d < cst[j])
124                 d = cst[j];
125             if(d < -cst[j])
126                 d = -cst[j];
127             if(d < cst[j])
128                 d = cst[j];
129             if(d < -cst[j])
130                 d = -cst[j];
131             if(d < cst[j])
132                 d = cst[j];
133             if(d < -cst[j])
134                 d = -cst[j];
135             if(d < cst[j])
136                 d = cst[j];
137             if(d < -cst[j])
138                 d = -cst[j];
139             if(d < cst[j])
140                 d = cst[j];
141             if(d < -cst[j])
142                 d = -cst[j];
143             if(d < cst[j])
144                 d = cst[j];
145             if(d < -cst[j])
146                 d = -cst[j];
147             if(d < cst[j])
148                 d = cst[j];
149             if(d < -cst[j])
150                 d = -cst[j];
151             if(d < cst[j])
152                 d = cst[j];
153             if(d < -cst[j])
154                 d = -cst[j];
155             if(d < cst[j])
156                 d = cst[j];
157             if(d < -cst[j])
158                 d = -cst[j];
159             if(d < cst[j])
160                 d = cst[j];
161             if(d < -cst[j])
162                 d = -cst[j];
163             if(d < cst[j])
164                 d = cst[j];
165             if(d < -cst[j])
166                 d = -cst[j];
167             if(d < cst[j])
168                 d = cst[j];
169             if(d < -cst[j])
170                 d = -cst[j];
171             if(d < cst[j])
172                 d = cst[j];
173             if(d < -cst[j])
174                 d = -cst[j];
175             if(d < cst[j])
176                 d = cst[j];
177             if(d < -cst[j])
178                 d = -cst[j];
179             if(d < cst[j])
180                 d = cst[j];
181             if(d < -cst[j])
182                 d = -cst[j];
183             if(d < cst[j])
184                 d = cst[j];
185             if(d < -cst[j])
186                 d = -cst[j];
187             if(d < cst[j])
188                 d = cst[j];
189             if(d < -cst[j])
190                 d = -cst[j];
191             if(d < cst[j])
192                 d = cst[j];
193             if(d < -cst[j])
194                 d = -cst[j];
195             if(d < cst[j])
196                 d = cst[j];
197             if(d < -cst[j])
198                 d = -cst[j];
199             if(d < cst[j])
200                 d = cst[j];
201             if(d < -cst[j])
202                 d = -cst[j];
203             if(d < cst[j])
204                 d = cst[j];
205             if(d < -cst[j])
206                 d = -cst[j];
207             if(d < cst[j])
208                 d = cst[j];
209             if(d < -cst[j])
210                 d = -cst[j];
211             if(d < cst[j])
212                 d = cst[j];
213             if(d < -cst[j])
214                 d = -cst[j];
215             if(d < cst[j])
216                 d = cst[j];
217             if(d < -cst[j])
218                 d = -cst[j];
219             if(d < cst[j])
220                 d = cst[j];
221             if(d < -cst[j])
222                 d = -cst[j];
223             if(d < cst[j])
224                 d = cst[j];
225             if(d < -cst[j])
226                 d = -cst[j];
227             if(d < cst[j])
228                 d = cst[j];
229             if(d < -cst[j])
230                 d = -cst[j];
231             if(d < cst[j])
232                 d = cst[j];
233             if(d < -cst[j])
234                 d = -cst[j];
235             if(d < cst[j])
236                 d = cst[j];
237             if(d < -cst[j])
238                 d = -cst[j];
239             if(d < cst[j])
240                 d = cst[j];
241             if(d < -cst[j])
242                 d = -cst[j];
243             if(d < cst[j])
244                 d = cst[j];
245             if(d < -cst[j])
246                 d = -cst[j];
247             if(d < cst[j])
248                 d = cst[j];
249             if(d < -cst[j])
250                 d = -cst[j];
251             if(d < cst[j])
252                 d = cst[j];
253             if(d < -cst[j])
254                 d = -cst[j];
255             if(d < cst[j])
256                 d = cst[j];
257             if(d < -cst[j])
258                 d = -cst[j];
259             if(d < cst[j])
260                 d = cst[j];
261             if(d < -cst[j])
262                 d = -cst[j];
263             if(d < cst[j])
264                 d = cst[j];
265             if(d < -cst[j])
266                 d = -cst[j];
267             if(d < cst[j])
268                 d = cst[j];
269             if(d < -cst[j])
270                 d = -cst[j];
271             if(d < cst[j])
272                 d = cst[j];
273             if(d < -cst[j])
274                 d = -cst[j];
275             if(d < cst[j])
276                 d = cst[j];
277             if(d < -cst[j])
278                 d = -cst[j];
279             if(d < cst[j])
280                 d = cst[j];
281             if(d < -cst[j])
282                 d = -cst[j];
283             if(d < cst[j])
284                 d = cst[j];
285             if(d < -cst[j])
286                 d = -cst[j];
287             if(d < cst[j])
288                 d = cst[j];
289             if(d < -cst[j])
290                 d = -cst[j];
291             if(d < cst[j])
292                 d = cst[j];
293             if(d < -cst[j])
294                 d = -cst[j];
295             if(d < cst[j])
296                 d = cst[j];
297             if(d < -cst[j])
298                 d = -cst[j];
299             if(d < cst[j])
300                 d = cst[j];
301             if(d < -cst[j])
302                 d = -cst[j];
303             if(d < cst[j])
304                 d = cst[j];
305             if(d < -cst[j])
306                 d = -cst[j];
307             if(d < cst[j])
308                 d = cst[j];
309             if(d < -cst[j])
310                 d = -cst[j];
311             if(d < cst[j])
312                 d = cst[j];
313             if(d < -cst[j])
314                 d = -cst[j];
315             if(d < cst[j])
316                 d = cst[j];
317             if(d < -cst[j])
318                 d = -cst[j];
319             if(d < cst[j])
320                 d = cst[j];
321             if(d < -cst[j])
322                 d = -cst[j];
323             if(d < cst[j])
324                 d = cst[j];
325             if(d < -cst[j])
326                 d = -cst[j];
327             if(d < cst[j])
328                 d = cst[j];
329             if(d < -cst[j])
330                 d = -cst[j];
331             if(d < cst[j])
332                 d = cst[j];
333             if(d < -cst[j])
334                 d = -cst[j];
335             if(d < cst[j])
336                 d = cst[j];
337             if(d < -cst[j])
338                 d = -cst[j];
339             if(d < cst[j])
340                 d = cst[j];
341             if(d < -cst[j])
342                 d = -cst[j];
343             if(d < cst[j])
344                 d = cst[j];
345             if(d < -cst[j])
346                 d = -cst[j];
347             if(d < cst[j])
348                 d = cst[j];
349             if(d < -cst[j])
350                 d = -cst[j];
351             if(d < cst[j])
352                 d = cst[j];
353             if(d < -cst[j])
354                 d = -cst[j];
355             if(d < cst[j])
356                 d = cst[j];
357             if(d < -cst[j])
358                 d = -cst[j];
359             if(d < cst[j])
360                 d = cst[j];
361             if(d < -cst[j])
362                 d = -cst[j];
363             if(d < cst[j])
364                 d = cst[j];
365             if(d < -cst[j])
366                 d = -cst[j];
367             if(d < cst[j])
368                 d = cst[j];
369             if(d < -cst[j])
370                 d = -cst[j];
371             if(d < cst[j])
372                 d = cst[j];
373             if(d < -cst[j])
374                 d = -cst[j];
375             if(d < cst[j])
376                 d = cst[j];
377             if(d < -cst[j])
378                 d = -cst[j];
379             if(d < cst[j])
380                 d = cst[j];
381             if(d < -cst[j])
382                 d = -cst[j];
383             if(d < cst[j])
384                 d = cst[j];
385             if(d < -cst[j])
386                 d = -cst[j];
387             if(d < cst[j])
388                 d = cst[j];
389             if(d < -cst[j])
390                 d = -cst[j];
391             if(d < cst[j])
392                 d = cst[j];
393             if(d < -cst[j])
394                 d = -cst[j];
395             if(d < cst[j])
396                 d = cst[j];
397             if(d < -cst[j])
398                 d = -cst[j];
399             if(d < cst[j])
400                 d = cst[j];
401             if(d < -cst[j])
402                 d = -cst[j];
403             if(d < cst[j])
404                 d = cst[j];
405             if(d < -cst[j])
406                 d = -cst[j];
407             if(d < cst[j])
408                 d = cst[j];
409             if(d < -cst[j])
410                 d = -cst[j];
411             if(d < cst[j])
412                 d = cst[j];
413             if(d < -cst[j])
414                 d = -cst[j];
415             if(d < cst[j])
416                 d = cst[j];
417             if(d < -cst[j])
418                 d = -cst[j];
419             if(d < cst[j])
420                 d = cst[j];
421             if(d < -cst[j])
422                 d = -cst[j];
423             if(d < cst[j])
424                 d = cst[j];
425             if(d < -cst[j])
426                 d = -cst[j];
427             if(d < cst[j])
428                 d = cst[j];
429             if(d < -cst[j])
430                 d = -cst[j];
431             if(d < cst[j])
432                 d = cst[j];
433             if(d < -cst[j])
434                 d = -cst[j];
435             if(d < cst[j])
436                 d = cst[j];
437             if(d < -cst[j])
438                 d = -cst[j];
439             if(d < cst[j])
440                 d = cst[j];
441             if(d < -cst[j])
442                 d = -cst[j];
443             if(d < cst[j])
444                 d = cst[j];
445             if(d < -cst[j])
446                 d = -cst[j];
447             if(d < cst[j])
448                 d = cst[j];
449             if(d < -cst[j])
450                 d = -cst[j];
451             if(d < cst[j])
452                 d = cst[j];
453             if(d < -cst[j])
454                 d = -cst[j];
455             if(d < cst[j])
456                 d = cst[j];
457             if(d < -cst[j])
458                 d = -cst[j];
459             if(d < cst[j])
460                 d = cst[j];
461             if(d < -cst[j])
462                 d = -cst[j];
463             if(d < cst[j])
464                 d = cst[j];
465             if(d < -cst[j])
466                 d = -cst[j];
467             if(d < cst[j])
468                 d = cst[j];
469             if(d < -cst[j])
470                 d = -cst[j];
471             if(d < cst[j])
472                 d = cst[j];
473             if(d < -cst[j])
474                 d = -cst[j];
475             if(d < cst[j])
476                 d = cst[j];
477             if(d < -cst[j])
478                 d = -cst[j];
479             if(d < cst[j])
480                 d = cst[j];
481             if(d < -cst[j])
482                 d = -cst[j];
483             if(d < cst[j])
484                 d = cst[j];
485             if(d < -cst[j])
486                 d = -cst[j];
487             if(d < cst[j])
488                 d = cst[j];
489             if(d < -cst[j])
490                 d = -cst[j];
491             if(d < cst[j])
492                 d = cst[j];
493             if(d < -cst[j])
494                 d = -cst[j];
495             if(d < cst[j])
496                 d = cst[j];
497             if(d < -cst[j])
498                 d = -cst[j];
499             if(d < cst[j])
500                 d = cst[j];
501             if(d < -cst[j])
502                 d = -cst[j];
503             if(d < cst[j])
504                 d = cst[j];
505             if(d < -cst[j])
506                 d = -cst[j];
507             if(d < cst[j])
508                 d = cst[j];
509             if(d < -cst[j])
510                 d = -cst[j];
511             if(d < cst[j])
512                 d = cst[j];
513             if(d < -cst[j])
514                 d = -cst[j];
515             if(d < cst[j])
516                 d = cst[j];
517             if(d < -cst[j])
518                 d = -cst[j];
519             if(d < cst[j])
520                 d = cst[j];
521             if(d < -cst[j])
522                 d = -cst[j];
523             if(d < cst[j])
524                 d = cst[j];
525             if(d < -cst[j])
526                 d = -cst[j];
527             if(d < cst[j])
528                 d = cst[j];
529             if(d < -cst[j])
530                 d = -cst[j];
531             if(d < cst[j])
532                 d = cst[j];
533             if(d < -cst[j])
534                 d = -cst[j];
535             if(d < cst[j])
536                 d = cst[j];
537             if(d < -cst[j])
538                 d = -cst[j];
539             if(d < cst[j])
540                 d = cst[j];
541             if(d < -cst[j])
542                 d = -cst[j];
543             if(d < cst[j])
544                 d = cst[j];
545             if(d < -cst[j])
546                 d = -cst[j];
547             if(d < cst[j])
548                 d = cst[j];
549             if(d < -cst[j])
550                 d = -cst[j];
551             if(d < cst[j])
552                 d = cst[j];
553             if(d < -cst[j])
554                 d = -cst[j];
555             if(d < cst[j])
556                 d = cst[j];
