

1 Computational_Geometry

1.1 Geometry.cpp

```

1 template<typename T>
2 struct point{
3     T x,y;
4     point(){}
5     point(const T&x,const T&y):x(x),y(y){}
6     point operator+(const point &b)const{
7         return point(x+b.x,y+b.y);
8     }
9     point operator-(const point &b)const{
10        return point(x-b.x,y-b.y);
11    }
12    point operator*(const T &b)const{return
13        point(x*b,y*b);}
14    point operator/(const T &b)const{return
15        point(x/b,y/b);}
16    bool operator==(const point &b)const{
17        return x==b.x&&y==b.y;
18    }
19    T dot(const point &b)const{
20        return x*b.x+y*b.y;
21    }
22    T cross(const point &b)const{
23        return x*b.y-y*b.x;
24    }
25    point normal()const{/*求法向量*/
26        return point(-y,x);
27    }
28    T abs2()const{/*向量長度的平方*/
29        return dot(*this);
30    }
31    T rad(const point &b)const{/*兩向量的弧度(
32        夾角)*/
33        return fabs(atan2(fabs(cross(b)),dot(b))
34            );
35    };
36    template<typename T>
37    struct line{
38        line(){}
39        point<T> p1,p2;
40        T a,b,c; /*ax+by+c=0*/
41        line(const point<T>&x,const point<T>&y):p1
42            (x),p2(y){}
43        void pton(){/*轉成一般式*/
44            a=p1.y-p2.y;
45            b=p2.x-p1.x;
46            c=-a*p1.x-b*p1.y;
47        }
48        T cross(const point<T> &p)const{/*點和有向
49            直線的關係，>0左邊、=0在線上<0右邊*/
50            return (p2-p1).cross(p-p1);
51        }
52        bool point_on_segment(const point<T>&p)
53            const{/*點是否線段上*/
54            return cross(p)==0&&(p1-p).dot(p2-p)<=0;
55        }
56        T dis2(const point<T> &p,bool is_segment
57            =0)const{/*點跟直線/線段的距離平方*/
58            point<T> v=p2-p1,v1=p-p1;
59            if(is_segment){
60                point<T> v2=p-p2;
61                if(v.dot(v1)<=0)return v1.abs2();
62                if(v.dot(v2)>=0)return v2.abs2();
63            }
64            T tmp=v.cross(v1);
65            return tmp*tmp/v.abs2();
66        }
67        point<T> projection(const point<T> &p)
68            const{/*點對直線的投影
69            point<T> n=(p2-p1).normal();
70            return p-n*(p-p1).dot(n)/n.abs2();
71        }
72        point<T> mirror(const point<T> &p)const{/*
73            點對直線的鏡射*/
74            /*要先呼叫 pton 轉成一般式*/
75            point<T> ans;
76            T d=a*a+b*b;
77            ans.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/
78                d;
79            ans.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/
80                d;
81            return ans;
82        }
83        bool equal(const line &l)const{/*直線相等
84            */
85            return cross(l.p1)==0&&cross(l.p2)==0;
86        }
87        bool parallel(const line &l)const{/*直線平
88            行*/
89            return (p1-p2).cross(l.p1-l.p2)==0;
90        }
91        bool cross_seg(const line &l)const{/*直線
92            是否交線段*/
93            return (p2-p1).cross(l.p1)*(p2-p1).cross
94                (l.p2)<=0;
95        }
96        char line_intersect(const line &l)const{/*
97            直線相交情況，-1無限多點、1交於一點、0
98            不相交*/
99            return parallel(l)?(cross(l.p1)==0?-1:0)
100                :1;
101        }
102        char seg_intersect(const line &l)const{/*
103            線段相交情況，-1無限多點、1交於一點、0
104            不相交*/
105            T c1=(p2-p1).cross(l.p1-p1);
106            T c2=(p2-p1).cross(l.p2-p1);
107            T c3=(l.p2-l.p1).cross(p1-l.p1);
108            T c4=(l.p2-l.p1).cross(p2-l.p1);
109            if(c1==0&&c2==0){
110                if(p1==l.p1&&(p2-p1).dot(l.p2)<=0)
111                    return 1;
112                if(p1==l.p2&&(p2-p1).dot(l.p1)<=0)
113                    return 1;
114                if(p2==l.p1&&(p1-p2).dot(l.p2)<=0)
115                    return 1;
116                if(p2==l.p2&&(p1-p2).dot(l.p1)<=0)
117                    return 1;
118                return -1;
119            }else if(c1*c2<=0&&c3*c4<=0)return 1;
120            return 0;
121        }
122        point<T> line_intersection(const line &l)
123            const{/*直線交點*/
124            point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
125            //if(a.cross(b)==0)return INF;
126            return p1+a*s.cross(b)/a.cross(b);
127        }
128        point<T> seg_intersection(const line &l)
129            const{/*線段交點*/
130            T c1=(p2-p1).cross(l.p1-p1);
131            T c2=(p2-p1).cross(l.p2-p1);
132            T c3=(l.p2-l.p1).cross(p1-l.p1);
133            T c4=(l.p2-l.p1).cross(p2-l.p1);
134            if(c1==0&&c2==0){
135                if(p1==l.p1&&(p2-p1).dot(l.p2)<=0)
136                    return p1;
137                if(p1==l.p2&&(p2-p1).dot(l.p1)<=0)
138                    return p1;
139                if(p2==l.p1&&(p1-p2).dot(l.p2)<=0)
140                    return p2;
141                if(p2==l.p2&&(p1-p2).dot(l.p1)<=0)
142                    return p2;
143            }else if(c1*c2<=0&&c3*c4<=0)return
144                line_intersection(l);
145            //return INF;
146        }
147    };
148    template<typename T>
149    struct polygon{
150        polygon(){}
151        vector<point<T> > p; //逆時針順序
152        T area()const{/*多邊形面積*/
153            T ans=0;
154            for(int i=p.size()-1,j=0;j<(int)p.size()
155                ;i=j++){
156                ans+=p[i].cross(p[j]);
157            }
158            return ans/2;
159        }
160        point<T> center_of_mass()const{/*多邊形重
161            心*/
162            T cx=0,cy=0,w=0;
163            for(int i=p.size()-1,j=0;j<(int)p.size()
164                ;i=j++){
165                T a=p[i].cross(p[j]);
166                cx+=(p[i].x+p[j].x)*a;
167                cy+=(p[i].y+p[j].y)*a;
168                w+=a;
169            }
170            return point<T>(cx/3/w,cy/3/w);
171        }
172        char ahas(const point<T>& t)const{/*點是否
173            在簡單多邊形內，是的話回傳1、在邊上回
174            傳-1、否則回傳0*/
175            bool c=0;
176            for(int i=0,j=p.size()-1;i<p.size();j=i
177                ++){
178                if(line<T>(p[i],p[j]).point_on_segment
179                    (t))return -1;
180                else if((p[i].y>t.y)!=p[j].y>t.y)&&
181                    t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j
182                        ].y-p[i].y)+p[i].x)
183                    c=!c;
184            }
185            return c;
186        }
187        char point_in_convex(const point<T>&x)
188            const{
189            int l=1,r=(int)p.size()-2;
190            while(l<=r){/*點是否在凸多邊形內，是的話
191                回傳1、在邊上回傳-1、否則回傳0*/
192                int mid=(l+r)/2;
193                T a1=(p[mid]-p[0]).cross(x-p[0]);
194                T a2=(p[mid+1]-p[0]).cross(x-p[0]);
195                if(a1==0&&a2<=0){
196                    T res=(p[mid+1]-p[mid]).cross(x-p[
197                        mid]);
198                    return res>0?1:(res>0?-1:0);
199                }else if(a1<0)r=mid-1;
200                else l=mid+1;
201            }
202            return 0;
203        }
204        polygon cut(const line<T> &l)const{/*凸包
205            對直線切割，得到直線L左側的凸包*/
206            polygon ans;
207            for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
208                if(l.cross(p[i])>=0){
209                    ans.p.push_back(p[i]);
210                    if(l.cross(p[j])<0)
211                        ans.p.push_back(l.
212                            line_intersection(line<T>(p[i
213                                ],p[j])));
214                }else if(l.cross(p[j])>0)
215                    ans.p.push_back(l.line_intersection(
216                        line<T>(p[i],p[j])));
217            }
218            return ans;
219        }
220        static bool graham_cmp(const point<T>& a,
221            const point<T>& b){
222            return (a.x<b.x)||((a.x==b.x&&a.y<b.y));/*
223                凸包排序函數*/
224        }
225        void graham(vector<point<T> > &s){/*凸包*/
226            sort(s.begin(),s.end(),graham_cmp);
227            p.resize(s.size()+1);
228            int m=0;
229            for(int i=0;i<(int)s.size();i++){
230                while(m>2&&(p[m-1]-p[m-2]).cross(s[i
231                    ]-p[m-2])<=0)--m;
232                p[m++]=s[i];
233            }
234            for(int i=s.size()-2,t=m+1;i>=0;--i){
235                while(m>2&&(p[m-1]-p[m-2]).cross(s[i
236                    ]-p[m-2])<=0)--m;
237                p[m++]=s[i];
238            }
239            if(s.size())>1--m;
240            p.resize(m);
241        }
242        inline static char sign(const point<T>&t){
243            return (t.y==0?t.x:t.y)<0;
244        }
245        inline static bool angle_cmp(const line<T
246            >& A,const line<T>& B){
247            point<T> a=A.p2-A.p1,b=B.p2-B.p1;
248            return sign(a)<sign(b)||((sign(a)==sign(b)
249                )&&a.cross(b)>0);
250        }
251        int halfplane_intersection(vector<line<T>
252            > &s){/*半平面交

```

```

1 sort(s.begin(),s.end(),angle_cmp);//線段
2     左側為該線段半平面
195 int L,R,n=s.size();
196 vector<point<T> > px(n);
197 vector<line<T> > q(n);
198 q[L=R=0]=s[0];
199 for(int i=1;i<n;++i){
200     while(L<R&&s[i].cross(px[R-1])<=0)--R;
201     while(L<R&&s[i].cross(px[L])<=0)++L;
202     q[++R]=s[i];
203     if(q[R].parallel(q[R-1])){
204         --R;
205         if(q[R].cross(s[i].p1)>0)q[R]=s[i];
206     }
207     if(L<R)px[R-1]=q[R-1].
208         line_intersection(q[R]);
209 }
210 while(L<R&&q[L].cross(px[R-1])<=0)--R;
211 p.clear();
212 if(R-L==1)return 0;
213 px[R]=q[R].line_intersection(q[L]);
214 for(int i=L;i<R;++i)p.push_back(px[i]);
215 return R-L+1;
216 }
217 template<typename T>
218 struct triangle{
219     point<T> a,b,c;
220     triangle(){}
221     triangle(const point<T> &a,const point<T>
222         &b,const point<T> &c):a(a),b(b),c(c){}
223     T area()const{
224         T t=(b-a).cross(c-a)/2;
225         return t>0?t:-t;
226     }
227     point<T> barycenter()const{//重心*
228         return (a+b+c)/3;
229     }
230     point<T> circumcenter()const{//外心*
231         static line<T> u,v;
232         u.p1=(a+b)/2;
233         u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
234             b.x);
235         v.p1=(a+c)/2;
236         v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
237             c.x);
238         return u.line_intersection(v);
239     }
240     point<T> incenter()const{//內心 · 用到根號
241         *
242         T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
243             ()),C=sqrt((a-b).abs2());
244         return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
245             B*b.y+C*c.y)/(A+B+C);
246     }
247     point<T> perpencenter()const{//垂心*
248         return barycenter()*3-circumcenter()*2;
249     }
250 };
251 template<typename T>
252 struct point3D{
253     T x,y,z;
254     point3D(){}
255     point3D(const T&x,const T&y,const T&z):x(x
256         ),y(y),z(z){}
257     point3D operator+(const point3D &b)const{
258         return point3D(x+b.x,y+b.y,z+b.z);
259     }
260     point3D operator-(const point3D &b)const{
261         return point3D(x-b.x,y-b.y,z-b.z);
262     }
263     point3D operator*(const T &b)const{
264         return point3D(x*b,y*b,z*b);
265     }
266     point3D operator/(const T &b)const{
267         return point3D(x/b,y/b,z/b);
268     }
269     bool operator==(const point3D &b)const{
270         return x==b.x&&y==b.y&&z==b.z;
271     }
272     T dot(const point3D &b)const{
273         return x*b.x+y*b.y+z*b.z;
274     }
275     T dot(const point3D &b)const{
276         return x*b.x+y*b.y+z*b.z;
277     }
278     point3D cross(const point3D &b)const{
279         return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
280             *b.y-y*b.x);
281     }
282     T abs2()const{//向量長度的平方*
283         return dot(*this);
284     }
285     T area2(const point3D &b)const{//和b、原點
286         圍成面積的平方
287         return cross(b).abs2()/4;
288     }
289 };
290 template<typename T>
291 struct line3D{
292     point3D<T> p1,p2;
293     line3D(){}
294     line3D(const point3D<T> &p1,const point3D<
295         T> &p2):p1(p1),p2(p2){}
296     T dis2(const point3D<T> &p,bool is_segment
297         =0)const{//點跟直線/線段的距離平方*
298         point3D<T> v=p2-p1,v1=p-p1;
299         if(is_segment){
300             point3D<T> v2=p-p2;
301             if(v.dot(v1)<=0)return v1.abs2();
302             if(v.dot(v2)>=0)return v2.abs2();
303         }
304         point3D<T> tmp=v.cross(v1);
305         return tmp.abs2()/v.abs2();
306     }
307     pair<point3D<T>,point3D<T> > closest_pair(
308         const line3D<T> &l)const{
309         point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
310         point3D<T> N=v1.cross(v2),ab=(p1-l.p1);
311         //if(N.abs2()==0)return NULL; 平行或重合
312         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
313         最近點對距離
314         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
315             cross(d2);
316         T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2
317             ();
318         T t2=((l.p1-p1).cross(d1)).dot(D)/D.
319             abs2();
320         return make_pair(p1+d1*t1,l.p1+d2*t2);
321     }
322     bool same_side(const point3D<T> &a,const
323         point3D<T> &b)const{
324         return (p2-p1).cross(a-p1).dot((p2-p1).
325             cross(b-p1))>0;
326     }
327 };
328 template<typename T>
329 struct plane{
330     point3D<T> p0,n;//平面上的點和法向量
331     plane(){}
332     plane(const point3D<T> &p0,const point3D<T>
333         &n):p0(p0),n(n){}
334     T dis2(const point3D<T> &p)const{//點到平
335         面距離的平方
336         T tmp=(p-p0).dot(n);
337         return tmp*tmp/n.abs2();
338     }
339     point3D<T> projection(const point3D<T> &p)
340         const{
341         return p-n*(p-p0).dot(n)/n.abs2();
342     }
343     point3D<T> line_intersection(const line3D<
344         T> &l)const{
345         T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
346         重合該平面
347         return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
348             tmp);
349     }
350     line3D<T> plane_intersection(const plane &
351         p1)const{
352         point3D<T> e=n.cross(p1.n),v=n.cross(e);
353         T tmp=p1.n.dot(v);//等於0表示平行或重合
354         該平面
355         point3D<T> q=p0+(v*(p1.n.dot(p1.p0-p0))/
356             tmp);
357         return line3D<T>(q,q+e);
358     }
359 };
360 template<typename T>
361 struct triangle3D{
362     point3D<T> a,b,c;
363     triangle3D(){}
364     triangle3D(const point3D<T> &a,const
365         point3D<T> &b,const point3D<T> &c):a(a
366         ),b(b),c(c){}
367     bool point_in(const point3D<T> &p)const{//
368         點在該平面上的投影在三角形中
369         return line3D<T>(b,c).same_side(p,a)&&
370             line3D<T>(a,c).same_side(p,b)&&
371             line3D<T>(a,b).same_side(p,c);
372     }
373 };
374 template<typename T>
375 struct tetrahedron{//四面體
376     point3D<T> a,b,c,d;
377     tetrahedron(){}
378     tetrahedron(const point3D<T> &a,const
379         point3D<T> &b,const point3D<T> &c,
380         const point3D<T> &d):a(a),b(b),c(c),d(
381             d){}
382     T volume6()const{//體積的六倍
383         return (d-a).dot((b-a).cross(c-a));
384     }
385     point3D<T> centroid()const{
386         return (a+b+c+d)/4;
387     }
388 };
389 bool point_in(const point3D<T> &p)const{
390     return triangle3D<T>(a,b,c).point_in(p)
391         &&triangle3D<T>(c,d,a).point_in(p);
392 }
393 template<typename T>
394 struct convexhull3D{
395     static const int MAXN=105;
396     struct face{
397         int a,b,c;
398         bool use;
399         face(){}
400         face(int a,int b,int c):a(a),b(b),c(c),
401             use(1){}
402     };
403     vector<point3D<T> > pt;
404     vector<face> fc;
405     int fid[MAXN][MAXN];
406     static bool point_cmp(const point3D<T> &a,
407         const point3D<T> &b){
408         return a.x<b.x||(a.x==b.x&&(a.y<b.y||(a.
409             y==b.y&&a.z<b.z)));
410     }
411     bool outside(int p,int a,int b,int c)const
412         {
413         return tetrahedron<T>(pt[a],pt[b],pt[c],
414             pt[p]).volume6()<0;
415     }
416     bool outside(int p,int f)const{return
417         outside(p,fc[f].a,fc[f].b,fc[f].c);}
418     void add_face(int a,int b,int c,int p){
419         if(outside(p,a,b,c))fid[c][b]=fid[b][a]=
420             fid[a][c]=fc.size(),fc.push_back(
421                 face(c,b,a));
422         else fid[a][b]=fid[b][c]=fid[c][a]=fc.
423             size(),fc.push_back(face(a,b,c));
424     }
425     bool dfs(int p,int f){
426         if(!fc[f].use)return true;
427         if(outside(p,f)){
428             int a=fc[f].a,b=fc[f].b,c=fc[f].c;
429             fc[f].use=false;
430             if(!dfs(p,fid[b][a]))add_face(p,a,b,c)
431                 ;
432             if(!dfs(p,fid[c][b]))add_face(p,b,c,a)
433                 ;
434             if(!dfs(p,fid[a][c]))add_face(p,c,a,b)
435                 ;
436             return true;
437         }else return false;
438     }
439     void build(){
440         bool ok=false;
441         fc.clear();
442         sort(pt.begin(),pt.end(),point_cmp);
443         pt.resize(unique(pt.begin(),pt.end())-pt.
444             begin());
445         for(size_t i=2;i<pt.size();++i){
446             if((pt[0]-pt[i]).area2(pt[1]-pt[i])
447                 !=0){
448                 ok=true;
449                 swap(pt[i],pt[2]);
450                 break;
451             }
452         }
453         if(!ok)return;
454     }
455 };

