

1 Computational_Geometry

1.1 Geometry.cpp

```

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

```

```

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

```

```

408     }
409   }
410   size_t sz=0;
411   for(size_t i=0;i<fc.size();++i)if(fc[i].
412       use)fc[sz++]=fc[i];
413   fc.resize(sz);
414   point3D<T> centroid(const{
415       point3D<T> res(0,0,0);
416       T vol=0;
417       for(size_t i=0;i<fc.size();++i){
418           T tmp=pt[fc[i].a].dot(pt[fc[i].b].
419               cross(pt[fc[i].c]));
420           res=res+(pt[fc[i].a]+pt[fc[i].b]+pt[fc
421               [i].c])*tmp;
422           vol+=tmp;
423       }
424       return res/(vol*4);
425   }
426 };

```

```

31   if(n==2) return TwoPointCircle(p[0],p
32       [1]);
33   random_shuffle(p.begin(),p.end());
34   Circle c = {p[0],0.0};
35   for(int i=0;i<n;++i){
36       if(c.incircle(p[i])) continue;
37       c=Circle{p[i],0.0};
38       for(int j=0;j<i;++j){
39           if(c.incircle(p[j])) continue;
40           c=TwoPointCircle(p[i],p[j]);
41           for(int k=0;k<j;++k){
42               if(c.incircle(p[k]))
43                   continue;
44               c=outCircle(p[i],p[j],p[k]);
45           }
46       }
47   }

```

```

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 };

```

```

32   void restore(int c){//恢復第c行和所有當前
33       覆蓋到第c行的列·remove的逆操作
34       DFOR(i,U,c)DFOR(j,L,i){++S[col[j]],U[D[j]]
35           =j,D[U[j]]=j;}
36       L[R[c]]=c,R[L[c]]=c;}
37   void remove2(int nd){//刪除nd所在的行當前
38       所有點(包括虛擬點)·只保留nd
39       DFOR(i,D,nd)L[R[i]]=L[i],R[L[i]]=R[i];
40   }
41   void restore2(int nd){//刪除nd所在的行當前
42       所有點·為remove2的逆操作
43       DFOR(i,U,nd)L[R[i]]=R[L[i]]=i;
44   }
45   bool vis[MAXM];
46   int h(){//估價函數 for IDA*
47       int res=0;
48       memset(vis,0,sizeof(vis));
49       DFOR(i,R,0)if(!vis[i]){
50           vis[i]=1;
51           ++res;
52           DFOR(j,D,i)DFOR(k,R,j)vis[col[k]]=1;
53       }
54       return res;
55   }
56   bool dfs(int d){//for精確覆蓋問題
57       if(d+h()>=ansd)return 0;//找最佳解用·找
58       任意解可以刪掉
59       if(!R[0]){ansd=d;return 1;}
60       int c=R[0];
61       DFOR(i,R,0)if(S[i]<S[c])c=i;
62       remove(c);
63       DFOR(i,D,c){
64           ans.push_back(row[i]);
65           DFOR(j,R,i)remove2(col[j]);
66           if(dfs(d+1))return 1;
67           ans.pop_back();
68           DFOR(j,L,i)restore2(col[j]);
69       }
70       restore(c);
71       return 0;
72   }
73   void dfs2(int d){//for最小重複覆蓋問題
74       if(d+h()>=ansd)return;
75       if(!R[0]){ansd=d;ans=ans;return;}
76       int c=R[0];
77       DFOR(i,R,0)if(S[i]<S[c])c=i;
78       DFOR(i,D,c){
79           ans.push_back(row[i]);
80           remove2(i);
81           DFOR(j,R,i)remove2(j),--S[col[j]];
82           dfs2(d+1);
83           ans.pop_back();
84           DFOR(j,L,i)restore2(j),++S[col[j]];
85       }
86       restore2(i);
87   }
88   bool exact_cover(){//解精確覆蓋問題
89       ans.clear();
90       return dfs(0);
91   }
92   void min_cover(){//解最小重複覆蓋問題
93       ans.clear();
94       //暫存用·答案還是存在ans裡
95   }

```

1.3 最近點對.cpp

```

1   #define INF LLONG_MAX/*預設是Long Long最大值
2   */
3   template<typename T>
4   T closest_pair(vector<point<T> >&v,vector<
5       point<T> >&t,int l,int r){
6       T dis=INF,tmd;
7       if(l==r)return dis;
8       int mid=(l+r)/2;
9       if((tmd=closest_pair(v,t,l,mid))<dis)dis=
10          tmd;
11       if((tmd=closest_pair(v,t,mid+1,r))<dis)dis=
12          tmd;
13       t.clear();
14       for(int i=l;i<=r;++i)
15           if((v[i].x-v[mid].x)*(v[i].x-v[mid].x)<
16               dis)t.push_back(v[i]);
17       sort(t.begin(),t.end(),point<T>::y_cmp);/*
18          如果用merge_sort的方式可以O(n)*/
19       for(int i=0;i<(int)t.size();++i)
20           for(int j=1;j<=3&&i+j<(int)t.size();++j)
21               if((tmd=(t[i]-t[i+j]).abs2())<dis)dis=
22                   tmd;
23       return dis;
24   }
25   template<typename T>
26   inline T closest_pair(vector<point<T> >&v){
27       vector<point<T> >t;
28       sort(v.begin(),v.end(),point<T>::x_cmp);
29       return closest_pair(v,t,0,v.size()-1);/*最
30          近點對距離*/
31   }

```

1.4 浮點數誤差模板.cpp

```

1   const double EPS=1e-9;
2   struct Double{
3       double d;

```

2 Data_Structure

2.1 DLX.cpp

```

1   #define MAXN 4100
2   #define MAXM 1030
3   #define MAXND 16390
4   struct DLX{
5       int n,m,sz,ansd;//高是n·寬是m的稀疏矩陣
6       int S[MAXN],H[MAXN];
7       int row[MAXN],col[MAXN];/*每個節點代表的
8          列跟行
9       int L[MAXN],R[MAXN],U[MAXN],D[MAXN];
10      vector<int> ans,anst;
11      void init(int _n,int _m){
12          n=_n,m=_m;
13          for(int i=0;i<=m;++i){
14              U[i]=D[i]=i,L[i]=i-1,R[i]=i+1;
15              S[i]=0;
16          }
17          R[m]=0,L[0]=m;
18          sz=m,ansd=INT_MAX;/*ansd存最優解的個數
19          for(int i=1;i<=n;++i)H[i]=-1;
20      }
21      void add(int r,int c){
22          ++S[col[++sz]]=c;
23          row[sz]=r;
24          D[sz]=D[c],U[D[c]]=sz,U[sz]=c,D[c]=sz;
25          if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
26          else R[sz]=R[H[r]],L[R[H[r]]]=sz,L[sz]=H
27              [r],R[H[r]]=sz;
28      }
29      #define DFOR(i,A,s) for(int i=A[i];i!=s;i=
30          A[i])
31      void remove(int c){//刪除第c行和所有當前覆
32          蓋到第c行的列
33          L[R[c]]=L[c],R[L[c]]=R[c];/*這裡刪除第c
34          行·若有些行不需要處理可以在開始時呼
35          叫他
36          DFOR(i,D,c)DFOR(j,R,i){U[D[j]]=U[j],D[U[
37              j]]=D[j],--S[col[j]];}
38      }

```

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).
8          abs2()<=r2;}
9   };
10  Circle TwoPointCircle(Circle::cp &a, Circle
11      ::cp &b) {
12      Circle::p m=(a+b)/2;
13      return (Circle){m,(a-m).abs2()};
14  }
15  Circle outCircle(Circle::p a, Circle::p b,
16      Circle::p c) {
17      if(TwoPointCircle(a,b).incircle(c))
18          return TwoPointCircle(a,b);
19      if(TwoPointCircle(b,c).incircle(a))
20          return TwoPointCircle(b,c);
21      if(TwoPointCircle(c,a).incircle(b))
22          return TwoPointCircle(c,a);
23      Circle::p ret;
24      double a1=b.x-a.x, b1=b.y-a.y, c1=(a1*a1
25          +b1*b1)/2;
26      double a2=c.x-a.x, b2=c.y-a.y, c2=(a2*a2
27          +b2*b2)/2;
28      double d = a1*b2 - a2*b1;
29      ret.x=a.x+(c1*b2-c2*b1)/d;
30      ret.y=a.y+(a1*c2-a2*c1)/d;
31      return (Circle){ret,(ret-a).abs2()};
32  }
33  //rand required
34  Circle SmallestCircle(std::vector<Circle::p>
35      &p){
36      int n=p.size();
37      if(n==1) return (Circle){p[0],0.0};

```

```

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::
9                  abs(d[i]-x.d[i]);
10             return ret;
11         }
12         bool operator==(const point &p){
13             for(size_t i=0;i<kd;++i)
14                 if(d[i]!=p.d[i])return 0;
15             return 1;
16         }
17         bool operator<(const point &b)const{
18             return d[0]<b.d[0];
19         }
20     };
21     private:
22     struct node{
23         node *l,*r;
24         point pid;
25         int s;
26         node(const point &p):l(0),r(0),pid(p),
27             s(1){}
28         ~node(){delete l;delete r;}
29         void up(){s=(l?l->s:0)+1+(r?r->s:0);}
30     }*root;
31     const double alpha,loga;
32     const T INF;記得要給INF，表示極大值
33     int maxn;
34     struct __cmp{
35         int sort_id;
36         bool operator()(const node*x,const
37             node*y)const{
38             return operator()(x->pid,y->pid);
39         }
40         bool operator()(const point &x,const
41             point &y)const{
42             if(x.d[sort_id]!=y.d[sort_id])
43                 return x.d[sort_id]<y.d[sort_id];
44             for(size_t i=0;i<kd;++i)
45                 if(x.d[i]!=y.d[i])return x.d[i]<y.
46                     d[i];
47             return 0;
48         }
49     }cmp;
50     int size(node *o){return o?o->s:0;}
51     std::vector<node*> A;
52     node* build(int k,int l,int r){
53         if(l>r)return 0;
54         if(k==kd)k=0;
55         int mid=(l+r)/2;

```

