

1 Computational_Geometry

1.1 Geometry.cpp

```

1 const double PI=atan2(0.0,-1.0);
2 template<typename T>
3 struct point{
4     T x,y;
5     point(){
6     point(const T&x,const T&y):x(x),y(y){
7     point operator+(const point &b)const{
8         return point(x+b.x,y+b.y);
9     point operator-(const point &b)const{
10        return point(x-b.x,y-b.y);
11    point operator*(const T &b)const{
12        return point(x*b,y*b);
13    point operator/(const T &b)const{
14        return point(x/b,y/b);
15    bool operator==(const point &b)const{
16        return x==b.x&&y==b.y;
17    T dot(const point &b)const{
18        return x*b.x+y*b.y;
19    T cross(const point &b)const{
20        return x*b.y-y*b.x;
21    point normal()const{//求法向量
22        return point(-y,x);
23    T abs2()const{//向量長度的平方
24        return dot(*this);
25    }
26    T rad(const point &b)const{//兩向量的弧度
27        return fabs(atan2(fabs(cross(b)),dot(b)));
28    }
29    T getA()const{//對x軸的弧度
30        T A=atan2(y,x);{//超過180度會變負的
31        if(A<=-PI/2)A+=PI*2;
32        return A;
33    }
34    };
35    template<typename T>
36    struct line{
37        line(){
38        point<T> p1,p2;
39        T a,b,c;//ax+by+c=0
40        line(const point<T>&x,const point<T>&y):p1(x),p2(y){
41        void pton()const{//轉成一般式
42            a=p1.y-p2.y;
43            b=p2.x-p1.x;
44            c=-a*p1.x-b*p1.y;
45        }
46        T cross(const point<T> &p)const{//點和有向
47            直線的關係，>0左邊、=0在線上、<0右邊
48            return (p2-p1).cross(p-p1);
49        }
50        bool point_on_segment(const point<T>&p)
51            const{//點是否線段上
52            return cross(p)==0&&(p1-p).dot(p2-p)<=0;
53        }
54        T dis2(const point<T> &p,bool is_segment
55            =0)const{//點跟直線/線段的距離平方
56        point<T> v=p2-p1,v1=p-p1;
57        if(is_segment){
58            point<T> v2=p-p2;
59            if(v.dot(v1)<=0)return v1.abs2();
60            if(v.dot(v2)>=0)return v2.abs2();
61        }
62        T tmp=v.cross(v1);
63        return tmp*tmp/v.abs2();
64    }
65    T seg_dis2(const line<T> &l)const{//兩線段
66        距離平方
67        return min({dis2(l.p1,1),dis2(l.p2,1),l.
68            dis2(p1,1),l.dis2(p2,1)});
69    }
70    point<T> projection(const point<T> &p)
71        const{//點對直線的投影
72    point<T> n=(p2-p1).normal();
73    return p-n*(p-p1).dot(n)/n.abs2();
74    }
75    point<T> mirror(const point<T> &p)const{//
76        點對直線的鏡射
77    //要先呼叫pton轉成一般式
78    point<T> ans;
79    T d=a*p+b*b;
80    ans.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/
81        d;
82    ans.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/
83        d;
84    return ans;
85    }
86    bool equal(const line &l)const{//直線相等
87    return cross(l.p1)==0&&cross(l.p2)==0;
88    }
89    bool parallel(const line &l)const{//直線平
90        行
91    return (p1-p2).cross(l.p1-l.p2)==0;
92    }
93    bool cross_seg(const line &l)const{//直線
94        是否交線段
95    return (p2-p1).cross(l.p1-p1)*(p2-p1).
96        cross(l.p2-p1)<=0;
97    }
98    }
99    char line_intersect(const line &l)const{//
100        直線相交情況，-1無限多點、1交於一點、0
101        不相交
102    return parallel(l)?(cross(l.p1)==0?-1:0)
103        :1;
104    }
105    char seg_intersect(const line &l)const{//
106        線段相交情況，-1無限多點、1交於一點、0
107        不相交
108    T c1=(p2-p1).cross(l.p1-p1);
109    T c2=(p2-p1).cross(l.p2-p1);
110    T c3=(l.p2-l.p1).cross(p1-l.p1);
111    T c4=(l.p2-l.p1).cross(p2-l.p1);
112    if(c1==0&&c2==0){
113        if(p1==l.p1&&(p2-p1).dot(l.p2)<=0)
114            return 1;
115        if(p1==l.p2&&(p2-p1).dot(l.p1)<=0)
116            return 1;
117        if(p2==l.p1&&(p1-p2).dot(l.p2)<=0)
118            return 1;
119        if(p2==l.p2&&(p1-p2).dot(l.p1)<=0)
120            return 1;
121    }
122    return 0;
123    }
124    }
125    template<typename T>
126    struct polygon{
127        polygon(){
128        vector<point<T>> p;//逆時針順序
129        T area()const{//面積
130            T ans=0;
131            for(int i=p.size()-1,j=0;j<(int)p.size()
132                ;i=j++){
133                ans+=p[i].cross(p[j]);
134            }
135            return ans/2;
136        }
137        point<T> center_of_mass()const{//重心
138            T cx=0,cy=0,w=0;
139            for(int i=p.size()-1,j=0;j<(int)p.size()
140                ;i=j++){
141                T a=p[i].cross(p[j]);
142                cx+=(p[i].x+p[j].x)*a;
143                cy+=(p[i].y+p[j].y)*a;
144                w+=a;
145            }
146            return point<T>(cx/3/w,cy/3/w);
147        }
148    }
149    char ahas(const point<T>&t)const{//點是否
150        在簡單多邊形內，是的話回傳1、在邊上回
151        傳-1、否則回傳0
152    bool c=0;
153    for(int i=0,j=p.size()-1;i<p.size();i=i
154        ++){
155        if(line<T>(p[i],p[j]).point_on_segment
156            (t))return -1;
157        else if((p[i].y>t.y)!=p[j].y>t.y)&&
158            t.x<(p[j].x+p[i].x)*(t.y-p[i].y)/(p[j]
159                .y-p[i].y)+p[i].x)
160            c=!c;
161    }
162    return c;
163    }
164    }
165    return c;
166    }
167    char point_in_convex(const point<T>&x)
168        const{
169    int l=1,r=(int)p.size()-2;
170    while(l<=r){//點是否在凸多邊形內，是的話
171        回傳1、在邊上回傳-1、否則回傳0
172        int mid=(l+r)/2;
173        T a1=(p[mid]-p[0]).cross(x-p[0]);
174        T a2=(p[mid+1]-p[0]).cross(x-p[0]);
175        if(a1>=0&&a2<=0){
176            T res=(p[mid+1]-p[mid]).cross(x-p[
177                mid]);
178            return res>0?1:(res>=0?-1:0);
179        }else if(a1<0)r=mid-1;
180        else l=mid+1;
181    }
182    return 0;
183    }
184    vector<T> getA()const{//凸包邊對x軸的夾角
185    vector<T> res;//一定是遞增的
186    for(size_t i=0;i<p.size();i++){
187        res.push_back((p[(i+1)%p.size()]-p[i])
188            .getA());
189    }
190    return res;
191    }
192    bool line_intersect(const vector<T>&A,
193        const line<T> &l)const{//O(LogN)
194    int f1=upper_bound(A.begin(),A.end(),(l.
195        p1-l.p2).getA())-A.begin();
196    int f2=upper_bound(A.begin(),A.end(),(l.
197        p2-l.p1).getA())-A.begin();
198    return l.cross_seg(line<T>(p[f1],p[f2]));
199    }
200    }
201    polygon cut(const line<T> &l)const{//凸包
202        對直線切割，得到直線L左側的凸包
203    polygon ans;
204    for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
205        if(l.cross(p[i])>=0){
206            ans.p.push_back(p[i]);
207            if(l.cross(p[j])<0)
208                ans.p.push_back(l.
209                    line_intersection(line<T>(p[i]
210                        ,p[j])));
211        }else if(l.cross(p[j])>=0)
212            ans.p.push_back(l.line_intersection(
213                line<T>(p[i],p[j]));
214        }
215    }
216    return ans;
217    }
218    static bool graham_cmp(const point<T>&a,
219        const point<T>&b){
220    return (a.x<b.x)||((a.x==b.x&&a.y<b.y));
221    }
222    }
223    void graham(vector<point<T>> &s){{//凸包
224    sort(s.begin(),s.end(),graham_cmp);
225    p.resize(s.size()+1);
226    int m=0;
227    for(int i=0;i<(int)s.size();i++){
228        while(m>2&&(p[m-1]-p[m-2]).cross(s[i]
229            -p[m-2])<=0)-m;
230        p[m++]=s[i];
231    }
232    }

