

# 1 Computational\_Geometry

## 1.1 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

## 1.2 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 template<typename T>
29 struct line{
30     line(){ }
31     point<T> p1,p2;
32     T a,b,c; //ax+by+c=0
33     line(const point<T>&x,const point<T>&y):p1(
34         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     T dis2(const point<T> &p,bool is_segment
47         =0) const{ //點跟直線/線段的距離平方
```

```
    point<T> v=p2-p1,v1=p-p1;
48     if(is_segment){
49         point<T> v2=p-p2;
50         if(v.dot(v1)<=0) return v1.abs2();
51         if(v.dot(v2)>=0) return v2.abs2();
52     }
53     T tmp=v.cross(v1);
54     return tmp*tmp/v.abs2();
55 }
56 point<T> projection(const point<T> &p)
57     const{ //點對直線的投影
58     point<T> n=(p2-p1).normal();
59     return p-n*(p-p1).dot(n)/n.abs2();
60 }
61 point<T> mirror(const point<T> &p) const{ //
62     點對直線的鏡射
63     //要先呼叫 pton 轉成一般式
64     point<T> ans;
65     T d=a*a+b*b;
66     ans.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/
67         d;
68     ans.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/
69         d;
70     return ans;
71 }
72 bool equal(const line &l) const{ //直線相等
73     return cross(l.p1)==0&&cross(l.p2)==0;
74 }
75 bool parallel(const line &l) const{ //直線平
76     行
77     return (p1-p2).cross(l.p1-l.p2)==0;
78 }
79 bool cross_seg(const line &l) const{ //直線
80     是否交線段
81     return (p2-p1).cross(l.p1)*(p2-p1).cross(
82         l.p2)<=0;
83 }
84 char line_intersect(const line &l) const{ //
85     直線相交情況，-1無限多點，1交於一點，0
86     不相交
87     return parallel(l)?(cross(l.p1)==0?-1:0)
88         :1;
89 }
90 char seg_intersect(const line &l) const{ //
91     線段相交情況，-1無限多點，1交於一點，0
92     不相交
93     T c1=(p2-p1).cross(l.p1-p1);
94     T c2=(p2-p1).cross(l.p2-p1);
95     T c3=(l.p2-l.p1).cross(p1-l.p1);
96     T c4=(l.p2-l.p1).cross(p2-l.p1);
97     if(c1==0&&c2==0){
98         if(p1==l.p1&&(p2-p1).dot(l.p2)<=0)
99             return 1;
100        if(p1==l.p2&&(p2-p1).dot(l.p1)<=0)
101            return 1;
102        if(p2==l.p1&&(p1-p2).dot(l.p2)<=0)
103            return 1;
104        if(p2==l.p2&&(p1-p2).dot(l.p1)<=0)
105            return 1;
106        return -1;
107    }else if(c1*c2<=0&&c3*c4<=0) return 1;
108    return 0;
109 }
110 }
111 point<T> v=p2-p1,v1=p-p1;
112 if(is_segment){
113     point<T> v2=p-p2;
114     if(v.dot(v1)<=0) return v1.abs2();
115     if(v.dot(v2)>=0) return v2.abs2();
116 }
117 T tmp=v.cross(v1);
118 return tmp*tmp/v.abs2();
119 }
120 point<T> projection(const point<T> &p)
121     const{ //點對直線的投影
122     point<T> n=(p2-p1).normal();
123     return p-n*(p-p1).dot(n)/n.abs2();
124 }
125 point<T> mirror(const point<T> &p) const{ //
126     點對直線的鏡射
127     //要先呼叫 pton 轉成一般式
128     point<T> ans;
129     T d=a*a+b*b;
130     ans.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/
131         d;
132     ans.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/
133         d;
134     return ans;
135 }
136 bool equal(const line &l) const{ //直線相等
137     return cross(l.p1)==0&&cross(l.p2)==0;
138 }
139 bool parallel(const line &l) const{ //直線平
140     行
141     return (p1-p2).cross(l.p1-l.p2)==0;
142 }
```

```
143 point<T> line_intersection(const line &l)
144     const{ //直線交點
145     point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
146     //if(a.cross(b)==0) return INF;
147     return p1+a*s.cross(b)/a.cross(b);
148 }
149 point<T> seg_intersection(const line &l)
150     const{ //線段交點
151     T c1=(p2-p1).cross(l.p1-p1);
152     T c2=(p2-p1).cross(l.p2-p1);
153     T c3=(l.p2-l.p1).cross(p1-l.p1);
154     T c4=(l.p2-l.p1).cross(p2-l.p1);
155     if(c1==0&&c2==0){
156         if(p1==l.p1&&(p2-p1).dot(l.p2)<=0)
157             return p1;
158         if(p1==l.p2&&(p2-p1).dot(l.p1)<=0)
159             return p1;
160         if(p2==l.p1&&(p1-p2).dot(l.p2)<=0)
161             return p2;
162         if(p2==l.p2&&(p1-p2).dot(l.p1)<=0)
163             return p2;
164     }else if(c1*c2<=0&&c3*c4<=0) return
165         line_intersection(l);
166     //return INF;
167 }
168 template<typename T>
169 struct polygon{
170     polygon(){ }
171     vector<point<T> > p; //逆時針順序
172     T area() const{ //面積
173         T ans=0;
174         for(int i=p.size()-1,j=0;j<(int)p.size()
175             ;i=j++){
176             ans+=p[i].cross(p[j]);
177         }
178         return ans/2;
179     }
180     point<T> center_of_mass() const{ //重心
181         T cx=0,cy=0,w=0;
182         for(int i=p.size()-1,j=0;j<(int)p.size()
183             ;i=j++){
184             T a=p[i].cross(p[j]);
185             cx+=(p[i].x*p[j].x)*a;
186             cy+=(p[i].y*p[j].y)*a;
187             w+=a;
188         }
189         return point<T>(cx/3/w,cy/3/w);
190     }
191     char ahas(const point<T> &t) const{ //點是否
192         在簡單多邊形內，是的話回傳1，在邊上回
193         傳-1，否則回傳0
194     bool c=0;
195     for(int i=0,j=p.size()-1;i<p.size();i=
196         ++j){
197         if(line<T>(p[i],p[j]).point_on_segment
198             (t)) return -1;
199         else if((p[i].y>t.y)!=p[j].y>t.y)&&
200             t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j]
201                 .y-p[i].y)+p[i].x)
202             c=!c;
203     }
204     return c;
205 }
206 char point_in_convex(const point<T> &x)
207     const{
```

```
143 int l=1,r=(int)p.size()-2;
144 while(l<=r){ //點是否在凸多邊形內，是的話
145     回傳1，在邊上回傳-1，否則回傳0
146     int mid=(l+r)/2;
147     T a1=(p[mid]-p[0]).cross(x-p[0]);
148     T a2=(p[mid+1]-p[0]).cross(x-p[0]);
149     if(a1>=0&&a2<=0){
150         T res=(p[mid+1]-p[mid]).cross(x-p[
151             mid]);
152         return res>0?1:(res>0?-1:0);
153     }else if(a1<0) r=mid-1;
154     else l=mid+1;
155 }
156 return 0;
157 }
158 polygon cut(const line<T> &l) const{ //凸包
159     對直線切割，得到直線L左側的凸包
160     polygon ans;
161     for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
162         if(1.cross(p[i])>=0){
163             ans.p.push_back(p[i]);
164             if(1.cross(p[j])<0)
165                 ans.p.push_back(1.
166                     line_intersection(line<T>(p[i
167                         ],p[j])));
168             }else if(1.cross(p[j])>0)
169                 ans.p.push_back(1.line_intersection(
170                     line<T>(p[i],p[j])));
171         }
172     }
173     return ans;
174 }
175 static bool graham_cmp(const point<T> &a,
176     const point<T> &b){
177     return (a.x<b.x)|| (a.x==b.x&&a.y<b.y); //
178     凸包排序函數
179 }
180 void graham(vector<point<T> > &s){ //凸包
181     sort(s.begin(),s.end(),graham_cmp);
182     p.resize(s.size()+1);
183     int m=0;
184     for(int i=0;i<(int)s.size();i++){
185         while(m>2&&(p[m-1]-p[m-2]).cross(s[i
186             ]-p[m-2])<=0)--m;
187         p[m++]=s[i];
188     }
189     for(int i=s.size()-2,t=m+1;i>0;--i){
190         while(m>2&&(p[m-1]-p[m-2]).cross(s[i
191             ]-p[m-2])<=0)--m;
192         p[m++]=s[i];
193     }
194     if(s.size())>1--m;
195     p.resize(m);
196 }
197 inline static char sign(const point<T> &t){
198     return (t.y==0?t.x:t.y)<0;
199 }
200 inline static bool angle_cmp(const line<T>
201     &A,const line<T> &B){
202     point<T> a=A.p2-A.p1,b=B.p2-B.p1;
203     return sign(a)<sign(b)|| (sign(a)==sign(b)
204         &&a.cross(b)>0);
205 }
206 int halfplane_intersection(vector<line<T>
207     > &s){ //半平面交
```

