

Code Library



Himemiyu Nanao @ Perfect Freeze

August 18, 2013

Contents

1	data structure	1	4.16 k Shortest Augmenting Path Algorithm	48	
1.1	atlantis	1	4.17 Kariv-Hakimi Algorithm	50	
1.2	Binary Indexed tree	1	4.18 Kuhn-Munkres algorithm	51	
1.3	COT	2	4.19 LCA - DA	52	
1.4	hose	3	4.20 LCA - tarjan - minmax	53	
1.5	Leftist tree	3	4.21 Minimum Ratio Spanning Tree	53	
1.6	Network	4	4.22 Minimum Steiner Tree	54	
1.7	OTOCI	6	4.23 Minimum-cost flow problem	55	
1.8	picture	8	4.24 Second-best MST	56	
1.9	Size Blanced Tree	9	4.25 Spanning tree	56	
1.10	Sparse Table - rectangle	10	4.26 Stable Marriage	57	
1.11	Sparse Table - square	11	4.27 Stoer-Wagner Algorithm	57	
1.12	Sparse Table	11	4.28 Strongly Connected Component	58	
1.13	Treap	11	4.29 ZKW's Minimum-cost flow	58	
2	geometry	12	5	math	59
2.1	3D	12	5.1	cantor	59
2.2	3DCH	14	5.2	Discrete logarithms - BSGS	59
2.3	circle's area	16	5.3	Divisor function	60
2.4	circle	17	5.4	Extended Euclidean Algorithm	60
2.5	closest point pair	19	5.5	Fast Fourier Transform	60
2.6	ellipse	20	5.6	Gaussian elimination	61
2.7	Graham's scan	21	5.7	inverse element	62
2.8	half-plane intersection	21	5.8	Linear programming	62
2.9	intersection of circle and poly	22	5.9	Lucas' theorem(2)	63
2.10	k-d tree	23	5.10	Lucas' theorem	64
2.11	Manhattan MST	24	5.11	Matrix	64
2.12	others	25	5.12	Miller-Rabin Algorithm	65
2.13	Pick's theorem	26	5.13	Multiset	65
2.14	PointInPoly	26	5.14	Pell's equation	65
2.15	rotating caliper	26	5.15	Pollard's rho algorithm	66
2.16	shit	28	5.16	Prime	67
2.17	sort - polar angle	29	5.17	Reduced Residue System	67
2.18	triangle	29	5.18	Simpson's rule	68
			5.19	System of linear congruences	68
3	geometry/tmp	29	6	string	68
3.1	circle	29	6.1	Aho-Corasick Algorithm	68
3.2	circles	31	6.2	Gusfield's Z Algorithm	69
3.3	halfplane	32	6.3	Manacher's Algorithm	70
3.4	line	33	6.4	Morris-Pratt Algorithm	70
3.5	line3d	34	6.5	smallest representation	70
3.6	plane	34	6.6	Suffix Array - DC3 Algorithm	70
3.7	point	35	6.7	Suffix Array - Prefix-doubling Algorithm	71
3.8	point3d	36	6.8	Suffix Automaton	72
3.9	polygon	37			
3.10	polygons	39	7	search	73
4	graph	40	7.1	dlx	73
4.1	2SAT	40	7.2	dlx - exact cover	73
4.2	Articulation	41	7.3	dlx - repeat cover	74
4.3	Augmenting Path Algorithm for Maximum Cardinality Bipartite Matching	41	7.4	fibonacci knapsack	75
4.4	Biconnected Component - Edge	41	8	dynamic programming	76
4.5	Biconnected Component	42	8.1	knapsack problem	76
4.6	Blossom algorithm	43	8.2	LCIS	76
4.7	Bridge	44	9	others	76
4.8	Chu-Liu:Edmonds' Algorithm	44	9.1	.vimrc	76
4.9	Covering problems	45	9.2	bigint	77
4.10	Difference constraints	45	9.3	Binary Search	78
4.11	Dinitz's algorithm	45	9.4		79
4.12	Flow network	46	9.5	others	80
4.13	Hamiltonian circuit	47			
4.14	Hopcroft-Karp algorithm	48			

1 data structure

1.1 atlantis

```
1 #include<stdio>
2 #include<algorithm>
3 #include<map>
4
5 #define MAXX 111
6 #define inf 333
7 #define MAX inf*5
8
9 int mid[MAX],cnt[MAX];
10 double len[MAX];
11
12 int n,i,cas;
13 double x1,x2,y1,y2;
14 double ans;
15 std::map<double,int>map;
16 std::map<double,int>::iterator it;
17 double rmap[inf];
18
19 void make(int id,int l,int r)
20 {
21     mid[id]=(l+r)>>1;
22     if(l!=r)
23     {
24         make(id<<1,l,mid[id]);
25         make(id<<1|1,mid[id]+1,r);
26     }
27 }
28
29 void update(int id,int ll,int rr,int l,int r,int
    val)
30 {
31     if(ll==l && rr==r)
32     {
33         cnt[id]+=val;
34         if(cnt[id])
35             len[id]=rmap[r]-rmap[l-1];
36     }
37     else
38     {
39         if(l!=r)
40             len[id]=len[id<<1]+len[id<<1|1];
41         else
42             len[id]=0;
43     }
44     return;
45 }
46 if(mid[id]>=r)
47     update(id<<1,ll,mid[id],l,r,val);
48 else
49     if(mid[id]<l)
50         update(id<<1|1,mid[id]+1,rr,l,r,val);
51     else
52     {
53         update(id<<1,ll,mid[id],l,mid[id],val);
54         update(id<<1|1,mid[id]+1,rr,mid[id]
55             +1,r,val);
56     }
57 if(!cnt[id])
58     len[id]=len[id<<1]+len[id<<1|1];
59 }
60
61 struct node
62 {
63     double l,r,h;
64     char f;
65     inline bool operator<(const node &a)const
66     {
67         return h<a.h;
```

```
64     }
65     inline void print()
66     {
67         printf("%lf_%lf_%lf_%d\n",l,r,h,f);
68     }
69 }ln[inf];
70
71 int main()
72 {
73     make(1,1,inf);
74     while(scanf("%d",&n),n)
75     {
76         n<=&1;
77         map.clear();
78         for(i=0;i<n;++i)
79         {
80             scanf("%lf%lf%lf%lf",&x1,&y1,&x2,&y2);
81             ;
82             if(x1>x2)
83                 std::swap(x1,x2);
84             if(y1>y2)
85                 std::swap(y1,y2);
86             ln[i].l=x1;
87             ln[i].r=x2;
88             ln[i].h=y1;
89             ln[i].f=1;
90             ln[++i].l=x1;
91             ln[i].r=x2;
92             ln[i].h=y2;
93             ln[i].f=-1;
94             map[x1]=1;
95             map[x2]=1;
96         }
97         i=1;
98         for(it=map.begin();it!=map.end();++it,++i)
99         {
100             it->second=i;
101             rmap[i]=it->first;
102         }
103         std::sort(ln,ln+n);
104         ans=0;
105         update(1,1,inf,map[ln[0].l]+1,map[ln[0].r]
106             ,ln[0].f);
107         for(i=1;i<n;++i)
108         {
109             ans+=len[i]*(ln[i].h-ln[i-1].h);
110             update(1,1,inf,map[ln[i].l]+1,map[ln[i]
111                 .r],ln[i].f);
112         }
113         printf("Test case %d\nTotal explored area: %.2lf\n",++cas,ans);
114     }
115     return 0;
```

1.2 Binary Indexed tree

```
1 int tree[MAXX];
2
3 inline int lowbit(const int &a)
4 {
5     return a&-a;
6 }
7
8 inline void update(int pos,const int &val)
9 {
10     while(pos<MAXX)
11     {
12         tree[pos]+=val;
```

```

13     pos+=lowbit(pos);
14 }
15 }
16
17 inline int read(int pos)
18 {
19     int re(0);
20     while(pos>0)
21     {
22         re+=tree[pos];
23         pos-=lowbit(pos);
24     }
25     return re;
26 }
27
28 int find_Kth(int k)
29 {
30     int now=0;
31     for (char i=20;i>=0;--i)
32     {
33         now|=(1<<i);
34         if (now>MAXX || tree[now]>=k)
35             now^=(1<<i);
36         else k-=tree[now];
37     }
38     return now+1;
39 }

```

1.3 COT

```

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

```

```

41             nid=lson[nid];
42             id=lson[id];
43             r=mid;
44         }
45     else
46     {
47         lson[nid]=lson[id];
48         rson[nid]=++cnt;
49         nid=rson[nid];
50         id=rson[id];
51         l=mid+1;
52     }
53     sz[nid]=sz[id]+1;
54 }
55 return re;
56 }
57
58 void rr(int now, int fa)
59 {
60     dg[now]=dg[fa]+1;
61     head[now]=update(head[fa], num[now]);
62     for(int i=edge[now]; i; i=nxt[i])
63         if(to[i]!=fa)
64         {
65             j=1;
66             for(pre[to[i]][0]=now; j<N; ++j)
67                 pre[to[i]][j]=pre[pre[to[i]][j]-1][j-1];
68             rr(to[i], now);
69         }
70 }
71
72 inline int query(int a, int b, int n, int k)
73 {
74     static int tmp, t;
75     l=1;
76     r=m;
77     a=head[a];
78     b=head[b];
79     t=num[n];
80     n=head[n];
81     while(l<r)
82     {
83         mid=(l+r)>>1;
84         tmp=sz[lson[a]]+sz[lson[b]]-2*sz[lson[n]]+(l<=t && t<=mid);
85         if(tmp>=k)
86         {
87             a=lson[a];
88             b=lson[b];
89             n=lson[n];
90             r=mid;
91         }
92     else
93     {
94         k-=tmp;
95         a=rson[a];
96         b=rson[b];
97         n=rson[n];
98         l=mid+1;
99     }
100 }
101 return l;
102 }
103
104 inline int lca(int a, int b)
105 {
106     static int i, j;
107     j=0;
108     if(dg[a]<dg[b])

```

```

109     std::swap(a,b);
110     for (i=dg[a]-dg[b]; i;i>=1,++j)
111         if (i&1)
112             a=pre[a][j];
113     if (a==b)
114         return a;
115     for (i=N-1; i>=0; --i)
116         if (pre[a][i]!=pre[b][i])
117         {
118             a=pre[a][i];
119             b=pre[b][i];
120         }
121     return pre[a][0];
122 }
123
124 int main()
125 {
126     scanf("%d%d",&n,&q);
127     for (i=1; i<=n; ++i)
128     {
129         scanf("%d",&num[i]);
130         map[i]=num[i];
131     }
132     std::sort(map+1,map+n+1);
133     m=std::unique(map+1,map+n+1)-map-1;
134     for (i=1; i<=n; ++i)
135         num[i]=std::lower_bound(map+1,map+m+1,num
136             [i])-map;
137     for (i=1; i<=n; ++i)
138     {
139         scanf("%d%d",&j,&k);
140         nxt[++cnt]=edge[j];
141         edge[j]=cnt;
142         to[cnt]=k;
143
144         nxt[++cnt]=edge[k];
145         edge[k]=cnt;
146         to[cnt]=j;
147     }
148     cnt=0;
149     head[0]=make(1,m);
150     rr(1,0);
151     while(q--)
152     {
153         scanf("%d%d%d",&i,&j,&k);
154         printf("%d\n",map[query(i,j,lca(i,j),k)]);
155     }
156     return 0;
157 }

```

1.4 hose

```

1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4 #include<cmath>
5
6 #define MAXX 50111
7
8 struct Q
9 {
10     int l,r,s,w;
11     bool operator<(const Q &i) const
12     {
13         return w==i.w?r<i.r:w<i.w;
14     }
15 } a[MAXX];
16
17 int c[MAXX];

```

```

18 long long col[MAXX],sz[MAXX],ans[MAXX];
19 int n,m,cnt,len;
20
21 long long gcd(long long a,long long b)
22 {
23     return a?gcd(b%a,a):b;
24 }
25
26 int i,j,k,now;
27 long long all,num;
28
29 int main()
30 {
31     scanf("%d%d",&n,&m);
32     for (i=1; i<=n; ++i)
33         scanf("%d",&c[i]);
34     len=sqrt(m);
35     for (i=1; i<=m; ++i)
36     {
37         scanf("%d%d",&a[i].l,&a[i].r);
38         if (a[i].l>a[i].r)
39             std::swap(a[i].l,a[i].r);
40         sz[i]=a[i].r-a[i].l+1;
41         a[i].w=a[i].l/len+1;
42         a[i].s=i;
43     }
44     std::sort(a+1,a+m+1);
45     i=1;
46     while(i<=m)
47     {
48         now=a[i].w;
49         memset(col,0,sizeof col);
50         for (j=a[i].l; j<=a[i].r; ++j)
51             ans[a[i].s]+=2*(col[c[j]]++);
52         for(++i; a[i].w==now; ++i)
53         {
54             ans[a[i].s]=ans[a[i-1].s];
55             for (j=a[i-1].r+1; j<=a[i].r; ++j)
56                 ans[a[i].s]+=2*(col[c[j]]++);
57             if (a[i-1].l<a[i].l)
58                 for (j=a[i-1].l; j<a[i].l; ++j)
59                     ans[a[i].s]-=2*(-col[c[j]]);
60             else
61                 for (j=a[i].l; j<a[i-1].l; ++j)
62                     ans[a[i].s]+=2*(col[c[j]]++);
63         }
64     }
65     for (i=1; i<=m; ++i)
66     {
67         if (sz[i]==1)
68             all=1ll;
69         else
70             all=sz[i]*(sz[i]-1);
71         num=gcd(ans[i],all);
72         printf("%lld/%lld\n",ans[i]/num,all/num);
73     }
74     return 0;
75 }

```

1.5 Leftist tree

```

1 #include<cstdio>
2 #include<algorithm>
3
4 #define MAXX 100111
5
6 int val[MAXX],l[MAXX],r[MAXX],d[MAXX];
7
8 int set[MAXX];
9

```

```

10 int merge(int a,int b)
11 {
12     if(!a)
13         return b;
14     if(!b)
15         return a;
16     if(val[a]<val[b]) // max-heap
17         std::swap(a,b);
18     r[a]=merge(r[a],b);
19     if(d[l[a]]<d[r[a]])
20         std::swap(l[a],r[a]);
21     d[a]=d[r[a]]+1;
22     set[l[a]]=set[r[a]]=a; // set a as father of
        its sons
23     return a;
24 }
25
26 inline int find(int &a)
27 {
28     while(set[a]) //brute-force to get the index
        of root
29         a=set[a];
30     return a;
31 }
32
33 inline void reset(int i)
34 {
35     l[i]=r[i]=d[i]=set[i]=0;
36 }
37
38 int n,i,j,k;
39
40 int main()
41 {
42     while(scanf("%d",&n)!=EOF)
43     {
44         for(i=1;i<=n;++i)
45         {
46             scanf("%d",val+i);
47             reset(i);
48         }
49         scanf("%d",&n);
50         while(n-->0)
51         {
52             scanf("%d%d",&i,&j);
53             if(find(i)==find(j))
54                 puts("-1");
55             else
56             {
57                 k=merge(l[i],r[i]);
58                 val[i]>=1;
59                 reset(i);
60                 set[i=merge(i,k)]=0;
61
62                 k=merge(l[j],r[j]);
63                 val[j]>=1;
64                 reset(j);
65                 set[j=merge(j,k)]=0;
66
67                 set[k=merge(i,j)]=0;
68                 printf("%d\n",val[k]);
69             }
70         }
71     }
72     return 0;
73 }

```

1.6 Network

1 | //HLD.....备忘....._(3JZ)_

```

2 #include<cstdio>
3 #include<algorithm>
4 #include<cstdlib>
5
6 #define MAXX 80111
7 #define MAXE (MAXX<<1)
8 #define N 18
9
10 int edge[MAXX],nxt[MAXE],to[MAXE],cnt;
11 int fa[MAXX][N],dg[MAXX];
12
13 inline int lca(int a,int b)
14 {
15     static int i,j;
16     j=0;
17     if(dg[a]<dg[b])
18         std::swap(a,b);
19     for(i=dg[a]-dg[b];i>=1;++j)
20         if(i&1)
21             a=fa[a][j];
22     if(a==b)
23         return a;
24     for(i=N-1;i>=0;--i)
25         if(fa[a][i]!=fa[b][i])
26         {
27             a=fa[a][i];
28             b=fa[b][i];
29         }
30     return fa[a][0];
31 }
32
33 inline void add(int a,int b)
34 {
35     nxt[++cnt]=edge[a];
36     edge[a]=cnt;
37     to[cnt]=b;
38 }
39
40 int sz[MAXX],pre[MAXX],next[MAXX];
41
42 void rr(int now)
43 {
44     sz[now]=1;
45     int max,id;
46     max=0;
47     for(int i=edge[now];i;i=nxt[i])
48         if(to[i]!=fa[now][0])
49         {
50             fa[to[i]][0]=now;
51             dg[to[i]]=dg[now]+1;
52             rr(to[i]);
53             sz[now]+=sz[to[i]];
54             if(sz[to[i]]>max)
55             {
56                 max=sz[to[i]];
57                 id=to[i];
58             }
59         }
60     if(max)
61     {
62         next[now]=id;
63         pre[id]=now;
64     }
65 }
66
67 #define MAXT (MAXX*N*5)
68
69 namespace Treap
70 {
71     int cnt;

```

```

72  int son[MAXT][2], key[MAXT], val[MAXT], sz[MAXT];
73
74  inline void init()
75  {
76      key[0]=RAND_MAX;
77      val[0]=0xc0c0c0c0;
78      cnt=0;
79  }
80
81  inline void up(int id)
82  {
83      sz[id]=sz[son[id][0]]+sz[son[id][1]]+1;
84  }
85  inline void rot(int &id,int tp)
86  {
87      static int k;
88      k=son[id][tp];
89      son[id][tp]=son[k][tp^1];
90      son[k][tp^1]=id;
91      up(id);
92      up(k);
93      id=k;
94  }
95  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
141 void print(int id)
142 {
143     if(!id)
144         return;
145     print(son[id][0]);
146     printf("%d ",val[id]);
147     print(son[id][1]);
148 }
149
150 int head[MAXX], root[MAXX], len[MAXX], pos[MAXX];
151
152 #define MAX (MAXX*6)
153 #define mid (l+r>>1)
154 #define lc lson[id],l,mid
155 #define rc rson[id],mid+1,r
156
157 int lson[MAX], rson[MAX];
158 int treap[MAX];
159
160 void make(int &id,int l,int r,int *the)
161 {
162     id==++cnt;
163     static int k;
164     for(k=l;k<=r;++k)
165         Treap::insert(treap[id],the[k]);
166     if(l!=r)
167     {
168         make(lc,the);
169         make(rc,the);
170     }
171 }
172
173 int query(int id,int l,int r,int a,int b,int q)
174 {
175     if(a<=l && r<=b)
176         return Treap::rank(treap[id],q);
177     int re(0);
178     if(a<=mid)
179         re=query(lc,a,b,q);
180     if(b>mid)
181         re+=query(rc,a,b,q);
182     return re;
183 }
184
185 inline int query(int a,int b,int v)
186 {
187     static int re;
188     for(re=0;root[a]!=root[b];a=fa[root[a]][0])
189         re+=query(head[root[a]],1,len[root[a]],1,
190             pos[a],v);
191     re+=query(head[root[a]],1,len[root[a]],pos[b],
192         pos[a],v);
193     return re;
194 }
195
196 inline void update(int id,int l,int r,int pos,int
197     val,int n)
198 {
199     while(l<=r)
200     {
201         Treap::del(treap[id],val);
202         Treap::insert(treap[id],n);
203         if(l==r)
204             return;
205         if(pos<=mid)
206         {
207             id=lson[id];
208             r=mid;
209         }
210     }
}

```

```

206     }
207     else
208     {
209         id=rson[id];
210         l=mid+1;
211     }
212 }
213 }
214
215 int n,q,i,j,k;
216 int val[MAXX];
217
218 int main()
219 {
220     srand(1e9+7);
221     scanf("%d%d",&n,&q);
222     for(i=1;i<=n;++i)
223         scanf("%d",&val[i]);
224     for(k=1;k<=n;++k)
225     {
226         scanf("%d%d",&i,&j);
227         add(i,j);
228         add(j,i);
229     }
230     rr(rand()%n+1);
231     for(j=1;j<=n;++j)
232         for(i=1;i<=n;++i)
233             fa[i][j]=fa[fa[i][j-1]][j-1];
234
235     Treap::init();
236     cnt=0;
237     for(i=1;i<=n;++i)
238         if(!pre[i])
239         {
240             static int tmp[MAXX];
241             for(k=1,j=i;j;j=next[j],++k)
242             {
243                 pos[j]=k;
244                 root[j]=i;
245                 tmp[k]=val[j];
246             }
247             —k;
248             len[i]=k;
249             make(head[i],1,k,tmp);
250         }
251     while(q--)
252     {
253         scanf("%d",&k);
254         if(k)
255         {
256             static int a,b,c,d,l,r,ans,m;
257             scanf("%d%d",&a,&b);
258             c=lca(a,b);
259             if(dg[a]+dg[b]-2*dg[c]+1<k)
260             {
261                 puts("invalid request!");
262                 continue;
263             }
264             k=dg[a]+dg[b]-2*dg[c]+1-k+1;
265             if(dg[a]<dg[b])
266                 std::swap(a,b);
267             l=-1e9;
268             r=1e9;
269             if(b!=c)
270             {
271                 d=a;
272                 for(i=0,j=dg[a]-dg[c]-1;j;j>>=1,++i)
273                     if(j&1)
274                         d=fa[d][i];

```

```

275         while(l<=r)
276         {
277             m=l+r>>1;
278             if(query(a,d,m)+query(b,c,m)
279                 >=k)
280             {
281                 ans=m;
282                 r=m-1;
283             }
284             else
285                 l=m+1;
286         }
287     }
288     else
289     {
290         while(l<=r)
291         {
292             m=l+r>>1;
293             if(query(a,c,m)>=k)
294             {
295                 ans=m;
296                 r=m-1;
297             }
298             else
299                 l=m+1;
300         }
301         printf("%d\n",ans);
302     }
303     else
304     {
305         scanf("%d%d",&i,&j);
306         update(head[root[i]],1,len[root[i]],
307             pos[i],val[i],j);
308         val[i]=j;
309     }
310     return 0;
311 }

```

1.7 OTOCI

```

1 //记得随手 down 啊……亲……
2 //debug 时记得优先检查 up/down/select
3 #include<cstdio>
4 #include<algorithm>
5
6 #define MAXX 30111
7
8 int nxt[MAXX][2], fa[MAXX], pre[MAXX], val[MAXX], sum
9     [MAXX];
10 bool rev[MAXX];
11
12 inline void up(int id)
13 {
14     static int i;
15     sum[id]=val[id];
16     for(i=0;i<2;++i)
17         if(nxt[id][i])
18             sum[id]+=sum[nxt[id][i]];
19 }
20
21 inline void rot(int id,int tp)
22 {
23     static int k;
24     k=pre[id];
25     nxt[k][tp^1]=nxt[id][tp];
26     if(nxt[id][tp])
27         pre[nxt[id][tp]]=k;
28     if(pre[k])

```



```

28     nxt[pre[k]][k==nxt[pre[k]][1]]=id;    97
29     pre[id]=pre[k];                        98
30     nxt[id][tp]=k;                         99
31     pre[k]=id;                             100
32     up(k);                                 101
33     up(id);                                102
34 }                                           103
35                                           104
36 inline void down(int id) //记得随手 down 啊……亲…105
37 {                                           106
38     static int i;                          107
39     if(rev[id])                             108
40     {                                       109
41         rev[id]=false;                     110
42         std::swap(nxt[id][0],nxt[id][1]);  111
43         for(i=0;i<2;++i)                   112
44             if(nxt[id][i])                 113
45                 rev[nxt[id][i]]^=true;    114
46     }                                       115
47 }                                           116
48                                           117
49 int freshen(int id)                        118
50 {                                           119
51     int re(id);                            120
52     if(pre[id])                             121
53         re=freshen(pre[id]);               122
54     down(id);                               123
55     return re;                             124
56 }                                           125
57                                           126
58 inline void splay(int id) //记得随手 down 啊……亲…127
59 {                                           128
60     static int rt;                          129
61     if(id!=(rt=freshen(id)))                130
62         for(std::swap(fa[id],fa[rt]);pre[id];rot(131
63             id,id==nxt[pre[id]][0]));        132
64     /* another faster method:              133
65     if(id!=rt)                              134
66     {                                       135
67         std::swap(fa[id],fa[rt]);          136
68         do                                 137
69         {                                   138
69             rt=pre[id];                    139
70             if(pre[rt])                    140
71             {                               141
72                 k=(nxt[pre[rt]][0]==rt);    142
73                 if(nxt[rt][k]==id)           143
74                     rot(id,k^1);            144
75                 else                         145
76                     rot(rt,k);              146
77                 rot(id,k);                  147
78             }                               148
79             else                             149
80                 rot(id,id==nxt[rt][0]);     150
81         }                                   151
82         while(pre[id]);                     152
83     }                                       153
84     */                                     154
85 }                                           155
86                                           156
87 inline void access(int id)                 157
88 {                                           158
89     static int to;                          159
90     for(to=0;id;id=fa[id])                 160
91     {                                       161
92         splay(id);                         162
93         if(nxt[id][1])                     163
94         {                                   164
95             pre[nxt[id][1]]=0;              165
96             fa[nxt[id][1]]=id;              166
97         }
98         nxt[id][1]=to;
99         if(to)
100         {
101             pre[to]=id;
102             fa[to]=0;
103         }
104         up(to=id);
105     }
106 }
107
108 inline int getrt(int id)
109 {
110     access(id);
111     splay(id);
112     while(nxt[id][0])
113     {
114         id=nxt[id][0];
115         down(id);
116     }
117     return id;
118 }
119
120 inline void makert(int id)
121 {
122     access(id);
123     splay(id);
124     if(nxt[id][0])
125         rev[id]^=true;
126 }
127
128 int n,i,j,k,q;
129 char buf[11];
130
131 int main()
132 {
133     scanf("%d",&n);
134     for(i=1;i<=n;++i)
135         scanf("%d",val+i);
136     scanf("%d",&q);
137     while(q--)
138     {
139         scanf("%s%d%d",buf,&i,&j);
140         switch(buf[0])
141         {
142             case 'b':
143                 if(getrt(i)==getrt(j))
144                     puts("no");
145             else
146             {
147                 puts("yes");
148                 makert(i);
149                 fa[i]=j;
150             }
151             break;
152             case 'p':
153                 access(i);
154                 splay(i);
155                 val[i]=j;
156                 up(i);
157                 break;
158             case 'e':
159                 if(getrt(i)!=getrt(j))
160                     puts("impossible");
161                 else
162                 {
163                     makert(i);
164                     access(j);
165                     splay(j);
166                     printf("%d\n",sum[j]);

```

```

167     }
168     break;
169 }
170 }
171 return 0;
172 }

1.8 picture

1 #include<cstdio>
2 #include<algorithm>
3 #include<map>
4
5 #define MAXX 5555
6 #define MAX MAXX<<3
7 #define inf 10011
8
9 int n,i;
10 int mid[MAX],cnt[MAX],len[MAX],seg[MAX];
11 bool rt[MAX],lf[MAX];
12
13 std::map<int,int>map;
14 std::map<int,int>::iterator it;
15 int rmap[inf];
16 long long sum;
17 int x1,x2,y1,y2,last;
18
19 void make(int id,int l,int r)
20 {
21     mid[id]=(l+r)>>1;
22     if(l!=r)
23     {
24         make(id<<1,l,mid[id]);
25         make(id<<1|1,mid[id]+1,r);
26     }
27 }
28
29 void update(int id,int ll,int rr,int l,int r,int
    val)
30 {
31     if(l==ll && rr==r)
32     {
33         cnt[id]+=val;
34         if(cnt[id])
35         {
36             rt[id]=lf[id]=true;
37             len[id]=rmap[r]-rmap[l-1];
38             seg[id]=1;
39         }
40     }
41     else
42     {
43         if(l!=r)
44         {
45             len[id]=len[id<<1]+len[id<<1|1];
46             seg[id]=seg[id<<1]+seg[id<<1|1];
47             if(rt[id<<1] && lf[id<<1|1])
48                 —seg[id];
49             rt[id]=rt[id<<1|1];
50             lf[id]=lf[id<<1];
51         }
52     }
53     else
54     {
55         len[id]=0;
56         rt[id]=lf[id]=false;
57         seg[id]=0;
58     }
59     return;
60 }
61 if(mid[id]<l)
62     update(id<<1|1,mid[id]+1,rr,l,r,val);
63 else
64 {
65     update(id<<1,ll,mid[id],l,mid[id],val);
66     update(id<<1|1,mid[id]+1,rr,mid[id]
        +1,r,val);
67 }
68 if(!cnt[id])
69 {
70     len[id]=len[id<<1]+len[id<<1|1];
71     seg[id]=seg[id<<1]+seg[id<<1|1];
72     if(rt[id<<1] && lf[id<<1|1])
73         —seg[id];
74     rt[id]=rt[id<<1|1];
75     lf[id]=lf[id<<1];
76 }
77 }
78
79 struct node
80 {
81     int l,r,h;
82     char val;
83     inline bool operator<(const node &a)const
84     {
85         return h==a.h?val<a.val:h<a.h; // trick
            watch out. val<a.val? val>a.val?
86     }
87     inline void print()
88     {
89         printf("%d_%d_%d_%d\n",l,r,h,val);
90     }
91 }ln[inf];
92
93 int main()
94 {
95     make(1,1,inf);
96     scanf("%d",&n);
97     n<<=1;
98     map.clear();
99     for(i=0;i<n;++i)
100     {
101         scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
102         ln[i].l=x1;
103         ln[i].r=x2;
104         ln[i].h=y1;
105         ln[i].val=1;
106         ln[++i].l=x1;
107         ln[i].r=x2;
108         ln[i].h=y2;
109         ln[i].val=-1;
110         map[x1]=1;
111         map[x2]=1;
112     }
113     i=1;
114     for(it=map.begin();it!=map.end();++it,++i)
115     {
116         it->second=i;
117         rmap[i]=it->first;
118     }
119     i=0;
120     std::sort(ln,ln+n);
121     update(1,1,inf,map[ln[0].l]+1,map[ln[0].r],ln
        [0].val);
122     sum+=len[1];
123     last=len[1];
124     for(i=1;i<n;++i)
125     {
126         sum+=2*seg[1]*(ln[i].h-ln[i-1].h);

```

```

127     update(1,1,inf,map[ln[i].l]+1,map[ln[i].l],ln[i].val);
128     sum+=abs(len[1]-last);
129     last=len[1];
130 }
131 printf("%lld\n",sum);
132 return 0;
133 }

```

1.9 Size Blanced Tree

```

1 template<class Tp>class sbt
2 {
3     public:
4         inline void init()
5         {
6             rt=cnt=l[0]=r[0]=sz[0]=0;
7         }
8         inline void ins(const Tp &a)
9         {
10             ins(rt,a);
11         }
12         inline void del(const Tp &a)
13         {
14             del(rt,a);
15         }
16         inline bool find(const Tp &a)
17         {
18             return find(rt,a);
19         }
20         inline Tp pred(const Tp &a)
21         {
22             return pred(rt,a);
23         }
24         inline Tp succ(const Tp &a)
25         {
26             return succ(rt,a);
27         }
28         inline bool empty()
29         {
30             return !sz[rt];
31         }
32         inline Tp min()
33         {
34             return min(rt);
35         }
36         inline Tp max()
37         {
38             return max(rt);
39         }
40         inline void delsmall(const Tp &a)
41         {
42             dels(rt,a);
43         }
44         inline int rank(const Tp &a)
45         {
46             return rank(rt,a);
47         }
48         inline Tp sel(const int &a)
49         {
50             return sel(rt,a);
51         }
52         inline Tp 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)