```

```

401 ok=false;
402 for(size_t i=3;i<pt.size();++i){
403     if(tetrahedron<T>(pt[0],pt[1],pt[2],pt
404         [i]).volume6()!=0){
405         ok=true;
406         swap(pt[i],pt[3]);
407         break;
408     }
409     if(!ok)return;
410     for(int i=0;i<4;++i)add_face(i,(i+1)%4,(
411         i+2)%4,(i+3)%4);
412     for(size_t i=4;i<pt.size();++i){
413         for(int j=fc.size()-1;j>=0;--j){
414             if(outside(i,j)){
415                 dfs(i,j);
416                 break;
417             }
418         }
419     }
420     size_t sz=0;
421     for(size_t i=0;i<fc.size();++i)if(fc[i].
422         use)fc[sz++]=fc[i];
423     fc.resize(sz);
424 }
425 point3D<T> centroid()const{
426     point3D<T> res(0,0,0);
427     T vol=0;
428     for(size_t i=0;i<fc.size();++i){
429         T tmp=pt[fc[i].a].dot(pt[fc[i].b].
430             cross(pt[fc[i].c]));
431         res=res+(pt[fc[i].a]+pt[fc[i].b]+pt[fc
432             [i].c])*tmp;
433         vol+=tmp;
434     }
435     return res/(vol*4);
436 }
437 };

```

1.2 SmallestCircle.cpp

```

1 #include "Geometry.cpp"
2 #include <vector>
3 struct Circle{
4     typedef point<double> p;
5     typedef const point<double> cp;
6     p x;
7     double r2;
8     bool incircle(cp &c)const{return (x-c).
9         abs2()<=r2;}
10 };
11 Circle TwoPointCircle(Circle::cp &a, Circle
12     ::cp &b) {
13     Circle::p m=(a+b)/2;
14     return (Circle){m,(a-m).abs2()};
15 }
16 Circle outcircle(Circle::p a, Circle::p b,
17     Circle::p c) {
18     if(TwoPointCircle(a,b).incircle(c))
19         return TwoPointCircle(a,b);

```

```

18     if(TwoPointCircle(b,c).incircle(a))
19         return TwoPointCircle(b,c);
20     if(TwoPointCircle(c,a).incircle(b))
21         return TwoPointCircle(c,a);
22     Circle::p ret;
23     double a1=b.x-a.x, b1=b.y-a.y, c1=(a1*a1
24         +b1*b1)/2;
25     double a2=c.x-a.x, b2=c.y-a.y, c2=(a2*a2
26         +b2*b2)/2;
27     double d = a1*b2 - a2*b1;
28     ret.x=a.x+(c1*b2-c2*b1)/d;
29     ret.y=a.y+(a1*c2-a2*c1)/d;
30     return (Circle){ret,(ret-a).abs2()};
31 }
32 //rand required
33 Circle SmallestCircle(std::vector<Circle::p>
34     &p){
35     int n=p.size();
36     if(n==1) return (Circle){p[0],0.0};
37     if(n==2) return TwoPointCircle(p[0],p
38         [1]);
39     random_shuffle(p.begin(),p.end());
40     Circle c = {p[0],0.0};
41     for(int i=0;i<n;++i){
42         if(c.incircle(p[i])) continue;
43         c=Circle{p[i],0.0};
44         for(int j=0;j<i;++j){
45             if(c.incircle(p[j])) continue;
46             c=TwoPointCircle(p[i],p[j]);
47             for(int k=0;k<j;++k){
48                 if(c.incircle(p[k]))
49                     continue;
50                 c=outcircle(p[i],p[j],p[k]);
51             }
52         }
53     }
54     return c;
55 }

```

1.3 最近點對.cpp

```

1 #define INF LLONG_MAX/*預設是Long Long最大值
2 */
3 template<typename T>
4 T closest_pair(vector<point<T>> &v,vector<
5     point<T>> &t,int l,int r){
6     T dis=INF,tmd;
7     if(l==r)return dis;
8     int mid=(l+r)/2;
9     if((tmd=closest_pair(v,t,l,mid))<dis)dis=
10         tmd;
11     if((tmd=closest_pair(v,t,mid+1,r))<dis)dis=
12         tmd;
13     t.clear();
14     for(int i=l;i<r;++i)
15         if((v[i].x-v[mid].x)*(v[i].x-v[mid].x)<
16             dis)t.push_back(v[i]);
17     sort(t.begin(),t.end(),point<T>::y_cmp);/*
18     如果用merge_sort的方式可以O(n)*
19     for(int i=0;i<(int)t.size();++i)
20         for(int j=1;j<=3&&i+j<(int)t.size();++j)
21             if((tmd=(t[i]-t[i+j]).abs2())<dis)dis=
22                 tmd;

```

```

16     return dis;
17 }
18 template<typename T>
19 inline T closest_pair(vector<point<T>> &v){
20     vector<point<T>> >t;
21     sort(v.begin(),v.end(),point<T>::x_cmp);
22     return closest_pair(v,t,0,v.size()-1);/*最
23     近點對距離*/

```

1.4 浮點數誤差模板.cpp

```

1 const double EPS=1e-9;
2 struct Double{
3     double d;
4     Double(double d=0):d(d){}
5     bool operator <(const Double &b)const{
6         return d-b.d<-EPS;}
7     bool operator >(const Double &b)const{
8         return d-b.d>EPS;}
9     bool operator ==(const Double &b)const{
10         return fabs(d-b.d)<=EPS;}
11     bool operator !=(const Double &b)const{
12         return fabs(d-b.d)>EPS;}
13     bool operator <=(const Double &b)const{
14         return d-b.d<=EPS;}
15     bool operator >=(const Double &b)const{
16         return d-b.d>=-EPS;}
17     operator double()const{return d;}

```

2 Data_Structure

2.1 DLX.cpp

```

1 #define MAXN 4100
2 #define MAXM 1030
3 #define MAXND 16390
4 struct DLX{
5     int n,m,sz,ansd; /*高是n，寬是m的稀疏矩陣
6     int S[MAXN],H[MAXN];
7     int row[MAXND],col[MAXND]; /*每個節點代表的
8     列跟行
9     int L[MAXND],R[MAXND],U[MAXND],D[MAXND];
10     vector<int> ans,anst;
11     void init(int _n,int _m){
12         n=_n,m=_m;
13         for(int i=0;i<=m;++i){
14             U[i]=D[i]=i,L[i]=i-1,R[i]=i+1;
15             S[i]=0;
16         }
17         R[m]=0,L[0]=m;
18         sz=m,ansd=INT_MAX; /*ansd存最優解的個數
19         for(int i=1;i<=n;++i)H[i]=-1;
20     }
21     void add(int r,int c){
22         ++S[col[++sz]=c];

```

```

23     row[sz]=r;
24     D[sz]=D[c],U[D[c]]=sz,U[sz]=c,D[c]=sz;
25     if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
26     else R[sz]=R[H[r]],L[R[H[r]]]=sz,L[sz]=H
27         [r],R[H[r]]=sz;
28 }
29 #define DFOR(i,A,s) for(int i=A[s];i!=s;i=
30     A[i])
31 void remove(int c){/*刪除第c行和所有當前覆
32     蓋到第c行的列
33     L[R[c]]=L[c],R[L[c]]=R[c];/*這裡刪除第c
34     行，若有些行不需要處理可以在開始時呼
35     叫他
36     DFOR(i,D,c)DFOR(j,R,i){U[D[j]]=U[j],D[U
37         [j]]=D[j],--S[col[j]];}
38 }
39 void restore(int c){/*恢復第c行和所有當前
40     覆蓋到第c行的列，remove的逆操作
41     DFOR(i,U,c)DFOR(j,L,i){++S[col[j]],U[D[j]
42         ]=j,D[U[j]]=j;}
43     L[R[c]]=c,R[L[c]]=c;
44 }
45 void remove2(int nd){/*刪除nd所在的行當前
46     所有點(包括虛擬節點)，只保留nd
47     DFOR(i,D,nd)L[R[i]]=L[i],R[L[i]]=R[i];
48 }
49 void restore2(int nd){/*刪除nd所在的行當前
50     所有點，為remove2的逆操作
51     DFOR(i,U,nd)L[R[i]]=R[L[i]]=i;
52 }
53 bool vis[MAXN];
54 int h(){/*估價函數 for IDA*
55     int res=0;
56     memset(vis,0,sizeof(vis));
57     DFOR(i,R,0)if(!vis[i]){
58         vis[i]=1;
59         ++res;
60         DFOR(j,D,i)DFOR(k,R,j)vis[col[k]]=1;
61     }
62     return res;
63 }
64 bool dfs(int d){/*for精確覆蓋問題
65     if(d+h())>=ansd)return 0; /*找最佳解用，找
66     任意解可以刪掉
67     if(!R[0]){ansd=d;return 1;}
68     int c=R[0];
69     DFOR(i,R,0)if(S[i]<S[c])c=i;
70     remove(c);
71     DFOR(i,D,c){
72         ans.push_back(row[i]);
73         DFOR(j,R,i)remove(col[j]);
74         if(dfs(d+1))return 1;
75         ans.pop_back();
76         DFOR(j,L,i)restore(col[j]);
77     }
78     restore(c);
79     return 0;
80 }
81 void dfs2(int d){/*for最小重複覆蓋問題
82     if(d+h())>=ansd)return;
83     if(!R[0]){ansd=d;ans=anst;return;}
84     int c=R[0];
85     DFOR(i,R,0)if(S[i]<S[c])c=i;

```

```

74 DFOR(i,D,c){
75     anst.push_back(row[i]);
76     remove2(i);
77     DFOR(j,R,i)remove2(j),--S[col[j]];
78     dfs2(d+1);
79     anst.pop_back();
80     DFOR(j,L,i)restore2(j),++S[col[j]];
81     restore2(i);
82 }
83 }
84 bool exact_cover(){//解精確覆蓋問題
85     ans.clear();//答案存在ans裡
86     return dfs(0);
87 }
88 void min_cover(){//解最小重複覆蓋問題
89     anst.clear();//這只是暫存用，答案還是存在ans裡
90     dfs2(0);
91 }
92 #undef DFOR
93 };

```

2.2 Dynamic_KD_tree.cpp

```

1 template<typename T,size_t kd>//kd的維度 3 `Xf`
2 class kd_tree{
3 public:
4     struct point{
5         T d[kd];
6         T dist(const point &x)const{
7             T ret=0;
8             for(size_t i=0;i<kd;++i)ret+=std::abs(d[i]-x.d[i]);
9             return ret;
10        }
11        bool operator==(const point &p){
12            for(size_t i=0;i<kd;++i)
13                if(d[i]!=p.d[i])return 0;
14            return 1;
15        }
16        bool operator<(const point &b)const{
17            return d[0]<b.d[0];
18        }
19    };
20    private:
21        struct node{
22            node *l,*r;
23            point pid;
24            int s;
25            node(const point &p):l(0),r(0),pid(p),s(1){}
26            ~node(){delete l;delete r;}
27            void up(){s=(l?l->s:0)+1+(r?r->s:0);}
28        }*root;
29        const double alpha,loga;
30        const T INF;//°onµ²INF;A°µ²Xj
31        int maxn;
32        struct __cmp{
33            int sort_id;
34            bool operator()(const node*x,const node*y)const{

```

```

35         return operator()(x->pid,y->pid);
36     }
37     bool operator()(const point &x,const point &y)const{
38         if(x.d[sort_id]!=y.d[sort_id])
39             return x.d[sort_id]<y.d[sort_id];
40         for(size_t i=0;i<kd;++i)
41             if(x.d[i]!=y.d[i])return x.d[i]<y.d[i];
42         return 0;
43     }
44 }cmp;
45 int size(node *o){return o?o->s:0;}
46 std::vector<node*> A;
47 node* build(int k,int l,int r){
48     if(l>r)return 0;
49     if(k==kd)k=0;
50     int mid=(l+r)/2;
51     cmp.sort_id=k;
52     std::nth_element(A.begin()+l,A.begin()+mid,A.begin()+r+1,cmp);
53     node *ret=A[mid];
54     ret->l=build(k+1,l,mid-1);
55     ret->r=build(k+1,mid+1,r);
56     ret->up();
57     return ret;
58 }
59 bool isbad(node*o){
60     return size(o->l)>alpha*o->s||size(o->r)>alpha*o->s;
61 }
62 void flatten(node *u,typename std::vector<node*>::iterator &it){
63     if(!u)return;
64     flatten(u->l,it);
65     *it=u;
66     flatten(u->r,++it);
67 }
68 void rebuild(node*&u,int k){
69     if(!((int)A.size()<u->s)A.resize(u->s);
70     typename std::vector<node*>::iterator it=A.begin();
71     flatten(u,it);
72     u=build(k,0,u->s-1);
73 }
74 bool insert(node*&u,int k,const point &x,int dep){
75     if(!u){
76         u=new node(x);
77         return dep<=0;
78     }
79     ++u->s;
80     cmp.sort_id=k;
81     if(insert(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x,dep-1)){
82         if(!isbad(u))return 1;
83         rebuild(u,k);
84     }
85     return 0;
86 }
87 node *findmin(node*o,int k){
88     if(!o)return 0;
89     if(cmp.sort_id==k)return o->l?findmin(o->l,(k+1)%kd):o;
90     *l=findmin(o->l,(k+1)%kd);
91     node *r=findmin(o->r,(k+1)%kd);

```

```

92     if(l&&!r)return cmp(l,o)?l:o;
93     if(!l&&r)return cmp(r,o)?r:o;
94     if(!l&&!r)return 0;
95     if(cmp(l,r))return cmp(l,o)?l:o;
96     return cmp(r,o)?r:o;
97 }
98 bool erase(node *u,int k,const point &x){
99     if(!u)return 0;
100     if(u->pid==x){
101         if(u->r){
102             else if(u->l){
103                 u->r=u->l;
104                 u->l=0;
105             }else{
106                 delete u;
107                 u=0;
108                 return 1;
109             }
110             --u->s;
111             cmp.sort_id=k;
112             u->pid=findmin(u->r,(k+1)%kd)->pid;
113             return erase(u->r,(k+1)%kd,u->pid);
114         }
115         cmp.sort_id=k;
116         if(erase(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x)){
117             --u->s;return 1;
118         }else return 0;
119     }
120     T heuristic(const T h[])const{
121         T ret=0;
122         for(size_t i=0;i<kd;++i)ret+=h[i];
123         return ret;
124     }
125     int qM;
126     std::priority_queue<std::pair<T,point >> pQ;
127     void nearest(node *u,int k,const point &x,T *h,T &mndist){
128         if(u==0||heuristic(h)>mndist)return;
129         T dist=u->pid.dist(x),old=h[k];
130         /*mndist=std::min(mndist,dist);*/
131         if(dist<mndist){
132             pQ.push(std::make_pair(dist,u->pid));
133             if((int)pQ.size()==qM+1)
134                 mndist=pQ.top().first,pQ.pop();
135         }
136         if(x.d[k]<u->pid.d[k]){
137             nearest(u->l,(k+1)%kd,x,h,mndist);
138             h[k]=std::abs(x.d[k]-u->pid.d[k]);
139             nearest(u->r,(k+1)%kd,x,h,mndist);
140         }else{
141             nearest(u->r,(k+1)%kd,x,h,mndist);
142             h[k]=std::abs(x.d[k]-u->pid.d[k]);
143             nearest(u->l,(k+1)%kd,x,h,mndist);
144         }
145         h[k]=old;
146     }
147     std::vector<point>in_range;
148     void range(node *u,int k,const point&mi,const point&ma){
149         if(!u)return;
150         bool is=1;
151         for(int i=0;i<kd;++i)

```

```

152         if(u->pid.d[i]<mi.d[i]||ma.d[i]<u->pid.d[i]){
153             is=0;break;
154         }
155         if(is)in_range.push_back(u->pid);
156         if(mi.d[k]<u->pid.d[k])range(u->l,(k+1)%kd,mi,ma);
157         if(ma.d[k]>u->pid.d[k])range(u->r,(k+1)%kd,mi,ma);
158     }
159 public:
160     kd_tree(const T &INF,double a=0.75):root(0),alpha(a),loga(log2(1.0/a)),INF(INF),maxn(1){}
161     ~kd_tree(){delete root;}
162     void clear(){delete root;root=0,maxn=1;}
163     void build(int n,const point *p){
164         delete root,A.resize(maxn=n);
165         for(int i=0;i<n;++i)A[i]=new node(p[i]);
166         root=build(0,0,n-1);
167     }
168     void insert(const point &x){
169         insert(root,x,__lg(size(root))/loga);
170     }
171     if(root->s>maxn)maxn=root->s;
172     }
173     bool erase(const point &p){
174         bool d=erase(root,0,p);
175         if(root&&root->s<alpha*maxn)rebuild();
176         return d;
177     }
178     void rebuild(){
179         if(root)rebuild(root,0);
180         maxn=root->s;
181     }
182     T nearest(const point &x,int k){
183         qM=k;
184         T mndist=INF,h[kd]={};
185         nearest(root,0,x,h,mndist);
186         mndist=pQ.top().first;
187         pQ=std::priority_queue<std::pair<T,point >>();
188         return mndist;/*!^Jµ²x²kµ²Iµ²JZµ²*/
189     }
190     const std::vector<point> &range(const point&mi,const point&ma){
191         in_range.clear();
192         range(root,0,mi,ma);
193         return in_range;/*!^JGµ²miµ²maµ²J;µ²µ²Ivector*/
194     }
195     int size(){return root?root->s:0;}
196 };

```

2.3 kd_tree_replace_segment_tree

```

1 /*kd樹代替高維線段樹*/
2 struct node{
3     node *l,*r;
4     point pid,mi,ma;
5     int s;
6     int data;

```



```

7 node(const point &p,int d):l(0),r(0),pid(p
    ),mi(p),ma(p),s(1),data(d),dmin(d),
      dmax(d){}
8 void up(){
9     mi=ma=pid;
10    s=1;
11    if(l){
12        for(int i=0;i<kd;++i){
13            mi.d[i]=min(mi.d[i],l->mi.d[i]);
14            ma.d[i]=max(ma.d[i],l->ma.d[i]);
15        }
16        s+=l->s;
17    }
18    if(r){
19        for(int i=0;i<kd;++i){
20            mi.d[i]=min(mi.d[i],r->mi.d[i]);
21            ma.d[i]=max(ma.d[i],r->ma.d[i]);
22        }
23        s+=r->s;
24    }
25 }
26 void up2(){
27     //其他懶惰標記向上更新
28 }
29 void down(){
30     //其他懶惰標記下推
31 }
32 }*root;
33
34 /*檢查區間包含用的函數*/
35 inline bool range_include(node *o,const
    point &L,const point &R){
36     for(int i=0;i<kd;++i){
37         if(L.d[i]>o->ma.d[i]||R.d[i]<o->mi.d[i])
38             return 0;
39     }//只要(L,R)區間有和o的區間有交集就回傳
40     true
41     return 1;
42 }
43 inline bool range_in_range(node *o,const
    point &L,const point &R){
44     for(int i=0;i<kd;++i){
45         if(L.d[i]>o->mi.d[i]||o->ma.d[i]>R.d[i])
46             return 0;
47     }//如果(L,R)區間完全包含o的區間就回傳true
48     return 1;
49 }
50
51 inline bool point_in_range(node *o,const
    point &L,const point &R){
52     for(int i=0;i<kd;++i){
53         if(L.d[i]>o->pid.d[i]||R.d[i]<o->pid.d[i]
54             )return 0;
55     }//如果(L,R)區間完全包含o->pid這個點就回傳
56     true
57     return 1;
58 }
59
60 /*單點修改 · 以單點改值為例*/
61 void update(node *u,const point &x,int data,
    int k=0){
62     if(!u)return;
63     u->down();
64     if(u->pid==x){
65         u->data=data;