```

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

```

```

403     for(int j=fc.size()-1;j>=0;--j){
404         if(outside(i,j)){
405             dfs(i,j);
406             break;
407         }
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);
}

```

### 1.3 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

```

```

28 Circle SmallestCircle(std::vector<Circle::p>
29     &p){
30     int n=p.size();
31     if(n==1) return (Circle){p[0],0.0};
32     if(n==2) return TwoPointCircle(p[0],p
33         [1]);
34     random_shuffle(p.begin(),p.end());
35     Circle c = {p[0],0.0};
36     for(int i=0;i<n;++i){
37         if(c.incircle(p[i])) continue;
38         c=Circle{p[i],0.0};
39         for(int j=0;j<i;++j){
40             if(c.incircle(p[j])) continue;
41             c=TwoPointCircle(p[i],p[j]);
42             for(int k=0;k<j;++k){
43                 if(c.incircle(p[k]))
44                     continue;
45                 c=outcircle(p[i],p[j],p[k]);
46             }
47         }
48     }
49     return c;
50 }

```

### 1.4 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 }

```

### 1.5 浮點數誤差模板.cpp

```

1 const double EPS=1e-9;
2 struct Double{
3     double d;
4     Double(double d=0):d(d){}
5     bool operator <(const Double &b)const{
6         return d-b.d<-EPS;}
7     bool operator >(const Double &b)const{
8         return d-b.d>EPS;}
9     bool operator ==(const Double &b)const{
10         return fabs(d-b.d)<=EPS;}
11     bool operator !=(const Double &b)const{
12         return fabs(d-b.d)>EPS;}
13     bool operator <=(const Double &b)const{
14         return d-b.d<=EPS;}
15     bool operator >=(const Double &b)const{
16         return d-b.d>=-EPS;}
17     operator double()const{return d;}
18 };

```

## 2 Data\_Structure

### 2.1 DLX.cpp

```

1 #define MAXN 4100
2 #define MAXM 1030
3 #define MAXND 16390
4 struct DLX{
5     int n,m,sz,ansd;/*高是n·寬是m的稀疏矩陣
6     int S[MAXM],H[MAXM];
7     int row[MAXND],col[MAXND];/*每個節點代表的
8     列跟行
9     int L[MAXND],R[MAXND],U[MAXND],D[MAXND];
10     vector<int> ans,anst;
11     void init(int _n,int _m){
12         n=_n,m=_m;
13         for(int i=0;i<=m;++i){
14             U[i]=D[i]=i,L[i]=i-1,R[i]=i+1;
15             S[i]=0;
16         }
17         R[m]=0,L[0]=m;
18         sz=m,ansd=INT_MAX;/*ansd存最優解的個數
19         for(int i=1;i<=n;++i)H[i]=-1;
20     }
21     void add(int r,int c){
22         ++S[col[++sz]=c];
23         row[sz]=r;
24         D[sz]=D[c],U[D[c]]=sz,U[sz]=c,D[c]=sz;
25         if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
26         else R[sz]=R[H[r]],L[R[H[r]]]=sz,L[sz]=H
27             [r],R[H[r]]=sz;
28     }
29     #define DFOR(i,A,S) for(int i=A[s];i!=s;i=
30         A[i])
31     void remove(int c){/*刪除第c行和所有當前覆
32         蓋到第c行的列
33         L[R[c]]=L[c],R[L[c]]=R[c];/*這裡刪除第c
34         行·若有些行不需要處理可以在開始時呼
35         叫他
36         DFOR(i,D,c)DFOR(j,R,i){U[D[j]]=U[j],D[U[
37             j]]=D[j],--S[col[j]]};
38     }
39 }

```

```

31 }
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 }
38 void remove2(int nd){/*刪除nd所在的行當前
39     所有點(包括虛擬節點)·只保留nd
40     DFOR(i,D,nd)L[R[i]]=L[i],R[L[i]]=R[i];
41 }
42 void restore2(int nd){/*刪除nd所在的行當前
43     所有點·為remove2的逆操作
44     DFOR(i,U,nd)L[R[i]]=R[L[i]]=i;
45 }
46 bool vis[MAXM];
47 int h(){/*估價函數 for IDA*
48     int res=0;
49     memset(vis,0,sizeof(vis));
50     DFOR(i,R,0)if(!vis[i]){
51         vis[i]=1;
52         ++res;
53         DFOR(j,D,i)DFOR(k,R,j)vis[col[k]]=1;
54     }
55     return res;
56 }
57 bool dfs(int d){/*for精確覆蓋問題
58     if(d+h()>=ansd)return 0;/*找最佳解用·找
59     任意解可刪掉
60     if(!R[0]){ansd=d;return 1;}
61     int c=R[0];
62     DFOR(i,R,0)if(S[i]<S[c])c=i;
63     remove(c);
64     DFOR(i,D,c){
65         ans.push_back(row[i]);
66         DFOR(j,R,i)remove2(col[j]);
67         if(dfs(d+1))return 1;
68         ans.pop_back();
69         DFOR(j,L,i)restore2(col[j]);
70     }
71     restore(c);
72     return 0;
73 }
74 void dfs2(int d){/*for最小重複覆蓋問題
75     if(d+h()>=ansd)return;
76     if(!R[0]){ansd=d;ans=anst;return;}
77     int c=R[0];
78     DFOR(i,R,0)if(S[i]<S[c])c=i;
79     DFOR(i,D,c){
80         ans.push_back(row[i]);
81         remove2(i);
82         DFOR(j,R,i)remove2(j),--S[col[j]];
83         dfs2(d+1);
84         ans.pop_back();
85         DFOR(j,L,i)restore2(j),++S[col[j]];
86         restore2(i);
87     }
88 }
89 bool exact_cover(){/*解精確覆蓋問題
90     ans.clear();/*答案
91     return dfs(0);
92 }
93 void min_cover(){/*解最小重複覆蓋問題
94 }

```

```

89 anst.clear();//暫存用，答案還是存在ans裡
90 dfs2(0);
91 }
92 #undef DFOR
93 };

```

## 2.2 Dynamic\_KD\_tree.cpp

```

1 template<typename T,size_t kd>//有kd個維度
2 class kd_tree{
3 public:
4     struct point{
5         T d[kd];
6         T dist(const point &x)const{
7             T ret=0;
8             for(size_t i=0;i<kd;++i)ret+=std::
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    private:
21        struct node{
22            node *l,*r;
23            point pid;
24            int s;
25            node(const point &p):l(0),r(0),pid(p),
26                s(1){}
27            ~node(){delete l;delete r;}
28            void up(){s=(l?l->s:0)+1+(r?r->s:0);}
29        }*root;
30        const double alpha,loga;
31        const T INF;//記得要給INF，表示極大值
32        int maxn;
33        struct __cmp{
34            int sort_id;
35            bool operator()(const node*x,const
36                node*y)const{
37                return operator()(x->pid,y->pid);
38            }
39            bool operator()(const point &x,const
40                point &y)const{
41                if(x.d[sort_id]!=y.d[sort_id])
42                    return x.d[sort_id]<y.d[sort_id];
43                for(size_t i=0;i<kd;++i)
44                    if(x.d[i]!=y.d[i])return x.d[i]<y.
45                        d[i];
46                return 0;
47            }
48        }cmp;
49        int size(node *o){return o?o->s:0;}
50        std::vector<node*> A;
51        node* build(int k,int l,int r){
52            if(l>r)return 0;
53            if(k==kd)k=0;

```