```

```

{
    int k(l[pos]);
    l[pos]=r[k];
    r[k]=pos;
    sz[k]=sz[pos];
    sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
    pos=k;
}
inline void lro(int &pos)
{
    int k(r[pos]);
    r[pos]=l[k];
    l[k]=pos;
    sz[k]=sz[pos];
    sz[pos]=sz[l[pos]]+sz[r[pos]]+1;
    pos=k;
}
inline void mt(int &pos,bool flag)
{
    if(!pos)
        return;
    if(flag)
        if(sz[r[r[pos]]]>sz[l[pos]])
            lro(pos);
        else
            if(sz[l[r[pos]]]>sz[l[pos]])
            {
                rro(r[pos]);
                lro(pos);
            }
            else
                return;
    else
        if(sz[l[l[pos]]]>sz[r[pos]])
            rro(pos);
        else
            if(sz[r[l[pos]]]>sz[r[pos]])
            {
                lro(l[pos]);
                rro(pos);
            }
            else
                return;
    mt(l[pos],false);
    mt(r[pos],true);
    mt(pos,false);
    mt(pos,true);
}
void ins(int &pos,const Tp &a)
{
    if(pos)
    {
        ++sz[pos];
        if(a<val[pos])
            ins(l[pos],a);
        else
            ins(r[pos],a);
        mt(pos,a>=val[pos]);
        return;
    }
    pos==++cnt;
    l[pos]=r[pos]=0;
    val[pos]=a;
    sz[pos]=1;
}
Tp del(int &pos,const Tp &a)
{
    --sz[pos];
    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]
135                             ]+1);
136             return ret;
137         }
138         else
139             if(a<val[pos])
140                 return del(l[pos], a);
141             else
142                 return del(r[pos], a);
143     }
144     bool find(int &pos, const Tp &a)
145     {
146         if(!pos)
147             return false;
148         if(a<val[pos])
149             return find(l[pos], a);
150         else
151             return (val[pos]==a || find(r[
152     ] , a));
153     }
154     Tp pred(int &pos, const Tp &a)
155     {
156         if(!pos)
157             return a;
158         if(a>val[pos])
159         {
160             Tp ret(pred(r[pos], a));
161             if(ret==a)
162                 return val[pos];
163             else
164                 return ret;
165         }
166         return pred(l[pos], a);
167     }
168     Tp succ(int &pos, const Tp &a)
169     {
170         if(!pos)
171             return a;
172         if(a<val[pos])
173         {
174             Tp ret(succ(l[pos], a));
175             if(ret==a)
176                 return val[pos];
177             else
178                 return ret;
179         }
180         return succ(r[pos], a);
181     }
182     Tp min(int &pos)
183     {
184         if(l[pos])
185             return min(l[pos]);
186         else
187             return val[pos];
188     }
189     Tp max(int &pos)
190     {
191         if(r[pos])
192             return max(r[pos]);
193         else
194             return val[pos];
195     }
196     void dels(int &pos, const Tp &v)
197     {
198         if(!pos)
199             return;
200         if(val[pos]<v)
201         {
202             pos=r[pos];
203             dels(pos, v);
204             return;
205         }
206         dels(l[pos], v);
207         sz[pos]=1+sz[l[pos]]+sz[r[pos]];
208     }
209     int rank(const int &pos, const Tp &v)
210     {
211         if(val[pos]==v)
212             return sz[l[pos]]+1;
213         if(v<val[pos])
214             return rank(l[pos], v);
215         return rank(r[pos], v)+sz[l[pos]]+1;
216     }
217     Tp sel(const int &pos, const int &v)
218     {
219         if(sz[l[pos]]+1==v)
220             return val[pos];
221         if(v>sz[l[pos]])
222             return sel(r[pos], v-sz[l[pos]]-1);
223         ;
224         return sel(l[pos], v);
225     }
226     Tp dsel(int &pos, int k)
227     {
228         --sz[pos];
229         if(sz[l[pos]]+1==k)
230         {
231             Tp re(val[pos]);
232             if(!l[pos] || !r[pos])
233                 pos=l[pos]+r[pos];
234             else
235                 val[pos]=del(l[pos], val[pos]
236                             ]+1);
237             return re;
238         }
239         if(k>sz[l[pos]])
240             return dsel(r[pos], k-1-sz[l[pos]
241                             ]]);
242         return dsel(l[pos], k);
243     }
244 };

```

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)      15
23     {                                16
24         std::cin >> mat[i][j];      17
25         table[0][0][i][j] = mat[i][j]; 18
26     }                                19
27                                     20
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)            21
34             continue;                22
35         for (int ii=0; ii+(1<<j)<=n; ++ii) 23
36             for (int jj=0; jj+(1<<i)<=n; ++jj) 24
37                 if (i==0)            25
38                     table[i][j][ii][jj]=26
39                         std::min(table[i]27
40                             ][j-1][ii][jj], 28
41                             table[i][j-1][ii29
42                                 ]+(1<<(j-1))[jj])
43                             ;
44     }
45     else
46         table[i][j][ii][jj]=31
47             std::min(table[i
48                 -1][j][ii][jj],
49                 table[i-1][j][ii
50                     ][jj+(1<<(i-1))])
51             ;
52 }
53 }
54 long long N;
55 std::cin >> N;
56 int r1, c1, r2, c2;
57 for (int i = 0; i < N; ++i)
58 {
59     scanf("%d%d%d%d",&r1,&c1,&r2,&c2);
60     --r1;
61     --c1;
62     --r2;
63     --c2;
64     int w=lg[c2-c1+1];
65     int h=lg[r2-r1+1];
66     printf("%d\n", std::min(table[w][h][r1][c1],
67         std::min(table[w][h][r1][c2-
68             (1<<w)+1], std::min(table[w][h][r2-
69                 (1<<h)+1][c1], table[w][h][r2-
70                     (1<<h)+1][c2-(1<<w)+1]))));
71 }
72 }
73 return 0;
74 }

```

1.11 Sparse Table - square

```

1 int num[MAXX][MAXX], max[MAXX][MAXX][10];
2 short lg[MAXX];
3
4 int main()
5 {
6     for (i=2; i<MAXX; ++i)
7         lg[i]=lg[i>>1]+1;
8     scanf("%hd_%d",&n,&q);
9     for (i=0; i<n; ++i)
10         for (j=0; j<n; ++j)
11         {
12             scanf("%d", num[i][j]);
13             max[i][j][0]=num[i][j];
14         }

```

```

15     for (k=1; k<=lg[n]; ++k)
16     {
17         l=n+1-(1<<k);
18         for (i=0; i<l; ++i)
19             for (j=0; j<l; ++j)
20                 max[i][j][k]=std::max(std::max(
21                     max[i][j][k-1], max[i+(1<<(k-1))][j][k-1]),
22                     std::max(max[i][j+(1<<(k-1))][k-1],
23                         max[i+(1<<(k-1))][j+(1<<(k-1))][k-1]));
24     }
25     printf("Case_%hd:\n", t);
26     while (q--)
27     {
28         scanf("%hd_%hd_%hd",&i,&j,&l);
29         --i;
30         --j;
31         k=lg[l];
32         printf("%d\n", std::max(std::max(max[i][j][k],
33             max[i][j+l-(1<<k)][k]), std::max(
34                 max[i+l-(1<<k)][j][k], max[i+l-(1<<k)][j+l-(1<<k)][k]));
35     }
36 }

```

1.12 Sparse Table

```

1 int num[MAXX], min[MAXX][20];
2 int lg[MAXX];
3
4 int main()
5 {
6     for (i=2; i<MAXX; ++i)
7         lg[i]=lg[i>>1]+1;
8     scanf("%d_%d",&n,&q);
9     for (i=1; i<=n; ++i)
10     {
11         scanf("%d", num[i]);
12         min[i][0]=num[i];
13     }
14     for (j=1; j<=lg[n]; ++j)
15     {
16         l=n+1-(1<<j);
17         j_-=j-1;
18         j_=(1<<j_);
19         for (i=1; i<=l; ++i)
20             min[i][j]=std::min(min[i][j_], min[i+j_][j_]);
21     }
22     printf("Case_%hd:\n", t);
23     while (q--)
24     {
25         scanf("%d_%d",&i,&j);
26         k=lg[j-i+1];
27         printf("%d\n", std::min(min[i][k], min[j-(1<<k)+1][k]));
28     }
29 }
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
14 [0]=ch[1]=null;}
15
16 class Treap
17 {
18     inline void up(node *pos)
19     {
20         pos->sz=pos->ch[0]->sz+pos->ch[1]->sz+1;
21     }
22     inline void rot(node *&pos, int tp)
23     {
24         node *k(pos->ch[tp]);
25         pos->ch[tp]=k->ch[tp^1];
26         k->ch[tp^1]=pos;
27         up(pos);
28         up(k);
29         pos=k;
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
73 [0]->key);
74                 if(pos->ch[t]==null)
75                 {
76                     delete pos;
77                     pos=null;
78                     return;
79                 }
80                 rot(pos, t);
81                 del(pos->ch[t^1], val);
82             }
83             else
84                 del(pos->ch[val>pos->val], val);
85             up(pos);
86         }
87     }
88     public:
89     node *rt;
90
91     Treap():rt(null){}
92     inline void insert(int val)
93     {
94         insert(rt, val);
95     }
96     inline void reset()
97     {
98         rec(rt);
99         rt=null;
100     }
101     inline int sel(int k)
102     {
103         if(k<1 || k>rt->sz)
104             return 0;
105         return sel(rt, rt->sz+1-k);
106     }
107     inline void del(int val)
108     {
109         del(rt, val);
110     }
111     inline int size()
112     {
113         return rt->sz;
114     }
115 }treap[MAXX];
116
117 init:
118 {
119     srand(time(0));
120     null=new node();
121     null->val=0xc0c0c0c0;
122     null->sz=0;
123     null->key=RAND_MAX;
124     null->ch[0]=null->ch[1]=null;
125     for(i=0; i<MAXX; ++i)
126         treap[i].rt=null;
127 }

```

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
6     (zz) {}
7     pv operator -(const pv& b) const
8     {
9         return pv(x-b.x, y-b.y, z-b.z);
10    }
11    pv operator *(const pv& b) const
12    {
13        return pv(y*b.z-z*b.y, z*b.x-x*b.z, x*b.y-y*b.x
14        );
15    }

```

```

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)
38             *pa.y+(x*z*(1-c)+y*s)*pa.z,
39             (y*x*(1-c)+z*s)*pa.x+(y*y*(1-c)-x*s)
40             *pa.y+(y*z*(1-c)-x*s)*pa.z,
41             (x*z*(1-c)-y*s)*pa.x+(y*z*(1-c)-x*s)
42             *pa.y+(z*z*(1-c)+c)*pa.z);
43     return pp;
44 }
45
46 //经纬度转换
47 x=r*sin ()*cos ();
48 y=r*sin ()*sin ();
49 z=r*cos ();
50
51 r=sqrt(x*2+y*2+z*2);///??
52 r=sqrt(x^2+y^2+z^2);///??
53
54 =atan(y/x);
55 =acos(z/r);
56
57 r∞[0,]
58 [0,2]
59 [0,]
60
61 lat1 [-/2,/2]
62 lng1 [-,]
63
64 pv getpv(double lat,double lng,double r)
65 {
66     lat += pi/2;
67     lng += pi;
68     return
69         pv(r*sin(lat)*cos(lng),r*sin(lat)*sin(lng),
70             cos(lat));
71 }
72
73 //经纬度球面距离
74
75 #include<cstdio>
76 #include<cmath>
77
78 #define MAXX 1111
79 char buf[MAXX];
80
81 const double r=6875.0/2,pi=acos(-1.0);
82 double a,b,c,x1,x2,y2,ans;
83
84 int main()
85 {
86     double y1;
87     while(gets(buf)!=NULL)
88     {
89         gets(buf);
90         gets(buf);
91
92         scanf("%lf^%lf'%lf\"_\"%s\n",&a,&b,&c,buf);
93         x1=a+b/60+c/3600;
94         x1=x1*pi/180;
95         if(buf[0]=='S')
96             x1=-x1;
97
98         scanf("%s",buf);
99         scanf("%lf^%lf'%lf\"_\"%s\n",&a,&b,&c,buf);
100         y1=a+b/60+c/3600;
101         y1=y1*pi/180;
102         if(buf[0]=='W')
103             y1=-y1;
104
105         gets(buf);
106
107         scanf("%lf^%lf'%lf\"_\"%s\n",&a,&b,&c,buf);
108         x2=a+b/60+c/3600;
109         x2=x2*pi/180;
110         if(buf[0]=='S')
111             x2=-x2;
112
113         scanf("%s",buf);
114         scanf("%lf^%lf'%lf\"_\"%s\n",&a,&b,&c,buf);
115         y2=a+b/60+c/3600;
116         y2=y2*pi/180;
117         if(buf[0]=='W')
118             y2=-y2;
119
120         ans=acos(cos(x1)*cos(x2)*cos(y1-y2)+sin(
121             x1)*sin(x2))*r;
122         printf("The distance to the iceberg: %.2
123             lf miles.\n",ans);
124         if(ans+0.005<100)
125             puts("DANGER!");
126
127         gets(buf);
128     }
129     return 0;
130 }
131
132 inline bool ZERO(const double &a)
133 {
134     return fabs(a)<eps;
135 }
136
137 //三维向量是否为零
138 inline bool ZERO(pv p)
139 {
140     return (ZERO(p.x) && ZERO(p.y) && ZERO(p.z));
141 }
142
143 //直线相交
144 bool LineIntersect(Line3D L1, Line3D L2)
145 {
146     pv s = L1.s-L1.e;
147     pv e = L2.s-L2.e;
148     pv p = s*e;
149     if (ZERO(p))
150         return false; //是否平行

```

```

147     p = (L2.s-L1.e)*(L1.s-L1.e);
148     return ZERO(p&L2.e);          //是否共面
149 }
150
151 //线段相交
152 bool inter(pv a,pv b,pv c,pv d)
153 {
154     pv ret = (a-b)*(c-d);
155     pv t1 = (b-a)*(c-a);
156     pv t2 = (b-a)*(d-a);
157     pv t3 = (d-c)*(a-c);
158     pv t4 = (d-c)*(b-c);
159     return sgn(t1&ret)*sgn(t2&ret) < 0 && sgn(t3&
        ret)*sgn(t4&ret) < 0;
160 }
161
162 //点在直线上
163 bool OnLine(pv p, Line3D L)
164 {
165     return ZERO((p-L.s)*(L.e-L.s));
166 }
167
168 //点在线段上
169 bool OnSeg(pv p, Line3D L)
170 {
171     return (ZERO((L.s-p)*(L.e-p)) && EQ(Norm(p-L
        s)+Norm(p-L.e),Norm(L.e-L.s)));
172 }
173
174 //点到直线距离
175 double Distance(pv p, Line3D L)
176 {
177     return (Norm((p-L.s)*(L.e-L.s))/Norm(L.e-L.s)
        );
178 }
179
180 //线段夹角
181 //范围值为 之间的弧度[0,/]
182 double Inclination(Line3D L1, Line3D L2)
183 {
184     pv u = L1.e - L1.s;
185     pv v = L2.e - L2.s;
186     return acos( (u & v) / (Norm(u)*Norm(v)) );
187 }

```

2.2 3DCH

```

1 #include<cstdio>
2 #include<cmath>
3 #include<vector>
4 #include<algorithm>
5
6 #define MAXX 1111
7 #define eps 1e-8
8 #define inf 1e20
9
10 struct pv
11 {
12     double x,y,z;
13     pv(){}
14     pv(const double &xx,const double &yy,const
        double &zz):x(xx),y(yy),z(zz){}
15     inline pv operator-(const pv &i) const
16     {
17         return pv(x-i.x,y-i.y,z-i.z);
18     }
19     inline pv operator*(const pv &i) const //叉积
20     {
21         return pv(y*i.z-z*i.y,z*i.x-x*i.z,x*i.y-
            y*i.x);
22     }
23     inline double operator^(const pv &i) const //
        点
        积
24     {
25         return x*i.x+y*i.y+z*i.z;
26     }
27     inline double len()
28     {
29         return sqrt(x*x+y*y+z*z);
30     }
31 };
32
33 struct pla
34 {
35     short a,b,c;
36     bool ok;
37     pla(){}
38     pla(const short &aa,const short &bb,const
        short &cc):a(aa),b(bb),c(cc),ok(true){}
39     inline void set();
40     inline void print()
41     {
42         printf("%hd_%hd_%hd\n",a,b,c);
43     }
44 };
45
46 pv pnt[MAXX];
47 std::vector<pla>fac;
48 short to[MAXX][MAXX];
49
50 inline void pla::set()
51 {
52     to[a][b]=to[b][c]=to[c][a]=fac.size();
53 }
54
55 inline double ptof(const pv &p,const pla &f) //点
        面距离?
56 {
57     return (pnt[f.b]-pnt[f.a])*(pnt[f.c]-pnt[f.a]
        )^(p-pnt[f.a]);
58 }
59
60 inline double vol(const pv &a,const pv &b,const
        pv &c,const pv &d)//有向体积,即六面体
        积*6
61 {
62     return (b-a)*(c-a)^(d-a);
63 }
64
65 inline double ptof(const pv &p,const short &f) //
        点到号面的距
        离pf
66 {
67     return fabs(vol(pnt[fac[f].a],pnt[fac[f].b],
        pnt[fac[f].c],p)/((pnt[fac[f].b]-pnt[fac[
        f].a])*(pnt[fac[f].c]-pnt[fac[f].a])).len
        ());
68 }
69
70 void dfs(const short&,const short&);
71
72 void deal(const short &p,const short &a,const
        short &b)
73 {
74     if(fac[to[a][b]].ok)
75         if(ptof(pnt[p],fac[to[a][b]])>eps)
76             dfs(p,to[a][b]);
77     else
78     {

```



```

79         pla.add(b,a,p);
80         add.set();
81         fac.push_back(add);
82     }
83 }
84
85 void dfs(const short &p,const short &now)
86 {
87     fac[now].ok=false;
88     deal(p,fac[now].b,fac[now].a);
89     deal(p,fac[now].c,fac[now].b);
90     deal(p,fac[now].a,fac[now].c);
91 }
92
93 inline void make()
94 {
95     fac.resize(0);
96     if(n<4)
97         return;
98
99     for(i=1;i<n;++i)
100         if((pnt[0]-pnt[i]).len(>eps)
101         {
102             std::swap(pnt[i],pnt[1]);
103             break;
104         }
105     if(i==n)
106         return;
107
108     for(i=2;i<n;++i)
109         if(((pnt[0]-pnt[1])*(pnt[1]-pnt[i])).len(>eps)
110         {
111             std::swap(pnt[i],pnt[2]);
112             break;
113         }
114     if(i==n)
115         return;
116
117     for(i=3;i<n;++i)
118         if(fabs((pnt[0]-pnt[1])*(pnt[1]-pnt[2])*(pnt[2]-pnt[i]))>eps)
119         {
120             std::swap(pnt[3],pnt[i]);
121             break;
122         }
123     if(i==n)
124         return;
125
126     for(i=0;i<4;++i)
127     {
128         pla.add((i+1)%4,(i+2)%4,(i+3)%4);
129         if(ptof(pnt[i],add)>0)
130             std::swap(add.c,add.b);
131         add.set();
132         fac.push_back(add);
133     }
134     for(;i<n;++i)
135         for(j=0;j<fac.size();++j)
136             if(fac[j].ok && ptof(pnt[i],fac[j]).len(>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)
163 //两面是否相等
164 {
165     pv &a=pnt[fac[s].a],&b=pnt[fac[s].b],&c=pnt[fac[s].c];
166     return fabs(vol(a,b,c,pnt[fac[t].a]))<eps &&
167         fabs(vol(a,b,c,pnt[fac[t].b]))<eps &&
168         fabs(vol(a,b,c,pnt[fac[t].c]))<eps;
169 }
170
171 //表面多边形数目
172 inline short facetcnt()
173 {
174     short ans=0;
175     for(short i=0;i<fac.size();++i)
176     {
177         for(j=0;j<i;++j)
178             if(same(i,j))
179                 break;
180         if(j==i)
181             ++ans;
182     }
183     return ans;
184 }
185
186 //表面三角形数目
187 inline short trianglecnt()
188 {
189     return fac.size();
190 }
191
192 //三点构成的三角形面积*2
193 inline double area(const pv &a,const pv &b,const pv &c)
194 {
195     return (b-a)*(c-a).len();
196 }
197
198 //表面积
199 inline double area()
200 {
201     double ret(0);
202     for(i=0;i<fac.size();++i)
203         ret+=area(pnt[fac[i].a],pnt[fac[i].b],pnt[fac[i].c]);
204     return ret/2;
205 }
206
207 //体积
208 inline double volume()
209 {

```

```

207     pv o(0,0,0);
208     double ret(0);
209     for(short i(0);i<fac.size();++i)
210         ret+=vol(o,pnt[fac[i].a],pnt[fac[i].b],
211             pnt[fac[i].c]);
212     return fabs(ret/6);
}

2.3 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[
6             i].r);
7         del[i] = false;
8     }
9     for (int i = 0; i < n; i++)
10         if (del[i] == false)
11         {
12             if (c[i].r == 0.0)
13                 del[i] = true;
14             for (int j = 0; j < n; j++)
15                 if (i != j)
16                     if (del[j] == false)
17                         if (cmp(Point(c[i].c,c[j
18                             ].c).Len()+c[i].r,c[j
19                             ].r) <= 0)
20                             del[i] = true;
21         }
22     }
23     tn = n;
24     n = 0;
25     for (int i = 0; i < tn; i++)
26         if (del[i] == false)
27             c[n++] = c[i];
28 }
29 //ans[i表示被覆盖/次的面积i
30 const double pi = acos(-1.0);
31 const double eps = 1e-8;
32 struct Point
33 {
34     double x,y;
35     Point(){}
36     Point(double _x,double _y)
37     {
38         x = _x;
39         y = _y;
40     }
41     double Length()
42     {
43         return sqrt(x*x+y*y);
44     }
45 };
46 struct Circle
47 {
48     Point c;
49     double r;
50 };
51 struct Event
52 {
53     double tim;
54     int typ;
55     Event(){}
56     Event(double _tim,int _typ)
57     {
58         tim = _tim;
59         typ = _typ;
60     }
61 };
62 int cmp(const double& a,const double& b)
63 {
64     if (fabs(a-b) < eps) return 0;
65     if (a < b) return -1;
66     return 1;
67 }
68 bool Eventcmp(const Event& a,const Event& b)
69 {
70     return cmp(a.tim,b.tim) < 0;
71 }
72 double Area(double theta,double r)
73 {
74     return 0.5*r*r*(theta-sin(theta));
75 }
76 double xmult(Point a,Point b)
77 {
78     return a.x*b.y-a.y*b.x;
79 }
80 int n,cur,tote;
81 Circle c[1000];
82 double ans[1001],pre[1001],AB,AC,BC,theta,fai,a0,
83     a1;
84 Event e[4000];
85 Point lab;
86 int main()
87 {
88     while (scanf("%d",&n) != EOF)
89     {
90         for (int i = 0; i < n; i++)
91             scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y
92                 ,&c[i].r);
93         for (int i = 1; i <= n; i++)
94             ans[i] = 0.0;
95         for (int i = 0; i < n; i++)
96         {
97             tote = 0;
98             e[tote++] = Event(-pi,1);
99             e[tote++] = Event(pi,-1);
100             for (int j = 0; j < n; j++)
101                 if (j != i)
102                 {
103                     lab = Point(c[j].c.x-c[i].c.x
104                         ,c[j].c.y-c[i].c.y);
105                     AB = lab.Length();
106                     AC = c[i].r;
107                     BC = c[j].r;
108                     if (cmp(AB+AC,BC) <= 0)
109                     {
110                         e[tote++] = Event(-pi,1);
111                         e[tote++] = Event(pi,-1);
112                         continue;
113                     }
114                     if (cmp(AB+BC,AC) <= 0)
115                         continue;
116                     if (cmp(AB,AC+BC) > 0)
117                         continue;
118                     theta = atan2(lab.y,lab.x);
119                     fai = acos((AC*AC+AB*AB-BC*BC
120                         )/(2.0*AC*AB));
121                     a0 = theta-fai;
122                     if (cmp(a0,-pi) < 0) a0 +=
123                         2*pi;
124                     a1 = theta+fai;

```

```

121         if (cmp(a1, pi) > 0) a1 -= 2*pi;
122         pi;
123         if (cmp(a0, a1) > 0)
124         {
125             e[tote++] = Event(a0, 1);
126             e[tote++] = Event(pi, -1);
127             e[tote++] = Event(-pi, 1);
128             e[tote++] = Event(a1, -1);
129         }
130         else
131         {
132             e[tote++] = Event(a0, 1);
133             e[tote++] = Event(a1, -1);
134         }
135         sort(e, e+tote, Eventcmp);
136         cur = 0;
137         for (int j = 0; j < tote; j++)
138         {
139             if (cur != 0 && cmp(e[j].tim, pre[
140                 cur]) != 0)
141             {
142                 ans[cur] += Area(e[j].tim - pre[
143                     cur], c[i].r);
144                 ans[cur] += xmult(Point(c[i].
145                     c.x+c[i].r*cos(pre[cur]),
146                     c[i].c.y+c[i].r*sin(pre[
147                         cur])),
148                     Point(c[i].c.x+c[i].r
149                         *cos(e[j].tim), c[i].
150                         c.y+c[i].r*sin(e[
151                             j].tim)))/2.0;
152             }
153             cur += e[j].typ;
154             pre[cur] = e[j].tim;
155         }
156     }
157     for (int i = 1; i < n; i++)
158         ans[i] -= ans[i+1];
159     for (int i = 1; i <= n; i++)
160         printf("%d] = %.3f\n", i, ans[i]);
161     return 0;
162 }

```

2.4 circle

```

1 //单位圆覆盖
2 #include<cstdio>
3 #include<cmath>
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)
15     (yy){}
16     inline pv operator-(const pv &i)const
17     {
18         return pv(x-i.x,y-i.y);
19     }
20     inline double cross(const pv &i)const
21     {
22         return x*i.y-y*i.x;
23     }
24 }
25
26 inline void print()
27 {
28     printf("%lf %lf\n",x,y);
29 }
30
31 inline double len()
32 {
33     return sqrt(x*x+y*y);
34 }
35
36 pnt[MAXX];
37
38 struct node
39 {
40     double k;
41     bool flag;
42     node(){}
43     node(const double &kk,const bool &ff):k(kk),
44     flag(ff){}
45     inline bool operator<(const node &i)const
46     {
47         return k<i.k;
48     }
49 };
50
51 std::vector<node>alpha;
52
53 short n,i,j,k,l;
54 short ans,sum;
55 double R=2;
56 double theta,phi,d;
57 const double pi(acos(-1.0));
58
59 int main()
60 {
61     alpha.reserve(MAXX<1);
62     while(scanf("%hd",&n),n)
63     {
64         for(i=0;i<n;++i)
65             scanf("%lf %lf",&pnt[i].x,&pnt[i].y);
66         ans=0;
67         for(i=0;i<n;++i)
68         {
69             alpha.resize(0);
70             for(j=0;j<n;++j)
71                 if(i!=j)
72                 {
73                     if((d=(pnt[i]-pnt[j]).len())>
74                         R)
75                         continue;
76                     if((theta=atan2(pnt[j].y-pnt[
77                         i].y,pnt[j].x-pnt[i].x))
78                         <0)
79                         theta+=2*pi;
80                     phi=acos(d/R);
81                     alpha.push_back(node(theta-
82                         phi,true));
83                     alpha.push_back(node(theta+
84                         phi,false));
85                 }
86             std::sort(alpha.begin(),alpha.end());
87             for(j=0;j<alpha.size();++j)
88             {
89                 if(alpha[j].flag)
90                     ++sum;
91                 else
92                     --sum;
93                 ans=std::max(ans,sum);
94             }
95             printf("%hd\n",ans+1);
96         }
97     }
98 }

```

```

87     return 0;
88 }
89 //最小覆盖圆
90 #include<stdio>
91 #include<cmath>
92 #define MAXX 511
93 #define eps 1e-8
94
95 struct pv
96 {
97     double x,y;
98     pv(){}
99     pv(const double &xx,const double &yy):x(xx),y(yy){}
100
101     inline pv operator-(const pv &i) const
102     {
103         return pv(x-i.x,y-i.y);
104     }
105     inline pv operator+(const pv &i) const
106     {
107         return pv(x+i.x,y+i.y);
108     }
109     inline double cross(const pv &i) const
110     {
111         return x*i.y-y*i.x;
112     }
113     inline double len()
114     {
115         return sqrt(x*x+y*y);
116     }
117     inline pv operator/(const double &a) const
118     {
119         return pv(x/a,y/a);
120     }
121     inline pv operator*(const double &a) const
122     {
123         return pv(x*a,y*a);
124     }
125 } pnt[MAXX],o,t1,lt,aa,bb,cc,dd;
126
127 short n,i,j,k,l;
128 double r,u;
129
130 inline pv ins(const pv &a1,const pv &a2,const pv &b1,const pv &b2)
131 {
132     t1=a2-a1;
133     lt=b2-b1;
134     u=(b1-a1).cross(lt)/(t1).cross(lt);
135     return a1+t1*u;
136 }
137
138 inline pv get(const pv &a,const pv &b,const pv &c)
139 {
140     aa=(a+b)/2;
141     bb.x=aa.x-a.y+b.y;
142     bb.y=aa.y+a.x-b.x;
143     cc=(a+c)/2;
144     dd.x=cc.x-a.y+c.y;
145     dd.y=cc.y+a.x-c.x;
146     return ins(aa,bb,cc,dd);
147 }
148
149 int main()
150 {
151     while(scanf("%hd",&n),n)
152     {
153         for(i=0;i<n;++i)
154             scanf("%lf%lf",&pnt[i].x,&pnt[i].y);
155         o=pnt[0];
156         r=0;
157         for(i=1;i<n;++i)
158             if((pnt[i]-o).len()>r+eps)
159             {
160                 o=pnt[i];
161                 r=0;
162                 for(j=0;j<i;++j)
163                     if((pnt[j]-o).len()>r+eps)
164                     {
165                         o=(pnt[i]+pnt[j])/2;
166                         r=(o-pnt[j]).len();
167                         for(k=0;k<j;++k)
168                             if((o-pnt[k]).len()>r+eps)
169                             {
170                                 o=get(pnt[i],pnt[j],pnt[k]);
171                                 r=(o-pnt[i]).len();
172                             }
173                         }
174                     }
175             }
176         printf("%.2lf %.2lf %.2lf\n",o.x,o.y,r);
177     }
178     return 0;
179 }
180
181 //两原面积交
182 double dis(int x,int y)
183 {
184     return sqrt((double)(x*x+y*y));
185 }
186
187 double area(int x1,int y1,int x2,int y2,double r1,double r2)
188 {
189     double s=dis(x2-x1,y2-y1);
190     if(r1+r2<s) return 0;
191     else if(r2-r1>s) return PI*r1*r1;
192     else if(r1-r2>s) return PI*r2*r2;
193     double q1=acos((r1*r1+s*s-r2*r2)/(2*r1*s));
194     double q2=acos((r2*r2+s*s-r1*r1)/(2*r2*s));
195     return (r1*r1*q1+r2*r2*q2-r1*s*sin(q1));
196 }
197
198 //三角形外接圆
199 {
200     for(int i=0;i<3;i++)
201         scanf("%lf%lf",&p[i].x,&p[i].y);
202     tp=pv((p[0].x+p[1].x)/2,(p[0].y+p[1].y)/2);
203     l[0]=Line(tp,pv(tp.x-(p[1].y-p[0].y),tp.y+(p[1].x-p[0].x)));
204     tp=pv((p[0].x+p[2].x)/2,(p[0].y+p[2].y)/2);
205     l[1]=Line(tp,pv(tp.x-(p[2].y-p[0].y),tp.y+(p[2].x-p[0].x)));
206     tp=LineToLine(l[0],l[1]);
207     r=pv(tp,p[0]).Length();
208     printf("%.6f,%.6f,%.6f\n",tp.x,tp.y,r);
209 }
210
211 //三角形内切圆
212 {
213     for(int i=0;i<3;i++)
214         scanf("%lf%lf",&p[i].x,&p[i].y);
215     if(xmult(pv(p[0],p[1]),pv(p[0],p[2]))<0)
216         swap(p[1],p[2]);
217