```

```

198 }
199 for(int i=s.size()-2,t=m+1;i>=0;--i){
200     while(m>=t&&(p[m-1]-p[m-2]).cross(s[i]
201         ]-p[m-2])<=0)--m;
202     p[m++]=s[i];
203 }
204 if(s.size()>1)--m;
205 p.resize(m);
206 }
207 T diam(){//直徑
208     int n=p.size(),t=1;
209     T ans=0;p.push_back(p[0]);
210     for(int i=0;i<n;i++){
211         point<T> now=p[i+1]-p[i];
212         while(now.cross(p[t+1]-p[i])>now.cross
213             (p[t]-p[i]))t=(t+1)%n;
214         ans=max(ans,max((p[i]-p[t]).abs2(),(p[
215             i+1]-p[t+1]).abs2()));
216     }
217     return p.pop_back(),ans;
218 }
219 T min_cover_rectangle(){//最小覆蓋矩形
220     int n=p.size(),t=1,r=1,l;
221     if(n<3)return 0;//也可以做最小周長矩形
222     T ans=1e99;p.push_back(p[0]);
223     for(int i=0;i<n;i++){
224         point<T> now=p[i+1]-p[i];
225         while(now.cross(p[t+1]-p[i])>now.cross
226             (p[t]-p[i]))t=(t+1)%n;
227         while(now.dot(p[r+1]-p[i])>now.dot(p[r
228             ]-p[i]))r=(r+1)%n;
229         if(l==r;
230         while(now.dot(p[l+1]-p[i])<=now.dot(p[
231             l]-p[i]))l=(l+1)%n;
232         T d=now.abs2();
233         T tmp=now.cross(p[t]-p[i])*(now.dot(p[
234             r]-p[i])-now.dot(p[l]-p[i]))/d;
235         ans=min(ans,tmp);
236     }
237     return p.pop_back(),ans;
238 }
239 T max_triangle(){//最大內接三角形
240     int n=p.size(),a=1,b=2;
241     if(n<3)return 0;
242     T ans=0,tmp;p.push_back(p[0]);
243     for(int i=0;i<n;i++){
244         while((p[a]-p[i]).cross(p[b+1]-p[i])>
245             (p[a]-p[i]).cross(p[b]-p[i]))
246             b=(b+1)%n;
247         ans=max(ans,tmp);
248         while((p[a+1]-p[i]).cross(p[b]-p[i])>
249             (p[a]-p[i]).cross(p[b]-p[i]))
250             a=(a+1)%n;
251         ans=max(ans,tmp);
252     }
253     return p.pop_back(),ans/2;
254 }
255 T dis2(polygon &p1){//凸包最近距離平方
256     vector<point<T> > &P=p,Q=p1.p;
257     int n=P.size(),m=Q.size(),l=0,r=0;
258     for(int i=0;i<n;i++){
259         if(P[i].y<P[l].y)l=i;
260         for(int i=0;i<m;i++){
261             if(Q[i].y<Q[r].y)r=i;
262         }
263         P.push_back(P[0]),Q.push_back(Q[0]);
264     }
265     T ans=1e99;
266     for(int i=0;i<n;i++){
267         while((P[i]-P[l+1]).cross(Q[r+1]-Q[r])
268             <0)r=(r+1)%m;
269         ans=min(ans,line<T>(P[l],P[l+1]).
270             seg_dis2(line<T>(Q[r],Q[r+1])));
271         l=(l+1)%n;
272     }
273     return P.pop_back(),Q.pop_back(),ans;
274 }
275 static char sign(const point<T>&t){
276     return (t.y==0?t.x:t.y)<0;
277 }
278 static bool angle_cmp(const line<T> &A,
279     const line<T> &B){
280     point<T> a=A.p2-A.p1,b=B.p2-B.p1;
281     return sign(a)<sign(b)||((sign(a)==sign(b)
282         )&&a.cross(b)>0);
283 }
284 int halfplane_intersection(vector<line<T>
285     > &s){//半平面交
286     sort(s.begin(),s.end(),angle_cmp);
287     //左側為該線段半平面
288     int L,R,n=s.size();
289     vector<point<T> > px(n);
290     vector<line<T> > q(n);
291     q[L=R=0]=s[0];
292     for(int i=1;i<n;i++){
293         while(L<R&&s[i].cross(px[R-1])<=0)--R;
294         while(L<R&&s[i].cross(px[L])<=0)+L;
295         q[++R]=s[i];
296         if(q[R].parallel(q[R-1])){
297             --R;
298             if(q[R].cross(s[i].p1)>0)q[R]=s[i];
299         }
300         if(L<R)px[R-1]=q[R-1].
301             line_intersection(q[R]);
302     }
303     while(L<R&&q[L].cross(px[R-1])<=0)--R;
304     p.clear();
305     if(R-L==1)return 0;
306     px[R]=q[R].line_intersection(q[L]);
307     for(int i=L;i<R;i++)p.push_back(px[i]);
308     return R-L+1;
309 }
310 template<typename T>
311 struct triangle{
312     point<T> a,b,c;
313     triangle(){
314         triangle(const point<T> &a,const point<T>
315             &b,const point<T> &c):a(a),b(b),c(c){
316             T area(){const
317                 T t=(b-a).cross(c-a)/2;
318                 return t>0?t:-t;
319             }
320         }
321     point<T> barycenter(){const{//重心
322         return (a+b+c)/3;
323     }
324     point<T> circumcenter(){const{//外心
325         static line<T> u,v;
326         u.p1=(a+b)/2;
327         u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
328             b.x);
329         v.p1=(a+c)/2;
330         v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
331             c.x);
332         return u.line_intersection(v);
333     }
334     point<T> incenter(){const{//內心
335         T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
336             ()),C=sqrt((a-b).abs2());
337         return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
338             B*b.y+C*c.y)/(A+B+C);
339     }
340     point<T> perpercenter(){const{//垂心
341         return barycenter()*3-circumcenter()*2;
342     }
343 };
344 template<typename T>
345 struct point3D{
346     T x,y,z;
347     point3D(){
348         point3D(const T&x,const T&y,const T&z):x(x
349             ),y(y),z(z){
350         point3D operator+(const point3D &b)const{
351             return point3D(x+b.x,y+b.y,z+b.z);
352         }
353         point3D operator-(const point3D &b)const{
354             return point3D(x-b.x,y-b.y,z-b.z);
355         }
356         point3D operator*(const T &b)const{
357             return point3D(x*b,y*b,z*b);
358         }
359         point3D operator/(const T &b)const{
360             return point3D(x/b,y/b,z/b);
361         }
362         bool operator==(const point3D &b)const{
363             return x==b.x&&y==b.y&&z==b.z;
364         }
365         T dot(const point3D &b)const{
366             return x*b.x+y*b.y+z*b.z;
367         }
368         point3D cross(const point3D &b)const{
369             return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
370                 *b.y-y*b.x);
371         }
372         T abs2(){const{//向量長度的平方
373             return dot(*this);
374         }
375         T area2(const point3D &b)const{//和b、原點
376             圍成面積的平方
377             return cross(b).abs2()/4;
378         }
379 };
380 template<typename T>
381 struct line3D{
382     point3D<T> p1,p2;
383     line3D(){
384         line3D(const point3D<T> &p1,const point3D<
385             T> &p2):p1(p1),p2(p2){
386         T dis2(const point3D<T> &p,bool is_segment
387             =0)const{//點跟直線/線段的距離平方
388             point3D<T> v=p2-p1,v1=p-p1;
389             if(is_segment){
390                 point3D<T> v2=p-p2;
391                 if(v.dot(v1)<=0)return v1.abs2();
392                 if(v.dot(v2)>=0)return v2.abs2();
393             }
394             point3D<T> tmp=v.cross(v1);
395             return tmp.abs2()/v.abs2();
396         }
397     pair<point3D<T>,point3D<T> > closest_pair(
398         const line3D<T> &l)const{
399         point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
400         point3D<T> N=v1.cross(v2),ab(p1-l.p1);
401         //if(N.abs2()==0)return NULL;平行或重合
402     }
403     template<typename T>
404     struct tetrahedron{//四面體
405         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
406             最近點對距離
407         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
408             cross(d2);
409         T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2
410             ();
411         T t2=((l.p1-p1).cross(d1)).dot(D)/D.abs2
412             ();
413         return make_pair(p1+d1*t1,l.p1+d2*t2);
414     }
415     bool same_side(const point3D<T> &a,const
416         point3D<T> &b)const{
417         return (p2-p1).cross(a-p1).dot((p2-p1).
418             cross(b-p1))>0;
419     }
420 };
421 template<typename T>
422 struct plane{
423     point3D<T> p0,n;//平面上的點和法向量
424     plane(){
425         plane(const point3D<T> &p0,const point3D<T>
426             &n):p0(p0),n(n){
427         T dis2(const point3D<T> &p)const{//點到平
428             面距離的平方
429             T tmp=(p-p0).dot(n);
430             return tmp*tmp/n.abs2();
431         }
432         point3D<T> projection(const point3D<T> &p)
433             const{
434             return p-n*(p-p0).dot(n)/n.abs2();
435         }
436         point3D<T> line_intersection(const line3D<
437             T> &l)const{
438             T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
439             重合該平面
440             return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
441                 tmp);
442         }
443         line3D<T> plane_intersection(const plane &
444             pl)const{
445             point3D<T> e=n.cross(pl.n),v=n.cross(e);
446             T tmp=pl.n.dot(v);//等於0表示平行或重合
447             該平面
448             point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/
449                 tmp);
450             return line3D<T>(q,q+e);
451         }
452     };
453     template<typename T>
454     struct triangle3D{
455         point3D<T> a,b,c;
456         triangle3D(){
457             triangle3D(const point3D<T> &a,const
458                 point3D<T> &b,const point3D<T> &c):a(a
459                     ),b(b),c(c){
460             bool point_in(const point3D<T> &p)const{//
461                 點在該平面上的投影在三角形中
462             return line3D<T>(b,c).same_side(p,a)&&
463                 line3D<T>(a,c).same_side(p,b)&&
464                 line3D<T>(a,b).same_side(p,c);
465             }
466         }
467     };
468     template<typename T>
469     struct tetrahedron{//四面體
470         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
471             最近點對距離
472         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
473             cross(d2);
474         T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2
475             ();
476         T t2=((l.p1-p1).cross(d1)).dot(D)/D.abs2
477             ();
478         return make_pair(p1+d1*t1,l.p1+d2*t2);
479     }
480     bool same_side(const point3D<T> &a,const
481         point3D<T> &b)const{
482         return (p2-p1).cross(a-p1).dot((p2-p1).
483             cross(b-p1))>0;
484     }
485 };
486 template<typename T>
487 struct plane{
488     point3D<T> p0,n;//平面上的點和法向量
489     plane(){
490         plane(const point3D<T> &p0,const point3D<T>
491             &n):p0(p0),n(n){
492         T dis2(const point3D<T> &p)const{//點到平
493             面距離的平方
494             T tmp=(p-p0).dot(n);
495             return tmp*tmp/n.abs2();
496         }
497         point3D<T> projection(const point3D<T> &p)
498             const{
499             return p-n*(p-p0).dot(n)/n.abs2();
500         }
501         point3D<T> line_intersection(const line3D<
502             T> &l)const{
503             T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
504             重合該平面
505             return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
506                 tmp);
507         }
508         line3D<T> plane_intersection(const plane &
509             pl)const{
510             point3D<T> e=n.cross(pl.n),v=n.cross(e);
511             T tmp=pl.n.dot(v);//等於0表示平行或重合
512             該平面
513             point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/
514                 tmp);
515             return line3D<T>(q,q+e);
516         }
517     };
518     template<typename T>
519     struct triangle3D{
520         point3D<T> a,b,c;
521         triangle3D(){
522             triangle3D(const point3D<T> &a,const
523                 point3D<T> &b,const point3D<T> &c):a(a
524                     ),b(b),c(c){
525             bool point_in(const point3D<T> &p)const{//
526                 點在該平面上的投影在三角形中
527             return line3D<T>(b,c).same_side(p,a)&&
528                 line3D<T>(a,c).same_side(p,b)&&
529                 line3D<T>(a,b).same_side(p,c);
530             }
531         }
532     };
533     template<typename T>
534     struct tetrahedron{//四面體
535         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
536             最近點對距離
537         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
538             cross(d2);
539         T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2
540             ();
541         T t2=((l.p1-p1).cross(d1)).dot(D)/D.abs2
542             ();
543         return make_pair(p1+d1*t1,l.p1+d2*t2);
544     }
545     bool same_side(const point3D<T> &a,const
546         point3D<T> &b)const{
547         return (p2-p1).cross(a-p1).dot((p2-p1).
548             cross(b-p1))>0;
549     }
550 };
551 template<typename T>
552 struct plane{
553     point3D<T> p0,n;//平面上的點和法向量
554     plane(){
555         plane(const point3D<T> &p0,const point3D<T>
556             &n):p0(p0),n(n){
557         T dis2(const point3D<T> &p)const{//點到平
558             面距離的平方
559             T tmp=(p-p0).dot(n);
560             return tmp*tmp/n.abs2();
561         }
562         point3D<T> projection(const point3D<T> &p)
563             const{
564             return p-n*(p-p0).dot(n)/n.abs2();
565         }
566         point3D<T> line_intersection(const line3D<
567             T> &l)const{
568             T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
569             重合該平面
570             return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
571                 tmp);
572         }
573         line3D<T> plane_intersection(const plane &
574             pl)const{
575             point3D<T> e=n.cross(pl.n),v=n.cross(e);
576             T tmp=pl.n.dot(v);//等於0表示平行或重合
577             該平面
578             point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/
579                 tmp);
580             return line3D<T>(q,q+e);
581         }
582     };
583     template<typename T>
584     struct triangle3D{
585         point3D<T> a,b,c;
586         triangle3D(){
587             triangle3D(const point3D<T> &a,const
588                 point3D<T> &b,const point3D<T> &c):a(a
589                     ),b(b),c(c){
590             bool point_in(const point3D<T> &p)const{//
591                 點在該平面上的投影在三角形中
592             return line3D<T>(b,c).same_side(p,a)&&
593                 line3D<T>(a,c).same_side(p,b)&&
594                 line3D<T>(a,b).same_side(p,c);
595             }
596         }
597     };
598     template<typename T>
599     struct tetrahedron{//四面體
600         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
601             最近點對距離
602         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
603             cross(d2);
604         T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2
605             ();
606         T t2=((l.p1-p1).cross(d1)).dot(D)/D.abs2
607             ();
608         return make_pair(p1+d1*t1,l.p1+d2*t2);
609     }
610     bool same_side(const point3D<T> &a,const
611         point3D<T> &b)const{
612         return (p2-p1).cross(a-p1).dot((p2-p1).
613             cross(b-p1))>0;
614     }
615 };
616 template<typename T>
617 struct plane{
618     point3D<T> p0,n;//平面上的點和法向量
619     plane(){
620         plane(const point3D<T> &p0,const point3D<T>
621             &n):p0(p0),n(n){
622         T dis2(const point3D<T> &p)const{//點到平
623             面距離的平方
624             T tmp=(p-p0).dot(n);
625             return tmp*tmp/n.abs2();
626         }
627         point3D<T> projection(const point3D<T> &p)
628             const{
629             return p-n*(p-p0).dot(n)/n.abs2();
630         }
631         point3D<T> line_intersection(const line3D<
632             T> &l)const{
633             T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
634             重合該平面
635             return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
636                 tmp);
637         }
638         line3D<T> plane_intersection(const plane &
639             pl)const{
640             point3D<T> e=n.cross(pl.n),v=n.cross(e);
641             T tmp=pl.n.dot(v);//等於0表示平行或重合
642             該平面
643             point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/
644                 tmp);
645             return line3D<T>(q,q+e);
646         }
647     };
648     template<typename T>
649     struct triangle3D{
650         point3D<T> a,b,c;
651         triangle3D(){
652             triangle3D(const point3D<T> &a,const
653                 point3D<T> &b,const point3D<T> &c):a(a
654                     ),b(b),c(c){
655             bool point_in(const point3D<T> &p)const{//
656                 點在該平面上的投影在三角形中
657             return line3D<T>(b,c).same_side(p,a)&&
658                 line3D<T>(a,c).same_side(p,b)&&
659                 line3D<T>(a,b).same_side(p,c);
660             }
661         }
662     };
663     template<typename T>
664     struct tetrahedron{//四面體
665         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
666             最近點對距離
667         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
668             cross(d2);
669         T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2
670             ();
671         T t2=((l.p1-p1).cross(d1)).dot(D)/D.abs2
672             ();
673         return make_pair(p1+d1*t1,l.p1+d2*t2);
674     }
675     bool same_side(const point3D<T> &a,const
676         point3D<T> &b)const{
677         return (p2-p1).cross(a-p1).dot((p2-p1).
678             cross(b-p1))>0;
679     }
680 };
681 template<typename T>
682 struct plane{
683     point3D<T> p0,n;//平面上的點和法向量
684     plane(){
685         plane(const point3D<T> &p0,const point3D<T>
686             &n):p0(p0),n(n){
687         T dis2(const point3D<T> &p)const{//點到平
688             面距離的平方
689             T tmp=(p-p0).dot(n);
690             return tmp*tmp/n.abs2();
691         }
692         point3D<T> projection(const point3D<T> &p)
693             const{
694             return p-n*(p-p0).dot(n)/n.abs2();
695         }
696         point3D<T> line_intersection(const line3D<
697             T> &l)const{
698             T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
699             重合該平面
700             return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
701                 tmp);
702         }
703         line3D<T> plane_intersection(const plane &
704             pl)const{
705             point3D<T> e=n.cross(pl.n),v=n.cross(e);
706             T tmp=pl.n.dot(v);//等於0表示平行或重合
707             該平面
708             point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/
709                 tmp);
710             return line3D<T>(q,q+e);
711         }
712     };
713     template<typename T>
714     struct triangle3D{
715         point3D<T> a,b,c;
716         triangle3D(){
717             triangle3D(const point3D<T> &a,const
718                 point3D<T> &b,const point3D<T> &c):a(a
719                     ),b(b),c(c){
720             bool point_in(const point3D<T> &p)const{//
721                 點在該平面上的投影在三角形中
722             return line3D<T>(b,c).same_side(p,a)&&
723                 line3D<T>(a,c).same_side(p,b)&&
724                 line3D<T>(a,b).same_side(p,c);
725             }
726         }
727     };
728     template<typename T>
729     struct tetrahedron{//四面體
730         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
731             最近點對距離
732         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
733             cross(d2);
734         T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2
735             ();
736         T t2=((l.p1-p1).cross(d1)).dot(D)/D.abs2
737             ();
738         return make_pair(p1+d1*t1,l.p1+d2*t2);
739     }
740     bool same_side(const point3D<T> &a,const
741         point3D<T> &b)const{
742         return (p2-p1).cross(a-p1).dot((p2-p1).
743             cross(b-p1))>0;
744     }
745 };
746 template<typename T>
747 struct plane{
748     point3D<T> p0,n;//平面上的點和法向量
749     plane(){
750         plane(const point3D<T> &p0,const point3D<T>
751             &n):p0(p0),n(n){
752         T dis2(const point3D<T> &p)const{//點到平
753             面距離的平方
754             T tmp=(p-p0).dot(n);
755             return tmp*tmp/n.abs2();
756         }
757         point3D<T> projection(const point3D<T> &p)
758             const{
759             return p-n*(p-p0).dot(n)/n.abs2();
760         }
761         point3D<T> line_intersection(const line3D<
762             T> &l)const{
763             T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
764             重合該平面
765             return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
766                 tmp);
767         }
768         line3D<T> plane_intersection(const plane &
769             pl)const{
770             point3D<T> e=n.cross(pl.n),v=n.cross(e);
771             T tmp=pl.n.dot(v);//等於0表示平行或重合
772             該平面
773             point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/
774                 tmp);
775             return line3D<T>(q,q+e);
776         }
777     };
778     template<typename T>
779     struct triangle3D{
780         point3D<T> a,b,c;
781         triangle3D(){
782             triangle3D(const point3D<T> &a,const
783                 point3D<T> &b,const point3D<T> &c):a(a
784                     ),b(b),c(c){
785             bool point_in(const point3D<T> &p)const{//
786                 點在該平面上的投影在三角形中
787             return line3D<T>(b,c).same_side(p,a)&&
788                 line3D<T>(a,c).same_side(p,b)&&
789                 line3D<T>(a,b).same_side(p,c);
790             }
791         }
792     };
793     template<typename T>
794     struct tetrahedron{//四面體
795         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
796             最近點對距離
797         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
798             cross(d2);
799         T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2
800             ();
801         T t2=((l.p1-p1).cross(d1)).dot(D)/D.abs2
802             ();
803         return make_pair(p1+d1*t1,l.p1+d2*t2);
804     }
805     bool same_side(const point3D<T> &a,const
806         point3D<T> &b)const{
807         return (p2-p1).cross(a-p1).dot((p2-p1).
808             cross(b-p1))>0;
809     }
810 };
811 template<typename T>
812 struct plane{
813     point3D<T> p0,n;//平面上的點和法向量
814     plane(){
815         plane(const point3D<T> &p0,const point3D<T>
816             &n):p0(p0),n(n){
817         T dis2(const point3D<T> &p)const{//點到平
818             面距離的平方
819             T tmp=(p-p0).dot(n);
820             return tmp*tmp/n.abs2();
821         }
822         point3D<T> projection(const point3D<T> &p)
823             const{
824             return p-n*(p-p0).dot(n)/n.abs2();
825         }
826         point3D<T> line_intersection(const line3D<
827             T> &l)const{
828             T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
829             重合該平面
830             return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
831                 tmp);
832         }
833         line3D<T> plane_intersection(const plane &
834             pl)const{
835             point3D<T> e=n.cross(pl.n),v=n.cross(e);
836             T tmp=pl.n.dot(v);//等於0表示平行或重合
837             該平面
838             point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/
839                 tmp);
840             return line3D<T>(q,q+e);
841         }
842     };
843     template<typename T>
844     struct triangle3D{
845         point3D<T> a,b,c;
846         triangle3D(){
847             triangle3D(const point3D<T> &a,const
848                 point3D<T> &b,const point3D<T> &c):a(a
849                     ),b(b),c(c){
850             bool point_in(const point3D<T> &p)const{//
851                 點在該平面上的投影在三角形中
852             return line3D<T>(b,c).same_side(p,a)&&
853                 line3D<T>(a,c).same_side(p,b)&&
854                 line3D<T>(a,b).same_side(p,c);
855             }
856         }
857     };
858     template<typename T>
859     struct tetrahedron{//四面體
860         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
861             最近點對距離
862         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
863             cross(d2);
864         T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2
865             ();
866         T t2=((l.p1-p1).cross(d1)).dot(D)/D.abs2
867             ();
868         return make_pair(p1+d1*t1,l.p1+d2*t2);
869     }
870     bool same_side(const point3D<T> &a,const
871         point3D<T> &b)const{
872         return (p2-p1).cross(a-p1).dot((p2-p1).
873             cross(b-p1))>0;
874     }
875 };
876 template<typename T>
877 struct plane{
878     point3D<T> p0,n;//平面上的點和法向量
879     plane(){
880         plane(const point3D<T> &p0,const point3D<T>
881             &n):p0(p0),n(n){
882         T dis2(const point3D<T> &p)const{//點到平
883             面距離的平方
884             T tmp=(p-p0).dot(n);
885             return tmp*tmp/n.abs2();
886         }
887         point3D<T> projection(const point3D<T> &p)
888             const{
889             return p-n*(p-p0).dot(n)/n.abs2();
890         }
891         point3D<T> line_intersection(const line3D<
892             T> &l)const{
893             T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
894             重合該平面
895             return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
896                 tmp);
897         }
898         line3D<T> plane_intersection(const plane &
899             pl)const{
900             point3D<T> e=n.cross(pl.n),v=n.cross(e);
901             T tmp=pl.n.dot(v);//等於0表示平行或重合
902             該平面
903             point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/
904                 tmp);
905             return line3D<T>(q,q+e);
906         }
907     };
908     template<typename T>
909     struct triangle3D{
910         point3D<T> a,b,c;
911         triangle3D(){
912             triangle3D(const point3D<T> &a,const
913                 point3D<T> &b,const point3D<T> &c):a(a
914                     ),b(b),c(c){
915             bool point_in(const point3D<T> &p)const{//
916                 點在該平面上的投影在三角形中
917             return line3D<T>(b,c).same_side(p,a)&&
918                 line3D<T>(a,c).same_side(p,b)&&
919                 line3D<T>(a,b).same_side(p,c);
920             }
921         }
922     };
923     template<typename T>
924     struct tetrahedron{//四面體
925         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
926             最近點對距離
927         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
928             cross(d2);
929         T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2
930             ();
931         T t2=((l.p1-p1).cross(d1)).dot(D)/D.abs2
932             ();
933         return make_pair(p1+d1*t1,l.p1+d2*t2);
934     }
935     bool same_side(const point3D<T> &a,const
936         point3D<T> &b)const{
937         return (p2-p1).cross(a-p1).dot((p2-p1).
938             cross(b-p1))>0;
939     }
940 };
941 template<typename T>
942 struct plane{
943     point3D<T> p0,n;//平面上的點和法向量
944     plane(){
945         plane(const point3D<T> &p0,const point3D<T>
946             &n):p0(p0),n(n){
947         T dis2(const point3D<T> &p)const{//點到平
948             面距離的平方
949             T tmp=(p-p0).dot(n);
950             return tmp*tmp/n.abs2();
951         }
952         point3D<T> projection(const point3D<T> &p)
953             const{
954             return p-n*(p-p0).dot(n)/n.abs2();
955         }
956         point3D<T> line_intersection(const line3D<
957             T> &l)const{
958             T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
959             重合該平面
960             return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
961                 tmp);
962         }
963         line3D<T> plane_intersection(const plane &
964             pl)const{
965             point3D<T> e=n.cross(pl.n),v=n.cross(e);
966             T tmp=pl.n.dot(v);//等於0表示平行或重合
967             該平面
968             point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/
969                 tmp);
970             return line3D<T>(q,q+e);
971         }
972     };
973     template<typename T>
974     struct triangle3D{
975         point3D<T> a,b,c;
976         triangle3D(){
977             triangle3D(const point3D<T> &a,const
978                 point3D<T> &b,const point3D<T> &c):a(a
979                     ),b(b),c(c){
980             bool point_in(const point3D<T> &p)const{//
981                 點在該平面上的投影在三角形中
982             return line3D<T>(b,c).same_side(p,a)&&
983                 line3D<T>(a,c).same_side(p,b)&&
984                 line3D<T>(a,b).same_side(p,c);
985             }
986         }
987     };
988     template<typename T>
989     struct tetrahedron{//四面體
990         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
991             最近點對距離
992         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
993             cross(d2);
994         T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2
995             ();
996         T t2=((l.p1-p1).cross(d1)).dot(D)/D.abs2
997             ();
998         return make_pair(p1+d1*t1,l.p1+d2*t2);
999     }
1000     bool same_side(const point3D<T> &a,const
1001         point3D<T> &b)const{
1002         return (p2-p1).cross(a-p1).dot((p2-p1).
1003             cross(b-p1))>0;
1004     }
1005 };
1006 template<typename T
```

```

404 point3D<T> a,b,c,d;
405 tetrahedron(){}
406 tetrahedron(const point3D<T> &a,const
    point3D<T> &b,const point3D<T> &c,
    const point3D<T> &d):a(a),b(b),c(c),d(
    d){}
407 T volume6()const{//體積的六倍
408     return (d-a).dot((b-a).cross(c-a));
409 }
410 point3D<T> centroid()const{
411     return (a+b+c+d)/4;
412 }
413 bool point_in(const point3D<T> &p)const{
414     return triangle3D<T>(a,b,c).point_in(p)
        &&triangle3D<T>(c,d,a).point_in(p);
415 }
416 };
417 template<typename T>
418 struct convexhull3D{
419     static const int MAXN=105;
420     struct face{
421         int a,b,c;
422         bool use;
423         face(){}
424         face(int a,int b,int c):a(a),b(b),c(c),
            use(1){}
425 };
426 vector<point3D<T> > pt;
427 vector<face> fc;
428 int fid[MAXN][MAXN];
429 static bool point_cmp(const point3D<T> &a,
    const point3D<T> &b){
430     return a.x<b.x|| (a.x==b.x&&(a.y<b.y|| (a.
        y==b.y&&a.z<b.z)));
431 }
432 bool outside(int p,int a,int b,int c)const
    {
433     return tetrahedron<T>(pt[a],pt[b],pt[c],
        pt[p]).volume6()<0;
434 }
435 bool outside(int p,int f)const{return
    outside(p,fc[f].a,fc[f].b,fc[f].c);}
436 void AddFace(int a,int b,int c,int p){
437     if(outside(p,a,b,c))fid[c][b]=fid[b][a]=
        fid[a][c]=fc.size(),fc.push_back(
        face(c,b,a));
438     else fid[a][b]=fid[b][c]=fid[c][a]=fc.
        size(),fc.push_back(face(a,b,c));
439 }
440 bool dfs(int p,int f){
441     if(!fc[f].use)return true;
442     if(outside(p,f)){
443         int a=fc[f].a,b=fc[f].b,c=fc[f].c;
444         fc[f].use=false;
445         if(!dfs(p,fid[b][a]))AddFace(p,a,b,c);
446         if(!dfs(p,fid[c][b]))AddFace(p,b,c,a);
447         if(!dfs(p,fid[a][c]))AddFace(p,c,a,b);
448         return true;
449     }else return false;
450 }
451 void build(){
452     bool ok=false;
453     fc.clear();
454     sort(pt.begin(),pt.end(),point_cmp);