```

54            int mid=(l+r)/2;
55            cmp.sort_id=k;
56            std::nth_element(A.begin()+l,A.begin()+
57                +mid,A.begin()+r+1,cmp);
58            node *ret=A[mid];
59            ret->l=build(k+1,l,mid-1);
60            ret->r=build(k+1,mid+1,r);
61            ret->up();
62            return ret;
63        }
64        bool isbad(node*o){
65            return size(o->l)>alpha*o->s||size(o->
66                r)>alpha*o->s;
67        }
68        void flatten(node *u,typename std::
69            vector<node*>::iterator &it){
70            if(!u)return;
71            flatten(u->l,it);
72            *it=u;
73            flatten(u->r,++it);
74        }
75        void rebuild(node*&u,int k){
76            if((int)A.size()<u->s)A.resize(u->s);
77            typename std::vector<node*>::iterator
78                it=A.begin();
79            flatten(u,it);
80            u=build(k,0,u->s-1);
81        }
82        bool insert(node*&u,int k,const point &
83            x,int dep){
84            if(!u){
85                u=new node(x);
86                return dep<=0;
87            }
88            ++u->s;
89            cmp.sort_id=k;
90            if(insert(cmp(x,u->pid)?u->l:u->r,(k
91                +1)%kd,x,dep-1)){
92                if(!isbad(u))return 1;
93                rebuild(u,k);
94            }
95            return 0;
96        }
97        node *findmin(node*o,int k){
98            if(!o)return 0;
99            if(cmp.sort_id==k)return o->l?findmin(
100                o->l,(k+1)%kd):0;
101            node *l=findmin(o->l,(k+1)%kd);
102            node *r=findmin(o->r,(k+1)%kd);
103            if(l&&l->r)return cmp(l,o)?l:0;
104            if(!l&&r)return cmp(r,o)?r:0;
105            if(!l&&l->r)return 0;
106            if(cmp(l,r))return cmp(l,o)?l:0;
107            return cmp(r,o)?r:0;
108        }
109        bool erase(node *&u,int k,const point &
110            x){
111            if(!u)return 0;
112            if(u->pid==x){
113                if(u->r){
114                    else if(u->l){
115                        u->r=u->l;
116                        u->l=0;
117                    }else{
118                        delete u;
119                        u=0;

```

```

120            return 1;
121        }
122        --u->s;
123        cmp.sort_id=k;
124        u->pid=findmin(u->r,(k+1)%kd)->pid;
125        return erase(u->r,(k+1)%kd,u->pid);
126    }
127    cmp.sort_id=k;
128    if(erase(cmp(x,u->pid)?u->l:u->r,(k+1)
129        %kd,x)){
130        --u->s;return 1;
131    }else return 0;
132    }
133    T heuristic(const T h[])const{
134        T ret=0;
135        for(size_t i=0;i<kd;++i)ret+=h[i];
136        return ret;
137    }
138    int qM;
139    std::priority_queue<std::pair<T,point >
140        >pQ;
141    void nearest(node *u,int k,const point &
142        x,T *h,T &mndist){
143        if(u==0||heuristic(h)>mndist)return;
144        T dist=u->pid.dist(x),old=h[k];
145        /*mndist=std::min(mndist,dist);*/
146        if(dist<mndist){
147            pQ.push(std::make_pair(dist,u->pid));
148            if((int)pQ.size()==qM+1)
149                mndist=pQ.top().first,pQ.pop();
150        }
151        if(x.d[k]<u->pid.d[k]){
152            nearest(u->l,(k+1)%kd,x,h,mndist);
153            h[k]=std::abs(x.d[k]-u->pid.d[k]);
154            nearest(u->r,(k+1)%kd,x,h,mndist);
155        }else{
156            nearest(u->r,(k+1)%kd,x,h,mndist);
157            h[k]=std::abs(x.d[k]-u->pid.d[k]);
158            nearest(u->l,(k+1)%kd,x,h,mndist);
159        }
160        h[k]=old;
161    }
162    std::vector<point> in_range;
163    void range(node *u,int k,const point&mi,
164        const point&ma){
165        if(!u)return;
166        bool is=1;
167        for(int i=0;i<kd;++i)
168            if(u->pid.d[i]<mi.d[i]||ma.d[i]<u->
169                pid.d[i]){
170                is=0;break;
171            }
172        if(is)in_range.push_back(u->pid);
173        if(mi.d[k]<u->pid.d[k])range(u->l,(k
174            +1)%kd,mi,ma);
175        if(ma.d[k]>u->pid.d[k])range(u->r,(k
176            +1)%kd,mi,ma);
177    }
178    public:
179    kd_tree(const T &INF,double a=0.75):root(
180        (0,alpha(a),loga(log2(1.0/a))),INF(
181            INF),maxn(1){}
182    ~kd_tree(){delete root;}
183    void clear(){delete root;root=0,maxn=1;}
184    void build(int n,const point *p){

```

```

185        delete root,A.resize(maxn=n);
186        for(int i=0;i<n;++i)A[i]=new node(p[i]
187            );
188        root=build(0,0,n-1);
189        void insert(const point &x){
190            insert(root,0,x,__lg(size(root))/loga
191                );
192            if(root->s>maxn)maxn=root->s;
193        }
194        bool erase(const point &p){
195            bool d=erase(root,0,p);
196            if(root&&root->s<alpha*maxn)rebuild();
197            return d;
198        }
199        void rebuild(){
200            if(root)rebuild(root,0);
201            maxn=root->s;
202        }
203        T nearest(const point &x,int k){
204            qM=k;
205            T mndist=INF,h[kd]={};
206            nearest(root,0,x,h,mndist);
207            mndist=pQ.top().first;
208            pQ=std::priority_queue<std::pair<T,
209                point >>();
210            return mndist;//回傳離x第k近的點的距離
211        }
212        const std::vector<point> &range(const
213            point&mi,const point&ma){
214            in_range.clear();
215            range(root,0,mi,ma);
216            return in_range;//回傳介於mi到ma之間的
217                點vector
218        }
219        int size(){return root?root->s:0;}
220    };

```

## 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            s+=l->s;
19        }
20        if(r){
21            for(int i=0;i<kd;++i){
22                mi.d[i]=min(mi.d[i],r->mi.d[i]);

```



```

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

```

```

74     o->down();
75     return;
76 }
77 if(point_in_range(o,L,R)){
78     //這個點在(L,R)區間 · 但是他的左右子樹不
79     一定在區間中
80     //單點懶惰標記修改
81 }
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);

```

```

29     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 _RefCount{
5     T data;
6     int ref;
7     _RefCount(const T&d=0):data(d),ref(0){}
8 };
9 template<typename T>
10 struct ref_pointer{
11     _RefCount<T> *p;
12     ~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(_RefCount<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 _RefCount<T>(
36         nd));
37 }
38 }
39 struct P{
40     int a,b;
41     P(int A,int B):a(A),b(B){}
42 }p(2,3);
43 int main(){
44     ref_pointer<int>b=new_ref(int(5));
45     ref_pointer<int>a=new_ref(*b);
46     ref_pointer<P>c=new_ref(p);
47     return 0;
48 }