```

56         cmp.sort_id=k;
57         std::nth_element(A.begin()+l,A.begin()
58             +mid,A.begin()+r+1,cmp);
59         node *ret=A[mid];
60         ret->l=build(k+1,l,mid-1);
61         ret->r=build(k+1,mid+1,r);
62         ret->up();
63         return ret;
64     }
65     bool isbad(node*o){
66         return size(o->l)>alpha*o->s||size(o->
67             r)>alpha*o->s;
68     }
69     void flatten(node *u,typename std::
70         vector<node*>::iterator &it){
71         if(!u)return;
72         flatten(u->l,it);
73         *it=u;
74         flatten(u->r,++it);
75     }
76     void rebuild(node*&u,int k){
77         if((!u).A.size()<u->s).A.resize(u->s);
78         typename std::vector<node*>::iterator
79             it=A.begin();
80         flatten(u,it);
81         u=build(k,0,u->s-1);
82     }
83     bool insert(node*&u,int k,const point &x
84         ,int dep){
85         if(!u){
86             u=new node(x);
87             return dep<=0;
88         }
89         ++u->s;
90         cmp.sort_id=k;
91         if(insert(cmp(x,u->pid)?u->l:u->r,(k
92             +1)%kd,x,dep-1)){
93             if(!isbad(u))return 1;
94             rebuild(u,k);
95         }
96         return 0;
97     }
98     node *findmin(node*o,int k){
99         if(!o)return 0;
100         if(cmp.sort_id==k)return o->l?findmin(
101             o->l,(k+1)%kd):o;
102         node *l=findmin(o->l,(k+1)%kd);
103         node *r=findmin(o->r,(k+1)%kd);
104         if(l&l&r)return cmp(l,o)?l:o;
105         if(!l&r)return cmp(r,o)?r:o;
106         if(!l&l&r)return o;
107         if(cmp(l,r))return cmp(l,o)?l:o;
108         return cmp(r,o)?r:o;

```

```

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)
117     %kd,x)){
118     --u->s;return 1;
119 }else return 0;
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 >
128     >pQ;
129 void nearest(node *u,int k,const point &
130     x,T *h,T &mndist){
131     if(u==0||heuristic(h)>mndist)return;
132     T dist=u->pid.dist(x),old=h[k];
133     /*mndist=std::min(mndist,dist);*/
134     if(dist<mndist){
135         pQ.push(std::make_pair(dist,u->pid));
136         if((!pQ.size()==qM+1)
137             mndist=pQ.top().first,pQ.pop());
138     }
139     if(x.d[k]<u->pid.d[k]){
140         nearest(u->l,(k+1)%kd,x,h,mndist);
141         h[k]=std::abs(x.d[k]-u->pid.d[k]);
142         nearest(u->r,(k+1)%kd,x,h,mndist);
143     }else{
144         nearest(u->r,(k+1)%kd,x,h,mndist);
145         h[k]=std::abs(x.d[k]-u->pid.d[k]);
146         nearest(u->l,(k+1)%kd,x,h,mndist);
147     }
148     h[k]=old;
149 }
150 std::vector<point>in_range;
151 void range(node *u,int k,const point&mi,
152     const point&ma){
153     if(!u)return;
154     bool is=1;
155     for(int i=0;i<kd;++i)
156         if(u->pid.d[i]<mi.d[i]||ma.d[i]<u->
157             pid.d[i]){
158             is=0;break;
159         }
160     if(is)in_range.push_back(u->pid);
161     if(mi.d[k]<u->pid.d[k])range(u->l,(k
162         +1)%kd,mi,ma);
163     if(ma.d[k]>u->pid.d[k])range(u->r,(k
164         +1)%kd,mi,ma);

```

```

165     for(int i=0;i<n;++i)A[i]=new node(p[i
166         ]);
167     root=build(0,0,n-1);
168     void insert(const point &x){
169         insert(root,0,x,__lg(size(root))/loga
170             );
171         if(root->s>maxn)maxn=root->s;
172     }
173     bool erase(const point &p){
174         bool d=erase(root,0,p);
175         if(root&&root->s<alpha*maxn)rebuild();
176         return d;
177     }
178     void rebuild(){
179         if(root)rebuild(root,0);
180         maxn=root->s;
181     }
182     T nearest(const point &x,int k){
183         qM=k;
184         T mndist=INF,h[kd]={};
185         nearest(root,0,x,h,mndist);
186         mndist=pQ.top().first;
187         pQ=std::priority_queue<std::pair<T,
188             point >>();
189         return mndist;回傳離x第k近的點的距離
190     }
191     const std::vector<point> &range(const
192         point&mi,const point&ma){
193         in_range.clear();
194         range(root,0,mi,ma);
195         return in_range;回傳介於mi到ma之間的
196         點vector
197     }
198     int size(){return root?root->s:0;}
199 };

```

2.3 kd_tree_replace_segment_tree

```

1  /*kd樹代替高維線段樹*/
2  struct node{
3      node *l,*r;
4      point pid,mi,ma;
5      int s;
6      int data;
7      node(const point &p,int d):l(0),r(0),pid(p
8          ),mi(p),ma(p),s(1),data(d),dmin(d),
9          dmax(d){}
10     void up(){
11         mi=ma=pid;
12         s=1;
13         if(l){
14             for(int i=0;i<kd;++i){
15                 mi.d[i]=min(mi.d[i],l->mi.d[i]);
16                 ma.d[i]=max(ma.d[i],l->ma.d[i]);
17             }
18         }
19         if(r){
20             for(int i=0;i<kd;++i){
21                 mi.d[i]=min(mi.d[i],r->mi.d[i]);
22                 ma.d[i]=max(ma.d[i],r->ma.d[i]);

```



```

22     }
23     s+=r->s;
24 }
25 void up2(){
26     //其他懶惰標記向上更新
27 }
28 void down(){
29     //其他懶惰標記下推
30 }
31 }*root;
32
33 /*檢查區間包含用的函數*/
34 inline bool range_include(node *o,const
35     point &L,const point &R){
36     for(int i=0;i<kd;++i){
37         if(L.d[i]>o->ma.d[i]||R.d[i]<o->mi.d[i])
38             return 0;
39     }//只要(L,R)區間有和o的區間有交集就回傳
40     true
41     return 1;
42 }
43 inline bool range_in_range(node *o,const
44     point &L,const point &R){
45     for(int i=0;i<kd;++i){
46         if(L.d[i]>o->mi.d[i]||o->ma.d[i]>R.d[i])
47             return 0;
48     }//如果(L,R)區間完全包含o的區間就回傳true
49     return 1;
50 }
51 inline bool point_in_range(node *o,const
52     point &L,const point &R){
53     for(int i=0;i<kd;++i){
54         if(L.d[i]>o->pid.d[i]||R.d[i]<o->pid.d[i])
55             return 0;
56     }//如果(L,R)區間完全包含o->pid這個點就回傳
57     true
58     return 1;
59 }
60
61 /*單點修改 · 以單點改值為例*/
62 void update(node *u,const point &x,int data,
63     int k=0){
64     if(!u)return;
65     u->down();
66     if(u->pid==x){
67         u->data=data;
68         u->up2();
69         return;
70     }
71     cmp.sort_id=k;
72     update(cmp(x,u->pid)?u->l:u->r,x,data,(k
73         +1)%kd);
74     u->up2();
75 }
76
77 /*區間修改*/
78 void update(node *o,const point &L,const
79     point &R,int data){
80     if(!o)return;
81     o->down();
82     if(range_in_range(o,L,R)){
83         //區間懶惰標記修改
84         o->down();

```

```

75     return;
76 }
77 if(point_in_range(o,L,R)){
78     //這個點在(L,R)區間 · 但是他的左右子樹不
79     一定在區間中
80     //單點懶惰標記修改
81 }
82 if(o->l&&range_include(o->l,L,R))update(o
83     ->l,L,R,data);
84 if(o->r&&range_include(o->r,L,R))update(o
85     ->r,L,R,data);
86 o->up2();
87 }
88
89 /*區間查詢 · 以總和為例*/
90 int query(node *o,const point &L,const point
91     &R){
92     if(!o)return 0;
93     o->down();
94     if(range_in_range(o,L,R))return o->sum;
95     int ans=0;
96     if(point_in_range(o,L,R))ans+=o->data;
97     if(o->l&&range_include(o->l,L,R))ans+=
98         query(o->l,L,R);
99     if(o->r&&range_include(o->r,L,R))ans+=
100         query(o->r,L,R);
101     return ans;
102 }

```

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;

```

```

30 }
31 int insert(int l,int r,int rt,int x,int d){
32     if(x<l||r<x)return rt;
33     int nd=new_node(nds[rt]);
34     if(l==r&&l==x)nds[nd].data+=d;
35     else{
36         int mid=(l+r)/2;
37         int L=insert(l,mid,nds[nd].l,x,d);
38         int R=insert(mid+1,r,nds[nd].r,x,d);
39         nds[nd].l=L;
40         nds[nd].r=R;
41         up(nd,L,R);
42     }
43     return nd;
44 }
45 inline int cal(int L,int R){
46     return nds[R].data-nds[L].data;
47 }
48 int find(int l,int r,int L,int R,int k){
49     if(l==r)return l;
50     int mid=(l+r)/2;
51     int add=cal(nds[L].l,nds[R].l);
52     if(k<=add)return find(l,mid,nds[L].l,nds[R]
53         .l,k);
54     return find(mid+1,r,nds[L].r,nds[R].r,k-
55         add);
56 }
57 int n,m;
58 int s[100005];
59 int root[100005];
60 int main(){
61     while(~scanf("%d%d",&n,&m)){
62         nds.clear();
63         vector<int> lsh;
64         for(int i=1;i<=n;++i){
65             scanf("%d",&s[i]);
66             lsh.push_back(s[i]);
67         }
68         sort(lsh.begin(),lsh.end());
69         lsh.resize(unique(lsh.begin(),lsh.end())
70             -lsh.begin());
71         int N=(int)lsh.size()-1;
72         root[0]=build_tree(0,N);
73         for(int i=1;i<=n;++i){
74             s[i]=lower_bound(lsh.begin(),lsh.end()
75                 ,s[i])-lsh.begin();
76             root[i]=insert(0,N,root[i-1],s[i],1);
77         }
78         while(m--){
79             int a,b,k;
80             scanf("%d%d%d",&a,&b,&k);
81             int res=find(0,N,root[a-1],root[b],k);
82             printf("%d\n",lsh[res]);
83         }
84     }
85     return 0;
86 }

```

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(){
43     ref_pointer<int>b=new_ref(int(5));
44     ref_pointer<int>a=new_ref(*b);
45     ref_pointer<P>c=new_ref(p);
46     return 0;
47 }

```

2.6 skew_heap.cpp

```

1 template<typename T,typename _Compare=std::
2     less<T>>
3 class skew_heap{
4 private:
5     struct node{
6         T data;
7         node *l,*r;
8         node(const T&d):data(d),l(0),r(0){}
9         ~node(){delete l;delete r;}
10    }*root;
11    int _size;
12    _Compare cmp;
13    node *merge(node *a,node *b){
14        if(!a||!b)return a?a:b;
15        if(cmp(a->data,b->data))return merge(b
16            ,a);

```