```

```

60     u->up2();
61     return;
62 }
63 cmp.sort_id=k;
64 update(cmp(x,u->pid)?u->l:u->r,x,data,(k
    +1)%kd);
65 u->up2();
66 }
67
68 /*區間修改*/
69 void update(node *o,const point &L,const
    point &R,int data){
70     if(!o)return;
71     o->down();
72     if(range_in_range(o,L,R)){
73         //區間懶惰標記修改
74         o->down();
75         return;
76     }
77     if(point_in_range(o,L,R)){
78         //這個點在(L,R)區間 · 但是他的左右子樹不
79         //一定在區間中
80         //單點懶惰標記修改
81         if(o->l&&range_include(o->l,L,R))update(o
            ->l,L,R,data);
82         if(o->r&&range_include(o->r,L,R))update(o
            ->r,L,R,data);
83         o->up2();
84     }
85
86 /*區間查詢 · 以總和為例*/
87 int query(node *o,const point &L,const point
    &R){
88     if(!o)return 0;
89     o->down();
90     if(range_in_range(o,L,R))return o->sum;
91     int ans=0;
92     if(point_in_range(o,L,R))ans+=o->data;
93     if(o->l&&range_include(o->l,L,R))ans+=
        query(o->l,L,R);
94     if(o->r&&range_include(o->r,L,R))ans+=
        query(o->r,L,R);
95     return ans;
96 }

```

2.4 persistent_segment_tree.cpp

```

1 #include<bits/stdc++.h>//POJ 2104
2 using namespace std;
3 struct node{
4     int l,r;
5     int data;
6     node(int l,int r,int d):l(l),r(r),data(d)
7     {}
8 };
9 vector<node> nds;
10 inline void up(int o,int l,int r){
11     nds[o].data=nds[l].data+nds[r].data;
12 }
13 inline int new_node(int l,int r,int d){
14     nds.push_back(node(l,r,d));

```

```

14     return nds.size()-1;
15 }
16 inline int new_node(const node &nd){
17     nds.push_back(nd);
18     return nds.size()-1;
19 }
20 int build_tree(int l,int r){
21     int nd=new_node(-1,-1,0);
22     if(l==r)return nd;
23     int mid=(l+r)/2;
24     int L=build_tree(l,mid);//執行時vector會被
25     重構
26     int R=build_tree(mid+1,r);//一定要這樣寫
27     nds[nd].l=L;
28     nds[nd].r=R;
29     //up(nd,L,R);
30     return nd;
31 }
32 int insert(int l,int r,int rt,int x,int d){
33     if(x<l||r<x)return rt;
34     int nd=new_node(nds[rt]);
35     if(l==r&&l==x)nds[nd].data+=d;
36     else{
37         int mid=(l+r)/2;
38         int L=insert(l,mid,nds[nd].l,x,d);
39         int R=insert(mid+1,r,nds[nd].r,x,d);
40         nds[nd].l=L;
41         nds[nd].r=R;
42         up(nd,L,R);
43     }
44     return nd;
45 }
46 inline int cal(int L,int R){
47     return nds[R].data-nds[L].data;
48 }
49 int find(int l,int r,int L,int R,int k){
50     if(l==r)return l;
51     int mid=(l+r)/2;
52     int add=cal(nds[L].l,nds[R].l);
53     if(k<=add)return find(l,mid,nds[L].l,nds[R]
        .l,k);
54     return find(mid+1,r,nds[L].r,nds[R].r,k-
        add);
55 }
56 int n,m;
57 int s[100005];
58 int root[100005];
59 int main(){
60     while(~scanf("%d",&n,&m)){
61         nds.clear();
62         vector<int> lsh;
63         for(int i=1;i<=n;++i){
64             scanf("%d",&s[i]);
65             lsh.push_back(s[i]);
66         }
67         sort(lsh.begin(),lsh.end());
68         lsh.resize(unique(lsh.begin(),lsh.end())
            -lsh.begin());
69         int N=(int)lsh.size()-1;
70         root[0]=build_tree(0,N);
71         for(int i=1;i<=n;++i){
72             s[i]=lower_bound(lsh.begin(),lsh.end()
73                 ,s[i])-lsh.begin();
74             root[i]=insert(0,N,root[i-1],s[i],1);
75         }

```

```

74     while(m--){
75         int a,b,k;
76         scanf("%d%d%d",&a,&b,&k);
77         int res=find(0,N,root[a-1],root[b],k);
78         printf("%d\n",lsh[res]);
79     }
80     return 0;
81 }
82 }

```

2.5 reference_point.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 template<typename T>
4 struct _RefCounter{
5     T data;
6     int ref;
7     _RefCounter(const T&d=0):data(d),ref(0){}
8 };
9 template<typename T>
10 struct reference_pointer{
11     _RefCounter<T> *p;
12     T *operator->(){return &(*p).data;}
13     T &operator*(){return p->data;}
14     operator int(){return(int)(long long)p;}
15     reference_pointer&operator=(const
        reference_pointer &t){
16         if(p&&--(*p).ref==0)delete p;
17         p=t.p;
18         p&&+(*p).ref;
19         return*this;
20 }
21 reference_pointer(_RefCounter<T> *t=0):p(t)
22 {}
23 p&&+(*p).ref;
24 reference_pointer(const reference_pointer
    &t):p(t.p){
25     p&&+(*p).ref;
26 }
27 ~reference_pointer(){
28     if(p&&--(*p).ref==0)delete p;
29 }
30 };
31 template<typename T>
32 inline const reference_pointer<T>
    new_reference(const T&nd){
33     return reference_pointer<T>(new
        _RefCounter<T>(nd));
34 }
35 struct P{
36     int a,b;
37     P(int A,int B):a(A),b(B){}
38 }p(2,3);
39 int main(){
40     reference_pointer<int> b=new_reference(int
        (5));
41     reference_pointer<int> a=new_reference(*b)
42     ;
43     reference_pointer<P> c=new_reference(p);
44     return 0;
45 }

```

2.6 skew_heap.cpp

```

1 template<typename T,typename _Compare=std::
2     less<T> >
3 class skew_heap{
4     private:
5         struct node{
6             T data;
7             node *l,*r;
8             node(const T&d):data(d),l(0),r(0){}
9             ~node(){delete l,delete r;}
10        }*root;
11        int _size;
12        _Compare cmp;
13        node *merge(node *a,node *b){
14            if(!a||!b)return a?a:b;
15            if(cmp(a->data,b->data))return merge(b,
16                a);
17            node *t=a->r;
18            a->r=a->l;
19            a->l=merge(b,t);
20            return a;
21        }
22        public:
23        skew_heap():root(0),_size(0){}
24        ~skew_heap(){delete root;}
25        void clear(){delete root,root=0,_size
26            =0;}
27        void join(skew_heap &o){
28            root=merge(root,o.root);
29            o.root=0;
30            _size+=o._size;
31            o._size=0;
32        }
33        void swap(skew_heap &o){
34            node *t=root;
35            root=o.root;
36            o.root=t;
37            int st=_size;
38            _size=o._size;
39            o._size=st;
40        }
41        void push(const T&data){
42            _size++;
43            root=merge(root,new node(data));
44        }
45        void pop(){
46            if(_size)_size--;
47            node *tmd=merge(root->l,root->r);
48            root->l=root->r=0;
49            delete root;
50            root=tmd;
51        }
52        const T& top(){return root->data;}
53        int size(){return _size;}
54        bool empty(){return !_size;}
55    };

```

2.7 split_merge.cpp

```

1 void split(node *o,node *&a,node *&b,int k){
2     if(!o)a=b=0;

```

```

3     else{
4         //o=new node(*o);
5         o->down();
6         if(k<=size(o->l)){
7             b=o;
8             split(o->l,a,b->l,k);
9         }else{
10            a=o;
11            split(o->r,a->r,b,k-size(o->l)-1);
12        }
13        o->up();
14    }
15 }
16 node *merge(node *a,node *b){
17     if(!a||!b)return a?a:b;
18     static int x;
19     if(x+%(a->s+b->s)<a->s){
20         //a=new node(*a);
21         a->down();
22         a->r=merge(a->r,b);
23         a->up();
24         return a;
25     }else{
26         //b=new node(*b);
27         b->down();
28         b->l=merge(a,b->l);
29         b->up();
30         return b;
31     }
32 }

```

2.8 treap.cpp

```

1 template<typename T>
2 class treap{
3     private:
4         struct node{
5             T data;
6             unsigned fix;
7             int s;
8             node *ch[2];
9             node(const T&d):data(d),s(1){}
10            node():s(0){ch[0]=ch[1]=this;}
11        }*nil,*root;
12        unsigned x;
13        unsigned ran()const{return x=x*0xdefaced+1;}
14        void rotate(node *&a,bool d){
15            node *b=a;
16            a=a->ch[!d];
17            a->s=b->s;
18            b->ch[!d]=a->ch[d];
19            a->ch[d]=b;
20            b->s=b->ch[0]->s+b->ch[1]->s+1;
21        }
22        void insert(node *&o,const T &data){
23            if(!o->s){
24                o=new node(data),o->fix=ran();
25                o->ch[0]=o->ch[1]=nil;
26            }else{
27                o->s++;
28                bool d=o->data<data;
29                insert(o->ch[d],data);

```

```

30            if(o->ch[d]->fix>o->fix)rotate(o,!d)
31            ;
32        }
33        node *merge(node *a,node *b){
34            if(!a->s||!b->s)return a->s?a:b;
35            if(a->fix>b->fix){
36                a->ch[1]=merge(a->ch[1],b);
37                a->s=a->ch[0]->s+a->ch[1]->s+1;
38                return a;
39            }else{
40                b->ch[0]=merge(a,b->ch[0]);
41                b->s=b->ch[0]->s+b->ch[1]->s+1;
42                return b;
43            }
44        }
45        bool erase(node *&o,const T &data){
46            if(!o->s)return 0;
47            if(o->data==data){
48                node *t=o;
49                o=merge(o->ch[0],o->ch[1]);
50                delete t;
51                return 1;
52            }
53            if(erase(o->ch[o->data<data],data)){
54                o->s--;return 1;
55            }else return 0;
56        }
57        void clear(node *&o){
58            if(o->s)clear(o->ch[0]),clear(o->ch
59                [1]),delete o;
60        }
61        public:
62        treap(unsigned s=20150119):nil(new node
63            ,root(nil),x(s){}
64        ~treap(){clear(root),delete nil;}
65        void clear(){clear(root),root=nil;}
66        void insert(const T &data){
67            insert(root,data);
68        }
69        bool erase(const T &data){
70            return erase(root,data);
71        }
72        bool find(const T&data){
73            for(node *o=root;o->s;){
74                if(o->data==data)return 1;
75                else o=o->ch[o->data<data];
76            }
77            return 0;
78        }
79        int rank(const T&data){
80            int cnt=0;
81            for(node *o=root;o->s;){
82                if(o->data<data)cnt+=o->ch[0]->s+1,o=
83                    o->ch[1];
84                else o=o->ch[0];
85            }
86            return cnt;
87        }
88        const T&kth(int k){
89            for(node *o=root;;){
90                if(k<=o->ch[0]->s)o=o->ch[0];
91                else if(k==o->ch[0]->s+1)return o->
92                    data;
93                else k-=o->ch[0]->s+1,o=o->ch[1];
94            }
95        }
96        const T&operator[](int k){
97            return kth(k);

```

```

91        }
92        const T&preorder(const T&data){
93            node *x=root,*y=0;
94            while(x->s){
95                if(x->data<data)y=x,x=x->ch[1];
96                else x=x->ch[0];
97                if(y)return y->data;
98                return data;
99            }
100        }
101        const T&successor(const T&data){
102            node *x=root,*y=0;
103            while(x->s){
104                if(data<x->data)y=x,x=x->ch[0];
105                else x=x->ch[1];
106                if(y)return y->data;
107                return data;
108            }
109        }
110        int size(){return root->s;}
111    };

```

2.9 操作分治.cpp

```

1 void dq(int l,int r){
2     if(l==r)return;
3     int mid=(l+r)/2;
4     dq(l,mid);
5     處理[l,mid]的操作對[mid+1,r]的影響
6     dq(mid+1,r);
7 }

```

2.10 整體二分.cpp

```

1 void BS(int l,int r,vector<Item> &vs){
2     //答案該<L會有的已經做完了
3     if(l==r)整個vs的答案=1;////?????
4     int mid=(l+r)/2;
5     do_thing(l,mid);//做答案<=mid會做的事
6     vector<Item> left=vs裡滿足的;
7     vector<Item> right=vs-left;
8     undo_thing(l,mid);
9     BS(l,mid,left);
10    do_thing(l,mid);
11    BS(mid+1,r,right);////?????
12 }

```

3 default

3.1 debug.cpp

```

1 #ifndef Jinkala
2 #define debug(...) {\
3     fprintf(stderr,"%s - %d : (%s) = ",
4         __PRETTY_FUNCTION__,__LINE__,__
5         __VA_ARGS__);

```

```

4  _DO(__VA_ARGS__); \
5  }
6  template<typename I> void _DO(I&&x){cerr<<x
    <<endl;}
7  template<typename I,typename...T> void _DO(I
    &&x,T&&...tail){cerr<<x<<" ";_DO(tail
    ...);}
8  #else
9  #define debug(...)
10 #endif

```

3.2 IncStack.cpp

```

1  //Magic
2  #pragma GCC optimize "Ofast"
3  //stack resize,change esp to rsp if 64-bit
    system
4  asm("mov %0,%esp\n" ::"g"(mem+1000000));
5  //linux stack resize
6  #include<sys/resource.h>
7  void increase_stack(){
8      const rlim_t ks=64*1024*1024;
9      struct rlimit rl;
10     int res=getrlimit(RLIMIT_STACK,&rl);
11     if(!res&&rl.rlim_cur<ks){
12         rl.rlim_cur=ks;
13         res=setrlimit(RLIMIT_STACK,&rl);
14     }
15 }

```

3.3 input.cpp

```

1  inline int read(){
2      int x=0; bool f=0; char c=getchar();
3      while(ch<'0' || '9'<ch)f|=ch=='-',ch=
        getchar();
4      while('0'<=ch&&ch<='9')x=x*10-'0'+ch,ch=
        getchar();
5      return f?-x:x;
6  }
7  inline int read(){//輸入不可以包含 : ; < > =
    ?
8      int x=0; bool f=0; char c=getchar();
9      while((c>>4)&3!=3)f|=ch=='-',c=getchar()
        ;
10     while((c>>4)&3==3)x=x*10-'0'+c,c=getchar
        ();
11     return f?-x:x;
12 }

```

4 Flow

4.1 dinic.cpp

```

1  #define MAXN 105
2  #define INF INT_MAX
3  int n; /*number of nodes*/
4  int level[MAXN], cur[MAXN]; /*Layer, current
    arc*/
5  struct edge{
6      int v, pre;
7      long long cap, flow, r;
8      edge(int v, int pre, long long cap):v(v), pre
        (pre), cap(cap), flow(0), r(cap){}
9  };
10 int g[MAXN];
11 std::vector<edge> e;
12 inline void init(){
13     memset(g, -1, sizeof(int)*(n+1));
14     e.clear();
15 }
16 inline void add_edge(int u, int v, long long
    cap, bool directed=false){
17     e.push_back(edge(v, g[u], cap));
18     g[u]=e.size()-1;
19     e.push_back(edge(u, g[v], directed?0:cap));
20     g[v]=e.size()-1;
21 }
22 inline int bfs(int s, int t){
23     memset(level, 0, sizeof(int)*(n+1));
24     memcpy(cur, g, sizeof(int)*(n+1));
25     std::queue<int> q;
26     q.push(s);
27     level[s]=1;
28     while(q.size()){
29         int u=q.front(); q.pop();
30         for(int i=g[u]; ~i; i=e[i].pre){
31             if(!level[e[i].v] && e[i].r){
32                 level[e[i].v]=level[u]+1;
33                 q.push(e[i].v);
34                 if(e[i].v==t) return 1;
35             }
36         }
37     }
38     return 0;
39 }
40 long long dfs(int u, int t, long long cur_flow
    =INF){
41     if(u==t || !cur_flow) return cur_flow;
42     long long df, tf=0;
43     for(int i=g[u]; ~i; i=e[i].pre){
44         if(level[e[i].v]==level[u]+1 && e[i].r){
45             if(df=dfs(e[i].v, t, std::min(cur_flow, e
                [i].r))){
46                 e[i].flow+=df;
47                 e[i^1].flow-=df;
48                 e[i].r-=df;
49                 e[i^1].r+=df;
50                 tf+=df;
51                 if(! (cur_flow-=df)) break;
52             }
53         }
54     }
55     if(!df) level[u]=0;
56     return tf;
57 }
58 inline long long dinic(int s, int t, bool
    clean=true){
59     if(clean){
60         for(size_t i=0; i<e.size(); ++i){

```

```

61         e[i].flow=0;
62         e[i].r=e[i].cap;
63     }
64 }
65 long long ans=0;
66 while(bfs(s, t)) ans+=dfs(s, t);
67 return ans;
68 }

```

4.2 ISAP.cpp

```

1  #define MAXN 105
2  #define INF INT_MAX
3  int n; /*點數*/
4  int d[MAXN], gap[MAXN], cur[MAXN];
5  /*層次、gap[i]=層次為i的點之個數、當前弧優化
    */
6  struct edge{
7      int v, pre;
8      long long cap, flow, r;
9      edge(int v, int pre, long long cap):v(v), pre
        (pre), cap(cap), flow(0), r(cap){}
10 };
11 int g[MAXN];
12 std::vector<edge> e;
13 inline void init(){
14     memset(g, -1, sizeof(int)*(n+1));
15     e.clear();
16 }
17 inline void add_edge(int u, int v, long long
    cap, bool directed=false){
18     e.push_back(edge(v, g[u], cap));
19     g[u]=e.size()-1;
20     e.push_back(edge(u, g[v], directed?0:cap));
21     g[v]=e.size()-1;
22 }
23 long long dfs(int u, int s, int t, long long
    cur_flow=INF){
24     if(u==t) return cur_flow;
25     long long tf=cur_flow, df;
26     for(int i=g[u]; ~i; i=e[i].pre){
27         if(e[i].r && d[u]==d[e[i].v]+1){
28             df=dfs(e[i].v, s, t, std::min(tf, e[i].r))
                ;
29             e[i].flow+=df;
30             e[i^1].flow-=df;
31             e[i].r-=df;
32             e[i^1].r+=df;
33             if(! (tf-=df) || d[s]==n) return cur_flow-
                tf;
34         }
35     }
36     int minh=n;
37     for(int i=g[u]; ~i; i=e[i].pre){
38         if(e[i].r && d[e[i].v]<minh) minh=d[e[i].v]
            ;
39     }
40     if(!--gap[d[u]]) d[s]=n;
41     else ++gap[d[u]=++minh];
42     return cur_flow-tf;
43 }
44 inline long long isap(int s, int t, bool clean
    =true){

```