557             if(d < -cst[j])
558                 d = -cst[j];
559             if(d < cst[j])
560                 d = cst[j];
561             if(d < -cst[j])
562                 d = -cst[j];
563             if(d < cst[j])
564                 d = cst[j];
565             if(d < -cst[j])
566                 d = -cst[j];
567             if(d < cst[j])
568                 d = cst[j];
569             if(d < -cst[j])
570                 d = -cst[j];
571             if(d < cst[j])
572                 d = cst[j];
573             if(d < -cst[j])
574                 d = -cst[j];
575             if(d < cst[j])
576                 d = cst[j];
577             if(d < -cst[j])
578                 d = -cst[j];
579             if(d < cst[j])
580                 d = cst[j];
581             if(d < -cst[j])
582                 d = -cst[j];
583             if(d < cst[j])
584                 d = cst[j];
585             if(d < -cst[j])
586                 d = -cst[j];
587             if(d < cst[j])
588                 d = cst[j];
589             if(d < -cst[j])
590                 d = -cst[j];
591             if(d < cst[j])
592                 d = cst[j];
593             if(d < -cst[j])
594                 d = -cst[j];
595             if(d < cst[j])
596                 d = cst[j];
597             if(d < -cst[j])
598                 d = -cst[j];
599             if(d < cst[j])
600                 d = cst[j];
601             if(d < -cst[j])
602                 d = -cst[j];
603             if(d < cst[j])
604                 d = cst[j];
605             if(d < -cst[j])
606                 d = -cst[j];
607             if(d < cst[j])
608                 d = cst[j];
609             if(d < -cst[j])
610                 d = -cst[j];
611             if(d < cst[j])
612                 d = cst[j];
613             if(d < -cst[j])
614                 d = -cst[j];
615             if(d < cst[j])
616                 d = cst[j];
617             if(d < -cst[j])
618                 d = -cst[j];
619             if(d < cst[j])
620                 d = cst[j];
621             if(d < -cst[j])
622                 d = -cst[j];
623             if(d < cst[j])
624                 d = cst[j];
625             if(d < -cst[j])
626                 d = -cst[j];
627             if(d < cst[j])
628                 d = cst[j];
629             if(d < -cst[j])
630                 d = -cst[j];
631             if(d < cst[j])
632                 d = cst[j];
633             if(d < -cst[j])
634                 d = -cst[j];
635             if(d < cst[j])
636                 d = cst[j];
637             if(d < -cst[j])
638                 d = -cst[j];
639             if(d < cst[j])
640                 d = cst[j];
641             if(d < -cst[j])
642                 d = -cst[j];
643             if(d < cst[j])
644                 d = cst[j];
645             if(d < -cst[j])
646                 d = -cst[j];
647             if(d < cst[j])
648                 d = cst[j];
649             if(d < -cst[j])
650                 d = -cst[j];
651             if(d < cst[j])
652                 d = cst[j];
653             if(d < -cst[j])
654                 d = -cst[j];
655             if(d < cst[j])
656                 d = cst[j];
657             if(d < -cst[j])
658                 d = -cst[j];
659             if(d < cst[j])
660                 d = cst[j];
661             if(d < -cst[j])
662                 d = -cst[j];
663             if(d < cst[j])
664                 d = cst[j];
665             if(d < -cst[j])
666                 d = -cst[j];
667             if(d < cst[j])
668                 d = cst[j];
669             if(d < -cst[j])
670                 d = -cst[j];
671             if(d < cst[j])
672                 d = cst[j];
673             if(d < -cst[j])
674                 d = -cst[j];
675             if(d < cst[j])
676                 d = cst[j];
677             if(d < -cst[j])
678                 d = -cst[j];
679             if(d < cst[j])
680                 d = cst[j];
681             if(d < -cst[j])
682                 d = -cst[j];
683             if(d < cst[j])
684                 d = cst[j];
685             if(d < -cst[j])
686                 d = -cst[j];
687             if(d < cst[j])
688                 d = cst[j];
689             if(d < -cst[j])
690                 d = -cst[j];
691             if(d < cst[j])
692                 d = cst[j];
693             if(d < -cst[j])
694                 d = -cst[j];
695             if(d < cst[j])
696                 d = cst[j];
697             if(d < -cst[j])
698                 d = -cst[j];
699             if(d < cst[j])
700                 d = cst[j];
701             if(d < -cst[j])
702                 d = -cst[j];
703             if(d < cst[j])
704                 d = cst[j];
705             if(d < -cst[j])
706                 d = -cst[j];
707             if(d < cst[j])
708                 d = cst[j];
709             if(d < -cst[j])
710                 d = -cst[j];
711             if(d < cst[j])
712                 d = cst[j];
713             if(d < -cst[j])
714                 d = -cst[j];
715             if(d < cst[j])
716                 d = cst[j];
717             if(d < -cst[j])
718                 d = -cst[j];
719             if(d < cst[j])
720                 d = cst[j];
721             if(d < -cst[j])
722                 d = -cst[j];
723             if(d < cst[j])
724                 d = cst[j];
725             if(d < -cst[j])
726                 d = -cst[j];
727             if(d < cst[j])
728                 d = cst[j];
729             if(d < -cst[j])
730                 d = -cst[j];
731             if(d < cst[j])
732                 d = cst[j];
733             if(d < -cst[j])
734                 d = -cst[j];
735             if(d < cst[j])
736                 d = cst[j];
737             if(d < -cst[j])
738                 d = -cst[j];
739             if(d < cst[j])
740                 d = cst[j];
741             if(d < -cst[j])
742                 d = -cst[j];
743             if(d < cst[j])
744                 d = cst[j];
745             if(d < -cst[j])
746                 d = -cst[j];
747             if(d < cst[j])
748                 d = cst[j];
749             if(d < -cst[j])
750                 d = -cst[j];
751             if(d < cst[j])
752                 d = cst[j];
753             if(d < -cst[j])
754                 d = -cst[j];
755             if(d < cst[j])
756                 d = cst[j];
757             if(d < -cst[j])
758                 d = -cst[j];
759             if(d < cst[j])
760                 d = cst[j];
761             if(d < -cst[j])
762                 d = -cst[j];
763             if(d < cst[j])
764                 d = cst[j];
765             if(d < -cst[j])
766                 d = -cst[j];
767             if(d < cst[j])
768                 d = cst[j];
769             if(d < -cst[j])
770                 d = -cst[j];
771             if(d < cst[j])
772                 d = cst[j];
773             if(d < -cst[j])
774                 d = -cst[j];
775             if(d < cst[j])
776                 d = cst[j];
777             if(d < -cst[j])
778                 d = -cst[j];
779             if(d < cst[j])
780                 d = cst[j];
781             if(d < -cst[j])
782                 d = -cst[j];
783             if(d < cst[j])
784                 d = cst[j];
785             if(d < -cst[j])
786                 d = -cst[j];
787             if(d < cst[j])
788                 d = cst[j];
789             if(d < -cst[j])
790                 d = -cst[j];
791             if(d < cst[j])
792                 d = cst[j];
793             if(d < -cst[j])
794                 d = -cst[j];
795             if(d < cst[j])
796                 d = cst[j];
797             if(d < -cst[j])
798                 d = -cst[j];
799             if(d < cst[j])
800                 d = cst[j];
801             if(d < -cst[j])
802                 d = -cst[j];
803             if(d < cst[j])
804                 d = cst[j];
805             if(d < -cst[j])
806                 d = -cst[j];
807             if(d < cst[j])
808                 d = cst[j];
809             if(d < -cst[j])
810                 d = -cst[j];
811             if(d < cst[j])
812                 d = cst[j];
813             if(d < -cst[j])
814                 d = -cst[j];
815             if(d < cst[j])
816                 d = cst[j];
817             if(d < -cst[j])
818                 d = -cst[j];
819             if(d < cst[j])
820                 d = cst[j];
821             if(d < -cst[j])
822                 d = -cst[j];
823             if(d < cst[j])
824                 d = cst[j];
825             if(d < -cst[j])
826                 d = -cst[j];
827             if(d < cst[j])
828                 d = cst[j];
829             if(d < -cst[j])
830                 d = -cst[j];
831             if(d < cst[j])
832                 d = cst[j];
833             if(d < -cst[j])
834                 d = -cst[j];
835             if(d < cst[j])
836                 d = cst[j];
837             if(d < -cst[j])
838                 d = -cst[j];
839             if(d < cst[j])
840                 d = cst[j];
841             if(d < -cst[j])
842                 d = -cst[j];
843             if(d < cst[j])
844                 d = cst[j];
845             if(d < -cst[j])
846                 d = -cst[j];
847             if(d < cst[j])
848                 d = cst[j];
849             if(d < -cst[j])
850                 d = -cst[j];
851             if(d < cst[j])
852                 d = cst[j];
853             if(d < -cst[j])
854                 d = -cst[j];
855             if(d < cst[j])
856                 d = cst[j];
857             if(d < -cst[j])
858                 d = -cst[j];
859             if(d < cst[j])
860                 d = cst[j];
861             if(d < -cst[j])
862                 d = -cst[j];
863             if(d < cst[j])
864                 d = cst[j];
865             if(d < -cst[j])
866                 d = -cst[j];
867             if(d < cst[j])
868                 d = cst[j];
869             if(d < -cst[j])
870                 d = -cst[j];
871             if(d < cst[j])
872                 d = cst[j];
873             if(d < -cst[j])
874                 d = -cst[j];
875             if(d < cst[j])
876                 d = cst[j];
877             if(d < -cst[j])
878                 d = -cst[j];
879             if(d < cst[j])
880                 d = cst[j];
881             if(d < -cst[j])
882                 d = -cst[j];
883             if(d < cst[j])
884                 d = cst[j];
885             if(d < -cst[j])
886                 d = -cst[j];
887             if(d < cst[j])
888                 d = cst[j];
889             if(d < -cst[j])
890                 d = -cst[j];
891             if(d < cst[j])
892                 d = cst[j];
893             if(d < -cst[j])
894                 d = -cst[j];
895             if(d < cst[j])
896                 d = cst[j];
897             if(d < -cst[j])
898                 d = -cst[j];
899             if(d < cst[j])
900                 d = cst[j];
901             if(d < -cst[j])
902                 d = -cst[j];
903             if(d < cst[j])
904                 d = cst[j];
905             if(d < -cst[j])
906                 d = -cst[j];
907             if(d < cst[j])
908                 d = cst[j];
909             if(d < -cst[j])
910                 d = -cst[j];
911             if(d < cst[j])
912                 d = cst[j];
913             if(d < -cst[j])
914                 d = -cst[j];
915             if(d < cst[j])
916                 d = cst[j];
917             if(d < -cst[j])
918                 d = -cst[j];
919             if(d < cst[j])
920                 d = cst[j];
921             if(d < -cst[j])
922                 d = -cst[j];
923             if(d < cst[j])
924                 d = cst[j];
925             if(d < -cst[j])
926                 d = -cst[j];
927             if(d < cst[j])
928                 d = cst[j];
929             if(d < -cst[j])
930                 d = -cst[j];
931             if(d < cst[j])
932                 d = cst[j];
933             if(d < -cst[j])
934                 d = -cst[j];
935             if(d < cst[j])
936                 d = cst[j];
937             if(d < -cst[j])
938                 d = -cst[j];
939             if(d < cst[j])
940                 d = cst[j];
941             if(d < -cst[j])
942                 d = -cst[j];
943             if(d < cst[j])
944                 d = cst[j];
945             if(d < -cst[j])
946                 d = -cst[j];
947             if(d < cst[j])
948                 d = cst[j];
949             if(d < -cst[j])
950                 d = -cst[j];
951             if(d < cst[j])
952                 d = cst[j];
953             if(d < -cst[j])
954                 d = -cst[j];
955             if(d < cst[j])
956                 d = cst[j];
957             if(d < -cst[j])
958                 d = -cst[j];
959             if(d < cst[j])
9
```

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  $O(\sqrt{\text{p}})$ 
3 //where p is n's largest prime factor.
4 #include<cstdio>
5 #include<cmath>
6 #include<cstring>
7 struct Hash // std::map is bad. clear()时会付出巨大的代价
8 {
9     static const int mod=100003; // 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]; //  $a^k \pmod{c}$ 
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!=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,mam,k;
59     static int i,cnt;
60     d=g=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*d%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%g%c;
97     }
98     return -1ll;
99 }
100
101 long long k,p,n;
102
103 int main()
104 {
105     while(scanf("%lld,%lld,%lld",&d,&p,&n)!=EOF)
106     {
107         if(n>p || (k=bsgs(k,n,p))!=-1ll)
108             puts("Orz,I can't find D!");
109         else
110             printf("%lld\n",k);
111     }
112     return 0;
113 }
```