```

```

218     for (int i = 0; i < 3; i++) 44
219         len[i] = pv(p[i], p[(i+1)%3]).Length(); 45
220     tr = (len[0]+len[1]+len[2])/2; 46
221     r = sqrt((tr-len[0])*(tr-len[1])*(tr-len[2])/ 47
222         tr);
223     for (int i = 0; i < 2; i++) 48
224     { 49
225         v = pv(p[i], p[i+1]); 49
226         tv = pv(-v.y, v.x); 50
227         tr = tv.Length(); 51
228         tv = pv(tv.x*r/tr, tv.y*r/tr); 52
229         tp = pv(p[i].x+tv.x, p[i].y+tv.y); 53
230         l[i].s = tp; 54
231         tp = pv(p[i+1].x+tv.x, p[i+1].y+tv.y); 55
232         l[i].e = tp; 56
233     } 57
234     tp = LineToLine(l[0], l[1]); 58
235     printf("%.6f, %.6f, %.6f)\n", tp.x, tp.y, r); 59
} 59

2.5 closest point pair 60
61
1 //演算法笔记1 62
2 63
3 struct Point {double x, y;} p[10], t[10]; 64
4 bool cmpx(const Point& i, const Point& j) {return 65
5     i.x < j.x;} 66
6 bool cmpy(const Point& i, const Point& j) {return 67
7     i.y < j.y;} 68
8 69
9 double DnC(int L, int R) 70
10 { 71
11     if (L >= R) return 1e9; // 沒有點、只有一個點。 72
12 73
13     /* : 把所有點分成左右兩側，點數盡量一樣多。Divide 74
14 75
15     int M = (L + R) / 2; 76
17
18     /* : 左側、右側分別遞迴求解。Conquer */ 77
19 78
20     double d = min(DnC(L, M), DnC(M+1, R)); 79
21     // if (d == 0.0) return d; // 提早結束 80
22 81
23     /* : 尋找靠近中線的點，並依座標排序。 82
24         MergeYO(NlogN)。 */ 83
25 84
26     int N = 0; // 靠近中線的點數目 85
27     for (int i=M; i>=L && p[M].x - p[i].x < d 86
28         —i) t[N++] = p[i]; 87
29     for (int i=M+1; i<=R && p[i].x - p[M].x < d 88
30         ++i) t[N++] = p[i]; 89
31     sort(t, t+N, cmpy); // Quicksort O(NlogN) 90
32 91
33     /* : 尋找橫跨兩側的最近點對。MergeO(N)。 */ 92
34 93
35     for (int i=0; i<N-1; ++i) 94
36     for (int j=1; j<=2 && i+j<N; ++j) 95
37         d = min(d, distance(t[i], t[i+j])); 96
38 97
39     return d; 98
40 } 99
41 100
42 double closest_pair() 101
43 { 102
44     sort(p, p+10, cmpx); 103
45     return DnC(0, N-1); 104
46 } 105
47 106
48 //演算法笔记2 107

```

```

108 //别忘了排序
109 bool operator<(const Point &a ,const Point &b) {
110     if(a.y != b.y) return a.x < b.x;
111     return a.x < b.x;
112 }
113 double Gao(int l ,int r ,Point pnts[]) {
114     double ret = inf;
115     if(l == r) return ret;
116     if(l+1 == r) {
117         ret = min(calc_dis(pnts[l], pnts[l+1]), ret);
118         return ret;
119     }
120     if(l+2 == r) {
121         ret = min(calc_dis(pnts[l], pnts[l+1]), ret);
122         ret = min(calc_dis(pnts[l], pnts[l+2]), ret);
123         ret = min(calc_dis(pnts[l+1], pnts[l+2]), ret);
124         return ret;
125     }
126
127     int mid = l+r>>1;
128     ret = min (ret ,Gao(l ,mid, pnts));
129     ret = min (ret , Gao(mid+1, r, pnts));
130
131     for(int c = l ; c<=r; c++)
132         for(int d = c+1; d <=c+7 && d<=r; d++) {
133             ret = min(ret , calc_dis(pnts[c], pnts[d]));
134         }
135     return ret;
136 }
137
138 //增量
139 #include <iostream>
140 #include <cstdio>
141 #include <cstring>
142 #include <map>
143 #include <vector>
144 #include <cmath>
145 #include <algorithm>
146 #define Point pair<double, double>
147 using namespace std;
148
149 const int step[9][2] =
150     {{-1,-1},{-1,0},{-1,1},{0,-1},{0,0},{0,1},{1,-1},{1,0},{1,1}};
151
152 int n,x,y,nx,ny;
153 map<pair<int,int>,vector<Point>>> g;
154 vector<Point> tmp;
155 Point p[20000];
156 double tx,ty,ans,nowans;
157 vector<Point>::iterator it,op,ed;
158 pair<int,int> gird;
159 bool flag;
160
161 double Dis(Point p0,Point p1)
162 {
163     return sqrt((p0.first-p1.first)*(p0.first-p1.first)+
164                 (p0.second-p1.second)*(p0.second-p1.second));
165 }
166
167 double CalcDis(Point p0,Point p1,Point p2)
168 {
169     return Dis(p0,p1)+Dis(p0,p2)+Dis(p1,p2);
170 }
171
172 void build(int n,double w)
173 {
174     g.clear();
175
176     for (int i = 0;i < n;i++)
177         g[make_pair((int) floor(p[i].first/w),(int) floor(p[i].second/w))].push_back(p[i]);
178 }
179
180 int main()
181 {
182     int t;
183     scanf("%d",&t);
184     for (int ft = 1;ft <= t;ft++)
185     {
186         scanf("%d",&n);
187         for (int i = 0;i < n;i++)
188         {
189             scanf("%lf%lf",&tx,&ty);
190             p[i] = make_pair(tx,ty);
191         }
192         random_shuffle(p,p+n);
193         ans = CalcDis(p[0],p[1],p[2]);
194         build(3,ans/2.0);
195         for (int i = 3;i < n;i++)
196         {
197             x = (int) floor(2.0*p[i].first/ans);
198             y = (int) floor(2.0*p[i].second/ans);
199             tmp.clear();
200             for (int k = 0;k < 9;k++)
201             {
202                 nx = x+step[k][0];
203                 ny = y+step[k][1];
204                 gird = make_pair(nx,ny);
205                 if (g.find(gird) != g.end())
206                 {
207                     op = g[gird].begin();
208                     ed = g[gird].end();
209                     for (it = op;it != ed;it++)
210                         tmp.push_back(*it);
211                 }
212             }
213             flag = false;
214             for (int j = 0;j < tmp.size();j++)
215                 for (int k = j+1;k < tmp.size();k++)
216                 {
217                     nowans = CalcDis(p[i],tmp[j],tmp[k]);
218                     if(nowans < ans)
219                     {
220                         ans = nowans;
221                         flag = true;
222                     }
223                 }
224             if (flag == true)
225                 build(i+1,ans/2.0);
226             else
227                 g[make_pair((int) floor(2.0*p[i].first/ans),
228                             (int) floor(2.0*p[i].second/ans))].push_back(p[i]);
229         }
230         printf("%.3f\n",ans);
231     }
232 }

```

2.6 ellipse

$$\begin{aligned}
 1 & \quad \frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2} = 1 \\
 2 & \\
 3 & \quad x = h + a \times \cos(t) \\
 4 & \quad y = k + b \times \sin(t) \\
 5 & \\
 6 & \quad \text{area} = \pi \times a \times b \\
 7 & \quad \text{distance from center to focus: } f = \sqrt{a^2 - b^2}
 \end{aligned}$$

```

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

```

2.7 Graham's scan

```

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

```

2.8 half-plane intersection

```

1 | //解析几何方式abc
2 inline pv ins(const pv &p1, const pv &p2)
3 {
4     u=fabs(a*p1.x+b*p1.y+c);
5     v=fabs(a*p2.x+b*p2.y+c);
6     return pv((p1.x*v+p2.x*u)/(u+v), (p1.y*v+p2.y*
7         u)/(u+v));
8 }
9 inline void get(const pv& p1, const pv& p2, double
10     &a, double &b, double &c)
11 {
12     a=p2.y-p1.y;
13     b=p1.x-p2.x;
14     c=p2.x*p1.y-p2.y*p1.x;
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-
20         a*e));
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
44                     [k][l]));
45             m=(l+1)%p[k].size();
46             if(a*p[k][m].x+b*p[k][m].y+c<-eps)
47                 p[!k].push_back(ins(p[k][m], p
48                     [k][l]));
49         }
50         k=!k;
51         if(p[k].empty())
52             break;
53     }
54     //结果在p[k]中
55     return p[k].empty();
56 }
57 //计算几何方式
58 //本例求多边形核
59 inline pv ins(const pv &a, const pv &b)
60 {
61     u=fabs(ln.cross(a-pnt[i]));
62     v=fabs(ln.cross(b-pnt[i]))+u;
63     tl=b-a;
64     return pv(u*tl.x/v+a.x, u*tl.y/v+a.y);
65 }
66
67 int main()

```

```

68 {
69     j=0;
70     for (i=0;i<n;++i)
71     {
72         ln=pnt[(i+1)%n]-pnt[i];
73         p[j].resize(0);
74         for(k=0;k<p[j].size();++k)
75             if(ln.cross(p[j][k]-pnt[i])<=0)
76                 p[j].push_back(p[j][k]);
77         else
78         {
79             l=(k-1+p[j].size())%p[j].size();
80             if(ln.cross(p[j][l]-pnt[i])<0)
81                 p[j].push_back(ins(p[j][k],p[j][l]));
82             l=(k+1)%p[j].size();
83             if(ln.cross(p[j][l]-pnt[i])<0)
84                 p[j].push_back(ins(p[j][k],p[j][l]));
85         }
86         j=!j;
87     }
88     //结果在p[j]中]
89 }
90
91 //mrzy
92
93 bool HPIcmp(Line a, Line b)
94 {
95     if (fabs(a.k - b.k) > eps)
96         return a.k < b.k;
97     return ((a.s - b.s) * (b.e-b.s)) < 0;
98 }
99
100 Line Q[100];
101
102 void HPI(Line line[], int n, Point res[], int & resn)
103 {
104     int tot = n;
105     std::sort(line, line + n, HPIcmp);
106     tot = 1;
107     for (int i = 1; i < n; i++)
108         if (fabs(line[i].k - line[i - 1].k) > eps)
109             line[tot++] = line[i];
110     int head = 0, tail = 1;
111     Q[0] = line[0];
112     Q[1] = line[1];
113     resn = 0;
114     for (int i = 2; i < tot; i++)
115     {
116         if (fabs((Q[tail].e-Q[tail].s)*(Q[tail-1].e-Q[tail-1].s)) < eps || fabs((Q[head].e-Q[head].s)*(Q[head+1].e-Q[head+1].s)) < eps)
117             return;
118         while (head < tail && (((Q[tail]&Q[tail-1]) - line[i].s) * (line[i].e-line[i].s)) > eps)
119             --tail;
120         while (head < tail && (((Q[head]&Q[head+1]) - line[i].s) * (line[i].e-line[i].s)) > eps)
121             ++head;
122         Q[++tail] = line[i];
123     }
124     while (head < tail && (((Q[tail]&Q[tail-1]) - Q[head].s) * (Q[head].e-Q[head].s)) > eps)
125         tail--;
126     while (head < tail && (((Q[head]&Q[head+1]) - Q[tail].s) * (Q[tail].e-Q[tail].s)) > eps)
127         head++;
128     if (tail <= head + 1)
129         return;
130     for (int i = head; i < tail; i++)
131         res[resn++] = Q[i] & Q[i + 1];
132     if (head < tail + 1)
133         res[resn++] = Q[head] & Q[tail];
134 }
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```



```

52     res += 0.5*r*r*theta;
53 }
54 }
55 return res;
56 }
57 //调用
58
59
60 area2 = 0.0;
61 for (int i = 0; i < resn; i++) //遍历每条边, 按照逆时针
62     area2 += CalcArea(p[i], p[(i+1)%resn], r);

```

2.10 k-d tree

```

1  /*
2  有个很关键的剪枝, 在计算完与 mid 点的距离后, 我们应该先进入
   左右哪个子树? 我们应该先进入对于当前维度, 查询点位于的
   那一边。显然, 在查询点所在的子树, 更容易查找出正确解。
3
4  那么当进入完左或右子树后, 以查询点为圆心做圆, 如果当前维度
   查询点距离 mid 的距离 (另一个子树中的点距离查询点的距
   离肯定大于这个距离) 比堆里的最大值还大, 那么就不再递归
   另一个子树。注意一下: 如果堆里的元素个数不足 M, 仍然还
   要进入另一棵子树。
5
6  说白了就是随便乱搞啦.....
7  */
8  // hysbz 2626
9  #include<cstdio>
10 #include<algorithm>
11 #include<queue>
12
13 inline long long sqr(long long a){ return a*a;}
14 typedef std::pair<long long, int> pli;
15
16 #define MAXX 100111
17 #define MAX (MAXX<<2)
18 #define inf 0x3f3f3f3fll
19 int idx;
20
21 struct PNT
22 {
23     long long x[2];
24     int lb;
25     bool operator<(const PNT &i) const
26     {
27         return x[idx]<i.x[idx];
28     }
29     pli dist(const PNT &i) const
30     {
31         return pli(-(sqr(x[0]-i.x[0])+sqr(x[1]-
32             x[1])), lb);
33     }
34 } a[MAXX], the[MAX], p;
35
36 #define mid (l+r>>1)
37 #define lson (id<<1)
38 #define rson (id<<1|1)
39 #define lc lson, l, mid-1
40 #define rc rson, mid+1, r
41 int n, m;
42
43 long long rg[MAX][2][2];
44
45 void make(int id=1, int l=1, int r=n, int d=0)
46 {
47     the[id].lb=-1;
48     rg[id][0][0]=rg[id][1][0]=inf;
49     rg[id][0][1]=rg[id][1][1]=-inf;
50     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(
60         rg[lson][0][0], rg[rson][0][0]));
61     rg[id][1][0]=std::min(rg[id][1][0], std::min(
62         rg[lson][1][0], rg[rson][1][0]));
63
64     rg[id][0][1]=std::max(rg[id][0][1], std::max(
65         rg[lson][0][1], rg[rson][0][1]));
66     rg[id][1][1]=std::max(rg[id][1][1], std::max(
67         rg[lson][1][1], rg[rson][1][1]));
68 }
69
70 inline long long cal(int id)
71 {
72     static long long a[2];
73     static int i;
74     for(i=0; i<2; ++i)
75         a[i]=std::max(abs(p.x[i]-rg[id][i][0]),
76             abs(p.x[i]-rg[id][i][1]));
77     return sqr(a[0])+sqr(a[1]);
78 }
79
80 std::priority_queue<pli> ans;
81
82 void query(const int id=1, const int d=0)
83 {
84     if(the[id].lb<0)
85         return;
86     pli tmp(the[id].dist(p));
87     int a(lson), b(rson);
88     if(p.x[d]<=the[id].x[d])
89         std::swap(a, b);
90     if(ans.size()<m)
91         ans.push(tmp);
92     else
93         if(tmp<ans.top())
94         {
95             ans.push(tmp);
96             ans.pop();
97         }
98     if(ans.size()<m || cal(a)>=ans.top().first)
99         query(a, d^1);
100     if(ans.size()<m || cal(b)>=ans.top().first)
101         query(b, d^1);
102 }
103
104 int q, i, j, k;
105
106 int main()
107 {
108     scanf("%d", &n);
109     for(i=1; i<=n; ++i)
110     {
111         scanf("%lld%lld", &a[i].x[0], &a[i].x[1]);
112         a[i].lb=i;
113     }
114     make();
115     scanf("%d", &q);
116     while(q-->0)
117     {
118         scanf("%lld%lld", &p.x[0], &p.x[1]);
119         scanf("%d", &m);

```

```

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

2.11 Manhattan MST

1 #include<iostream>
2 #include<cstdio>
3 #include<cstring>
4 #include<queue>
5 #include<cmath>
6 using namespace std;
7 const int srange = 1000000; //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; //点个数
16
17 int swap( int &a, int &b ) //交换两个数
18 {
19     int t = a; a = b; b = t;
20 }
21
22 int insert( int a, int b, int i ) //向线段树中插入
    一个数
23 {
24     a += ra;
25     while ( a != 0 )
26     {
27         if ( c[ a ] > b )
28         {
29             c[ a ] = b;
30             d[ a ] = i;
31         }
32         else break;
33         a >>= 1;
34     }
35 }
36
37 int find( int a ) //从c[0..a]中找最小的数, 线段
    树查询/
38 {
39     a += ra;
40     int ret = d[ a ], max = c[ a ];
41     while ( a > 1 )
42     {
43         if ( ( a & 1 ) == 1 )
44             if ( c[ —a ] < max )
45             {
46                 max = c[ a ];
47                 ret = d[ a ];
48             }
49         a >>= 1;
50     }
51     return ret;
52 }
53
54 int ta[ 65536 ], tb[ 100000 ]; //基数排序临时变
    量
55
56 int radixsort( int *p ) //基数排序, 以为基准p
57 {
58     memset( ta, 0, sizeof( ta ) );
59     for ( int i = 0; i < n; i++ ) ta[ p[ i ] & 0
        xffff ]++;
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 ] = 0
        x7fffffff; //线段树初始
            化
92     memset( d, 0xff, sizeof( d ) );
93     for ( int i = 0; i < n; i++ ) //线段树插入删除调
        用
94     {
95         int tt = torder[ i ];
96         road[ tt ][ ii ] = find( Index[ tt ] );
97         insert( Index[ tt ], y[ tt ] + x[ tt ],
            tt );
98     }
99 }
100
101 int distanc( int a, int b ) //求两点的距离,
    之所以少一个是因为编译器不让使用作为函数名 edistance
102 {
103     return abs( x[ a ] - x[ b ] ) + abs( y[ a ] -
        y[ b ] );
104 }
105
106 int ttb[ 400000 ]; //边排序的临时变量
107 int rx[ 400000 ], ry[ 400000 ], rd[ 400000 ]; //
    边的存
        储
108 int rr = 0;
109
110 int radixsort_2( int *p ) //还是基数排序,

```

	<i>copy+的产物paste</i>	162		
111	{			<code>int &rkx = rank[x], &rky = rank[y];</code>
112	<code>memset(ta, 0, sizeof(ta));</code>	163		<code>if (rkx > rky) father[y] = x;</code>
113	<code>for (int i = 0; i < rr; i++) ta[p[i] & 0xffff]++;</code>	164		<code>else</code>
114	<code>for (int i = 0; i < 65535; i++) ta[i + 1] += ta[i];</code>	165		<code>{</code>
115	<code>for (int i = rr - 1; i >= 0; i--) ttb[-ttb[p[order[i]] & 0xffff]] = order[i];</code>	166		<code>father[x] = y;</code>
		167		<code>if (rkx == rky) rky++;</code>
		168		<code>}</code>
		169		<code>}</code>
		170		<code>}</code>
116	<code>memmove(order, ttb, rr * sizeof(int));</code>	171		<code>return ans;</code>
117	<code>memset(ta, 0, sizeof(ta));</code>	172		<code>}</code>
118	<code>for (int i = 0; i < rr; i++) ta[p[i] >> 16]++;</code>	173		<code>int casenum = 0;</code>
119	<code>for (int i = 0; i < 65535; i++) ta[i + 1] += ta[i];</code>	174		
120	<code>for (int i = rr - 1; i >= 0; i--) ttb[-ttb[p[order[i]] >> 16]] = order[i];</code>	175		<code>int main()</code>
121	<code>memmove(order, ttb, rr * sizeof(int));</code>	176		<code>{</code>
122	}	177		<code>while (cin >> n)</code>
123		178		<code>{</code>
124	<code>int father[100000], rank[100000];</code>	179		<code>if (n == 0) break;</code>
125	<code>int findfather(int x)</code>	180		<code>for (int i = 0; i < n; i++)</code>
	<i>寻找代表元</i>	181		<code>scanf("%d%d", &x[i], &y[i]);</code>
126	{	182		<code>memset(road, 0xff, sizeof(road));</code>
127	<code>if (father[x] != -1)</code>	183		<code>for (int i = 0; i < 4; i++)</code>
128	<code>return (father[x] = findfather(father[x]));</code>	184		<code>//为了减少编程复杂度, work()函数只写了一种,</code>
129	<code>else return x;</code>	185		<code>其他情况用转换坐标的方式类似处理</code>
130	}	186		<code>{</code>
131		187		<code>//为了降低算法复杂度, 只求出个方向的边</code>
132	<code>long long kruskal()</code>	188		<code>if (i == 2)</code>
	<i>成树</i>	189		<code>{</code>
133	{	190		<code>for (int j = 0; j < n; j++) swap</code>
134	<code>rr = 0;</code>	191		<code>(x[j], y[j]);</code>
135	<code>int tot = 0;</code>	192		<code>}</code>
136	<code>long long ans = 0;</code>			<code>if ((i & 1) == 1)</code>
137	<code>for (int i = 0; i < n; i++)</code>			<code>{</code>
	<i>表</i>			<code>for (int j = 0; j < n; j++) x[j] = srangle - x[j];</code>
138	{			<code>}</code>
139	<code>for (int j = 0; j < 4; j++)</code>			<code>work(i);</code>
140	{			<code>printf("Case%d: Total Weight=", ++casenum);</code>
141	<code>if (road[i][j] != -1)</code>			<code>cout << kruskal() << endl;</code>
142	{			<code>}</code>
143	<code>rx[rr] = i;</code>			<code>return 0;</code>
144	<code>ry[rr] = road[i][j];</code>			<code>}</code>
145	<code>rd[rr++] = distanc(i, road[i][j]);</code>			
146	}			
147	}			
148	}			
149	<code>for (int i = 0; i < rr; i++) order[i] = i;</code>			
	<i>//排序</i>			
150	<code>radixsort_2(rd);</code>			
151	<code>memset(father, 0xff, sizeof(father));</code>			
	<i>并查集初始化</i>			
152	<code>memset(rank, 0, sizeof(rank));</code>			
153	<code>for (int i = 0; i < rr; i++)</code>			
	<i>标准算法kruskal</i>			
154	{			
155	<code>if (tot == n - 1) break;</code>			
156	<code>int t = order[i];</code>			
157	<code>int x = findfather(rx[t]), y = findfather(ry[t]);</code>			
158	<code>if (x != y)</code>			
159	{			
160	<code>ans += rd[t];</code>			
161	<code>tot++;</code>			

2.12 others

eps

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

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

如果 a 为正, 则输出 $a + \text{eps}$, 否则输出 $a - \text{eps}$ 。

不要输出 -0.000

注意 double 的数据范围

$a=b \quad \text{fabs}(a-b)<\text{eps}$

```

13 a!=b    fabs(a-b)>eps
14 a<b    a+eps<b
15 a<=b   a<b+eps
16 a>b    a>b+eps
17 a>=b   a+eps>b
18
19 三角函数
20
21 cos/sin/tan 输入弧度
22 acos 输入 [-1,+1], 输出 [0,π]
23 asin 输入 [-1,+1], 输出  $[-\frac{\pi}{2}, +\frac{\pi}{2}]$ 
24 atan 输出  $[-\frac{\pi}{2}, +\frac{\pi}{2}]$ 
25 atan2 输入 (y,x)(注意顺序), 返回  $\tan(\frac{y}{x}) \in [-\pi, +\pi]$ 。xy 都是零
    的时候会发生除零错误
26
27 other
28
29 log 自然对数(ln)
30 log10 你猜.....
31 ceil 向上
32 floor 向下
33
34 round
35
36 cpp: 四舍六入五留双
37 java: add 0.5, then floor
38 cpp:
39 (一) 当尾数小于或等于 4 时, 直接将尾数舍去。
40 (二) 当尾数大于或等于 6 时, 将尾数舍去并向前一位进位。
41 (三) 当尾数为 5, 而尾数后面的数字均为 0 时, 应看尾数 “5” 的
    前一位: 若前一位数字此时为奇数, 就应向前进一位; 若前
    一位数字此时为偶数, 则应将尾数舍去。数字 “0” 在此时应被
    视为偶数。
42 (四) 当尾数为 5, 而尾数 “5” 的后面还有任何不是 0 的数字时,
    无论前一位在此时为奇数还是偶数, 也无论 “5” 后面不为 0
    的数字在哪一位上, 都应向前进一位。
43
44 rotate mat:
45  $\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$ 

```

2.13 Pick's theorem

```

1 给定顶点坐标均是整点（或正方形格点）的简单多边形
2
3 A: 面积
4 i: 内部格点数目
5 b: 边上格点数目
6  $A = i + \frac{b}{2} - 1$  取格点的组成图形的面积为一单位。在平行四边形格
    点, 皮克定理依然成立。套用于任意三角形格点, 皮克定理则
    是
7
8
9  $A = 2 \times i + b - 2$ 

```

2.14 PointInPoly

```

1 /*射线法
2 , 多边形可以是凸的或凹的的顶点数目要大于等于
3 poly3返回值为:
4
5 0 — 点在内poly
6 1 — 点在边界上poly
7 2 — 点在外poly
8 */
9
10 int inPoly(pv p, pv poly[], int n)
11 {
12     int i, count;
13     Line ray, side;

```

```

count = 0;
ray.s = p;
ray.e.y = p.y;
ray.e.x = -1; //-, 注意取值防止越界! INF

for (i = 0; i < n; i++)
{
    side.s = poly[i];
    side.e = poly[(i+1)%n];

    if (OnSeg(p, side))
        return 1;

    // 如果平行轴则不考虑 side x
    if (side.s.y == side.e.y)
        continue;

    if (OnSeg(side.s, ray))
    {
        if (side.s.y > side.e.y)
            count++;
    }
    else
        if (OnSeg(side.e, ray))
        {
            if (side.e.y > side.s.y)
                count++;
        }
        else
            if (inter(ray, side))
                count++;
    }
    return ((count % 2 == 1) ? 0 : 2);
}

```

2.15 rotating caliper

```

1 //最远点对
2
3 inline double go()
4 {
5     l=ans=0;
6     for (i=0; i<n; ++i)
7     {
8         tl=pnt[(i+1)%n]-pnt[i];
9         while(abs(tl.cross(pnt[(l+1)%n]-pnt[l]))
            >=abs(tl.cross(pnt[l]-pnt[i])))
            l=(l+1)%n;
            ans=std::max(ans, std::max(dist(pnt[l], pnt
                [i]), dist(pnt[l], pnt[(i+1)%n])));
        }
        return ans;
    }
}

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     tp=sp;
24     tq=sq;
25     ans=(ch[0][sp]-ch[1][sq]).len();
26     do
27     {
28         a1=ch[0][sp];
29         a2=ch[0][(sp+1)%ch[0].size()];
30         b1=ch[1][sq];

```

31	b2=ch[1][(sq+1)%ch[1].size()];	99	{
32	tpv=b1-(b2-a1);	100	d = c;
33	tpv.x = b1.x - (b2.x - a1.x);	101	sd = sc;
34	tpv.y = b1.y - (b2.y - a1.y);	102	}
35	len=(tpv-a1).cross(a2-a1);	103	while (xmult(vd,Point(p[d],p[(d+1)%n])) <
36	if(fabs(len)<eps)	104	0)
37	{	105	{
38	ans=std::min(ans,p2l(a1,b1,b2));	106	d = (d+1)%n;
39	ans=std::min(ans,p2l(a2,b1,b2));	107	sd++;
40	ans=std::min(ans,p2l(b1,a1,a2));	108	}
41	ans=std::min(ans,p2l(b2,a1,a2));	109	//卡在 p[a],p[b],p[c],p[d] 上
42	sp=(sp+1)%ch[0].size();	110	sa++;
43	sq=(sq+1)%ch[1].size();	111	}
44	}	112	}
45	else	113	
46	if(len<=eps)	114	//合并凸包给定凸多边形
47	{	115	P = { p(1) , ... , p(m) } 和 Q = { q(1) , ... ,
48	ans=std::min(ans,p2l(b1,a1,a2));	116	q(n) , 一个点对 (p(i) , q(j)) 形成 P 和 Q 之间的
49	sp=(sp+1)%ch[0].size();	117	桥当且仅当:
50	}	118	(p(i) , q(j)) 形成一个并踵点对。
51	else	119	p(i-1) , p(i+1) , q(j-1) , q(j+1) 都位于
52	{	120	由 (p(i) , q(j)) 组成的线的同一侧。假设多边形以标准
53	ans=std::min(ans,p2l(a1,b1,b2));	121	形式给出并且顶点是以顺时针序排列, 算法如下: 、 分别计算
54	sq=(sq+1)%ch[1].size();	122	
55	}	123	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,
56	}while(tp!=sp tq!=sq);	124	取 x 坐标最大的。、构造这些点的逐平切线,
57	return ans;	125	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)。、
58	}	126	同时顺时针旋转两条切线直到其中一条与边相交。
59		127	3 得到一个新的并踵点对 (p(i) , q(j)) 。对于平行边的情况,
60	//外接矩形 by mzry	128	得到三个并踵点对。、对于所有有效的并踵点对
61	inline void solve()	129	4 (p(i) , q(j)): 判
62	{	130	定 p(i-1) , p(i+1) , q(j-1) , q(j+1) 是否都位于连接
63	resa = resb = 1e100;	131	点 (p(i) , q(j)) 形成的线的同一侧。如果是, 这个并踵
64	double dis1,dis2;	132	点对就形成了一个桥, 并标记他。、重复执行步骤和步骤直到
65	Point xp[4];	133	切线回到他们原来的位置。
66	Line l[4];	134	534、所有可能的桥此时都已经确定了。
67	int a,b,c,d;	135	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
68	int sa,sb,sc,sd;	136	定了算法的正确性。运行时间受步骤, , 约束。
69	a = b = c = d = 0;	137	
70	sa = sb = sc = sd = 0;	138	156 他们都为 O(N) 运行时间 (N 是顶点总数)。因此算法拥有
71	Point va,vb,vc,vd;	139	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
72	for (a = 0; a < n; a++)	140	个有用的概念: 多边形间公切线。同时, 桥也是计算凸多边
73	{	141	形交的算法核心。
74	va = Point(p[a],p[(a+1)%n]);		
75	vc = Point(-va.x,-va.y);		
76	vb = Point(-va.y,va.x);		
77	vd = Point(-vb.x,-vb.y);		
78	if (sb < sa)		
79	{		
80	b = a;		
81	sb = sa;		
82	}		
83	while (xmult(vb,Point(p[b],p[(b+1)%n])) <		
84	0)		
85	{		
86	b = (b+1)%n;		
87	sb++;		
88	}		
89	if (sc < sb)		
90	{		
91	c = b;		
92	sc = sb;		
93	}		
94	while (xmult(vc,Point(p[c],p[(c+1)%n])) <		
95	0)		
96	{		
97	c = (c+1)%n;		
98	sc++;		
	}		
	if (sd < sc)		