```

```

455 pt.resize(unique(pt.begin(),pt.end())-pt
    .begin());
456 for(size_t i=2;i<pt.size();++i){
457     if((pt[0]-pt[i]).area2(pt[1]-pt[i])
        !=0){
458         ok=true;
459         swap(pt[i],pt[2]);
460         break;
461     }
462 }
463 if(!ok)return;
464 ok=false;
465 for(size_t i=3;i<pt.size();++i){
466     if(tetrahedron<T>(pt[0],pt[1],pt[2],pt
        [i]).volume6()!=0){
467         ok=true;
468         swap(pt[i],pt[3]);
469         break;
470     }
471 }
472 if(!ok)return;
473 for(int i=0;i<4;++i)AddFace(i,(i+1)%4,(i
    +2)%4,(i+3)%4);
474 for(size_t i=4;i<pt.size();++i){
475     for(int j=fc.size()-1;j>=0;--j){
476         if(outside(i,j)){
477             dfs(i,j);
478             break;
479         }
480     }
481 }
482 size_t sz=0;
483 for(size_t i=0;i<fc.size();++i)if(fc[i].
    use)fc[sz++]=fc[i];
484 fc.resize(sz);
485 }
486 point3D<T> centroid()const{
487     point3D<T> res(0,0,0);
488     T vol=0;
489     for(size_t i=0;i<fc.size();++i){
490         T tmp=pt[fc[i].a].dot(pt[fc[i].b].
            cross(pt[fc[i].c]));
491         res=res+(pt[fc[i].a]+pt[fc[i].b]+pt[fc
            [i].c])*tmp;
492         vol+=tmp;
493     }
494     return res/(vol*4);
495 }
496 };

```

1.2 SmallestCircle.cpp

```

1 #include "Geometry.cpp"
2 struct Circle{
3     typedef point<double> p;
4     typedef const point<double> cp;
5     p x;
6     double r2;
7     bool incircle(cp &c)const{return (x-c).
        abs2()<=r2;}
8 };
9

```

```

10 Circle TwoPointCircle(Circle::cp &a, Circle
    ::cp &b) {
11     Circle::p m=(a+b)/2;
12     return (Circle){m,(a-m).abs2()};
13 }
14
15 Circle outcircle(Circle::p a, Circle::p b,
    Circle::p c) {
16     if(TwoPointCircle(a,b).incircle(c))
        return TwoPointCircle(a,b);
17     if(TwoPointCircle(b,c).incircle(a))
        return TwoPointCircle(b,c);
18     if(TwoPointCircle(c,a).incircle(b))
        return TwoPointCircle(c,a);
19     Circle::p ret;
20     double a1=b.x-a.x, b1=b.y-a.y, c1=(a1*a1
        +b1*b1)/2;
21     double a2=c.x-a.x, b2=c.y-a.y, c2=(a2*a2
        +b2*b2)/2;
22     double d = a1*b2 - a2*b1;
23     ret.x=a.x+(c1*b2-c2*b1)/d;
24     ret.y=a.y+(a1*c2-a2*c1)/d;
25     return (Circle){ret,(ret-a).abs2()};
26 }
27 //rand required
28 Circle SmallestCircle(std::vector<Circle::p>
    &p){
29     int n=p.size();
30     if(n==1) return (Circle){p[0],0.0};
31     if(n==2) return TwoPointCircle(p[0],p
        [1]);
32     random_shuffle(p.begin(),p.end());
33     Circle c = {p[0],0.0};
34     for(int i=0;i<n;++i){
35         if(c.incircle(p[i])) continue;
36         c=Circle{p[i],0.0};
37         for(int j=0;j<i;++j){
38             if(c.incircle(p[j])) continue;
39             c=TwoPointCircle(p[i],p[j]);
40             for(int k=0;k<j;++k){
41                 if(c.incircle(p[k]))
                    continue;
42                 c=outcircle(p[i],p[j],p[k]);
43             }
44         }
45     }
46     return c;
47 }

```

1.3 最近點對.cpp

```

1 #define INF LLONG_MAX/*預設是Long Long最大值
    */
2 template<typename T>
3 T closest_pair(vector<point<T> >&v,vector<
    point<T> >&t,int l,int r){
4     T dis=INF,tmd;
5     if(l==r)return dis;
6     int mid=(l+r)/2;
7     if((tmd=closest_pair(v,t,l,mid))<dis)dis=
        tmd;
8     if((tmd=closest_pair(v,t,mid+1,r))<dis)dis
        =tmd;

```

```

9     t.clear();
10     for(int i=l;i<=r;++i)
11         if((v[i].x-v[mid].x)*(v[i].x-v[mid].x)<
            dis)t.push_back(v[i]);
12     sort(t.begin(),t.end(),point<T>::y_cmp);/*
        如果用merge_sort的方式可以O(n)*/
13     for(int i=0;i<(int)t.size();++i)
14         for(int j=1;j<=3&&i+j<(int)t.size();++j)
15             if((tmd=(t[i]-t[i+j]).abs2())<dis)dis=
                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
        近點對距離*/
23 }

```

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;}
18 };

```

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[MAXN],col[MAXND];//每個節點代表的
        列行
8     int L[MAXN],R[MAXN],U[MAXN],D[MAXND];
9     vector<int> ans,ansst;
10    void init(int _n,int _m){
11        n=_n,m=_m;

```

```

12 for(int i=0;i<=m;++i){
13     U[i]=D[i]=i,L[i]=i-1,R[i]=i+1;
14     S[i]=0;
15 }
16 R[m]=0,L[0]=m;
17 sz=m,ansd=INT_MAX; //ansd存最優解的個數
18 for(int i=1;i<=n;++i)H[i]=-1;
19 }
20 void add(int r,int c){
21     ++S[col[+sz]=c];
22     row[sz]=r;
23     D[sz]=D[c],U[D[c]]=sz,U[sz]=c,D[c]=sz;
24     if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
25     else R[sz]=R[H[r]],L[R[H[r]]]=sz,L[sz]=H
        [r],R[H[r]]=sz;
26 }
27 #define DFOR(i,A,s) for(int i=A[s];i!=s;i=
        A[i])
28 void remove(int c){ //刪除第c行和所有當前覆
        蓋到第c行的列
29     L[R[c]]=L[c],R[L[c]]=R[c]; //這裡刪除第c
        行，若有些行不需要處理可以在開始時呼
        叫他
30     DFOR(i,D,c)DFOR(j,R,i){U[D[j]]=U[j],D[U[
        j]]=D[j],--S[col[j]]};
31 }
32 void restore(int c){ //恢復第c行和所有當前
        覆蓋到第c行的列，remove的逆操作
33     DFOR(i,U,c)DFOR(j,L,i){++S[col[j]],U[D[j]
        ]=j,D[U[j]]=j};
34     L[R[c]]=c,R[L[c]]=c;
35 }
36 void remove2(int nd){ //刪除nd所在的行當前
        所有點(包括虛擬節點)，只保留nd
37     DFOR(i,D,nd)L[R[i]]=L[i],R[L[i]]=R[i];
38 }
39 void restore2(int nd){ //刪除nd所在的行當前
        所有點，為remove2的逆操作
40     DFOR(i,U,nd)L[R[i]]=R[L[i]]=i;
41 }
42 bool vis[MAXM];
43 int h(){ //估價函數 for IDA*
44     int res=0;
45     memset(vis,0,sizeof(vis));
46     DFOR(i,R,0)if(!vis[i]){
47         vis[i]=1;
48         ++res;
49         DFOR(j,D,i)DFOR(k,R,j)vis[col[k]]=1;
50     }
51     return res;
52 }
53 bool dfs(int d){ //for精確覆蓋問題
54     if(d+h()==ansd)return 0; //找最佳解用，找
        任意解可以刪掉
55     if(!R[0]){ansd=d;return 1;}
56     int c=R[0];
57     DFOR(i,R,0)if(S[i]<S[c])c=i;
58     remove(c);
59     DFOR(i,D,c){
60         ans.push_back(row[i]);
61         DFOR(j,R,i)remove(col[j]);
62         if(dfs(d+1))return 1;
63         ans.pop_back();

```

```

64     DFOR(j,L,i)restore(col[j]);
65 }
66 restore(c);
67 return 0;
68 }
69 void dfs2(int d){ //for最小重複覆蓋問題
70     if(d+h()==ansd)return;
71     if(!R[0]){ansd=d;ans=ans;return;}
72     int c=R[0];
73     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
86     return dfs(0);
87 }
88 void min_cover(){ //解最小重複覆蓋問題
89     anst.clear(); //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> //有kd個維度
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; //記得要給INF，表示極大值
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         node *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 o;
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     }
121     T heuristic(const T h[])const{
122         T ret=0;
123         for(size_t i=0;i<kd;++i)ret+=h[i];
124         return ret;
125     }
126     int qM;
127     std::priority_queue<std::pair<T,point >
        >pQ;
128     void nearest(node *u,int k,const point &
        x,T *h,T &mndist){
129         if(u==0||heuristic(h)>=mndist)return;
130         T dist=u->pid.dist(x),old=h[k];
131         /*mndist=std::min(mndist,dist);*/
132         if(dist<mndist){
133             pQ.push(std::make_pair(dist,u->pid))
134             ;
135             if((int)pQ.size()==qM+1)
136                 mndist=pQ.top().first,pQ.pop();
137         }
138         if(x.d[k]<u->pid.d[k]){
139             nearest(u->l,(k+1)%kd,x,h,mndist);
140             h[k]=std::abs(x.d[k]-u->pid.d[k]);
141             nearest(u->r,(k+1)%kd,x,h,mndist);
142         }else{
143             nearest(u->r,(k+1)%kd,x,h,mndist);
144             h[k]=std::abs(x.d[k]-u->pid.d[k]);
145             nearest(u->l,(k+1)%kd,x,h,mndist);
146         }

```


2.3 kd_tree_replace_segment

```

144 }
145 h[k]=old;
146 }
147 std::vector<point> in_range;
148 void range(node *u, int k, const point &mi,
149           const point &ma){
150     if(!u) return;
151     bool is=1;
152     for(int i=0; i<kd; ++i)
153         if(u->pid.d[i]<mi.d[i] || ma.d[i]<u->
154            pid.d[i]){
155             is=0; break;
156         }
157     if(is) in_range.push_back(u->pid);
158     if(mi.d[k]<u->pid.d[k] || range(u->l, (k
159                                     +1)%kd, mi, ma);
160     if(ma.d[k]>u->pid.d[k] || range(u->r, (k
161                                     +1)%kd, mi, ma);
162 }
163 public:
164 kd_tree(const T &INF, double a=0.75): root(
165     (0), alpha(a), loga(log2(1.0/a)), INF(
166     INF), maxn(1)){
167     ~kd_tree(){ delete root; }
168     void clear(){ delete root; root=0; maxn=1; }
169     void build(int n, const point *p){
170         delete root; A.resize(maxn=n);
171         for(int i=0; i<n; ++i) A[i]=new node(p[i
172         ]);
173         root=build(0, 0, n-1);
174     }
175     void insert(const point &x){
176         insert(root, 0, x, __lg(size(root))/loga)
177         ;
178         if(root->s>maxn) maxn=root->s;
179     }
180     bool erase(const point &p){
181         bool d=erase(root, 0, p);
182         if(root&&root->s<alpha*maxn) rebuild();
183         return d;
184     }
185     void rebuild(){
186         if(root) rebuild(root, 0);
187         maxn=root->s;
188     }
189     T nearest(const point &x, int k){
190         qM=k;
191         T mndist=INF, h[kd]={};
192         nearest(root, 0, x, h, mndist);
193         mndist=pQ.top().first;
194         pQ=std::priority_queue<std::pair<T,
195         point >>(){};
196         return mndist; //回傳離x第k近的點的距離
197     }
198     const std::vector<point> &range(const
199         point &mi, const point &ma){
200         in_range.clear();
201         range(root, 0, mi, ma);
202         return in_range; //回傳介於mi到ma之間的
203         點vector
204     }
205     int size(){ return root?root->s:0; }
206 };

```

/*kd樹代替高維線段樹*/

```

256 struct node{
257     node *l, *r;
258     point pid, mi, ma;
259     int s;
260     int data;
261     node(const point &p, int d): l(0), r(0), pid(p
262         ), mi(p), ma(p), s(1), data(d), dmin(d),
263         dmax(d){}
264     void up(){
265         mi=ma=pid;
266         s=1;
267         if(l){
268             for(int i=0; i<kd; ++i){
269                 mi.d[i]=min(mi.d[i], l->mi.d[i]);
270                 ma.d[i]=max(ma.d[i], l->ma.d[i]);
271             }
272             s+=l->s;
273         }
274         if(r){
275             for(int i=0; i<kd; ++i){
276                 mi.d[i]=min(mi.d[i], r->mi.d[i]);
277                 ma.d[i]=max(ma.d[i], r->ma.d[i]);
278             }
279             s+=r->s;
280         }
281     }
282     void up2(){
283         //其他懶惰標記向上更新
284     }
285     void down(){
286         //其他懶惰標記下推
287     }
288 } *root;
289 /*檢查區間包含用的函數*/
290 inline bool range_include(node *o, const
291     point &L, const point &R){
292     for(int i=0; i<kd; ++i){
293         if(L.d[i]>o->ma.d[i] || R.d[i]<o->mi.d[i])
294             return 0;
295     }
296     //只要(L,R)區間有和o的區間有交集就回傳
297     true
298     return 1;
299 }
300 inline bool range_in_range(node *o, const
301     point &L, const point &R){
302     for(int i=0; i<kd; ++i){
303         if(L.d[i]>o->mi.d[i] || o->ma.d[i]>R.d[i])
304             return 0;
305     }
306     //如果(L,R)區間完全包含o的區間就回傳true
307     return 1;
308 }
309 inline bool point_in_range(node *o, const
310     point &L, const point &R){
311     for(int i=0; i<kd; ++i){
312         if(L.d[i]>o->pid.d[i] || R.d[i]<o->pid.d[i]
313             ) return 0;
314     }
315     //如果(L,R)區間完全包含o->pid這個點就回傳
316     true
317     return 1;
318 }
319 }

```

2.3 kd_tree_replace_segment

```

534 /*單點修改 · 以單點改值為例*/
535 void update(node *u, const point &x, int data,
536             int k=0){
537     if(!u) return;
538     u->down();
539     if(u->pid==x){
540         u->data=data;
541         u->up2();
542         return;
543     }
544     cmp.sort_id=k;
545     update(cmp(x, u->pid)?u->l:u->r, x, data, (k
546             +1)%kd);
547     u->up2();
548 }
549 /*區間修改*/
550 void update(node *o, const point &L, const
551     point &R, int data){
552     if(!o) return;
553     o->down();
554     if(range_in_range(o, L, R)){
555         //區間懶惰標記修改
556         o->down();
557         return;
558     }
559     if(point_in_range(o, L, R)){
560         //這個點在(L,R)區間 · 但是他的左右子樹不
561         一定在區間中
562         //單點懶惰標記修改
563         if(o->l&&range_include(o->l, L, R)) update(o
564             ->l, L, R, data);
565         if(o->r&&range_include(o->r, L, R)) update(o
566             ->r, L, R, data);
567         o->up2();
568     }
569     //區間查詢 · 以總和為例*/
570 int query(node *o, const point &L, const point
571     &R){
572     if(!o) return 0;
573     o->down();
574     if(range_in_range(o, L, R)) return o->sum;
575     int ans=0;
576     if(point_in_range(o, L, R)) ans+=o->data;
577     if(o->l&&range_include(o->l, L, R)) ans+=
578         query(o->l, L, R);
579     if(o->r&&range_include(o->r, L, R)) ans+=
580         query(o->r, L, R);
581     return ans;
582 }