5.3 Divisor function

```

1 sum of positive divisors function
2 (n)=(pow(p[0],a[0]+1)-1)/(p[0]-1)* (pow(p[1],a[1]+1)-1)/(p[1]-1)* ... (pow(p[n-1],a[n-1]+1)-1)/(p[n-1]-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<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+=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;j<=1); // size must be such many
72         x.resize(0);
73         for(i=0;i<=n;++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         for(i=0;i<x.size();++i)
90             cnt[i]=ceil(x[i].real()); // maybe we need (x[i].real()+0.5f) or nearbyint(x[i].real())
91         x.resize(2*a[n-1]); // result here
92     }
93     return 0;
94 }
```

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=0;j<n;j++) //第j行,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     if(v==n)
31     {
32         static int x[MAXX],ta[MAXX][MAXX];
33         static int tmp;
34         memcpy(x,ans,sizeof(x));
35         memcpy(ta,a,sizeof(ta));
36         for(i=-1;i>=0;-i)
37         {
38             for(j=i+1;j<n;j++)
39                 ta[i][n]=(x[j]&&ta[i][j]); //迭代消元求解
40             x[i]=ta[i][n];
41         }
42         for(tmp=i=0;i<n;i++)
43             if(x[i])
44                 ++tmp;
45         cnt=std::min(cnt,tmp);
46         return;
47     }
48     ans[v]=0;
49     dfs(v+1);
50     ans[v]=1;
51     dfs(v+1);
52 }
53
54 inline int ge(int a[N][N],int n)
55 {
56     static int i,j,k,l;
57     for(i=j=0;j<n;j++)
58     {
59         for(k=i;k<n;k++)
60             if(a[k][i])
61                 break;
62         if(k<n)
63         {
64             for(l=0;l<n;l++)
65                 std::swap(a[i][l],a[k][l]);
66             for(k=0;k<n;k++)
67                 if(k!=i && a[k][i])
68                     for(l=0;l<n;l++)
69                         a[k][l]^=a[i][l];
70             ++i;
71         }
72         else //将不定元交换到后面去
73         {
74             l=n-1-++i;
75             for(k=0;k<n;k++)
76                 std::swap(a[k][l],a[k][i]);
77         }
78     }
79     if(i==n)
80     {
81         for(i=cnt=0;i<n;i++)
82             if(a[i][n])
83                 ++cnt;
84         printf("%d\n",cnt);
85         continue;
86     }
87     for(j=i;j<n;j++)
88         if(a[j][n])
89             break;
90     if(j<n)
91         puts("impossible");
92     else
93     {
94         memset(ans,0,sizeof(ans));
95         cnt=111;
96         dfs(l=i);
97         printf("%d\n",cnt);
98     }
99 }
100
101 /*
102 */
103
104 inline void ge(int a[N][N],int m,int n) // m*n
105 {
106     static int i,j,k,l,b,c;
107     for(i=j=0;i<n && j<n;j++)
108     {
109         for(k=i;k<n;k++)
110             if(a[k][j])
111                 break;
112         if(k==n)
113             continue;
114         for(l=0;l<n;l++)
115             std::swap(a[i][l],a[k][l]);
116         for(k=0;k<n;k++)
117             if(k!=i && a[k][j])
118             {
119                 b=a[k][j];
120                 c=a[i][j];
121                 for(l=0;l<n;l++)
122                     a[k][l]=(a[k][l]*c-a[i][l]*b)%7+7%7;
123             }
124         ++i;
125     }
126     for(j=i;j<n;j++)
127         if(a[j][n])
128             break;
129

```