```

15     node *t=a->r;
16     a->r=a->l;
17     a->l=merge(b,t);
18     return a;
19 }
20 public:
21     skew_heap():root(0),_size(0){}
22     ~skew_heap(){delete root;}
23     void clear(){delete root;root=0,_size
24     =0;}
25     void join(skew_heap &o){
26         root=merge(root,o.root);
27         o.root=0;
28         _size+=o._size;
29         o._size=0;
30     }
31     void swap(skew_heap &o){
32         node *t=root;
33         root=o.root;
34         o.root=t;
35         int st=_size;
36         _size=o._size;
37         o._size=st;
38     }
39     void push(const T&data){
40         _size++;
41         root=merge(root,new node(data));
42     }
43     void pop(){
44         if(_size)_size--;
45         node *tmd=merge(root->l,root->r);
46         root->l=root->r=0;
47         delete root;
48         root=tmd;
49     }
50     const T& top(){return root->data;}
51     int size(){return _size;}
52     bool empty(){return !_size;}
53 };

```

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
59        [1]),delete o;
60    }
61    public:
62    treap(unsigned s=20150119):nil(new node)
63    ,root(nil),x(s){}
64    ~treap(){clear(root),delete nil;}
65    void clear(){clear(root),root=nil;}
66    void insert(const T &data){
67        insert(root,data);
68    }
69    bool erase(const T &data){
70        return erase(root,data);
71    }
72    bool find(const T&data){
73        for(node *o=root;o->s;){
74            if(o->data==data)return 1;
75            else o=o->ch[o->data<data];
76            return 0;
77        }
78    }
79    int rank(const T&data){
80        int cnt=0;
81        for(node *o=root;o->s;){
82            if(o->data<data)cnt+=o->ch[0]->s+1,o=o
83            ->ch[1];
84            else o=o->ch[0];
85            return cnt;
86        }
87    }
88    const T&kth(int k){
89        for(node *o=root;;){
90            if(k<=o->ch[0]->s)o=o->ch[0];
91            else if(k==o->ch[0]->s+1)return o->
92            data;
93            else k-=o->ch[0]->s+1,o=o->ch[1];
94        }
95    }
96    const T&operator[](int k){
97        return kth(k);
98    }
99    const T&preorder(const T&data){
100        node *x=root,*y=0;
101        while(x->s){
102            if(x->data<data)y=x,x=x->ch[1];
103            else x=x->ch[0];
104            if(y)return y->data;
105            return data;
106        }
107    }
108    const T&successor(const T&data){
109        node *x=root,*y=0;
110        while(x->s){
111            if(data<x->data)y=x,x=x->ch[0];
112            else x=x->ch[1];
113            if(y)return y->data;
114            return data;
115        }
116    }
117    int size(){return root->s;}
118 };

```

2.9 操作分治.cpp

```

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

```

2.10 整體二分.cpp

```

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

```

3 default

3.1 debug.cpp

```

1 #ifndef Jinkala
2 #define debug(...) {\
3     fprintf(stderr,"%s - %d : (%s) = ",
4         __PRETTY_FUNCTION__,__LINE__,#
5         __VA_ARGS__); \
6         _DO(__VA_ARGS__); \
7 }
8 #else
9 #define debug(...)
10 #endif

```

3.2 ext.cpp

```

1 __gnu_pbds::tree<int,null_type,less<int>,
2     rb_tree_tag,
3     tree_order_statistics_node_update>

```

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=
4     getchar();
5     while('0'<=ch&&ch<='9')x=x*10-'0'+ch,ch=
6     getchar();
7     return f?-x:x;
8 }
9 //g++ -std=c++11 -O2 -Wall -Wextra -Wno-
  unused-variable $1 && ./a.out

```

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),
11        cap(cap), flow(0), r(cap){}
12    };
13    int g[MAXN];
14    vector<edge> e;
15    void init(int _n){
16        memset(g, -1, sizeof(int)*((n=_n)+1));
17        e.clear();
18    }
19    void add_edge(int u, int v, T cap, bool
20        directed=false){
21        e.push_back(edge(v, g[u], cap));
22    }
23 }

```

```

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]
49                 .r))){
50                 e[i].flow+=df;
51                 e[i^1].flow-=df;
52                 e[i].r-=df;
53                 e[i^1].r+=df;
54                 return df;
55             }
56         }
57     }
58     return level[u]=0;
59 }
60 T dinic(int s, int t, bool clean=true){
61     if(clean){
62         for(size_t i=0; i<e.size(); ++i){
63             e[i].flow=0;
64             e[i].r=e[i].cap;
65         }
66     }
67     T ans=0, mf=0;
68     while(bfs(s, t)) while(mf=dfs(s, t)) ans+=mf;
69     return ans;
70 }

```

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),
11    cap(cap), flow(0), r(cap){}
12 };
13 int g[MAXN];
14 vector<edge> e;
15 void init(int _n){
16     memset(g, -1, sizeof(int)*((n=_n)+1));
17     e.clear();
18 }
19 void add_edge(int u, int v, T cap, bool
20     directed=false){
21     e.push_back(edge(v, g[u], cap));
22     g[u]=e.size()-1;
23     e.push_back(edge(u, g[v], directed?0:cap));
24     ;
25     g[v]=e.size()-1;
26 }
27 T dfs(int u, int s, int t, T cur_flow=INF){
28     if(u==t) return cur_flow;
29     T tf=cur_flow, df;
30     for(int &i=cur[u]; ~i; i=e[i].pre){
31         if(e[i].r && d[u]==d[e[i].v]+1){
32             df=dfs(e[i].v, s, t, min(tf, e[i].r));
33             e[i].flow+=df;
34             e[i^1].flow-=df;
35             e[i].r-=df;
36             e[i^1].r+=df;
37             if(!((tf-=df)||d[s]==n)) return
38                 cur_flow-tf;
39         }
40     }
41     int mh=n;
42     for(int i=cur[u]=g[u]; ~i; i=e[i].pre){
43         if(e[i].r && d[e[i].v]<mh) mh=d[e[i].v];
44     }
45     if(!--gap[d[u]]) d[s]=n;
46     else ++gap[d[u]=++mh];
47     return cur_flow-tf;
48 }
49 T isap(int s, int t, bool clean=true){
50     memset(d, 0, sizeof(int)*(n+1));
51     memset(gap, 0, sizeof(int)*(n+1));
52     memcpy(cur, g, sizeof(int)*(n+1));
53     if(clean){
54         for(size_t i=0; i<e.size(); ++i){
55             e[i].flow=0;
56             e[i].r=e[i].cap;
57         }
58     }
59     T max_flow=0;
60     for(gap[0]=n; d[s]<n; ) max_flow+=dfs(s, s, t);
61     return max_flow;
62 }
63 vector<int> cut_e; //最小割邊集
64 bool vis[MAXN];
65 void dfs_cut(int u){
66     vis[u]=1; //表示u屬於source的最小割集
67     for(int i=g[u]; ~i; i=e[i].pre){
68         if(e[i].flow<e[i].cap && !vis[e[i].v])
69             dfs_cut(e[i].v);
70     }
71 }

```

```

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             ){
74             if(!vis[e[i].v]) cut_e.push_back(i);
75         }
76     }
77     return ans;
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;
47        dis[T]=0, q.push_back(T);
48    }

```

```

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.cpp

```

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;

```

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;
10        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;qpush(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                     return 1;
37                 }
38                 qpush(match[y]);
39             }else if(!S[y]&&st[y]!=st[x]){
40                 int l=lca(y,x);
41                 flower(y,x,l,q),flower(x,y,l,q);
42             }
43         }
44     }
45     return 0;
46 }
47 inline int blossom(){
48     int ans=0;
49     for(int i=1;i<=n;++i)
50         if(!match[i]&&bfs(i))++ans;
51     return ans;

```

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);

```

```

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·
24                 把這行刪掉·執行一次後f[K]就會是所
25                 有點的答案
26             f[t][i]%=P;
27         }
28     }
29     return f[K][u];
30 }
31 inline vector<long long> graph_hash(){
32     vector<long long> ans;
33     for(int i=0;i<n;++i)ans.push_back(
34         point_hash(i));
35     sort(ans.begin(),ans.end());
36     return ans;

```

5.5 KM.cpp

```

1 #define MAXN 100
2 int n;
3 int g[MAXN][MAXN],lx[MAXN],ly[MAXN],slack_y[
4     MAXN];
5 int match_y[MAXN];
6 bool vx[MAXN],vy[MAXN];//要保證g是完全二分圖
7 bool dfs(int x,bool adjust=1){//DFS找增廣
8     路·is=1表示要交換邊
9     if(vx[x])return 0;
10    vx[x]=1;
11    for(int y=0;y<n;++y){
12        if(vy[y])continue;
13        int t=lx[x]+ly[y]-g[x][y];
14        if(t==0){
15            vy[y]=1;
16            if(match_y[y]==-1||dfs(match_y[y],
17                adjust)){
18                if(adjust)match_y[y]=x;
19                return 1;
20            }
21        }else if(slack_y[y]>t)slack_y[y]=t;
22    }
23    return 0;
24 }
25 inline int km(){
26     memset(ly,0,sizeof(int)*n);
27     memset(match_y,-1,sizeof(int)*n);
28     for(int x=0;x<n;++x){
29         lx[x]=0;
30         for(int y=0;y<n;++y){
31             lx[x]=max(lx[x],g[x][y]);
32         }
33     }
34     for(int x=0;x<n;++x){
35         for(int y=0;y<n;++y)slack_y[y]=INT_MAX;
36         memset(vx,0,sizeof(bool)*n);

```



```

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

```

5.6 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<n;++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<n;++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 Minimum_General_Weighted

```

1 struct Graph {
2     // Minimum General Weighted Matching (
3     // Perfect Match) 0-base
4     static const int MXN = 105;
5
6     int n, edge[MXN][MXN];
7     int match[MXN],dis[MXN],onstk[MXN];
8     vector<int> stk;
9
10    void init(int _n) {
11        n = _n;
12        for (int i=0; i<n; i++)
13            for (int j=0; j<n; j++)
14                edge[i][j] = 0;
15
16    void add_edge(int u, int v, int w) {
17        edge[u][v] = edge[v][u] = w;
18    }
19
20    bool SPFA(int u){
21        if (onstk[u]) return true;
22        stk.push_back(u);
23        onstk[u] = 1;
24        for (int v=0; v<n; v++){
25            if (u != v && match[u] != v && !onstk[
                v]){
26                int m = match[v];
27                if (dis[m] > dis[u] - edge[v][m] +
                    edge[u][v]){
28                    dis[m] = dis[u] - edge[v][m] +
                        edge[u][v];
29                    onstk[v] = 1;
30                    stk.push_back(v);
31                    if (SPFA(m)) return true;
32                    stk.pop_back();
33                    onstk[v] = 0;
34                }
35            }
36        }
37        onstk[u] = 0;
38        stk.pop_back();
39        return false;