```

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

```

```

67 lsh.resize(unique(lsh.begin(),lsh.end())
68             -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 }
76 while(m--){
77     int a,b,k;
78     scanf("%d%d%d",&a,&b,&k);
79     int res=find(0,N,root[a-1],root[b],k);
80     printf("%d\n",lsh[res]);
81 }
82 return 0;

```

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 ref_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     ref_pointer&operator=(const ref_pointer &t)
16     ){
17         if(p&&--(*p).ref==0)delete p;
18         p=t.p;
19         p&&+(*p).ref;
20         return*this;
21     }
22     ref_pointer(_RefCounter<T> *t=0):p(t){
23         p&&+(*p).ref;
24     }
25     ref_pointer(const ref_pointer &t):p(t.p){
26         p&&+(*p).ref;
27     }
28     ~ref_pointer(){
29         if(p&&--(*p).ref==0)delete p;
30     }
31 };
32 template<typename T>
33 inline const ref_pointer<T> new_ref(const T&
34     nd){
35     return ref_pointer<T>(new _RefCounter<T>(
36         nd));
37 }
38 struct P{
39     int a,b;
40     P(int A,int B):a(A),b(B){}
41 }p(2,3);
42 int main(){

```

```

40 ref_pointer<int>b=new_ref(int(5));
41 ref_pointer<int>a=new_ref(*b);
42 ref_pointer<P>c=new_ref(p);
43 return 0;
44 }

```

2.6 skew_heap.cpp

```

1 node *merge(node *a,node *b){
2     if(!a||!b)return a?a:b;
3     if(b->data<a->data)swap(a,b);
4     swap(a->l,a->r);
5     a->l=merge(b,a->l);
6     return a;
7 }

```

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(){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[1]),delete o;
59    }
60 public:
61     treap(unsigned s=20150119):nil(new node)
62         ,root(nil),x(s){}
63     ~treap(){clear(root),delete nil;}
64     void clear(){clear(root),root=nil;}
65     void insert(const T &data){
66         insert(root,data);

```

```

67     bool erase(const T &data){
68         return erase(root,data);
69     }
70     bool find(const T&data){
71         for(node *o=root;o->s;){
72             if(o->data==data)return 1;
73             if(o->ch[o->data<data]);
74             return 0;
75         }
76     }
77     int rank(const T&data){
78         int cnt=0;
79         for(node *o=root;o->s;){
80             if(o->data<data)cnt+=o->ch[0]->s+1,o=o->ch[1];
81             else o=o->ch[0];
82             return cnt;
83         }
84     }
85     const T&kth(int k){
86         for(node *o=root;;){
87             if(k<=o->ch[0]->s)o=o->ch[0];
88             else if(k==o->ch[0]->s+1)return o->data;
89             else k-=o->ch[0]->s+1,o=o->ch[1];
90         }
91     }
92     const T&operator[](int k){
93         return kth(k);
94     }
95     const T&preorder(const T&data){
96         node *x=root,*y=0;
97         while(x->s){
98             if(x->data<data)y=x,x=x->ch[1];
99             else x=x->ch[0];
100             if(y)return y->data;
101             return data;
102         }
103     }
104     const T&successor(const T&data){
105         node *x=root,*y=0;
106         while(x->s){
107             if(data<x->data)y=x,x=x->ch[0];
108             else x=x->ch[1];
109             if(y)return y->data;
110             return data;
111         }
112     }
113     int size(){return root->s;}
114 };

```

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 #ifdef DEBUG
2 #define dbg(...) {\
3     fprintf(stderr,"%s - %d : (%s) = ",\
4         __PRETTY_FUNCTION__, __LINE__,#\
5         __VA_ARGS__); \
6     _DO(__VA_ARGS__); \
7 }
8 template<typename I> void _DO(I&x){cerr<<x<<endl;}
9 template<typename I,typename...T> void _DO(I&x,T&&...tail){cerr<<x<<" ";_DO(tail...);}
10 #else
11 #define dbg(...)
12 #endif

```

3.2 ext.cpp

```

1 #include<bits/extc++.h>
2 #include<ext/pd_ds/assoc_container.hpp>
3 #include<ext/pd_ds/tree_policy.hpp>
4 using namespace __gnu_cxx;
5 using namespace __gnu_pbds;
6 template<typename T>
7 using pbds_set = tree<T,null_type,less<T>,rb_tree_tag,tree_order_statistics_node_update>;
8 template<typename T,typename U>
9 using pbds_map = tree<T,U,less<T>,rb_tree_tag,tree_order_statistics_node_update>;
10 using heap = __gnu_pbds::priority_queue<int>;
11 //s.find_by_order(1);//0 base
12 //s.order_of_key(1);

```

3.3 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 }

```

3.4 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 // #!/bin/bash
8 // g++ -std=c++11 -O2 -Wall -Wextra -Wno-unused-result -DDEBUG $1 && ./a.out
9 // -fsanitize=address -fsanitize=undefined -fsanitize=return

```

4 Flow

4.1 dinic.cpp

```

1 template<typename T>
2 struct DINIC{
3     static const int MAXN=105;
4     static const T INF=INT_MAX;
5     int n;//點數
6     int level[MAXN],cur[MAXN];
7     struct edge{
8         int v,pre;
9         T cap,flow,r;
10     }edge(int v,int pre,T cap):v(v),pre(pre),cap(cap),flow(0),r(cap){}
11 };
12 int g[MAXN];
13 vector<edge> e;
14 void init(int _n){
15     memset(g,-1,sizeof(int)*((n=_n)+1));
16     e.clear();
17 }
18 void add_edge(int u,int v,T cap,bool directed=false){

```

```

19     e.push_back(edge(v,g[u],cap));
20     g[u]=e.size()-1;
21     e.push_back(edge(u,g[v],directed?0:cap));
22     ;
23     g[v]=e.size()-1;
24 }
25 int bfs(int s,int t){
26     memset(level,0,sizeof(int)*(n+1));
27     memcpy(cur,g,sizeof(int)*(n+1));
28     queue<int> q;
29     q.push(s);
30     level[s]=1;
31     while(q.size()){
32         int u=q.front();q.pop();
33         for(int i=g[u];~i;i=e[i].pre){
34             if(!level[e[i].v]&&e[i].r){
35                 level[e[i].v]=level[u]+1;
36                 q.push(e[i].v);
37                 if(e[i].v==t)return 1;
38             }
39         }
40     }
41     return 0;
42 }
43 T dfs(int u,int t,T cur_flow=INF){
44     if(u==t)return cur_flow;
45     T df;
46     for(int &i=cur[u];~i;i=e[i].pre){
47         if(level[e[i].v]==level[u]+1&&e[i].r){
48             if(df=dfs(e[i].v,t,min(cur_flow,e[i].r))){
49                 e[i].flow+=df;
50                 e[i^1].flow-=df;
51                 e[i].r-=df;
52                 e[i^1].r+=df;
53                 return df;
54             }
55         }
56     }
57     return level[u]=0;
58 }
59 T dinic(int s,int t,bool clean=true){
60     if(clean){
61         for(size_t i=0;i<e.size();++i){
62             e[i].flow=0;
63             e[i].r=e[i].cap;
64         }
65     }
66     T ans=0,mf=0;
67     while(bfs(s,t))while(mf=dfs(s,t))ans+=mf;
68     return ans;
69 };

```

4.2 ISAP_with_cut.cpp

```

1 template<typename T>
2 struct ISAP{
3     static const int MAXN=105;
4     static const T INF=INT_MAX;
5     int n;//點數
6     int d[MAXN],gap[MAXN],cur[MAXN];

```

```

7 struct edge{
8     int v,pre;
9     T cap,flow,r;
10     edge(int v,int pre,T cap):v(v),pre(pre),cap(cap),flow(0),r(cap){}
11 };
12 int g[MAXN];
13 vector<edge> e;
14 void init(int _n){
15     memset(g,-1,sizeof(int)*((n=_n)+1));
16     e.clear();
17 }
18 void add_edge(int u,int v,T cap,bool directed=false){
19     e.push_back(edge(v,g[u],cap));
20     g[u]=e.size()-1;
21     e.push_back(edge(u,g[v],directed?0:cap));
22     ;
23     g[v]=e.size()-1;
24 }
25 T dfs(int u,int s,int t,T cur_flow=INF){
26     if(u==t)return cur_flow;
27     T tf=cur_flow,df;
28     for(int &i=cur[u];~i;i=e[i].pre){
29         if(e[i].r&&d[u]==d[e[i].v]+1){
30             df=dfs(e[i].v,s,t,min(tf,e[i].r));
31             e[i].flow+=df;
32             e[i^1].flow-=df;
33             e[i].r-=df;
34             e[i^1].r+=df;
35             if(!((tf-=df)||d[s]==n))return cur_flow-tf;
36         }
37     }
38     int mh=n;
39     for(int i=cur[u]=g[u];~i;i=e[i].pre){
40         if(e[i].r&&d[e[i].v]<mh)mh=d[e[i].v];
41     }
42     if(!--gap[d[u]])d[s]=n;
43     else ++gap[d[u]]=++mh;
44     return cur_flow-tf;
45 }
46 T isap(int s,int t,bool clean=true){
47     memset(d,0,sizeof(int)*(n+1));
48     memset(gap,0,sizeof(int)*(n+1));
49     memcpy(cur,g,sizeof(int)*(n+1));
50     if(clean){
51         for(size_t i=0;i<e.size();++i){
52             e[i].flow=0;
53             e[i].r=e[i].cap;
54         }
55     }
56     T max_flow=0;
57     for(gap[0]=n;d[s]<n;)max_flow+=dfs(s,s,t);
58     return max_flow;
59 }
60 vector<int> cut_e;//最小割邊集
61 bool vis[MAXN];
62 void dfs_cut(int u){
63     vis[u]=1;//表示u屬於source的最小割集
64     for(int i=g[u];~i;i=e[i].pre){
65         if(e[i].flow<e[i].cap&&vis[e[i].v])dfs_cut(e[i].v);
66     }
67 }

```

```

66 }
67 T min_cut(int s,int t){
68     T ans=isap(s,t);
69     memset(vis,0,sizeof(bool)*(n+1));
70     dfs_cut(s),cut_e.clear();
71     for(int u=0;u<=n;++u){
72         if(vis[u])for(int i=g[u];~i;i=e[i].pre)
73             if(!vis[e[i].v])cut_e.push_back(i);
74     }
75     return ans;
76 }
77 }
78 };

```

4.3 MinCostMaxFlow.cpp

```

1 template<typename _T>
2 struct MCMF{
3     static const int MAXN=440;
4     static const _T INF=999999999;
5     struct edge{
6         int v,pre;
7         _T cap,cost;
8         edge(int v,int pre,_T cap,_T cost):v(v),
9             pre(pre),cap(cap),cost(cost){}
10    };
11    int n,S,T;
12    _T dis[MAXN],piS,ans;
13    bool vis[MAXN];
14    vector<edge> e;
15    int g[MAXN];
16    void init(int _n){
17        memset(g,-1,sizeof(int)*((n=_n)+1));
18        e.clear();
19    }
20    void add_edge(int u,int v,_T cap,_T cost,
21        bool directed=false){
22        e.push_back(edge(v,g[u],cap,cost));
23        g[u]=e.size()-1;
24        e.push_back(edge(u,g[v],directed?0:cap,-
25            cost));
26        g[v]=e.size()-1;
27    }
28    _T augment(int u,_T cur_flow){
29        if(u==T||!cur_flow)return ans+=piS*
30            cur_flow,cur_flow;
31        vis[u]=1;
32        _T r=cur_flow,d;
33        for(int i=g[u];~i;i=e[i].pre){
34            if(e[i].cap&&!e[i].cost&&!vis[e[i].v])
35                {
36                    d=augment(e[i].v,min(r,e[i].cap));
37                    e[i].cap-=d;
38                    e[i^1].cap+=d;
39                    if(!(r-=d))break;
40                }
41        }
42        return cur_flow-r;
43    }
44    bool modlabel(){
45        for(int u=0;u<=n;++u)dis[u]=INF;
46        static deque<int>q;

```

```

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

```

5 Graph

5.1 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];//屬於n2的點匹配了哪個點
5 vector<int> g[MAXN1];//圖
6 bool vis[MAXN2];//是否走訪過
7 bool dfs(int u){
8     for(size_t i=0;i<g[u].size();++i){
9         int v=g[u][i];
10        if(vis[v])continue;
11        vis[v]=1;
12        if(match[v]==-1||dfs(match[v])){
13            match[v]=u;
14            return 1;
15        }
16    }
17    return 0;
18 }
19 inline int max_match(){
20     int ans=0;
21     memset(match,-1,sizeof(int)*n2);
22     for(int i=0;i<n1;++i){
23         memset(vis,0,sizeof(bool)*n2);
24         if(dfs(i))++ans;
25     }

```

```

26     return ans;
27 }

```

5.2 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
4     匹配很多邊
5 vector<int> g[MAXN1];//圖
6 int c[MAXN2];//每個屬於n2點最多可以接受幾條
7     匹配邊
8 vector<int> match_list[MAXN2];//每個屬於n2的
9     點匹配了那些點
10 bool vis[MAXN2];//是否走訪過
11 bool dfs(int u){
12     for(size_t i=0;i<g[u].size();++i){
13         int v=g[u][i];
14         if(vis[v])continue;
15         vis[v]=true;
16         if((int)match_list[v].size()<c[v]){
17             match_list[v].push_back(u);
18             return true;
19         }else{
20             for(size_t j=0;j<match_list[v].size()
21                 ;++j){
22                 int next_u=match_list[v][j];
23                 if(dfs(next_u)){
24                     match_list[v][j]=u;
25                     return true;
26                 }
27             }
28         }
29     }
30     return false;
31 }
32 inline int max_match(){
33     for(int i=0;i<n2;++i)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;
41 }

```

5.3 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;

```

```

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

```

5.4 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;++i){
9         f[0][i]=1;
10        g[i].clear();
11        rg[i].clear();
12    }
13 }
14 inline void add_edge(int u,int v){
15     g[u].push_back(v);
16     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(
32         point_hash(i));//O(N^2)
33     sort(ans.begin(),ans.end());
34     return ans;

```

5.5 KM.cpp

```

1 #define MAXN 405
2 #define INF 0x3f3f3f3f
3 int n;// 1-base，0表示沒有匹配
4 int g[MAXN][MAXN],lx[MAXN],ly[MAXN],pa[MAXN],slack_y[MAXN];
5 int match_y[MAXN],match_x[MAXN];
6 bool vx[MAXN],vy[MAXN];
7 void augment(int y){
8     for(int x,z;y;y=z){
9         x=pa[y],z=match_x[x];
10        match_y[y]=x,match_x[x]=y;
11    }
12 }
13 void bfs(int st){
14     for(int i=1;i<=n;++i)slack_y[i]=INF,vx[i]=vy[i]=0;
15     queue<int> q;q.push(st);
16     for(;;){
17         while(q.size()){
18             int x=q.front();q.pop();
19             vx[x]=1;
20             for(int y=1;y<=n;++y)if(!vy[y]){
21                 int t=lx[x]+ly[y]-g[x][y];
22                 if(t==0){
23                     pa[y]=x;
24                     if(!match_y[y]){augment(y);return;}
25                     vy[y]=1,q.push(match_y[y]);
26                 }else if(slack_y[y]>t)pa[y]=x,slack_y[y]=t;
27             }
28         }
29         int cut=INF;
30         for(int y=1;y<=n;++y){

```

```

31             if(!vy[y]&&cut>slack_y[y])cut=slack_y[y];
32         }
33         for(int j=1;j<=n;++j){
34             if(vx[j])lx[j]-=cut;
35             if(vy[j])ly[j]+=cut;
36             else slack_y[j]-=cut;
37         }
38         for(int y=1;y<=n;++y){
39             if(!vy[y]&&slack_y[y]==0){
40                 if(!match_y[y]){augment(y);return;}
41                 vy[y]=1,q.push(match_y[y]);
42             }
43         }
44     }
45 }
46 long long KM(){
47     memset(match_y,0,sizeof(int)*(n+1));
48     memset(ly,0,sizeof(int)*(n+1));
49     for(int x=1;x<=n;++x){
50         lx[x]=-INF;
51         for(int y=1;y<=n;++y)
52             lx[x]=max(lx[x],g[x][y]);
53     }
54     for(int x=1;x<=n;++x)bfs(x);
55     long long ans=0;
56     for(int y=1;y<=n;++y)ans+=g[match_y[y]][y];
57     return ans;
58 }

```

5.6 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 };

```

5.7 MinimumMeanCycle.cpp

```

1 #include<cstdint>//for DBL_MAX
2 int dp[maxN+1][maxN+1];
3 double mnc(int n){
4     int u,v,w;
5     const int inf=0x7f7f7f7f;
6     memset(dp,0x7f,sizeof(dp));
7     memset(dp[0],0,sizeof(dp[0]));
8     for(int i=0;i<n;++i){
9         for(auto e:E){//tuple<int,int,int>
10             of u,v,w
11             tie(u,v,w)=e;
12             if(dp[i][u]!=inf)
13                 dp[i+1][v]=min(dp[i+1][v],dp[i][u]+w);
14         }
15         double res = DBL_MAX;
16         for(int i=1;i<=n;++i){
17             double val = DBL_MIN;
18             for(int j=0;j<=n;++j)
19                 val=max(val,double(dp[n][i]-dp[i][j])/(n-j));
20             res=min(res,val);
21         }
22     }
23     return res;

```

5.8 Minimum_General_Weighted