```

## 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;

```

```

14     if(cmp(a->data,b->data))return merge(b
15         ,a);
16     node *t=a->r;
17     a->r=a->l;
18     a->l=merge(b,t);
19     return a;
20 }
21 public:
22 skew_heap():root(0),_size(0){}
23 ~skew_heap(){delete root;}
24 void clear(){delete root,root=0,_size
25     =0;}
26 void join(skew_heap &o){
27     root=merge(root,o.root);
28     o.root=0;
29     _size+=o._size;
30     o._size=0;
31 }
32 void swap(skew_heap &o){
33     node *t=root;
34     root=o.root;
35     o.root=t;
36     int st=_size;
37     _size=o._size;
38     o._size=st;
39 }
40 void push(const T&data){
41     _size++;
42     root=merge(root,new node(data));
43 }
44 void pop(){
45     if(_size-->0){
46         node *tmd=merge(root->l,root->r);
47         root->l=root->r=0;
48         delete tmd;
49         root=tmd;
50 }
51 const T& top(){return root->data;}
52 int size(){return _size;}
53 bool empty(){return !_size;}
54 };

```

## 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=0;
8             split(o->l,a,b->l,k);
9         }else{
10            a=0;
11            split(o->r,a->r,b->r,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->ch[1];
83             else o=o->ch[0];
84             return cnt;
85         }
86     }
87     const T&kth(int k){
88         for(node *o=root;){
89             if(k<=o->ch[0]->s)o=o->ch[0];
90             else if(k==o->ch[0]->s+1)return o->data;
91             else k=k-o->ch[0]->s+1,o=o->ch[1];
92         }
93     }
94     const T&operator[](int k){
95         return kth(k);
96     }
97     const T&preorder(const T&data){
98         node *x=root,*y=0;
99         while(x->s){
100            if(x->data<data)y=x,x=x->ch[1];
101            else x=x->ch[0];
102            if(y)return y->data;
103            return data;
104        }
105    }
106    const T&successor(const T&data){
107        node *x=root,*y=0;
108        while(x->s){
109            if(data<x->data)y=x,x=x->ch[0];
110            else x=x->ch[1];
111            if(y)return y->data;
112            return data;
113        }
114    }
115    int size(){return root->s;}

```

```
109 };
```

## 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左滿足的;
7     vector<Item> right=vs-左;
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 template<typename I> void _DO(I&&x){cerr<<x
9     <<endl;}
10 template<typename I,typename...T> void _DO(I
11     &&x,T&&...tail){cerr<<x<<" ";_DO(tail
12     ...);}
13 #else
14 #define debug(...)
15 #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
4 //system
5 asm("mov %0, %%esp\n" :: "g"(mem+1000000));
6 //linux stack resize
7 #include <sys/resource.h>
8 void increase_stack(){
9     const rlim_t ks=64*1024*1024;
10    struct rlimit rl;
11    int res=getrlimit(RLIMIT_STACK,&rl);
12    if(!res&&rl.rlim_cur<ks){
13        rl.rlim_cur=ks;
14        res=setrlimit(RLIMIT_STACK,&rl);
15    }
16 }
```

### 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-
10 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));
```

```
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    }
71 }
```

### 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{
```

```
1 int v, pre;
2 T cap, flow, r;
3 edge(int v, int pre, T cap):v(v), pre(pre),
4     cap(cap), flow(0), r(cap){}
5 };
6 int g[MAXN];
7 vector<edge> e;
8 void init(int _n){
9     memset(g, -1, sizeof(int)*((n=_n)+1));
10    e.clear();
11 }
12 void add_edge(int u, int v, T cap, bool
13     directed=false){
14     e.push_back(edge(v, g[u], cap));
15     g[u]=e.size()-1;
16     e.push_back(edge(u, g[v], directed?0:cap));
17     ;
18     g[v]=e.size()-1;
19 }
20 T dfs(int u, int s, int t, T cur_flow=INF){
21     if(u==t) return cur_flow;
22     T tf=cur_flow, df;
23     for(int &i=cur[u]; ~i; i=e[i].pre){
24         if(e[i].r && d[u]==d[e[i].v]+1){
25             df=dfs(e[i].v, s, t, min(tf, e[i].r));
26             e[i].flow+=df;
27             e[i^1].flow-=df;
28             e[i].r-=df;
29             e[i^1].r+=df;
30             if(!((tf-=df)||d[s]==n)) return
31                 cur_flow-tf;
32         }
33     }
34     int mh=n;
35     for(int i=cur[u]=g[u]; ~i; i=e[i].pre){
36         if(e[i].r && d[e[i].v]<mh) mh=d[e[i].v];
37     }
38     if(!--gap[d[u]]) d[s]=n;
39     else ++gap[d[u]=++mh];
40     return cur_flow-tf;
41 }
42 T isap(int s, int t, bool clean=true){
43     memset(d, 0, sizeof(int)*(n+1));
44     memset(gap, 0, sizeof(int)*(n+1));
45     memcpy(cur, g, sizeof(int)*(n+1));
46     if(clean){
47         for(size_t i=0; i<e.size(); ++i){
48             e[i].flow=0;
49             e[i].r=e[i].cap;
50         }
51     }
52     T max_flow=0;
53     for(gap[0]=n; d[s]<n; ) max_flow+=dfs(s, s, t
54         );
55     return max_flow;
56 }
57 vector<int> cut_e; //最小割邊集
58 bool vis[MAXN];
59 void dfs_cut(int u){
60     vis[u]=1; //表示u屬於source的最小割集
61     for(int i=g[u]; ~i; i=e[i].pre){
62         if(e[i].flow<e[i].cap && !vis[e[i].v])
63             dfs_cut(e[i].v);
64     }
65 }
66 }
```

```
67 T min_cut(int s, int t){
68     T ans=isap(s, t);
69     memset(vis, 0, sizeof(bool)*(n+1));
70     dfs_cut(s, cut_e.clear());
71     for(int u=0; u<n; ++u){
72         if(vis[u]) for(int i=g[u]; ~i; i=e[i].pre
73             ){
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);
```

```

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;

```

### 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         }
20         else{
21             for(size_t j=0;j<match_list[v].size()
22                 ;++j){
23                 int next_u=match_list[v][j];
24                 if(dfs(next_u)){
25                     match_list[v][j]=u;
26                     return true;
27                 }
28             }
29         }
30     }
31     return false;
32 }
33 inline int max_match(){
34     for(int i=0;i<n2;++i)match_list[i].clear()
35     ;
36     int cnt=0;
37     for(int u=0;u<n1;++u){
38         memset(vis,0,sizeof(bool)*n2);
39         if(dfs(u))++cnt;
40     }
41     return cnt;
42 }

```

### 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 formula.tex

對於連通圖  $G$  最大獨立點集的大小設為  $I(G)$  最大匹配大小設為  $M(G)$  最小點覆蓋設為  $C_v(G)$  最小邊覆蓋設為  $C_e(G)$  對於任意連通圖  $I(G)+C_v(G)=|V|$   $M(G)+C_e(G)=|V|$  對於連通二分圖  $I(G)=C_v(G)$   $M(G)=C_e(G)$  最大團 = 補圖的最大獨立集

### 5.5 graphISO.cpp

```

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

```

```

7     for(int i=0;i<n;++i){
8         f[0][i]=1;
9         g[i].clear();
10        rg[i].clear();
11    }
12 }
13 inline void add_edge(int u,int v){
14     g[u].push_back(v);
15     rg[v].push_back(u);
16 }
17 inline long long point_hash(int u){//O(N)
18     for(int t=1;t<=K;++t){
19         for(int i=0;i<n;++i){
20             f[t][i]=f[t-1][i]*A%P;
21             for(int j:g[i])f[t][i]=(f[t][i]+f[t-1][j]*B%P)%P;
22             for(int j:rg[i])f[t][i]=(f[t][i]+f[t-1][j]*C%P)%P;
23             if(i==u)f[t][i]+=D;//如果圖太大的話·
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;
37 }

```

### 5.6 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);

```



```

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

```

## 5.7 MaximumClique.cpp