```

130     if(j<n)
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*d%mod)
12         if (y&1)
13             ret=ret*a%mod;
14     return ret;
15 }
16
17 inline int getInv(int x,int mod)//为素数mod
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[MAXN];
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-1;
22     for(i=0;i<n;i++)
23         ix[i]=i;
24     memset(d,0,sizeof d);
25     for(i=0;i<n;i++)
26     {
27         for(j=0;j<n;j++)
28             d[i][j]=-a[i][j];
29         d[i][m]=1;
30         d[i][m]=b[i];
31         if(d[r][m]>d[i][m])
32             r=i;
33     }
34     for(j=0;j<n;j++)
35         d[n][j]=c[j];
36     d[n+1][m]=-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<n;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<n;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<n;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+1;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+1;i)
71         if(ix[i]+1<n)
72             x[ix[i]]=d[i-m][m]; // answer

```

```

73     ans=d[n][n]; // maximum value
74     return true;
75 }
76
77 int main()
78 {
79     while(scanf("%d %d",&n,&m)!=EOF)
80     {
81         for(i=0;i<n+1;i)
82             scanf("%lf",&c[i]); // max{ sum{c[i]*x[i]} }
83         for(i=0;i<n+1;i)
84         {
85             for(j=0;j<n+1;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_can_spend_%.01f_taka.\n",ceil(ans));
92     }
93     return 0;
94 }

```

5.9 Lucas' theorem(2)

```

1  #include<stdio>
2  #include<string>
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%p; y>=1;a=a*a%p)
35         if (y&1)
36             ret=ret*a%p;
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)%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("%d\n",calc(n,m,p));
68     }
69     return 0;
70 }

```

5.10 Lucas' theorem