```

142 //最小最大周长面积外接矩形//、计算全部四个多边形的端点，
143 1 称之为， xminP , xmaxP , yminP 。 ymaxP、通过四个点构造
144 2 P 的四条切线。他们确定了两个“卡壳”集合。、如果一条（或
    两条）线与一条边重合，
145 3 那么计算由四条线决定的矩形的面积，并且保存为当前最小值
    否则将当前最小值定义为无穷大。、顺时针旋转线直到其中
    一条和多边形的一条边重合。
146 4、计算新矩形的周长面积，
147 5/ 并且和当前最小值比较。如果小于当前最小值则更新，并保存
    确定最小值的矩形信息。、重复步骤和步骤，
148 645 直到线旋转过的角度大于度。90、输出外接矩形的最小周长。
149 7
2.16 shit
1 struct pv
2 {
3     double x,y;
4     pv():x(0),y(0){}
5     pv(double xx,double yy):x(xx),y(yy){}
6     inline pv operator+(const pv &i) const
7     {
8         return pv(x+i.x,y+i.y);
9     }
10    inline pv operator-(const pv &i) const
11    {
12        return pv(x-i.x,y-i.y);
13    }
14    inline bool operator ==(const pv &i) const
15    {
16        return fabs(x-i.x)<eps && fabs(y-i.y)<eps;
17    }
18    inline bool operator <(const pv &i) const
19    {
20        return y==i.y?x<i.x:y<i.y;
21    }
22    inline double cross(const pv &i) const
23    {
24        return x*i.y-y*i.x;
25    }
26    inline double dot(const pv &i) const
27    {
28        return x*i.x+y*i.y;
29    }
30    inline double len()
31    {
32        return sqrt(x*x+y*y);
33    }
34    inline pv rotate(pv p,double theta)
35    {
36        static pv v;
37        v=*this-p;
38        static double c,s;
39        c=cos(theta);
40        s=sin(theta);
41        return pv(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
42    }
43 };
44
45 inline int dblcmp(double d)
46 {
47     if(fabs(d)<eps)
48         return 0;
49     return d>eps?1:-1;
50 }
51
52 inline int cross(pv *a,pv *b) // 不相交0 不规范1 规
范2
{
    int d1=dblcmp((a[1]-a[0]).cross(b[0]-a[0]));
    int d2=dblcmp((a[1]-a[0]).cross(b[1]-a[0]));
    int d3=dblcmp((b[1]-b[0]).cross(a[0]-b[0]));
    int d4=dblcmp((b[1]-b[0]).cross(a[1]-b[0]));
    if((d1^d2)==-2 && (d3^d4)==-2)
        return 2;
    return ((d1==0 && dblcmp((b[0]-a[0]).dot(b[0]-a[1]))<=0) ||
            (d2==0 && dblcmp((b[1]-a[0]).dot(b[1]-a[1]))<=0) ||
            (d3==0 && dblcmp((a[0]-b[0]).dot(a[0]-b[1]))<=0) ||
            (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 //preciseness should not be too
        high ( compare with eps )
89         if(fabs(b)>eps)
90         {
91             pnt[0]=pv(maxl,(c+a*maxl)/(-b));
92             pnt[1]=pv(-maxl,(c-a*maxl)/(-b));
93         }
94         else
95         {
96             pnt[0]=pv(-c/a,maxl);
97             pnt[1]=pv(-c/a,-maxl);
98         }
99 #undef maxl
100     }
101     pv cross(const line &v) const
102     {
103         double a=(v.pnt[1]-v.pnt[0]).cross(pnt[0]-v.pnt[0]);
104         double b=(v.pnt[1]-v.pnt[0]).cross(pnt[1]-v.pnt[0]);
105         return pv((pnt[0].x*b-pnt[1].x*a)/(b-a),(pnt[0].y*b-pnt[1].y*a)/(b-a));
106     }
107 };
108
109 inline std::pair<pv,double> getcircle(const pv &a,
    const pv &b,const pv &c)

```



```

44 double area() 105
45 { 106
46     return pi*sqr(r); 107
47 } 108
48 double circumference() 109
49 { 110
50     return 2*pi*r; 111
51 }
52 //0 圆外 112
53 //1 圆上 113
54 //2 圆内 114
55 int relation(point b)
56 {
57     double dst=b.distance(p); 115
58     if (dblcmp(dst-r)<0)return 2;
59     if (dblcmp(dst-r)==0)return 1;
60     return 0; 116
61 }
62 int relationseg(line v) 117
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; 118
68 } 119
69 int relationline(line v) 120
70 { 121
71     double dst=v.dispointtoline(p); 122
72     if (dblcmp(dst-r)<0)return 2; 123
73     if (dblcmp(dst-r)==0)return 1; 124
74     return 0; 125
75 } 126
76 //过a 两点b 半径的两个圆r 127
77 int getcircle(point a,point b,double r,circle 128
    &c1,circle&c2) 129
78 { 130
79     circle x(a,r),y(b,r); 131
80     int t=x.pointcrosscircle(y,c1.p,c2.p); 132
81     if (!t)return 0; 133
82     c1.r=c2.r=r; 134
83     return t;
84 }
85 //与直线相切u 过点q 半径的圆r1 135
86 int getcircle(line u,point q,double r1,circle 136
    &c1,circle &c2) 137
87 { 138
88     double dis=u.dispointtoline(q); 139
89     if (dblcmp(dis-r1*2)>0)return 0; 140
90     if (dblcmp(dis)==0) 141
91     { 142
92         c1.p=q.add(u.b.sub(u.a).rotleft().trunc( 143
            r1)); 144
93         c2.p=q.add(u.b.sub(u.a).rotright().trunc( 145
            r1)); 146
94         c1.r=c2.r=r1; 147
95         return 2; 148
96     } 149
97     line u1=line(u.a.add(u.b.sub(u.a).rotleft( 150
        .trunc(r1)),u.b.add(u.b.sub(u.a).
        rotleft().trunc(r1))); 151
98     line u2=line(u.a.add(u.b.sub(u.a).rotright 152
        (.trunc(r1)),u.b.add(u.b.sub(u.a).
        rotright().trunc(r1))); 153
99     circle cc=circle(q,r1); 154
100     point p1,p2; 155
101     if (!cc.pointcrossline(u1,p1,p2))cc. 156
        pointcrossline(u2,p1,p2); 157
102     c1=circle(p1,r1); 158
103     if (p1==p2) 159
104     { 160
        c2=c1;return 1; 161
    }
    }
    //同时与直线u,相切v 半径的圆r1
    int getcircle(line u,line v,double r1,circle
        &c1,circle &c2,circle &c3,circle &c4)
    {
        if (u.parallel(v))return 0;
        line u1=line(u.a.add(u.b.sub(u.a).rotleft()
            .trunc(r1)),u.b.add(u.b.sub(u.a).
            rotleft().trunc(r1)));
        line u2=line(u.a.add(u.b.sub(u.a).rotright
            ().trunc(r1)),u.b.add(u.b.sub(u.a).
            rotright().trunc(r1)));
        line v1=line(v.a.add(v.b.sub(v.a).rotleft()
            .trunc(r1)),v.b.add(v.b.sub(v.a).
            rotleft().trunc(r1)));
        line v2=line(v.a.add(v.b.sub(v.a).rotright
            ().trunc(r1)),v.b.add(v.b.sub(v.a).
            rotright().trunc(r1)));
        c1.r=c2.r=c3.r=c4.r=r1;
        c1.p=u1.crosspoint(v1);
        c2.p=u1.crosspoint(v2);
        c3.p=u2.crosspoint(v1);
        c4.p=u2.crosspoint(v2);
        return 4;
    }
    //同时与不相交圆cx,相切cy 半径为的圆r1
    int getcircle(circle cx,circle cy,double r1,
        circle&c1,circle&c2)
    {
        circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
        int t=x.pointcrosscircle(y,c1.p,c2.p);
        if (!t)return 0;
        c1.r=c2.r=r1;
        return t;
    }
    int pointcrossline(line v,point &p1,point &p2
        )//求与线段交要先判
        断relationseg
    {
        if (!(*this).relationline(v))return 0;
        point a=v.lineprog(p);
        double d=v.dispointtoline(p);
        d=sqrt(r*r-d*d);
        if (dblcmp(d)==0)
        {
            p1=a;
            p2=a;
            return 1;
        }
        p1=a.sub(v.b.sub(v.a).trunc(d));
        p2=a.add(v.b.sub(v.a).trunc(d));
        return 2;
    }
    }
    //5 相离
    //4 外切
    //3 相交
    //2 内切
    //1 内含
    int relationcircle(circle v)
    {
        double d=p.distance(v.p);
        if (dblcmp(d-r-v.r)>0)return 5;
        if (dblcmp(d-r-v.r)==0)return 4;
        double l=fabs(r-v.r);
        if (dblcmp(d-r-v.r)<0&&dblcmp(d-l)>0)return
            3;
    }

```



```

162     if (dblcmp(d-1)==0)return 2;
163     if (dblcmp(d-1)<0)return 1;
164 }
165 int pointcrosscircle(circle v,point &p1,point &p2)
166 {
167     int rel=relationcircle(v);
168     if (rel==1||rel==5)return 0;
169     double d=p.distance(v.p);
170     double l=(d+(sqr(r)-sqr(v.r))/d)/2;
171     double h=sqrt(sqr(r)-sqr(l));
172     p1=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotleft().trunc(h)));
173     p2=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotright().trunc(h)));
174     if (rel==2||rel==4)
175     {
176         return 1;
177     }
178     return 2;
179 }
180 //过一点做圆的切线 先判断点和圆关系()
181 int tangentline(point q,line &u,line &v)
182 {
183     int x=relation(q);
184     if (x==2)return 0;
185     if (x==1)
186     {
187         u=line(q,q.add(q.sub(p).rotleft()));
188         v=u;
189         return 1;
190     }
191     double d=p.distance(q);
192     double l=sqr(r)/d;
193     double h=sqrt(sqr(r)-sqr(l));
194     u=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotleft().trunc(h))));
195     v=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotright().trunc(h))));
196     return 2;
197 }
198 double areacircle(circle v)
199 {
200     int rel=relationcircle(v);
201     if (rel>=4)return 0.0;
202     if (rel<=2)return min(area(),v.area());
203     double d=p.distance(v.p);
204     double hf=(r+v.r+d)/2.0;
205     double ss=2*sqrt(hf*(hf-r)*(hf-v.r)*(hf-d));
206     double a1=acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
207     a1=a1*r*r;
208     double a2=acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
209     a2=a2*v.r*v.r;
210     return a1+a2-ss;
211 }
212 double areatriangle(point a,point b)
213 {
214     if (dblcmp(p.sub(a).det(p.sub(b))==0))
215         return 0.0;
216     point q[5];
217     int len=0;
218     q[len++]=a;
219     line l(a,b);
220     point p1,p2;
221     if (pointcrossline(l,q[1],q[2])==2)
222     {
223         if (dblcmp(a.sub(q[1]).dot(b.sub(q
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

42     }
43     if (j<n) mark[i]=1;
44 }
45 for (i=0;i<n;i++) if (!mark[i]) c[k++]=c[i];
46 n=k;
47 }
48 double areaarc(double th,double r)
49 {
50     return 0.5*sqr(r)*(th-sin(th));
51 }
52 void getarea()
53 {
54     int i,j,k;
55     memset(ans,0,sizeof(ans));
56     vector<pair<double,int>> v;
57     for (i=0;i<n;i++)
58     {
59         v.clear();
60         v.push_back(make_pair(-pi,1));
61         v.push_back(make_pair(pi,-1));
62         for (j=0;j<n;j++) if (i!=j)
63         {
64             point q=c[j].p.sub(c[i].p);
65             double ab=q.len(),ac=c[i].r,bc=c[j].r;
66             if (dblcmp(ab+ac-bc)<=0)
67             {
68                 v.push_back(make_pair(-pi,1));
69                 v.push_back(make_pair(pi,-1));
70                 continue;
71             }
72             if (dblcmp(ab+bc-ac)<=0) continue;
73             if (dblcmp(ab-ac-bc)>0) continue;
74             double th=atan2(q.y,q.x),fai=acos((ac*ac+
75                 ab*ab-bc*bc)/(2.0*ac*ab));
76             double a0=th-fai;
77             if (dblcmp(a0+pi)<0) a0+=2*pi;
78             double a1=th+fai;
79             if (dblcmp(a1-pi)>0) a1-=2*pi;
80             if (dblcmp(a0-a1)>0)
81             {
82                 v.push_back(make_pair(a0,1));
83                 v.push_back(make_pair(pi,-1));
84                 v.push_back(make_pair(-pi,1));
85                 v.push_back(make_pair(a1,-1));
86             }
87             else
88             {
89                 v.push_back(make_pair(a0,1));
90                 v.push_back(make_pair(a1,-1));
91             }
92         }
93     }
94     sort(v.begin(),v.end());
95     int cur=0;
96     for (j=0;j<v.size();j++)
97     {
98         if (cur&&dblcmp(v[j].first-pre[cur])>0)
99         {
100             ans[cur]+=areaarc(v[j].first-pre[cur],
101                 c[i].r);
102             ans[cur]+=0.5*point(c[i].p.x+c[i].r*cos(
103                 pre[cur]),c[i].p.y+c[i].r*sin(pre[
104                 cur])).det(point(c[i].p.x+c[i].r*
105                 cos(v[j].first),c[i].p.y+c[i].r*
106                 sin(v[j].first)));
107         }
108         cur+=v[j].second;
109         pre[cur]=v[j].first;
110     }
111     for (i=1;i<=n;i++)

```

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)>0) hp[m++] = hp[i];
42             else if (dblcmp(hp[m-1].b.sub(hp[m-1].a).
43                 det(hp[i].a.sub(hp[m-1].a))>0) hp[m-1] =
44                 hp[i];
45         }
46         n=m;
47     }
48     bool halfplaneinsert()
49     {
50         int i;
51         for (i=0;i<n;i++) hp[i].calcangle();
52         sort(hp, hp+n);
53         unique();
54         que[st=0]=0;
55         que[ed=1]=1;
56         p[1]=hp[0].crosspoint(hp[1]);
57         for (i=2;i<n;i++)
58         {
59             while (st<ed&&dblcmp((hp[i].b.sub(hp[i].a).
60                 det(p[ed].sub(hp[i].a)))<0) ed++;
61             while (st<ed&&dblcmp((hp[i].b.sub(hp[i].a).

```

```

    det(p[st+1].sub(hp[i].a)))<0)st++; 42
    que[++ed]=i; 43
    if (hp[i].parallel(hp[que[ed-1]]))return 44
        false; 45
    p[ed]=hp[i].crosspoint(hp[que[ed-1]]); 46
} 47
while (st<ed&&dblcmp(hp[que[st]].b.sub(hp[que[
[st]].a).det(p[ed].sub(hp[que[st]].a))) 48
<0)ed--; 49
while (st<ed&&dblcmp(hp[que[ed]].b.sub(hp[que[
[ed]].a).det(p[st+1].sub(hp[que[ed]].a))) 50
<0)st++; 51
if (st+1==ed)return false; 52
return true; 53
} 54
void getconvex(polygon &con) 55
{ 56
    p[st]=hp[que[st]].crosspoint(hp[que[ed]]); 57
    con.n=ed-st+1; 58
    int j=st, i=0; 59
    for (;j<=ed;i++,j++) 60
    { 61
        con.p[i]=p[j]; 62
    } 63
} 64
}; 65

```

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        {

```

```

        a=point(0,-_c/_b);
        b=point(1,(-_c-_a)/_b);
    }
}
void input()
{
    a.input();
    b.input();
}
void adjust()
{
    if (b<a)swap(a,b);
}
double length()
{
    return a.distance(b);
}
double angle()//直线倾斜角 0<=angle<180
{
    double k=atan2(b.y-a.y,b.x-a.x);
    if (dblcmp(k)<0)k+=pi;
    if (dblcmp(k-pi)==0)k-=pi;
    return k;
}
//点和线段关系
//1 在逆时针
//2 在顺时针
//3 平行
int relation(point p)
{
    int c=dblcmp(p.sub(a).det(b.sub(a)));
    if (c<0)return 1;
    if (c>0)return 2;
    return 3;
}
bool pointonseg(point p)
{
    return dblcmp(p.sub(a).det(b.sub(a)))
        ==0&&dblcmp(p.sub(a).dot(p.sub(b)))
        <=0;
}
bool parallel(line v)
{
    return dblcmp(b.sub(a).det(v.b.sub(v.a)))
        ==0;
}
//2 规范相交
//1 非规范相交
//0 不相交
int segcrossseg(line v)
{
    int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
    int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
    int d3=dblcmp(v.b.sub(v.a).det(a.sub(v.a)
    ));
    int d4=dblcmp(v.b.sub(v.a).det(b.sub(v.a)
    ));
    if ((d1^d2)==-2&&(d3^d4)==-2)return 2;
    return (d1==0&&dblcmp(v.a.sub(a).dot(v.a.
    sub(b)))<=0||
        d2==0&&dblcmp(v.b.sub(a).dot(v.b.
    sub(b)))<=0||
        d3==0&&dblcmp(a.sub(v.a).dot(a.
    sub(v.b)))<=0||
        d4==0&&dblcmp(b.sub(v.a).dot(b.
    sub(v.b)))<=0);
}
int linecrossseg(line v)//*this seg v line
{
    int d1=dblcmp(b.sub(a).det(v.a.sub(a)));

```

```

103     int d2=dblcmp(b.sub(a).det(v.b.sub(a))) 20
104     if ((d1^d2)==-2)return 2; 21
105     return (d1==0||d2==0); 22
106 } 23
107 //0 平行 24
108 //1 重合 25
109 //2 相交
110 int linecrossline(line v) 26
111 { 27
112     if ((*this).parallel(v)) 28
113     { 29
114         return v.relation(a)==3; 30
115     } 31
116     return 2; 32
117 } 33
118 point crosspoint(line v) 34
119 { 35
120     double a1=v.b.sub(v.a).det(a.sub(v.a)); 36
121     double a2=v.b.sub(v.a).det(b.sub(v.a)); 37
122     return point((a.x*a2-b.x*a1)/(a2-a1),(a.y 38
123                 *a2-b.y*a1)/(a2-a1)); 39
124 } 40
125 double dispointtoline(point p) 41
126 { 42
127     return fabs(p.sub(a).det(b.sub(a)))/ 43
128     length(); 44
129 } 45
130 double dispointtoseg(point p) 46
131 { 47
132     if (dblcmp(p.sub(b).dot(a.sub(b)))<0|| 48
133         dblcmp(p.sub(a).dot(b.sub(a)))<0) 49
134     { 50
135         return min(p.distance(a),p.distance(b 51
136                 )); 52
137     } 53
138     return dispointtoline(p); 54
139 } 55
140 point lineprog(point p) 56
141 { 57
142     return a.add(b.sub(a).mul(b.sub(a).dot(p 58
143                 sub(a))/b.sub(a).len2())); 59
144 } 60
145 point symmetrpoint(point p) 61
146 { 62
147     point q=lineprog(p); 63
148     return point(2*q.x-p.x,2*q.y-p.y); 64
149 } 65
}; 66

```

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()

```

```

{
    return a.distance(b);
}
bool pointonseg(point3 p)
{
    return dblcmp(p.sub(a).det(p.sub(b)).len())
        ==0&&dblcmp(a.sub(p).dot(b.sub(p)))<=0;
}
double dispointtoline(point3 p)
{
    return b.sub(a).det(p.sub(a)).len()/a.
        distance(b);
}
double dispointtoseg(point3 p)
{
    if (dblcmp(p.sub(b).dot(a.sub(b)))<0||
        dblcmp(p.sub(a).dot(b.sub(a)))<0)
    {
        return min(p.distance(a),p.distance(b
            ));
    }
    return dispointtoline(p);
}
point3 lineprog(point3 p)
{
    return a.add(b.sub(a).trunc(b.sub(a).dot(p.
        sub(a))/b.distance(a)));
}
point3 rotate(point3 p,double ang)//绕此向量逆
    时针角度parg
{
    if (dblcmp((p.sub(a).det(p.sub(b)).len())
        ==0)return p;
    point3 f1=b.sub(a).det(p.sub(a));
    point3 f2=b.sub(a).det(f1);
    double len=fabs(a.sub(p).det(b.sub(p)).len()/
        a.distance(b));
    f1=f1.trunc(len);f2=f2.trunc(len);
    point3 h=p.add(f2);
    point3 pp=h.add(f1);
    return h.add((p.sub(h)).mul(cos(ang*1.0)).
        add((pp.sub(h)).mul(sin(ang*1.0)));
}
};

```

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    {
15        //ax+by+cz+d=0
16        o=point3(_a,_b,_c);
17        if (dblcmp(_a)!=0)
18        {
19            a=point3((-_d-_c-_b)/_a,1,1);
20        }
21        else if (dblcmp(_b)!=0)
22        {
23            a=point3(1,(-_d-_c-_a)/_b,1);

```

```

23     }
24     else if (dblcmp(_c)!=0)
25     {
26         a=point3(1,1,(-_d-_a-_b)/_c);
27     }
28     }
29     void input()
30     {
31         a.input();
32         b.input();
33         c.input();
34         o=pvec();
35     }
36     point3 pvec()
37     {
38         return b.sub(a).det(c.sub(a));
39     }
40     bool pointonplane(point3 p)//点是否在平面上
41     {
42         return dblcmp(p.sub(a).dot(o))==0;
43     }
44     //0 不在
45     //1 在边界上
46     //2 在内部
47     int pointontriangle(point3 p)//点是否在空间三角
    形上abc
48     {
49         if (!pointonplane(p))return 0;
50         double s=a.sub(b).det(c.sub(b)).len();
51         double s1=p.sub(a).det(p.sub(b)).len();
52         double s2=p.sub(a).det(p.sub(c)).len();
53         double s3=p.sub(b).det(p.sub(c)).len();
54         if (dblcmp(s-s1-s2-s3))return 0;
55         if (dblcmp(s1)&&dblcmp(s2)&&dblcmp(s3))
            return 2;
56         return 1;
57     }
58     //判断两平面关系
59     //0 相交
60     //1 平行但不重合
61     //2 重合
62     bool relationplane(plane f)
63     {
64         if (dblcmp(o.det(f.o).len()))return 0;
65         if (pointonplane(f.a))return 2;
66         return 1;
67     }
68     double angleplane(plane f)//两平面夹角
69     {
70         return acos(o.dot(f.o)/(o.len()*f.o.len())
            );
71     }
72     double dispoint(point3 p)//点到平面距离
73     {
74         return fabs(p.sub(a).dot(o)/o.len());
75     }
76     point3 pttoplane(point3 p)//点到平面最近点
77     {
78         line3 u=line3(p,p.add(o));
79         crossline(u,p);
80         return p;
81     }
82     int crossline(line3 u,point3 &p)//平面和直线的
    交点
83     {
84         double x=o.dot(u.b.sub(a));
85         double y=o.dot(u.a.sub(a));
86         double d=x-y;
87         if (dblcmp(fabs(d))==0)return 0;
88         p=u.a.mul(x).sub(u.b.mul(y)).div(d);
89         return 1;
90     }
91     int crossplane(plane f,line3 &u)//平面和平面的
    交线
92     {
93         point3 oo=o.det(f.o);
94         point3 v=o.det(oo);
95         double d=fabs(f.o.dot(v));
96         if (dblcmp(d)==0)return 0;
97         point3 q=a.add(v.mul(f.o.dot(f.a.sub(a))/d)
            );
98         u=line3(q,q.add(oo));
99         return 1;
100    }
101 };

```

3.7 point

```

1 using namespace std;
2
3 #define mp make_pair
4 #define pb push_back
5
6 const double eps=1e-8;
7 const double pi=acos(-1.0);
8 const double inf=1e20;
9 const int maxp=8;
10
11 int dblcmp(double d)
12 {
13     if (fabs(d)<eps)return 0;
14     return d>eps?1:-1;
15 }
16
17 inline double sqr(double x)
18 {
19     return x*x;
20 }
21
22 struct point
23 {
24     double x,y;
25     point(){}
26     point(double _x,double _y):
27     x(_x),y(_y){};
28     void input()
29     {
30         scanf("%lf%lf",&x,&y);
31     }
32     void output()
33     {
34         printf("%.2f_%.2f\n",x,y);
35     }
36     bool operator==(point a)const
37     {
38         return dblcmp(a.x-x)==0&&dblcmp(a.y-y)
            ==0;
39     }
40     bool operator<(point a)const
41     {
42         return dblcmp(a.x-x)==0?dblcmp(y-a.y)<0:x
            <a.x;
43     }
44     double len()
45     {
46         return hypot(x,y);
47     }
48     double len2()
49     {
50         return x*x+y*y;

```

```

51     }
52     double distance(point p)
53     {
54         return hypot(x-p.x,y-p.y);
55     }
56     point add(point p)
57     {
58         return point(x+p.x,y+p.y);
59     }
60     point sub(point p)
61     {
62         return point(x-p.x,y-p.y);
63     }
64     point mul(double b)
65     {
66         return point(x*b,y*b);
67     }
68     point div(double b)
69     {
70         return point(x/b,y/b);
71     }
72     double dot(point p)
73     {
74         return x*p.x+y*p.y;
75     }
76     double det(point p)
77     {
78         return x*p.y-y*p.x;
79     }
80     double rad(point a,point b)
81     {
82         point p=*this;
83         return fabs(atan2(fabs(a.sub(p).det(b.sub(p))),a.sub(p).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 rotright()
93     {
94         return point(-y,x);
95     }
96     point rotleft()
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("%lf%lf%lf",&x,&y,&z);
10    }
11    void output()
12    {
13        printf("%.2lf_%.2lf_%.2lf\n",x,y,z);
14    }
15    bool operator==(point3 a)
16    {
17        return dblcmp(a.x-x)==0&&dblcmp(a.y-y)
18            ==0&&dblcmp(a.z-z)==0;
19    }
20    bool operator<(point3 a)const
21    {
22        return dblcmp(a.x-x)==0?dblcmp(y-a.y)==0?
23            dblcmp(z-a.z)<0:y<a.y:x<a.x;
24    }
25    double len()
26    {
27        return sqrt(len2());
28    }
29    double len2()
30    {
31        return x*x+y*y+z*z;
32    }
33    double distance(point3 p)
34    {
35        return sqrt((p.x-x)*(p.x-x)+(p.y-y)*(p.y-
36            y)+(p.z-z)*(p.z-z));
37    }
38    point3 add(point3 p)
39    {
40        return point3(x+p.x,y+p.y,z+p.z);
41    }
42    point3 sub(point3 p)
43    {
44        return point3(x-p.x,y-p.y,z-p.z);
45    }
46    point3 mul(double d)
47    {
48        return point3(x*d,y*d,z*d);
49    }
50    point3 div(double d)
51    {
52        return point3(x/d,y/d,z/d);
53    }
54    double dot(point3 p)
55    {
56        return x*p.x+y*p.y+z*p.z;
57    }
58    point3 det(point3 p)
59    {
60        return point3(y*p.z-p.y*z,p.x*z-x*p.z,x*p
61            .y-p.x*y);
62    }
63    double rad(point3 a,point3 b)
64    {
65        point3 p=*this;
66        return acos(a.sub(p).dot(b.sub(p))/(a.
67            distance(p)*b.distance(p)));
68    }
69    point3 trunc(double r)
70    {
71        r/=len();
72        return point3(x*r,y*r,z*r);
73    }
74    point3 rotate(point3 o,double r) // building?
75    {
76    }
77 };

```

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/
14        sqrt(2.0)));
15        p[3]=p[2].rotate(p[0],2*pi-pi/4);
16        p[3]=p[0].add((p[3].sub(p[0])).trunc(dis/
17        sqrt(2.0)));
18    }
19    void add(point q)
20    {
21        p[n++]=q;
22    }
23    void getline()
24    {
25        for (int i=0;i<n;i++)
26        {
27            l[i]=line(p[i],p[(i+1)%n]);
28        }
29    }
30    struct cmp
31    {
32        point p;
33        cmp(const point &p0){p=p0;}
34        bool operator()(const point &aa,const point
35        &bb)
36        {
37            point a=aa,b=bb;
38            int d=dblcmp(a.sub(p).det(b.sub(p)));
39            if (d==0)
40            {
41                return dblcmp(a.distance(p)-b.
42                distance(p))<0;
43            }
44            return d>0;
45        }
46    };
47    void norm()
48    {
49        point mi=p[0];
50        for (int i=1;i<n;i++)mi=min(mi,p[i]);
51        sort(p,p+n,cmp(mi));
52    }
53    void getconvex(polygon &convex)
54    {
55        int i,j,k;
56        sort(p,p+n);
57        convex.n=n;
58        for (i=0;i<min(n,2);i++)
59        {
60            convex.p[i]=p[i];
61        }
62        if (n<=2)return;
63        int &top=convex.n;
64        top=1;
65        for (i=2;i<n;i++)
66        {
67            while (top&&convex.p[top].sub(p[i]).
68            det(convex.p[top-1].sub(p[i]))
69            <=0)
70            {
71                top--;
72                convex.p[+top]=p[i];
73            }
74        }
75        bool isconvex()
76        {
77            bool s[3];
78            memset(s,0,sizeof(s));
79            int i,j,k;
80            for (i=0;i<n;i++)
81            {
82                j=(i+1)%n;
83                k=(j+1)%n;
84                s[dblcmp(p[j].sub(p[i]).det(p[k].sub(p[i]
85                )))>0]=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                )))>0;
112                int u=dblcmp(p[i].y-q.y);
113                int v=dblcmp(p[j].y-q.y);
114                if (k>0&&u<0&&v>=0)cnt++;
115                if (k<0&&v<0&&u>=0)cnt--;
116            }
117            return cnt!=0;
118        }
119        //1 在多边形内长度为正
120        //2 相交或与边平行
121        //0 无任何交点
122        int relationline(line u)
123        {
124            int i,j,k=0;
125            getline();
126            for (i=0;i<n;i++)
127            {
128                if (l[i].segcrossseg(u)==2)return 1;
129                if (l[i].segcrossseg(u)==1)k=1;
130            }
131        }
132    };
133 }