```

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

```

```

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

```

## 5.8 Minimum\_General\_Weighted

```

1 struct Graph {
2     // Minimum General Weighted Matching (
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    bool SPFA(int u){
20        if (onstk[u]) return true;
21        stk.push_back(u);
22        onstk[u] = 1;
23        for (int v=0; v<n; v++){
24            if (u != v && match[u] != v && !onstk[
25                v]){
26                int m = match[v];
27                if (dis[m] > dis[u] - edge[v][m] +

```

```

28                stk.push_back(v);
29                if (SPFA(m)) return true;
30                stk.pop_back();
31                onstk[v] = 0;
32            }
33        }
34    }
35    onstk[u] = 0;
36    stk.pop_back();
37    return false;
38 }
39
40 int solve() {
41     // find a match
42     for (int i=0; i<n; i+=2){
43         match[i] = i+1;
44         match[i+1] = i;
45     }
46     for(;;){
47         int found = 0;
48         for (int i=0; i<n; i++){
49             dis[i] = onstk[i] = 0;
50             for (int i=0; i<n; i++){
51                 stk.clear();
52                 if (!onstk[i] && SPFA(i)){
53                     found = 1;
54                     while (stk.size()>2){
55                         int u = stk.back(); stk.pop_back
56                             ();
57                         int v = stk.back(); stk.pop_back
58                             ();
59                         match[u] = v;
60                         match[v] = u;
61                     }
62                 }
63             }
64             if (!found) break;
65         }
66         int ret = 0;
67         for (int i=0; i<n; i++)
68             ret += edge[i][match[i]];
69         ret /= 2;
70         return ret;
71     }
72 } graph;

```

## 5.9 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 }
10 point(){}
11 T dist(const point &p) const{
12     return std::abs(x-p.x)+std::abs(y-p.y);
13 };

```

```

14 inline bool cmpx(const point &a, const point
15     &b){
16     return a.x<b.x || (a.x==b.x && a.y<b.y);
17 }
18 struct edge{
19     int u, v;
20     T cost;
21     edge(int u, int v, const T&c):u(u),v(v),cost
22         (c){}
23     bool operator<(const edge&e) const{
24         return cost<e.cost;
25     }
26 };
27 struct bit_node{
28     T mi;
29     int id;
30     bit_node(const T&mi=INF, int id=-1):mi(mi),
31         id(id){}
32 };
33 std::vector<bit_node> bit;
34 inline void bit_update(int i, const T&data,
35     int id){
36     for(;;i-=i&(-i)){
37         if(data<bit[i].mi) bit[i]=bit_node(data,
38             id);
39     }
40 }
41 inline int bit_find(int i, int m){
42     bit_node x;
43     for(;;i+=i&(-i)){
44         if(bit[i].mi<x.mi) x=bit[i];
45     }
46     return x.id;
47 }
48 inline std::vector<edge> build_graph(int n,
49     point p[]){
50     std::vector<edge> e; //回傳的邊就可以用來求
51     //最小生成樹
52     for(int dir=0; dir<4; ++dir){ //4種座標變換
53         if(dir%2){
54             for(int i=0; i<n; ++i) std::swap(p[i].x, p
55                 [i].y);
56             } else if(dir==2){
57                 for(int i=0; i<n; ++i) p[i].x=-p[i].x;
58             }
59             std::sort(p, p+n, cmpx);
60             std::vector<T> ga(n), gb;
61             for(int i=0; i<n; ++i) ga[i]=p[i].y-p[i].x;
62             gb=ga;
63             std::sort(gb.begin(), gb.end());
64             gb.resize(std::unique(gb.begin(), gb.end
65                 ())-gb.begin());
66             int m=gb.size();
67             bit=std::vector<bit_node>(m+1);
68             for(int i=n-1; i>=0; --i){
69                 int pos=std::lower_bound(gb.begin(), gb
70                     .end(), ga[i])-gb.begin()+1;
71                 int ans=bit_find(pos, m);
72                 if(~ans) e.push_back(edge(p[i].id, p[ans
73                     ].id, p[i].dist(p[ans])));
74                 bit_update(pos, p[i].x+p[i].y, i);
75             }
76         }
77     }
78     return e;
79 }

```

## 5.10 treeISO.cpp

```

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

```

## 5.11 一般圖最大權匹配.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define INF INT_MAX
4 #define MAXN 400
5 struct edge{
6     int u,v,w;
7     edge(){
8         edge(int u,int v,int w):u(u),v(v),w(w){}
9     };
10    int n,n_x;
11    edge g[MAXN*2+1][MAXN*2+1];
12    int lab[MAXN*2+1];
13    int match[MAXN*2+1],slack[MAXN*2+1],st[MAXN
14        *2+1],pa[MAXN*2+1];
15    int flower_from[MAXN*2+1][MAXN+1],S[MAXN
16        *2+1],vis[MAXN*2+1];
17    vector<int> flower[MAXN*2+1];
18    queue<int> q;
19    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            ++i)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);

```

```

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

```

```

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

```

```

153 for(int b=n+1;b<=n_x;b++){
154     if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
155 }
156 for(int x=1;x<=n_x;x++){
157     if(st[x]==x&&slack[x]){
158         if(S[x]==-1)d=min(d,e_delta(g[slack[
159             x]][x]));
160         else if(S[x]==0)d=min(d,e_delta(g[
161             slack[x]][x])/2);
162     }
163 }
164 for(int u=1;u<=n;u++){
165     if(S[st[u]]==0){
166         if(lab[u]<=d)return 0;
167         lab[u]-=d;
168     }else if(S[st[u]]==1)lab[u]+=d;
169 }
170 for(int b=n+1;b<=n_x;b++){
171     if(st[b]==b){
172         if(S[st[b]]==0)lab[b]+=d*2;
173         else if(S[st[b]]==1)lab[b]-=d*2;
174     }
175 }
176 q=queue<int>();
177 for(int x=1;x<=n_x;x++){
178     if(st[x]==x&&slack[x]&&st[slack[x]]!=x
179         &&e_delta(g[slack[x]][x])<0)
180         if(on_found_edge(g[slack[x]][x]))
181             return true;
182 }
183 for(int b=n+1;b<=n_x;b++){
184     if(st[b]==b&&S[b]==1&&lab[b]==0)
185         expand_blossom(b);
186 }
187 return false;
188 }
189 pair<long long,int> weight_blossom(){
190     memset(match+1,0,sizeof(int)*n);
191     n_x=n;
192     int n_matches=0;
193     long long tot_weight=0;
194     for(int u=0;u<=n;u++)st[u]=u,flower[u].
195         clear();
196     int w_max=0;
197     for(int u=1;u<=n;u++){
198         for(int v=1;v<=n;v++){
199             flower_from[u][v]=(u==v?u:0);
200             w_max=max(w_max,g[u][v].w);
201         }
202     }
203     for(int u=1;u<=n;u++)lab[u]=w_max;
204     while(matching())n_matches++;
205     for(int u=1;u<=n;u++){
206         if(match[u]&&match[u]<u)
207             tot_weight+=g[u][match[u]].w;
208         return make_pair(tot_weight,n_matches);
209     }
210 }
211 void init_weight_graph(){
212     for(int u=1;u<=n;u++){
213         for(int v=1;v<=n;v++){
214             g[u][v]=edge(u,v,0);
215         }
216     }
217 }
218 int main(){
219     int m;
220     scanf("%d",&m);
221     init_weight_graph();
222     for(int i=0;i<m;i++){
223         int u,v,w;
224         scanf("%d%d%d",&u,&v,&w);
225         g[u][v].w=g[v][u].w=w;

```

```

212 }
213 printf("%lld\n", weight_blossom().first);
214 for(int u=1; u<n; ++u) printf("%d ", match[u]); puts("");
215 return 0;
216 }/* 7 20
217 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 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*/

```

## 5.12 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[ind]][nd[ind-1]]+g[nd[i]][nd[ind]];
31         }
32         return ans;
33     }
34 };