```

1  #include <stdio>
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/=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("%d%d%d",&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: %d\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+1;i)
10             for (j=0;j<N+1+j)
11             {
12                 res.a[i][j]=0;
13                 for (k=0;k<N+1+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+1+i)
23             {
24                 for (j=0;j<N+1+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,unsigned long long b,
2      const unsigned long long &n)
3  {
4      unsigned long long exp(a),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 inline unsigned long long exp_mod(unsigned long long a,unsigned long long b,const
20 unsigned long long &c)
21 {
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)=(1+4+4+2)!/1!/4!/4!/2!
4 |
5 | Combination:
6 | MultiSet S={aa1,aa2,...,aaak}
7 | C(S,r)=(r+k-1)!/r!/(k-1)!=C(r,r+k-1)
8 |
9 | if(r>min{count(element[i])})
10 |     you have to resolve this problem with inclusion-exclusion principle.
11 |
12 | MS T={3 a,4 b,5 c}
13 | MS T* = {aa,ab,ac}
14 | A1 = {(T*)|count(a) > 3}/(8)
15 | A2 = {(T*)|count(b) > 4}/(5)
16 | A3 = {(T*)|count(c) > 5}/(6)
17 |
18 | C(T,10)=C(T*,10) - (|A1|+|A2|+|A3|) + (|A1 A2|+|A1 A3|+|A2 A3|) - |A1 A2 A3|
19 | C(10,12) C(1,3) C(0,2) 0 0
20 | ans=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 slow in some situation )
7 | or we can enumerate the continued fraction of  $\sqrt{n}$ , as  $\frac{x}{y}$ , it will be much more faster
8 |
9 | other solution pairs' matrix:
10 | x1  n x y1
11 | y1  x1
12 | k-th solution is {matrix}k
13 | */
14 | import java.util.*;
15 | import java.math.*;
16 |
17 | public class Main
18 | {
19 |     static BigInteger p,q,p1,p2,p3,q1,q2,q3,a1,a2,a0,h1,h2,g1,g2,n0;
20 |     static int n,t;
21 |     static void solve()
22 |     {
23 |         p2=BigInteger.ONE;
24 |         p1=BigInteger.ZERO;
25 |         q2=BigInteger.ZERO;
26 |         q1=BigInteger.ONE;
27 |         a0=a1=BigInteger.valueOf((long)Math.sqrt(n));
28 |         g1=BigInteger.ZERO;
29 |         h1=BigInteger.ONE;
30 |         n0=BigInteger.valueOf(n);
31 |         while(true)
32 |         {
33 |             g2=a1.multiply(h1).subtract(g1);
34 |             h2=(n0.subtract(g2.multiply(g2))).divide(h1);
35 |             a2=(g2.add(a0)).divide(h2);
36 |             p=p2.multiply(a1).add(p1);
37 |             q=q2.multiply(a1).add(q1);
38 |             if(p.multiply(p).subtract(n0.multiply(q.multiply(q))).equals(BigInteger.ONE)
39 |                 )
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<stdio>
2 | #include<stdlib>
3 | #include<list>
4 |
5 | short T;
6 | unsigned long long a;
7 | std::list<unsigned long long>fac;
8 |
9 |
10 | inline unsigned long long multi_mod(const unsigned long long &a,unsigned long long b,
11 | const unsigned long long &n)
12 | {
13 |     unsigned long long exp(a/n),tmp(0);
14 |     while(b)
15 |     {
16 |         if(b&1)
17 |         {
18 |             tmp+=exp;
19 |             if(tmp>n)
20 |                 tmp=n;
21 |         }
22 |         exp<<=1;
23 |         if(exp>n)
24 |             exp=n;
25 |         b>>=1;
26 |     }
27 |     return tmp;
28 | }
29 |
30 | inline unsigned long long exp_mod(unsigned long long a,unsigned long long b,const
31 | unsigned long long &c)
32 | {
33 |     unsigned long long tmp(1);
34 |     while(b)
35 |     {
36 |         if(b&1)
37 |             tmp=multi_mod(tmp,a,c);
38 |         a=multi_mod(a,a,c);
39 |         b>>=1;
40 |     }
41 |     return tmp;
42 | }
43 |
44 | inline bool miller_rabbin(const unsigned long long &n,short T)
45 | {
46 |     if(n==2)
47 |         return true;
48 |     if(n<2 || !(n&1))
49 |         return false;
50 |     unsigned long long a,u(n-1),x,y;
51 |     short t(0),i;
52 |     while(!(u&1))
53 |     {
54 |         ++t;
55 |         u>>=1;
56 |     }
57 |     while(T--)
58 |     {
59 |         a=rand()%(n-1)+1;
60 |         x=exp_mod(a,u,n);
61 |         for(i=0;i<t;++i)
62 |         {
63 |             y=multi_mod(x,x,n);
64 |             if(y==1 && x!=1 && x!=n-1)
65 |                 return false;
66 |             x=y;
67 |         }
68 |         if(y!=1)
69 |             return false;
70 |     }
71 |     return true;
72 | }
73 |
74 | unsigned long long gcd(const unsigned long long &a,const unsigned long long &b)
75 | {
76 |     return b?gcd(b,a%b):a;
77 | }
78 |
79 | inline unsigned long long pollar_rho(const unsigned long long n,const unsigned long long
80 | &c)
81 | {
82 |     unsigned long long x(rand()%(n-1)+1),y,d,i(1),k(2);
83 |     y=x;
84 |     while(true)
85 |     {
86 |         ++i;
87 |         x=(multi_mod(x,x,n)+c)%n;
88 |         d=gcd((x-y+n)%n,n);
89 |         if(d>1 && d<n)
90 |             return d;
91 |         if(x==y)
92 |             return n;
93 |         if(i==k)
94 |         {
95 |             k<<=1;
96 |             y=x;
97 |         }
98 |     }
99 | }
100 |
101 | void find(const unsigned long long &n,short c)
102 | {
103 |     if(n==1)
104 |         return;
105 |     if(miller_rabbin(n,6))
106 |     {
107 |         fac.push_back(n);
108 |         return;
109 |     }
110 |     unsigned long long p(n);
111 |     short k(c);
112 |     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("%d",&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[MX];
5
6  int main()
7  {
8      prm.reserve(MX); // pi(x)~x/ln(x);
9      for(i=2;i<=MX;i++)
10     {
11         if(!flag[i])
12             prm.push_back(i);
13         for(j=0;j<prm.size() && i*prm[j]<MX;i++)
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
7  对于 n 来说，所有这样的数的和为  $\frac{n \times \varphi(n)}{2}$ 。
8
9  inline long long phi(int n)
10 {
11     static int i;
12     static int re;
13     re=1;
14     for(i=0;prm[i]*prm[i]<=n;i++)
15     {
16         if(n%prm[i]==0)
17         {
18             re=re/prm[i];
19             do
20                 n/=prm[i];
21             while(n%prm[i]==0);
22         }
23     }
24     if(n!=1)
25         re=re/n;
26     return re;
27 }
28
29 inline void Euler()
30 {
31     static int i,j;
32     phi[1]=1;
33     for(i=2;i<=MX;i++)
34     {
35         if(!phi[i])
36         {
37             for(j=i;j<=MX;j+=i)
38             {
39                 phi[j]=j;
40                 phi[j]=phi[j]/i*(i-1);
41             }
42         }
43     }
44
45     Multiplicative order:
46
47     the multiplicative order of a modulo n is the smallest positive integer k with
48      $a^k \equiv 1 \pmod n$ 
49
50     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)
51
52     求:
53     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，找到最小的一个 d，满足  $x^d \equiv 1 \pmod m$ ；
54     method 2
55     inline long long ord(long long x,long long m)
56     {
57         static long long ans;
58         static int i,j;
59         ans=phi(m);
60         for(i=0;i<fac.size();i++)
61             for(j=0;j<fac[i].second && pow(x,ans/fac[i].first,m)!=1;i++)
62                 ans/=fac[i].first;
63     }
64     return ans;
65 }
66
67 Primitive root:
68
69 若  $\text{ord}(x) = \varphi(m)$ ，则 x 为 m 的一个原根
70
71 因此只需检查所有  $x^d \equiv 1 \pmod m$  的约数 d，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
72
73 当且仅当  $m = 1, 2, 4, p^k, 2 \times p^k$  {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……？

```

```

69 当 m 存在原根时，原根数目为  $\varphi(\varphi(m))$ 
70
71 求:
72
73 枚举每一个简化剩余系中的数 i，若对于 i 的每一个质因子 p[j],  $i^{\frac{\varphi(m)}{p[j]}} \not\equiv 1 \pmod m$ ，那么 i 为 m 的一个原根。也就是说， $\text{ord}(i) = \varphi(m)$ 。
74
75 最小原根通常极小。
76
77 Carmichael function:
78
79  $\lambda(n)$  is defined as the smallest positive integer m such that
80  $a^m \equiv 1 \pmod n$  { forall a!=1 && gcd(a,n)=1 }
81 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  $\text{lcm}\{\text{ord}(x)\}$ 
82
83 if  $n = p[0]^a[0] \times p[1]^a[1] \times \dots \times p[m-1]^a[m-1]$ 
84 then  $\lambda(n) = \text{lcm}(\lambda(p[0]^a[0]), \lambda(p[1]^a[1]), \dots, \lambda(p[m-1]^a[m-1]))$ ;
85
86 if  $n = 2^c \times p[0]^a[0] \times p[1]^a[1] \times \dots \times p[m-1]^a[m-1]$ 
87 then  $\lambda(n) = \text{lcm}(2^{c-1}, \varphi(p[0]^a[0]), \varphi(p[1]^a[1]), \dots, \varphi(p[m-1]^a[m-1]))$ ;
88 {  $c=0$  if  $a<2$ ;  $c=1$  if  $a=2$ ;  $c=a-2$  if  $a>3$ ; }
89
90 Carmichael's theorem:
91 if  $\text{gcd}(a,n)=1$ 
92 then  $\lambda(n) \equiv 1 \pmod n$ 

```

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(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 MX 11
5
6  int T,t;
7  int m[MX],a[MX];
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=m[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]; // 使用本身的存串的时候注意已被重载
50     }
51 }
52
53 // normal version
54
55 #define N 128
56
57 char buf[MAX];
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->nxt[i];
101                 while (q)
102                 {
103                     if (q->nxt[i])
104                     {
105                         p->nxt[i]->fal=q->nxt[i];
106                         break;
107                     }
108                     q=q->nxt[i];
109                 }
110                 if (!q)
111                     p->nxt[i]->fal=rt;
112                 Q.push(p->nxt[i]);
113             }
114     }
115 }
116
117 inline void match(const char *s)
118 {
119     static node *p, *q;
120     for (p=rt; *s; ++s)
121     {
122         while (p!=rt && !p->nxt[*s])
123             p=p->fal;
124         p=p->nxt[*s];
125         if (!p)
126             p=rt;
127         for (q=p; q!=rt && q->idx; q=q->fal) // why q->idx ? looks like not necessary at all, I delete it in an other solution
128             ++cnt[q->idx];
129     }