```

1 struct Graph {
2     // Minimum General Weighted Matching (
3     // Perfect Match) 0-base
4     static const int MXN = 105;
5     int n, edge[MXN][MXN];
6     int match[MXN],dis[MXN],onstk[MXN];
7     vector<int> stk;
8     void init(int _n) {
9         n = _n;
10        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    int solve() {
40        // find a match
41        for (int i=0; i<n; i+=2){
42            match[i] = i+1;
43            match[i+1] = i;
44        }
45        for(;;){
46            int found = 0;
47            for (int i=0; i<n; i++)
48                dis[i] = onstk[i] = 0;
49            for (int i=0; i<n; i++){
50                stk.clear();
51                if (!onstk[i] && SPFA(i)){
52                    found = 1;
53                    while (stk.size()>=2){
54                        int u = stk.back(); stk.pop_back();
55                        int v = stk.back(); stk.pop_back();
56                        match[u] = v;
57                        match[v] = u;
58                    }
59                }
60            }
61            if (!found) break;
62        }
63        int ret = 0;
64        for (int i=0; i<n; i++)
65            ret += edge[i][match[i]];
66        ret /= 2;
67        return ret;
68    }
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(c){}
21     bool operator<(const edge&e) const{
22         return cost<e.cost;
23     }
24 };
25 struct bit_node{
26     T mi;
27     int id;
28     bit_node(const T&mi=INF, int id=-1):mi(mi),id(id){}
29 };
30 std::vector<bit_node> bit;
31 inline void bit_update(int i, const T&data, int id){
32     for(;i=i&(-i)){
33         if(data<bit[i].mi) bit[i]=bit_node(data, id);
34     }
35 }
36 inline int bit_find(int i, int m){
37     bit_node x;
38     for(;i<=m;i=i&(-i)){
39         if(bit[i].mi<x.mi) x=bit[i];
40     }
41     return x.id;
42 }
43 inline std::vector<edge> build_graph(int n, point p[]){
44     std::vector<edge> e; //回傳的邊就可以用來求最小生成樹
45     for(int dir=0; dir<4; ++dir){ //4種座標變換
46         if(dir%2){
47             for(int i=0; i<n; ++i) std::swap(p[i].x, p[i].y);
48         } else if(dir==2){
49             for(int i=0; i<n; ++i) p[i].x=-p[i].x;
50         }
51         std::sort(p, p+n, cmpx);
52         std::vector<T> ga(n), gb;
53         for(int i=0; i<n; ++i) ga[i]=p[i].y-p[i].x;

```

```

54         gb=ga;
55         std::sort(gb.begin(), gb.end());
56         gb.resize(std::unique(gb.begin(), gb.end())-gb.begin());
57         int m=gb.size();
58         bit=std::vector<bit_node>(m+1);
59         for(int i=n-1; i>=0; --i){
60             int pos=std::lower_bound(gb.begin(), gb.end(), ga[i])-gb.begin()+1;
61             int ans=bit_find(pos, m);
62             if(~ans) e.push_back(edge(p[i].id, p[ans].id, p[i].dist(p[ans])));
63             bit_update(pos, p[i].x+p[i].y, i);
64         }
65     }
66     return e;
67 }

```

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){ //hash ver
6     vis[u]=1;
7     vector<long long> tmp;
8     for(auto v:g[u]) if(!vis[v]) tmp.push_back(dfs(v));
9     if(tmp.empty()) return 177;
10    long long ret=4931;
11    sort(tmp.begin(), tmp.end());
12    for(auto v:tmp) ret=((ret*X)^v)%P;
13    return ret;
14 }
15 //-----
16 string dfs(int x, int p){
17     vector<string> c;
18     for(int y:g[x])
19         if(y!=p) c.emplace_back(dfs(y, x));
20     sort(c.begin(), c.end());
21     string ret("(");
22     for(auto &s:c) ret+=s;
23     ret+=")";
24     return ret;
25 }

```

5.11 全局最小割.cpp

```

1 const int INF=0x3f3f3f3f;
2 template<typename T>
3 struct stoer_wagner{ //0-base
4     static const int MAXN=150;
5     T g[MAXN][MAXN], dis[MAXN];
6     int nd[MAXN], n, s, t;
7     void init(int _n){
8         n=_n;
9         for(int i=0; i<n; ++i)
10             for(int j=0; j<n; ++j) g[i][j]=0;
11     }

```

```

12 void add_edge(int u, int v, T w){
13     g[u][v]=g[v][u]+=w;
14 }
15 T min_cut(){
16     T ans=INF;
17     for(int i=0; i<n; ++i) nd[i]=i;
18     for(int ind, tn=n; tn>1; --tn){
19         for(int i=1; i<tn; ++i) dis[ind[i]]=0;
20         for(int i=1; i<tn; ++i){
21             ind=i;
22             for(int j=i; j<tn; ++j){
23                 dis[ind[j]]+=g[ind[i-1]][nd[j]];
24                 if(dis[ind[j]]<dis[ind[j]]) ind=j;
25             }
26             swap(nd[ind], nd[i]);
27         }
28         if(ans>dis[nd[ind]]) ans=dis[t=nd[ind]], s=nd[ind-1];
29         for(int i=0; i<tn; ++i)
30             g[nd[ind-1]][nd[i]]=g[nd[i]][nd[ind-1]]+=g[nd[i]][nd[ind]];
31     }
32     return ans;
33 }
34 }

```

5.12 平面圖判定.cpp

```

1 static const int MAXN = 20;
2 struct Edge{
3     int u, v;
4     Edge(int s, int d):u(s),v(d){}
5 };
6 bool isK33(int n, int degree[]){
7     int t = 0, z = 0;
8     for(int i=0; i<n; ++i){
9         if(degree[i] == 3) ++t;
10        else if(degree[i] == 0) ++z;
11        else return false;
12    }
13    return t == 6 && t + z == n;
14 }
15 bool isK5(int n, int degree[]){
16     int f = 0, z = 0;
17     for(int i=0; i<n; ++i){
18         if(degree[i] == 4) ++f;
19         else if(degree[i] == 0) ++z;
20         else return false;
21     }
22     return f == 5 && f + z == n;
23 }
24 // it judge a given graph is Homeomorphic with K33 or K5
25 bool isHomeomorphic(bool G[MAXN][MAXN], const int n){
26     for(;;){
27         int cnt = 0;
28         for(int i=0; i<n; ++i){
29             vector<Edge> E;
30             for(int j=0; j<n && E.size()<3; ++j)
31                 if(G[i][j] && i != j)
32                     E.push_back(Edge(i, j));
33             if(E.size() == 1){

```

```

34                 G[i][E[0].v] = G[E[0].v][i] = false;
35             } else if(E.size() == 2){
36                 G[i][E[0].v] = G[E[0].v][i] = false;
37                 G[i][E[1].v] = G[E[1].v][i] = false;
38                 G[E[0].v][E[1].v] = G[E[1].v][E[0].v] = true;
39                 ++cnt;
40             }
41         }
42         if(cnt == 0) break;
43     }
44     static int degree[MAXN];
45     fill(degree, degree + n, 0);
46     for(int i=0; i<n; ++i){
47         for(int j=i+1; j<n; ++j){
48             if(!G[i][j]) continue;
49             ++degree[i];
50             ++degree[j];
51         }
52     }
53     return !(isK33(n, degree) || isK5(n, degree));
54 }

```

5.13 弦圖完美消除序列.cpp

```

1 struct chordal{
2     static const int MAXN=1005;
3     int n; //0-base
4     vector<int> G[MAXN];
5     int rank[MAXN], label[MAXN];
6     bool mark[MAXN];
7     void init(int _n){ n=_n;
8         for(int i=0; i<n; ++i) G[i].clear();
9     }
10    void add_edge(int u, int v){
11        G[u].push_back(v);
12        G[v].push_back(u);
13    }
14    vector<int> MCS(){
15        memset(rank, -1, sizeof(int)*n);
16        memset(label, 0, sizeof(int)*n);
17        priority_queue<pair<int, int>> pq;
18        for(int i=0; i<n; ++i) pq.push(make_pair(0, i));
19        for(int i=n-1; i>=0; --i) for(;;){
20            int u=pq.top().second; pq.pop();
21            if(~rank[u]) continue;
22            rank[u]=i;
23            for(auto v:G[u]) if(rank[v]==-1){
24                pq.push(make_pair(++label[v], v));
25            }
26            break;
27        }
28        vector<int> res(n);
29        for(int i=0; i<n; ++i) res[rank[i]]=i;
30        return res;
31    }
32    bool check(vector<int> ord){ //弦圖判定
33        for(int i=0; i<n; ++i) rank[ord[i]]=i;
34        memset(mark, 0, sizeof(bool)*n);
35        for(int i=0; i<n; ++i){
36            vector<pair<int, int>> tmp;

```

```

37 for(auto u:G[ord[i]])if(!mark[u])
38     tmp.push_back(make_pair(rank[u],u));
39 sort(tmp.begin(),tmp.end());
40 if(tmp.size()){
41     int u=tmp[0].second;
42     set<int> S;
43     for(auto v:G[u])S.insert(v);
44     for(size_t j=1;j<tmp.size();++j)
45         if(!S.count(tmp[j].second))return
46             0;
47     mark[ord[i]]=1;
48 }
49 return 1;
50 }
51 };

```

5.14 最小斯坦納樹 DP.cpp

```

1 //n個點，其中r個要構成斯坦納樹
2 //答案在max(dp[(1<<r)-1][k]) k=0~n-1
3 //p表示要構成斯坦納樹的點集
4 //O( n^3 + n*3^n + n^2*2^n )
5 #define REP(i,n) for(int i=0;i<(int)n;++i)
6 const int MAXN=30,MAXM=8; // 0-base
7 const int INF=0x3f3f3f3f;
8 int dp[1<<MAXN][MAXN];
9 int g[MAXN][MAXN]; //圖
10 void init(){memset(g,0x3f,sizeof(g));}
11 void add_edge(int u,int v,int w){
12     g[u][v]=g[v][u]=min(g[v][u],w);
13 }
14 void steiner(int n,int r,int *p){
15     REP(k,n)REP(i,n)REP(j,n)
16         g[i][j]=min(g[i][j],g[i][k]+g[k][j]);
17     REP(i,n)g[i][i]=0;
18     REP(i,r)REP(j,n)dp[1<<i][j]=g[p[i]][j];
19     for(int i=1;i<(1<<r);++i){
20         if(!(i&(i-1)))continue;
21         REP(j,n)dp[i][j]=INF;
22         REP(j,n){
23             int tmp=INF;
24             for(int s=i&(i-1);s;s=s^(s-1))
25                 tmp=min(tmp,dp[s][j]+dp[i^s][j]);
26             REP(k,n)dp[i][k]=min(dp[i][k],g[j][k]+
27                 tmp);
28         }
29 }

```

5.15 最小樹形圖 朱劉.cpp

```

1 #define INF 0x3f3f3f3f
2 template<typename T>
3 struct zhu_liu{
4     static const int MAXN=110;
5     struct edge{
6         int u,v;
7         T w;

```

```

8     edge(int u=0,int v=0,T w=0):u(u),v(v),w(
9         w){}
10 };
11 vector<edge>E; // 0-base
12 int pe[MAXN],id[MAXN],vis[MAXN];
13 T in[MAXN];
14 void init(){E.clear();}
15 void add_edge(int u,int v,T w){
16     if(u!=v)E.push_back(edge(u,v,w));
17 }
18 T build(int root,int n){
19     T ans=0;int N=n;
20     for(;;){
21         for(int u=0;u<n;++u)in[u]=INF;
22         for(size_t i=0;i<E.size();++i)
23             if(E[i].u!=E[i].v&&E[i].w<in[E[i].v])
24                 pe[E[i].v]=i,in[E[i].v]=E[i].w;
25         for(int u=0;u<n;++u) // 無解
26             if(u!=root&&in[u]==INF)return -INF;
27         int cntnode=0;
28         memset(id,-1,sizeof(int)*N);
29         memset(vis,-1,sizeof(int)*N);
30         for(int u=0;u<n;++u){
31             if(u!=root)ans+=in[u];
32             int v=u;
33             for(;vis[v]!&id[v]==-1&&v!=root;v
34                 =E[pe[v]].u)
35                 vis[v]=u;
36             if(v!=root&&id[v]==-1){
37                 for(int x=E[pe[v]].u;x!=v;x=E[pe[x]
38                     ].u)
39                     id[x]=cntnode;
40                 id[v]=cntnode++;
41             }
42             if(!cntnode)break; // 無環
43             for(int u=0;u<n;++u)if(id[u]==-1)id[u]
44                 =cntnode++;
45             for(size_t i=0;i<E.size();++i){
46                 int v=E[i].v;
47                 E[i].u=id[E[i].u];
48                 E[i].v=id[E[i].v];
49                 if(E[i].u!=E[i].v)E[i].w-=in[v];
50             }
51             n=cntnode;
52             root=id[root];
53         }
54     }
55     return ans;
56 }

```

5.16 穩定婚姻模板.cpp

```

1 queue<int> Q;
2 for ( i : 所有考生 ) {
3     設定在第0志願;
4     Q.push(考生i);
5 }
6 while(Q.size()){
7     當前考生=Q.front();Q.pop();
8     while ( 此考生未分發 ) {
9         指標移到下一志願;

```

```

10 if ( 已經沒有志願 or 超出志願總數 )
11     break;
12 計算該考生在該科系加權後的總分;
13 if ( 不符合科系需求 ) continue;
14 if ( 目前科系有餘額 ) {
15     依加權後分數高低順序將考生id加入科系錄
16     取名單中;
17     break;
18 }
19 if ( 目前科系已額滿 ) {
20     if ( 此考生成績比最低分數還高 ) {
21         依加權後分數高低順序將考生id加入科系
22         錄取名單;
23         Q.push(被踢出的考生);
24     }
25 }
26 }

```

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
7         ),cost(c){}
8 };
9 map<char,int> rule; //每個字元對應到的規則，
10 小寫字母為終端字符
11 vector<CNF> cnf;
12 inline void init(){
13     state=0;
14     rule.clear();
15     cnf.clear();
16 }
17 inline void add_to_cnf(char s,const string &
18     p,int cost){
19     //加入一個s -> <p>的文法，代價為cost
20     if(rule.find(s)==rule.end())rule[s]=state
21         ++;
22     for(auto c:p)if(rule.find(c)==rule.end())
23         rule[c]=state++;
24     if(p.size()==1){
25         cnf.push_back(CNF(rule[s],rule[p[0]],-1,
26             cost));
27     }else{
28         int left=rule[s];
29         int sz=p.size();
30         for(int i=0;i<sz-2;++i){
31             cnf.push_back(CNF(left,rule[p[i]],
32                 state,0));
33             left=state++;
34         }
35     }
36 }

```

```

29 cnf.push_back(CNF(left,rule[p[sz-2]],
30     rule[p[sz-1]],cost));
31 }
32 }
33 vector<long long> dp[MAXN][MAXN];
34 vector<bool> neg_INF[MAXN][MAXN]; //如果花費
35 是負的可能會有無限小的情形
36 inline void relax(int l,int r,const CNF &c,
37     long long cost,bool neg_c=0){
38     if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x
39         ]||cost<dp[l][r][c.s])){
40         if(neg_c||neg_INF[l][r][c.x]){
41             dp[l][r][c.s]=0;
42             neg_INF[l][r][c.s]=true;
43         }else dp[l][r][c.s]=cost;
44     }
45 }
46 inline void bellman(int l,int r,int n){
47     for(int k=1;k<=state;++k)
48         for(auto c:cnf)
49             if(c.y==-1)relax(l,r,c,dp[l][r][c.x]+
50                 c.cost,k==n);
51 }
52 inline void cyk(const vector<int> &tok){
53     for(int i=0;i<(int)tok.size();++i){
54         for(int j=0;j<(int)tok.size();++j){
55             dp[i][j]=vector<long long>(state+1,
56                 INT_MAX);
57             neg_INF[i][j]=vector<bool>(state+1,
58                 false);
59         }
60         dp[i][i][tok[i]]=0;
61         bellman(i,i,tok.size());
62     }
63     for(int r=1;r<(int)tok.size();++r){
64         for(int l=r-1;l>=0;--l){
65             for(int k=1;k<=r;++k)
66                 for(auto c:cnf)
67                     if(~c.y)relax(l,r,c,dp[l][k][c.x]+
68                         dp[k+1][r][c.y]+c.cost);
69             bellman(l,r,tok.size());
70         }
71     }
72 }

```

7 Linear_Programming

7.1 最大密度子圖.cpp

```

1 typedef double T; //POJ 3155
2 const int MAXN=105;
3 struct edge{
4     int u,v;
5     T w;
6     edge(int u=0,int v=0,T w=0):u(u),v(v),w(w)
7     {}
8 };
9 vector<edge> E;
10 int n,m; // 1-base

```