```

45     memset(d, 0, sizeof(int)*(n+1));
46     memset(gap, 0, sizeof(int)*(n+1));
47     memcpy(cur, g, sizeof(int)*(n+1));
48     if(clean){
49         for(size_t i=0; i<e.size(); ++i){
50             e[i].flow=0;
51             e[i].r=e[i].cap;
52         }
53     }
54     long long max_flow=0;
55     for(gap[0]=n; d[s]<n; ) max_flow+=dfs(s, s, t);
56     return max_flow;
57 }

```

4.3 MinCostMaxFlow.cpp

```

1  #define MAXN 440
2  #define INF 999999999
3  struct edge{
4      int v, pre;
5      int cap, cost;
6      edge(int v, int pre, int cap, int cost):v(v),
        pre(pre), cap(cap), cost(cost){}
7  };
8  int n, s, t;
9  int dis[MAXN], piS, ans;
10 bool vis[MAXN];
11 std::vector<edge> e;
12 int g[MAXN];
13 inline void init(){
14     memset(g, -1, sizeof(int)*n);
15     e.clear();
16 }
17 inline void add_edge(int u, int v, int cost,
    int cap, bool directed=false){
18     e.push_back(edge(v, g[u], cap, cost));
19     g[u]=e.size()-1;
20     e.push_back(edge(u, g[v], directed?0:cap, -
        cost));
21     g[v]=e.size()-1;
22 }
23 int augment(int u, int cur_flow){
24     if(u==T || !cur_flow) return ans+=piS*
        cur_flow, cur_flow;
25     vis[u]=1;
26     int r=cur_flow, d;
27     for(int i=g[u]; ~i; i=e[i].pre){
28         if(e[i].cap && !e[i].cost && vis[e[i].v]){
29             d=augment(e[i].v, std::min(r, e[i].cap))
                ;
30             e[i].cap-=d;
31             e[i^1].cap+=d;
32             if(! (r-=d)) break;
33         }
34     }
35     return cur_flow-r;
36 }
37 inline bool modlabel(){
38     for(int i=0; i<n; ++i) dis[i]=INF;
39     dis[T]=0;
40     static std::deque<int> q;
41     q.push_back(T);
42     while(q.size()){

```

```

43 int u=q.front();
44 q.pop_front();
45 int dt;
46 for(int i=g[u];~i;i=e[i].pre){
47     if(e[i^1].cap&&(dt=dis[u]-e[i].cost)<
48         dis[e[i].v]){
49         if((dis[e[i].v]=dt)<=dis[q.size() ? q.
50             front():S]){
51             q.push_front(e[i].v);
52         }else q.push_back(e[i].v);
53     }
54 }
55 for(int u=0;u<n;u++){
56     for(int i=g[u];~i;i=e[i].pre){
57         e[i].cost+=dis[e[i].v]-dis[u];
58     }
59 }
60 piS+=dis[S];
61 return dis[S]<INF;
62 }
63 inline int mincost(){
64     piS=ans=0;
65     while(modlabel()){
66         do memset(vis,0,sizeof(bool)*n);
67         while(augment(S,INF));
68     }
69     return ans;

```

5 Graph

5.1 Arborescence_EV.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct node {
5     int from, to, cost;
6     node(int from=0,int to=0,int cost=0):
7         from(from),to(to),cost(cost){};
8 } edge[M];
9
10 int m, n, m, c;
11 int far[N], In[N], ID[N], vis[N];
12
13 bool MST(int cost,int n,int root)
14 {
15     long long int ans=0;
16     while(true)
17     {
18         for(int i=0;i<n;u++){
19             IN[i].first = INF;
20         }
21         for(int i=0;i<m;u++){
22             if(edge[i].from!=edge[i].to)
23                 IN[edge[i].to] = min(IN[edge[i].to],make_pair(edge[i].cost,edge[i].from));
24         }
25         for(int i=0;i<n;u++){
26             if(i!=root && IN[i].first==INF)

```

```

23         return false; // NO
24         Arborescence
25
26         int cntnode = 0;
27         memset(ID,-1,sizeof(ID));
28         memset(vis,-1,sizeof(vis));
29         In[root] = 0;
30         for(int i=0;i<n;u++){
31             first;
32             for(int i=0;i<n;u++){
33                 int x;
34                 for(x=i;vis[x]!=i&&ID[x]==-1&&x
35                     !=root;x=IN[x].second)
36                     vis[x] = i;
37                 if(ID[x]==-1 && x!=root) {
38                     for(int i=IN[x].second;u!=x;
39                         u=IN[u].second)
40                         ID[u] = cntnode;
41                     ++cntnode;
42                 }
43             }
44             if(cntnode==0) break; // END
45
46             for(int i=0;i<n;u++){
47                 if(ID[i]==-1)
48                     ID[i] = cntnode++;
49             }
50
51             for(int i=0;i<m;u++){
52                 int v = edge[i].to;
53                 edge[i].from = ID[edge[i].from];
54                 edge[i].to = ID[edge[i].to];
55                 if(edge[i].from!=edge[i].to)
56                     edge[i].cost -= IN[edge[i].to].first;
57             }
58             n=cntnode;
59             root=ID[root];
60         }
61         return ans<=cost;
62     }

```

5.2 Augmenting_Path.cpp

```

1 #define MAXN1 505
2 #define MAXN2 505
3 int n1,n2; /*n1個點連向n2個點*/
4 int match[MAXN2]; /*每個屬於n2的點匹配了哪個點*/
5
6 vector<int> g[MAXN1]; /*圖*/
7 bool vis[MAXN2]; /*是否走訪過*/
8 bool dfs(int u){
9     for(size_t i=0;i<g[u].size();u++){
10         int v=g[u][i];
11         if(vis[v])continue;
12         vis[v]=1;
13         if(match[v]==-1||dfs(match[v])){
14             match[v]=u;
15             return 1;
16         }
17     }
18     return 0;

```

```

19 inline int max_match(){
20     int ans=0;
21     memset(match,-1,sizeof(int)*n2);
22     for(int i=0;i<n1;u++){
23         memset(vis,0,sizeof(bool)*n2);
24         if(dfs(i))++ans;
25     }
26     return ans;
27 }

```

5.3 Augmenting_Path_multiple

```

1 #define MAXN1 1005
2 #define MAXN2 505
3 int n1,n2; /*n1個點連向n2個點，其中n2個點可以匹配很多邊*/
4 vector<int> g[MAXN1]; /*圖*/
5 int c[MAXN2]; /*每個屬於n2點最多可以接受幾條匹配邊*/
6 vector<int> match_list[MAXN2]; /*每個屬於n2的點匹配了那些點*/
7
8 bool vis[MAXN2]; /*是否走訪過*/
9 bool dfs(int u){
10     for(size_t i=0;i<g[u].size();u++){
11         int v=g[u][i];
12         if(vis[v])continue;
13         vis[v]=true;
14         if((int)match_list[v].size()<c[v]){
15             match_list[v].push_back(u);
16             return true;
17         }else{
18             for(size_t j=0;j<match_list[v].size();u++){
19                 ++j;
20                 int next_u=match_list[v][j];
21                 if(dfs(next_u)){
22                     match_list[v][j]=u;
23                     return true;
24                 }
25             }
26         }
27     }
28     return false;
29 }
30
31 inline int max_match(){
32     for(int i=0;i<n2;u++){
33         match_list[i].clear();
34     }
35     int cnt=0;
36     for(int u=0;u<n1;u++){
37         memset(vis,0,sizeof(bool)*n2);
38         if(dfs(u))++cnt;
39     }
40     return cnt;

```

5.4 blossom_matching.cpp

```

1 #define MAXN 505
2 vector<int>g[MAXN];

```

```

3 int pa[MAXN],match[MAXN],st[MAXN],S[MAXN],v[
4     MAXN];
5 int t,n;
6 inline int lca(int x,int y){
7     for(++t;swap(x,y)){
8         if(x==0)continue;
9         if(v[x]==t)return x;
10        v[x]=t;
11        x=st[pa[match[x]]];
12    }
13 }
14 #define qpush(x) q.push(x),S[x]=0
15 inline void flower(int x,int y,int l,queue<
16     int> &q){
17     while(st[x]!=1){
18         pa[x]=y;
19         if(S[y==match[x]]==1)qpush(y);
20         st[x]=st[y]=1,x=pa[y];
21     }
22 }
23 inline bool bfs(int x){
24     for(int i=1;i<n;u++){
25         st[i]=i;
26         memset(S+1,-1,sizeof(int)*n);
27         queue<int>q;qpush(x);
28         while(q.size()){
29             x=q.front(),q.pop();
30             for(size_t i=0;i<g[x].size();u++){
31                 int y=g[x][i];
32                 if(S[y]==-1){
33                     pa[y]=x,S[y]=1;
34                     if(!match[y]){
35                         for(int lst;x=y,lst,x=pa[y])
36                             lst=match[x],match[x]=y,match[y]=x;
37                     }
38                     return 1;
39                 }
40                 qpush(match[y]);
41             }
42         }
43         if(!S[y]&&st[y]!=st[x]){
44             int l=lca(y,x);
45             flower(y,x,l,q),flower(x,y,l,q);
46         }
47     }
48     return 0;
49 }
50
51 inline int blossom(){
52     int ans=0;
53     for(int i=1;i<n;u++){
54         if(!match[i]&&bfs(i))++ans;
55     }
56     return ans;

```

5.5 graphISO.cpp

```

1 const int MAXN=1005,K=30; /*K要夠大*/
2 const long long A=3,B=11,C=2,D=19,P=0
3     xdefaced;
4 long long f[K+1][MAXN];
5 vector<int> g[MAXN],rg[MAXN];
6 int n;
7 inline void init(){
8     for(int i=0;i<n;u++){
9         f[0][i]=1;

```



```

9   g[i].clear();
10  rg[i].clear();
11  }
12  }
13  inline void add_edge(int u,int v){
14  g[u].push_back(v);
15  rg[v].push_back(u);
16  }
17  inline long long point_hash(int u){//O(N)
18  for(int t=1;t<=K;++t){
19  for(int i=0;i<n;++i){
20  f[t][i]=f[t-1][i]*A%P;
21  for(int j:g[i])f[t][i]=(f[t][i]+f[t-1][j]*B%P)%P;
22  for(int j:rg[i])f[t][i]=(f[t][i]+f[t-1][j]*C%P)%P;
23  if(i==u)f[t][i]+=D;//如果圖太大的話，
    把這行刪掉，執行一次後f[K]就會是所有
    有點的答案
24  f[t][i]=P;
25  }
26  }
27  return f[K][u];
28  }
29  inline vector<long long> graph_hash(){
30  vector<long long> ans;
31  for(int i=0;i<n;++i)ans.push_back(
    point_hash(i));//O(N^2)
32  sort(ans.begin(),ans.end());
33  return ans;
34  }

```

5.6 KM.cpp

```

1  #define MAXN 100
2  int n;
3  int g[MAXN][MAXN],lx[MAXN],ly[MAXN],slack_y[
    MAXN];
4  int match_y[MAXN];
5  bool vx[MAXN],vy[MAXN];//要保證g是完全二分圖
6  bool dfs(int x,bool adjust=1){//DFS找增廣
    路，is=1表示要交換邊
7  if(vx[x])return 0;
8  vx[x]=1;
9  for(int y=0;y<n;++y){
10  if(vy[y])continue;
11  int t=lx[x]+ly[y]-g[x][y];
12  if(t==0){
13  vy[y]=1;
14  if(match_y[y]==-1||dfs(match_y[y],
    adjust)){
15  if(adjust)match_y[y]=x;
16  return 1;
17  }
18  }else if(slack_y[y]>t)slack_y[y]=t;
19  }
20  return 0;
21  }
22  inline int km(){
23  memset(ly,0,sizeof(int)*n);
24  memset(match_y,-1,sizeof(int)*n);
25  for(int x=0;x<n;++x){

```

```

26  lx[x]=0;
27  for(int y=0;y<n;++y){
28  lx[x]=max(lx[x],g[x][y]);
29  }
30  }
31  for(int x=0;x<n;++x){
32  for(int y=0;y<n;++y)slack_y[y]=INT_MAX;
33  memset(vx,0,sizeof(bool)*n);
34  memset(vy,0,sizeof(bool)*n);
35  if(dfs(x))continue;
36  bool flag=1;
37  while(flag){
38  int cut=INT_MAX;
39  for(int y=0;y<n;++y){
40  if(!vy[y]&&cut>slack_y[y])cut=
    slack_y[y];
41  }
42  for(int j=0;j<n;++j){
43  if(vx[j])lx[j]-=cut;
44  if(vy[j])ly[j]+=cut;
45  else slack_y[j]-=cut;
46  }
47  for(int y=0;y<n;++y){
48  if(!vy[y]&&slack_y[y]==0){
49  vy[y]=1;
50  if(match_y[y]==-1||dfs(match_y[y]
    ],0)){
51  flag=0;//測試成功，有增廣路
52  break;
53  }
54  }
55  }
56  }
57  memset(vx,0,sizeof(bool)*n);
58  memset(vy,0,sizeof(bool)*n);
59  dfs(x);//最後要記得將邊翻反轉
60  }
61  int ans=0;
62  for(int y=0;y<n;++y)ans+=g[match_y[y]][y];
63  return ans;
64  }

```

5.7 MaximumClique.cpp

```

1  struct MaxClique{
2  static const int MAXN=105;
3  int N,ans;
4  int g[MAXN][MAXN],dp[MAXN],stk[MAXN][MAXN
    ];
5  int sol[MAXN],tmp[MAXN];//sol[0~ans-1]為答
    案
6  void init(int n){
7  N=n;//0-base
8  memset(g,0,sizeof(g));
9  }
10 void add_edge(int u,int v){
11 g[u][v]=g[v][u]=1;
12 }
13 int dfs(int ns,int dep){
14 if(!ns){
15 if(dep>ans){
16 ans=dep;

```

```

17 memcpy(sol,tmp,sizeof tmp);
18 return 1;
19 }else return 0;
20 }
21 for(int i=0;i<ns;++i){
22 if(dep+ns-i<=ans)return 0;
23 int u=stk[dep][i],cnt=0;
24 if(dep+dp[u]<=ans)return 0;
25 for(int j=i+1;j<ns;++j){
26 int v=stk[dep][j];
27 if(g[u][v])stk[dep+1][cnt++]=v;
28 }
29 tmp[dep]=u;
30 if(dfs(cnt,dep+1))return 1;
31 }
32 return 0;
33 }
34 int clique(){
35 int u,v,ns;
36 for(ans=0,u=N-1;u>=0;--u){
37 for(ns=0,tmp[0]=u,v=u+1;v<N;++v)
38 if(g[u][v])stk[1][ns++]=v;
39 dfs(ns,1),dp[u]=ans;
40 }
41 return ans;
42 }
43 };

```

5.8 Minimum_General_Weighted

```

1 struct Graph {
2 // Minimum General Weighted Matching (
    Perfect Match) 0-base
3 static const int MXN = 105;
4
5 int n, edge[MXN][MXN];
6 int match[MXN],dis[MXN],onstk[MXN];
7 vector<int> stk;
8
9 void init(int _n) {
10 n = _n;
11 for (int i=0; i<n; i++)
12 for (int j=0; j<n; j++)
13 edge[i][j] = 0;
14 }
15 void add_edge(int u, int v, int w) {
16 edge[u][v] = edge[v][u] = w;
17 }
18 bool SPFA(int u){
19 if (onstk[u]) return true;
20 stk.push_back(u);
21 onstk[u] = 1;
22 for (int v=0; v<n; v++){
23 if (u != v && match[u] != v && !onstk[
    v]){
24 int m = match[v];
25 if (dis[m] > dis[u] - edge[v][m] +
    edge[u][v]){
26 dis[m] = dis[u] - edge[v][m] +
    edge[u][v];
27 onstk[v] = 1;
28 stk.push_back(v);
29 if (SPFA(m)) return true;

```

```

30 stk.pop_back();
31 onstk[v] = 0;
32 }
33 }
34 }
35 onstk[u] = 0;
36 stk.pop_back();
37 return false;
38 }
39
40 int solve() {
41 // find a match
42 for (int i=0; i<n; i+=2){
43 match[i] = i+1;
44 match[i+1] = i;
45 }
46 for(;;){
47 int found = 0;
48 for (int i=0; i<n; i++)
49 dis[i] = onstk[i] = 0;
50 for (int i=0; i<n; i++){
51 stk.clear();
52 if (!onstk[i] && SPFA(i)){
53 found = 1;
54 while (stk.size()>=2){
55 int u = stk.back(); stk.pop_back
    ();
56 int v = stk.back(); stk.pop_back
    ();
57 match[u] = v;
58 match[v] = u;
59 }
60 }
61 }
62 if (!found) break;
63 }
64 int ret = 0;
65 for (int i=0; i<n; i++)
66 ret += edge[i][match[i]];
67 ret /= 2;
68 return ret;
69 }
70 }graph;

```

5.9 Rectilinear_Steiner_tree.cpp

```

1 //平面曼哈頓最小生成樹構造圖(去除非必要邊)
2 #include<vector>
3 #include<algorithm>
4 #define T int
5 #define INF 0x3f3f3f3f
6 struct point{
7 T x,y;
8 int id;//每個點的編號都要不一樣，從0開始編
    號
9 point(){
10 T dist(const point &p)const{
11 return std::abs(x-p.x)+std::abs(y-p.y);
12 }
13 };
14 inline bool cmpx(const point &a,const point
    &b){

```

```

15 return a.x<b.x||(a.x==b.x&&a.y<b.y);
16 }
17 struct edge{
18     int u,v;
19     T cost;
20     edge(int u,int v,const T&c):u(u),v(v),cost
21         (c){}
22     bool operator<(const edge&e)const{
23         return cost<e.cost;
24     };
25 struct bit_node{
26     T mi;
27     int id;
28     bit_node(const T&mi=INF,int id=-1):mi(mi),
29         id(id){}
30 };
31 std::vector<bit_node> bit;
32 inline void bit_update(int i,const T&data,
33     int id){
34     for(;i=i&(-i)){
35         if(data<bit[i].mi)bit[i]=bit_node(data,
36             id);
37     }
38 }
39 inline int bit_find(int i,int m){
40     bit_node x;
41     for(;i<m;i=i&(-i)){
42         if(bit[i].mi<x.mi)x=bit[i];
43     }
44     return x.id;
45 }
46 inline std::vector<edge> build_graph(int n,
47     point p[]){
48     std::vector<edge> e;//回傳的邊就可以用來求
49     最小生成樹
50     for(int dir=0;dir<4;dir++){//4種座標變換
51         if(dir%2){
52             for(int i=0;i<n;i++)std::swap(p[i].x,p
53                 [i].y);
54         }else if(dir==2){
55             for(int i=0;i<n;i++)p[i].x=-p[i].x;
56         }
57         std::sort(p,p+n,cmpx);
58         std::vector<T>ga(n),gb;
59         for(int i=0;i<n;i++)ga[i]=p[i].y-p[i].x;
60         gb=ga;
61         std::sort(gb.begin(),gb.end());
62         gb.resize(std::unique(gb.begin(),gb.end
63             ())-gb.begin());
64         int m=gb.size();
65         bit=std::vector<bit_node>(m+1);
66         for(int i=n-1;i>=0;--i){
67             int pos=std::lower_bound(gb.begin(),gb
68                 .end(),ga[i])-gb.begin()+1;
69             int ans=bit_find(pos,m);
70             if(~ans)e.push_back(edge(p[i].id,p[ans
71                 ].id,p[i].dist(p[ans])));
72             bit_update(pos,p[i].x+p[i].y,i);
73         }
74     }
75     return e;
76 }