```

```

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.b27
166             sub(u.a)));
167         int d2=dblcmp(p[(i+1)%n].sub(u.a).det
168             (u.b.sub(u.a)));
169         if (d1>=0) po.p[top++]=p[i];
170         if (d1*d2<0) po.p[top++]=u.crosspoint
171             line(p[i], p[(i+1)%n]));
172     }
173 }
174 double getcircumference()
175 {
176     double sum=0;
177     int i;
178     for (i=0; i<n; i++)
179     {
180         sum+=p[i].distance(p[(i+1)%n]);
181     }
182     return sum;
183 }
184 double getarea()
185 {
186     double sum=0;
187     int i;
188     for (i=0; i<n; i++)
189     {
190         sum+=p[i].det(p[(i+1)%n]);
191     }
192     return fabs(sum)/2;
193 }
194 bool getdir() //代表逆时针 1 代表顺时针 0
195 {
196     double sum=0;
197     int i;
198     for (i=0; i<n; i++)
199     {
200         sum+=p[i].det(p[(i+1)%n]);
201     }
202     if (dblcmp(sum)>0) return 1;
203     return 0;
204 }
205 point getbarycentre() // centroid
206 {
207     point ret(0,0);
208     double area=0;
209     int i;
210     for (i=1; i<n-1; i++)
211     {
212         double tmp=p[i].sub(p[0]).det(p[i+1].
213             sub(p[0]));
214         if (dblcmp(tmp)==0) continue;
215         area+=tmp;
216         ret.x+=(p[0].x+p[i].x+p[i+1].x)/3*tmp;
217         ret.y+=(p[0].y+p[i].y+p[i+1].y)/3*tmp;
218     }
219     if (dblcmp(area)) ret=ret.div(area);
220     return ret;
221 }
222 double areaintersection(polygon po) // refer:
223     HPI
224 {
225 }
226 double areaunion(polygon po)
227 {
228     return getarea()+po.getarea()-
229         areaintersection(po);
230 }
231 double areacircle(circle c)
232 {
233     int i, j, k, l, m;
234     double ans=0;
235     for (i=0; i<n; i++)
236     {
237         int j=(i+1)%n;
238         if (dblcmp(p[j].sub(c.p).det(p[i].sub(c.p))
239             )>=0)
240         {
241             ans+=c.areastriangle(p[i], p[j]);
242         }
243         else
244         {
245             ans-=c.areastriangle(p[i], p[j]);
246         }
247     }
248     return fabs(ans);
249 }
250 //多边形和圆关系
251 //0 一部分在圆外
252 //1 与圆某条边相切
253 //2 完全在圆内
254 int relationcircle(circle c)
255 {
256     getline();
257     int i, x=2;
258     if (relationpoint(c.p)!=1) return 0;
259     for (i=0; i<n; i++)
260     {
261         if (c.relationseg(l[i])==2) return 0;
262         if (c.relationseg(l[i])==1) x=1;
263     }
264     return x;
265 }
266 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
272             [0].distance(tri[1])/2.0);
273     }
274     if (st==3)
275     {
276         c=circle(tri[0],tri[1],tri[2]);
277     }
278 void solve(int cur,int st,point tri[],circle
279     &c)
280 {
281     find(st,tri,c);
282     if (st==3)return;
283     int i;
284     for (i=0;i<cur;i++)
285     {
286         if (dblcmp(p[i].distance(c.p)-c.r)>0)
287         {
288             tri[st]=p[i];
289             solve(i,st+1,tri,c);
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         sort(v.begin(),v.end());
321         int cur=0;
322         for (j=0;j<v.size();j++)
323         {
324             if (v[j].second==-1)++cur;
325             else --cur;
326             ans=max(ans,cur);
327         }
328     }
329     return ans+1;
330 }
331 int pointinpolygon(point q)//点在凸多边形内部的判定
332 {
333     if (getdir())reverse(p,p+n);
334     if (dblcmp(q.sub(p[0]).det(p[n-1].sub(p[0]))
335         ==0)
336     {
337         if (line(p[n-1],p[0]).pointonseg(q))return
338             n-1;
339         return -1;
340     }
341     int low=1,high=n-2,mid;
342     while (low<=high)
343     {
344         mid=(low+high)>>1;
345         if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0]))
346             )>=0&&dblcmp(q.sub(p[0]).det(p[mid+1].
347                 sub(p[0]))<0)
348         {
349             polygon c;
350             c.p[0]=p[mid];
351             c.p[1]=p[mid+1];
352             c.p[2]=p[0];
353             c.n=3;
354             if (c.relationpoint(q))return mid;
355             return -1;
356         }
357         if (dblcmp(q.sub(p[0]).det(p[mid].sub(p[0]))
358             )>0)
359         {
360             low=mid+1;
361         }
362         else
363         {
364             high=mid-1;
365         }
366     }
367     return -1;
368 }
369 };

```

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
20             ): (X.y-s.y)/(t.y-s.y);
21         r=min(r,1.0);r=max(r,0.0);
22         e.pb(mp(r,i));
23     }
24     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[
30             p[i].n);
31     }
32     for (i=0;i<p.size();i++)
33     {
34         for (k=0;k<p[i].n;k++)
35         {
36             point &s=p[i].p[k],&t=p[i].p[(k+1)%p[i].
37                 ];
38             if (!dblcmp(s.det(t)))continue;
39             e.clear();
40             e.pb(mp(0.0,1));
41             e.pb(mp(1.0,-1));
42             for (j=0;j<p.size();j++)if (i!=j)
43             {
44                 for (w=0;w<p[j].n;w++)
45                 {
46                     point a=p[j].p[w],b=p[j].p[(w+1)%p[
47                         j].n],c=p[j].p[(w-1+p[j].n)%p[j].
48                         ];
49                     c0=dblcmp(t.sub(s).det(c.sub(s)));
50                     c1=dblcmp(t.sub(s).det(a.sub(s)));
51                     c2=dblcmp(t.sub(s).det(b.sub(s)));
52                     if (c1*c2<0)ins(s,t,line(s,t).
53                         crosspoint(line(a,b),-c2);
54                     else if (!c1&&c0*c2<0)ins(s,t,a,-c2);
55                     else if (!c1&&!c2)
56                     {
57                         int c3=dblcmp(t.sub(s).det(p[j].p[
58                             w+2)%p[j].n].sub(s));
59                         int dp=dblcmp(t.sub(s).dot(b.sub(s)
60                             ));
61                         if (dp&&c0)ins(s,t,a,dp>0?c0*((j>i)
62                             ^ (c0<0)):- (c0<0));
63                         if (dp&&c3)ins(s,t,b,dp>0?-c3*((j>i)
64                             ^ (c3<0)):c3<0);
65                     }
66                 }
67             }
68             sort(e.begin(),e.end());
69             int ct=0;
70             double tot=0.0,last;
71             for (j=0;j<e.size();j++)
72             {
73                 if (ct==p.size())tot+=e[j].first-last;
74                 ct+=e[j].second;
75                 last=e[j].first;
76             }
77             ans+=s.det(t)*tot;
78         }
79     }
80     return fabs(ans)*0.5;
81 }
82 };

```

4 graph

4.1 2SAT

```

1 /*
2   $x \& y == true:$ 
3   $\sim x \rightarrow x$ 
4   $\sim y \rightarrow y$ 
5
6   $x \& y == false:$ 
7   $x \rightarrow \sim y$ 
8   $y \rightarrow \sim x$ 
9
10  $x / y == true:$ 
11  $\sim x \rightarrow y$ 
12  $\sim y \rightarrow x$ 
13
14  $x / y == false:$ 
15  $x \rightarrow \sim x$ 
16  $y \rightarrow \sim y$ 
17
18  $x \wedge y == true:$ 
19  $\sim x \rightarrow y$ 
20  $y \rightarrow \sim x$ 
21  $x \rightarrow \sim y$ 
22  $\sim y \rightarrow x$ 
23
24  $x \wedge y == false:$ 
25  $x \rightarrow y$ 
26  $y \rightarrow x$ 
27  $\sim x \rightarrow \sim y$ 
28  $\sim y \rightarrow \sim x$ 
29 */
30 #include<stdio>
31 #include<cstring>
32
33 #define MAXX 16111
34 #define MAXE 200111
35 #define v to[i]
36
37 int edge[MAXX],to[MAXE],nxt[MAXE],cnt;
38 inline void add(int a,int b)
39 {
40     nxt[++cnt]=edge[a];
41     edge[a]=cnt;
42     to[cnt]=b;
43 }
44
45 bool done[MAXX];
46 int st[MAXX];
47
48 bool dfs(const int now)
49 {
50     if(done[now^1])
51         return false;
52     if(done[now])
53         return true;
54     done[now]=true;
55     st[cnt++]=now;
56     for(int i=edge[now];i;i=nxt[i])
57         if(!dfs(v))
58             return false;
59     return true;
60 }
61
62 int n,m;
63 int i,j,k;
64
65 inline bool go()
66 {
67     memset(done,0,sizeof done);
68     for(i=0;i<n;i+=2)
69         if(!done[i] && !done[i^1])
70         {
71             cnt=0;
72             if(!dfs(i))
73             {
74                 while(cnt)
75                     done[st[--cnt]]=false;
76                 if(!dfs(i^1))

```

```

77         return false;
78     }
79 }
80 return true;
81 }
82 //done array will be a solution with minimal
    lexicographical order
83 // or maybe we can solve it with dual SCC method
    and get a solution by reverse the edges of
    DAG then product a topsort

```

4.2 Articulation

```

1 void dfs(int now,int fa) // now 从 1 开始
2 {
3     int p(0);
4     dfn[now]=low[now]=cnt++;
5     for(std::list<int>::const_iterator it(edge[
        now].begin());it!=edge[now].end();++it)
6         if(dfn[*it]==-1)
7         {
8             dfs(*it,now);
9             ++p;
10            low[now]=std::min(low[now],low[*it]);
11            if((now==1 && p>1) || (now!=1 && low[
                *it]>=dfn[now])) // 如果从出发点出
                发的子节点不能由兄弟节点到达, 那么出发点
                为割点。如果现节点不是出发点, 但是其子
                节点不能达到祖先节点, 那么该节点为割点
12                ans.insert(now);
13        }
14        else
15            if(*it!=fa)
16                low[now]=std::min(low[now],dfn[*
                it]);
17 }

```

4.3 Augmenting Path Algorithm for Maximum Cardinality Bipartite Matching

```

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

```

```

31         memset(link,-1,sizeof(link));
32         int ans = 0;
33         for (int i=0; i<n; i++){
34             memset(visit,false,sizeof(visit));
35             if (dfs(i))
36                 ans++;
37         }
38         printf("%d\n",ans);
39     }
40 }

```

4.4 Biconnected Component - Edge

```

1 // hdu 4612
2 #include<cstdio>
3 #include<algorithm>
4 #include<set>
5 #include<cstring>
6 #include<stack>
7 #include<queue>
8
9 #define MAXX 200111
10 #define MAXE (1000111*2)
11 #pragma comment(linker, "/STACK:16777216")
12
13 int edge[MAXX],to[MAXE],nxt[MAXE],cnt;
14 #define v to[i]
15 inline void add(int a,int b)
16 {
17     nxt[++cnt]=edge[a];
18     edge[a]=cnt;
19     to[cnt]=b;
20 }
21
22 int dfn[MAXX],low[MAXX],col[MAXX],belong[MAXX];
23 int idx,bcnt;
24 std::stack<int>st;
25
26 void tarjan(int now,int last)
27 {
28     col[now]=1;
29     st.push(now);
30     dfn[now]=low[now]=++idx;
31     bool flag(false);
32     for (int i(edge[now]);i;i=nxt[i])
33     {
34         if(v==last && !flag)
35         {
36             flag=true;
37             continue;
38         }
39         if(!col[v])
40         {
41             tarjan(v,now);
42             low[now]=std::min(low[now],low[v]);
43             /*
44              if (low[v]>dfn[now])
45                  then this is a bridge
46              */
47         }
48         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         printf("%d\n",dist[go(go(1))]);
123         for(i=1;i<=bcnt;++i)
124             printf("%d\n",dist[i]);
125         puts("");
126         */
127         printf("%d\n",bcnt-1-dist[go(go(1))]);
128     }
129     return 0;
130 }

```

4.5 Biconnected Component

```

1  #include<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 [MAXM<1];
17
18 int head [MAXN],low [MAXN],dpt [MAXN],L;
19 bool visit [MAXN],cut [MAXN];
20 int idx;
21 std::stack<int> st;
22 int bcc [MAXM];
23
24 void init(int n)
25 {
26     L=0;
27     memset(head,-1,4*n);
28     memset(visit,0,n);
29 }
30
31 void add_edge(int u,int v)
32 {
33     edge[L].cut=edge[L].visit=false;
34     edge[L].to=v;
35     edge[L].next=head[u];
36     head[u]=L++;
37 }
38
39 void dfs(int u,int fu,int deg)
40 {
41     cut[u]=false;
42     visit[u]=true;
43     low[u]=dpt[u]=deg;
44     int tot=0;
45     for (int i=head[u]; i!=-1; i=edge[i].next)
46     {
47         int v=edge[i].to;
48         if (edge[i].visit)
49             continue;
50         st.push(i/2);
51         edge[i].visit=edge[i^1].visit=true;
52         if (visit[v])
53         {
54             low[u]=dpt[v]>low[u]?low[u]:dpt[v];
55             continue;
56         }
57         dfs(v,u,deg+1);
58         edge[i].cut=edge[i^1].cut=(low[v]>dpt[u]||edge[i].cut);
59         if (u!=fu) cut[u]=low[v]>=dpt[u]?1:cut[u];

```

```

    ];
60     if (low[v]>=dpt[u] || u==fu)
61     {
62         while (st.top()!=i/2)
63         {
64             int x=st.top()*2,y=st.top()*2+1;
65             bcc[st.top()]=idx;
66             st.pop();
67         }
68         bcc[i/2]=idx++;
69         st.pop();
70     }
71     low[u]=low[v]>low[u]?low[u]:low[v];
72     tot++;
73 }
74 if (u==fu && tot>1)
75     cut[u]=true;
76 }
77
78 int main()
79 {
80     int n,m;
81     while (scanf("%d%d",&n,&m)!=EOF)
82     {
83         init(n);
84         for (int i=0; i<m; i++)
85         {
86             int u,v;
87             scanf("%d%d",&u,&v);
88             add_edge(u,v);
89             add_edge(v,u);
90         }
91         idx=0;
92         for (int i=0; i<n; i++)
93             if (!visit[i])
94                 dfs(i,i,0);
95     }
96     return 0;
97 }

```

4.6 Blossom algorithm

```

1  #include<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(),
24                         p[x].rend()-i);
25             vis[z]=0;
26             *qb++=z;
27         }
28 }

```

```

28
29 inline bool bfs(int now)
30 {
31     static int i,x,y,z,b;
32     for (i=0;i<n;++i)
33         p[i].resize(0);
34     p[now].push_back(now);
35     memset(vis,-1,sizeof vis);
36     vis[now]=0;
37     qf=qb=q;
38     *qb++=now;
39
40     while (qf<qb)
41         for (x=*qf++;y=0;y<n;++y)
42             if (map[x][y] && m[y]!=y && vis[y]!=1)
43             {
44                 if (vis[y]==-1)
45                     if (m[y]==-1)
46                     {
47                         for (i=0;i+1<p[x].size();i
48                             +=2)
49                         {
50                             m[p[x][i]]=p[x][i+1];
51                             m[p[x][i+1]]=p[x][i];
52                         }
53                         m[x]=y;
54                         m[y]=x;
55                         return true;
56                     }
57                 else
58                 {
59                     p[z=m[y]]=p[x];
60                     p[z].push_back(y);
61                     p[z].push_back(z);
62                     vis[y]=1;
63                     vis[z]=0;
64                     *qb++=z;
65                 }
66             }
67         else
68         {
69             for (b=0;b<p[x].size() && b<p[
70                 y].size() && p[x][b]==p[y
71                 ][b];++b);
72             —b;
73             label(x,y,b);
74             label(y,x,b);
75         }
76     }
77     return false;
78 }
79
80 int i,j,k;
81 int ans;
82
83 int main()
84 {
85     scanf("%d",&n);
86     for (i=0;i<n;++i)
87         p[i].reserve(n);
88     while (scanf("%d%d",&i,&j)!=EOF)
89     {
90         —i;
91         —j;
92         map[i][j]=map[j][i]=true;
93     }
94     memset(m,-1,sizeof m);
95     for (i=0;i<n;++i)
96         if (m[i]==-1)
97         {
98             if (bfs(i))

```

```

95         ++ans;
96     else
97         m[i]=i;
98     }
99     printf("%d\n",ans<=1);
100    for (i=0;i<n;++i)
101        if (i<m[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
10                子节点不能够走到父节点之前去,那么该边为
11                桥
12            {
13                if (edge[now][i]<now)
14                {
15                    j=edge[now][i];
16                    k=now;
17                }
18                else
19                {
20                    j=now;
21                    k=edge[now][i];
22                }
23                ans.push_back(node(j,k));
24            }
25        }
26        else
27            if (edge[now][i]!=fa)
28                low[now]=std::min(low[now],low[edge[now][i]]);
29 }

```

4.8 Chu-Liu:Edmonds' Algorithm

```

1 #include<cstdio>
2 #include<cstring>
3 #include<vector>
4
5 #define MAXX 1111
6 #define MAXE 10111
7 #define inf 0x3f3f3f3f
8
9 int n,m,i,j,k,ans,u,v,tn,rt,sum,on,om;
10 int pre[MAXX],id[MAXX],in[MAXX],vis[MAXX];
11
12 struct edge
13 {
14     int a,b,c;
15     edge(){}
16     edge(int aa,int bb,int cc):a(aa),b(bb),c(cc)
17     {}
18 };
19 std::vector<edge>ed(MAXE);
20
21 int main()
22 {
23     while (scanf("%d_%d",&n,&m)!=EOF)
24

```

```

25 {
26     on=n;
27     om=m;
28     ed.resize(0);
29     sum=1;
30     while(m--)
31     {
32         scanf("%d_%d_%d",&i,&j,&k);
33         if (i!=j)
34         {
35             ed.push_back(edge(i,j,k));
36             sum+=k;
37         }
38     }
39     ans=0;
40     rt=n;
41     for (i=0;i<n;++i)
42         ed.push_back(edge(n,i,sum));
43     ++n;
44     while(true)
45     {
46         memset(in,0x3f,sizeof in);
47         for (i=0;i<ed.size();++i)
48             if (ed[i].a!=ed[i].b && in[ed[i].b]
49                 >ed[i].c)
50             {
51                 in[ed[i].b]=ed[i].c;
52                 pre[ed[i].b]=ed[i].a;
53                 if (ed[i].a==rt)
54                     j=i;
55             }
56         for (i=0;i<n;++i)
57             if (i!=rt && in[i]==inf)
58                 goto ot;
59         memset(id,-1,sizeof id);
60         memset(vis,-1,sizeof vis);
61         tn=in[rt]=0;
62         for (i=0;i<n;++i)
63         {
64             ans+=in[i];
65             for (v=i;vis[v]!=i && id[v]==-1 &&
66                 v!=rt;v=pre[v])
67                 vis[v]=i;
68             if (v!=rt && id[v]==-1)
69             {
70                 for (u=pre[v];u!=v;u=pre[u])
71                     id[u]=tn;
72                 id[v]=tn++;
73             }
74         }
75         if (!tn)
76             break;
77         for (i=0;i<n;++i)
78             if (id[i]==-1)
79                 id[i]=tn++;
80         for (i=0;i<ed.size();++i)
81         {
82             v=ed[i].b;
83             ed[i].a=id[ed[i].a];
84             ed[i].b=id[ed[i].b];
85             if (ed[i].a!=ed[i].b)
86                 ed[i].c=in[v];
87         }
88         n=tn;
89         rt=id[rt];
90     }
91     if (ans>=2*sum)
92         puts("impossible");
93     else
94         printf("%d_%d\n",ans-sum,j-om);
95 }

```

ot:

```

91     puts("");
92 }
93 return 0;
94 }

```

4.9 Covering problems

1 最大团以及相关知识

2

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

4

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

6

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

8

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

10

11 性质：

12 最大独立集 + 最小覆盖集 = V

13 最大团 = 补图的最大独立集

14 最小覆盖集 = 最大匹配

15

16 minimum cover:

17 vertex cover vertex bipartite graph = maximum cardinality bipartite matching

18 找完最大二分匹配後，有三種情況要分別處理：

19 甲、 X 側未匹配點的交錯樹們。

20 乙、 Y 側未匹配點的交錯樹們。

21 丙、層層疊疊的交錯環們 (包含單獨的匹配邊)。

22 這三個情況互不干涉。用 Graph Traversal 建立甲、乙的交錯樹們，剩下部分就是丙。

23 要找點覆蓋，甲、乙是取盡奇數距離的點，丙是取盡偶數距離的點，或者是取盡奇數距離的點，每塊連通分量可以各自為政。另外，小心處理的話，是可以印出字典順序最小的點覆蓋的。

24 已經有最大匹配時，求點覆蓋的時間複雜度等同於一次 Graph Traversal 的時間。

25

26 vertex cover edge

27

28 edge cover vertex

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

30

31 edge cover edge

32

33 path cover vertex

34 general graph: NP-H

35 tree: DP

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

37

38 path cover edge

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

40

41 cycle cover vertex

42 general: NP-H

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

44

45 cycle cover edge

46 NP-H

4.10 Difference constraints

for $a - b \leq c$
add(b, a, c);

最短路得最远解
最长路得最近解
//根据情况反转边?(反转方向及边权)

全 0 点得普通解

4.11 Dinitz's algorithm

```

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

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

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

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

inline bool bfs()
{
    static int *qb, *qb;
    static int i;
    memset(h, -1, sizeof h);
    qb = qb;
    h[*qb++ = source] = 0;
    for (; qb != sink; ++qb)
        for (i = edge[*qb]; i != -1; i = nxt[i])
            if (cap[i] && h[to[i]] == -1)
                h[*qb++ = to[i]] = h[*qb] + 1;
    return h[sink] != -1;
}

int dfs(int now, int maxcap)
{
    if (now == sink)
        return maxcap;
    int flow(maxcap), d;
    for (int &i(w[now]); i != -1; i = nxt[i])

```

43	<code>if (cap[i] && h[to[i]]==h[now]+1) // 66 (110</code>	<code>*/</code>
	<code>flow=dfs(to[i], std::min(maxcap, cap[</code>	<code>}</code>
	<code>)))</code>	<code>printf("%d\n", ans);</code>
44	<code>{</code>	<code>}</code>
45	<code>d=dfs(to[i], std::min(flow, cap[i]));</code>	<code>return 0;</code>
46	<code>cap[i]-=d;</code>	<code>}</code>
47	<code>cap[i^1]+=d;</code>	
48	<code>flow-=d;</code>	
49	<code>if (!flow)</code>	
50	<code>return maxcap;</code>	
51	<code>}</code>	
52	<code>return maxcap-flow;</code>	
53	<code>}</code>	
54		
55	<code>int nc, np, m, i, j, k;</code>	
56	<code>int ans;</code>	
57		
58	<code>int main()</code>	
59	<code>{</code>	
60	<code>while (scanf("%d%d%d", &n, &np, &nc, &m) != EOF)</code>	1 Maximum weighted closure of a graph:
	<code>{</code>	2
61	<code>{</code>	3 所有由这个子图中的点出发的边都指向这个子图，那么这个子图为
62	<code>cnt=0;</code>	原图的一个 closure (闭合子图)
63	<code>memset(edge, -1, sizeof edge);</code>	4
64	<code>while(m--)</code>	5 每个节点向其所有依赖节点连边，容量 inf
65	<code>{</code>	6 源点向所有正权值节点连边，容量为该权值
66	<code>while (getchar() != ' ');</code>	7 所有负权值节点向汇点连边，容量为该权值绝对值
67	<code>scanf("%d", &i);</code>	8 以上均为有向边
68	<code>while (getchar() != ' ');</code>	9 最大权为 sum{正权值} - {新图的最小割}
69	<code>scanf("%d", &j);</code>	10 残量图中所有由源点可达的点即为所选子图
70	<code>while (getchar() != ' ');</code>	11
71	<code>scanf("%d", &k);</code>	12
72	<code>if (i != j)</code>	13
73	<code>{</code>	14 Eulerian circuit:
74	<code>++i;</code>	15 计入度和出度之差
75	<code>++j;</code>	16 无向边任意定向
76	<code>add(i, j, k);</code>	17 出入度之差为奇数则无解
77	<code>add(j, i, 0);</code>	18 然后构图:
78	<code>}</code>	19 原图有向边不变，容量 1 // 好像需要在新图中忽略有向边?
79	<code>}</code>	20 无向边按之前认定方向，容量 1
80	<code>source=++n;</code>	21 源点向所有度数为正的点连边，容量 abs(度数/2)
81	<code>while(np--)</code>	22 所有度数为负的点向汇点连边，容量 abs(度数/2)
82	<code>{</code>	23 两侧均满流则有解
83	<code>while (getchar() != ' ');</code>	24 相当于规约为可行流问题
84	<code>scanf("%d", &i);</code>	25 注意连通性的 trick
85	<code>while (getchar() != ' ');</code>	26
86	<code>scanf("%d", &j);</code>	27 终点到起点加一条有向边即可将 path 问题转为 circuit 问题
87	<code>++i;</code>	28
88	<code>add(source, i, j);</code>	29
89	<code>add(i, source, 0);</code>	30
90	<code>}</code>	31 Feasible flow problem:
91	<code>sink=++n;</code>	32 由超级源点出发的边全部满流则有解
92	<code>while(nc--)</code>	33 有源汇时，由汇点向源点连边，下界 0 上界 inf 即可转化为无源无
93	<code>{</code>	汇上下界流
94	<code>while (getchar() != ' ');</code>	34
95	<code>scanf("%d", &i);</code>	35 对于每条边 <a->b capu, d>, 建边 <ss->b cap(u)>, <a->st
96	<code>while (getchar() != ' ');</code>	cap(u)>, <a->b cap(d-u)>
97	<code>scanf("%d", &j);</code>	36
98	<code>++i;</code>	37 Maximum flow: //好像也可以二分
99	<code>add(i, sink, j);</code>	38 //将流量还原至原图后，在残量网络上继续完成最大流
100	<code>add(sink, i, 0);</code>	39 直接把 source 和 sink 设为原来的 st，此时输出的最大流即是答案
101	<code>}</code>	40 不需要删除或者调整 t->s 弧
102	<code>ans=0;</code>	41 Minimum flow: //好像也可以二分
103	<code>while(bfs())</code>	42 建图时先不连汇点到源点的边，新图中完成最大流之后再连原汇至
104	<code>{</code>	原源的边完成第二次最大流，此时 t->s 这条弧的流量即为最
105	<code>memcpy(w, edge, sizeof edge);</code>	小流
106	<code>ans+=dfs(source, inf);</code>	43 判断可行流存在还是必须连原汇 -> 原源的边之后查看满流
107	<code>/*</code>	44 所以可以使用跑流 -> 加 ts 弧 -> 跑流，最后检查超级源点满流情
108	<code>while ((k=dfs(source, inf)))</code>	况来一步搞定
109	<code>ans+=k;</code>	45 tips:
		46 合并流量、减少边数来加速
		47
		48
		49
		50 Minimum cost feasible flow problem:
		51 TODO
		52 看起来像是在上面那样跑费用流就行了……
		53
		54
		55

56	Minimum weighted vertex cover edge for bipartite graph:	122	不参与决策的边权设为 inf 来排除掉;
57	for all vertex in X:	123	贪心一个初始不合法情况, 然后通过可行流调整; // refer: 混合图
58	edge < s->x cap(weight(x)) >		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
59	for all vertex in Y:	124	一次后回到原点)
60	edge < y->t cap(weight(y)) >		按时间拆点 (时间层……?);
61	for original edges		
62	edge < x->y cap(inf) >		
63			
64	ans={maximum flow}={minimum cut}	1	//if every point connect with not less than [(N
65	残量网络中的所有简单割 ((源点可达 && 汇点不可达) (源点不可达 && 汇点可达)) 对应着解	2	+1)/2] points
66		3	#include<stdio>
67		4	#include<algorithm>
68		5	#include<cstring>
69	Maximum weighted vertex independent set for bipartite graph:	6	#define MAXX 177
70	ans=Sum 点权 -valueMinimum weighted vertex cover edge	7	#define MAX (MAXX*MAXX)
71	解应该就是最小覆盖集的补图吧……	8	
72		9	int edge[MAXX],nxt[MAX],to[MAX],cnt;
73		10	
74		11	inline void add(int a,int b)
75	方格取数: // refer: hdu 3820 golden eggs 取方格获得收益当取了相邻方格时付出边的代价	12	{
76		13	nxt[++cnt]=edge[a];
77		14	edge[a]=cnt;
78		15	to[cnt]=b;
79	必取的方格到源/汇的边的容量 inf	16	}
80	相邻方格之间的边的容量为 {代价}*2	17	
81	ans=sum{方格收益}-{最大流}	18	bool done[MAXX];
82		19	int n,m,i,j,k;
83		20	
84		21	inline int find(int a)
85	最小割的唯一性: // refer: 关键边。有向边起点为 s 集, 终点为 t 集	22	{
86	从源和汇分别能够到的点集是所有点时, 最小割唯一	23	static int i;
87	也就是每一条增广路径都仅有一条边满流	24	for(i=edge[a];i;i=nxt[i])
88	注意查看的是实际的网络, 不是残量网络	25	if(!done[to[i]])
89		26	{
90	具体来说	27	edge[a]=nxt[i];
91		28	return to[i];
92	void rr(int now)	29	}
93	{	30	return 0;
94	done[now]=true;	31	}
95	++cnt;	32	
96	for(int i(edge[now]);i!=-1;i=nxt[i])	33	int a,b;
97	if(cap[i] && !done[v])	34	int next[MAXX],pre[MAXX];
98	rr(v);	35	bool mat[MAXX][MAXX];
99	}	36	
100		37	int main()
101	void dfs(int now)	38	{
102	{	39	while(scanf("%d%d",&n,&m)!=EOF)
103	done[now]=true;	40	{
104	++cnt;	41	for(i=1;i<=n;++i)
105	for(int i(edge[now]);i!=-1;i=nxt[i])	42	next[i]=done[i]=edge[i]=0;
106	if(cap[i^1] && !done[v])	43	memset(mat,0,sizeof mat);
107	dfs(v);	44	cnt=0;
108	}	45	while(m--)
109		46	{
110	memset(done,0,sizeof done);	47	scanf("%d%d",&i,&j);
111	cnt=0;	48	add(i,j);
112	rr(source);	49	add(j,i);
113	dfs(sink);	50	mat[i][j]=mat[j][i]=true;
114	puts(cnt==n?"UNIQUE":"AMBIGUOUS");	51	}
115		52	a=1;
116		53	b=to[edge[a]];
117		54	cnt=2;
118	Tips:	55	done[a]=done[b]=true;
119	两点间可以不止有一种边, 也可以不止有一条边, 无论有向无向;	56	next[a]=b;
120	两点间容量 inf 则可以设法化简为一个点;	57	while(cnt<n)
121	点权始终要转化为边权;	58	{
		59	while(i=find(a))
		60	{
		61	next[i]=a;

4.13 Hamiltonian circuit

```

62         done[a=i]=true;
63         ++cnt;
64     }
65     while(i=find(b))
66     {
67         next[b]=i;
68         done[b=i]=true;
69         ++cnt;
70     }
71     if(!mat[a][b])
72         for(i=next[a]; next[i]!=b; i=next[i])
73             if(mat[a][next[i]] && mat[i][b])
74                 {
75                     for(j=next[i]; j!=b; j=next[j])
76                         pre[next[j]]=j;
77                     for(j=b; j!=next[i]; j=pre[j])
78                         next[j]=pre[j];
79                     std::swap(next[i], b);
80                     break;
81                 }
82     next[b]=a;
83     for(i=a; i!=b; i=next[i])
84         if(find(i))
85         {
86             a=next[b=i];
87             break;
88         }
89     }
90     while(a!=b)
91     {
92         printf("%d ", a);
93         a=next[a];
94     }
95     printf("%d\n", b);
96 }
97 return 0;
98 }

```

4.14 Hopcroft-Karp algorithm