```

## 5.13 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                }
25            for(int u=0; u<n; ++u) //μL
26                if(u!=root && in[u]==INF) return -INF;
27            int cntnode=0;
28            memset(id, -1, sizeof(int)*N);
29            memset(vis, -1, sizeof(int)*N);
30            for(int u=0; u<n; ++u){
31                if(u!=root) ans+=in[u];
32                int v=u;
33                for(;; vis[v]!&u&id[v]==-1 && v!=root; v=E[pe[v]].u)
34                    if(v!=root && id[v]==-1){
35                        for(int x=E[pe[v]].u; x!=v; x=E[pe[x]].u)
36                            id[x]=cntnode;
37                        id[v]=cntnode++;
38                    }
39            }
40            if(!cntnode) break; //μL
41            for(int u=0; u<n; ++u) if(id[u]==-1) id[u]=cntnode++;
42            for(size_t i=0; i<E.size(); ++i){
43                int v=E[i].v;
44                E[i].u=id[E[i].u];
45                E[i].v=id[E[i].v];
46                if(E[i].u!=E[i].v) E[i].w-=in[v];
47            }
48            n=cntnode;
49            root=id[root];
50        }
51        return ans;
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 · 小寫字母為終端字元
10 vector<CNF> cnf;
11 inline void init(){
12     state=0;
13     rule.clear();
14     cnf.clear();
15 }
16 inline void add_to_cnf(char s, const string & p, int cost){
17     // 加入一個 s -> <p> 的文法 · 代價為 cost
18     if(rule.find(s)==rule.end()) rule[s]=state++;
19     for(auto c:p) if(rule.find(c)==rule.end()) rule[c]=state++;
20     if(p.size()==1){
21         cnf.push_back(CNF(rule[s], rule[p[0]], -1, cost));
22     }else{
23         int left=rule[s];
24         int sz=p.size();
25         for(int i=0; i<sz-2; ++i){
26             cnf.push_back(CNF(left, rule[p[i]], state, 0));
27             left=state++;
28         }
29         cnf.push_back(CNF(left, rule[p[sz-2]], rule[p[sz-1]], cost));
30     }
31 }
32 vector<long long> dp[MAXN][MAXN];
33 vector<bool> neg_INF[MAXN][MAXN]; // 如果花費是負的可能會有無限小的情形
34 inline void relax(int l, int r, const CNF &c, long long cost, bool neg_c=0){
35     if(!neg_INF[l][r][c.s] && (neg_INF[l][r][c.x] || cost<dp[l][r][c.s])){
36         dp[l][r][c.s]=0;
37         if(neg_c || neg_INF[l][r][c.x]){
38             dp[l][r][c.s]=0;
39             neg_INF[l][r][c.s]=true;
40         }else dp[l][r][c.s]=cost;
41     }
42     inline void bellman(int l, int r, int n){
43         for(int k=1; k<=state; ++k)
44             for(auto c:cnf)
45                 if(c.y==1) relax(l, r, c, dp[l][r][c.x]+c.cost, k==n);
46     }
47     inline void cyk(const vector<int> &tok){
48         for(int i=0; i<(int)tok.size(); ++i){
49             for(int j=0; j<(int)tok.size(); ++j){
50                 dp[i][j]=vector<long long>(state+1, INT_MAX);
51                 neg_INF[i][j]=vector<bool>(state+1, false);
52             }
53             dp[i][i][tok[i]]=0;
54             bellman(i, i, tok.size());
55         }
56         for(int r=1; r<(int)tok.size(); ++r){

```

```

57         for(int l=r-1; l>=0; --l){
58             for(int k=1; k<r; ++k)
59                 for(auto c:cnf)
60                     if(~c.y) relax(l, r, c, dp[l][k][c.x]+dp[k+1][r][c.y]+c.cost);
61             bellman(l, r, tok.size());
62         }
63     }
64 }

```

## 6.2 earley.cpp

```

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

```

```

46     }
47     cout<<" "<<"["<<start<<" "<<end<<"]"<<
48     endl;
49 }
50 struct Column{
51     Rule *term;
52     string value;
53     vector<State> s;
54     map<State, set<pair<State, State> > > div;
55     //div比較像一棵 左兄右子的樹
56     Column(Rule *r, const string &s):term(r),
57         value(s){
58         Column();
59         bool add(const State &st, int col){
60             if(div.find(st)==div.end()){
61                 div[st];
62                 s.push_back(st);
63                 s.back().end=col;
64                 return true;
65             }else return false;
66         }
67     };
68     inline vector<Column> lexer(string text){
69         //tokenize · 要自己寫 · 以下為範例
70         //他會把 input stream 變成 token stream ·
71         就是 (terminal, value) pair
72         vector<Column> token;
73         replace(text.begin(), text.end(), ' ', ' ');
74         stringstream ss(text);
75         while(ss>>text){
76             if(text=="a" || text=="of") continue;
77             if(text=="list"){
78                 token.push_back(Column(NameRule["(", "
79                 ("));
80             }else if(text=="and"){
81                 token.push_back(Column(NameRule["&", "
82                 &"));
83             }else token.push_back(Column(NameRule["T
84             ", text]));
85         }
86         return token;
87     }
88     vector<Column> table;
89     inline void predict(int col, Rule *rul){
90         for(size_t i=0; i<rul->p.size(); ++i){
91             table[col].add(State(rul, i, 0, col), col);
92         }
93     }
94     inline void scan(int col, const State &s, Rule
95     *r){
96         if(r!=table[col].term) return;
97         State ns(s.r, s.rid, s.dot_id+1, s.start);
98         table[col].add(ns, col);
99         table[col].div[ns].insert(make_pair(s,
100         State(r, col)));
101     }
102     inline pair<bool, State> parse(Rule *GAMMA,
103     const vector<Column> &token){
104         table.resize(token.size()+1);
105         for(size_t i=0; i<token.size(); ++i) table[i
106         +1]=Column(token[i]);
107         table[0]=Column();
108         table[0].add(State(GAMMA, 0, 0, 0), 0);
109         for(size_t i=0; i<table.size(); ++i){
110             for(size_t j=0; j<table[i].s.size(); ++j){
111                 State state=table[i].s[j];
112                 if(state.completed()) complete(i, state)
113                 ;
114             }else{
115                 Rule *term=state.next_term();
116                 if(term->p.size()>0) predict(i, term);
117                 else if(i+1<table.size()) scan(i+1,
118                 state, term);
119             }
120         }
121         for(size_t i=0; i<table.back().s.size(); ++i
122         ){
123             if(table.back().s[i].r==GAMMA&&table.
124             back().s[i].completed()){
125                 return make_pair(true, table.back().s[i
126                 ]);
127             }
128         }
129         return make_pair(false, State(0, -1));
130     }
131     struct node{//語法樹的節點
132         State s;
133         vector<vector<node*> > child; //vector<node
134         *>.size()>1 表示 ambiguous
135         node(const State &s):s(s){}
136         node(){}
137     };
138     struct State_end_cmp{
139         bool operator()(const State &a, const State
140         &b) const{
141             return a.end<b.end || (a.end==b.end&&a<b);
142         }
143     };
144     map<State, node*, State_end_cmp> cache;
145     vector<node*> node_set;
146     inline void init_cache(){
147         for(auto d:node_set) delete d;
148         cache.clear();
149         node_set.clear();
150     }
151     void build_tree(const State &s, node *pa,
152     bool amb=0){
153         if(cache.find(s)!=cache.end()){
154             pa->child.push_back(vector<node*>(1,
155             cache[s]));
156             return;
157         }
158         node *o;
159         if(s.completed()){
160             o=new node(s);
161             if(amb) pa->child.back().push_back(o);
162             else pa->child.push_back(vector<node
163             *>(1, o));
164         }else o=pa->child.back().back();
165         amb=0;
166         for(auto div:table[s.end].div[s]){
167             if(!amb) _build_tree(div.first, pa);
168             _build_tree(div.second, o, amb);
169             amb=1;
170         }
171         if(s.completed()) cache[s]=o;
172     }
173     inline node *build_tree(const State &s){
174         init_cache();
175         node o;
176         _build_tree(s, &o);
177         assert(o.child.size()==1);
178         assert(o.child.back().size()==1);
179         return o.child.back().back();
180     }
181     void print_tree(node *o, int dep=0){
182         cout<<string(dep, ' ')<<o->s.print();
183         for(auto div:o->child){
184             for(auto nd:div){
185                 print_tree(nd, dep+2);
186             }
187         }
188     }
189     //開始寫 code: 以下為加入語法的範例
190     inline Rule *get_my_Rule(){
191         Rule *S=add_rule("S"), *E=add_rule("E"), *L=
192         add_rule("L");
193         Rule *list=add_rule("(", *AND=add_rule("&")
194         ), *T=add_rule("T");
195         S->add({list, E});
196         S->add({list, L});
197         L->add({E, L});
198         L->add({E, AND, E});
199         E->add({T});
200         E->add({S});
201         Rule *GAMMA=add_rule("GAMMA"); //一定要有
202         gamma rule當作是最上層的語法
203         GAMMA->add({S});
204         return GAMMA;
205     }
206 }
207 };
208 vector<edge> E;
209 int n, m; // 1-base
210 T de[1MAXN], pv[1MAXN]; //每個點的邊權和和點權 (
211 有些題目會給)
212 void init(){
213     E.clear();
214     for(int i=1; i<=n; ++i) de[i]=pv[i]=0;
215 }
216 void add_edge(int u, int v, T w){
217     E.push_back(edge(u, v, w));
218     de[u]+=w, de[v]+=w;
219 }
220 T U; //二分搜的最大值
221 void get_U(){
222     U=0;
223     for(int i=1; i<=n; ++i) U+=2*pv[i];
224     for(size_t i=0; i<E.size(); ++i) U+=E[i].w;
225 }
226 ISAP<T> isap; //網路流
227 int s, t; //原匯點
228 void build(T L){
229     isap.init(n+2);
230     for(size_t i=0; i<E.size(); ++i){
231         isap.add_edge(E[i].u, E[i].v, E[i].w);
232     }
233     for(int v=1; v<=n; ++v){
234         isap.add_edge(s, v, U);
235         isap.add_edge(v, t, U+2*L-de[v]-2*pv[v]);
236     }
237 }
238 int main(){
239     while(~scanf("%d%d", &n, &m)){
240         if(!m){
241             puts("1\n1");
242             continue;
243         }
244         init();
245         int u, v;
246         for(int i=0; i<m; ++i){
247             scanf("%d%d", &u, &v);
248             add_edge(u, v, 1);
249         }
250         get_U();
251         s=n+1, t=n+2;
252         T l=0, r=U, k=1.0/(n*n);
253         while(r-l>k){ //二分搜最大值
254             T mid=(l+r)/2;
255             build(mid);
256             T res=(U*n-isap.isap(s, t))/2;
257             if(res>0) l=mid;
258             else r=mid;
259         }
260         build(l);
261         isap.min_cut(s, t);
262         vector<int> ans;
263         for(int i=1; i<=n; ++i){
264             if(isap.vis[i]) ans.push_back(i);
265         }
266         printf("%d\n", ans.size());
267         for(size_t i=0; i<ans.size(); ++i){
268             printf("%d\n", ans[i]);
269         }
270     }
271     return 0;
272 }

```

## 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     {}