```

5.10 treeISO.cpp

```

1 const int MAXN=100005;
2 const long long X=12327,P=0xdefaced;
3 vector<int> g[MAXN];
4 bool vis[MAXN];
5 long long dfs(int u){
6     vis[u]=1;
7     vector<long long> tmp;
8     for(auto v:g[u])if(!vis[v])tmp.push_back(
9         dfs(v));
10    if(tmp.empty())return 177;
11    long long ret=4931;
12    sort(tmp.begin(),tmp.end());
13    for(auto v:tmp)ret=((ret*X)^v)%P;
14    return ret;
15 }

```

5.11 一般圖最大權匹配.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define INF INT_MAX
4 #define MAXN 400
5 struct edge{
6     int u,v,w;
7     edge(){}
8     edge(int u,int v,int w):u(u),v(v),w(w){}
9 };
10 int n,n_x;
11 edge g[MAXN*2+1][MAXN*2+1];
12 int lab[MAXN*2+1];
13 int match[MAXN*2+1],slack[MAXN*2+1],st[MAXN
14     *2+1],pa[MAXN*2+1];
15 int flower_from[MAXN*2+1][MAXN+1],S[MAXN
16     *2+1],vis[MAXN*2+1];
17 vector<int> flower[MAXN*2+1];
18 queue<int> q;
19 inline int e_delta(const edge &e){ // does
20     not work inside blossoms
21     return lab[e.u]+lab[e.v]-g[e.u][e.v].w*2;
22 }
23 inline void update_slack(int u,int x){
24     if(!slack[x]||e_delta(g[u][x])<e_delta(g[
25         slack[x]][x]))slack[x]=u;
26 }
27 inline void set_slack(int x){
28     slack[x]=0;
29     for(int u=1;u<=n;u++){
30         if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)
31             update_slack(u,x);
32 }
33 void q_push(int x){
34     if(x<=n)q.push(x);
35     else for(size_t i=0;i<flower[x].size();i
36         ++i)q.push(flower[x][i]);
37 }
38 inline void set_st(int x,int b){
39     st[x]=b;
40     if(x>n)for(size_t i=0;i<flower[x].size()
41         ;++i)
42         set_st(flower[x][i],b);
43 }

```

```

36 }
37 inline int get_pr(int b,int xr){
38     int pr=find(flower[b].begin(),flower[b].
39         end(),xr)-flower[b].begin();
40     if(pr%2==1){//檢查他在前一層圖是奇點還是偶
41         點
42         reverse(flower[b].begin()+1,flower[b].
43             end());
44         return (int)flower[b].size()-pr;
45     }else return pr;
46 }
47 inline void set_match(int u,int v){
48     match[u]=g[u][v].v;
49     if(u>n){
50         edge e=g[u][v];
51         int xr=flower_from[u][e.u],pr=get_pr(u,
52             xr);
53         for(int i=0;i<pr;i++)set_match(flower[u
54             ][i],flower[u][i^1]);
55         set_match(xr,v);
56         rotate(flower[u].begin(),flower[u].begin
57             ())+pr,flower[u].end());
58     }
59 }
60 inline void augment(int u,int v){
61     for(;;){
62         int xnv=st[match[u]];
63         set_match(u,v);
64         if(!xnv)return;
65         set_match(xnv,st[pa[xnv]]);
66         u=st[pa[xnv]],v=xnv;
67     }
68 }
69 inline int get_lca(int u,int v){
70     static int t=0;
71     for(++t;u||v;swap(u,v)){
72         if(u==0)continue;
73         if(vis[u]==t)return u;
74         vis[u]=t;//這種方法可以不用清空v陣列
75         u=st[match[u]];
76         if(u)u=st[pa[u]];
77     }
78     return 0;
79 }
80 inline void add_blossom(int u,int lca,int v)
81     {
82         int b=n+1;
83         while(b<=n_x&&st[b]==0)b++;
84         if(b>n_x)b=n_x;
85         lab[b]=0,S[b]=0;
86         match[b]=match[lca];
87         flower[b].clear();
88         flower[b].push_back(lca);
89         for(int x=u,y;lca;x=st[pa[y]])
90             flower[b].push_back(x),flower[b].
91                 push_back(y=st[match[x]]),q_push(y);
92         reverse(flower[b].begin()+1,flower[b].end
93             ());
94         for(int x=v,y;lca;x=st[pa[y]])
95             flower[b].push_back(x),flower[b].
96                 push_back(y=st[match[x]]),q_push(y);
97         set_st(b,b);
98         for(int x=1;x<=n_x;x++)g[b][x].w=g[x][b].w
99             =0;
100         for(int x=1;x<=n;xx++)flower_from[b][x]=0;
101 }

```

```

90 for(size_t i=0;i<flower[b].size();++i){
91     int xs=flower[b][i];
92     for(int x=1;x<=n_x;xx++){
93         if(g[b][x].w==0||e_delta(g[xs][x])<
94             e_delta(g[b][x]))
95             g[b][x]=g[xs][x],g[x][b]=g[x][xs];
96         for(int x=1;x<=n;xx++){
97             if(flower_from[xs][x])flower_from[b][x
98                 ]=xs;
99         }
100     }
101     set_slack(b);
102 }
103 inline void expand_blossom(int b){ // S[b]
104     == 1
105     for(size_t i=0;i<flower[b].size();++i)
106         set_st(flower[b][i],flower[b][i]);
107     int xr=flower_from[b][g[b][pa[b]].u],pr=
108         get_pr(b,xr);
109     for(int i=0;i<pr;i+=2){
110         int xs=flower[b][i],xns=flower[b][i+1];
111         pa[xs]=g[xns][xs].u;
112         S[xs]=1,S[xns]=0;
113         slack[xs]=0,set_slack(xns);
114         q_push(xns);
115     }
116     S[xr]=1,pa[xr]=pa[b];
117     for(size_t i=pr+1;i<flower[b].size();++i){
118         int xs=flower[b][i];
119         S[xs]=-1,set_slack(xs);
120     }
121     st[b]=0;
122 }
123 inline bool on_found_edge(const edge &e){
124     int u=st[e.u],v=st[e.v];
125     if(S[v]==-1){
126         pa[v]=e.u,S[v]=1;
127         int nu=st[match[v]];
128         slack[v]=slack[nu]=0;
129         S[nu]=0,q_push(nu);
130     }else if(S[v]==0){
131         int lca=get_lca(u,v);
132         if(!lca){
133             augment(u,v),augment(v,u);
134             return true;
135         }else add_blossom(u,lca,v);
136     }
137     return false;
138 }
139 inline bool matching(){
140     memset(S+1,-1,sizeof(int)*n_x);
141     memset(slack+1,0,sizeof(int)*n_x);
142     q=queue<int>();
143     for(int x=1;x<=n_x;xx++){
144         if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,
145             q_push(x);
146     }
147     if(q.empty())return false;
148     for(;;){
149         while(q.size()){
150             int u=q.front();q.pop();
151             if(S[st[u]]==1)continue;
152             for(int v=1;v<=n;vv++){
153                 if(g[u][v].w>0&&st[u]!=st[v]){
154                     if(e_delta(g[u][v])==0){
155                         if(on_found_edge(g[u][v]))return
156                             true;
157                     }else update_slack(u,st[v]);
158                 }
159             }
160         }
161     }

```

```

150     }
151 }
152 int d=INF;
153 for(int b=n+1;b<=n_x;++b)
154     if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
155
156 for(int x=1;x<=n_x;++x)
157     if(st[x]==x&&slack[x]){
158         if(S[x]==-1)d=min(d,e_delta(g[slack[x]][x]));
159         else if(S[x]==0)d=min(d,e_delta(g[slack[x]][x])/2);
160     }
161 for(int u=1;u<=n;++u){
162     if(S[st[u]]==0){
163         if(lab[u]<=d)return 0;
164         lab[u]-=d;
165     }else if(S[st[u]]==1)lab[u]+=d;
166 }
167 for(int b=n+1;b<=n_x;++b)
168     if(st[b]==b){
169         if(S[st[b]]==0)lab[b]+=d*2;
170         else if(S[st[b]]==1)lab[b]-=d*2;
171     }
172 q=queue<int>();
173 for(int x=1;x<=n_x;++x)
174     if(st[x]==x&&slack[x]&&st[slack[x]]!=x
175        &&e_delta(g[slack[x]][x])==0)
176         if(on_found_edge(g[slack[x]][x]))
177             return true;
178 for(int b=n+1;b<=n_x;++b)
179     if(st[b]==b&&S[b]==1&&lab[b]==0)
180         expand_blossom(b);
181 }
182 return false;
183
184 inline pair<long long,int> weight_blossom(){
185     memset(match+1,0,sizeof(int)*n);
186     n_x=n;
187     int n_matches=0;
188     long long tot_weight=0;
189     for(int u=0;u<=n;++u)st[u]=u,flower[u].clear();
190     int w_max=0;
191     for(int u=1;u<=n;++u)
192         for(int v=1;v<=n;++v){
193             flower_from[u][v]=(u==v?u:0);
194             w_max=max(w_max,g[u][v].w);
195         }
196     for(int u=1;u<=n;++u)lab[u]=w_max;
197     while(matching()+n_matches;
198     for(int u=1;u<=n;++u)
199         if(match[u]&&match[u]<u)
200             tot_weight+=g[u][match[u]].w;
201     return make_pair(tot_weight,n_matches);
202 }
203
204 inline void init_weight_graph(){
205     for(int u=1;u<=n;++u)
206         for(int v=1;v<=n;++v)
207             g[u][v]=edge(u,v,0);
208 }

```

6 language

6.1 CNF.cpp

```

1 #define MAXN 55
2 struct CNF{
3     int s,x,y;//s->xy | s->x, if y== -1
4     int cost;
5     CNF(){}
6     CNF(int s,int x,int y,int c):s(s),x(x),y(y),cost(c){}
7 };
8 int state;//規則數量
9 map<char,int> rule;//每個字元對應到的規則，小寫字母為終端字元
10 vector<CNF> cnf;
11 inline void init(){
12     state=0;
13     rule.clear();
14     cnf.clear();
15 }
16 inline void add_to_cnf(char s,const string &p,int cost){
17     //加入一個 s -> <p> 的文法，代價為 cost
18     if(rule.find(s)==rule.end())rule[s]=state++;
19     for(auto c:p)if(rule.find(c)==rule.end())rule[c]=state++;
20     if(p.size()==1){
21         cnf.push_back(CNF(rule[s],rule[p[0]],-1,cost));
22     }else{
23         int left=rule[s];
24         int sz=p.size();
25         for(int i=0;i<sz-2;++i){
26             cnf.push_back(CNF(left,rule[p[i]],state,0));
27             left=state++;
28         }
29         cnf.push_back(CNF(left,rule[p[sz-2]],rule[p[sz-1]],cost));
30     }
31 }
32 vector<long long> dp[MAXN][MAXN];
33 vector<bool> neg_INF[MAXN][MAXN]; //如果花費是負的可能會有無限小的情形
34 inline void relax(int l,int r,const CNF &c, long long cost,bool neg_c=0){
35     if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x]||cost<dp[l][r][c.s])){

```

```

36     if(neg_c||neg_INF[l][r][c.x]){
37         dp[l][r][c.s]=0;
38         neg_INF[l][r][c.s]=true;
39     }else dp[l][r][c.s]=cost;
40 }
41
42 inline void bellman(int l,int r,int n){
43     for(int k=1;k<=state;++k)
44         for(auto c:cnf)
45             if(c.y== -1)relax(l,r,c,dp[l][r][c.x]+c.cost,k==n);
46 }
47
48 inline void cyk(const vector<int> &tok){
49     for(int i=0;i<(int)tok.size();++i){
50         for(int j=0;j<(int)tok.size();++j){
51             dp[i][j]=vector<long long>(state+1,INT_MAX);
52             neg_INF[i][j]=vector<bool>(state+1,false);
53         }
54         dp[i][i][tok[i]]=0;
55         bellman(i,i,tok.size());
56     }
57     for(int r=1;r<(int)tok.size();++r){
58         for(int l=r-1;l>=0;--l){
59             for(int k=l;k<r;++k)
60                 for(auto c:cnf)
61                     if(~c.y)relax(l,r,c,dp[l][k][c.x]+dp[k+1][r][c.y]+c.cost);
62             bellman(l,r,tok.size());
63         }
64     }

```

6.2 earley.cpp

```

1 struct Rule{
2     vector<vector<Rule*> > p;
3     void add(const vector<Rule*> &l){
4         p.push_back(l);
5     }
6 };
7 map<string,Rule*> NameRule;
8 map<Rule*,string> RuleName;
9 inline void init_Rule(){
10     for(auto r:RuleName)delete r.first;
11     RuleName.clear();
12     NameRule.clear();
13 }
14 inline Rule *add_rule(const string &s){
15     if(NameRule.find(s)!=NameRule.end())return NameRule[s];
16     Rule *r=new Rule();
17     RuleName[r]=s;
18     NameRule[s]=r;
19     return r;
20 }
21 typedef vector<Rule*> production;
22 struct State{
23     Rule *r;
24     int rid,dot_id,start,end;
25     State(Rule *r,int rid,int dot,int start):r(r),rid(rid),dot_id(dot),start(start),

```

```

26     end(-1){}
27 State(Rule *r=0,int col=0):r(r),rid(-1),dot_id(-1),start(-1),end(col){}
28 bool completed()const{
29     return rid== -1||dot_id>=(int)r->p[rid].size();
30 }
31 Rule *next_term()const{
32     if(completed())return 0;
33     return r->p[rid][dot_id];
34 }
35 bool operator<(const State& b)const{
36     if(start!=b.start)return start<b.start;
37     if(dot_id!=b.dot_id)return dot_id<b.dot_id;
38     if(r!=b.r)return r<b.r;
39     return rid<b.rid;
40 }
41 void print()const{
42     cout<<RuleName[r]<<"->";
43     if(rid!= -1)for(size_t i=0;i<rid){
44         if((int)i==dot_id)cout<<" "<<"$";
45         if(i>r->p[rid].size())break;
46         cout<<" "<<RuleName[r->p[rid][i]];
47     }
48     cout<<" "<<"["<<start<<" "<<end<<" "]"<<endl;
49 };
50 struct Column{
51     Rule *term;
52     string value;
53     vector<State> s;
54     map<State,set<pair<State,State> > > div;
55     //div比較像一棵 左兄右子的樹
56     Column(Rule *r,const string &s):term(r),value(s){}
57     Column(){}
58     bool add(const State &st,int col){
59         if(div.find(st)==div.end()){
60             div[st];
61             s.push_back(st);
62             s.back().end=col;
63             return true;
64         }else return false;
65     }
66 };
67 inline vector<Column> lexer(string text){
68     //tokenize，要自己寫，以下為範例
69     //他會把 input stream 變成 token stream，就是 (terminal,value)pair
70     vector<Column> token;
71     replace(text.begin(),text.end(),',',' ');
72     stringstream ss(text);
73     while(ss>>text){
74         if(text=="a"||text=="of")continue;
75         if(text=="List"){
76             token.push_back(Column(NameRule["("],text));
77         }else if(text=="and"){
78             token.push_back(Column(NameRule[")"],text));
79         }else token.push_back(Column(NameRule["T"],text));
80 }

```