```

```

38    }
39
40    int solve() {
41        // find a match
42        for (int i=0; i<n; i+=2){
43            match[i] = i+1;
44            match[i+1] = i;
45        }
46        for(;;){
47            int found = 0;
48            for (int i=0; i<n; i++)
49                dis[i] = onstk[i] = 0;
50            for (int i=0; i<n; i++){
51                stk.clear();
52                if (!onstk[i] && SPFA(i)){
53                    found = 1;
54                    while (stk.size()>=2){
55                        int u = stk.back(); stk.pop_back
                            ();
56                        int v = stk.back(); stk.pop_back
                            ();
57                        match[u] = v;
58                        match[v] = u;
59                    }
60                }
61                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;

```

5.8 Rectilinear_Steiner_tree

```

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=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 }

```

5.9 treeISO.cpp

```

1 const int MAXN=100005;
2 const long long X=12327,P=0xdefaced;
3 vector<int> g[MAXN];
4 bool vis[MAXN];

```

```

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

```

5.10 一般圖最大權匹配.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define INF INT_MAX
4 #define MAXN 400
5 struct edge{
6   int u,v,w;
7   edge(){}
8   edge(int u,int v,int w):u(u),v(v),w(w){}
9 };
10 int n,n_x;
11 edge g[MAXN*2+1][MAXN*2+1];
12 int lab[MAXN*2+1];
13 int match[MAXN*2+1],slack[MAXN*2+1],st[MAXN
14   *2+1],pa[MAXN*2+1];
15 int flower_from[MAXN*2+1][MAXN+1],S[MAXN
16   *2+1],vis[MAXN*2+1];
17 vector<int> flower[MAXN*2+1];
18 queue<int> q;
19 int e_delta(const edge &e){ // does not work
20   inside blossoms
21   return lab[e.u]+lab[e.v]-g[e.u][e.v].w*2;
22 }
23 void update_slack(int u,int x){
24   if(!slack[x]||e_delta(g[u][x])<e_delta(g[
25     slack[x]][x]))slack[x]=u;
26 }
27 void set_slack(int x){
28   slack[x]=0;
29   for(int u=1;u<=n;u++){
30     if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)
31       update_slack(u,x);
32 }
33 void q_push(int x){
34   if(x<=n)q.push(x);
35   else for(size_t i=0;i<flower[x].size();i
36     ++)q_push(flower[x][i]);
37 }
38 void set_st(int x,int b){
39   st[x]=b;
40   if(x>n)for(size_t i=0;i<flower[x].size()
41     ++i)
42     set_st(flower[x][i],b);
43 }
44 int get_pr(int b,int xr){
45   int pr=find(flower[b].begin(),flower[b].
46     end(),xr)-flower[b].begin();
47   if(pr%2==1){//檢查他在前一層是奇點還是偶點
48     reverse(flower[b].begin()+1,flower[b].
49       end());
50   }
51 }

```

```

41   return (int)flower[b].size()-pr;
42 }else return pr;
43 }
44 void set_match(int u,int v){
45   match[u]=g[u][v].v;
46   if(u>n){
47     edge e=g[u][v];
48     int xr=flower_from[u][e.u],pr=get_pr(u,
49       xr);
50     for(int i=0;i<pr;i++)set_match(flower[u
51       ][i],flower[u][i^1]);
52     set_match(xr,v);
53     rotate(flower[u].begin(),flower[u].begin
54       ()+pr,flower[u].end());
55 }
56 void augment(int u,int v){
57   for(;;){
58     int xnv=st[match[u]];
59     set_match(u,v);
60     if(!xnv)return;
61     set_match(xnv,st[pa[xnv]]);
62     u=st[pa[xnv]],v=xnv;
63 }
64 int get_lca(int u,int v){
65   static int t=0;
66   for(++t;u|v;swap(u,v)){
67     if(u==0)continue;
68     if(vis[u]==t)return u;
69     vis[u]=t;//這種方法可以不用清空v陣列
70     u=st[match[u]];
71     if(u==0)continue;
72     if(u==st[pa[u]])return u;
73   }
74   return 0;
75 }
76 void add_blossom(int u,int lca,int v){
77   int b=n+1;
78   while(b<=n_x&&st[b]==0){
79     if(b>n_x)++n_x;
80     lab[b]=0,S[b]=0;
81     match[b]=match[lca];
82     flower[b].clear();
83     flower[b].push_back(lca);
84     for(int x=u,y; x!=lca; x=st[pa[y]])
85       flower[b].push_back(x),flower[b].
86       push_back(y=st[match[x]]),q_push(y);
87     reverse(flower[b].begin()+1,flower[b].end
88       ());
89     for(int x=v,y; x!=lca; x=st[pa[y]])
90       flower[b].push_back(x),flower[b].
91       push_back(y=st[match[x]]),q_push(y);
92     set_st(b,b);
93     for(int x=1;x<=n_x;x++){
94       g[b][x].w=g[x][b].w
95       =0;
96       for(int x=1;x<=n_x;x++){
97         if(g[b][x].w==0||e_delta(g[xs][x])<
98           e_delta(g[b][x]))
99           g[b][x]=g[xs][x],g[x][b]=g[x][xs];
100       }
101       for(int x=1;x<=n_x;x++){
102         if(flower_from[xs][x])flower_from[b][x
103           ]=xs;
104       }
105     }
106   }
107 }

```

```

97   }
98   set_slack(b);
99 }
100 void expand_blossom(int b){ // S[b] == 1
101   for(size_t i=0;i<flower[b].size();++i)
102     set_st(flower[b][i],flower[b][i]);
103   int xr=flower_from[b][g[b][pa[b]].u],pr=
104     get_pr(b,xr);
105   for(int i=0;i<pr;i++){
106     int xs=flower[b][i],xns=flower[b][i+1];
107     pa[xs]=g[xns][xs].u;
108     S[xs]=1,S[xns]=0;
109     slack[xs]=0,set_slack(xns);
110     q_push(xns);
111   }
112   S[xr]=1,pa[xr]=pa[b];
113   for(size_t i=pr+1;i<flower[b].size();++i){
114     int xs=flower[b][i];
115     S[xs]=-1,set_slack(xs);
116   }
117   st[b]=0;
118 bool on_found_edge(const edge &e){
119   int u=st[e.u],v=st[e.v];
120   if(S[v]==-1){
121     pa[v]=e.u,S[v]=1;
122     int nu=st[match[v]];
123     slack[v]=slack[nu]=0;
124     S[nu]=0,q_push(nu);
125   }else if(S[v]==0){
126     int lca=get_lca(u,v);
127     if(!lca){
128       augment(u,v),augment(v,u);
129       return true;
130     }else add_blossom(u,lca,v);
131   }
132   return false;
133 }
134 bool matching(){
135   memset(S+1,-1,sizeof(int)*n_x);
136   memset(slack+1,0,sizeof(int)*n_x);
137   q=queue<int>();
138   for(int x=1;x<=n_x;x++){
139     if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,
140       q_push(x);
141   }
142   if(q.empty())return false;
143   for(;;){
144     while(q.size()){
145       int u=q.front();q.pop();
146       if(S[st[u]]==1)continue;
147       for(int v=1;v<=n_x;v++){
148         if(g[u][v].w>0&&st[u]!=st[v]){
149           if(e_delta(g[u][v])<e_delta(g[
150             slack[u]][v])){
151             if(on_found_edge(g[u][v]))return
152               true;
153             else update_slack(u,st[v]);
154           }
155         }
156       }
157       int d=INF;
158       for(int b=n+1;b<=n_x;b++){
159         if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2)
160           ;
161       }
162       for(int x=1;x<=n_x;x++){
163         if(st[x]==x&&slack[x]){
164           if(S[x]==-1)d=min(d,e_delta(g[slack[
165             x]][x]));
166         }
167       }
168     }
169   }
170 }

```

```

171   else if(S[x]==0)d=min(d,e_delta(g[
172     slack[x]][x])/2);
173 }
174 for(int u=1;u<=n;u++){
175   if(S[st[u]]==0){
176     if(lab[u]<=d)return 0;
177     lab[u]-=d;
178   }else if(S[st[u]]==1)lab[u]+=d;
179 }
180 for(int b=n+1;b<=n_x;b++){
181   if(st[b]==b){
182     if(S[st[b]]==0)lab[b]+=d*2;
183     else if(S[st[b]]==1)lab[b]-=d*2;
184   }
185   q=queue<int>();
186   for(int x=1;x<=n_x;x++){
187     if(st[x]==x&&slack[x]&&st[slack[x]]!=x
188       &&e_delta(g[slack[x]][x])<0){
189       if(on_found_edge(g[slack[x]][x]))
190         return true;
191     }
192     for(int b=n+1;b<=n_x;b++){
193       if(st[b]==b&&S[b]==1&&lab[b]==0)
194         expand_blossom(b);
195     }
196   }
197   return false;
198 }
199 pair<long long,int> weight_blossom(){
200   memset(match+1,0,sizeof(int)*n);
201   n_x=n;
202   int n_matches=0;
203   long long tot_weight=0;
204   for(int u=1;u<=n;u++){
205     st[u]=u,flower[u].
206     clear();
207     int w_max=0;
208     for(int u=1;u<=n;u++){
209       for(int v=1;v<=n;v++){
210         flower_from[u][v]=(u==v?u:0);
211         w_max=max(w_max,g[u][v].w);
212       }
213     }
214     for(int u=1;u<=n;u++){
215       lab[u]=w_max;
216       while(matching())++n_matches;
217       for(int u=1;u<=n;u++){
218         if(match[u]&&match[u]<u)
219           tot_weight+=g[u][match[u]].w;
220       }
221       return make_pair(tot_weight,n_matches);
222     }
223   }
224 void init_weight_graph(){
225   for(int u=1;u<=n;u++){
226     for(int v=1;v<=n;v++){
227       g[u][v]=edge(u,v,0);
228     }
229   }
230 }
231 int main(){
232   int m;
233   scanf("%d",&n,&m);
234   init_weight_graph();
235   for(int i=0;i<m;i++){
236     int u,v,w;
237     scanf("%d%d%d",&u,&v,&w);
238     g[u][v].w=g[v][u].w=w;
239   }
240   printf("%lld\n",weight_blossom().first);
241   for(int u=1;u<=n;u++)printf("%d ",match[u]
242     );puts("");
243   return 0;
244 }/* 20
245 5 7 9 3 7 4 3 6 6 2 5 8 5 1 9 1 3 6 6 5 1

```

```

218 2 7 4 2 3 5 6 4 2 7 1 5 5 4 4 1 3 5 3 9
219 7 6 4 2 1 3 4 3 9 6 2 7 4 2 8 6 1 10
220 -----
221 28
222 6 0 4 3 7 1 5*/

```

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[nd[i]]=0;
20             for(int i=1;i<tn;++i){
21                 ind=i;
22                 for(int j=i;j<tn;++j){
23                     dis[nd[j]]+=g[nd[i-1]][nd[j]];
24                     if(dis[nd[ind]]<dis[nd[j]])ind=j;
25                 }
26                 swap(nd[ind],nd[i]);
27             }
28             if(ans>dis[nd[ind]])ans=dis[t=nd[ind]];
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 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(w){}
9     };
10    vector<edge>E;// 0-base
11    int pe[MAXN],id[MAXN],vis[MAXN];
12    T in[MAXN];