```

```

130 }
131
132 //可以考虑一下，拉直指针来跳过无效的匹配dfs fal
133 //在线调整关键字存在性的时候，可以考虑欧拉序压扁之后使用或者线段树进行区间修改BIT
134 //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，然后沿着指针构成的往上
    传递覆盖次数fal DAG

```

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可能最小循环环节长度=

```

6.3 Manacher's Algorithm

```

1 #include<cstdio>
2 #include<vector>
3
4 #define MAXX 1111
5
6 std::vector<char> str;
7 char buf[MAXX];
8 int z[MAXX<<1];
9 int i, j, l, r;
10 int ii, n, c;
11
12 inline int match(const int &a, const int &b)
13 {
14     int i(0);
15     while (a-i>=0 && b+i<str.size() && str[a-i]==str[b+i]) //注意是不是，打错过很多次了i1
16         ++i;
17     return i;
18 }
19
20 int main()
21 {
22     gets(buf);
23     str.reserve(MAXX<<1);
24     for (i=0; buf[i]; ++i)
25     {
26         str.push_back('S');
27         str.push_back(buf[i]);
28     }
29     str.push_back('S');
30
31     z[0]=1;
32     c=l=r=0;
33     for (i=1; i<str.size(); ++i)
34     {
35         ii=(l<<1)-i;
36         r=i+l-i;
37
38         if (i>r)
39         {
40             z[i]=match(i, i);
41             l=i;
42             r=i+z[i]-1;
43         }
44         else
45             if (z[i]==n)
46             {
47                 z[i]=n+match(i-n, i+n);
48                 l=i;
49                 r=i+z[i]-1;
50             }
51             else
52                 z[i]=std::min(z[ii], n);
53         if (z[i]>z[c])
54             c=i;
55     }
56
57     for (i=c-z[c]+2, r=c+z[c]; i<n; i+=2)
58         putchar(str[i]);
59     puts("");
60     return 0;
61 }
62
63 //package:
64
65 inline int match(const int a, const int b, const std::vector<int> &str)
66 {
67     static int i;
68     i=0;
69     while (a-i>=0 && b+i<str.size() && str[a-i]==str[b+i])
70         ++i;
71     return i;
72 }
73
74 inline void go(int *z, const std::vector<int> &str)
75 {
76     static int c, l, r, i, ii, n;
77     z[0]=1;
78     c=l=r=0;
79     for (i=1; i<str.size(); ++i)
80     {
81         ii=(l<<1)-i;
82         r=i+l-i;
83
84         if (i>r)
85         {
86             z[i]=match(i, i, str);
87             l=i;

```

```

88         r=i+z[i]-1;
89     }
90     else
91     {
92         if(z[i]==n)
93         {
94             z[i]=n+match(i-n,i+n,str);
95             l=i;
96             r=i+z[i]-1;
97         }
98         else
99         {
100             z[i]=std::min(z[i],n);
101             if(z[i]>z[c])
102                 c=i;
103         }
104     }
105 }
106 inline bool check(int *z,int a,int b) //检查子串[a,b是否回文]
107 {
108     a=a*2-1;
109     b=b*2-1;
110     int m=(a+b)/2;
111     return z[m]>=b-m+1;
112 }

```

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)
8              k++;
9          else
10             {
11                 if (cmp>0)
12                     j+=k+1;
13                 else
14                     i+=k+1;
15                 if (i==j) j++;
16                 k=0;
17             }
18     }
19     return std::min(i,j);
20 }

```

6.6 Suffix Array - DC3 Algorithm

```

1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4
5  #define MAXX 1111
6  #define F(x) ((x)/3+((x)%3==1?0:tb))
7  #define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
8
9  int wa[MAXX],wb[MAXX],wv[MAXX],ws[MAXX];
10
11 inline bool c0(const int *str,const int &a,const int &b)
12 {
13     return str[a]==str[b] && str[a+1]==str[b+1] && str[a+2]==str[b+2];
14 }
15
16 inline bool c12(const int *str,const int &k,const int &a,const int &b)
17 {
18     if(k==2)
19         return str[a]<str[b] || str[a]==str[b] && c12(str,1,a+1,b+1);
20     else
21         return str[a]<str[b] || str[a]==str[b] && wv[a+1]<wv[b+1];
22 }
23
24 inline void sort(int *str,int *a,int *b,const int &n,const int &m)
25 {
26     memset(ws,0,sizeof(ws));
27     int i;
28     for(i=0;i<n;++i)
29         ++ws[wv[i]=str[a[i]]];
30     for(i=1;i<n;++i)
31         ws[i]+=ws[i-1];
32     for(i=m-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]<tbc)
56             wb[ta++]=san[i]*3;
57     if(tb%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]);
62     for(i=j=k=0;i<ta&&j<tbc;)
63         sa[k++]=c12(str,wb[j],wb[i])?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: 中取值范围, 桶排序str
85
86     for(i=1;i<=n;++i) // 数组rank
87         rk[sa[i]]=i;
88     for(i=k=0;i<n;++i) // 数组lcp
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]=1;
103     for(i=1;i<=n;++i) //sparse table RMQ
104     {
105         k=n+1-(1<<i);
106         for(j=1;j<=n-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=g[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=k[l];
127     r=k[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<n;i++)
13         wss[i]=0;
14     for(i=0;i<n;i++)
15         wss[x[i]=s[i]]++;
16     for(i=1;i<n;i++)
17         wss[i]+=wss[i-1];
18     for(i=m-1;i>=0;i--)
19         sa[-wss[x[i]]]=i;
20     for(j=l,p=1;p<n&&j<n;j*=2,m=p)
21     {
22         for(i=m-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<n; i++)
30         wss[i] = 0;
31     for(i=0; i<n; i++)
32         wss[wv[i]]++;
33     for(i=1; i<n; 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 && str[i+k]==str[j+k]; ++k);
45 }

```

6.8 Suffix Automaton

```

1  /*
2  length(s) ∈ [ min(s), max(s) ] = [ max[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-->0)
69     {
70         now=the[i];
71         // topsort already
72     }
73 }
74 /*
75 size of 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         }
45         n=0;
46         p=-1;
47         for(i=0;i<the[1].size();++i)
48             if(dp[i]>n)
49                 n=dp[i];
50         printf("%d\n",n);
51         for(i=n-1;i>=0;--i)
52         {
53             ans[i]=the[1][path[i]];
54             path[i]=p;
55         }
56         for(i=0;i<n;++i)
57             printf("%d_",ans[i]);
58         puts("");
59     }
60     return 0;
61 }