```

81 return token;
82 }
83 vector<Column> table;
84 inline void predict(int col, Rule *rul){
85     for(size_t i=0; i<rul->p.size(); ++i){
86         table[col].add(State(rul, i, 0, col), col);
87     }
88 }
89 inline void scan(int col, const State &s, Rule
    *r){
90     if(r!=table[col].term) return;
91     State ns(s.r, s.rid, s.dot_id+1, s.start);
92     table[col].add(ns, col);
93     table[col].div[ns].insert(make_pair(s,
        State(r, col)));
94 }
95 inline void complete(int col, const State &s)
    {
96     for(size_t i=0; i<table[s.start].s.size()
        ; ++i){
97         State st=table[s.start].s[i];
98         Rule *term=st.next_term();
99         if(!term || term->p.size()==0) continue;
100         if(term==s.r){
101             State nst(st.r, st.rid, st.dot_id+1, st.
                start);
102             table[col].add(nst, col);
103             table[col].div[nst].insert(make_pair(
                st, s));
104         }
105     }
106 }
107 inline pair<bool, State> parse(Rule *GAMMA,
    const vector<Column> &token){
108     table.resize(token.size()+1);
109     for(size_t i=0; i<token.size(); ++i) table[i
        +1]=Column(token[i]);
110     table[0]=Column();
111     table[0].add(State(GAMMA, 0, 0, 0), 0);
112     for(size_t i=0; i<table.size(); ++i){
113         for(size_t j=0; j<table[i].s.size(); ++j){
114             State state=table[i].s[j];
115             if(state.completed()) complete(i, state)
                ;
116             else{
117                 Rule *term=state.next_term();
118                 if(term->p.size()>0) predict(i, term);
119                 else if(i+1<table.size()) scan(i+1,
                    state, term);
120             }
121         }
122     }
123     for(size_t i=0; i<table.back().s.size(); ++i
        ){
124         if(table.back().s[i].r==GAMMA&&table.
            back().s[i].completed()){
125             return make_pair(true, table.back().s[i]
                );
126         }
127     }
128     return make_pair(false, State(0, -1));
129 }
130 struct node{//語法樹的節點
131     State s;
132     vector<vector<node*> > child; //vector<node
        *>.size()>1表示ambiguous
133     node(const State &s):s(s){}
134     node(){}
135 };
136 struct State_end_cmp{
137     bool operator()(const State &a, const State
        &b) const{
138         return a.end<b.end || (a.end==b.end&&a<b);
139     }
140 };
141 map<State, node*, State_end_cmp> cache;
142 vector<node*> node_set;
143 inline void init_cache(){
144     for(auto d:node_set) delete d;
145     cache.clear();
146     node_set.clear();
147 }
148 void build_tree(const State &s, node *pa,
    bool amb=0){
149     if(cache.find(s)!=cache.end()){
150         pa->child.push_back(vector<node*>(1,
            cache[s]));
151         return;
152     }
153     node *o;
154     if(s.completed()){
155         o=new node(s);
156         if(amb) pa->child.back().push_back(o);
157         else pa->child.push_back(vector<node
            *>(1, o));
158     } else o=pa->child.back().back();
159     amb=0;
160     for(auto div:table[s.end].div[s]){
161         if(!amb) build_tree(div.first, pa);
162         _build_tree(div.second, o, amb);
163         amb=1;
164     }
165     if(s.completed()) cache[s]=o;
166 }
167 inline node *build_tree(const State &s){
168     init_cache();
169     node o;
170     _build_tree(s, &o);
171     assert(o.child.size()==1);
172     assert(o.child.back().size()==1);
173     return o.child.back().back();
174 }
175 void print_tree(node *o, int dep=0){
176     cout<<string(dep, ' '), o->s.print();
177     for(auto div:o->child){
178         for(auto nd:div){
179             print_tree(nd, dep+2);
180         }
181     }
182 }
183 //開始寫code:以下為加入語法的範例
184 inline Rule *get_my_Rule(){
185     Rule *S=add_rule("S"), *E=add_rule("E"), *L=
        add_rule("L");
186     Rule *list=add_rule("("), *AND=add_rule("&")
        ), *T=add_rule("T");
187     S->add({list, E});
188     S->add({list, L});
189     L->add({E, L});
190     L->add({E, AND, E});
191     E->add({T});
192     E->add({S});
193     Rule *GAMMA=add_rule("GAMMA"); //一定要有
        gamma rule當作是最上層的語法
194     GAMMA->add({S});
195     return GAMMA;
196 }
197 LL primitive_root(const LL &p) {
198     if(p==2) return 1;
199     vector<LL> v;
200     Factor(p-1, v);
201     v.erase(unique(v.begin(), v.end()), v.
        end());
202     for(LL g=2; g<p; ++g)
203         if(g_test(g, p, v))
204             return g;
205     puts("primitive_root NOT FOUND");
206     return -1;
207 }
208 int Legendre(const LL &a, const LL &p) {
209     return modexp(a%p, (p-1)/2, p);
210 }
211 LL inv(const LL &a, const LL &n) {
212     LL d, x, y;
213     gcd(a, n, d, x, y);
214     return d==1 ? (x+n)%n : -1;
215 }
216 LL log_mod(const LL &a, const LL &b, const
    LL &p) {
217     // a ^ x = b ( mod p )
218     int m=sqrt(p+.5), e=1;
219     LL v=inv(modexp(a, m, p), p);
220     map<LL, int> x;
221     x[1]=0;
222     for(int i=1; i<m; ++i) {
223         e = LLMul(e, a, p);
224         if(x.count(e)) x[e] = i;
225     }
226     for(int i=0; i<m; ++i) {
227         if(x.count(b)) return i*m + x[b];
228         b = LLMul(b, v, p);
229     }
230     return -1;
231 }
232 LL Tonelli_Shanks(const LL &n, const LL &p)
    {
233     // x^2 = n ( mod p )
234     if(n==0) return 0;
235     if(Legendre(n, p)!=1) while(1) { puts("
        SQRT ROOT does not exist"); }
236     int S = 0;
237     LL Q = p-1;
238     while( !(Q&1) ) { Q>>=1; ++S; }
239     if(S==1) return modexp(n%p, (p+1)/4, p);
240     LL z = 2;
241     for(; Legendre(z, p)!=-1; ++z)
242         LL c = modexp(z, Q, p);
243     LL R = modexp(n%p, (Q+1)/2, p), t = modexp
        (n%p, Q, p);
244     int M = S;
245     while(1) {
246         if(t==1) return R;
247         LL b = modexp(c, 1L<<(M-i-1), p);
248         R = LLMul(R, b, p);
249         t = LLMul( LLMul(b, b, p), t, p);
250         c = LLMul(b, b, p);
251         M = i;
252     }
253     return -1;
254 }
255 typedef long long int LL;
256 template<typename T>
257 void gcd(const T &a, const T &b, T &d, T &x, T &
    y){
258     if(!b) d=a, x=1, y=0;
259     else gcd(b, a%b, d, y, x), y-=x*(a/b);
260 }
261 const int MAXPRIME = 1000000;
262 int iscom[MAXPRIME], prime[MAXPRIME],
    primecnt;
263 int phi[MAXPRIME], mu[MAXPRIME];
264 void sieve(void)
    {
265     memset(iscom, 0, sizeof(iscom));
266     primecnt = 0;
267     phi[1] = mu[1] = 1;
268     for(int i=2; i<MAXPRIME; ++i) {
269         if(!iscom[i]) {
270             prime[primecnt++] = i;
271             mu[i] = -1;
272             phi[i] = i-1;
273         }
274         for(int j=0; j<primecnt; ++j) {
275             int k = i * prime[j];
276             if(k>=MAXPRIME) break;
277             iscom[k] = prime[j];
278             if(i%prime[j]==0) {
279                 mu[k] = 0;
280                 phi[k] = phi[i] * prime[j];
281                 break;
282             } else {
283                 mu[k] = -mu[i];
284                 phi[k] = phi[i] * (prime[j]
                    -1);
285             }
286         }
287     }
288 }
289 bool g_test(const LL &g, const LL &p, const
    vector<LL> &v) {
290     for(int i=0; i<v.size(); ++i)
291         if(modexp(g, (p-1)/v[i], p)==1)
292             return false;
293     return true;
294 }

```

7 Number_Theory

7.1 basic.cpp

7.2 bit_set.cpp

```

1 void sub_set(int S){
2     int sub=S;
3     do{
4         //對某集合的子集合的處理
5         sub=(sub-1)&S;
6     }while(sub!=S);
7 }
8 void k_sub_set(int k,int n){
9     int comb=(1<<k)-1,S=1<<n;
10    while(comb<S){
11        //對大小為k的子集合的處理
12        int x=comb&~comb,y=comb+x;
13        comb=((comb&~y)/x>>1)|y;
14    }
15 }

```

7.3 cantor_expansion.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define MAXN 11
4 int factorial[MAXN];
5 inline void init(){
6     factorial[0]=1;
7     for(int i=1;i<=MAXN;++i)factorial[i]=
8         factorial[i-1]*i;
9 }
10 inline int encode(const std::vector<int> &s)
11 {
12     int n=s.size(),res=0;
13     for(int i=0;i<n;++i){
14         int t=0;
15         for(int j=i+1;j<n;++j)
16             if(s[j]<s[i])++t;
17         res+=t*factorial[n-i-1];
18     }
19     return res;
20 }
21 inline std::vector<int> decode(int a,int n){
22     std::vector<int> res;
23     std::vector<bool> vis(n,0);
24     for(int i=n-1;i>=0;--i){
25         int t=a/factorial[i],j;
26         for(j=0;j<n;++j)
27             if(!vis[j]){
28                 if(t==0)break;
29                 --t;
30             }
31         res.push_back(j);
32         vis[j]=1;
33         a%=factorial[i];
34     }
35     return res;
36 }
37 int main(){
38     init();
39     vector<int> p={0,1,2,3,4,5,6,7,8};
40     for(int i=0;i<factorial[9];++i){
41         vector<int> s=decode(i,9);

```

```

40     if(s!=p)puts("XX");
41     next_permutation(p.begin(),p.end());
42 }
43 return 0;
44 }

```

7.4 Chinese_remainder_theorem

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #ifndef CHINESE_REMAINDER_THEOREM
4 #define CHINESE_REMAINDER_THEOREM
5 template<typename T>
6 inline T Euler(T n){
7     T ans=n;
8     for(T i=2;i*i<=n;++i){
9         if(n%i==0){
10             ans=ans/i*(i-1);
11             while(n%i==0)n/=i;
12         }
13     }
14     if(n>1)ans=ans/n*(n-1);
15     return ans;
16 }
17 template<typename T>
18 inline T pow_mod(T n,T k,T m){
19     T ans=1;
20     for(n=(n>=m?n%m:n);k>=1){
21         if(k&1)ans=ans*n%m;
22         n=n*n%m;
23     }
24     return ans;
25 }
26 template<typename T>
27 inline T crt(std::vector<T> &m,std::vector<T>
28             > &a){
29     T M=1,tM,ans=0;
30     for(int i=0;i<(int)m.size();++i)M*=m[i];
31     for(int i=0;i<(int)a.size();++i){
32         tM=M/m[i];
33         ans=(ans+(a[i]*tM%M)*pow_mod(tM,Euler(m[i])-1,m[i])%M)%M;
34         /*如果m[i]是質數·Euler(m[i])-1=m[i]-2·
35         就不用算Euler了*/
36     }
37     return ans;
38 }
39 #endif
40 int n;
41 vector<long long> a,m;
42 int main(){
43     while(~scanf("%d",&n)){
44         for(int i=0;i<n;++i){
45             long long x,y;
46             scanf("%lld%lld",&x,&y);
47             m.push_back(x);
48             a.push_back(y);
49         }
50         long long ans=crt(m,a);
51         printf("%lld\n",ans);
52         for(int i=0;i<n;++i)printf("%lld %lld\n",
53             m[i],ans%m[i]);

```

```

51     m.clear();
52     a.clear();
53 }
54 return 0;
55 }
56 /*
57 4
58 199 198
59 200 199
60 201 197
61 137 88
62 2
63 265163 465
64 66546165 7122
65 5
66 379 46
67 853 852
68 971 777
69 659 128
70 281 256
71 4
72 6359 1
73 4877 5
74 1627 6
75 8941 7122
76 */

```

7.5 enumerate.cpp

```

1 void all_divdown(const LL &n) { // all n/x
2     for(LL a=1;a<=n;a=n/(n/(a+1))) {
3         // dosomething;
4     }
5 }

```

7.6 eulerphi.cpp

```

1 int eulerPhi(int n){
2     int m = sqrt(n+0.5);
3     int res=n;
4     for(int i=2; i<=m; i++){
5         if(n%i==0){
6             res = res*(i-1)/i;
7             while(n%i==0)n/=i;
8         }
9     }
10    if(n>1) res = res*(n-1)/n;
11    return res;
12 }
13
14 vector<int> phiTable(int n){
15     vector<int> phi(n+1, 0);
16     phi[1] = 1;
17     for(int i=2; i<=n; i++) if(!phi[i])
18         for(int j=i; j<=n; j+=i){
19             if(!phi[j])phi[j] = j;
20             phi[j] = phi[j]*(i-1)/i;
21         }
22     return phi;
23 }

```

7.7 Factor.cpp

```

1 LL Llmul(LL a, LL b, const LL &mod) {
2     LL ans=0;
3     while(b) {
4         if(b&1) {
5             ans+=a;
6             if(ans>=mod) ans-=mod;
7         }
8         a<<=1, b>>=1;
9         if(a>=mod) a-=mod;
10    }
11    return ans;
12 }
13 inline long long mod_mul(long long a,long
14 long b,long long m){
15     a%=m,b%=m;
16     long long y=(long long)((double)a*b/m+0.5)
17     ;/* fast for m < 2^58 */
18     long long r=(a*b-y*m)%m;
19     return r<0?r+m:r;
20 }
21 template<typename T>
22 inline T pow(T a,T b,T mod){//a^b%mod
23     T ans=1;
24     for(;b;a=mod_mul(a,a,mod),b>>=1)
25         if(b&1)ans=mod_mul(ans,a,mod);
26     return ans;
27 }
28 int sprp[3]={2,7,61};//int範圍可解
29 int llsprp
30 [7]={2,325,9375,28178,450775,9780504,1795265}
31 //至少unsigned Long Long範圍
32 template<typename T>
33 inline bool isprime(T n,int *sprp,int num){
34     if(n==2)return 1;
35     if(n<2||n%2==0)return 0;
36     int t=0;
37     T u=n-1;
38     for(;u%2==0;u>>=1)
39         for(int i=0;i<num;++i){
40             T a=sprp[i]%n;
41             if(a==0||a==1||a==n-1)continue;
42             T x=pow(a,u,n);
43             if(x==1||x==n-1)continue;
44             for(int j=0;j<t;++j){
45                 x=mod_mul(x,x,n);
46                 if(x==1)return 0;
47                 if(x==n-1)break;
48             }
49             if(x==n-1)continue;
50             return 0;
51         }
52     return 1;
53 }
54 LL func(const LL n,const LL mod,const int c)
55 {
56     return (Llmul(n,n,mod)+c+mod)%mod;
57 }
58 LL pollorroho(const LL n, const int c){//循
59     環節長度
60     LL a=1, b=1;

```

```

57 a=func(a,n,c)%n;
58 b=func(b,n,c)%n; b=func(b,n,c)%n;
59 while(gcd(abs(a-b),n)==1) {
60     a=func(a,n,c)%n;
61     b=func(b,n,c)%n; b=func(b,n,c)%n;
62 }
63 return gcd(abs(a-b),n);
64 }
65
66 void prefactor(LL &n, vector<LL> &v) {
67     for(int i=0;i<12;++i) {
68         while(n%prime[i]==0) {
69             v.push_back(prime[i]);
70             n/=prime[i];
71         }
72     }
73 }
74
75 void smallfactor(LL n, vector<LL> &v) {
76     if(n<MAXPRIME) {
77         while(isp[(int)n]) {
78             v.push_back(isp[(int)n]);
79             n/=isp[(int)n];
80         }
81         v.push_back(n);
82     } else {
83         for(int i=0;i<primecnt&&prime[i]*
84             prime[i]<=n;++i) {
85             while(n%prime[i]==0) {
86                 v.push_back(prime[i]);
87                 n/=prime[i];
88             }
89             if(n!=1) v.push_back(n);
90         }
91     }
92
93 void comfactor(const LL &n, vector<LL> &v) {
94     if(n<1e9) {
95         smallfactor(n,v);
96         return;
97     }
98     if(Isprime(n)) {
99         v.push_back(n);
100         return;
101     }
102     LL d;
103     for(int c=3;++c) {
104         d = pollrho(n,c);
105         if(d!=n) break;
106     }
107     comfactor(d,v);
108     comfactor(n/d,v);
109 }
110
111 void Factor(const LL &x, vector<LL> &v) {
112     LL n = x;
113     if(n==1) { puts("Factor 1"); return; }
114     prefactor(n,v);
115     if(n==1) return;
116     comfactor(n,v);
117     sort(v.begin(),v.end());
118 }
119
120 void AllFactor(const LL &n,vector<LL> &v) {
121     vector<LL> tmp;

```

```

122 Factor(n,tmp);
123 v.clear();
124 v.push_back(1);
125 int len;
126 LL now=1;
127 for(int i=0;i<tmp.size();++i) {
128     if(i==0 || tmp[i]!=tmp[i-1]) {
129         len = v.size();
130         now = 1;
131     }
132     now*=tmp[i];
133     for(int j=0;j<len;++j)
134         v.push_back(v[j]*now);
135 }
136 }

```

7.8 FFT.cpp

```

1 template<typename T,typename VT=std::vector<
2     std::complex<T> > >
3 struct FFT{
4     const T pi;
5     FFT(const T pi=acos((T)-1)):pi(pi){}
6     unsigned int bit_reverse(unsigned int a,
7         int len){
8         a=((a&0x55555555U)<<1)|((a&0xAAAAAAAU)
9             >>1);
10        a=((a&0x33333333U)<<2)|((a&0xCCCCCCU)
11            >>2);
12        a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0FU)
13            >>4);
14        a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)
15            >>8);
16        a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)
17            >>16);
18        return a>>(32-len);
19    }
20    void fft(bool is_inv,VT &in,VT &out,int N)
21    {
22        int bitlen=std::lg(N),num=is_inv?-1:1;
23        for(int i=0;i<N;++i)out[bit_reverse(i,
24            bitlen)]=in[i];
25        for(int step=2;step<=N;step<=1){
26            const int mh=step>>1;
27            for(int i=0;i<mh;++i){
28                std::complex<T> wi=exp(std::complex<
29                    T>(0,i*num*pi/mh));
30                for(int j=i;j<N;j+=step){
31                    int k=j+mh;
32                    std::complex<T> u=out[j],t=wi*out[
33                        k];
34                    out[j]=u+t;
35                    out[k]=u-t;
36                }
37            }
38        }
39        if(is_inv)for(int i=0;i<N;++i)out[i]/=N;
40    }
41 };

```

7.9 find_real_root.cpp

```

1 // an*x^n + ... + a1x + a0 = 0;
2 int sign(double x){
3     return x < -eps ? -1 : x > eps;
4 }
5
6 double get(const vector<double>&coef, double
7     x){
8     double e = 1, s = 0;
9     for(auto i : coef) s += i*e, e *= x;
10    return s;
11 }
12
13 double find(const vector<double>&coef, int n
14     , double lo, double hi){
15     double sign_lo, sign_hi;
16     if( !(sign_lo = sign(get(coef,lo))) )
17         return lo;
18     if( !(sign_hi = sign(get(coef,hi))) )
19         return hi;
20     if(sign_lo * sign_hi > 0) return INF;
21     for(int stp = 0; stp < 100 && hi - lo >
22         eps; ++stp){
23         double m = (lo+hi)/2.0;
24         int sign_mid = sign(get(coef,m));
25         if(!sign_mid) return m;
26         if(sign_lo*sign_mid < 0) hi = m;
27         else lo = m;
28     }
29     return (lo+hi)/2.0;
30 }
31
32 vector<double> cal(vector<double>coef, int n
33 ) {
34     vector<double>res;
35     if(n == 1){
36         if(sign(coef[1])) res.pb(-coef[0]/
37             coef[1]);
38         return res;
39     }
40     vector<double>dcoef(n);
41     for(int i = 0; i < n; ++i) dcoef[i] =
42         coef[i+1]*(i+1);
43     vector<double>droot = cal(dcoef, n-1);
44     droot.insert(droot.begin(), -INF);
45     droot.pb(INF);
46     for(int i = 0; i+1 < droot.size(); ++i){
47         double tmp = find(coef, n, droot[i],
48             droot[i+1]);
49         if(tmp < INF) res.pb(tmp);
50     }
51     return res;
52 }
53
54 int main () {
55     vector<double>ve;
56     vector<double>ans = cal(ve, n);
57     // 視情況把答案 +eps, 避免 -0
58 }

```

7.10 Gauss_Elimination.cpp

```

1 const int MAX = 300;
2 const double EPS = 1e-8;
3
4 double mat[MAX][MAX];
5 void Gauss(int n) {
6     for(int i=0;i<n;i++) {
7         bool ok = 0;
8         for(int j=i;j<n;j++) {
9             if(fabs(mat[j][i]) > EPS) {
10                 swap(mat[j], mat[i]);
11                 ok = 1;
12                 break;
13             }
14         }
15         if(!ok) continue;
16         double fs = mat[i][i];
17         for(int j=i+1;j<n;j++) {
18             double r = mat[j][i] / fs;
19             for(int k=i;k<n;k++) {
20                 mat[j][k] -= mat[i][k] * r;
21             }
22         }
23     }
24 }
25 }

```

7.11 LinearCongruence.cpp

```

1 pair<LL,LL> LinearCongruence(LL a[],LL b[],
2     LL m[],int n) {
3     // a[i]*x = b[i] (mod m[i])
4     for(int i=0;i<n;++i) {
5         LL x, y, d = extgcd(a[i],m[i],x,y);
6         if(b[i]%d!=0) return make_pair(-1LL,
7             0LL);
8         m[i] /= d;
9         b[i] = LLmul(b[i]/d,x,m[i]);
10    }
11    LL lastb = b[0], lastm = m[0];
12    for(int i=1;i<n;++i) {
13        LL x, y, d = extgcd(m[i],lastm,x,y);
14        if((lastb-b[i])%d!=0) return
15            make_pair(-1LL,0LL);
16        lastb = LLmul((lastb-b[i])/d,x,(
17            lastm/d))*m[i];
18        lastm = (lastm/d)*m[i];
19        lastb = (lastb+b[i])%lastm;
20    }
21    return make_pair(lastb<0?lastb+lastm:
22        lastb,lastm);
23 }

```

7.12 Lucas.cpp