```



71| }

## 8 Number\_Theory

### 8.1 basic.cpp

```

1 typedef long long int LL;
2 template<typename T>
3 void gcd(const T &a, const T &b, T &d, T &x, T &y){
4     if(!b) d=a, x=1, y=0;
5     else gcd(b, a%b, d, y, x), y-=x*(a/b);
6 }
7
8 const int MAXPRIME = 1000000;
9 int iscom[MAXPRIME], prime[MAXPRIME],
10 primecnt;
11 int phi[MAXPRIME], mu[MAXPRIME];
12 void sieve(void){
13     memset(iscom, 0, sizeof(iscom));
14     primecnt = 0;
15     phi[1] = mu[1] = 1;
16     for(int i=2; i<MAXPRIME; ++i) {
17         if(!iscom[i]) {
18             prime[primecnt++] = i;
19             mu[i] = -1;
20             phi[i] = i-1;
21         }
22         for(int j=0; j<primecnt; ++j) {
23             int k = i * prime[j];
24             if(k>=MAXPRIME) break;
25             iscom[k] = prime[j];
26             if(i%prime[j]==0) {
27                 mu[k] = 0;
28                 phi[k] = phi[i] * prime[j];
29                 break;
30             } else {
31                 mu[k] = -mu[i];
32                 phi[k] = phi[i] * (prime[j]-1);
33             }
34         }
35     }
36 }
37 bool g_test(const LL &g, const LL &p, const
38 vector<LL> &v) {
39     for(int i=0; i<v.size(); ++i)
40         if(modexp(g, (p-1)/v[i], p)==1)
41             return false;
42     return true;
43 }
44 LL primitive_root(const LL &p) {
45     if(p==2) return 1;
46     vector<LL> v;
47     Factor(p-1, v);
48     v.erase(unique(v.begin(), v.end()), v.end());
49     for(LL g=2; g<p; ++g)
50         if(g_test(g, p, v))
51             return g;
52     puts("primitive_root NOT FOUND");

```

```

52     return -1;
53 }
54
55 int Legendre(const LL &a, const LL &p) {
56     return modexp(a%p, (p-1)/2, p);
57 }
58 LL inv(const LL &a, const LL &n) {
59     LL d, x, y;
60     gcd(a, n, d, x, y);
61     return d==1 ? (x+n)%n : -1;
62 }
63 LL log_mod(const LL &a, const LL &b, const
64 LL &p) {
65     // a ^ x = b ( mod p )
66     int m=sqrt(p+.5), e=1;
67     LL v=inv(modexp(a, m, p), p);
68     map<LL, int> x;
69     x[1]=0;
70     for(int i=1; i<m; ++i) {
71         e = LLMul(e, a, p);
72         if(!x.count(e)) x[e] = i;
73     }
74     for(int i=0; i<m; ++i) {
75         if(x.count(b)) return i*m + x[b];
76         b = LLMul(b, v, p);
77     }
78     return -1;
79 }
80 LL Tonelli_Shanks(const LL &n, const LL &p) {
81     // x^2 = n ( mod p )
82     if(n==0) return 0;
83     if(Legendre(n, p)!=1) while(1) { puts("SQRT
84         ROOT does not exist"); }
85     int S = 0;
86     LL Q = p-1;
87     while( !(Q&1) ) { Q>>=1; ++S; }
88     if(S==1) return modexp(n%p, (p+1)/4, p);
89     LL z = 2;
90     for(; Legendre(z, p)!=-1; ++z)
91         LL c = modexp(z, Q, p);
92     LL R = modexp(n%p, (Q+1)/2, p), t = modexp(n
93         %p, Q, p);
94     int M = S;
95     while(1) {
96         if(t==1) return R;
97         LL b = modexp(c, 1<((M-i-1), p);
98         R = LLMul(R, b, p);
99         t = LLMul(LLmul(b, b, p), t, p);
100         c = LLMul(b, b, p);
101         M = i;
102     }
103     return -1;
104 }