```

```

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

```

6 language

6.1 CNF.cpp

```

1 #define MAXN 55
2 struct CNF{
3     int s,x,y;//s->xy | s->x, if y==1
4     int cost;
5     CNF(){}
6     CNF(int s,int x,int y,int c):s(s),x(x),y(y),cost(c){}
7 };
8 int state;//規則數量
9 map<char,int> rule;//每個字元對應到的規則
10 //小寫字母為終端字符

```

```

10 vector<CNF> cnf;
11 inline void init(){
12     state=0;
13     rule.clear();
14     cnf.clear();
15 }
16 inline void add_to_cnf(char s,const string &
17     p,int cost){
18     //加入一個s -> <p>的文法，代價為cost
19     if(rule.find(s)==rule.end())rule[s]=state++;
20     for(auto c:p)if(rule.find(c)==rule.end())
21         rule[c]=state++;
22     if(p.size()==1){
23         cnf.push_back(CNF(rule[s],rule[p[0]],-1,
24             cost));
25     }else{
26         int left=rule[s];
27         int sz=p.size();
28         for(int i=0;i<sz-2;++i){
29             cnf.push_back(CNF(left,rule[p[i]],
30                 state,0));
31             left=state++;
32         }
33         cnf.push_back(CNF(left,rule[p[sz-2]],
34             rule[p[sz-1]],cost));
35     }
36 }
37 vector<long long> dp[MAXN][MAXN];
38 vector<bool> neg_INF[MAXN][MAXN];//如果花費
39 //是負的可能會有無限小的情形
40 inline void relax(int l,int r,const CNF &c,
41     long long cost,bool neg_c=0){
42     if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x]
43         ||cost<dp[l][r][c.s])){
44         if(neg_c||neg_INF[l][r][c.x]){
45             dp[l][r][c.s]=0;
46             neg_INF[l][r][c.s]=true;
47         }else dp[l][r][c.s]=cost;
48     }
49 }
50 inline void bellman(int l,int r,int n){
51     for(int k=1;k<=state;++k)
52         for(auto c:cnf)
53             if(c.y==1)relax(l,r,c,dp[l][r][c.x]+c
54                 .cost,k=n);
55 }
56 inline void cyk(const vector<int> &tok){
57     for(int i=0;i<(int)tok.size();++i){
58         for(int j=0;j<(int)tok.size();++j){
59             dp[i][j]=vector<long long>(state+1,
60                 INT_MAX);
61             neg_INF[i][j]=vector<bool>(state+1,
62                 false);
63         }
64     }
65     dp[i][i][tok[i]]=0;
66     bellman(i,i,tok.size());
67 }
68 for(int r=1;r<(int)tok.size();++r){
69     for(int l=r-1;l>=0;--l){
70         for(int k=1;k<r;++k)
71             for(auto c:cnf)
72                 if(~c.y)relax(l,r,c,dp[l][k][c.x]+
73                     dp[k+1][r][c.y]+c.cost);
74         bellman(l,r,tok.size());
75     }
76 }

```

6.2 earley.cpp

```

1 struct Rule{
2     vector<vector<Rule*> > p;
3     void add(const vector<Rule*> &l){
4         p.push_back(l);
5     }
6 };
7 map<string,Rule*> NameRule;
8 map<Rule*,string> RuleName;
9 inline void init_Rule(){
10     for(auto r:RuleName)delete r.first;
11     RuleName.clear();
12     NameRule.clear();
13 }
14 inline Rule *add_rule(const string &s){
15     if(NameRule.find(s)!=NameRule.end())return
16         NameRule[s];
17     Rule *r=new Rule();
18     RuleName[r]=s;
19     NameRule[s]=r;
20     return r;
21 }
22 typedef vector<Rule*> production;
23 struct State{
24     Rule *r;
25     int rid,dot_id,start,end;
26     State(Rule *r,int rid,int dot_id,int start):r(
27         r),rid(rid),dot_id(dot_id),start(start),
28         end(-1){}
29     State(Rule *r=0,int col=0):r(r),rid(-1),
30         dot_id(-1),start(-1),end(col){}
31     bool completed()const{
32         return rid==-1||dot_id>=(int)r->p[rid].
33             size();
34     }
35     Rule *next_term()const{
36         if(completed())return 0;
37         return r->p[rid][dot_id];
38     }
39 }
40 bool operator<(const State& b)const{
41     if(start!=b.start)return start<b.start;
42     if(dot_id!=b.dot_id)return dot_id<b.
43         dot_id;
44     if(r!=b.r)return r<b.r;
45     return rid<b.rid;
46 }
47 void print()const{
48     cout<<RuleName[r]<<"-";
49     if(rid!=-1)for(size_t i=0; i<p[rid].size(); ++i){
50         if((int)i==dot_id)cout<<" "<<"$";
51         if(i>r->p[rid].size())break;
52         cout<<" "<<RuleName[r->p[rid][i]];
53     }
54     cout<<" "<<"["<<start<<" "<<end<<""]<<
55         endl;
56 }
57 };
58 struct Column{

```

```

51 Rule *term;
52 string value;
53 vector<State> s;
54 map<State, set<pair<State, State> > > div;
55 //div比較像一棵 左兄右子的樹
56 Column(Rule *r, const string &s):term(r),
    value(s){
57 Column(){
58 bool add(const State &st, int col){
59     if(div.find(st)==div.end()){
60         div[st];
61         s.push_back(st);
62         s.back().end=col;
63         return true;
64     }else return false;
65 }
66 };
67 inline vector<Column> lexer(string text){
68     //tokenize · 要自己寫 · 以下為範例
69     //他會把 input stream 變成 token stream ·
    就是 (terminal, value) pair
70     vector<Column> token;
71     replace(text.begin(), text.end(), ' ', ' ');
72     stringstream ss(text);
73     while(ss>>text){
74         if(text=="a" || text=="of") continue;
75         if(text=="list"){
76             token.push_back(Column(NameRule["(", ")", "list"], "list"));
77         }else if(text=="and"){
78             token.push_back(Column(NameRule["(", ")", "and"], "and"));
79         }else token.push_back(Column(NameRule["(", ")", "T"], "T"));
80     }
81     return token;
82 }
83 vector<Column> table;
84 inline void predict(int col, Rule *rul){
85     for(size_t i=0; i<rul->p.size(); ++i){
86         table[col].add(State(rul, i, 0, col), col);
87     }
88 }
89 inline void scan(int col, const State &s, Rule
    *r){
90     if(r!=table[col].term) return;
91     State ns(s.r, s.rid, s.dot_id+1, s.start);
92     table[col].add(ns, col);
93     table[col].div[ns].insert(make_pair(s,
        State(r, col)));
94 }
95 inline void complete(int col, const State &s)
    {
96     for(size_t i=0; i<table[s.start].s.size()
        ; ++i){
97         State &st=table[s.start].s[i];
98         Rule *term=st.next_term();
99         if(!term || term->p.size()==0) continue;
100         if(term==s.r){
101             State nst(st.r, st.rid, st.dot_id+1, st.
                start);
102             table[col].add(nst, col);
103             table[col].div[nst].insert(make_pair(
                st, s));
104 }
105 }
106 }
107 inline pair<bool, State> parse(Rule *GAMMA,
    const vector<Column> &tokens){
108     table.resize(tokens.size()+1);
109     for(size_t i=0; i<tokens.size(); ++i) table[i
        +1]=Column(tokens[i]);
110     table[0]=Column();
111     table[0].add(State(GAMMA, 0, 0, 0), 0);
112     for(size_t i=0; i<table.size(); ++i){
113         for(size_t j=0; j<table[i].s.size(); ++j){
114             State state=table[i].s[j];
115             if(state.completed()) complete(i, state)
                ;
116             else{
117                 Rule *term=state.next_term();
118                 if(term->p.size()==0) predict(i, term);
119                 else if(i+1<table.size()) scan(i+1,
                    state, term);
120             }
121         }
122     }
123     for(size_t i=0; i<table.back().s.size(); ++i
        ){
124         if(table.back().s[i].r==GAMMA && table.
            back().s[i].completed()){
125             return make_pair(true, table.back().s[i
                ]);
126         }
127     }
128     return make_pair(false, State(0, -1));
129 }
130 struct node{//語法樹的節點
131     State s;
132     vector<vector<node*> > child; //vector<node
        *>.size()>1表示ambiguous
133     node(const State &s):s(s){
134         node{};
135     };
136     struct State_end_cmp{
137         bool operator()(const State &a, const State
            &b) const{
138             return a.end<b.end || (a.end==b.end && a<b);
139         }
140     };
141     map<State, node*, State_end_cmp> cache;
142     vector<node*> node_set;
143     inline void init_cache(){
144         for(auto d: node_set) delete d;
145         cache.clear();
146         node_set.clear();
147     }
148     void build_tree(const State &s, node *pa,
        bool amb=0){
149         if(cache.find(s)!=cache.end()){
150             pa->child.push_back(vector<node*>(1,
                cache[s]));
151             return;
152         }
153         node *o;
154         if(s.completed()){
155             o=new node(s);
156             if(amb) pa->child.back().push_back(o);
157             else pa->child.push_back(vector<node
                *>(1, o));
158 }else o=pa->child.back().back();
159 amb=0;
160 for(auto div: table[s.end].div[s]){
161     if(!amb) build_tree(div.first, pa);
162     _build_tree(div.second, o, amb);
163     amb=1;
164 }
165 if(s.completed()) cache[s]=o;
166 }
167 inline node *build_tree(const State &s){
168     init_cache();
169     node o;
170     _build_tree(s, &o);
171     assert(o.child.size()==1);
172     assert(o.child.back().size()==1);
173     return o.child.back().back();
174 }
175 void print_tree(node *o, int dep=0){
176     cout<<string(dep, ' '), o->s.print();
177     for(auto div: o->child){
178         for(auto nd: div){
179             print_tree(nd, dep+2);
180         }
181     }
182 }
183 //開始寫code: 以下為加入語法的範例
184 inline Rule *get_my_Rule(){
185     Rule *S=add_rule("S"), *E=add_rule("E"), *L=
        add_rule("L");
186     Rule *list=add_rule("("), *AND=add_rule("&")
        , *T=add_rule("T");
187     S->add({list, E});
188     S->add({list, L});
189     L->add({E, L});
190     L->add({E, AND, E});
191     E->add({T});
192     E->add({S});
193     Rule *GAMMA=add_rule("GAMMA"); //一定要有
        gamma rule當作是最上層的語法
194     GAMMA->add({S});
195     return GAMMA;
196 }
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(1);
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 }