```

1 int mod_fact(int n,int &e){
2     e=0;
3     if(n==0)return 1;

```

```

4 // (n/p)! % p
5 int res=mod_fact(n/P,e);
6 e += n/P;
7 if( (n/P) %2 == 0){// = 1
8     return res*fact[n%P]%P;
9 }
10 // = -1
11 return res*(P-fact[n%P])%P;
12 }
13 int extGCD(int a,int b,int &x,int &y){
14     int d=a;
15     if(b!=0){
16         d=extGCD(b,a%b,y,x);
17         y-= (a/b)*x;
18     }else{
19         x=1;y=0;
20     }
21     return d;
22 }
23 int modInverse(int n){
24     int x,y;
25     extGCD(n,P,x,y);
26     return (P+x%P)%P;
27 }
28 int Cmod(int n,int m){
29     int a1,a2,a3,e1,e2,e3;
30     a1=mod_fact(n,e1);
31     a2=mod_fact(m,e2);
32     a3=mod_fact(n-m,e3);
33     if(e1>e2+e3)return 0;
34     return a1*modInverse(a2*a3%P)%P;
35 }

```

7.13 NTT.cpp

```

1 2615053605667*(2^18)+1,3
2 15*(2^27)+1,31
3 479*(2^21)+1,3
4 7*17*(2^23)+1,3
5 3*3*211*(2^19)+1,5
6 25*(2^22)+1,3
7 template<typename T,typename VT=std::vector<
8     T>>
9 struct NTT{
10     const T P,G;
11     NTT(T p=(1<<23)*7*17+1,T g=3):P(p),G(g){}
12     unsigned int bit_reverse(unsigned int a,
13         int len){
14         a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)
15             >>1);
16         a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)
17             >>2);
18         a=((a&0x0F0F0F0FU)<<4)|((a&0xFF0F0F0FU)
19             >>4);
20         a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)
21             >>8);
22         a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)
23             >>16);
24         return a>>(32-len);
25     }
26     T pow_mod(T n,T k,T m){
27         T ans=1;
28         for(n=(n>=m?n%m:n);k;k>>=1){

```

```

22     if(k&1)ans=ans*n%m;
23     n=n*n%m;
24 }
25 return ans;
26 }
27 void ntt(bool is_inv,VT &in,VT &out,int N)
28 {
29     int bitlen=std::__lg(N);
30     for(int i=0;i<N;++i)out[bit_reverse(i,
31         bitlen)]=in[i];
32     for(int step=2,id=1;step<=N;step<=1,++
33         id){
34         T wn=pow_mod(G,(P-1)>>id,P),wi=1,u,t;
35         const int mh=step>>1;
36         for(int i=0;i<mh;++i){
37             for(int j=i;j<N;j+=step){
38                 u=out[j];t=wi*out[j+mh]%P;
39                 out[j]=u+t;
40                 out[j+mh]=u-t;
41                 if(out[j]>=P)out[j]-=P;
42                 if(out[j+mh]<0)out[j+mh]+=P;
43             }
44             wi=wi*wn%P;
45         }
46     }
47     if(is_inv){
48         for(int i=1;i<N/2;++i)std::swap(out[i],
49             out[N-i]);
50     }
51     T invn=pow_mod(N,P-2,P);
52     for(int i=0;i<N;++i)out[i]=out[i]*invn
53         %P;
54 }
55 }
56 }

```

7.14 random.cpp

```

1 inline int random_int(){
2     static int seed=20160424;
3     return seed+=(seed<<16)+0x1db3d743;
4 }
5 inline long long random_long_long(){
6     static long long seed=20160424;
7     return seed+=(seed<<32)+0xdb3d742c265539d;
8 }

```

7.15 外星模運算.cpp

```

1 //a[0]^(a[1]^a[2]^...)
2 #include<bits/stdc++.h>
3 using namespace std;
4 #define maxn 1000000
5 int euler[maxn+5];
6 bool is_prime[maxn+5];
7 inline void init_euler(){
8     is_prime[1]=1;//不是質數
9     for(int i=1;i<=maxn;i++)euler[i]=i;
10    for(int i=2;i<=maxn;i++){
11        if(!is_prime[i]){//是質數

```

```

12        euler[i]--;
13        for(int j=i<1;j<=maxn;j+=i){
14            is_prime[j]=1;
15            euler[j]=euler[j]/i*(i-1);
16        }
17    }
18 }
19 }
20 inline long long pow(long long a,long long b
21     ,long long mod){//a^b%mod
22     long long ans=1;
23     for(;b;a=a*a%mod,b>>=1)
24         if(b&1)ans=ans*a%mod;
25     return ans;
26 }
27 bool isless(long long *a,int n,int k){
28     if(*a==1)return k>1;
29     if(--n==0)return *a<k;
30     int next=0;
31     for(long long b=1;b<k;++next)
32         b*=*a;
33     return isless(a+1,n,next);
34 }
35 long long high_pow(long long *a,int n,long
36     long mod){
37     if(*a==1||--n==0)return *a%mod;
38     int k=0,r=euler[mod];
39     for(long long tma=1;tma!=pow(*a,k+r,mod)
40         ;++k)
41         tma=tma*(*a)%mod;
42     if(isless(a+1,n,k))return pow(*a,high_pow(
43         a+1,n,k),mod);
44     int tmd=high_pow(a+1,n,r);
45     int t=(tmd-k+r)%r;
46     return pow(*a,k+t,mod);
47 }
48 }
49 long long a[1000005];
50 int t,mod;
51 int main(){
52     init_euler();
53     scanf("%d",&t);
54     #define n 4
55     while(t--){
56         for(int i=0;i<n;++i)scanf("%lld",&a[i]);
57         scanf("%d",&mod);
58         printf("%lld\n",high_pow(a,n,mod));
59     }
60     return 0;
61 }

```

7.16 模運算模板.cpp

```

1 template<typename T,long long mod>
2 struct mod_t{//mod只能是質數
3     T data;
4     mod_t(){}
5     mod_t(const T &d):data((d%mod+mod)%mod){}
6     mod_t pow(T b)const{
7         mod_t ans(1);
8         for(mod_t now=*this;b;now=now*b,b/=2)
9             if(b%2)ans=ans*now;
10        return ans;
11    }

```

```

12    mod_t operator-(int)const{
13        return mod_t(mod-data);
14    }
15    mod_t operator+(const mod_t &b)const{
16        return mod_t((data+b.data)%mod);
17    }
18    mod_t operator-(const mod_t &b)const{
19        return mod_t((data-b.data)%mod);
20    }
21    mod_t operator*(const mod_t &b)const{
22        return mod_t((data*b.data)%mod);
23    }
24    mod_t operator/(const mod_t &b)const{
25        return *this*b.pow(mod-2);//*this *
26        Inverse(b)
27    }
28    operator T()const{return data;}
29    friend istream &operator>>(istream &i,
30        mod_t &b){
31        T d;
32        i>>d;
33        b=mod_t(d);
34        return i;
35    }
36 };

```

8 String

8.1 AC 自動機.cpp

```

1 template<char L='a',char R='z'>
2 class ac_automaton{
3     private:
4         struct joe{
5             int next[R-L+1],fail,efl,ed,cnt_dp,vis;
6         };
7         joe(){};
8         joe():ed(0),cnt_dp(0),vis(0){
9             for(int i=0;i<=R-L;++i)next[i]=0;
10        };
11     public:
12        std::vector<joe> S;
13        std::vector<int> q;
14        int qs,qe,vt;
15        ac_automaton():S(1),qs(0),qe(0),vt(0){}
16        void clear(){
17            q.clear();
18            S.resize(1);
19            for(int i=0;i<=R-L;++i)S[0].next[i]=0;
20            S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
21        }
22        void insert(const char *s){
23            int o=0;
24            for(int i=0;i<=s.length()-1){
25                id=s[i]-L;
26                if(!S[o].next[id]){
27                    S.push_back(joe());
28                    S[o].next[id]=S.size()-1;
29                }
30                o=S[o].next[id];
31            }
32        }

```

```

31 ++S[o].ed;
32 }
33 void build_fail(){
34     S[0].fail=S[0].efl=-1;
35     q.clear();
36     q.push_back(0);
37     ++qe;
38     while(qs!=qe){
39         int pa=q[qs++],id,t;
40         for(int i=0;i<R-L;++i){
41             t=S[pa].next[i];
42             if(!t)continue;
43             id=S[pa].fail;
44             while(~id&&!S[id].next[i])id=S[id]
45                 .fail;
46             S[t].fail=~id?S[id].next[i]:0;
47             S[t].efl=S[S[t].fail].ed?S[t].fail
48                 :S[S[t].fail].efl;
49             q.push_back(t);
50             ++qe;
51         }
52     }
53     /*DP出每個前綴在字串s出現的次數並傳回所
54     有字串被s匹配成功的次數O(N*M)*/
55     int match_0(const char *s){
56         int ans=0,id,p=0,i;
57         for(i=0;s[i];++i){
58             id=s[i]-L;
59             while(!S[p].next[id]&&p=S[p].fail;
60                 if(!S[p].next[id])continue;
61                 p=S[p].next[id];
62             ++S[p].cnt_dp; /*匹配成功則它所有後綴
63             都可以被匹配(DP計算)*/
64         }
65         for(i=qe-1;i>=0;--i){
66             ans+=S[q[i]].cnt_dp*S[q[i]].ed;
67             if(~S[q[i]].fail)S[q[i]].fail.
68                 cnt_dp+=S[q[i]].cnt_dp;
69         }
70         return ans;
71     }
72     /*多串匹配走efl邊並傳回所有字串被s匹配成
73     功的次數O(N*M*1.5)*/
74     int match_1(const char *s) const{
75         int ans=0,id,p=0,t;
76         for(int i=0;s[i];++i){
77             id=s[i]-L;
78             while(!S[p].next[id]&&p=S[p].fail;
79                 if(!S[p].next[id])continue;
80                 p=S[p].next[id];
81             if(S[p].ed)ans+=S[p].ed;
82             for(t=S[p].efl;~t;t=S[t].efl){
83                 ans+=S[t].ed; /*因為都走efl邊所以保
84                 證匹配成功*/
85             }
86         }
87         return ans;
88     }
89     /*枚舉(s的子串nA)的所有相異字串各恰一
90     次並傳回次數O(N*M^(1/3))*/
91     int match_2(const char *s){
92         int ans=0,id,p=0,t;
93         ++vt;

```

```

87     /*把截記vt+=1，只要vt沒溢位，所有S[p].
88     vis==vt就會變成false
89     這種利用vt的方法可以O(1)歸零vis陣列*/
90     for(int i=0;s[i];++i){
91         id=s[i]-L;
92         while(!S[p].next[id]&&p=S[p].fail;
93             if(!S[p].next[id])continue;
94             p=S[p].next[id];
95         if(S[p].ed&&S[p].vis!=vt){
96             S[p].vis=vt;
97             ans+=S[p].ed;
98         }
99         for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t]
100             .efl){
101             S[t].vis=vt;
102             ans+=S[t].ed; /*因為都走efl邊所以保
103             證匹配成功*/
104         }
105     }
106     return ans;
107 }
108 /*把AC自動機變成真的自動機*/
109 void evolution(){
110     for(qs=1;qs!=qe;){
111         int p=q[qs++];
112         for(int i=0;i<R-L;++i)
113             if(S[p].next[i]==0)S[p].next[i]=S[
114                 S[p].fail].next[i];
115     }
116 }

```

8.2 hash.cpp

```

1 #define MAXN 1000000
2 #define prime_mod 1073676287
3 /*prime_mod 必須要是質數*/
4 typedef long long T;
5 char s[MAXN+5];
6 T h[MAXN+5]; /*hash陣列*/
7 T h_base[MAXN+5]; /*h_base[n]=(prime^n)%
8     prime_mod*/
9 inline void hash_init(int len,T prime=0
10     xdefaced){
11     h_base[0]=1;
12     for(int i=1;i<len;++i){
13         h[i]=(h[i-1]*prime+s[i-1])%prime_mod;
14         h_base[i]=(h_base[i-1]*prime)%prime_mod;
15     }
16 inline T get_hash(int l,int r){/*閉區間寫
17     法，設編號為0 ~ len-1*/
18     return (h[r+1]-(h[l]*h_base[r-l+1])%
19         prime_mod+prime_mod)%prime_mod;
20 }

```

8.3 KMP.cpp

```

1 /*產生fail function*/
2 inline void kmp_fail(char *s,int len,int *
3     fail){
4     int id=-1;
5     fail[0]=-1;
6     for(int i=1;i<len;++i){
7         while(~id&&s[id+1]!=s[i])id=fail[id];
8         if(s[id+1]==s[i])++id;
9         fail[i]=id;
10    }
11 }
12 /*以字串B匹配字串A，傳回匹配成功的數量(用B的
13     fail)*/
14 inline int kmp_match(char *A,int lenA,char *
15     B,int lenB,int *fail){
16     int id=-1,ans=0;
17     for(int i=0;i<lenA;++i){
18         while(~id&&B[id+1]!=A[i])id=fail[id];
19         if(B[id+1]==A[i])++id;
20         if(id==lenB-1){/*匹配成功*/
21             ++ans;
22             id=fail[id];
23         }
24     }
25     return ans;
26 }

```

8.4 manacher.cpp

```

1 //原字串: asdsasdsa
2 //先把字串變成這樣: @a#s#d#s#a#s#d#s#a#
3 inline void manacher(char *s,int len,int *z)
4 {
5     int l=0,r=0;
6     for(int i=1;i<len;++i){
7         z[i]=r>i?min(z[2*i-l],r-i):1;
8         while(s[i+z[i]]==s[i-z[i]])++z[i];
9         if(z[i]+i>r)r=z[i]+i,l=i;
10    }

```

8.5 minimal_string_rotation.cpp

```

1 int min_string_rotation(const string &s){
2     int n=s.size(),i=0,j=1,k=0;
3     while(i<n&&j<n&&k<n){
4         int t=s[(i+k)%n]-s[(j+k)%n];
5         ++k;
6         if(t>0)i+=k;
7         else j+=k;
8         if(i==j)++j;
9         k=0;
10    }
11    return min(i,j); /*傳回最小循環表示法起始位
12    置
13 }

```

8.6 suffix_array_lcp.cpp

```

1 #define radix_sort(x,y){\
2     for(i=0;i<A;++i)c[i]=0;\
3     for(i=0;i<len;++i)c[x[y[i]]]++;\
4     for(i=1;i<A;++i)c[i]+=c[i-1];\
5     for(i=len-1;i>=0;--i)sa[--c[x[y[i]]]]=y[i]
6     ];\
7 }
8 void suffix_array(const char *s,int len,int
9     *sa,int *rank,int *tmp,int *c){
10     int A='z'+1,i,k,id,*t;
11     for(i=0;i<len;++i){
12         tmp[i]=i;
13         rank[i]=s[i];
14     }
15     radix_sort(rank,tmp);
16     for(k=1;id<len-1;k<=1){
17         id=0;
18         for(i=len-k;i<len;++i)tmp[id++]=(i;
19             for(i=0;i<len;++i){
20                 if(sa[i]>=k)tmp[id++]=(sa[i]-k;
21             }
22     }
23     radix_sort(rank,tmp);
24     t=rank;rank=tmp;tmp=t;
25     id=0;
26     rank[sa[0]]=0;
27     for(i=1;i<len;++i){
28         if(tmp[sa[i-1]]!=tmp[sa[i]]||sa[i-1]+k
29             >=len||tmp[sa[i-1]+k]!=tmp[sa[i]+k]
30             )++id;
31         rank[sa[i]]=id;
32     }
33     A=id+1;
34 }
35 #undef radix_sort
36 //h:高度數組 sa:後綴數組 rank:排名
37 inline void suffix_array_lcp(const char *s,
38     int len,int *h,int *sa,int *rank){
39     for(int i=0;i<len;++i)rank[sa[i]]=i;
40     for(int i=0,k=0;i<len;++i){
41         if(rank[i]==0)continue;
42         if(k--<0)
43             while(s[i+k]==s[sa[rank[i]-1]+k])++k;
44         h[rank[i]]=k;
45     }
46     h[0]=0;
47 }

```

8.7 Z.cpp

```

1 inline void z_alg(char *s,int len,int *z){
2     int l=0,r=0;
3     z[0]=len;
4     for(int i=1;i<len;++i){
5         z[i]=i>r?0:(i-l+z[i-l]<z[i-l]?z[i-l]:r-i
6             +1);
7         while(i+z[i]<len&&s[i+z[i]]==s[z[i]]++)z[i]
8             ++i;
9     }
10 }

```


9 | }

9 Tarjan

9.1 dominator_tree.cpp

```

1 struct dominator_tree{
2     static const int MAXN=5005;
3     int n;// 1-base
4     vector<int> suc[MAXN],pre[MAXN]; //存圖和反
        向圖
5     int fa[MAXN],dfn[MAXN],id[MAXN],Time;//for
        dfs
6     int semi[MAXN],idom[MAXN];
7     int anc[MAXN],best[MAXN]; //disjoint set
8     vector<int> dom[MAXN]; //dominator_tree存這
        裡
9     void init(int _n){
10         n=_n;
11         for(int i=1;i<=n;++i)suc[i].clear(),pre[
            i].clear();
12     }
13     void add_edge(int u,int v){
14         suc[u].push_back(v);
15         pre[v].push_back(u);
16     }
17     void dfs(int u){
18         dfn[u]++Time,id[Time]=u;
19         for(auto v:suc[u]){
20             if(dfn[v])continue;
21             dfs(v),fa[dfn[v]]=dfn[u];
22         }
23     }
24     int find(int x){
25         if(x==anc[x])return x;
26         int y=find(anc[x]);
27         if(semi[best[x]]>semi[best[anc[x]]])best
            [x]=best[anc[x]];
28         return anc[x]=y;
29     }
30     void tarjan(int r){
31         Time=0;
32         for(int t=1;t<=n;++t){
33             dfn[t]=idom[t]=0; //u=r或是u無法到達r時
                idom[id[u]]=0
34             dom[t].clear();
35             anc[t]=best[t]=semi[t]=t;
36         }
37         dfs(r);
38         for(int y=Time;y>=2;--y){
39             int x=fa[y],idy=id[y];
40             for(auto z:pre[idy]){
41                 if(!(z=dfn[z]))continue;
42                 find(z);
43                 semi[y]=min(semi[y],semi[best[z]]);
44             }
45             dom[semi[y]].push_back(y);
46             anc[y]=x;
47             for(auto z:dom[x]){
48                 find(z);

```

```

49         idom[z]=semi[best[z]]<x?best[z]:x;
50     }
51     dom[x].clear();
52 }
53 for(int u=2;u<=Time;++u){
54     if(idom[u]!=semi[u])idom[u]=idom[idom[
        u]];
55     dom[id[idom[u]]].push_back(id[u]);
56 }
57 }
58 }dom;

```

9.2 tnfsb017_2_sat.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define MAXN 8001
4 #define MAXN2 MAXN*4
5 #define n(X) ((X)+2*N)
6 vector<int> v[MAXN2];
7 vector<int> rv[MAXN2];
8 vector<int> vis_t;
9 int N,M;
10 void addedge(int s,int e){
11     v[s].push_back(e);
12     rv[e].push_back(s);
13 }
14 int scc[MAXN2];
15 bool vis[MAXN2]={false};
16 void dfs(vector<int> *uv,int n,int k=-1){
17     vis[n]=true;
18     for(int i=0;i<uv[n].size();++i)
19         if(!vis[uv[n][i]])
20             dfs(uv,uv[n][i],k);
21     if(uv==v)vis_t.push_back(n);
22     scc[n]=k;
23 }
24 void solve(){
25     for(int i=1;i<=N;++i){
26         if(!vis[i])dfs(v,i);
27         if(!vis[n(i)])dfs(v,n(i));
28     }
29     memset(vis,0,sizeof(vis));
30     int c=0;
31     for(int i=vis_t.size()-1;i>=0;--i)
32         if(!vis[vis_t[i]])
33             dfs(rv,vis_t[i],c++);
34 }
35 int main(){
36     int a,b;
37     scanf("%d%d",&N,&M);
38     for(int i=1;i<=N;++i){
39         // (A or B)&(!A & !B) A^B
40         a=i*2-1;
41         b=i*2;
42         addedge(n(a),b);
43         addedge(n(b),a);
44         addedge(a,n(b));
45         addedge(b,n(a));
46     }
47     while(M--){
48         scanf("%d%d",&a,&b);
49         a = a>0?a*2-1:-a*2;