```

1  #include<cstdio>
2  #include<cstring>
3
4  #define MAXX 50111
5  #define MAX 150111
6
7  int nx, p;
8  int i, j, k;
9  int x, y;
10 int ans;
11 bool flag;
12
13 int edge[MAXX], nxt[MAX], to[MAX], cnt;
14
15 int cx[MAXX], cy[MAXX];
16 int px[MAXX], py[MAXX];
17
18 int q[MAXX], *qf, *qb;
19
20 bool ag(int i)
21 {
22     int j, k;
23     for(k=edge[i]; k; k=nxt[k])
24         if(py[j=to[k]]==px[i]+1)
25             {
26                 py[j]=0;

```

```

27         if(cy[j]==-1 || ag(cy[j]))
28             {
29                 cx[i]=j;
30                 cy[j]=i;
31                 return true;
32             }
33     }
34     return false;
35 }
36
37 int main()
38 {
39     scanf("%d%d", &nx, &p);
40     while(p--)
41     {
42         scanf("%d", &i, &j);
43         nxt[++cnt]=edge[i];
44         edge[i]=cnt;
45         to[cnt]=j;
46     }
47     memset(cx, -1, sizeof cx);
48     memset(cy, -1, sizeof cy);
49     while(true)
50     {
51         memset(px, 0, sizeof px);
52         memset(py, 0, sizeof py);
53         qf=qb=q;
54         flag=false;
55
56         for(i=1; i<=nx; ++i)
57             if(cx[i]==-1)
58                 *qb++=i;
59         while(qf!=qb)
60             for(k=edge[i=*qf++]; k; k=nxt[k])
61                 if(!py[j=to[k]])
62                     {
63                         py[j]=px[i]+1;
64                         if(cy[j]==-1)
65                             flag=true;
66                         else
67                             {
68                                 px[cy[j]]=py[j]+1;
69                                 *qb++=cy[j];
70                             }
71                     }
72         if(!flag)
73             break;
74         for(i=1; i<=nx; ++i)
75             if(cx[i]==-1 && ag(i))
76                 ++ans;
77     }
78     printf("%d\n", ans);
79     return 0;
80 }

```

4.15 Improved Shortest Augmenting Path Algorithm

```

1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4
5  #define MAXX 5111
6  #define MAXM (30111*4)
7  #define inf 0x3f3f3f3f3f3f3f3f
8
9  int edge[MAXX], to[MAXM], nxt[MAXM], cnt;
10 #define v to[i]
11 long long cap[MAXM];
12

```

```

13 int n;
14 int h[MAXX], gap[MAXX], pre[MAXX], w[MAXX];
15
16 inline void add(int a, int b, long long c)
17 {
18     nxt[++cnt]=edge[a];
19     edge[a]=cnt;
20     to[cnt]=b;
21     cap[cnt]=c;
22 }
23
24 int source, sink;
25
26 inline long long go(const int N=sink)
27 {
28     static int now, N, i;
29     static long long min, mf;
30     memset(gap, 0, sizeof gap);
31     memset(h, 0, sizeof h);
32     memcpy(w, edge, sizeof w);
33     gap[0]=N;
34     mf=0;
35
36     pre[now=source]=-1;
37     while(h[source]<N)
38     {
39 rep:
40         if(now==sink)
41         {
42             min=inf;
43             for(i=pre[sink]; i!=-1; i=pre[to[i^1]])
44                 if(min>=cap[i])
45                 {
46                     min=cap[i];
47                     now=to[i^1];
48                 }
49             for(i=pre[sink]; i!=-1; i=pre[to[i^1]])
50             {
51                 cap[i]-=min;
52                 cap[i^1]+=min;
53             }
54             mf+=min;
55         }
56         for(int &i(w[now]); i!=-1; i=nxt[i])
57             if(cap[i] && h[v]+1==h[now])
58             {
59                 pre[now=v]=i;
60                 goto rep;
61             }
62         if(!--gap[h[now]])
63             return mf;
64         min=N;
65         for(i=w[now]=edge[now]; i!=-1; i=nxt[i])
66             if(cap[i])
67                 min=std::min(min, (long long)h[v]
68                                     ;
69                 ++gap[h[now]=min+1];
70                 if(now!=source)
71                     now=to[pre[now]^1];
72             }
73         return mf;
74 }
75
76 int m, i, j, k;
77 long long ans;
78
79 int main()
80 {
81     scanf("%d%d", &n, &m);
82     source=1;

```

```

82     sink=n;
83     cnt=-1;
84     memset(edge, -1, sizeof edge);
85     while(m--)
86     {
87         scanf("%d%d", &i, &j, &ans);
88         add(i, j, ans);
89         add(j, i, ans);
90     }
91     printf("%lld\n", go());
92     return 0;
93 }

```

4.16 k Shortest Path

```

1 #include<cstdio>
2 #include<cstring>
3 #include<queue>
4 #include<vector>
5
6 int K;
7
8 class states
9 {
10 public:
11     int cost, id;
12 };
13
14 int dist[1000];
15
16 class cmp
17 {
18 public:
19     bool operator()(const states &i, const
20                     states &j)
21     {
22         return i.cost>j.cost;
23     }
24 };
25
26 class cmp2
27 {
28 public:
29     bool operator()(const states &i, const
30                     states &j)
31     {
32         return i.cost+dist[i.id]>j.cost+dist[
33             j.id];
34     }
35 };
36
37 struct edges
38 {
39     int to, next, cost;
40 } edger[100000], edge[100000];
41
42 int headr[1000], head[1000], Lr, L;
43
44 void dijkstra(int s)
45 {
46     states u;
47     u.id=s;
48     u.cost=0;
49     dist[s]=0;
50     std::priority_queue<states, std::vector<states>
51         , cmp> q;
52     q.push(u);
53     while(!q.empty())
54     {
55         u=q.top();

```

```

52     q.pop();
53     if (u.cost!=dist[u.id])
54         continue;
55     for (int i=headr[u.id]; i!=-1; i=edger[i]
56         .next)
57     {
58         states v=u;
59         v.id=edger[i].to;
60         if (dist[v.id]>dist[u.id]+edger[i].
61             cost)
62         {
63             v.cost=dist[v.id]=dist[u.id]+
64                 edger[i].cost;
65             q.push(v);
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>
96         ,>,cmp2> q;
97     states tmp;
98     tmp.id=s;
99     tmp.cost=0;
100    q.push(tmp);
101    while (!q.empty())
102    {
103        states u=q.top();
104        q.pop();
105        num[u.id]++;
106        if (num[t]==K)
107            return u.cost;
108        for (int i=head[u.id]; i!=-1; i=edge[i]
109            next)
110        {
111            int v=edge[i].to;
112            tmp.id=v;
113            tmp.cost=u.cost+edge[i].cost;
114            q.push(tmp);
115        }
116    }

```

```

117
118 int main()
119 {
120     int n,m;
121     scanf("%d%d",&n,&m);
122     init(n);
123     for (int i=0; i<m; i++)
124     {
125         int u,v,x;
126         scanf("%d%d%d",&u,&v,&x);
127         add_edge(u-1,v-1,x);
128     }
129     int s,t;
130     scanf("%d%d%d",&s,&t,&K);
131     if (s==t)
132         ++K;
133     dijkstra(t-1);
134     printf("%d\n",a_star(s-1,t-1));
135     return 0;
136 }

```

4.17 Kariv-Hakimi Algorithm

```

1 //Absolute Center of a graph, not only a tree
2 #include<cstdio>
3 #include<algorithm>
4 #include<vector>
5 #include<cstring>
6 #include<set>
7
8 #define MAXX 211
9 #define inf 0x3f3f3f3f
10
11 int e[MAXX][MAXX],dist[MAXX][MAXX];
12 double dp[MAXX],ta;
13 int ans,d;
14 int n,m,a,b;
15 int i,j,k;
16 typedef std::pair<int,int> pii;
17 std::vector<pii>vt[2];
18 bool done[MAXX];
19 typedef std::pair<double,int> pdi;
20 std::multiset<pdi>q;
21 int pre[MAXX];
22
23 int main()
24 {
25     vt[0].reserve(MAXX);
26     vt[1].reserve(MAXX);
27     scanf("%d%d",&n,&m);
28     memset(e,0x3f,sizeof(e));
29     while(m--)
30     {
31         scanf("%d%d%d",&i,&j,&k);
32         e[i][j]=e[j][i]=std::min(e[i][j],k);
33     }
34     for (i=1;i<=n;++i)
35         e[i][i]=0;
36     memcpy(dist,e,sizeof(dist));
37     for (k=1;k<=n;++k)
38         for (i=1;i<=n;++i)
39             for (j=1;j<=n;++j)
40                 dist[i][j]=std::min(dist[i][j],
41                     dist[i][k]+dist[k][j]);
42     ans=inf;
43     for (i=1;i<=n;++i)
44         for (j=i;j<=n;++j)
45             if (e[i][j]!=inf)
46                 vt[0].resize(0);

```

```

47     vt[1].resize(0); 104
48     static int i; 105
49     for(i=1;i<=n;++i) 106
50         vt[0].push_back(pii(dist[::i][i], dist[j][i])); 107
51     std::sort(vt[0].begin(), vt[0].end()); 108
52     for(i=0;i<vt[0].size();++i) 110
53     { 111
54         while(!vt[1].empty() && vt[1].back().second<=vt[0][i].second) 112
55             vt[1].pop_back(); 113
56         vt[1].push_back(vt[0][i]); 114
57     } 115
58     d=inf; 116
59     if(vt[1].size()==1) 117
60         if(vt[1][0].first<vt[1][0].second) 118
61         { 1
62             ta=0; 2
63             d=(vt[1][0].first<<1); 3
64         } 4
65         else 5
66         { 6
67             ta=e[::i][j]; 7
68             d=(vt[1][0].second<<1); 8
69         } 9
70     else 10
71     for(i=1;i<vt[1].size();++i) 11
72     if(d>e[::i][j]+vt[1][i-1].first+vt[1][i].second) 12
73     { 13
74         ta=(e[::i][j]+vt[1][i-1].first)/(double)2.0f; 14
75         d=e[::i][j]+vt[1][i-1].first+vt[1][i].second; 15
76     } 16
77     if(d<ans) 17
78     { 18
79         ans=d; 19
80         a=::i; 20
81         b=j; 21
82         dp[::i]=ta; 22
83         dp[j]=e[::i][j]-ta; 23
84     } 24
85     } 25
86     printf("%d\n", ans); 26
87     for(i=1;i<=n;++i) 27
88         if(i!=a && i!=b) 28
89             dp[i]=1e20; 29
90     q.insert(pdi(dp[a], a)); 30
91     if(a!=b) 31
92         q.insert(pdi(dp[b], b)); 32
93     if(a!=b) 33
94         pre[b]=a; 34
95     while(!q.empty()) 35
96     { 36
97         k=q.begin()->second; 37
98         q.erase(q.begin()); 38
99         if(done[k]) 39
100             continue; 40
101         done[k]=true; 41
102         for(i=1;i<=n;++i) 42
103             if(e[k][i]!=inf && dp[k]+e[k][i]<dp[i]) 43
104             { 44
105                 dp[i]=dp[k]+e[k][i]; 45
106                 q.insert(pdi(dp[i], i)); 46
107                 pre[i]=k; 47
108             } 48
109     } 49
110     vt[0].resize(0); 50
111     for(i=1;i<=n;++i) 51
112         if(pre[i]) 52
113             if(i<pre[i]) 53
114                 printf("%d %d\n", i, pre[i]); 54
115             else 55
116                 printf("%d %d\n", pre[i], i); 56
117     return 0; 57
118 } 58

```

4.18 Kuhn-Munkres algorithm

```

bool match(int u)//匈牙利
{
    vx[u]=true;
    for(int i=1;i<=n;++i)
        if(lx[u]+ly[i]==g[i][i]&&!vy[i])
        {
            vy[i]=true;
            if(!d[i]||match(d[i]))
            {
                d[i]=u;
                return true;
            }
        }
    return false;
}

inline void update()//
{
    int i, j;
    int a=1<<30;
    for(i=1;i<=n;++i) if(vx[i])
        for(j=1;j<=n;++j) if(!vy[j])
            a=min(a, lx[i]+ly[j]-g[i][j]);
    for(i=1;i<=n;++i)
    {
        if(vx[i]) lx[i]-=a;
        if(vy[i]) ly[i]+=a;
    }
}

void km()
{
    int i, j;
    for(i=1;i<=n;++i)
    {
        lx[i]=ly[i]=d[i]=0;
        for(j=1;j<=n;++j)
            lx[i]=max(lx[i], g[i][j]);
    }
    for(i=1;i<=n;++i)
    {
        while(true)
        {
            memset(vx, 0, sizeof(vx));
            memset(vy, 0, sizeof(vy));
            if(match(i))
                break;
            update();
        }
    }
    int ans=0;
    for(i=1;i<=n;++i)
        if(d[i]!=0)
            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 //算法：求二分图最佳匹配km n复杂度~3
71 //匈牙利求增广路
72 int dfs(int u)
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km
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
121                     lx[i]-=dx;
122                     if (sy[i])
123                         ly[i]+=dx;
124                 }
125             }
126             int sum=0;
127             for (i=1; i<=n; i++)
128                 sum+=map[match[i]][i];
129             return sum;
130         }
131     }
132 }
133
134 4.19 LCA - DA
135
136 int edge[MAXX],nxt[MAXX<1],to[MAXX<1],cnt;
137 int pre[MAXX][N],dg[MAXX];
138
139 inline void add(int j,int k)
140 {
141     nxt[++cnt]=edge[j];
142     edge[j]=cnt;
143     to[cnt]=k;
144 }
145
146 void rr(int now,int fa)
147 {
148     dg[now]=dg[fa]+1;
149     for(int i=edge[now]; i;i=nxt[i])
150         if(to[i]!=fa)
151         {
152             static int j;
153             j=1;
154             for(pre[to[i]][0]=now; j<N;++j)
155                 pre[to[i]][j]=pre[pre[to[i]][j-1]][j-1];
156             rr(to[i],now);
157         }
158 }
159
160 inline int lca(int a,int b)
161 {
162     static int i,j;
163     j=0;
164     if(dg[a]<dg[b])
165         std::swap(a,b);
166     for(i=dg[a]-dg[b]; i;i>=1,++j)
167         if(i&1)
168             a=pre[a][j];
169     if(a==b)
170         return a;
171     for(i=N-1; i>=0; --i)
172         if(pre[a][i]!=pre[b][i])
173         {
174             a=pre[a][i];
175             b=pre[b][i];
176         }
177     return pre[a][0];
178 }
179
180 // looks like above is a wrong version
181
182 static int i,log;
183 for(log=0;(1<=(log+1))<=dg[a];++log);
184 for(i=log; i>=0; --i)
185     if(dg[a]-(1<=(i))>=dg[b])
186         a=pre[a][i];
187 if(a==b)
188     return a;
189 for(i=log; i>=0; --i)
190     if(pre[a][i]!=-1 && pre[a][i]!=pre[b][i])
191         a=pre[a][i],b=pre[b][i];

```

```

56     return pre[a][0];
57 }

```

4.20 LCA - tarjan - minmax

```

1  #include<cstdio>
2  #include<list>
3  #include<algorithm>
4  #include<cstring>
5
6  #define MAXX 100111
7  #define inf 0x5fffffff
8
9  short T,t;
10 int set[MAXX],min[MAXX],max[MAXX],ans[2][MAXX];
11 bool done[MAXX];
12 std::list<std::pair<int,int>> edge[MAXX];
13 std::list<std::pair<int,int>> q[MAXX];
14 int n,i,j,k,l,m;
15
16 struct node
17 {
18     int a,b,id;
19     node() {}
20     node(const int &aa,const int &bb,const int &
21         idd): a(aa),b(bb),id(idd) {}
22 };
23 std::list<node> to[MAXX];
24
25 int find(const int &a)
26 {
27     if(set[a]==a)
28         return a;
29     int b(set[a]);
30     set[a]=find(set[a]);
31     max[a]=std::max(max[a],max[b]);
32     min[a]=std::min(min[a],min[b]);
33     return set[a];
34 }
35
36 void tarjan(const int &now)
37 {
38     done[now]=true;
39     for(std::list<std::pair<int,int>> >::
40         const_iterator it(q[now].begin());it!=q[
41         now].end();++it)
42         if(done[it->first])
43             if(it->second>0)
44                 to[find(it->first)].push_back(
45                     node(now,it->first,it->second)
46                     ));
47         else
48             to[find(it->first)].push_back(
49                 node(it->first,now,-it->
50                 second));
51     for(std::list<std::pair<int,int>> >::
52         const_iterator it(edge[now].begin());it!=
53         edge[now].end();++it)
54         if(!done[it->first])
55         {
56             tarjan(it->first);
57             set[it->first]=now;
58             min[it->first]=it->second;
59             max[it->first]=it->second;
60         }
61     for(std::list<node>::const_iterator it(to[now]
62         .begin());it!=to[now].end();++it)
63     {
64         find(it->a);

```

```

56     find(it->b);
57     ans[0][it->id]=std::min(min[it->b],min[it
58         ->a]);
59     ans[1][it->id]=std::max(max[it->a],max[it
60         ->b]);
61 }
62 }
63
64 int main()
65 {
66     scanf("%hd",&T);
67     for(t=1;t<=T;++t)
68     {
69         scanf("%d",&n);
70         for(i=1;i<=n;++i)
71         {
72             edge[i].clear();
73             q[i].clear();
74             to[i].clear();
75             done[i]=false;
76             set[i]=i;
77             min[i]=inf;
78             max[i]=0;
79         }
80         for(i=1;i<n;++i)
81         {
82             scanf("%d%d%d",&j,&k,&l);
83             edge[j].push_back(std::make_pair(k,l)
84             );
85             edge[k].push_back(std::make_pair(j,l)
86             );
87         }
88         scanf("%d",&m);
89         for(i=0;i<m;++i)
90         {
91             scanf("%d_%d",&j,&k);
92             q[j].push_back(std::make_pair(k,i));
93             q[k].push_back(std::make_pair(j,-i));
94         }
95         tarjan(1);
96         printf("Case_%hd:\n",t);
97         for(i=0;i<m;++i)
98             printf("%d_%d\n",ans[0][i],ans[1][i])
99             ;
100     }
101     return 0;
102 }

```

4.21 Minimum Ratio Spanning Tree

```

1  #include<cstdio>
2  #include<cstring>
3  #include<cmath>
4
5  #define MAXX 1111
6
7  struct
8  {
9      int x,y;
10     double z;
11 } node[MAXX];
12
13 struct
14 {
15     double l,c;
16 } map[MAXX][MAXX];
17
18 int n,l,f[MAXX],pre[MAXX];
19 double dis[MAXX];
20

```

```

21 double mst(double x)
22 {
23     int i, j, tmp;
24     double min, s=0, t=0;
25     memset(f, 0, sizeof(f));
26     f[1]=1;
27     for (i=2; i<=n; i++)
28     {
29         dis[i]=map[1][i].c-map[1][i].l*x;
30         pre[i]=1;
31     }
32     for (i=1; i<n; i++)
33     {
34         min=1e10;
35         for (j=1; j<=n; j++)
36             if (!f[j] && min>dis[j])
37             {
38                 min=dis[j];
39                 tmp=j;
40             }
41         f[tmp]=1;
42         t+=map[pre[tmp]][tmp].l;
43         s+=map[pre[tmp]][tmp].c;
44         for (j=1; j<=n; j++)
45             if (!f[j] && map[tmp][j].c-map[tmp][j].l*x<dis[j])
46             {
47                 dis[j]=map[tmp][j].c-map[tmp][j].l*x;
48                 pre[j]=tmp;
49             }
50     }
51     return s/t;
52 }
53
54 int main()
55 {
56     int i, j;
57     double a, b;
58     while (scanf("%d", &n), n);
59     {
60         for (i=1; i<=n; i++)
61             scanf("%d%d%lf", &node[i].x, &node[i].y, &node[i].z);
62         for (i=1; i<=n; i++)
63             for (j=i+1; j<=n; j++)
64             {
65                 map[j][i].l=map[i][j].l=sqrt
66                     (1.0*(node[i].x-node[j].x)*(node[i].x-node[j].x)+(node[i].y-node[j].y)*(node[i].y-node[j].y));
67                 map[j][i].c=map[i][j].c=fabs(node[i].x-node[j].x);
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("%.3lf\n", b);
76     }
77     return 0;
78 }

```

4.22 Minimum Steiner Tree

```
1 #include<stdio>
```

```

2 #include<cstring>
3 #include<algorithm>
4 #include<queue>
5
6 #define MAXX 211
7 #define MAXE 10111
8 #define inf 0x3f3f3f3f
9
10 int edge[MAXX], nxt[MAXE], to[MAXE], wg[MAXE], cnt;
11 inline void add(int a, int b, int c)
12 {
13     nxt[++cnt]=edge[a];
14     edge[a]=cnt;
15     to[cnt]=b;
16     wg[cnt]=c;
17 }
18
19 int dp[1<<8];
20 int s[MAXX];
21 int d[1<<8][MAXX];
22 int S[MAXX], P[MAXX];
23 int fac[8];
24
25 struct node
26 {
27     int a, b, dist;
28     node() {}
29     node(int i, int j, int k):a(i), b(j), dist(k) {}
30     bool operator<(const node &i) const
31     {
32         return dist>i.dist;
33     }
34     int &get()
35     {
36         return d[b][a];
37     }
38 } now;
39
40 std::priority_queue<node> q;
41
42 int n, m, nn, i, j, k;
43 int cs, cf, x, y;
44 int ans, cst;
45
46 inline bool check(int x)
47 {
48     static int re, i;
49     for (i=re=0; x>=1; ++i)
50         re+=(x&1)*(i<cf?fac[i]:-1);
51     return re>=0;
52 }
53
54 inline int count(int x)
55 {
56     static int i, re;
57     x>=cf;
58     for (re=0; x>=1)
59         re+=(x&1);
60     return re;
61 }
62
63 int main()
64 {
65     while (scanf("%d", &n)!=EOF)
66     {
67         memset(s, 0, sizeof s);
68         memset(d, 0x3f, sizeof d);
69         memset(dp, 0x3f, sizeof dp);
70         ans=cnt=cf=cs=0;
71         memset(edge, 0, sizeof edge);

```



```

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

] + 1 + n);
cnt = cst = 0;
for (i=1; i<nn; ++i)
    if (check(i))
    {
        for (j=(i-1)&i; j; j=(j-1)&i)
            if (check(j) && check(i^j))
                dp[i] = std::min(dp[i], dp[j
                    ] + dp[i^j]);
        k = count(i);
        if (dp[i] != inf && (k > cnt || (k ==
            cnt && dp[i] < cst)))
        {
            cnt = k;
            cst = dp[i];
        }
    }
    printf("%d_%d\n", ans + cnt, cst);
}
return 0;
}

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[
12     MAXE], cnt;
13 #define v to[i]
14 inline void adde(int a, int b, int c, int d)
15 {
16     nxt[++cnt] = edge[a];
17     edge[a] = cnt;
18     to[cnt] = b;
19     cap[cnt] = c;
20     cst[cnt] = d;
21 }
22 inline void add(int a, int b, int c, int d)
23 { adde(a, b, c, d); adde(b, a, 0, -d); }
24
25 int dist[MAXX], pre[MAXX];
26 int source, sink;
27 std::queue<int> q;
28 bool in[MAXX];
29
30 inline bool go()
31 {
32     static int now, i;
33     memset(dist, 0x3f, sizeof dist);
34     dist[source] = 0;
35     pre[source] = -1;
36     q.push(source);
37     in[source] = true;
38     while (!q.empty())
39     {
40         in[now = q.front()] = false;
41         q.pop();
42         for (i = edge[now]; i != -1; i = nxt[i])
43             if (cap[i] && dist[v] > dist[now] + cst[i
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

45         pre[v]=i;
46         if(!in[v])
47         {
48             q.push(v);
49             in[v]=true;
50         }
51     }
52 }
53 return dist[sink]!=inf;
54 }
55
56 inline int mcmf(int &flow)
57 {
58     static int ans,i;
59     flow=ans=0;
60     while(go())
61     {
62         static int min;
63         min=inf;
64         for(i=pre[sink];i!=-1;i=pre[to[i^1]])
65             min=std::min(min,cap[i]);
66         flow+=min;
67         ans+=min*dist[sink];
68         for(i=pre[sink];i!=-1;i=pre[to[i^1]])
69         {
70             cap[i]-=min;
71             cap[i^1]+=min;
72         }
73     }
74     return ans;
75 }

```

4.24 Second-best MST

```

1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4
5 #define MAXN 511
6 #define MAXM 250011
7 #define v to[i]
8
9 int set[MAXN];
10 int find(int a)
11 {
12     return set[a]?set[a]=find(set[a]):a;
13 }
14
15 int n,m,i,j,k,ans;
16
17 struct edge
18 {
19     int a,b,c;
20     bool in;
21     bool operator<(const edge &i)const
22     {
23         return c<i.c;
24     }
25 }ed[MAXM];
26
27 int map[MAXN][MAXN];
28 bool done[MAXN];
29
30 int head[MAXN],to[MAXN<<1],nxt[MAXN<<1],wg[MAXN<<1],cnt;
31 inline void add(int a,int b,int c)
32 {
33     nxt[++cnt]=head[a];
34     head[a]=cnt;
35     to[cnt]=b;

```

```

36     wg[cnt]=c;
37 }
38
39 void dfs(const int now,const int fa)
40 {
41     done[now]=true;
42     for(int i(head[now]);i;i=nxt[i])
43         if(v!=fa)
44         {
45             for(int j(1);j<=n;++j)
46                 if(done[j])
47                     map[v][j]=map[j][v]=std::max(
48                         map[j][now],wg[i]);
49             dfs(v,now);
50         }
51 }
52
53 int main()
54 {
55     scanf("%d%d",&n,&m);
56     for(i=0;i<m;++i)
57         scanf("%d%d%d",&ed[i].a,&ed[i].b,&ed[i].c);
58     std::sort(ed,ed+m);
59     for(i=0;i<m;++i)
60         if(find(ed[i].a)!=find(ed[i].b))
61         {
62             j+=ed[i].c;
63             ++k;
64             set[find(ed[i].a)]=find(ed[i].b);
65             ed[i].in=true;
66             add(ed[i].a,ed[i].b,ed[i].c);
67             add(ed[i].b,ed[i].a,ed[i].c);
68         }
69     if(k+1!=n)
70         puts("Cost: -1\nCost: -1");
71     else
72     {
73         printf("Cost: %d\n",j);
74         if(m==n-1)
75         {
76             puts("Cost: -1");
77             return 0;
78         }
79         ans=0x3f3f3f3f;
80         memset(map,0x3f,sizeof map);
81         for(i=1;i<=n;++i)
82             map[i][i]=0;
83         dfs(1,0);
84         for(i=0;i<m;++i)
85             if(!ed[i].in)
86                 ans=std::min(ans,j+ed[i].c-map[ed[i].a][ed[i].b]);
87         printf("Cost: %d\n",ans);
88     }
89     return 0;
90 }

```

4.25 Spanning tree

```

1 Minimum Bottleneck Spanning Tree:
2 Krusal
3
4 All-pairs vertexes' Minimum Bottleneck Path:
5 DP in the Krusal's MST
6 O(n^2)*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
16 then enumerate all no-tree-edges to replace
17 the longest edge between two vertexes to get
18 a worse MST
19
20 Degree-constrained MST:
21 remove the vertex from the whole graph, then add
22 edges to increase degrees and connect
23 different connected components together ( O(
24 mlogm + n) with kruscal )
25
26 if we can't connect all connected components
27 together, there exists no any spanning tree
28 next step is add edges to root vertex greedily,
29 increase degrees, and decrease our answer ( O(
30 (k*n) )
31
32 need all vertexes' minimum bottleneck path to
33 root vertex
34
35 Minimum Ratio Spanning Tree:
36 Binary search
37
38 Manhattan MST:
39 combining line sweep with divide-and-conquer
40 algorithm
41
42 Minimum Steiner Tree:
43 the MST contain all k vertexes
44 bit-mask with dijkstra O( (1<<k)*( {dijkstra} ) )
45 then run a bit-mask DP( O( n*(1<<k) ) )
46
47 Count Spanning Trees:
48 TODO
49 Kirchhoff's theorem
50
51 k-best MST:
52 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
2 列匹配对象
3
4 while(!g.empty()) // 预备匹配队列
5 {
6     if(dfn[edge[g.front()].front()] == -1)
7         dfn[edge[g.front()].front()] = g.front();
8         // 如果目前还没尝试匹配过的对象没有被任何别的
9         对象占据
10
11     else
12     {
13         for(it=edge[edge[g.front()].front()].
14             begin(); it!=edge[edge[g.front()].
15             front()].end(); ++it)
16             if(*it==dfn[edge[g.front()].front()] ||
17                 *it==g.front()) //如果被匹配对象
18                 更喜欢正在被匹配的人或现在准备匹配的对象
19                 break;
20         if(*it==g.front()) //如果更喜欢新的
21         {
22             g.push_back(dfn[edge[g.front()].front()]
23                 ());
24             dfn[edge[g.front()].front()] = g.front();
25         }
26     }
27     else
28         g.push_back(g.front()); //否则放到队尾

```

重新等待匹配

```

}
edge[g.front()].pop_front(); //每组匹配最多只考
    虑一次
g.pop_front();
}

```

4.27 Stoer-Wagner Algorithm

```

#include<stdio>
#include<cstring>

const int maxn=510;

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

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

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

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

int main()
{
    int i,j,k,m;
    while (scanf("%d%d",&n,&m)!=EOF)
    {
        memset(map,0,sizeof(map));
        while (m--)
        {
            scanf("%d%d%d",&i,&j,&k);
            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
7         now).begin(); it != edge[now].end(); ++it)
8         if(dfn[*it] == -1)
9         {
10             dfs(*it);
11             low[now] = std::min(low[now], low[*it]);
12         }
13         else
14             if(sc[*it] == -1)
15                 low[now] = std::min(low[now], dfn[*
16                     it]);
17     if(dfn[now] == low[now])
18     {
19         while(sc[now] == -1)
20         {
21             sc[st.top()] = p;
22             st.pop();
23         }
24         ++p;
25     }
26 }

```

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
15     [MAXE], cnt;
16
17 inline void adde(int a, int b, int c, int k)
18 {
19     nxt[cnt] = edge[a];
20     edge[a] = cnt;
21     to[cnt] = b;
22     cap[cnt] = c;
23     cst[cnt] = k;
24     ++cnt;
25 }