```

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 vector<edge> E;
9 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(){

```


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 LL log_mod(const LL &a, const LL &b, const
72     LL &p) {
73     // a ^ x = b ( mod p )
74     int m=sqrt(p+.5), e=1;
75     LL v=inv(modexp(a, m, p), p);
76     map<LL, int> x;
77     x[1]=0;
78     for(int i=1; i<m; i++) {
79         e = LLmul(e, a, p);
80         if(!x.count(e)) x[e] = i;
81     }
82     for(int i=0; i<m; i++) {
83         if(x.count(b)) return i*m + x[b];
84         b = LLmul(b, v, p);
85     }
86     return -1;
87 }
88 LL Tonelli-Shanks(const LL &n, const LL &p)
89 {
90     // x^2 = n ( mod p )
91     if(n==0) return 0;
92     if(Legendre(n, p)!=1) while(1) { puts("SQRT
93         ROOT does not exist"); }
94     int S = 0;
95     LL Q = p-1;
96     while( !(Q&1) ) { Q>=1; ++S; }
97     if(S==1) return modexp(n%p, (p+1)/4, p);
98     LL z = 2;
99     for(; Legendre(z, p)!=-1; ++z)
100         LL c = modexp(z, Q, p);
101     LL R = modexp(n%p, (Q+1)/2, p), t = modexp(n
102         %p, Q, p);
103     int M = S;
104     while(1) {
105         if(t==1) return R;
106         LL b = modexp(c, 1L<<(M-i-1), p);
107         R = LLmul(R, b, p);
108         t = LLmul( LLmul(b, b, p), t, p);
109         c = LLmul(b, b, p);
110         M = i;
111     }
112     return -1;
113 }
114 template<typename T>
115 T Euler(T n) {
116     T ans=n;
117     for(T i=2; i*i<=n; i++) {

```

```

115     if(n%i==0) {
116         ans=ans/i*(i-1);
117         while(n%i==0) n/=i;
118     }
119     if(n>1) ans=ans/n*(n-1);
120     return ans;
121 }
122 //Chinese_remainder_theorem
123 template<typename T>
124 T pow_mod(T n, T k, T m) {
125     T ans=1;
126     for(n=(n>m?n%m:n); k>=1; k>>=1) {
127         if(k&1) ans=ans*n%m;
128         n=n*n%m;
129     }
130     return ans;
131 }
132 template<typename T>
133 T crt(vector<T> &m, vector<T> &a) {
134     T M=1, tM, ans=0;
135     for(int i=0; i<(int)m.size(); i++) M*=m[i];
136     for(int i=0; i<(int)a.size(); i++) {
137         tM=M/m[i];
138         ans=(ans+(a[i]*tM%M)*pow_mod(tM, Euler(m
139             [i])-1, m[i])%M)%M;
140     }
141     /*如果m[i]是質數 · Euler(m[i])-1=m[i]-2 ·
142         就不用算Euler了*/
143     return ans;
144 }

```

8.2 bit_set.cpp

```

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

```

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

```

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((-1)):pi(pi)) {}
6     unsigned int bit_reverse(unsigned int a,
7         int len) {
8         a=((a&0x55555555)<<1)|((a&0xAAAAAAAA)>>1);
9         a=((a&0x33333333)<<2)|((a&0xCCCCCCCC)>>2);
10        a=((a&0x0F0F0F0F)<<4)|((a&0xF0F0F0F0)>>4);
11        a=((a&0x00FF00FF)<<8)|((a&0xFF00FF00)>>8);
12        a=((a&0x0000FFFF)<<16)|((a&0xFFFF0000)>>16);
13        return a>>(32-len);
14    }
15    void fft(bool is_inv, VT &in, VT &out, int N)
16    {
17        int bitlen=std::lg(N), num=is_inv?-1:1;
18        for(int i=0; i<N; i++) out[bit_reverse(i,
19            bitlen)]=in[i];
20        for(int step=2; step<=N; step<=1) {
21            const int mh=step>>1;
22            for(int i=0; i<N; i+=step) {
23                std::complex<T> wi=exp(std::complex<
24                    T>(0, i*num*pi/mh));
25                for(int j=i; j<N; j+=step) {
26                    int k=j+mh;
27                    std::complex<T> u=out[j], t=wi*out[
28                        k];
29                    out[j]=u+t;

```

```

24         out[k]=u-t;
25     }
26 }
27 }
28 if(is_inv)for(int i=0;i<N;++i)out[i]/=N;
29 }
30 };

```

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

```

```

45 int main () {
46     vector<double>ve;
47     vector<double>ans = cal(ve, n);
48     // 視情況把答案 +eps · 避免 -0
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];
18        lastm = (lastm/d)*m[i];
19        lastb = (lastb+b[i])%lastm;
20    }
21    return make_pair(lastb<0?lastb+lastm:lastb
22        ,lastm);
23 }

```

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;
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<c;++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        }
49        return true;
50    }
51    T gas(){
52        vector<T> lazy(r,1);
53        bool sign=false;
54        for(int i=0;i<r;++i){
55            if( m[i][i]==0 ){
56                int j=i+1;
57                while(j<r&&!m[j][i])j++;
58                if(j==r)continue;
59                if(j==r)continue;
60                m[i].swap(m[j]);
61                sign=!sign;
62            }
63            for(int j=0;j<r;++j){
64                if(i==j)continue;
65                lazy[j]=lazy[j]*m[i][i];
66                T mx=m[j][i];
67                for(int k=0;k<c;++k)
68                    m[j][k]=m[j][k]*m[i][i]-m[i][k]*mx

```

```

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 long long mod_mul(long long a,long long b,
14     long long m){
15     a%=m,b%=m;
16     long long y=(long long)((double)a*b/m+0.5)
17         ;/* fast for m < 2^58 */
18     long long r=(a*b-y*m)%m;
19     return r<0?r+m:r;
20 }
21 template<typename T>
22 T pow(T a,T b,T mod){/*a^b%mod
23     T ans=1;
24     for(;b;a=mod_mul(a,a,mod),b>>=1)
25         if(b&1)ans=mod_mul(ans,a,mod);
26     return ans;
27 }
28 int sprp[3]={2,7,61};/*int%d^3@i,@
29 int llsp[3]
30     [7]={2,325,9375,28178,450775,9780504,1795265
31         };/*@unsigned Long Long%d^3@
32 template<typename T>
33 bool isprime(T n,int *sprp,int num){
34     if(n==2)return 1;
35     if(n<2||n%2==0)return 0;
36     int t=0;
37     T u=n-1;
38     for(;u%2==0;++t)u>>=1;
39     for(int i=0;i<num;++i){
40         T a=sprp[i]%n;
41         if(a==0||a==1||a==n-1)continue;
42         T x=pow(a,u,n);
43         if(x==1||x==n-1)continue;
44         for(int j=0;j<t;++j){
45             x=mod_mul(x,x,n);
46             if(x==1)return 0;
47             if(x==n-1)break;
48         }
49         if(x==n-1)continue;
50         return 0;
51     }
52 }

```

```

48 return 1;
49 }

```

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<
  T>>
8 struct NTT{
9   const T P,G;
10   NTT(T p=(1<<23)*7*17+1,T g=3):P(p),G(g){}
11   unsigned int bit_reverse(unsigned int a,
    int len){
12     a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)
      >>1);
13     a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)
      >>2);
14     a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0F0U)
      >>4);
15     a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)
      >>8);
16     a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)
      >>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     int bitlen=std::__lg(N);
29     for(int i=0;i<N;++i)out[bit_reverse(i,
      bitlen)]=in[i];
30     for(int step=2,id=1;step<=N;step<=1,++
      id){
31       T wn=pow_mod(G,(P-1)>>id,P),wi=1,u,t;
32       const int mh=step>>1;
33       for(int i=0;i<mh;++i){
34         for(int j=i;j<N;j+=step){
35           u=out[j],t=wi*out[j+mh]%P;
36           out[j]=u+t;
37           out[j+mh]=u-t;
38           if(out[j]>=P)out[j]-=P;
39           if(out[j+mh]<0)out[j+mh]+=P;
40         }
41         wi=wi*wn%P;
42       }
43     }
44     if(is_inv){
45       for(int i=1;i<N/2;++i)std::swap(out[i]
        ],out[N-i]);
46     T invn=pow_mod(N,P-2,P);

```

```

47   for(int i=0;i<N;++i)out[i]=out[i]*invn
    %P;
48   }
49   }
50 };

```

8.11 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 100000
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
  ,long long mod){//a^b%mod
21   long long ans=1;
22   for(;b;a=a%mod,b>>=1)
23     if(b&1)ans=ans*a%mod;
24   return ans;
25 }
26 bool isless(long long *a,int n,int k){
27   if(*a==1)return k>1;
28   if(--n==0)return *a<k;
29   int next=0;
30   for(long long b=1;b<k;++next)
31     b*=a;
32   return isless(a+1,n,next);
33 }
34 long long high_pow(long long *a,int n,long
  long mod){
35   if(*a==1||--n==0)return *a%mod;
36   int k=0,r=euler[mod];
37   for(long long tma=1;tma!=pow(*a,k+r,mod)
    ;++k)
38     tma=tma*(*a)%mod;
39   if(isless(a+1,n,k))return pow(*a,high_pow(
    a+1,n,k),mod);
40   int tmd=high_pow(a+1,n,r);
41   int t=(tmd-k+r)%r;
42   return pow(*a,k+t,mod);
43 }
44 long long a[100005];
45 int t,mod;
46 int main(){
47   init_euler();
48   scanf("%d",&t);
49   #define n 4
50   while(t--){

```

```

51   for(int i=0;i<n;++i)scanf("%lld",&a[i]);
52   scanf("%d",&mod);
53   printf("%lld\n",high_pow(a,n,mod));
54   }
55   return 0;
56 }

```

8.12 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 *
    Inverse(b)
26  }
27  operator T()const{return data;}
28  friend istream &operator>>(istream &i,
    mod_t &b){
29    T d;
30    i>>d;
31    b=mod_t(d);
32    return i;
33  }
34 };

```

8.13 質因數分解.cpp