```

```

50         b = b>0?b*2-1:-b*2;
51         // A or B
52         addedge(n(a),b);
53         addedge(n(b),a);
54     }
55     solve();
56     bool check=true;
57     for(int i=1;i<=2*N;++i)
58         if(scc[i]==scc[n(i)])
59             check=false;
60     if(check){
61         printf("%d\n",N);
62         for(int i=1;i<=2*N;i+=2){
63             if(scc[i]>scc[i+2*N])
64                 putchar('+');
65             else
66                 putchar('-');
67         }
68         putchar('\n');
69     }else puts("0");
70     return 0;
71 }

```

9.3 橋連通分量.cpp

```

1 #define N 1005
2 struct edge{
3     int u,v;
4     bool is_bridge;
5     edge(int u=0,int v=0):u(u),v(v),is_bridge
        (0){}
6 };
7 vector<edge> E;
8 vector<int> G[N]; // 1-base
9 int low[N],vis[N],Time;
10 int bcc_id[N],bridge_cnt,bcc_cnt; // 1-base
11 int st[N],top; //BCC用
12 inline void add_edge(int u,int v){
13     G[u].push_back(E.size());
14     E.push_back(edge(u,v));
15     G[v].push_back(E.size());
16     E.push_back(edge(v,u));
17 }
18 void dfs(int u,int re=-1){ //u當前點，re為u連
        接前一個點的邊
19     int v;
20     low[u]=vis[u]++Time;
21     st[top++] = u;
22     for(size_t i=0;i<G[u].size();++i){
23         int e=G[u][i];v=E[e].v;
24         if(!vis[v]){
25             dfs(v,e^1); //e^1反向邊
26             low[u]=min(low[u],low[v]);
27             if(vis[u]<low[v]){
28                 E[e].is_bridge=E[e^1].is_bridge=1;
29                 ++bridge_cnt;
30             }
31         }else if(vis[v]<vis[u]&&v!=re)
32             low[u]=min(low[u],vis[v]);
33     }
34     if(vis[u]==low[u]){ //處理BCC
35         ++bcc_cnt; // 1-base

```

```

36         do bcc_id[v=st[--top]]=bcc_cnt; //每個點
            所在的BCC
37         while(v!=u);
38     }
39 }
40 inline void bcc_init(int n){
41     Time=bcc_cnt=bridge_cnt=top=0;
42     E.clear();
43     for(int i=1;i<=n;++i){
44         G[i].clear();
45         vis[i]=bcc_id[i]=0;
46     }
47 }

```

9.4 雙連通分量 & 割點.cpp

```

1 #define N 1005
2 vector<int> G[N]; // 1-base
3 vector<int> bcc[N]; //存每塊雙連通分量的點
4 int low[N],vis[N],Time;
5 int bcc_id[N],bcc_cnt; // 1-base
6 bool is_cut[N]; //是否為割點，割點的bcc_id沒
        意義
7 int st[N],top;
8 void dfs(int u,int pa=-1){ //u當前點，pa父親
9     int v,child=0;
10     low[u]=vis[u]++Time;
11     st[top++] = u;
12     for(size_t i=0;i<G[u].size();++i){
13         if(!vis[v=G[u][i]]){
14             dfs(v,u,++child;
15             low[u]=min(low[u],low[v]);
16             if(vis[u]<=low[v]){
17                 is_cut[u]=1;
18                 bcc[++bcc_cnt].clear();
19                 int t;
20                 do{
21                     bcc_id[t=st[--top]]=bcc_cnt;
22                     bcc[bcc_cnt].push_back(t);
23                 }while(t!=v);
24                 bcc_id[u]=bcc_cnt;
25                 bcc[bcc_cnt].push_back(u);
26             }
27             else if(vis[v]<vis[u]&&v!=pa) //反向邊
28                 low[u]=min(low[u],vis[v]);
29         }
30         if(pa!=-1&&child<2)is_cut[u]=0; //u是dfs樹
            的根要特判
31     }
32 }
33 inline void bcc_init(int n){
34     Time=bcc_cnt=top=0;
35     for(int i=1;i<=n;++i){
36         G[i].clear();
37         is_cut[i]=vis[i]=bcc_id[i]=0;
38     }
39 }

```

10 Tree_problem

10.1 HeavyLight.cpp

```
1 #include<vector>
2 #define MAXN 100005
3 typedef std::vector<int> >::iterator VIT;
4 int siz[MAXN], max_son[MAXN], pa[MAXN], dep[
    MAXN];
5 /*節點大小、大小最大的孩子、父母節點、深度*/
6 int link_top[MAXN], link[MAXN], cnt;
7 /*每個點所在鍊的鏈頭、樹鏈剖分的DFS序、時間
    戳*/
8 std::vector<int> >G[MAXN]; /*用vector存樹*/
9 void find_max_son(int x){
10     siz[x]=1;
11     max_son[x]=-1;
12     for(VIT i=G[x].begin(); i!=G[x].end(); ++i){
13         if(*i==pa[x]) continue;
14         pa[*i]=x;
15         dep[*i]=dep[x]+1;
16         find_max_son(*i);
17         if(max_son[x]==-1 || siz[*i]>siz[max_son[x]
            ]) max_son[x]=*i;
18         siz[x]+=siz[*i];
19     }
20 }
21 void build_link(int x, int top){
22     link[x]=++cnt; /*記錄x點的時間戳*/
23     link_top[x]=top;
24     if(max_son[x]==-1) return;
25     build_link(max_son[x], top); /*優先走訪最大
        孩子*/
26     for(VIT i=G[x].begin(); i!=G[x].end(); ++i){
27         if(*i==max_son[x] || *i==pa[x]) continue;
28         build_link(*i, *i);
29     }
30 }
31 inline int find_lca(int a, int b){
32     /*求LCA，可以在過程中對區間進行處理*/
33     int ta=link_top[a], tb=link_top[b];
34     while(ta!=tb){
35         if(dep[ta]<dep[tb]){
36             std::swap(ta, tb);
37             std::swap(a, b);
38         }
39         /*這裡可以對a所在的鏈做區間處理
40         /*區間為(Link[ta], Link[a])
41         ta=link_top[a=pa[ta]];
42     }
43     /*最後a,b會在同一條鏈，若a!=b還要在進行一
44     次區間處理*/
45     return dep[a]<dep[b]?a:b;
```

10.2 LCA.cpp

```
1 #define MAXN 100000
```

```
2 #define MAX_LOG 17
3 int pa[MAX_LOG+1][MAXN+5];
4 int dep[MAXN+5];
5 vector<int> G[MAXN+5];
6 void dfs(int x, int p){ //dfs(1, -1);
7     pa[0][x]=p;
8     for(int i=0; i+1<MAX_LOG; ++i) pa[i+1][x]=pa[
        i][pa[i][x]];
9     for(auto &i:G[x]){
10         if(i==p) continue;
11         dep[i]=dep[x]+1;
12         dfs(i, x);
13     }
14 }
15 inline int jump(int x, int d){
16     for(int i=0; i<d; ++i) if((x>>i)&1) x=pa[i][x]
        ;
17     return x;
18 }
19 inline int find_lca(int a, int b){
20     if(dep[a]>dep[b]) swap(a, b);
21     b=jump(b, dep[b]-dep[a]);
22     if(a==b) return a;
23     for(int i=MAX_LOG; i>0; --i){
24         if(pa[i][a]!=pa[i][b]){
25             a=pa[i][a];
26             b=pa[i][b];
27         }
28     }
29     return pa[0][a];
30 }
```

10.3 link_cut_tree.cpp

```
1 #include<vector>
2 struct splay_tree{
3     int ch[2], pa; /*子節點跟父母*/
4     bool rev; /*反轉的懶惰標記*/
5     splay_tree(): pa(0), rev(0){ ch[0]=ch[1]=0; }
6 };
7 std::vector<splay_tree> node;
8 /*
9 有的時候用vector會TLE，要注意
10 這邊以node[0]作為null節點
11 */
12 inline bool isroot(int x){ /*判斷是否為這棵
    splay tree的根*/
13     return node[node[x].pa].ch[0]!=x && node[
        node[x].pa].ch[1]!=x;
14 }
15 inline void down(int x){ /*懶惰標記下推*/
16     if(node[x].rev){
17         if(node[x].ch[0]) node[node[x].ch[0]].rev
            ^=1;
18         if(node[x].ch[1]) node[node[x].ch[1]].rev
            ^=1;
19         std::swap(node[x].ch[0], node[x].ch[1]);
20         node[x].rev=1;
21     }
22 }
23 void push_down(int x){ /*將所有祖先的懶惰標記
    下推*/
```

```
24     if(!isroot(x)) push_down(node[x].pa);
25     down(x);
26 }
27 inline void up(int x){ /*將子節點的資訊向上
    更新*/
28 inline void rotate(int x){ /*旋轉，會自行判斷
    轉的方向*/
29     int y=node[x].pa, z=node[y].pa, d=(node[y].
        ch[1]==x);
30     node[x].pa=z;
31     if(!isroot(y)) node[z].ch[node[z].ch[1]==y
        ]=x;
32     node[y].ch[d]=node[x].ch[d^1];
33     node[node[y].ch[d]].pa=y;
34     node[y].pa=x, node[x].ch[d^1]=y;
35     up(y);
36     up(x);
37 }
38 inline void splay(int x){ /*將節點x伸展到所在
    splay tree的根*/
39     push_down(x);
40     while(!isroot(x)){
41         int y=node[x].pa;
42         if(!isroot(y)){
43             int z=node[y].pa;
44             if((node[z].ch[0]==y)^(node[y].ch[0]==
                x)) rotate(y);
45             else rotate(x);
46         }
47         rotate(x);
48     }
49 }
50 inline int access(int x){
51     int last=0;
52     while(x){
53         splay(x);
54         node[x].ch[1]=last;
55         up(x);
56         last=x;
57         x=node[x].pa;
58     }
59     return last; /*回傳access後splay tree的根*/
60 }
61 inline void access(int x, bool is=0){ /*is=0就
    是一般的access*/
62     int last=0;
63     while(x){
64         splay(x);
65         if(is && !node[x].pa){
66             //printf("%d\n", max(node[last].ma, node
                [node[x].ch[1]].ma));
67         }
68         node[x].ch[1]=last;
69         up(x);
70         last=x;
71         x=node[x].pa;
72     }
73 }
74 inline void query_edge(int u, int v){
75     access(u);
76     access(v, 1);
77 }
78 inline void make_root(int x){
79     access(x), splay(x);
```

```
80     node[x].rev^=1;
81 }
82 inline void make_root(int x){
83     node[access(x)].rev^=1;
84     splay(x);
85 }
86 inline void cut(int x, int y){
87     make_root(x);
88     access(y);
89     splay(y);
90     node[y].ch[0]=0;
91     node[x].pa=0;
92 }
93 inline void cut_parents(int x){
94     access(x);
95     splay(x);
96     node[node[x].ch[0]].pa=0;
97     node[x].ch[0]=0;
98 }
99 inline void link(int x, int y){
100     make_root(x);
101     node[x].pa=y;
102 }
103 inline int find_root(int x){
104     x=access(x);
105     while(node[x].ch[0]) x=node[x].ch[0];
106     splay(x);
107     return x;
108 }
109 inline int query(int u, int v){
110     /*
111     傳回uv路徑splay tree的根結點
112     這種寫法無法求LCA
113     */
114     make_root(u);
115     return access(v);
116 }
117 inline int query_lca(int u, int v){
118     /*假設求鏈上點權的總和，sum是子樹的權重和，
119     data是節點的權重*/
119     access(u);
120     int lca=access(v);
121     splay(u);
122     if(u==lca){
123         //return node[lca].data+node[node[lca].
            ch[1]].sum;
124     }else{
125         //return node[lca].data+node[node[lca].
            ch[1]].sum+node[u].sum;
126     }
127 }
128 struct EDGE{
129     int a, b, w;
130 }e[10005];
131 int n;
132 std::vector<std::pair<int, int> > >G[10005];
133 /*first表示子節點，second表示邊的編號*/
134 int pa[10005], edge_node[10005];
135 /*pa是父母節點，暫存用的，edge_node是每個編
    被存在哪個點裡面的陣列*/
136 inline void bfs(int root){
137     /*在建構的時候把每個點都設成一個splay tree，
    不會壞掉*/
138     std::queue<int> > q;
```

```

139 for(int i=1;i<=n;++i)pa[i]=0;
140 q.push(root);
141 while(q.size()){
142     int u=q.front();
143     q.pop();
144     for(int i=0;i<(int)G[u].size();++i){
145         int v=G[u][i].first;
146         if(v!=pa[u]){
147             pa[v]=u;
148             node[v].pa=u;
149             node[v].data=e[G[u][i].second].w;
150             edge_node[G[u][i].second]=v;
151             up(v);
152             q.push(v);
153         }
154     }
155 }
156 }
157 inline void change(int x,int b){
158     splay(x);
159     //node[x].data=b;
160     up(x);
161 }

```

10.4 POJ_tree.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define MAXN 10005
4 int n,k;
5 vector<pair<int,int> >g[MAXN];
6 int size[MAXN];
7 bool vis[MAXN];
8 inline void init(){
9     for(int i=0;i<=n;++i){
10         g[i].clear();
11         vis[i]=0;
12     }
13 }
14 void get_dis(vector<int> &dis,int u,int pa,
15     int d){
16     dis.push_back(d);
17     for(size_t i=0;i<g[u].size();++i){
18         int v=g[u][i].first,w=g[u][i].second;
19         if(v!=pa&&!vis[v])get_dis(dis,v,u,d+w);
20     }
21 }
22 vector<int> dis;//這東西如果放在函數裡會TLE
23 int cal(int u,int d){
24     dis.clear();
25     get_dis(dis,u,-1,d);
26     sort(dis.begin(),dis.end());
27     int l=0,r=dis.size()-1,res=0;
28     while(l<r){
29         while(l<r&&dis[l]+dis[r]>k)--r;
30         res+=r-(l++);
31     }
32     return res;
33 }
34 pair<int,int> tree_centroid(int u,int pa,
35     const int sz){
36     size[u]=1;//找樹重心，second是重心

```

```

35 pair<int,int> res(INT_MAX,-1);
36 int ma=0;
37 for(size_t i=0;i<g[u].size();++i){
38     int v=g[u][i].first;
39     if(v==pa||vis[v])continue;
40     res=min(res,tree_centroid(v,u,sz));
41     size[u]+=size[v];
42     ma=max(ma,size[v]);
43 }
44 ma=max(ma,sz-size[u]);
45 return min(res,make_pair(ma,u));
46 }
47 int tree_DC(int u,int sz){
48     int center=tree_centroid(u,-1,sz).second;
49     int ans=cal(center,0);
50     vis[center]=1;
51     for(size_t i=0;i<g[center].size();++i){
52         int v=g[center][i].first,w=g[center][i].
53             second;
54         if(vis[v])continue;
55         ans-=cal(v,w);
56         ans+=tree_DC(v,size[v]);
57     }
58     return ans;
59 }
60 int main(){
61     while(scanf("%d%d",&n,&k),n||k){
62         init();
63         for(int i=1;i<=n;++i){
64             int u,v,w;
65             scanf("%d%d%d",&u,&v,&w);
66             g[u].push_back(make_pair(v,w));
67             g[v].push_back(make_pair(u,w));
68         }
69         printf("%d\n",tree_DC(1,n));
70     }
71     return 0;

```

ACM ICPC Team Reference - NTHU Jinkela

Contents

1 Computational_Geometry	1	2.2 Dynamic_KD_tree.cpp	4	5.5 graphISO.cpp	8	7.13NTT.cpp	15
1.1 Geometry.cpp	1	2.3 kd_tree_replace_segment_tree.cpp	4	5.6 KM.cpp	9	7.14random.cpp	15
1.2 SmallestCircle.cpp	3	2.4 persistent_segment_tree.cpp	5	5.7 MaximumClique.cpp	9	7.15外星模運算.cpp	15
1.3 最近點對.cpp	3	2.5 reference_point.cpp	5	5.8 Minimum_General_Weighted_Matching.cpp	9	7.16模運算模板.cpp	15
1.4 浮點數誤差模板.cpp	3	2.6 skew_heap.cpp	6	5.9 Rectilinear_Steiner_tree.cpp	9		
2 Data_Structure	3	2.7 split_merge.cpp	6	5.10treeISO.cpp	10	8 String	15
2.1 DLX.cpp	3	2.8 treap.cpp	6	5.11一般圖最大權匹配.cpp	10	8.1 AC 自動機.cpp	15
		2.9 操作分治.cpp	6			8.2 hash.cpp	16
		2.10 整體二分.cpp	6	6 language	11	8.3 KMP.cpp	16
				6.1 CNF.cpp	11	8.4 manacher.cpp	16
				6.2 earley.cpp	11	8.5 minimal_string_rotation.cpp	16
		3 default	6	7 Number_Theory	12	8.6 suffix_array_lcp.cpp	16
		3.1 debug.cpp	6	7.1 basic.cpp	12	8.7 Z.cpp	16
		3.2 IncStack.cpp	7	7.2 bit_set.cpp	13	9 Tarjan	17
		3.3 input.cpp	7	7.3 cantor_expansion.cpp	13	9.1 dominator_tree.cpp	17
				7.4 Chinese_remainder_theorem.cpp	13	9.2 tnfsb017_2_sat.cpp	17
		4 Flow	7	7.5 enumerate.cpp	13	9.3 橋連通分量.cpp	17
		4.1 dinic.cpp	7	7.6 eulerphi.cpp	13	9.4 雙連通分量 & 割點.cpp	17
		4.2 ISAP.cpp	7	7.7 Factor.cpp	13		
		4.3 MinCostMaxFlow.cpp	7	7.8 FFT.cpp	14	10 Tree_problem	18
				7.9 find_real_root.cpp	14	10.1HeavyLight.cpp	18
		5 Graph	8	7.10Gauss_Elimination.cpp	14	10.2LCA.cpp	18
		5.1 Arborescence_EV.cpp	8	7.11LinearCongruence.cpp	14	10.3link_cut_tree.cpp	18
		5.2 Augmenting_Path.cpp	8	7.12Lucas.cpp	14	10.4POJ_tree.cpp	19
		5.3 Augmenting_Path_multiple.cpp	8				
		5.4 blossom_matching.cpp	8				