```

```

25
26 inline void add(int a, int b, int c, int k)
27 {
28     adde(a, b, c, k);
29     adde(b, a, 0, -k);
30 }
31
32 int n, mf, cost, pil;
33 int source, sink;
34 bool done[MAXN];
35
36 int aug(int now, int maxcap)
37 {
38     if(now == sink)
39     {
40         mf += maxcap;
41         cost += maxcap * pil;
42         return maxcap;
43     }
44     done[now] = true;
45     int l = maxcap;
46     for(int i(edge[now]); i != -1; i = nxt[i])
47         if(cap[i] && !cst[i] && !done[to[i]])
48         {
49             int d(aug(to[i], std::min(l, cap[i])));
50             cap[i] -= d;
51             cap[i^1] += d;
52             l -= d;
53             if(!l)
54                 return maxcap;
55         }
56     return maxcap - l;
57 }
58
59 inline bool label()
60 {
61     static int d, i, j;
62     d = inf;
63     for(i = 1; i <= n; ++i)
64         if(done[i])
65             for(j = edge[i]; j != -1; j = nxt[j])
66                 if(cap[j] && !done[to[j]] && cst[
67                     j] < d)
68                     d = cst[j];
69
70     if(d == inf)
71         return false;
72     for(i = 1; i <= n; ++i)
73         if(done[i])
74             for(j = edge[i]; j != -1; j = nxt[j])
75                 {
76                     cst[j] -= d;
77                     cst[j^1] += d;
78                 }
79     pil += d;
80     return true;
81 }
82 /* primal-dual approach
83 static int d[MAXN], i, j;
84 static std::deque<int> q;
85 memset(d, 0x3f, sizeof d);
86 d[sink] = 0;
87 q.push_back(sink);
88 while(!q.empty())
89 {
90     static int dt, now;
91     now = q.front();
92     q.pop_front();
93     for(i = edge[now]; i != -1; i = nxt[i])
94         if(cap[i^1] && (dt = d[now] - cst[i]) < d[
95             to[i]])
96             if((d[to[i]] = dt) <= d[q.empty() ? 0 : q

```

```

93         .front())
94         q.push_front(to[i]);
95     else
96         q.push_back(to[i]);
97 }
98 for (i=1; i<=n; ++i)
99     for (j=edge[i]; j!=-1; j=nxt[j])
100         cst[j]+=d[to[j]]-d[i];
101 pi1+=d[source];
102 return d[source]!=inf;
103 */
104 }
105 int m,i,j,k;
106 typedef std::pair<int,int> pii;
107 std::vector<pii> M(MAXN), H(MAXN);
108
109 int main()
110 {
111     while (scanf("%d%d",&n,&m),(n||m))
112     {
113         M.resize(0);
114         H.resize(0);
115         for (i=0; i<n; ++i)
116         {
117             scanf("%s",buf);
118             for (j=0; j<m; ++j)
119                 if (buf[j]=='m')
120                     M.push_back(pii(i,j));
121             else
122                 if (buf[j]=='H')
123                     H.push_back(pii(i,j));
124         }
125         n=M.size()+H.size();
126         source=++n;
127         sink=++n;
128         memset(edge,-1,sizeof edge);
129         cnt=0;
130         for (i=0; i<M.size(); ++i)
131             for (j=0; j<H.size(); ++j)
132                 add(i+1,j+1+M.size(),1,abs(M[i].first-H[j].first)+abs(M[i].second-H[j].second));
133         for (i=0; i<M.size(); ++i)
134             add(source,i+1,1,0);
135         for (i=0; i<H.size(); ++i)
136             add(i+1+M.size(),sink,1,0);
137         mf=cost=pi1=0;
138         do
139             do
140                 memset(done,0,sizeof done);
141                 while (aug(source,inf));
142             while (label());
143             /* primal-dual approach
144             while (label())
145                 do
146                     memset(done,0,sizeof done);
147                     while (aug(source,inf));
148             */
149             printf("%d\n",cost);
150         }
151         return 0;
152     }

```

5 math

5.1 cantor

```
1 const int PermSize = 12;
```

```

2 int fac[PermSize] = {1, 1, 2, 6, 24, 120, 720,
3     5040, 40320, 362880, 3628800, 39916800};
4 inline int Cantor(int a[])
5 {
6     int i, j, cnt;
7     int res = 0;
8     for (i = 0; i < PermSize; ++i)
9     {
10         cnt = 0;
11         for (j = i + 1; j < PermSize; ++j)
12             if (a[i] > a[j])
13                 ++cnt;
14         res = res + cnt * fac[PermSize - i - 1];
15     }
16     return res;
17 }
18
19 bool h[13];
20
21 inline void UnCantor(int x, int res[])
22 {
23     int i, j, l, t;
24     for (i = 1; i <= 12; i++)
25         h[i] = false;
26     for (i = 1; i <= 12; i++)
27     {
28         t = x / fac[12 - i];
29         x -= t * fac[12 - i];
30         for (j = 1, l = 0; l <= t; j++)
31             if (!h[j])
32                 l++;
33         j--;
34         h[j] = true;
35         res[i - 1] = j;
36     }
37 }

```

5.2 Discrete logarithms - BSGS

```

1 //The running time of BSGS and the space
2 //complexity is  $O(\sqrt{n})$ 
3 //Pollard's rho algorithm for logarithms' running
4 //time is approximately  $O(\sqrt{p})$  where  $p$  is
5 //n's largest prime factor.
6
7 #include<cstdio>
8 #include<cmath>
9 #include<cstring>
10
11 struct Hash // std::map is bad. clear()时会付出巨大的
12     代价
13 {
14     static const int mod=100003; // prime is good
15     static const int MAXX=47111; // bigger than
16         sqrt(c)
17     int hd[mod], nxt[MAXX], cnt;
18     long long v[MAXX], k[MAXX]; //  $a^k \equiv v \pmod{c}$ 
19     inline void init()
20     {
21         memset(hd,0,sizeof hd);
22         cnt=0;
23     }
24     inline long long find(long long v)
25     {
26         static int now;
27         for (now=hd[v%mod]; now; now=nxt[now])
28             if (this->v[now]==v)
29                 return k[now];
30         return -111;
31     }
32 }

```

```

26 inline void insert(long long k, long long v)
27 {
28     if (find(v) != -1)
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);
47         x = y;
48         y = tmp - (a / b) * y;
49         return re;
50     }
51     x = 1;
52     y = 0;
53     return a;
54 }
55
56 inline long long bsgs(long long a, long long b,
    long long c) //  $a^x \equiv b \pmod{c}$ 
57 {
58     static long long x, y, d, g, m, am, k;
59     static int i, cnt;
60     a %= c;
61     b %= c;
62     x = 1; // 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 = 1;
70     cnt = 0;
71     while ((g = gcd(a, c)) != 1)
72     {
73         if (b % g)
74             return -1;
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 = 1;
83     hash.insert(0, am);
84     for (i = 1; i <= m; ++i)
85     {
86         am = am * a % c;
87         hash.insert(i, am);
88     }
89     for (i = 0; i <= m; ++i)
90     {
91         g = exgcd(d, c, x, y);
92         x = (x * b / g % c + c) % c;
93         k = hash.find(x);

```

```

        if (k != -1)
            return i * m + k + cnt;
        d = d * am % c;
    }
    return -1;
}

long long k, p, n;

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

```

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);

```

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);
7         x = y;
8         y = tmp - (a / b) * y;
9         return ret;
10    }
11    else
12    {
13        x = 1;
14        y = 0;
15        return a;
16    }
17 }

```

5.5 Fast Fourier Transform

```

1 #include <cstdio>
2 #include <cstring>
3 #include <complex>
4 #include <vector>
5 #include <algorithm>
6
7 #define MAXX 100111
8 #define MAXN (MAXX < 2)
9
10 int T;
11 int n, i, j, k;
12
13 typedef std::complex<long double> com;
14 std::vector<com> x(MAXN);
15 int a[MAXX];
16 long long pre[MAXN], cnt[MAXN];
17 long long ans;
18
19 inline void fft(std::vector<com> &y, int sign)

```

```

20 {
21     static int i,j,k,h;
22     static com u,t,w,wn;
23     for(i=1,j=y.size()/2;i+1<y.size();++i)
24     {
25         if(i<j)
26             std::swap(y[i],y[j]);
27         k=y.size()/2;
28         while(j>=k)
29         {
30             j-=k;
31             k/=2;
32         }
33         if(j<k)
34             j+=k;
35     }
36     for(h=2;h<=y.size();h<=1)
37     {
38         wn=com(cos(-sign*2*M_PI/h),sin(-sign*2*
39             M_PI/h));
40         for(j=0;j<y.size();j+=h)
41         {
42             w=com(1,0);
43             for(k=j;k<j+h/2;++k)
44             {
45                 u=y[k];
46                 t=w*y[k+h/2];
47                 y[k]=u+t;
48                 y[k+h/2]=u-t;
49                 w*=wn;
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].
55                     imag());
56     }
57 int main()
58 {
59     scanf("%d",&T);
60     while(T--)
61     {
62         memset(cnt,0,sizeof cnt);
63         scanf("%d",&n);
64         for(i=0;i<n;++i)
65         {
66             scanf("%d",a[i]);
67             ++cnt[a[i]];
68         }
69         std::sort(a,a+n);
70         k=a[n-1]+1;
71         for(j=1;j<(k<=1);j<=1); // size must be
72             such many
73         x.resize(0);
74         for(i=0;i<k;++i)
75             x.push_back(com(cnt[i],0));
76         x.insert(x.end(),j-k,com(0,0));
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
91             we need (x[i].real()+0.5f) or
92             nearbyint(x[i].real())
93         x.resize(2*a[n-1]); // result here
94     }
95     return 0;
}

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
4     秩
5 {
6     static int i,j,k,l;
7     for(j=i=0;j<n;++j) //第j行,列j
8     {
9         for(k=i;k<n;++k)
10             if(a[k][j])
11                 break;
12         if(k==n)
13             continue;
14         for(l=0;l<=n;++l)
15             std::swap(a[i][l],a[k][l]);
16         for(l=0;l<=n;++l)
17             if(l!=i && a[l][j])
18                 for(k=0;k<=n;++k)
19                     a[l][k]^=a[i][k];
20         ++i;
21     }
22     for(j=i;j<n;++j)
23         if(a[j][n])
24             return -1; //无解
25     return i;
26 }
27 /*
28 */
29 void dfs(int v)
30 {
31     if(v==n)
32     {
33         static int x[MAXX],ta[MAXX][MAXX];
34         static int tmp;
35         memcpy(x,ans,sizeof(x));
36         memcpy(ta,a,sizeof(ta));
37         for(i=1-1;i>=0;--i)
38         {
39             for(j=i+1;j<n;++j)
40                 ta[i][n]^=(x[j]&&ta[i][j]); //迭
41                 代消元求解
42             x[i]=ta[i][n];
43         }
44         for(tmp=i=0;i<n;++i)
45             if(x[i])
46                 ++tmp;
47         cnt=std::min(cnt,tmp);
48         return;
49     }
50     ans[v]=0;
51     dfs(v+1);
52     ans[v]=1;
53     dfs(v+1);
54 }

```

```

55 inline int ge(int a[N][N], int n)
56 {
57     static int i, j, k, l;
58     for (i=j=0; j<n; ++j)
59     {
60         for (k=i; k<n; ++k)
61             if (a[k][i])
62                 break;
63         if (k<n)
64         {
65             for (l=0; l<=n; ++l)
66                 std::swap(a[i][l], a[k][l]);
67             for (k=0; k<n; ++k)
68                 if (k!=i && a[k][i])
69                     for (l=0; l<=n; ++l)
70                         a[k][l]^=a[i][l];
71             ++i;
72         }
73         else //将不定元交换到后面去
74         {
75             l=n-1-j+i;
76             for (k=0; k<n; ++k)
77                 std::swap(a[k][l], a[k][i]);
78         }
79     }
80     if (i==n)
81     {
82         for (i=cnt=0; i<n; ++i)
83             if (a[i][n])
84                 ++cnt;
85         printf("%d\n", cnt);
86         continue;
87     }
88     for (j=i; j<n; ++j)
89         if (a[j][n])
90             break;
91     if (j<n)
92         puts("impossible");
93     else
94     {
95         memset(ans, 0, sizeof(ans));
96         cnt=111;
97         dfs(1=i);
98         printf("%d\n", cnt);
99     }
100 }
101
102 /*
103 */
104
105 inline void ge(int a[N][N], int m, int n) // m*n
106 {
107     static int i, j, k, l, b, c;
108     for (i=j=0; i<m && j<n; ++j)
109     {
110         for (k=i; k<m; ++k)
111             if (a[k][j])
112                 break;
113         if (k==m)
114             continue;
115         for (l=0; l<=n; ++l)
116             std::swap(a[i][l], a[k][l]);
117         for (k=0; k<m; ++k)
118             if (k!=i && a[k][j])
119             {
120                 b=a[k][j];
121                 c=a[i][j];
122                 for (l=0; l<=n; ++l)
123                     a[k][l]=((a[k][l]*b-a[i][l]*c)%7+7)%7;

```

```

124     }
125     ++i;
126 }
127 for (j=i; j<n; ++j)
128     if (a[j][n])
129         break;
130 if (j<m)
131 {
132     puts("Inconsistent data.");
133     return;
134 }
135 if (i<n)
136     puts("Multiple solutions.");
137 else
138 {
139     memset(ans, 0, sizeof(ans));
140     for (i=n-1; i>=0; --i)
141     {
142         k=a[i][n];
143         for (j=i+1; j<n; ++j)
144             k=((k-a[i][j]*ans[j])%7+7)%7;
145         while (k%a[i][i])
146             k+=7;
147         ans[i]=(k/a[i][i])%7;
148     }
149     for (i=0; i<n; ++i)
150         printf("%d%c", ans[i], i+1==n?' '\n': ' ');
151     }
152 }

```

5.7 inverse element

```

1 inline void getInv2(int x, int mod)
2 {
3     inv[1]=1;
4     for (int i=2; i<=x; ++i)
5         inv[i]=(mod-(mod/i)*inv[mod%i]%mod)%mod;
6 }
7
8 long long power(long long x, long long y, int mod)
9 {
10     long long ret=1;
11     for (long long a=x%mod; y; y>>=1, a=a*a%mod)
12         if (y&1)
13             ret=ret*a%mod;
14     return ret;
15 }
16
17 inline int getInv(int x, int mod) //为素数mod
18 {
19     return power(x, mod-2);
20 }

```

5.8 Linear programming

```

1 #include<stdio>
2 #include<cstring>
3 #include<cmath>
4 #include<algorithm>
5
6 #define MAXN 33
7 #define MAXM 33
8 #define eps 1e-8
9
10 double a[MAXN][MAXM], b[MAXN], c[MAXM];
11 double x[MAXM], d[MAXN][MAXM];
12 int ix[MAXN+MAXM];
13 double ans;
14 int n, m;

```



```

15 int i, j, k, r, s;
16 double D;
17
18 inline bool simplex()
19 {
20     r=n;
21     s=m++;
22     for (i=0; i<n+m; ++i)
23         ix[i]=i;
24     memset(d, 0, sizeof d);
25     for (i=0; i<n; ++i)
26     {
27         for (j=0; j+1<m; ++j)
28             d[i][j]=-a[i][j];
29         d[i][m-1]=1;
30         d[i][m]=b[i];
31         if (d[r][m]>d[i][m])
32             r=i;
33     }
34     for (j=0; j+1<m; ++j)
35         d[n][j]=c[j];
36     d[n+1][m-1]=-1;
37     while(true)
38     {
39         if (r<n)
40         {
41             std::swap(ix[s], ix[r+m]);
42             d[r][s]=1./d[r][s];
43             for (j=0; j<m; ++j)
44                 if (j!=s)
45                     d[r][j]*=-d[r][s];
46             for (i=0; i<n+1; ++i)
47                 if (i!=r)
48                 {
49                     for (j=0; j<m; ++j)
50                         if (j!=s)
51                             d[i][j]+=d[r][j]*d[i][s];
52                     d[i][s]*=d[r][s];
53                 }
54             }
55         r=-1;
56         s=-1;
57         for (j=0; j<m; ++j)
58             if ((s<0 || ix[s]>ix[j]) && (d[n+1][j]>eps || (d[n+1][j]>-eps && d[n][j]>eps)))
59                 s=j;
60         if (s<0)
61             break;
62         for (i=0; i<n; ++i)
63             if (d[i][s]<-eps && (r<0 || (D=(d[r][m]-d[i][m])/d[r][s]-d[i][m]/d[i][s])<-eps || (D<eps && ix[r+m]>ix[i+m])))
64                 r=i;
65         if (r<0)
66             return false;
67     }
68     if (d[n+1][m]<-eps)
69         return false;
70     for (i=m; i<n+m; ++i)
71         if (ix[i]+1<m)
72             x[ix[i]]=d[i-m][m]; // answer
73     ans=d[n][m]; // maxium value
74     return true;
75 }
76
77 int main()
78 {
79     while (scanf("%d%d", &m, &n) != EOF)

```

```

80     {
81         for (i=0; i<m; ++i)
82             scanf("%lf", c+i); // max{ sum{c[i]*x[i]} }
83         for (i=0; i<n; ++i)
84         {
85             for (j=0; j<m; ++j)
86                 scanf("%lf", a[i+j]); // sum{ a[i+j]*x[j] } <= b
87             scanf("%lf", b+i);
88             b[i]*=n;
89         }
90         simplex();
91         printf("Nasa can spend %.0lf taka.\n", ceil(ans));
92     }
93     return 0;
94 }

```

5.9 Lucas' theorem(2)

```

1 #include<cstdio>
2 #include<cstring>
3 #include<iostream>
4
5 int mod;
6 long long num[100000];
7 int ni[100], mi[100];
8 int len;
9
10 void init(int p)
11 {
12     mod=p;
13     num[0]=1;
14     for (int i=1; i<p; i++)
15         num[i]=i*num[i-1]%p;
16 }
17
18 void get(int n, int ni[], int p)
19 {
20     for (int i=0; i<100; i++)
21         ni[i]=0;
22     int tlen=0;
23     while (n!=0)
24     {
25         ni[tlen++] = n%p;
26         n /= p;
27     }
28     len = tlen;
29 }
30
31 long long power(long long x, long long y)
32 {
33     long long ret=1;
34     for (long long a=x%mod; y; y>>=1, a=a*a%mod)
35         if (y&1)
36             ret=ret*a%mod;
37     return ret;
38 }
39
40 long long getInv(long long x) // mod 为素数
41 {
42     return power(x, mod-2);
43 }
44
45 long long calc(int n, int m, int p) // C(n, m)%p
46 {
47     init(p);
48     long long ans=1;
49     for (; n && m && ans; n/=p, m/=p)

```

```

50     {
51         if (n%p>=m%p)
52             ans = ans*num[n%p]%p *getInv(num[n%p-18
53             ]%p)%p *getInv(num[n%p-m%p])%p;
54         else
55             ans=0;
56     }
57     return ans;
58 }
59 int main()
60 {
61     int t;
62     scanf("%d",&t);
63     while (t--)
64     {
65         int n,m,p;
66         scanf("%d%d%d",&n,&m,&p);
67         printf("%lld\n",calc(n+m,m,p));
68     }
69     return 0;
70 }

```

5.10 Lucas' theorem

```

1  #include <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
60 long p)
61 {
62     long long ans=1;
63     while(m && n && ans)
64     {
65         ans*=(CmodP(n%p,m%p,p));
66         ans=ans%p;
67         n=n/p;
68         m=m/p;
69     }
70     return ans;
71 }
72 int main()
73 {
74     long long n,k,p,ans;
75     int cas=0;
76     while(scanf("%I64d%I64d%I64d",&n,&k,&p)!=EOF)
77     {
78         if(k>n-k)
79             k=n-k;
80         ans=Lucas(n+1,k,p)+n-k;
81         printf("Case_#%d: %I64d\n",++cas,ans%p);
82     }
83     return 0;

```

5.11 Matrix

```

1  struct Matrix
2  {
3      const int N(52);
4      int a[N][N];
5      inline Matrix operator*(const Matrix &b)const
6      {
7          static Matrix res;
8          static int i,j,k;
9          for(i=0;i<N;++i)
10             for(j=0;j<N;++j)
11             {
12                 res.a[i][j]=0;
13                 for(k=0;k<N;++k)
14                     res.a[i][j]+=a[i][k]*b.a[k][j];
15             }
16         return res;
17     }
18     inline Matrix operator^(int y)const
19     {
20         static Matrix res,x;
21         static int i,j;
22         for(i=0;i<N;++i)
23         {
24             for(j=0;j<N;++j)
25             {
26                 res.a[i][j]=0;
27                 x.a[i][j]=a[i][j];

```

```

28     }
29     res.a[i][i]=1;
30 }
31 for (;y;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,
  const unsigned long long &n)
2 {
3     unsigned long long exp(a%n),tmp(0);
4     while(b)
5     {
6         if(b&1)
7         {
8             tmp+=exp;
9             if(tmp>n)
10                 tmp-=n;
11         }
12         exp<<=1;
13         if(exp>n)
14             exp-=n;
15         b>>=1;
16     }
17     return tmp;
18 }
19
20 inline unsigned long long exp_mod(unsigned long
  long a,unsigned long long b,const 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={∞a1,∞a2,...∞ak}
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
11     inclusion-exclusion principle.
12
13 MS T={3 a,4 b,5 c}
14 MS T* = {∞a,∞b,∞c}
15 A1 = {(T10)*|count(a) > 3} // (6)
16 A2 = {(T10)*|count(b) > 4} // (5)
17 A3 = {(T10)*|count(c) > 5} // (4)
18
19 C(T,10) = C(T*,10) - (|A1| + |A2| + |A3|) + (|A1 A2| + |A1
  A3| + |A2 A3|) - |A1 A2 A3|
20
21 C(10,12) C(1,3) C
  (0,2) 0 0
22 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
11 x1 n x y1
12 y1 x1
13
14 k-th solution is {matrix}k
15 */
16
17 import java.util.*;
18 import java.math.*;
19
20 public class Main
21 {
22     static BigInteger p,q,p1,p2,p3,q1,q2,q3,a1,a2
23     ,a0,h1,h2,g1,g2,n0;
24     static int n,t;
25     static void solve()
26     {
27         p2=BigInteger.ONE;

```

```

24 p1=BigInteger.ZERO;
25 q2=BigInteger.ZERO;
26 q1=BigInteger.ONE;
27 a0=a1=BigInteger.valueOf((long)Math.sqrt(
    n));
28 g1=BigInteger.ZERO;
29 h1=BigInteger.ONE;
30 n0=BigInteger.valueOf(n);
31 while(true)
32 {
33     g2=a1.multiply(h1).subtract(g1);
34     h2=(n0.subtract(g2.multiply(g2))).
        divide(h1);
35     a2=(g2.add(a0)).divide(h2);
36     p=p2.multiply(a1).add(p1);
37     q=q2.multiply(a1).add(q1);
38     if(p.multiply(p).subtract(n0.multiply(
        q.multiply(q))).equals(
        BigInteger.ONE))
39         return ;
40     a1=a2;
41     g1=g2;
42     h1=h2;
43     p1=p2;
44     p2=p;
45     q1=q2;
46     q2=q;
47 }
48 }
49 public static void main(String[] args)
50 {
51     Scanner in=new Scanner(System.in);
52     t=in.nextInt();
53     for(int i=0;i<t;++i)
54     {
55         n=in.nextInt();
56         solve();
57         System.out.println(p+" "+q);
58     }
59 }
60 }

```

5.15 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 inline unsigned long long multi_mod(const
    unsigned long long &a,unsigned long long b,
    const unsigned long long &n)
10 {
11     unsigned long long exp(a%n),tmp(0);
12     while(b)
13     {
14         if(b&1)
15         {
16             tmp+=exp;
17             if(tmp>n)
18                 tmp-=n;
19         }
20         exp<<=1;
21         if(exp>n)
22             exp-=n;
23         b>>=1;
24     }

```

```

25     return tmp;
26 }
27
28 inline unsigned long long exp_mod(unsigned long
    long a,unsigned long long b,const unsigned
    long long &c)
29 {
30     unsigned long long tmp(1);
31     while(b)
32     {
33         if(b&1)
34             tmp=multi_mod(tmp,a,c);
35         a=multi_mod(a,a,c);
36         b>>=1;
37     }
38     return tmp;
39 }
40
41 inline bool miller_rabbin(const unsigned long
    long &n,short T)
42 {
43     if(n==2)
44         return true;
45     if(n<2 || !(n&1))
46         return false;
47     unsigned long long a,u(n-1),x,y;
48     short t(0),i;
49     while(!(u&1))
50     {
51         ++t;
52         u>>=1;
53     }
54     while(T--)
55     {
56         a=rand()%(n-1)+1;
57         x=exp_mod(a,u,n);
58         for(i=0;i<t;++i)
59         {
60             y=multi_mod(x,x,n);
61             if(y==1 && x!=1 && x!=n-1)
62                 return false;
63             x=y;
64         }
65         if(y!=1)
66             return false;
67     }
68     return true;
69 }
70
71 unsigned long long gcd(const unsigned long long &
    a,const unsigned long long &b)
72 {
73     return b?gcd(b,a%b):a;
74 }
75
76 inline unsigned long long pollar_rho(const
    unsigned long long n,const unsigned long long
    &c)
77 {
78     unsigned long long x(rand()%(n-1)+1),y,d,i(1)
        ,k(2);
79     y=x;
80     while(true)
81     {
82         ++i;
83         x=(multi_mod(x,x,n)+c)%n;
84         d=gcd((x-y+n)%n,n);
85         if(d>1 && d<n)
86             return d;
87         if(x==y)

```

```

88         return n;
89     if (i==k)
90     {
91         k<<=1;
92         y=x;
93     }
94 }
95 }
96
97 void find(const unsigned long long &n,short c)
98 {
99     if(n==1)
100         return;
101     if(miller_rabbin(n,6))
102     {
103         fac.push_back(n);
104         return;
105     }
106     unsigned long long p(n);
107     short k(c);
108     while(p>=n)
109         p=pollar_rho(p,c--);
110     find(p,k);
111     find(n/p,k);
112 }
113
114 int main()
115 {
116     scanf("%hd",&T);
117     while(T-->0)
118     {
119         scanf("%llu",&a);
120         fac.clear();
121         find(a,120);
122         if(fac.size()==1)
123             puts("Prime");
124         else
125         {
126             fac.sort();
127             printf("%llu\n",fac.front());
128         }
129     }
130     return 0;
131 }

```

5.16 Prime

```

1 #include<vector>
2
3 std::vector<int>prm;
4 bool flag[MAXX];
5
6 int main()
7 {
8     prm.reserve(MAXX); // pi(x)=x/ln(x);
9     for(i=2;i<MAXX;++i)
10     {
11         if(!flag[i])
12             prm.push_back(i);
13         for(j=0;j<prm.size() && i*prm[j]<MAXX;+5j)
14         {
15             flag[i*prm[j]]=true;
16             if(i%prm[j]==0)
17                 break;
18         }
19     }
20     return 0;
21 }

```

5.17 Reduced Residue System

```

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

```

```

63
64 若  $\text{ord}(x) = \varphi(m)$ , 则  $x$  为  $m$  的一个原根
65 因此只需检查所有  $x^d \{d \text{ 为 } \varphi(m) \text{ 的约数}\}$  找到使  $x^d \equiv 1$ 
    (mod  $m$ ) 的所有  $d$ , 当且仅当这样的  $d$  只有一个, 并且为
     $\varphi(m)$  的时候,  $x$  是  $m$  的一个原根
66
67 当且仅当  $m = 1, 2, 4, p^n, 2 \times p^n$   $\{p \text{ 为奇质数}, n \text{ 为正整数}\}$  时,  $m$  存在
    在原根 // 应该是指存在对于完全剩余系的原根……?
68
69 当  $m$  存在原根时, 原根数目为  $\varphi(\varphi(m))$ 
70
71 求:
72 枚举每一个简化剩余系中的数  $i$ , 若对于  $i$  的每一个质因子
     $p[j], i^{\frac{\varphi(m)}{p[j]}} \not\equiv 1 \pmod{m}$ , 那么  $i$  为  $m$  的一个原根。也就是说
     $\text{ord}(i) = \varphi(m)$ 。
73 最小原根通常极小。
74
75 Carmichael function:
76
77  $\lambda(n)$  is defined as the smallest positive integer  $m$  such that
     $a^m \equiv 1 \pmod{n}$   $\{ \text{forall } a! \equiv 1 \ \&\& \ \gcd(a,n) = 1 \}$ 
78 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)
79 中所有  $x$  的  $\text{lcm}\{\text{ord}(x)\}$ 
80
81 if  $n = p[0]^{a[0]} \times p[1]^{a[1]} \times \dots \times p[m-1]^{a[m-1]}$ 
82 then  $\lambda(n) = \text{lcm}(\lambda(p[0]^{a[0]}), \lambda(p[1]^{a[1]}), \dots, \lambda(p[m-1]^{a[m-1]}))$ ;
83
84 if  $n = 2^c \times p[0]^{a[0]} \times p[1]^{a[1]} \times \dots \times p[m-1]^{a[m-1]}$ 
85 then  $\lambda(n) = \text{lcm}(2^c, \varphi(p[0]^{a[0]}), \varphi(p[1]^{a[1]}), \dots, \varphi(p[m-1]^{a[m-1]}))$ ;
86  $\{ c=0 \text{ if } a<2; c=1 \text{ if } a==2; c=a-2 \text{ if } a>3; \}$ 
87
88 Carmichael's theorem:
89 if  $\gcd(a,n) = 1$ 
90 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(l, mid)-simp(mid, r)))/15 < eps)
19         return simp(l, r);
20     else
21         return rsimp(l, mid)+rsimp(mid, r);
22 }

```

5.19 System of linear congruences

```

1 // minimal val that for all (m,a) , val%m == a
2 #include<stdio>
3
4 #define MAXX 11

```

```

5
6 int T, t;
7 int m[MAXX], a[MAXX];
8 int n, i, j, k;
9 int x, y, c, d;
10 int lcm;
11
12 int exgcd(int a, int b, int &x, int &y)
13 {
14     if (b)
15     {
16         int re = exgcd(b, a%b, x, y); tmp(x);
17         x=y;
18         y=tmp-(a/b)*y;
19         return re;
20     }
21     x=1;
22     y=0;
23     return a;
24 }
25
26 int main()
27 {
28     scanf("%d", &T);
29     for (t=1; t<=T; ++t)
30     {
31         scanf("%d", &n);
32         lcm=1;
33         for (i=0; i<n; ++i)
34         {
35             scanf("%d", &m[i]);
36             lcm*=m[i]/exgcd(lcm, m[i], x, y);
37         }
38         for (i=0; i<n; ++i)
39             scanf("%d", &a[i]);
40         for (i=1; i<n; ++i)
41         {
42             c=a[i]-a[0];
43             d=exgcd(m[0], m[i], x, y);
44             if (c%d)
45                 break;
46             y=m[i]/d;
47             c/=d;
48             x=(x*c%y+y)%y;
49             a[0]+=m[0]*x;
50             m[0]*=y;
51         }
52         printf("Case_%d: %d\n", t, i<n?-1:(a[0]?a[0]:lcm));
53     }
54     return 0;
55 }

```

6 string

6.1 Aho-Corasick Algorithm

```

1 //trie graph
2 #include<cstring>
3 #include<queue>
4
5 #define MAX 1000111
6 #define N 26
7
8 int nxt[MAX][N], fal[MAX], cnt;
9 bool ed[MAX];
10 char buf[MAX];
11
12 inline void init(int a)
13 {

```

```

14  memset(nxt[a],0,sizeof(nxt[0]));
15  fal[a]=0;
16  ed[a]=false;
17  }
18
19  inline void insert()
20  {
21      static int i,p;
22      for(i=p=0;buf[i];++i)
23      {
24          if(!nxt[p][map[buf[i]]])
25              init(nxt[p][map[buf[i]]]=++cnt);
26          p=nxt[p][map[buf[i]]];
27      }
28      ed[p]=true;
29  }
30
31  inline void make()
32  {
33      static std::queue<int>q;
34      int i,now,p;
35      q.push(0);
36      while(!q.empty())
37      {
38          now=q.front();
39          q.pop();
40          for(i=0;i<N;++i)
41              if(nxt[now][i])
42              {
43                  q.push(p=nxt[now][i]);
44                  if(now)
45                      fal[p]=nxt[fal[now]][i];
46                  ed[p]|=ed[fal[p]];
47              }
48              else
49                  nxt[now][i]=nxt[fal[now]][i];

```

使用本身的 trie 存串的时候注意 nxt 会被重载