```

10 T de[MAXN],pv[MAXN]; //每個點的邊權和和點權(
    有些題目會給)
11 void init(){
12     E.clear();
13     for(int i=1;i<=n;++i)de[i]=pv[i]=0;
14 }
15 void add_edge(int u,int v,T w){
16     E.push_back(edge(u,v,w));
17     de[u]+=w,de[v]+=w;
18 }
19 T U; //二分搜的最大值
20 void get_U(){
21     U=0;
22     for(int i=1;i<=n;++i)U+=2*pv[i];
23     for(size_t i=0;i<E.size();++i)U+=E[i].w;
24 }
25 ISAP<T> isap; //網路流
26 int s,t; //原匯點
27 void build(T L){
28     isap.init(n+2);
29     for(size_t i=0;i<E.size();++i){
30         isap.add_edge(E[i].u,E[i].v,E[i].w);
31     }
32     for(int v=1;v<=n;++v){
33         isap.add_edge(s,v,U);
34         isap.add_edge(v,t,U+2*L-de[v]-2*pv[v]);
35     }
36 }
37 int main(){
38     while(~scanf("%d%d",&n,&m)){
39         if(!m){
40             puts("1\n1");
41             continue;
42         }
43         init();
44         int u,v;
45         for(int i=0;i<m;++i){
46             scanf("%d%d",&u,&v);
47             add_edge(u,v,1);
48         }
49         get_U();
50         s=n+1,t=n+2;
51         T l=0,r=U,k=1.0/(n*n);
52         while(r-l>k){ //二分搜最大值
53             T mid=(l+r)/2;
54             build(mid);
55             T res=(U*n-isap.isap(s,t))/2;
56             if(res>0)l=mid;
57             else r=mid;
58         }
59         build(l);
60         isap.min_cut(s,t);
61         vector<int> ans;
62         for(int i=1;i<=n;++i){
63             if(isap.vis[i])ans.push_back(i);
64         }
65         printf("%d\n",ans.size());
66         for(size_t i=0;i<ans.size();++i){
67             printf("%d\n",ans[i]);
68         }
69     }
70     return 0;
71 }

```

8 Number_Theory

8.1 basic.cpp

```

1 template<typename T>
2 void gcd(const T &a,const T &b,T &d,T &x,T &y){
3     if(!b) d=a,x=1,y=0;
4     else gcd(b,a%b,d,y,x), y-=x*(a/b);
5 }
6 long long int phi[N+1];
7 void phiTable(){
8     for(int i=1;i<=N;i++)phi[i]=i;
9     for(int i=1;i<=N;i++)for(x=i*2;x<=N;x+=i)
10         phi[x]-=phi[i];
11 }
12 void all_divdown(const LL &n){ // all n/x
13     for(LL a=1;a<=n;a=n/(n/(a+1))){
14         // dosomething;
15     }
16 }
17 const int MAXPRIME = 1000000;
18 int iscom[MAXPRIME], prime[MAXPRIME],
19     primecnt;
20 int phi[MAXPRIME], mu[MAXPRIME];
21 void sieve(void){
22     memset(iscom,0,sizeof(iscom));
23     primecnt = 0;
24     phi[1] = mu[1] = 1;
25     for(int i=2;i<MAXPRIME;++i){
26         if(!iscom[i]){
27             prime[primecnt++] = i;
28             mu[i] = -1;
29             phi[i] = i-1;
30         }
31         for(int j=0;j<primecnt;++j){
32             int k = i * prime[j];
33             if(k>=MAXPRIME) break;
34             iscom[k] = prime[j];
35             if(i%prime[j]==0){
36                 mu[k] = 0;
37                 phi[k] = phi[i] * prime[j];
38                 break;
39             } else {
40                 mu[k] = -mu[i];
41                 phi[k] = phi[i] * (prime[j]-1);
42             }
43         }
44     }
45 }
46 bool g_test(const LL &g, const LL &p, const
47     vector<LL> &v){
48     for(int i=0;i<v.size();++i)
49         if(modexp(g,(p-1)/v[i],p)==1)
50             return false;
51     return true;
52 }
53 LL primitive_root(const LL &p){
54     if(p==2) return 1;
55     vector<LL> v;
56     Factor(p-1,v);

```

```

55     v.erase(unique(v.begin(), v.end()), v.end
56         ());
57     for(LL g=2;g<p;++g)
58         if(g_test(g,p,v))
59             return g;
60     puts("primitive_root NOT FOUND");
61     return -1;
62 }
63 int Legendre(const LL &a, const LL &p){
64     return modexp(a%p,(p-1)/2,p);
65 }
66 LL inv(const LL &a, const LL &n){
67     LL d,x,y;
68     gcd(a,n,d,x,y);
69     return d==1 ? (x+n)%n : -1;
70 }
71 int inv[maxN];
72 LL invtable(int n,LL P){
73     inv[1]=1;
74     for(int i=2;i<n;++i)
75         inv[i]=(P-(P/i))*inv[P%i]%P;
76 }
77 LL log_mod(const LL &a, const LL &b, const
78     LL &p){
79     // a ^ x = b ( mod p )
80     int m=sqrt(p+.5), e=1;
81     LL v=inv(modexp(a,m,p), p);
82     map<LL,int> x;
83     x[1]=0;
84     for(int i=1;i<m;++i){
85         e = LLMul(e,a,p);
86         if(x.count(e)) x[e] = i;
87     }
88     for(int i=0;i<m;++i){
89         if(x.count(b)) return i*m + x[b];
90         b = LLMul(b,v,p);
91     }
92     return -1;
93 }
94 LL Tonelli_Shanks(const LL &n, const LL &p){
95     {
96         // x^2 = n ( mod p )
97         if(n==0) return 0;
98         if(Legendre(n,p)!=1) while(1){ puts("SQRT
99             ROOT does not exist"); }
100     }
101     int S = 0;
102     LL Q = p-1;
103     while( !(Q&1) ) { Q>>=1; ++S; }
104     if(S==1) return modexp(n%p,(p+1)/4,p);
105     LL z = 2;
106     for(; Legendre(z,p)!=-1;++z)
107         LL c = modexp(z,Q,p);
108     LL R = modexp(n%p,(Q+1)/2,p), t = modexp(n
109         %p,Q,p);
110     int M = S;
111     while(1){
112         LL b = modexp(c,1<<(M-i-1),p);
113         R = LLMul(R,b,p);
114         t = LLMul(LLmul(b,b,p), t, p);
115         c = LLMul(b,b,p);
116         M = i;
117     }

```

```

115     return -1;
116 }
117 template<typename T>
118 T Euler(T n){
119     T ans=n;
120     for(T i=2;i*i<=n;++i){
121         if(n%i==0){
122             ans=ans/i*(i-1);
123             while(n%i==0)n/=i;
124         }
125     }
126     if(n>1)ans=ans/n*(n-1);
127     return ans;
128 }
129 //Chinese_remainder_theorem
130 template<typename T>
131 T pow_mod(T n,T k,T m){
132     T ans=1;
133     for(n=(n==m?n%m:n);k;k>>=1){
134         if(k&1)ans=ans*n%m;
135         n=n*n%m;
136     }
137     return ans;
138 }
139 template<typename T>
140 T crt(vector<T> &m,vector<T> &a){
141     T M=1,tM,ans=0;
142     for(int i=0;i<(int)m.size();++i)M*=m[i];
143     for(int i=0;i<(int)a.size();++i){
144         tM=M/m[i];
145         ans=(ans+(a[i]*tM%M)*pow_mod(tM,Euler(m
146             [i])-1,m[i])%M)%M;
147     }
148     /*如果m[i]是質數·Euler(m[i])-1=m[i]-2·
149     就不用算Euler了*/
150     return ans;
151 }
152 //java code
153 //求sqrt(N)的連分數
154 public static void Pell(int n){
155     BigInteger N,p1,p2,q1,q2,a0,a1,a2,g1,g2,h1
156         ,h2,p,q;
157     g1=q2=p1=BigInteger.ZERO;
158     h1=q1=p2=BigInteger.ONE;
159     a0=a1=BigInteger.valueOf((int)Math.sqrt
160         (1.0*n));
161     BigInteger ans=a0.multiply(a0);
162     if(ans.equals(BigInteger.valueOf(n))){
163         System.out.println("No solution!");
164         return ;
165     }
166     while(true){
167         g2=a1.multiply(h1).subtract(g1);
168         h2=N.subtract(g2.pow(2)).divide(h1);
169         a2=g2.add(a0).divide(h2);
170         p=a1.multiply(p2).add(p1);
171         q=a1.multiply(q2).add(q1);
172         if(p.pow(2).subtract(N.multiply(q.pow
173             (2))).compareTo(BigInteger.ONE)==0)
174             break;
175         g1=g2;h1=h2;a1=a2;
176         p1=p2;p2=p;

```


8.2 bit_set.cpp

```

174     q1=q2;q2=q;
175 }
176 System.out.println(p+" "+q);
177 }

```

8.3 cantor_expansion.cpp

```

1 int factorial[MAXN];
2 void init(){
3     factorial[0]=1;
4     for(int i=1;i<=MAXN;++i)factorial[i]=
5         factorial[i-1]*i;
6 }
7 int encode(const vector<int> &s){
8     int n=s.size(),res=0;
9     for(int i=0;i<n;++i){
10         int t=0;
11         for(int j=i+1;j<n;++j)
12             if(s[j]<s[i])++t;
13         res+=t*factorial[n-i-1];
14     }
15     return res;
16 }
17 vector<int> decode(int a,int n){
18     vector<int> res;
19     vector<bool> vis(n,0);
20     for(int i=n-1;i>=0;--i){
21         int t=a/factorial[i],j;
22         for(j=0;j<n;++j)
23             if(!vis[j]){
24                 if(t==0)break;
25                 --t;
26             }
27         res.push_back(j);
28         vis[j]=1;
29         a%=factorial[i];
30     }
31     return res;

```

8.4 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&0xAAAAAAAAU)
9             >>1);
10        a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)
11            >>2);
12        a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0F0U)
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 };

```

8.5 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 double get(const vector<double> &coef, double
6     x){
7     double e = 1, s = 0;
8     for(auto i : coef) s += i*e, e *= x;
9     return s;
10 }
11 double find(const vector<double> &coef, int n
12     , double lo, double hi){
13     double sign_lo, sign_hi;

```

```

14 if( !(sign_lo = sign(get(coef,lo))) )
15     return lo;
16 if( !(sign_hi = sign(get(coef,hi))) )
17     return hi;
18 if(sign_lo * sign_hi > 0) return INF;
19 for(int stp = 0; stp < 100 && hi - lo >
20     eps; ++stp){
21     double m = (lo+hi)/2.0;
22     int sign_mid = sign(get(coef,m));
23     if(!sign_mid) return m;
24     if(sign_lo*sign_mid < 0) hi = m;
25     else lo = m;
26 }
27 return (lo+hi)/2.0;
28 }
29 vector<double> cal(vector<double>coef, int n
30 ){
31     vector<double>res;
32     if(n == 1){
33         if(sign(coef[1])) res.pb(-coef[0]/coef
34             [1]);
35         return res;
36     }
37     vector<double>dcoef(n);
38     for(int i = 0; i < n; ++i) dcoef[i] = coef
39         [i+1]*(i+1);
40     vector<double>droot = cal(dcoef, n-1);
41     droot.insert(droot.begin(), -INF);
42     droot.pb(INF);
43     for(int i = 0; i+1 < droot.size(); ++i){
44         double tmp = find(coef, n, droot[i],
45             droot[i+1]);
46         if(tmp < INF) res.pb(tmp);
47     }
48     return res;
49 }

```

8.6 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,0LL)
7             ;
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 make_pair
15            (-1LL,0LL);
16        lastb = LLmul((lastb-b[i])/d,x,(lastm/d)
17            )*m[i];

```

```

14     lastm = (lastm/d)*m[i];
15     lastb = (lastb+b[i])%lastm;
16 }
17 return make_pair(lastb<0?lastb+lastm:lastb
18     ,lastm);

```

8.7 Lucas.cpp

```

1 int mod_fact(int n,int &e){
2     e=0;
3     if(n==0)return 1;
4     int res=mod_fact(n/P,e);
5     e += n/P;
6     if((n/P)%2==0)return res*fact[n%P]%P;
7     return res*(P-fact[n%P])%P;
8 }
9 int Cmod(int n,int m){
10     int a1,a2,a3,e1,e2,e3;
11     a1=mod_fact(n,e1);
12     a2=mod_fact(m,e2);
13     a3=mod_fact(n-m,e3);
14     if(e1>e2+e3)return 0;
15     return a1*inv(a2*a3%P,P)%P;
16 }

```

8.8 Matrix.cpp

```

1 template<typename T>
2 struct Matrix{
3     using rt = std::vector<T>;
4     using mt = std::vector<rt>;
5     using matrix = Matrix<T>;
6     int r,c;
7     mt m;
8     Matrix(int r,int c):r(r),c(c),m(r,rt(c)){
9         rt& operator[](int i){return m[i];}
10    matrix operator+(const matrix &a){
11        matrix rev(r,c);
12        for(int i=0;i<r;++i)
13            for(int j=0;j<c;++j)
14                rev[i][j]=m[i][j]+a.m[i][j];
15        return rev;
16    }
17    matrix operator-(const matrix &a){
18        matrix rev(r,c);
19        for(int i=0;i<r;++i)
20            for(int j=0;j<c;++j)
21                rev[i][j]=m[i][j]-a.m[i][j];
22        return rev;
23    }
24    matrix operator*(const matrix &a){
25        matrix rev(r,a.c);
26        matrix tmp(a.c,a.r);
27        for(int i=0;i<a.r;++i)
28            for(int j=0;j<a.c;++j)
29                tmp[j][i]=a.m[i][j];
30        for(int i=0;i<r;++i)
31            for(int j=0;j<a.c;++j)
32                for(int k=0;k<a.r;++k)

```

```

33     rev.m[i][j]+=m[i][k]*tmp[j][k];
34     return rev;
35 }
36 bool inverse(){
37     Matrix t(r,r+c);
38     for(int y=0;y<r;y++){
39         t.m[y][c+y] = 1;
40         for(int x=0;x<c;+x)
41             t.m[y][x]=m[y][x];
42     }
43     if( !t.gas() )
44         return false;
45     for(int y=0;y<r;y++)
46         for(int x=0;x<c;+x)
47             m[y][x]=t.m[y][c+x]/t.m[y][y];
48     return true;
49 }
50 T gas(){
51     vector<T> lazy(r,1);
52     bool sign=false;
53     for(int i=0;i<r;+i){
54         if( m[i][i]==0 ){
55             int j=i+1;
56             while(j<r&&!m[j][i])j++;
57             if(j==r)continue;
58             m[i].swap(m[j]);
59             sign=!sign;
60         }
61         for(int j=0;j<r;+j){
62             if(i==j)continue;
63             lazy[j]=lazy[j]*m[i][i];
64             T mx=m[j][i];
65             for(int k=0;k<c;+k)
66                 m[j][k]=m[j][k]*m[i][i]-m[i][k]*mx;
67         }
68     }
69     T det=sign?-1:1;
70     for(int i=0;i<r;+i){
71         det = det*m[i][i];
72         det = det/lazy[i];
73         for(auto &j:m[i])j/=lazy[i];
74     }
75     return det;
76 }
77 };

```

8.9 MillerRobin.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 LL mod_mul(LL a,LL b,LL m){
14     a%=m,b%=m; /* fast for m < 2^58 */

```

```

15     LL y=(LL)((double)a*b/m+0.5);
16     LL r=(a*b-y*m)%m;
17     return r<0?r+m:r;
18 }
19 template<typename T>
20 T pow(T a,T b,T mod){ //a^b%mod
21     T ans=1;
22     for(;b;a=mod_mul(a,a,mod),b>>=1)
23         if(b&1)ans=mod_mul(ans,a,mod);
24     return ans;
25 }
26 int sprp[3]={2,7,61}; //int範圍可解
27 int llsprp
28 [7]={2,325,9375,28178,450775,9780504,179526
29 //至少unsigned Long Long範圍
30 template<typename T>
31 bool isprime(T n,int *sprp,int num){
32     if(n==2)return 1;
33     if(n<2||n%2==0)return 0;
34     int t=0;
35     T u=n-1;
36     for(;u%2==0;+t)u>>=1;
37     for(int i=0;i<num;+i){
38         T a=sprp[i]%n;
39         if(a==0||a==1||a==n-1)continue;
40         T x=pow(a,u,n);
41         if(x==1||x==n-1)continue;
42         for(int j=0;j<t;+j){
43             x=mod_mul(x,x,n);
44             if(x==1)return 0;
45             if(x==n-1)break;
46         }
47         if(x==n-1)continue;
48         return 0;
49     }
50     return 1;
51 }

```

8.10 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&0xAAAAAAAU)
15                 >>1);
16             a=((a&0x33333333U)<<2)|((a&0xCCCCCCU)
17                 >>2);
18             a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0FU)
19                 >>4);
20             a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)
21                 >>8);
22             a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)
23                 >>16);

```

```

17     return a>>(32-len);
18 }
19 T pow_mod(T n,T k,T m){
20     T ans=1;
21     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 };

```

8.11 Simpson.cpp

```

1 double simpson(double a,double b){
2     double c=a+(b-a)/2;
3     return (F(a)+4*F(c)+F(b))*(b-a)/6;
4 }
5 double asr(double a,double b,double eps,
6     double A){
7     double c=a+(b-a)/2;
8     double L=simpson(a,c),R=simpson(c,b);
9     if( abs(L-R-A)<15*eps )
10         return L+R+(L+R-A)/15.0;
11     return asr(a,c,eps/2,L)+asr(c,b,eps/2,R)
12         ;
13 }
14 double asr(double a,double b,double eps){
15     return asr(a,b,eps,simpson(a,b));
16 }

```

8.12 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 long long a[1000005];
49 int t,mod;
50 int main(){
51     init_euler();
52     scanf("%d",&t);
53     #define n 4
54     while(t--){
55         for(int i=0;i<n;+i)scanf("%lld",&a[i]);
56         scanf("%d",&mod);
57         printf("%lld\n",high_pow(a,n,mod));
58     }
59     return 0;
60 }

```

8.13 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)%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    };