```

1 LL func(const LL n,const LL mod,const int c)
  {
2   return (LLmul(n,n,mod)+c+mod)%mod;
3 }
4
5 LL pollorroho(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
16 void prefactor(LL &n, vector<LL> &v) {
17   for(int i=0;i<12;++i) {
18     while(n%prime[i]==0) {
19       v.push_back(prime[i]);
20       n/=prime[i];
21     }
22   }
23 }
24
25 void smallfactor(LL n, vector<LL> &v) {
26   if(n<MAXPRIME) {
27     while(isp[prime[n]]) {
28       v.push_back(isp[prime[n]]);
29       n/=isp[prime[n]];
30     }
31     v.push_back(n);
32   } else {
33     for(int i=0;i<primecnt;&&prime[i]*prime[i]
      <=n;++i) {
34       while(n%prime[i]==0) {
35         v.push_back(prime[i]);
36         n/=prime[i];
37       }
38     }
39     if(n!=1) v.push_back(n);
40   }
41 }
42
43 void comfactor(const LL &n, vector<LL> &v) {
44   if(n<1e9) {
45     smallfactor(n,v);
46     return;
47   }
48   if(Isprime(n)) {
49     v.push_back(n);
50     return;
51   }
52   LL d;
53   for(int c=3; ; ++c) {
54     d = pollorroho(n,c);
55     if(d!=n) break;
56   }
57   comfactor(d,v);
58   comfactor(n/d,v);
59 }
60
61 void Factor(const LL &x, vector<LL> &v) {
62   LL n = x;
63   if(n==1) { puts("Factor 1"); return; }
64   prefactor(n,v);
65   if(n==1) return;
66   comfactor(n,v);
67   sort(v.begin(),v.end());
68 }
69
70 void AllFactor(const LL &n,vector<LL> &v) {
71   vector<LL> tmp;
72   Factor(n,tmp);
73   v.clear();
74   v.push_back(1);

```

```

75 int len;
76 LL now=1;
77 for(int i=0;i<tmp.size();++i) {
78     if(i==0 || tmp[i]!=tmp[i-1]) {
79         len = v.size();
80         now = 1;
81     }
82     now*=tmp[i];
83     for(int j=0;j<len;++j)
84         v.push_back(v[j]*now);
85 }
86 }

```

9 String

9.1 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():ed(0),cnt_dp(0),vis(0){
8         for(int i=0;i<=R-L;++i)next[i]=0;
9     };
10 public:
11     std::vector<joe> S;
12     std::vector<int> q;
13     int qs, qe, vt;
14     ac_automaton():S(1),qs(0),qe(0),vt(0){
15         void clear(){
16             q.clear();
17             S.resize(1);
18             for(int i=0;i<=R-L;++i)S[0].next[i]=0;
19             S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
20         }
21         void insert(const char *s){
22             int o=0;
23             for(int i=0,id;s[i];++i){
24                 id=s[i]-L;
25                 if(!S[o].next[id]){
26                     S.push_back(joe());
27                     S[o].next[id]=S.size()-1;
28                 }
29                 o=S[o].next[id];
30             }
31             ++S[o].ed;
32         }
33         void build_fail(){
34             S[0].fail=S[0].efl=-1;
35             q.clear();
36             q.push_back(0);
37             ++qe;
38             while(qs!=qe){
39                 int pa=q[qs++],id,t;
40                 for(int i=0;i<=R-L;++i){
41                     t=S[pa].next[i];
42                     if(!t)continue;
43                     id=S[pa].fail;

```

```

44         while(~id&&!S[id].next[i])id=S[id]
45             ].fail;
46         S[t].fail=~id?S[id].next[i]:0;
47         S[t].efl=S[S[t].fail].ed?S[t].fail
48             :S[S[t].fail].efl;
49         q.push_back(t);
50         ++qe;
51     }
52     /*DP出每個前綴在字串s出現的次數並傳回所
53        有字串被s匹配成功的次數O(N*M)*/
54     int match_0(const char *s){
55         int ans=0,id,p=0,t;
56         for(i=0;s[i];++i){
57             id=s[i]-L;
58             while(!S[p].next[id]&&p=S[p].fail;
59                 if(!S[p].next[id])continue;
60             p=S[p].next[id];
61             ++S[p].cnt_dp; /*匹配成功則它所有後綴
62                            都可以被匹配(DP計算)*/
63         }
64         for(i=qe-1;i>=0;--i){
65             ans+=S[q[i]].cnt_dp*S[q[i]].ed;
66             if(~S[q[i]].fail)S[S[q[i]].fail].
67                 cnt_dp+=S[q[i]].cnt_dp;
68         }
69         return ans;
70     }
71     /*多串匹配走efl邊並傳回所有字串被s匹配成
72        功的次數O(N*M^1.5)*/
73     int match_1(const char *s)const{
74         int ans=0,id,p=0,t;
75         for(int i=0;s[i];++i){
76             id=s[i]-L;
77             while(!S[p].next[id]&&p=S[p].fail;
78                 if(!S[p].next[id])continue;
79             p=S[p].next[id];
80             if(S[p].ed)ans+=S[p].ed;
81             for(t=S[p].efl;~t;t=S[t].efl){
82                 ans+=S[t].ed; /*因為都走efl邊所以保
83                               證匹配成功*/
84             }
85         }
86         return ans;
87     }
88     /*枚舉(s的子字串nA)的所有相異字串各恰一
89        次並傳回次數O(N*M^(1/3))*/
90     int match_2(const char *s){
91         int ans=0,id,p=0,t;
92         ++vt;
93         /*把戳記vt+=1，只要vt沒溢位，所有S[p].
94            vis==vt就會變成false
95            這種利用vt的方法可以O(1)歸零vis陣列*/
96         for(int i=0;s[i];++i){
97             id=s[i]-L;
98             while(!S[p].next[id]&&p=S[p].fail;
99                 if(!S[p].next[id])continue;
100             p=S[p].next[id];
101             if(S[p].ed&&S[p].vis!=vt){
102                 S[p].vis=vt;
103                 ans+=S[p].ed;
104             }
105         }
106     }
107     for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[
108         t].efl){
109         ans+=S[t].ed; /*因為都走efl邊所以保
110                       證匹配成功*/
111     }
112     }
113     };
114     inline int kmp_match(char *A,int lenA,char *
115         B,int lenB,int *fail){
116         int id=-1,ans=0;
117         for(int i=0;i<lenA;++i){
118             while(~id&&B[id+1]!=A[i])id=fail[id];
119             if(B[id+1]==A[i])++id;
120             if(id==lenB-1){ /*匹配成功*/
121                 ++ans;
122                 id=fail[id];
123             }
124         }
125         return ans;
126     }

```

```

98         for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[
99             t].efl){
100             ans+=S[t].ed; /*因為都走efl邊所以保
101                           證匹配成功*/
102         }
103     }
104     return ans;
105     /*把AC自動機變成真的自動機*/
106     void evolution(){
107         for(qs=1;qs!=qe;){
108             int p=q[qs++];
109             for(int i=0;i<=R-L;++i)
110                 if(S[p].next[i]==0)S[p].next[i]=S[
111                     p].fail.next[i];
112         }
113     }

```

```

12 inline int kmp_match(char *A,int lenA,char *
13     B,int lenB,int *fail){
14     int id=-1,ans=0;
15     for(int i=0;i<lenA;++i){
16         while(~id&&B[id+1]!=A[i])id=fail[id];
17         if(B[id+1]==A[i])++id;
18         if(id==lenB-1){ /*匹配成功*/
19             ++ans;
20             id=fail[id];
21         }
22     }
23     return ans;
24 }

```

9.4 manacher.cpp

```

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

```

9.2 hash.cpp

```

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

```

9.3 KMP.cpp

```

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

```

9.5 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
14        置

```

9.6 suffix_array_lcp.cpp

```

1 #define radix_sort(x,y){\
2     for(i=0;i<A;++i)c[i]=0;\
3     for(i=0;i<len;++i)c[x[y[i]]]++;\
4     for(i=1;i<A;++i)c[i]+=c[i-1];\
5     for(i=len-1;i>=0;--i)sa[--c[x[y[i]]]]=y[i]
6     ];\

```



```

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

```

9.7 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 }

```

10 Tarjan

10.1 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[x]=best[anc[x]];
28    return anc[x]=y;
29  }
30  void tarjan(int r){
31    Time=0;
32    for(int t=1; t<=n; ++t){
33      dfn[t]=idom[t]=0; // u=r 或是 u 無法到達 r 時
34      idom[id[u]]=0
35      dom[t].clear();
36      anc[t]=best[t]=semi[t]=t;
37    }
38    dfs(r);
39    for(int y=Time; y>=2; --y){
40      int x=fa[y], idy=id[y];
41      for(auto z: pre[idy]){
42        if(!z=dfn[z]) continue;
43        find(z);
44        semi[y]=min(semi[y], semi[best[z]]);
45      }
46      dom[semi[y]].push_back(y);
47      anc[y]=x;
48      for(auto z: dom[x]){
49        find(z);
50        idom[z]=semi[best[z]]<x?best[z]:x;
51      }
52      dom[x].clear();
53    }
54    for(int u=2; u<=Time; ++u){
55      if(idom[u]!=semi[u]) idom[u]=idom[idom[u]];
56      dom[id[idom[u]]].push_back(id[u]);
57    }
58  } dom;

```

10.2 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 }

```

10.3 橋連通分量.cpp

```

1 #define N 1005
2 struct edge{
3   int u, v;
4   bool is_bridge;
5   edge(int u=0, int v=0): u(u), v(v), is_bridge(0){}
6 };
7 vector<edge> E;
8 vector<int> G[N]; // 1-base
9 int low[N], vis[N], Time;
10 int bcc_id[N], bridge_cnt, bcc_cnt; // 1-base
11 int st[N], top; // BCC 用
12 inline void add_edge(int u, int v){
13   G[u].push_back(E.size());
14   E.push_back(edge(u, v));
15   G[v].push_back(E.size());
16   E.push_back(edge(v, u));
17 }
18 void dfs(int u, int re=-1){ // u 當前點, re 為 u 連
19   接前一個點的邊
20   int v;
21   low[u]=vis[u]=++Time;
22   st[top++]=u;
23   for(size_t i=0; i<G[u].size(); ++i){
24     int e=G[u][i]; v=E[e].v;
25     if(!vis[v]){
26       dfs(v, e^1); // e^1 反向邊
27       low[u]=min(low[u], low[v]);
28       if(vis[u]<low[v]){
29         E[e].is_bridge=E[e^1].is_bridge=1;
30         ++bridge_cnt;
31       }
32     }else if(vis[v]<vis[u]&&e!=re)
33       low[u]=min(low[u], vis[v]);
34   }
35   if(vis[u]==low[u]){ // 處理 BCC
36     ++bcc_cnt; // 1-base
37     do bcc_id[v]=st[--top]=bcc_cnt; // 每個點
38     所在的 BCC
39     while(v!=u);
40   }
41 }
42 inline void bcc_init(int n){
43   Time=bcc_cnt=bridge_cnt=top=0;
44   E.clear();
45   for(int i=1; i<=n; ++i){
46     G[i].clear();
47     vis[i]=bcc_id[i]=0;
48   }
49 }