```

8 search

8.1 dlx

```

1  精确覆盖：给定一个矩阵，现在要选择一些行，使得每一列有且仅有一个。
2  011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。重复覆盖：
   给定一个矩阵，现在要选择一些行，使得每一列至少有一个。
3
4
5  011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[MAXN],d[MAXN],l[MAXN],r[MAXN],ch[MAXN],rh[MAXN];
15 int sz[MAXN];
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=r[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=l[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==hd)
91     {
92         ans.resize(k);
93         return true;
94     }
95     int s=inf,c;
96     int i,j;
97     for(i=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], cntcol[N];
126 int dcnt = 0;
127 //初始化一个节点
128 inline void addnode(int &x)
129 {
130     ++x;
131     r[x] = l[x] = u[x] = d[x] = x;
132 }
133 //将加入到后xrowx
134 inline void insert_row(int rowx, int x)
135 {
136     r[l[rowx]] = x;
137     l[x] = l[rowx];
138     r[x] = rowx;
139     l[rowx] = x;
140 }
141 //将加入到后xcolx
142 inline void insert_col(int colx, int x)
143 {
144     d[u[colx]] = x;
145     u[x] = u[colx];
146     d[x] = colx;
147     u[colx] = x;
148 }
149 //全局初始化
150 inline void dlx_init(int cols)
151 {
152     memset(h, -1, sizeof(h));
153     memset(cntcol, 0, sizeof(cntcol));
154     dcnt = -1;
155     addnode(dcnt);
156     for (int i = 1; i <= cols; ++i)
157     {
158         addnode(dcnt);
159         insert_row(0, dcnt);
160     }
161 }
162 //删除一列以及相关的所有行
163 inline void remove(int c)
164 {
165     l[r[c]] = l[c];
166     r[l[c]] = r[c];
167     for (int i = d[c]; i != c; i = d[i])
168         for (int j = r[i]; j != i; j = r[j])
169         {
170             u[d[j]] = u[j];

```

```

171             d[u[j]] = d[j];
172             cntcol[col[j]]--;
173         }
174     }
175 //恢复一列以及相关的所有行
176 inline void resume(int c)
177 {
178     for (int i = u[c]; i != c; i = u[i])
179         for (int j = l[i]; j != i; j = l[j])
180         {
181             u[d[j]] = j;
182             d[u[j]] = j;
183             cntcol[col[j]]++;
184         }
185     l[r[c]] = c;
186     r[l[c]] = c;
187 }
188 //搜索部分
189 bool DLX(int deep)
190 {
191     if (r[0] == 0)
192     {
193         //Do anything you want to do here
194         printf("%d", deep);
195         for (int i = 0; i < deep; ++i) printf("%d", res[i]);
196         puts("");
197         return true;
198     }
199     int min = INT_MAX, tempc;
200     for (int i = r[0]; i != 0; i = r[i])
201         if (cntcol[i] < min)
202         {
203             min = cntcol[i];
204             tempc = i;
205         }
206     remove(tempc);
207     for (int i = d[tempc]; i != tempc; i = d[i])
208     {
209         res[deep] = row[i];
210         for (int j = r[i]; j != i; j = r[j]) remove(col[j]);
211         if (DLX(deep + 1)) return true;
212         for (int j = l[i]; j != i; j = l[j]) resume(col[j]);
213     }
214     resume(tempc);
215     return false;
216 }
217 //插入矩阵中的节点"1"
218 inline void insert_node(int x, int y)
219 {
220     cntcol[y]++;
221     addnode(dcnt);
222     row[dcnt] = x;
223     col[dcnt] = y;
224     insert_col(y, dcnt);
225     if (h[x] == -1) h[x] = dcnt;
226     else insert_row(h[x], dcnt);
227 }
228 int main()
229 {
230     int n, m;
231     while (~scanf("%d%d", &n, &m))
232     {
233         dlx_init(m);
234         for (int i = 1; i <= n; ++i)
235         {
236             int k, x;
237             scanf("%d", &k);
238             while (k--)
239             {
240                 scanf("%d", &x);
241                 insert_node(i, x);
242             }
243         }
244         if (DLX(0))
245             puts("NO");
246     }
247     return 0;
248 }

```

8.3 dlx - repeat cover

```

1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4
5 #define MAXN 110
6 #define MAXM 1000000
7 #define INF 0x7FFFFFFF
8
9 using namespace std;
10
11 int G[MAXN][MAXN];
12 int L[MAXN], R[MAXN], U[MAXN], D[MAXN];
13 int size, ans, S[MAXN], H[MAXN], C[MAXN];
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 = 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 = 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>goods[i-1].cost))
40         dfs(i+1,cost_n+goods[i].cost,carry_n-goods[i].weig,1);
41     dfs(i+1,cost_n,carry_n,0);
42 }
43
44 int main()
45 {
46     // freopen("asdf","r",stdin);
47     scanf("%d",&T);
48     for(t=1;t<=T;++t)
49     {
50         scanf("%d%d",&n,&carry);
51         sumw=0;
52         sumc=0;
53         ans=0;
54         for(i=0;i<n;++i)
55         {
56             scanf("%d%d",&goods[i].weig,&goods[i].cost);
57             sumw+=goods[i].weig;
58             sumc+=goods[i].cost;
59         }
60         if(sumw<=carry)
61         {
62             printf("Case %d: %d\n",t,sumc);
63             continue;

```

```

64     }
65     // qsort(goods,n,sizeof(struct mono),com);
66     std::sort(goods,goods+n,comp);
67     for(i=0;i<n;++i)
68     {
69         // printf("%d %d\n",goods[i].weig,goods[i].cost);
70         las[i]=sumc;
71         sumc+=goods[i].cost;
72     }
73     dfs(0,0,carry,1);
74     printf("Case %d: %d\n",t,ans);
75 }
76 return 0;
77 }

```

9 others

9.1 vimrc

```

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

```

9.2 bigint

```

1  // header files
2  #include <cstdio>
3  #include <string>
4  #include <algorithm>
5  #include <iostream>
6
7  struct Bigint
8  {
9      // representations and structures
10     std::string a; // to store the digits
11     int sign; // sign = -1 for negative numbers, sign = 1 otherwise
12     // constructors
13     Bigint() {} // default constructor
14     Bigint( std::string b ) { (*this) = b; } // constructor for std::string
15     // some helpful methods
16     int size() // returns number of digits
17     {
18         return a.size();
19     }
20     Bigint inverseSign() // changes the sign
21     {
22         sign *= -1;
23         return (*this);
24     }
25     Bigint normalize( int newSign ) // removes leading 0, fixes sign
26     {
27         for( int i = a.size() - 1; i > 0 && a[i] == '0'; i-- )
28             a.erase(a.begin() + i);
29         sign = ( a.size() == 1 && a[0] == '0' ) ? 1 : newSign;
30         return (*this);
31     }
32     // assignment operator
33     void operator = ( std::string b ) // assigns a std::string to Bigint
34     {
35         a = b[0] == '-' ? b.substr(1) : b;
36         reverse( a.begin(), a.end() );
37         this->normalize( b[0] == '-' ? -1 : 1 );
38     }
39     // conditional operators
40     bool operator < ( const Bigint &b ) const // less than operator
41     {
42         if( sign != b.sign )
43             return sign < b.sign;
44         if( a.size() != b.a.size() )
45             return sign == 1 ? a.size() < b.a.size() : a.size() > b.a.size();
46         for( int i = a.size() - 1; i >= 0; i-- )
47             if( a[i] != b.a[i] )
48                 return sign == 1 ? a[i] < b.a[i] : a[i] > b.a[i];
49         return false;
50     }
51     bool operator == ( const Bigint &b ) const // operator for equality
52     {
53         return a == b.a && sign == b.sign;
54     }
55
56     // mathematical operators
57     Bigint operator + ( Bigint b ) // addition operator overloading
58     {
59         if( sign != b.sign )
60             return (*this) - b.inverseSign();
61         Bigint c;
62         for(int i = 0, carry = 0; i<a.size() || i<b.size() || carry; i++)
63         {
64             carry+=(i<a.size() ? a[i]-48 : 0)+(i<b.a.size() ? b.a[i]-48 : 0);
65             c.a += (carry % 10 + 48);
66             carry /= 10;
67         }
68         return c.normalize(sign);
69     }
70
71     Bigint operator - ( Bigint b ) // subtraction operator overloading
72     {
73         if( sign != b.sign )
74             return (*this) + b.inverseSign();
75         int s = sign; sign = b.sign = 1;
76         if( (*this) < b )
77             return ((b - (*this)).inverseSign()).normalize(-s);
78         Bigint c;
79         for( int i = 0, borrow = 0; i < a.size(); i++ )
80         {
81             borrow = a[i] - borrow - (i < b.size() ? b.a[i] : 48);
82             c.a += borrow >= 0 ? borrow + 48 : borrow + 58;
83             borrow = borrow >= 0 ? 0 : 1;
84         }