```

8.14 質因數分解.cpp

```

1 LL func(const LL n,const LL mod,const int c)
2 {
3     return (LLmul(n,n,mod)+c+mod)%mod;
4 }
5 LL pollorrho(const LL n, const int c) { //循環節長度
6     LL a=1, b=1;
7     a=func(a,n,c)%n;
8     b=func(b,n,c)%n; b=func(b,n,c)%n;
9     while(gcd(abs(a-b),n)==1) {
10        a=func(a,n,c)%n;
11        b=func(b,n,c)%n; b=func(b,n,c)%n;
12    }
13    return gcd(abs(a-b),n);
14 }
15 void prefactor(LL &n, vector<LL> &v) {
16     for(int i=0;i<12;++i) {
17         while(n%prime[i]==0) {

```

```

19         v.push_back(prime[i]);
20         n/=prime[i];
21     }
22 }
23 void smallfactor(LL n, vector<LL> &v) {
24     if(n<MAXPRIME) {
25         while(isp[(int)n]) {
26             v.push_back(isp[(int)n]);
27             n/=isp[(int)n];
28         }
29         v.push_back(n);
30     }
31     else {
32         for(int i=0;i<primecnt&&prime[i]*prime[i]
33             ]<=n;++i) {
34             while(n%prime[i]==0) {
35                 v.push_back(prime[i]);
36                 n/=prime[i];
37             }
38             if(n!=1) v.push_back(n);
39         }
40     }
41 }
42 void comfactor(const LL &n, vector<LL> &v) {
43     if(n<1e9) {
44         smallfactor(n,v);
45         return;
46     }
47     if(Isprime(n)) {
48         v.push_back(n);
49         return;
50     }
51     LL d;
52     for(int c=3; c<=n; c++) {
53         d = pollorrho(n,c);
54         if(d!=n) break;
55     }
56     comfactor(d,v);
57     comfactor(n/d,v);
58 }
59 void Factor(const LL &x, vector<LL> &v) {
60     LL n = x;
61     if(n==1) { puts("Factor 1"); return; }
62     prefactor(n,v);
63     if(n==1) return;
64     comfactor(n,v);
65     sort(v.begin(),v.end());
66 }
67 void AllFactor(const LL &n,vector<LL> &v) {
68     vector<LL> tmp;
69     Factor(n,tmp);
70     v.clear();
71     v.push_back(1);
72     int len;
73     LL now=1;
74     for(int i=0;i<tmp.size();++i) {
75         if(i==0 || tmp[i]!=tmp[i-1]) {
76             len = v.size();
77             now = 1;
78         }
79         now*=tmp[i];
80         for(int j=0;j<len;++j)

```

```

84         v.push_back(v[j]*now);
85     }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

9 other

9.1 WhatDay.cpp

```

1 int whatday(int y,int m,int d){
2     if(m<=2)m+=12,--y;
3     if(y<1752||y==1752&&m<9||y==1752&&m==9&&
4         d<3)
5         return (d+2*m+3*(m+1)/5+y+y/4+5)%7;
6     return (d+2*m+3*(m+1)/5+y+y/4-y/100+y
7         /400)%7;
8 }

```

9.2 上下最大正方形.cpp

```

1 void solve(int n,int a[],int b[]){ // 1-base
2     int ans=0;
3     deque<int>da,db;
4     for(int l=1,r=1;r<=n;++r){
5         while(da.size()&&a[da.back()]>=a[r]){
6             da.pop_back();
7         }
8         da.push_back(r);
9         while(db.size()&&b[db.back()]>=b[r]){
10            db.pop_back();
11        }
12        db.push_back(r);
13        for(int d=a[da.front()]+b[db.front()];r-
14            l+1>d;++l){
15            if(da.front()==l)da.pop_front();
16            if(db.front()==l)db.pop_front();
17            if(da.size()&&db.size()){
18                d=a[da.front()]+b[db.front()];
19            }
20        }
21        ans=max(ans,r-l+1);
22    }
23    printf("%d\n",ans);
24 }

```

9.3 最大矩形.cpp

```

1 long long max_rectangle(vector<int> s){
2     stack<pair<int,int>> st;
3     st.push(make_pair(-1,0));
4     s.push_back(0);
5     long long ans=0;
6     for(size_t i=0;i<s.size();++i){
7         int h=s[i];
8         pair<int,int> now=make_pair(h,i);

```

```

9         while(h<st.top().first){
10            now=st.top();
11            st.pop();
12            ans=max(ans,(long long)(i-now.second)*
13                now.first);
14        }
15        if(h>st.top().first){
16            st.push(make_pair(h,now.second));
17        }
18    }
19    return ans;
20 }

```

10 String

10.1 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(int ed, cnt_dp, vis){
9             for(int i=0;i<R-L+1;++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+1;++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[i];++i){
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            ++S[o].ed;
33        }
34        void build_fail(){
35            S[0].fail=S[0].efl=-1;
36            q.clear();
37            q.push_back(0);
38            ++qe;
39            while(qs!=qe){
40                int pa=q[qs++],id,t;
41                for(int i=0;i<R-L+1;++i){
42                    t=S[pa].next[i];
43                    if(!t)continue;
44                    id=S[pa].fail;

```

```

44 while(~id&&S[id].next[i])id=S[id] 98 for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[
    ].fail; 99 t].efl){
45 S[t].fail=~id?S[id].next[i]:0; 100 S[t].vis=vt;
46 S[t].efl=S[t].fail.ed?S[t].fail 101 ans+=S[t].ed; /*因為都走efl邊所以保
    :S[S[t].fail].efl; 證匹配成功*/
47 q.push_back(t); 102 }
48 ++qe; 103 }
49 } 104 return ans;
50 } 105 }
51 } /*把AC自動機變成真的自動機*/
52 /*DP出每個前綴在字串s出現的次數並傳回所 106 void evolution(){
    有字串被s匹配成功的次數O(N+M)*/ 107 for(qs=1;qs!=qe;){
53 int match_0(const char *s){ 108 int p=q[qs++];
54 int ans=0,id,p=0,i; 109 for(int i=0;i<R-L;++i)
55 for(i=0;s[i];++i){ 110 if(S[p].next[i]==0)S[p].next[i]=S[
56 id=s[i]-L; S[p].fail].next[i];
57 while(!S[p].next[id]&&p=S[p].fail; 111 }
58 if(!S[p].next[id])continue; 112 }
59 p=S[p].next[id]; 113 };
60 ++S[p].cnt_dp; /*匹配成功則它所有後綴
    都可以被匹配(DP計算)*/
61 }
62 for(i=qe-1;i>=0;--i){
63 ans+=S[q[i]].cnt_dp*S[q[i]].ed;
64 if(~S[q[i]].fail)S[S[q[i]].fail].
    cnt_dp+=S[q[i]].cnt_dp;
65 }
66 return ans;
67 }
68 /*多串匹配走efl邊並傳回所有字串被s匹配成
    功的次數O(N*M^1.5)*/
69 int match_1(const char *s)const{
70 int ans=0,id,p=0,t;
71 for(int i=0;s[i];++i){
72 id=s[i]-L;
73 while(!S[p].next[id]&&p=S[p].fail;
74 if(!S[p].next[id])continue;
75 p=S[p].next[id];
76 if(S[p].ed)ans+=S[p].ed;
77 for(t=S[p].efl;~t;t=S[t].efl){
78 ans+=S[t].ed; /*因為都走efl邊所以保
    證匹配成功*/
79 }
80 }
81 return ans;
82 }
83 /*枚舉(s的子字串nA)的所有相異字串各恰一
    次並傳回次數O(N*M^(1/3))*/
84 int match_2(const char *s){
85 int ans=0,id,p=0,t;
86 ++vt;
87 /*把截記vt+=1，只要vt沒溢位，所有S[p].
    vis==vt就會變成false
88 這種利用vt的方法可以O(1)歸零vis陣列*/
89 for(int i=0;s[i];++i){
90 id=s[i]-L;
91 while(!S[p].next[id]&&p=S[p].fail;
92 if(!S[p].next[id])continue;
93 p=S[p].next[id];
94 if(S[p].ed&&S[p].vis!=vt){
95 S[p].vis=vt;
96 ans+=S[p].ed;
97 }

```

10.2 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)%
    prime_mod*/
8 inline void hash_init(int len,T prime=0
    xdefaced){
9 h_base[0]=1;
10 for(int i=1;i<len;++i){
11 h[i]=(h[i-1]*prime+s[i-1])%prime_mod;
12 h_base[i]=(h_base[i-1]*prime)%prime_mod;
13 }
14 }
15 inline T get_hash(int l,int r){/*閉區間寫
    法，設編號為0 ~ Len-1*/
16 return (h[r+1]-(h[l]*h_base[r-l+1])%
    prime_mod+prime_mod)%prime_mod;
17 }

```

10.3 KMP.cpp

```

1 /*產生fail function*/
2 inline void kmp_fail(char *s,int len,int *
    fail){
3 int id=-1;
4 fail[0]=-1;
5 for(int i=1;i<len;++i){
6 while(~id&&s[id+1]!=s[i])id=fail[id];
7 if(s[id+1]==s[i])++id;
8 fail[i]=id;
9 }
10 }
11 /*以字串B匹配字串A，傳回匹配成功的數量(用B的
    fail)*/

```

```

12 inline int kmp_match(char *A,int lenA,char *
    B,int lenB,int *fail){
13 int id=-1,ans=0;
14 for(int i=0;i<lenA;++i){
15 while(~id&&B[id+1]!=A[i])id=fail[id];
16 if(B[id+1]==A[i])++id;
17 if(id==lenB-1){/*匹配成功*/
18 ++ans;
19 id=fail[id];
20 }
21 }
22 return ans;
23 }

```

10.4 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 int l=0,r=0;
5 for(int i=1;i<len;++i){
6 z[i]=r>i?min(z[2*i-l],r-i):1;
7 while(s[i+z[i]]==s[i-z[i]])++z[i];
8 if(z[i]+i>r)r=z[i]+i,l=i;
9 }
10 }

```

10.5 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){
7 if(t>0)i+=k;
8 else j+=k;
9 if(i==j)++j;
10 k=0;
11 }
12 }
13 return min(i,j); /*傳回最小循環表示法起始位
    置
14 }

```

10.6 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<n;++i)c[x[y[i]]]++; \
4 for(i=1;i<A;++i)c[i]+=c[i-1]; \
5 for(i=n-1;~i;--i)sa[-c[x[y[i]]]]=y[i]; \
6 }
7 #define sac(r,a,b) r[a]!=r[b]||a+k>n||r[a+k
    ]!=r[b+k]

```

```

8 void suffix_array(const char *s,int n,int *
    sa,int *rank,int *tmp,int *c){
9 int A='z'+1,i,k,id=0;
10 for(i=0;i<n;++i)rank[tmp[i]]=s[i];
11 radix_sort(rank,tmp);
12 for(k=1;id<n-1;k<=<1){
13 for(id=0,i=n-k;i<n;++i)tmp[id++]=i;
14 for(i=0;i<n;++i)if(sa[i]>=k)tmp[id++]=sa
    [i]-k;
15 radix_sort(rank,tmp);
16 swap(rank,tmp);
17 for(rank[sa[0]]=id=0,i=1;i<n;++i)
18 rank[sa[i]]=id+=sac(tmp,sa[i-1],sa[i])
    ;
19 A=id+1;
20 }
21 }
22 //h:高度數組 sa:後綴數組 rank:排名
23 void suffix_array_lcp(const char *s,int len,
    int *h,int *sa,int *rank){
24 for(int i=0;i<len;++i)rank[sa[i]]=i;
25 for(int i=0,k=0;i<len;++i){
26 if(rank[i]==0)continue;
27 if(k)--k;
28 while(s[i+k]==s[sa[rank[i]-1]+k])++k;
29 h[rank[i]]=k;
30 }
31 h[0]=0;
32 }

```

10.7 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[l]?z[i-l]:r-i
    +1);
6 while(i+z[i]<len&&s[i+z[i]]==s[z[i]])++z
    [i];
7 if(i+z[i]-1>r)r=i+z[i]-1,l=i;
8 }
9 }

```

11 Tarjan

11.1 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];
5 int fa[MAXN],dfn[MAXN],id[MAXN],Time;
6 int semi[MAXN],idom[MAXN];
7 int anc[MAXN],best[MAXN]; /*disjoint set
8 vector<int> dom[MAXN]; /*dominator_tree
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
28     [x]=best[anc[x]];
29     return anc[x]=y;
30 }
31 void tarjan(int r){
32     Time=0;
33     for(int t=1;t<=n;++t){
34         dfn[t]=idom[t]=0;//u=r或是u無法到達r時
35         idom[id[u]]=0
36         dom[t].clear();
37         anc[t]=best[t]=semi[t]=t;
38     }
39     dfs(r);
40     for(int y=Time;y>2;--y){
41         int x=fa[y],idy=id[y];
42         for(auto z:pre[idy]){
43             if(!(z=dfn[z]))continue;
44             find(z);
45             semi[y]=min(semi[y],semi[best[z]]);
46         }
47         dom[semi[y]].push_back(y);
48         anc[y]=x;
49         for(auto z:dom[x]){
50             find(z);
51             idom[z]=semi[best[z]]<x?best[z]:x;
52         }
53         dom[x].clear();
54     }
55     for(int u=2;u<=Time;++u){
56         if(idom[u]!=semi[u])idom[u]=idom[idom[
57             u]];
58         dom[id[idom[u]]].push_back(id[u]);
59     }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }

```

11.2 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 }

```

11.3 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
6     (0){}
7 };
8 vector<edge> E;
9 vector<int> G[N];// 1-base
10 int low[N],vis[N],Time;
11 int bcc_id[N],bridge_cnt,bcc_cnt;// 1-base
12 int st[N],top;//BCC用
13 inline void add_edge(int u,int v){
14     G[u].push_back(E.size());
15     E.push_back(edge(u,v));
16     G[v].push_back(E.size());
17     E.push_back(edge(v,u));
18 }
19 void dfs(int u,int re=-1){//u當前點·re為u連
20     接前一個點的邊
21     int v;
22     low[u]=vis[u]=++Time;
23     st[top++]=u;
24     for(size_t i=0;i<G[u].size();++i){
25         int e=G[u][i],v=E[e].v;
26         if(!vis[v]){
27             dfs(v,e^1);//e^1反向邊
28             low[u]=min(low[u],low[v]);
29             if(vis[u]<low[v]){
30                 E[e].is_bridge=E[e^1].is_bridge=1;
31                 ++bridge_cnt;
32             }
33         }else if(vis[v]<vis[u]&&e!=re)
34             low[u]=min(low[u],vis[v]);
35     }
36     if(vis[u]==low[u]){//處理BCC
37         ++bcc_cnt;// 1-base
38         do bcc_id[v=st[--top]]=bcc_cnt;//每個點
39         所在的BCC
40         while(v!=u);
41     }
42 }
43 inline void bcc_init(int n){
44     Time=bcc_cnt=bridge_cnt=top=0;
45     E.clear();
46     for(int i=1;i<=n;++i){
47         G[i].clear();
48         vis[i]=bcc_id[i]=0;
49     }
50 }

```

11.4 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

```

```

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
31         的根要特判
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     }

```

12 Tree_problem

12.1 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[
5     MAXN];
6 int link_top[MAXN],link[MAXN],cnt;
7 std::vector<int> G[MAXN];
8 void find_max_son(int x){
9     siz[x]=1;
10     max_son[x]=-1;
11     for(VIT i=G[x].begin();i!=G[x].end();++i){
12         if(*i==pa[x])continue;
13         pa[*i]=x;
14         dep[*i]=dep[x]+1;
15         find_max_son(*i);
16         if(max_son[x]==-1||siz[*i]>siz[max_son[x]
17             ])max_son[x]=*i;
18         siz[x]+=siz[*i];
19     }
20 }
21 void build_link(int x,int top){
22     link[x]=++cnt;

```

```

21 link_top[x]=top;
22 if(max_son[x]==-1)return;
23 build_link(max_son[x],top);
24 for(VIT i=G[x].begin();i!=G[x].end();++i){
25     if(*i==max_son[x]||*i==pa[x])continue;
26     build_link(*i,*i);
27 }
28 }
29 inline int find_lca(int a,int b){
30     //求LCA，可以在過程中對區間進行處理
31     int ta=link_top[a],tb=link_top[b];
32     while(ta!=tb){
33         if(dep[ta]<dep[tb]){
34             std::swap(ta,tb);
35             std::swap(a,b);
36         }
37         //這裡可以對a所在的鏈做區間處理
38         //區間為(Link[ta],Link[a])
39         ta=link_top[a=pa[ta]];
40     }
41     //最後a,b會在同一條鏈，若a!=b還要在進行一次區間處理
42     return dep[a]<dep[b]?a:b;
43 }

```

12.2 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 }

```

12.3 link_cut_tree.cpp

```

1 struct splay_tree{
2     int ch[2],pa; //子節點跟父母
3     bool rev; //反轉的懶惰標記
4     splay_tree():pa(0),rev(0){ch[0]=ch[1]=0;}
5 };
6 vector<splay_tree> node;
7 //有的時候用vector會TLE，要注意
8 //這邊以node[0]作為null節點
9 bool isroot(int x){//判斷是否為這棵splay tree的根
10     return node[node[x].pa].ch[0]!=x&&node[node[x].pa].ch[1]!=x;
11 }
12 void down(int x){//懶惰標記下推
13     if(node[x].rev){
14         if(node[x].ch[0])node[node[x].ch[0]].rev^=1;
15         if(node[x].ch[1])node[node[x].ch[1]].rev^=1;
16         std::swap(node[x].ch[0],node[x].ch[1]);
17         node[x].rev^=1;
18     }
19 }
20 void push_down(int x){//將所有祖先的懶惰標記下推
21     if(!isroot(x))push_down(node[x].pa);
22     down(x);
23 }
24 void up(int x){//將子節點的資訊向上更新
25 void rotate(int x){//旋轉，會自行判斷轉的方向
26     int y=node[x].pa,z=node[y].pa,d=(node[y].ch[1]==x);
27     node[x].pa=z;
28     if(!isroot(y))node[z].ch[node[z].ch[1]==y]=x;
29     node[y].ch[d]=node[x].ch[d^1];
30     node[node[y].ch[d]].pa=y;
31     node[y].pa=x,node[x].ch[d^1]=y;
32     up(y),up(x);
33 }
34 void splay(int x){//將節點x伸展到所在splay tree的根
35     push_down(x);
36     while(!isroot(x)){
37         int y=node[x].pa;
38         if(!isroot(y)){
39             int z=node[y].pa;
40             if((node[z].ch[0]==y)^(node[y].ch[0]==x))rotate(y);
41             else rotate(x);
42         }
43         rotate(x);
44     }
45 }
46 int access(int x){
47     int last=0;
48     while(x){
49         splay(x);
50         node[x].ch[1]=last;
51         up(x);

```