```

10.4 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
    的根要特判
31 }
32 inline void bcc_init(int n){
33     Time=bcc_cnt=top=0;
34     for(int i=1; i<=n; ++i){
35         G[i].clear();
36         is_cut[i]=vis[i]=bcc_id[i]=0;
37     }
38 }

```

11 Tree_problem

11.1 HeavyLight.cpp

```

1 #include<vector>
2 #define MAXN 100005
3 typedef std::vector<int>::iterator VIT;
4 int siz[MAXN], max_son[MAXN], pa[MAXN], dep[
    MAXN];
5 int link_top[MAXN], link[MAXN], cnt;
6 std::vector<int> >G[MAXN];
7 void find_max_son(int x){
8     siz[x]=1;
9     max_son[x]=-1;
10    for(VIT i=G[x].begin(); i!=G[x].end(); ++i){
11        if(*i==pa[x]) continue;
12        pa[*i]=x;

```

```

13        dep[*i]=dep[x]+1;
14        find_max_son(*i);
15        if(max_son[x]==-1 || siz[*i]>siz[max_son[x]
            ]]) max_son[x]=*i;
16        siz[x]+=siz[*i];
17    }
18 }
19 void build_link(int x, int top){
20     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 }

```

11.2 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[k][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     return pa[0][a];
29 }
30 }

```

11.3 link_cut_tree.cpp

```

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

```

```

44         }
45         rotate(x);
46     }
47 }
48 int access(int x){
49     int last=0;
50     while(x){
51         splay(x);
52         node[x].ch[1]=last;
53         up(x);
54         last=x;
55         x=node[x].pa;
56     }
57     return last; // 回傳access後splay tree的根
58 }
59 void access(int x, bool is=0) { // is=0就是一般
    的access
60     int last=0;
61     while(x){
62         splay(x);
63         if(is&&!node[x].pa){
64             // printf("%d\n", max(node[last].ma, node
                [node[x].ch[1]].ma));
65         }
66         node[x].ch[1]=last;
67         up(x);
68         last=x;
69         x=node[x].pa;
70     }
71 }
72 void query_edge(int u, int v){
73     access(u);
74     access(v, 1);
75 }
76 void make_root(int x){
77     access(x), splay(x);
78     node[x].rev^=1;
79 }
80 void make_root(int x){
81     node[access(x)].rev^=1;
82     splay(x);
83 }
84 void cut(int x, int y){
85     make_root(x);
86     access(y);
87     splay(y);
88     node[y].ch[0]=0;
89     node[x].pa=0;
90 }
91 void cut_parents(int x){
92     access(x);
93     splay(x);
94     node[node[x].ch[0]].pa=0;
95     node[x].ch[0]=0;
96 }
97 void link(int x, int y){
98     make_root(x);
99     node[x].pa=y;
100 }
101 int find_root(int x){
102     x=access(x);
103     while(node[x].ch[0]) x=node[x].ch[0];
104     splay(x);
105     return x;
106 }

```

```

107 int query(int u,int v){
108 //傳回uv路徑splay tree的根結點
109 //這種寫法無法求LCA
110 make_root(u);
111 return access(v);
112 }
113 int query_lca(int u,int v){
114 //假設求鏈上點權的總和・sum是子樹的權重和・
115     data是節點的權重
116     access(u);
117     int lca=access(v);
118     splay(u);
119     if(u==lca){
120         //return node[lca].data+node[node[lca].
121             ch[1]].sum
122     }else{
123         //return node[lca].data+node[node[lca].
124             ch[1]].sum+node[u].sum
125     }
126 }
127 struct EDGE{
128     int a,b,w;
129 }e[10005];
130 int n;
131 vector<pair<int ,int > >G[10005];
132 //first表示子節點・second表示邊的編號
133 int pa[10005],edge_node[10005];
134 //pa是父母節點・暫存用的・edge_node是每個編
135     被存在哪個點裡面的陣列
136 void bfs(int root){
137 //在建構的時候把每個點都設成一個splay tree・
138     不會壞掉
139 queue<int > q;
140 for(int i=1;i<=n;++i)pa[i]=0;
141 q.push(root);
142 while(q.size()){
143     int u=q.front();
144     q.pop();
145     for(int i=0;i<(int)G[u].size();++i){
146         int v=G[u][i].first;
147         if(v!=pa[u]){
148             pa[v]=u;
149             node[v].pa=u;

```

```

145     node[v].data=e[G[u][i].second].w;
146     edge_node[G[u][i].second]=v;
147     up(v);
148     q.push(v);
149 }
150 }
151 }
152 }
153 void change(int x,int b){
154     splay(x);
155     //node[x].data=b;
156     up(x);
157 }

```

11.4 POJ_tree.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define MAXN 10005
4 int n,k;
5 vector<pair<int,int> >g[MAXN];
6 int size[MAXN];
7 bool vis[MAXN];
8 inline void init(){
9     for(int i=0;i<=n;++i){
10         g[i].clear();
11         vis[i]=0;
12     }
13 }
14 void get_dis(vector<int> &dis,int u,int pa,
15     int d){
16     dis.push_back(d);
17     for(size_t i=0;i<g[u].size();++i){
18         int v=g[u][i].first,w=g[u][i].second;
19         if(v!=pa&&!vis[v])get_dis(dis,v,u,d+w);
20     }
21 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,
34     const int sz){
35     size[u]=1;//找樹重心・second是重心
36     pair<int,int> res(INT_MAX,-1);
37     int ma=0;
38     for(size_t i=0;i<g[u].size();++i){
39         int v=g[u][i].first;
40         if(v==pa||vis[v])continue;
41         res=min(res,tree_centroid(v,u,sz));
42         size[u]+=size[v];
43         ma=max(ma,size[v]);
44     }
45     ma=max(ma,sz-size[u]);
46     return min(res,make_pair(ma,u));
47 }
48 int tree_DC(int u,int sz){
49     int center=tree_centroid(u,-1,sz).second;
50     int ans=cal(center,0);
51     vis[center]=1;
52     for(size_t i=0;i<g[center].size();++i){
53         int v=g[center][i].first,w=g[center][i].
54             second;
55         if(vis[v])continue;
56         ans-=cal(v,w);
57         ans+=tree_DC(v,size[v]);
58     }
59     return ans;
60 }
61 int main(){
62     init();
63     while(scanf("%d%d",&n,&k),n||k){
64         int u,v,w;
65         scanf("%d%d%d",&u,&v,&w);
66         g[u].push_back(make_pair(v,w));
67         g[v].push_back(make_pair(u,w));
68     }
69     printf("%d\n",tree_DC(1,n));

```

```

69     }
70     return 0;
71 }

```

12 zformula

12.1 formula.tex

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

連通圖
 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 Cv(G)
 最小邊覆蓋設為 Ce(G)
 對於任意連通圖
 I(G)+Cv(G)=|V|
 M(G)+Ce(G)=|V|

對於連通二分圖
 I(G)=Cv(G)
 M(G)=Ce(G)

最大團 = 補圖的最大獨立集

公式

$$\sum_{d|n} \phi(n) = n$$

$$\sum_{d|n} \mu(n) = (n == 1)$$

$$g(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d) * g(n/d)$$
 Catalannumber : $(2n)! / n! / n! / (n+1)$
 Harmonicseries $H_n = \ln(n) + \gamma + 1/(2n) - 1/(12n) + 1/(120n^3) - \dots$
 $\gamma = 0.57721566490153286060651209008240243104215\dots$
 $i - th \text{ graycode} : i^{(i > 1)}$
 $SG(A+B) = SG(A) \oplus SG(B)$

Count on a tree
 1. Rooted tree :S

ACM ICPC Team

Reference - NTHU

Jinkela

Contents

1 Computational_Geometry	1				
1.1 Geometry.cpp	1				
1.2 SmallestCircle.cpp	3				
1.3 最近點對.cpp	3				
1.4 浮點數誤差模板.cpp	3				
2 Data_Structure	3				
2.1 DLX.cpp	3				
2.2 Dynamic_KD_tree.cpp	4				
2.3 kd_tree_replace_segment_tree.cpp .	4				
2.4 persistent_segment_tree.cpp	5				
2.5 reference_point.cpp	5				
2.6 skew_heap.cpp	5				
2.7 split_merge.cpp	6				
2.8 treap.cpp	6				
2.9 操作分治.cpp	6				
2.10 整體二分.cpp	6				
3 default	6				
3.1 debug.cpp	6				
3.2 ext.cpp	6				
		3.3 IncStack.cpp	7		
		3.4 input.cpp	7		
		4 Flow	7		
		4.1 dinic.cpp	7		
		4.2 ISAP_with_cut.cpp	7		
		4.3 MinCostMaxFlow.cpp	7		
		5 Graph	8		
		5.1 Augmenting_Path.cpp	8		
		5.2 Augmenting_Path_multiple.cpp . . .	8		
		5.3 blossom_matching.cpp	8		
		5.4 graphISO.cpp	8		
		5.5 KM.cpp	8		
		5.6 MaximumClique.cpp	9		
		5.7 Minimum_General_Weighted_Matching.cpp	9		
		5.8 Rectilinear_Steiner_tree.cpp . . .	9		
		5.9 treeISO.cpp	9		
		5.10 一般圖最大權匹配.cpp	10		
		5.11 全局最小割.cpp	11		
		5.12 最小樹形圖 _ 朱劉.cpp	11		
		6 language	11		
		6.1 CNF.cpp	11		
		6.2 earley.cpp	11		
		7 Linear_Programming	12		
		7.1 最大密度子圖.cpp	12		
		8 Number_Theory	13		
		8.1 basic.cpp	13		
		8.2 bit_set.cpp	13		
		8.3 cantor_expansion.cpp	13		
		8.4 FFT.cpp	13		
		8.5 find_real_root.cpp	14		
		8.6 LinearCongruence.cpp	14		
		8.7 Lucas.cpp	14		
		8.8 Matrix.cpp	14		
		8.9 MillerRobin.cpp	14		
		8.10 NTT.cpp	15		
		8.11 外星模運算.cpp	15		
		8.12 模運算模板.cpp	15		
		8.13 質因數分解.cpp	15		
		9 String	16		
		9.1 AC 自動機.cpp	16		
		9.2 hash.cpp	16		
		9.3 KMP.cpp	16		
		9.4 manacher.cpp	16		
		9.5 minimal_string_rotation.cpp	16		
		9.6 suffix_array_lcp.cpp	16		
		9.7 Z.cpp	17		
		10 Tarjan	17		
		10.1 dominator_tree.cpp	17		
		10.2 tnfsb017_2_sat.cpp	17		
		10.3 橋連通分量.cpp	17		
		10.4 雙連通分量 & 割點.cpp	18		
		11 Tree_problem	18		
		11.1 HeavyLight.cpp	18		
		11.2 LCA.cpp	18		
		11.3 link_cut_tree.cpp	18		
		11.4 POJ_tree.cpp	19		
		12 zformula	19		
		12.1 formula.tex	19		