```

50  }
51  }
52
53  // normal version
54
55  #define N 128
56
57  char buf[MAXX];
58  int cnt[1111];
59
60  struct node
61  {
62      node *fal,*nxt[N];
63      int idx;
64      node() { memset(this,0,sizeof node); }
65  }*rt;
66  std::queue<node*>q;
67
68  void free(node *p)
69  {
70      for(int i(0);i<N;++i)
71          if(p->nxt[i])
72              free(p->nxt[i]);
73      delete p;
74  }
75
76  inline void add(char *s,int idx)
77  {
78      static node *p;
79      for(p=rt;*s;++s)
80      {
81          if(!p->nxt[*s])

```

```

82          p->nxt[*s]=new node();
83          p=p->nxt[*s];
84      }
85      p->idx=idx;
86  }
87
88  inline void make()
89  {
90      Q.push(rt);
91      static node *p,*q;
92      static int i;
93      while(!Q.empty())
94      {
95          p=Q.front();
96          Q.pop();
97          for(i=0;i<N;++i)
98              if(p->nxt[i])
99              {
100                  q=p->fal;
101                  while(q)
102                  {
103                      if(q->nxt[i])
104                      {
105                          p->nxt[i]->fal=q->nxt[i];
106                          break;
107                      }
108                      q=q->fal;
109                  }
110                  if(!q)
111                      p->nxt[i]->fal=rt;
112                  Q.push(p->nxt[i]);
113              }
114          }
115      }
116
117  inline void match(const char *s)
118  {
119      static node *p,*q;
120      for(p=rt;*s;++s)
121      {
122          while(p!=rt && !p->nxt[*s])
123              p=p->fal;
124          p=p->nxt[*s];
125          if(!p)
126              p=rt;
127          for(q=p;q!=rt && q->idx;q=q->fal) // why
128              // q->idx ? looks like not necessary at
129              // all, I delete it in an other solution
130              ++cnt[q->idx];
131      }
132
133      //可以考虑 dfs 一下，拉直 fal 指针来跳过无效的匹配
134      //在线调整关键字存在性的时候，可以考虑欧拉序压扁之后使用
135      //BIT 或者线段树进行区间修改
136      //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
137      //录每个节点被覆盖的次数，然后沿着 fal 指针构成的 DAG 往
138      //上传递覆盖次数

```

6.2 Gusfield's Z Algorithm

```

1  inline void make(int *z,char *buf)
2  {
3      int i,j,l,r;
4      l=0;
5      r=1;
6      z[0]=strlen(buf);
7      for(i=1;i<z[0];++i)
8          if(r<=i || z[i-l]>=r-i)
9          {

```

```

10     j=std::max(i,r);
11     while(j<z[0] && buf[j]==buf[j-i])
12         ++j;
13     z[i]=j-i;
14     if(i<j)
15     {
16         l=i;
17         r=j;
18     }
19 }
20 else
21     z[i]=z[i-1];
22 }
23
24 for(i=1;i<len && i+z[i]<len;++i); //i= 可能最小循环
    节长度

```

6.3 Manacher's Algorithm

```

1 inline int match(const int a,const int b,const
    std::vector<int> &str)
2 {
3     static int i;
4     i=0;
5     while(a-i>=0 && b+i<str.size() && str[a-i]==
        str[b+i]) //注意是 i 不是 1, 打错过很多次
        ++i;
6     return i;
7 }
8
9
10 inline void go(int *z,const std::vector<int> &str)
11 {
12     static int c,l,r,i,ii,n;
13     z[0]=1;
14     c=l=r=0;
15     for(i=1;i<str.size();++i)
16     {
17         ii=(l<<1)-i;
18         n=r+l-i;
19
20         if(i>r)
21         {
22             z[i]=match(i,i,str);
23             l=i;
24             r=i+z[i]-1;
25         }
26         else
27             if(z[ii]==n)
28             {
29                 z[i]=n+match(i-n,i+n,str);
30                 l=i;
31                 r=i+z[i]-1;
32             }
33             else
34                 z[i]=std::min(z[ii],n);
35         if(z[i]>z[c])
36             c=i;
37     }
38 }
39
40 inline bool check(int *z,int a,int b) //检查子串
    [a,b] 是否回文
41 {
42     a=a*2-1;
43     b=b*2-1;
44     int m=(a+b)/2;
45     return z[m]>=b-m+1;
46 }

```

6.4 Morris-Pratt Algorithm

```

1 inline void make(char *buf,int *fal)
2 {
3     static int i,j;
4     fal[0]=-1;
5     for(i=1,j=-1;buf[i];++i)
6     {
7         while(j>=0 && buf[j+1]!=buf[i])
8             j=fal[j];
9         if(buf[j+1]==buf[i])
10             ++j;
11         fal[i]=j;
12     }
13 }
14
15
16 inline int match(char *p,char *t,int* fal)
17 {
18     static int i,j,re;
19     re=0;
20     for(i=0,j=-1;t[i];++i)
21     {
22         while(j>=0 && p[j+1]!=t[i])
23             j=fal[j];
24         if(p[j+1]==t[i])
25             ++j;
26         if(!p[j+1])
27         {
28             ++re;
29             j=fal[j];
30         }
31     }
32     return re;
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<stdio>
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]
14         && str[a+2]==str[b+2];
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] &&
20             c12(str, 1, a+1, b+1);
21     else
22         return str[a]<str[b] || str[a]==str[b] &&
23             wv[a+1]<wv[b+1];
24 }
25 inline void sort(int *str, int *a, int *b, const int &n, const int &m)
26 {
27     memset(ws, 0, sizeof(ws));
28     int i;
29     for(i=0; i<n; ++i)
30         ++ws[wv[i]=str[a[i]]];
31     for(i=1; i<=m; ++i)
32         ws[i]+=ws[i-1];
33     for(i=n-1; i>=0; --i)
34         b[--ws[wv[i]]]=a[i];
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, l;
40     str[n]=str[n+1]=0;
41     for(i=0; i<n; ++i)
42         if(i%3)
43             wa[tbc++]=i;
44     sort(str+2, wa, wb, tbc, m);
45     sort(str+1, wb, wa, tbc, m);
46     sort(str, wa, wb, tbc, m);
47     for(i=j=1, strn[F(wb[0])]=0; i<tbc; ++i)
48         strn[F(wb[i])]=c0(str, wb[i-1], wb[i])?j-1:j;
49     if(j<tbc)
50         dc3(strn, san, tbc, j);
51     else
52         for(i=0; i<tbc; ++i)
53             san[strn[i]]=i;
54     for(i=0; i<tbc; ++i)
55         if(san[i]<tb)
56             wb[ta++]=san[i]*3;
57     if(n%3==1)
58         wb[ta++]=n-1;
59     sort(str, wb, wa, ta, m);
60     for(i=0; i<tbc; ++i)
61         wv[wb[i]=G(san[i])]=i;
62     for(i=j=k=0; i<ta && j<tbc;)
63         sa[k++]=c12(str, wb[j]%3, wa[i], wb[j])?wa[i++]:wb[j++];
64     while(i<ta)
65         sa[k++]=wa[i++];
66     while(j<tbc)
67         sa[k++]=wb[j++];
68 }
69
70 int rk[MAXX], lcpa[MAXX], sa[MAXX*3];
71 int str[MAXX*3]; //必须 int
72
73 int main()
74 {
75     scanf("%d%d", &n, &j);
76     for(i=0; i<n; ++i)
77     {
78         scanf("%d", &k);
79         num[i]=k-j+100;
80         j=k;
81     }
82     num[n]=0;
83
84     dc3(num, sa, n+1, 191); //191: str 中取值范围, 桶排序
85
86     for(i=1; i<=n; ++i) // rank 数组
87         rk[sa[i]]=i;
88     for(i=k=0; i<n; ++i) // lcp 数组
89         if(!rk[i])
90             lcpa[0]=0;
91         else
92         {
93             j=sa[rk[i]-1];
94             if(k>0)
95                 --k;
96             while(num[i+k]==num[j+k])
97                 ++k;
98             lcpa[rk[i]]=k;
99         }
100
101     for(i=1; i<=n; ++i)
102         sptb[0][i]=i;
103     for(i=1; i<=lg[n]; ++i) //sparse table RMQ
104     {
105         k=n+1-(1<<i);
106         for(j=1; j<=k; ++j)
107         {
108             a=sptb[i-1][j];
109             b=sptb[i-1][j+(1<<(i-1))];
110             sptb[i][j]=lcpa[a]<lcpa[b]?a:b;
111         }
112     }
113
114     inline int ask(int l, int r)
115     {
116         a=lg[r-l+1];
117         r--=(1<<a)-1;
118         l=sptb[a][l];
119         r=sptb[a][r];
120         return lcpa[l]<lcpa[r]?l:r;
121     }
122
123     inline int lcp(int l, int r) // 字符串上 [l,r] 区间的
124     rmq
125     {
126         l=rk[l];
127         r=rk[r];
128         if(l>r)
129             std::swap(l, r);
130         return lcpa[ask(l+1, r)];
131     }
132 }
133
134 6.7 Suffix Array - Prefix-doubling Algorithm
135
136 int wx[maxn], wy[maxn], *x, *y, wss[maxn], wv[maxn];
137

```

```

3 bool cmp(int *r,int n,int a,int b,int l)      17
4 {                                              18
5     return a+l<n && b+l<n && r[a]==r[b]&&r[a+l]==r[b+l]; 20
6 }                                              21
7 void da(int str[],int sa[],int rank[],int height[],int n,int m) 22
8 {                                              23
9     int *s = str;                            24
10    int *x=wx,*y=wy,*t,p;                    25
11    int i,j;                                  26
12    for(i=0; i<m; i++)                        27
13        wss[i]=0;                             28
14    for(i=0; i<n; i++)                        29
15        wss[x[i]=s[i]]++;                    30
16    for(i=1; i<m; i++)                        31
17        wss[i]+=wss[i-1];                    32
18    for(i=n-1; i>=0; i--)                    33
19        sa[--wss[x[i]]]=i;                  34
20    for(j=1,p=1; p<n && j<n; j*=2,m=p)        35
21    {                                          36
22        for(i=n-j,p=0; i<n; i++)            37
23            y[p++]=i;                        38
24        for(i=0; i<n; i++)                    39
25            if(sa[i]-j>=0)                   40
26                y[p++]=sa[i]-j;             41
27        for(i=0; i<n; i++)                    42
28            wv[i]=x[y[i]];                   43
29        for(i=0; i<m; i++)                    44
30            wss[i]=0;                         45
31        for(i=0; i<n; i++)                    46
32            wss[wv[i]]++;                     47
33        for(i=1; i<m; i++)                    48
34            wss[i]+=wss[i-1];                 49
35        for(i=n-1; i>=0; i--)                50
36            sa[--wss[wv[i]]]=y[i];           51
37        for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0; i<n; i++) 52
38            x[sa[i]]=cmp(y,n,sa[i-1],sa[i],j)?p+1:p-1; 53
39    }                                          54
40    for(int i=0; i<n; i++)                    55
41        rank[sa[i]]=i;                       56
42    for(int i=0,j=0,k=0; i<n; height[rank[i++]]<=0) 57
43    {                                          58
44        if(rank[i]>0)                         59
45            for(k?k--:0,j=sa[rank[i]-1]; i+k<n && j+k<n && str[i+k]==str[j+k]; ++k); 60
46    }                                          61
47 }                                              62
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82

```

6.8 Suffix Automaton

```

1  /*
2  length(s) ∈ [ min(s), max(s) ] = [ val[fal[s]]+1, val[s] ]
3  */
4  #define MAXX 90111
5  #define MAXN (MAXX<<1)
6
7  int fal[MAXN],nxt[MAXN][26],val[MAXN],cnt,rt,last;
8
9  inline int neww(int v=0)
10 {
11     val[++cnt]=v;
12     fal[cnt]=0;
13     memset(nxt[cnt],0,sizeof nxt[0]);
14     return cnt;
15 }
16
17 inline void add(int w)
18 {
19     static int p,np,q,nq;
20     p=last;
21     np=neww(val[p]+1);
22     while(p && !nxt[p][w])
23     {
24         nxt[p][w]=np;
25         p=fal[p];
26     }
27     if(!p)
28         fal[np]=rt;
29     else
30     {
31         q=nxt[p][w];
32         if(val[p]+1==val[q])
33             fal[np]=q;
34         else
35         {
36             nq=neww(val[p]+1);
37             memcpy(nxt[nq],nxt[q],sizeof nxt[0]);
38             fal[nq]=fal[q];
39
40             fal[q]=fal[np]=nq;
41             while(p && nxt[p][w]==q)
42             {
43                 nxt[p][w]=nq;
44                 p=fal[p];
45             }
46         }
47     }
48     last=np;
49 }
50
51 int v[MAXN],the[MAXN];
52
53 inline void make(char *str)
54 {
55     cnt=0;
56     rt=last=neww();
57     static int i,len,now;
58     for(i=0;str[i];++i)
59         add(str[i]-'a');
60     len=i;
61     memset(v,0,sizeof v);
62     for(i=1;i<=cnt;++i)
63         ++v[val[i]];
64     for(i=1;i<=len;++i)
65         v[i]+=v[i-1];
66     for(i=1;i<=cnt;++i)
67         the[v[val[i]]--]=i;
68     for(i=cnt;i--i)
69     {
70         now=the[i];
71         // topsort already
72     }
73 }
74
75 /*
76 sizeof right(s):
77     init:
78         for all np:
79             count[np]=1;
80     process:
81         for all status s:
82             count[fal[s]]+=count[s];
83 */

```

7 search

7.1 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。

7.2 dlx - exact cover

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

```
52         {
53             k=node(u[ch[j]], ch[j], l[r], r);
54             ;
55             rh[k]=i;
56             ch[k]=ch[j];
57         }
58         ++sz[j];
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==r[hd])
91     {
92         ans.resize(k);
93         return true;
94     }
95     int s=inf, c;
96     int i, j;
97     for(i=r[hd]; i!=hd; i=r[i])
98         if(sz[i]<s)
99         {
100             s=sz[i];
101             c=i;
102         }
103     rm(c);
104     for(i=d[c]; i!=c; i=d[i])
105     {
106         ans[k]=rh[i];
107         for(j=r[i]; j!=i; j=r[j])
108             rm(ch[j]);
109         if(dlx(k+1))
110             return true;
111         for(j=l[i]; j!=i; j=l[j])
112             add(ch[j]);
113     }
114     add(c);
115     return false;
116 }
117
118 #include <cstdio>
119 #include <cstring>
```

```

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],
126     res[N], cntcol[N];
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rowx
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(" ");
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])
211             remove(col[j]);
212         if (DLX(deep + 1)) return true;
213         for (int j = l[i]; j != i; j = l[j])
214             resume(col[j]);
215     }
216     resume(tempc);
217     return false;
218 }
219 //插入矩阵中的节点"1"
220 inline void insert_node(int x, int y)
221 {
222     cntcol[y]++;
223     addnode(dcnt);
224     row[dcnt] = x;
225     col[dcnt] = y;
226     insert_col(y, dcnt);
227     if (h[x] == -1) h[x] = dcnt;
228     else insert_row(h[x], dcnt);
229 }
230 int main()
231 {
232     int n, m;
233     while (~scanf("%d%d", &n, &m))
234     {
235         dlx_init(m);
236         for (int i = 1; i <= n; ++i)
237         {
238             int k, x;
239             scanf("%d", &k);
240             while (k--)
241             {
242                 scanf("%d", &x);
243                 insert_node(i, x);
244             }
245             if (!DLX(0))
246                 puts("NO");
247         }
248         return 0;
249     }
250 }

```

7.3 dlx - repeat cover

```

1 #include<cstdio>
2 #include<cstring>
3 #include<algorithm>
4
5 #define MAXN 110

```

```

6 #define MAXM 1000000
7 #define INF 0x7FFFFFFF
8
9 using namespace std;
10
11 int G[MAXN][MAXN];
12 int L[MAXM], R[MAXM], U[MAXM], D[MAXM];
13 int size, ans, S[MAXM], H[MAXM], C[MAXM];
14 bool vis[MAXN * 100];
15 void Link(int r, int c)
16 {
17     U[size] = c;
18     D[size] = D[c];
19     U[D[c]] = size;
20     D[c] = size;
21     if (H[r] < 0)
22         H[r] = L[size] = R[size] = size;
23     else
24     {
25         L[size] = H[r];
26         R[size] = R[H[r]];
27         L[R[H[r]]] = size;
28         R[H[r]] = size;
29     }
30     S[c]++;
31     C[size++] = c;
32 }
33 void Remove(int c)
34 {
35     int i;
36     for (i = D[c]; i != c; i = D[i])
37     {
38         L[R[i]] = L[i];
39         R[L[i]] = R[i];
40     }
41 }
42 void Resume(int c)
43 {
44     int i;
45     for (i = D[c]; i != c; i = D[i])
46         L[R[i]] = R[L[i]] = i;
47 }
48 int A()
49 {
50     int i, j, k, res;
51     memset(vis, false, sizeof(vis));
52     for (res = 0, i = R[0]; i; i = R[i])
53     {
54         if (!vis[i])
55         {
56             res++;
57             for (j = D[i]; j != i; j = D[j])
58             {
59                 for (k = R[j]; k != j; k = R[k])
60                     vis[C[k]] = true;
61             }
62         }
63     }
64     return res;
65 }
66 void Dance(int now)
67 {
68     if (R[0] == 0)
69         ans = min(ans, now);
70     else if (now + A() < ans)
71     {
72         int i, j, temp, c;
73         for (temp = INF, i = R[0]; i; i = R[i])
74         {
75             if (temp > S[i])

```

```

76         {
77             temp = S[i];
78             c = i;
79         }
80     }
81     for (i = D[c]; i != c; i = D[i])
82     {
83         Remove(i);
84         for (j = R[i]; j != i; j = R[j])
85             Remove(j);
86         Dance(now + 1);
87         for (j = L[i]; j != i; j = L[j])
88             Resume(j);
89         Resume(i);
90     }
91 }
92 }
93 void Init(int m)
94 {
95     int i;
96     for (i = 0; i <= m; i++)
97     {
98         R[i] = i + 1;
99         L[i + 1] = i;
100        U[i] = D[i] = i;
101        S[i] = 0;
102    }
103    R[m] = 0;
104    size = m + 1;
105 }

```

7.4 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
19         mono *)m;
20     if (a->weig != b->weig)
21         return a->weig - b->weig;
22     else
23         return b->cost - a->cost;
24 }
25
26 bool comp(const struct mono a, const struct mono b)
27 {
28     if (a.weig != b.weig)
29         return a.weig < b.weig;
30     else
31         return b.cost < a.cost;
32 }
33
34 void dfs(short i, long long cost_n, long long
    carry_n, short last)

```

```

35     if (ans < cost_n)
36         ans = cost_n;
37     if (i == n || goods[i].weig > carry_n || cost_n + 11
        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("%hd", &T);
48     for (t = 1; t <= T; ++t)
49     {
50         scanf("%hd%lld", &n, &carry);
51         sumw = 0;
52         sumc = 0;
53         ans = 0;
54         for (i = 0; i < n; ++i)
55         {
56             scanf("%lld%lld", &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_%hd: %lld\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("%lld %lld\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_%hd: %lld\n", t, ans);
75     }
76     return 0;
77 }

```

8 dynamic programming

8.1 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 k
6             f[v] = max{f[v], f[v-c[i]]+w[i]}

```

8.2 LCIS

```

1 #include <cstdio>
2 #include <cstring>
3 #include <vector>
4
5 #define MAXX 1111
6
7 int T;
8 int n, m, p, i, j, k;

```

```

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

```

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
12                sign = 1 otherwise
13    // constructors
14    Bigint() {} // default constructor
15    Bigint( std::string b ) { (*this) = b; } //
16                constructor for std::string
17    // some helpful methods
18    int size() // returns number of digits
19    {
20        return a.size();
21    }
22    Bigint inverseSign() // changes the sign
23    {
24        sign *= -1;
25        return (*this);
26    }
27    Bigint normalize( int newSign ) // removes
28        leading 0, fixes sign
29    {
30        for( int i = a.size() - 1; i > 0 && a[i]
31            == '0'; i-- )
32            a.erase(a.begin() + i);
33        sign = ( a.size() == 1 && a[0] == '0' ) ?
34            1 : newSign;
35        return (*this);
36    }
37    // assignment operator
38    void operator = ( std::string b ) // assigns
39        a std::string to Bigint
40    {
41        a = b[0] == '-' ? b.substr(1) : b;
42        reverse( a.begin(), a.end() );
43        this->normalize( b[0] == '-' ? -1 : 1 );
44    }
45    // conditional operators
46    bool operator < ( const Bigint &b ) const //
47        less than operator
48    {
49        if( sign != b.sign )
50            return sign < b.sign;
51        if( a.size() != b.a.size() )
52            return sign == 1 ? a.size() < b.a.
53                size() : a.size() > b.a.size();
54        for( int i = a.size() - 1; i >= 0; i-- )
55            if( a[i] != b.a[i] )
56                return sign == 1 ? a[i] < b.a[i]
57                    : a[i] > b.a[i];
58        return false;
59    }
60    bool operator == ( const Bigint &b ) const
61        operator for equality
62    {
63        return a == b.a && sign == b.sign;
64    }
65
66    // mathematical operators
67    Bigint operator + ( Bigint b ) // addition
68        operator overloading
69    {
70        if( sign != b.sign )
71            return (*this) - b.inverseSign();
72        Bigint c;
73        for( int i = 0, carry = 0; i < a.size() || i
74            < b.size() || carry; i++ )
75        {
76            carry += (i < a.size() ? a[i] - 48 : 0) + (i <
77                b.a.size() ? b.a[i] - 48 : 0);
78            c.a += (carry % 10 + 48);
79            carry /= 10;
80        }
81        return c.normalize(sign);
82    }
83
84    Bigint operator - ( Bigint b ) // subtraction
85        operator overloading
86    {
87        if( sign != b.sign )
88            return (*this) + b.inverseSign();
89        int s = sign; sign = b.sign = 1;
90        if( (*this) < b )
91            return ((b - (*this)).inverseSign()).
92                normalize(-s);
93        Bigint c;
94        for( int i = 0, borrow = 0; i < a.size();
95            i++ )
96        {
97            borrow = a[i] - borrow - (i < b.size()
98                ? b.a[i] : 48);
99            c.a += borrow >= 0 ? borrow + 48 :
100                borrow + 58;
101            borrow = borrow >= 0 ? 0 : 1;
102        }
103        return c.normalize(s);
104    }
105
106    Bigint operator * ( Bigint b ) //
107        multiplication operator overloading
108    {
109        Bigint c("0");
110        for( int i = 0, k = a[i] - 48; i < a.size()
111            (); i++, k = a[i] - 48 )
112        {
113            while(k-->0)
114                c = c + b; // ith digit is k, so,
115                    we add k times
116            b.a.insert(b.a.begin(), '0'); //
117                multiplied by 10
118        }
119        return c.normalize(sign * b.sign);
120    }
121
122    Bigint operator / ( Bigint b ) // division
123        operator overloading
124    {
125        if( b.size() == 1 && b.a[0] == '0' )
126            b.a[0] /= ( b.a[0] - 48 );
127        Bigint c("0"), d;
128        for( int j = 0; j < a.size(); j++ )
129            d.a += "0";
130        int dSign = sign * b.sign;
131        b.sign = 1;
132        for( int i = a.size() - 1; i >= 0; i-- )
133        {
134            c.a.insert( c.a.begin(), '0' );
135            c = c + a.substr( i, 1 );
136        }
137    }

```

```

111         while( !( c < b ) )
112         {
113             c = c - b;
114             d.a[i]++;
115         }
116     }
117     return d.normalize(dSign);
118 }
119 Bigint operator % ( Bigint b ) // modulo
    operator overloading
120 {
121     if( b.size() == 1 && b.a[0] == '0' )
122         b.a[0] /= ( b.a[0] - 48 );
123     Bigint c("0");
124     b.sign = 1;
125     for( int i = a.size() - 1; i >= 0; i— )
126     {
127         c.a.insert( c.a.begin(), '0' );
128         c = c + a.substr( i, 1 );
129         while( !( c < b ) )
130             c = c - b;
131     }
132     return c.normalize(sign);
133 }
134
135 // output method
136 void print()
137 {
138     if( sign == -1 )
139         putchar('-');
140     for( int i = a.size() - 1; i >= 0; i— )
141         putchar(a[i]);
142 }
143 };
144
145
146
147 int main()
148 {
149     Bigint a, b, c; // declared some Bigint
        variables
150     ///////////////////////////////////
151     // taking Bigint input //
152     ///////////////////////////////////
153
154     std::string input; // std::string to take
        input
155     std::cin >> input; // take the Big integer
        std::string
156     a = input; // assign the std::string to
        Bigint a
157
158     std::cin >> input; // take the Big integer
        std::string
159     b = input; // assign the std::string to
        Bigint b
160
161     ///////////////////////////////////
162     // Using mathematical operators //
163     ///////////////////////////////////
164
165     c = a + b; // adding a and b
166     c.print(); // printing the Bigint
167     puts(""); // newline
168
169     c = a - b; // subtracting b from a
170     c.print(); // printing the Bigint
171     puts(""); // newline
172
173     c = a * b; // multiplying a and b
174
175     c.print(); // printing the Bigint
176     puts(""); // newline
177
178     c = a / b; // dividing a by b
179     c.print(); // printing the Bigint
180     puts(""); // newline
181
182     c = a % b; // a modulo b
183     c.print(); // printing the Bigint
184     puts(""); // newline
185
186     ///////////////////////////////////
187     // Using conditional operators //
188     ///////////////////////////////////
189
190     if( a == b )
191         puts("equal"); // checking equality
192     else
193         puts("not equal");
194
195     if( a < b )
196         puts("a is smaller than b"); // checking
        less than operator
197
198     return 0;
199 }

```

9.3 Binary Search

```

1 // [0, n)
2 inline int go(int A[], int n, int x) // return the
    least i that 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) // return the
45     largest i that make A[i]<x;
46 {
47     static int l, r, mid, re;
48     l=0;
49     r=n-1;
50     re=-1;
51     while(l<=r)
52     {
53         mid=l+r>>1;
54         if(A[mid]<x)
55         {
56             l=mid+1;
57             re=mid;
58         }
59         else
60             r=mid-1;
61     }
62     return re;
63 }
64
65 inline int go(int A[], int n, int x) // return the
66     largest i that make A[i]<=x;
67 {
68     static int l, r, mid, re;
69     l=0;
70     r=n-1;
71     re=-1;
72     while(l<=r)
73     {
74         mid=l+r>>1;
75         if(A[mid]<=x)
76         {
77             l=mid+1;
78             re=mid;
79         }
80         else
81             r=mid-1;
82     }
83     return re;
84 }
85
86 inline int go(int A[], int n, int x) // return the
87     least i that make A[i]>x;
88 {
89     static int l, r, mid, re;
90     l=0;
91     r=n-1;
92     re=-1;
93     while(l<=r)
94     {
95         mid=l+r>>1;
96         if(A[mid]<=x)
97         {
98             l=mid+1;
99         }
100         else
101             r=mid-1;
102     }
103     return re;
104 }
105
106 inline int go(int A[], int n, int x) // upper_bound
107     ();
108 {
109     static int l, r, mid;
110     l=0;
111     r=n-1;
112     while(l<r)
113     {
114         mid=l+r>>1;
115         if(A[mid]<=x)
116             l=mid+1;
117         else
118             r=mid;
119     }
120     return r;
121 }
122
123 inline int go(int A[], int n, int x) // lower_bound
124     ();
125 {
126     static int l, r, mid;
127     l=0;
128     r=n-1;
129     while(l<r)
130     {
131         mid=l+r>>1;
132         if(A[mid]<x)
133             l=mid+1;
134         else
135             r=mid;
136     }
137     return r;
138 }

```

9.4

```

1 //Scanner
2
3 Scanner in=new Scanner(new FileReader("asdf"));
4 PrintWriter pw=new PrintWriter(new FileWriter("
5 out"));
6
7 boolean in.hasNext();
8 String in.next();
9 BigDecimal in.nextBigDecimal();
10 BigInteger in.nextBigInteger();
11 BigInteger in.nextBigInteger(int radix);
12 double in.nextDouble();
13 int in.nextInt();
14 int in.nextInt(int radix);
15 String in.nextLine();
16 long in.nextLong();
17 long in.nextLong(int radix);
18 short in.nextShort();
19 short in.nextShort(int radix);
20 int in.nextInt(); //Returns this scanner's
21     s default radix.
22 Scanner in.useRadix(int radix); // Sets this
23     scanner's default radix to the specified
24     radix.
25 void in.close(); //Closes this scanner.
26
27 //String
28
29 char str.charAt(int index);
30 int str.compareTo(String anotherString);
31 ; // <0 if less. ==0 if equal. >0 if greater.
32 int str.compareToIgnoreCase(String str);
33
34 String str.concat(String str);
35 boolean str.contains(CharSequence s);
36 boolean str.endsWith(String suffix);
37 boolean str.startsWith(String prefix);
38 boolean str.startsWith(String prefix, int

```

	toffset);		the[i].b);
32	int str.hashCode();	86	}
33	int str.indexOf(int ch);	87	}
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);	1	god damn it windows:
38	int str.lastIndexOf(int ch,int fromIndex);	2	#pragma comment(linker, "/STACK:16777216")
39	//(ry	3	#pragma comment(linker, "/STACK:102400000,102400000")
40	int str.length();	4	
41	String str.substring(int beginIndex);	5	
42	String str.substring(int beginIndex,int endIndex);	6	chmod +x [filename]
43	String str.toLowerCase();	7	
44	String str.toUpperCase();	8	while true; do
45	String str.trim();// Returns a copy of the string, with leading and trailing whitespace omitted.	9	./gen > input
		10	./sol < input > output.sol
			./bf < input > output.bf
46		13	diff output.sol output.bf
47	//StringBuilder	14	if[\$? -ne 0];then break fi
48	StringBuilder str.insert(int offset,...);	15	done、状态状态状态状态状态状态状态状态状态状态
49	StringBuilder str.reverse();	16	
50	void str.setCharAt(int index,int ch);	17	
51		18	1、
52	//BigInteger	19	2calm_down();calm_down();calm_down();、读完题目读完题目读完题目
53	compareTo(); equals(); doubleValue(); longValue(); hashCode(); toString(); toString(int radix); max(); min(); mod(); modPow(BigInteger exp, BigInteger m); nextProbablePrime(); pow(); andNot(); and(); xor(); not(); or(); getLowestSetBit(); bitCount(); bitLength(); setBit(int n); shiftLeft(int n); shiftRight(int n); add(); divide(); divideAndRemainder(); remainder(); multiply(); subtract(); gcd(); abs(); signum(); negate();	20	3、不盲目跟版
54		21	4、考虑换题换想法
55		22	5、对数离线
56		23	6//hash观察问题本身点 区间互转//、对数调整精度
57		24	6.1 or 将乘法转换成加法、点化区间，区间化点
58		25	6.2、数组大小.....
59		26	7
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,		