```

52     last=x;
53     x=node[x].pa;
54 }
55 return last; //回傳access後splay tree的根
56 }
57 void access(int x,bool is=0){//is=0就是一種的access
58     int last=0;
59     while(x){
60         splay(x);
61         if(is&&!node[x].pa){
62             //printf("%d\n",max(node[last].ma,node[x].ch[1].ma));
63         }
64         node[x].ch[1]=last;
65         up(x);
66         last=x;
67         x=node[x].pa;
68     }
69 }
70 void query_edge(int u,int v){
71     access(u);
72     access(v,1);
73 }
74 void make_root(int x){
75     access(x),splay(x);
76     node[x].rev^=1;
77 }
78 void make_root(int x){
79     node[access(x)].rev^=1;
80     splay(x);
81 }
82 void cut(int x,int y){
83     make_root(x);
84     access(y);
85     splay(y);
86     node[y].ch[0]=0;
87     node[x].pa=0;
88 }
89 void cut_parents(int x){
90     access(x);
91     splay(x);
92     node[node[x].ch[0]].pa=0;
93     node[x].ch[0]=0;
94 }
95 void link(int x,int y){
96     make_root(x);
97     node[x].pa=y;
98 }
99 int find_root(int x){
100     x=access(x);
101     while(node[x].ch[0])x=node[x].ch[0];
102     splay(x);
103     return x;
104 }
105 int query(int u,int v){
106     //傳回uv路徑splay tree的根結點
107     //這種寫法無法求LCA
108     make_root(u);
109     return access(v);
110 }
111 int query_lca(int u,int v){
112     //假設求鏈上點權的總和，sum是子樹的權重和，data是節點的權重

```

```

113     access(u);
114     int lca=access(v);
115     splay(u);
116     if(u==lca){
117         //return node[lca].data+node[node[lca].ch[1]].sum
118     }else{
119         //return node[lca].data+node[node[lca].ch[1]].sum+node[u].sum
120     }
121 }
122 struct EDGE{
123     int a,b,w;
124 }e[10005];
125 int n;
126 vector<pair<int,int>>G[10005];
127 //first表示子節點，second表示邊的編號
128 int pa[10005],edge_node[10005];
129 //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
130 void bfs(int root){
131     //在建構的時候把每個點都設成一個splay tree，不會壞掉
132     queue<int> q;
133     for(int i=1;i<=n;++i)pa[i]=0;
134     q.push(root);
135     while(q.size()){
136         int u=q.front();
137         q.pop();
138         for(int i=0;i<(int)G[u].size();++i){
139             int v=G[u][i].first;
140             if(v!=pa[u]){
141                 pa[v]=u;
142                 node[v].pa=u;
143                 node[v].data=e[G[u][i].second].w;
144                 edge_node[G[u][i].second]=v;
145                 up(v);
146                 q.push(v);
147             }
148         }
149     }
150 }
151 void change(int x,int b){
152     splay(x);
153     //node[x].data=b;
154     up(x);
155 }

```

12.4 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,
    int d){
15     dis.push_back(d);
16     for(size_t i=0;i<g[u].size();++i){
17         int v=g[u][i].first,w=g[u][i].second;
18         if(v!=pa&&vis[v])get_dis(dis,v,u,d+w);
19     }
20 }
21 vector<int> dis;//這東西如果放在函數裡會 TLE
22 int cal(int u,int d){
23     dis.clear();
24     get_dis(dis,u,-1,d);
25     sort(dis.begin(),dis.end());
26     int l=0,r=dis.size()-1,res=0;
27     while(l<r){
28         while(l<r&&dis[l]+dis[r]>k)--r;
29         res+=r-(l++);
30     }
31     return res;
32 }
33 pair<int,int> tree_centroid(int u,int pa,
    const int sz){
34     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].second;
53         if(vis[v])continue;
54         ans-=cal(v,w);
55         ans+=tree_DC(v,size[v]);
56     }
57     return ans;
58 }
59 int main(){
60     while(scanf("%d%d",&n,&k),n||k){
61         init();
62         for(int i=1;i<n;++i){
63             int u,v,w;
64             scanf("%d%d%d",&u,&v,&w);
65             g[u].push_back(make_pair(v,w));
66             g[v].push_back(make_pair(u,w));
67         }
68         printf("%d\n",tree_DC(1,n));
69     }
70     return 0;
71 }

```

13 zformula

13.1 formula.tex

13.1.1 Pick 公式

給定頂點坐標均是整點的簡單多邊形，面積 = 內部格點數 + 邊上格點數/2-1

13.1.2 圖論

- $V - E + F = 2$
- 對於平面圖， $F = E - V + n + 1$ ， n 是連通分量
- 對於平面圖， $E \leq 3V - 6$
- 對於連通圖 G ，最大獨立點集的大小設為 $I(G)$ ，最大匹配大小設為 $M(G)$ ，最小點覆蓋設為 $C_v(G)$ ，最小邊覆蓋設為 $C_e(G)$ 。對於任意連通圖：

- $I(G) + C_v(G) = |V|$
- $M(G) + C_e(G) = |V|$

- 對於連通二分圖：

- $I(G) = C_v(G)$
- $M(G) = C_e(G)$

- 最大權閉合圖：

- $C(u, V) = \infty, (u, v) \in E$
- $C(S, v) = W_v, W_v > 0$
- $C(v, T) = -W_v, W_v < 0$

- 最大密度子圖：

- $C(u, v) = 1, (u, v) \in E$
- $C(S, v) = U_v, v \in V$
- $C(v, T) = U + 2g - d_v, v \in V$

- 弦圖：

- 完美消除序列從後往前依次給每個點染色，給每個點染上可以染的最小顏色
- 最大團大小 = 色數
- 最大獨立集：完美消除序列從前往後能選就選
- 最小團覆蓋：最大獨立集的點和他延伸的邊構成
- 區間圖是弦圖
- 區間圖的完美消除序列：將區間按造又端點由小到大排序
- 區間圖染色：用線段樹做

```

1 double l=0,m,stop=1.0/n/n;
2 while(r-l>=stop){
3     double(mid);
4     if((n*m-sol.maxFlow(s,t))/2>eps)l=mid;
5     else r=mid;
6 }
7 build(1);
8 sol.maxFlow(s,t);
9 vector<int> ans;
10 for(int i=1;i<=n;++i)
11     if(sol.vis[i])ans.push_back(i);

```

13.1.3 學長公式

- $\sum_{d|n} \phi(n) = n$
- $g(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d) * g(n/d)$
- $Harmonicseries H_n = \ln(n) + \gamma + 1/(2n) - 1/(12n^2) + 1/(120n^4)$
- $\gamma = 0.5772156649015328606061209008240243104215$
- 格雷碼 $= n \oplus (n >> 1)$
- $SG(A + B) = SG(A) \oplus SG(B)$
- 選轉矩陣 $M(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

13.1.4 基本數論

- $\sum_{d|n} \mu(n) = (n == 1)$
- $g(m) = \sum_{d|m} f(d) \Leftrightarrow f(m) = \sum_{d|m} \mu(d) * g(m/d)$
- $\sum_{i=1}^n \sum_{j=1}^m \text{互質數量} = \sum \mu(d) \lfloor \frac{n}{d} \rfloor \lfloor \frac{m}{d} \rfloor$
- $\sum_{i=1}^n \sum_{j=1}^m lcm(i, j) = n \sum_{d|n} d \phi(d)$

13.1.5 排組公式

- k 卡特蘭 $\frac{C_n^{kn}}{n(k-1)+1} \cdot C_m^n = \frac{n!}{m!(n-m)!}$
- $H(n, m) \cong x_1 + x_2 \dots + x_n = k, num = C_k^{n+k-1}$
- Stirling number of 2^{nd} , n 入分 k 組方法數目
 - $S(0, 0) = S(n, n) = 1$
 - $S(n, 0) = 0$
 - $S(n, k) = kS(n-1, k) + S(n-1, k-1)$

- Bell number, n 入分任意多組方法數目

- $B_0 = 1$
- $B_n = \sum_{i=0}^n S(n, i)$
- $B_{n+1} = \sum_{k=0}^n C_k^n B_k$
- $B_{p+n} \equiv B_n + B_{n+1} \pmod{p}$, p is prime
- $B_p^{m+n} \equiv mB_n + B_{n+1} \pmod{p}$, p is prime
- From D0:1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975

- Derangement, 錯排，沒有人在自己位置上

- $D_n = n!(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} \dots + (-1)^n \frac{1}{n!})$
- $D_n = (n-1)(D_{n-1} + D_{n-2}), D_0 = 1, D_1 = 0$
- From D0:1, 0, 1, 2, 9, 44, 265, 1854, 14833, 133496

13.1.6 冪次，冪次和

- $a^b \% P = a^{b \% \varphi(P) + \varphi(P)}, b \geq \varphi(P)$
- $1^3 + 2^3 + 3^3 + \dots + n^3 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$
- $1^4 + 2^4 + 3^4 + \dots + n^4 = \frac{n^5}{5} + \frac{n^4}{2} + \frac{n^3}{3} - \frac{n}{30}$
- $1^5 + 2^5 + 3^5 + \dots + n^5 = \frac{n^6}{6} + \frac{n^5}{2} + \frac{5n^4}{12} - \frac{n^2}{12}$
- $0^k + 1^k + 2^k + \dots + n^k = P(k), P(k) = \frac{(n+1)^{k+1} - \sum_{i=0}^{k-1} C_i^{k+1} P(i)}{k+1}, P(0) = n + 1$
- $\sum_{k=0}^{m-1} k^n = \frac{1}{n+1} \sum_{k=0}^n C_k^{n+1} B_k m^{n+1-k}$
- $\sum_{j=0}^m C_j^{m+1} B_j = 0, B_0 = 1$
- 除了 $B_1 = -1/2$ ，剩下的奇數項都是 0
- $B_2 = 1/6, B_4 = -1/30, B_6 = 1/42, B_8 = -1/30, B_{10} = 5/66, B_{12} = -691/2730, B_{14} = 7/6, B_{16} = -3617/510, B_{18} = 43867/798, B_{20} = -174611/330,$

13.1.7 Burnside's lemma

- $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$
- $X^g = t^{c(g)}$
- G 表示有幾種轉法， X^g 表示在那種轉法下，有幾種是會保持對稱的， t 是顏色數， $c(g)$ 是循環節不動的面數。
- 正立方體塗三顏色，轉 0 有 3^6 個元素不變，轉 90 有 6 種，每種有 3^3 不變，180 有 3×3^4 ，120(角) 有 $8 \times 3^2 \cdot 180$ (邊) 有 6×3^3 ，全部 $\frac{1}{24}(3^6 + 6 \times 3^3 + 3 \times 3^4 + 8 \times 3^2 + 6 \times 3^3) = 57$

13.1.8 Count on a tree

- Rooted tree: $s_{n+1} = \frac{1}{n} \sum_{i=1}^n (i \times a_i \times \sum_{j=1}^{\lfloor n/i \rfloor} a_{n+1-i \times j})$
- Unrooted tree:
 - Odd: $a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$
 - Even: $Odd + \frac{1}{2} a_{n/2} (a_{n/2} + 1)$
- Spanning Tree

- 完全圖 $n^n - 2$
- 一般圖 (Kirchhoff's theorem) $M[i][i] = \text{degree}(V_i), M[i][j] = -1, \text{if have } E(i, j), 0 \text{ if no edge. delete any one row and col in } A, \text{ans} = \det(A)$

13.2 java.tex

13.2.1 文件操作

```

1 import java.io.*;
2 import java.util.*;
3 import java.math.*;
4 import java.text.*;
5
6 public class Main
7 {
8
9     public static void main(String args[])
10        throws FileNotFoundException,
11        IOException
12    {
13        Scanner sc = new Scanner(new FileReader(
14            "a.in"));
15        PrintWriter pw = new PrintWriter(new
16            FileWriter("a.out"));
17        int n,m;
18        n=sc.nextInt();//读入下一个INT
19        m=sc.nextInt();
20
21        for(ci=1; ci<=c; ++ci)
22        {
23            pw.println("Case #"+ci+": easy for
24                output");
25        }
26
27        pw.close();//关闭流并释放，这个很重要，
28            否则是没有输出的
29        sc.close();//关闭流并释放
30    }
31 }

```

```

5 for(Object obj : map.keySet()){
6     Object value = map.get(obj );
7 }

```

13.2.4 sort

```

1 static class cmp implements Comparator
2 {
3     public int compare(Object o1,Object o2)
4     {
5         BigInteger b1=(BigInteger)o1;
6         BigInteger b2=(BigInteger)o2;
7         return b1.compareTo(b2);
8     }
9 }
10 public static void main(String[] args)
11    throws IOException
12 {
13     Scanner cin = new Scanner(System.in);
14     int n;
15     n=cin.nextInt();
16     BigInteger[] seg = new BigInteger[n];
17     for (int i=0;i<n;i++)
18         seg[i]=cin.nextBigInteger();
19     Arrays.sort(seg,new cmp());
20 }

```

13.2.2 优先队列

```

1 PriorityQueue queue = new PriorityQueue( 1,
2     new Comparator()
3 {
4     public int compare( Point a, Point b )
5     {
6         if( a.x < b.x || a.x == b.x && a.y < b.y )
7             return -1;
8         else if( a.x == b.x && a.y == b.y )
9             return 0;
10        else
11            return 1;
12    }
13 });

```

13.2.3 Map

```

1 Map map = new HashMap();
2 map.put("sa","dd");
3 String str = map.get("sa").toString;
4

```


ACM ICPC TEAM REFERENCE - NTHU JINKELA

Contents

1 Computational_Geometry	1	2.7 split_merge.cpp	6	5.15 最小樹形圖 _ 朱劉.cpp	11	10 String	15
1.1 Geometry.cpp	1	2.8 treap.cpp	6	5.16 穩定婚姻模板.cpp	11	10.1 AC 自動機.cpp	15
1.2 SmallestCircle.cpp	3	2.9 操作分治.cpp	6			10.2 hash.cpp	16
1.3 最近點對.cpp	3	2.10 整體二分.cpp	6	6 language	11	10.3 KMP.cpp	16
1.4 浮點數誤差模板.cpp	3			6.1 CNF.cpp	11	10.4 manacher.cpp	16
2 Data_Structure	3	3 default	7	7 Linear_Programming	11	10.5 minimal_string_rotation.cpp	16
2.1 DLX.cpp	3	3.1 debug.cpp	7	7.1 最大密度子圖.cpp	11	10.6 suffix_array_lcp.cpp	16
2.2 Dynamic_KD_tree.cpp	4	3.2 ext.cpp	7			10.7 Z.cpp	16
2.3 kd_tree_replace_segment_tree.cpp	5	3.3 IncStack.cpp	7	8 Number_Theory	12		
2.4 persistent_segment_tree.cpp	5	3.4 input.cpp	7	8.1 basic.cpp	12	11 Tarjan	16
2.5 reference_point.cpp	6			8.2 bit_set.cpp	13	11.1 dominator_tree.cpp	16
2.6 skew_heap.cpp	6	4 Flow	7	8.3 cantor_expansion.cpp	13	11.2 tnfsb017_2_sat.cpp	17
		4.1 dinic.cpp	7	8.4 FFT.cpp	13	11.3 橋連通分量.cpp	17
		4.2 ISAP_with_cut.cpp	7	8.5 find_real_root.cpp	13	11.4 雙連通分量 & 割點.cpp	17
		4.3 MinCostMaxFlow.cpp	8	8.6 LinearCongruence.cpp	13	12 Tree_problem	17
				8.7 Lucas.cpp	13	12.1 HeavyLight.cpp	17
		5 Graph	8	8.8 Matrix.cpp	13	12.2 LCA.cpp	18
		5.1 Augmenting_Path.cpp	8	8.9 MillerRobin.cpp	14	12.3 link_cut_tree.cpp	18
		5.2 Augmenting_Path_multiple.cpp	8	8.10 NTT.cpp	14	12.4 POJ_tree.cpp	18
		5.3 blossom_matching.cpp	8	8.11 Simpson.cpp	14		
		5.4 graphISO.cpp	8	8.12 外星模運算.cpp	14	13 zformula	19
		5.5 KM.cpp	9	8.13 模運算模板.cpp	15	13.1 formula.tex	19
		5.6 MaximumClique.cpp	9	8.14 質因數分解.cpp	15	13.1.1 Pick 公式	19
		5.7 MinimumMeanCycle.cpp	9			13.1.2 圖論	19
		5.8 Minimum_General_Weighted_Matching.cpp	9	9 other	15	13.1.3 學長公式	19
		5.9 Rectilinear_Steiner_tree.cpp	10	9.1 WhatDay.cpp	15	13.1.4 基本數論	19
		5.10 treeISO.cpp	10	9.2 上下最大正方形.cpp	15	13.1.5 排組公式	19
		5.11 全局最小割.cpp	10	9.3 最大矩形.cpp	15	13.1.6 冪次, 冪次和	19
		5.12 平面圖判定.cpp	10			13.1.7 Burnside's lemma	19
		5.13 弦圖完美消除序列.cpp	10			13.1.8 Count on a tree	19
		5.14 最小斯坦納樹 DP.cpp	11				