```

```

85     return c.normalize(s);
86 }
87 Bigint operator * ( Bigint b ) // multiplication operator overloading
88 {
89     Bigint c("0");
90     for( int i = 0, k = a[i] - 48; i < a.size(); i++, k = a[i] - 48 )
91     {
92         while(k-->0)
93             c = c + b; // ith digit is k, so, we add k times
94         b.a.insert(b.a.begin(), '0'); // multiplied by 10
95     }
96     return c.normalize(sign * b.sign);
97 }
98 Bigint operator / ( Bigint b ) // division operator overloading
99 {
100     if( b.size() == 1 && b.a[0] == '0' )
101         b.a[0] /= ( b.a[0] - 48 );
102     Bigint c("0"), d;
103     for( int j = 0; j < a.size(); j++ )
104         d.a += "0";
105     int dSign = sign * b.sign;
106     b.sign = 1;
107     for( int i = a.size() - 1; i >= 0; i-- )
108     {
109         c.a.insert( c.a.begin(), '0' );
110         c = c + a.substr( i, 1 );
111         while( !( c < b ) )
112         {
113             c = c - b;
114             d.a[i]++;
115         }
116     }
117     return d.normalize(dSign);
118 }
119 Bigint operator % ( Bigint b ) // modulo operator overloading
120 {
121     if( b.size() == 1 && b.a[0] == '0' )
122         b.a[0] /= ( b.a[0] - 48 );
123     Bigint c("0");
124     b.sign = 1;
125     for( int i = a.size() - 1; i >= 0; i-- )
126     {
127         c.a.insert( c.a.begin(), '0' );
128         c = c + a.substr( i, 1 );
129         while( !( c < b ) )
130             c = c - b;
131     }
132     return c.normalize(sign);
133 }
134 // output method
135 void print()
136 {
137     if( sign == -1 )
138         putchar('-');
139     for( int i = a.size() - 1; i >= 0; i-- )
140         putchar(a[i]);
141 }
142 };
143
144
145
146 int main()
147 {
148     Bigint a, b, c; // declared some Bigint variables
149     ///////////////////////////////////
150     // taking Bigint input //
151     ///////////////////////////////////
152
153     std::string input; // std::string to take input
154     std::cin >> input; // take the Big integer as std::string
155     a = input; // assign the std::string to Bigint a
156
157     std::cin >> input; // take the Big integer as std::string
158     b = input; // assign the std::string to Bigint b
159
160     ///////////////////////////////////
161     // Using mathematical operators //
162     ///////////////////////////////////
163
164     c = a + b; // adding a and b
165     c.print(); // printing the Bigint
166     puts(""); // newline
167
168     c = a - b; // subtracting b from a
169     c.print(); // printing the Bigint
170     puts(""); // newline
171
172     c = a * b; // multiplying a and b
173     c.print(); // printing the Bigint
174     puts(""); // newline
175
176     c = a / b; // dividing a by b
177     c.print(); // printing the Bigint
178     puts(""); // newline
179
180     c = a % b; // a modulo b
181     c.print(); // printing the Bigint
182     puts(""); // newline
183
184     ///////////////////////////////////
185     // Using conditional operators //
186     ///////////////////////////////////
187
188     if( a == b )
189         puts("equal"); // checking equality
190     else
191         puts("not_equal");
192
193     if( a < b )
194         puts("a_is_smaller_than_b"); // checking less than operator
195
196     return 0;
197 }
198

```

9.3 Binary Search

```

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

```

```

10         mid=l+r>>1;
11         if(A[mid]<x)
12             l=mid+1;
13         else
14         {
15             r=mid-1;
16             if(A[mid]==x)
17                 re=mid;
18         }
19     }
20     return re;
21 }
22
23 inline int go(int A[],int n,int x) // return the largest i that make A[i]==x;
24 {
25     static int l,r,mid,re;
26     l=0;
27     r=n-1;
28     re=-1;
29     while(l<=r)
30     {
31         mid=l+r>>1;
32         if(A[mid]<=x)
33         {
34             l=mid+1;
35             if(A[mid]==x)
36                 re=mid;
37         }
38         else
39             r=mid-1;
40     }
41     return re;
42 }
43
44 inline int go(int A[],int n,int x) // retrun the largest i that make A[i]<x;
45 {
46     static int l,r,mid,re;
47     l=0;
48     r=n-1;
49     re=-1;
50     while(l<=r)
51     {
52         mid=l+r>>1;
53         if(A[mid]<x)
54         {
55             l=mid+1;
56             re=mid;
57         }
58         else
59             r=mid-1;
60     }
61     return re;
62 }
63
64 inline int go(int A[],int n,int x)// return the largest i that make A[i]<=x;
65 {
66     static int l,r,mid,re;
67     l=0;
68     r=n-1;
69     re=-1;
70     while(l<=r)
71     {
72         mid=l+r>>1;
73         if(A[mid]<=x)
74         {
75             l=mid+1;
76             re=mid;
77         }
78         else
79             r=mid-1;
80     }
81     return re;
82 }
83
84 inline int go(int A[],int n,int x)// return the least i that make A[i]>x;
85 {
86     static int l,r,mid,re;
87     l=0;
88     r=n-1;
89     re=-1;
90     while(l<=r)
91     {
92         mid=l+r>>1;
93         if(A[mid]<=x)
94             l=mid+1;
95         else
96         {
97             r=mid-1;
98             re=mid;
99         }
100     }
101     return re;
102 }
103
104 inline int go(int A[],int n,int x)// upper_bound();
105 {
106     static int l,r,mid;
107     l=0;
108     r=n-1;
109     while(l<r)
110     {
111         mid=l+r>>1;
112         if(A[mid]<=x)
113             l=mid+1;
114         else
115             r=mid;
116     }
117     return r;
118 }
119
120 inline int go(int A[],int n,int x)// lower_bound();
121 {
122     static int l,r,mid;
123     l=0;
124     r=n-1;
125     while(l<r)
126     {
127         mid=l+r>>1;
128         if(A[mid]<x)
129             l=mid+1;
130         else
131             r=mid;
132     }
133     return r;
134 }

```

```

1 //Scanner
2
3 Scanner in=new Scanner(new FileReader("asdf"));
4 PrintWriter pw=new PrintWriter(new FileWriter("out"));
5 boolean    in.hasNext();
6 String     in.next();
7 BigDecimal in.nextBigDecimal();
8 BigInteger in.nextBigInteger();
9 BigInteger in.nextBigInteger(int radix);
10 double     in.nextDouble();
11 int        in.nextInt();
12 int        in.nextInt(int radix);
13 String     in.nextLine();
14 long       in.nextLong();
15 long       in.nextLong(int radix);
16 short      in.nextShort();
17 short      in.nextShort(int radix);
18 int        in.radix(); //Returns this scanner's default radix.
19 Scanner    in.useRadix(int radix); // Sets this scanner's default radix to the
    specified radix.
20 void        in.close(); //Closes this scanner.
21
22 //String
23
24 char        str.charAt(int index);
25 int         str.compareTo(String anotherString); // <0 if less. ==0 if equal. >0 if
    greater.
26 int         str.compareToIgnoreCase(String str);
27 String      str.concat(String str);
28 boolean     str.contains(CharSequence s);
29 boolean     str.endsWith(String suffix);
30 boolean     str.startsWith(String prefix);
31 boolean     str.startsWith(String prefix,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(); setBit(
    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, 状态状态状态状态状态状态状态状态状态状态状态
16
17
18 1、
19 2.calm_down();calm_down();calm_down();、读完题目读完题目读完题目
20 3、不盲目刷题
21 4、考虑换题换想法
22 5/、对数高线
23 6//hash观察问题本身点 区间互转//、对数调整精度
24 6.1 or 将乘法转换成加法、点化区间、区间化点
25 6.2、数组大小……

```