```

### 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 #include<bits/stdc++.h>
2 using namespace std;
3 #define MAXN 11
4 int factorial[MAXN];
5 inline void init(){
6     factorial[0]=1;
7     for(int i=1; i<=MAXN; ++i) factorial[i]=
8         factorial[i-1]*i;
9 }
10 inline int encode(const std::vector<int> &s)
11 {
12     int n=s.size(), res=0;
13     for(int i=0; i<n; ++i){
14         int t=0;
15         for(int j=i+1; j<n; ++j)
16             if(s[j]<s[i]) ++t;
17         res+=t*factorial[n-i-1];
18     }
19     return res;
20 }
21 inline std::vector<int> decode(int a, int n){
22     std::vector<int> res;
23     std::vector<bool> vis(n, 0);
24     for(int i=n-1; i>=0; --i){
25         int t=a/factorial[i];
26         for(j=0; j<n; ++j)
27             if(!vis[j]){
28                 if(t==0) break;
29                 res.push_back(j);
30                 vis[j]=1;
31                 a%=factorial[i];
32             }
33     }
34     return res;
35 }
36 int main(){
37     init();
38     vector<int> p={0,1,2,3,4,5,6,7,8};
39     for(int i=0; i<factorial[9]; ++i){
40         vector<int> s=decode(i, 9);
41         if(s!=p) puts("XX");
42         next_permutation(p.begin(), p.end());
43     }
44     return 0;

```

### 8.4 Chinese\_remainder\_theorem.c

```

1 template<typename T>
2 T Euler(T n){
3     T ans=n;
4     for(T i=2; i*i<=n; ++i){
5         if(n%i==0){
6             ans=ans/i*(i-1);
7             while(n%i==0) n/=i;
8         }
9     }
10    if(n>1) ans=ans/n*(n-1);
11    return ans;
12 }
13 template<typename T>
14 T pow_mod(T n, T k, T m){
15     T ans=1;
16     for(n=(n>=m?n%m:n); k>=1){
17         if(k&1) ans=ans*n%m;
18         n=n*n%m;
19     }
20     return ans;
21 }
22 template<typename T>
23 T crt(vector<T> &m, vector<T> &a){
24     T M=1, tM, ans=0;
25     for(int i=0; i<(int)m.size(); ++i) M*=m[i];
26     for(int i=0; i<(int)a.size(); ++i){
27         tM=M/m[i];
28         ans=(ans+(a[i]*tM%M)*pow_mod(tM, Euler(m[i])
29             -1, m[i])%M)%M;
30         /*如果m[i]是質數 · Euler(m[i])-1=m[i]-2 ·
31             就不用算Euler了*/
32     }
33     return ans;
34 }

```

### 8.5 enumerate.cpp

```

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

```

### 8.6 eulerphi.cpp

```

1 int eulerPhi(int n){
2     int m = sqrt(n+.5);
3     int res=n;
4     for(int i=2; i<=m; ++i){
5         if(n%i==0){
6             res = res*(i-1)/i;
7             while(n%i==0) n/=i;
8         }
9     }
10    if(n>1) res = res*(n-1)/n;
11    return res;

```

```

12 }
13
14 long long int phi[N+1];
15 void phiTable(){
16     for(int i=1;i<=N;i++)phi[i]=i;
17     for(int i=1;i<=N;i++)for(x=i*2;x<=N;x+=i)
18         phi[x]-=phi[i];
19 }

```

## 8.7 Factor.cpp

```

1 LL LLMul(LL a, LL b, const LL &mod) {
2     LL ans=0;
3     while(b) {
4         if(b&1) {
5             ans+=a;
6             if(ans>=mod) ans-=mod;
7         }
8         a<<=1, b>>=1;
9         if(a>=mod) a-=mod;
10    }
11    return ans;
12 }
13 inline long long mod_mul(long long a,long
14     long b,long long m){
15     a%=m,b%=m;
16     long long y=(long long)((double)a*b/m+0.5)
17     ;/* fast for m < 2^58 */
18     long long r=(a*b-y*m)%m;
19     return r<0?r+m:r;
20 }
21 template<typename T>
22 inline T pow(T a,T b,T mod){//a^b%mod
23     T ans=1;
24     for(;b;a=mod_mul(a,a,mod),b>>=1)
25         if(b&1)ans=mod_mul(ans,a,mod);
26     return ans;
27 }
28 int sprp[3]={2,7,61};//int範圍可解
29 int llsprp
30     [7]={2,325,9375,28178,450775,9780504,179526
31     //至少unsigned Long Long範圍
32 }
33 template<typename T>
34 inline bool isprime(T n,int *sprp,int num){
35     if(n==2)return 1;
36     if(n<2||n%2==0)return 0;
37     int t=0;
38     T u=n-1;
39     for(;u%2==0;++t)u>>=1;
40     for(int i=0;i<num;++i){
41         T a=sprp[i]%n;
42         if(a==0||a==1||a==n-1)continue;
43         T x=pow(a,u,n);
44         if(x==1||x==n-1)continue;
45         for(int j=0;j<t;++j){
46             x=mod_mul(x,x,n);
47             if(x==1)return 0;
48             if(x==n-1)break;
49         }
50         if(x==n-1)continue;
51         return 0;
52     }
53 }

```

```

48     return 1;
49 }
50
51 LL func(const LL n,const LL mod,const int c) {
52     {
53         return (LLmul(n,n,mod)+c+mod)%mod;
54     }
55 }
56 LL pollrho(const LL n, const int c) { //循環
57     環節長度
58     LL a=1, b=1;
59     a=func(a,n,c)%n;
60     b=func(b,n,c)%n; b=func(b,n,c)%n;
61     while(gcd(abs(a-b),n)!=1) {
62         a=func(a,n,c)%n;
63         b=func(b,n,c)%n; b=func(b,n,c)%n;
64     }
65     return gcd(abs(a-b),n);
66 }
67 void prefactor(LL &n, vector<LL> &v) {
68     for(int i=0;i<12;++i) {
69         while(n%prime[i]==0) {
70             v.push_back(prime[i]);
71             n/=prime[i];
72         }
73     }
74 }
75 void smallfactor(LL n, vector<LL> &v) {
76     if(n<MAXPRIME) {
77         while(isp[(int)n]) {
78             v.push_back(isp[(int)n]);
79             n/=isp[(int)n];
80         }
81         v.push_back(n);
82     } else {
83         for(int i=0;i<primecnt&&prime[i]*
84             prime[i]<=n;++i) {
85             while(n%prime[i]==0) {
86                 v.push_back(prime[i]);
87                 n/=prime[i];
88             }
89             if(n!=1) v.push_back(n);
90         }
91     }
92 }
93 void comfactor(const LL &n, vector<LL> &v) {
94     if(n<1e9) {
95         smallfactor(n,v);
96         return;
97     }
98     if(Isprime(n)) {
99         v.push_back(n);
100        return;
101    }
102    LL d;
103    for(int c=3; c<=n; c++) {
104        d = pollrho(n,c);
105        if(d!=n) break;
106    }
107    comfactor(d,v);
108    comfactor(n/d,v);
109 }

```

```

110
111 void Factor(const LL &x, vector<LL> &v) {
112     LL n = x;
113     if(n==1) { puts("Factor 1"); return; }
114     prefactor(n,v);
115     if(n==1) return;
116     comfactor(n,v);
117     sort(v.begin(),v.end());
118 }
119
120 void AllFactor(const LL &n,vector<LL> &v) {
121     vector<LL> tmp;
122     Factor(n,tmp);
123     v.clear();
124     v.push_back(1);
125     int len;
126     LL now=1;
127     for(int i=0;i<tmp.size();++i) {
128         if(i==0 || tmp[i]!=tmp[i-1]) {
129             len = v.size();
130             now = 1;
131         }
132         now*=tmp[i];
133         for(int j=0;j<len;++j)
134             v.push_back(v[j]*now);
135     }
136 }

```

## 8.8 FFT.cpp

```

1 template<typename T,typename VT=std::vector<
2     std::complex<T>> > >
3 struct FFT{
4     const T pi;
5     FFT(const T pi=acos((-1.0)):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&0xF0F0F0F0)<<4)|((a&0xFF0F0F0F)>>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<mh;++i){
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                }
30            }
31        }
32    }

```

```

23         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 };

```

## 8.9 find\_real\_root.cpp

```

1 // an*x^n + ... + a1x + a0 = 0;
2 int sign(double x){
3     return x < -eps ? -1 : x > eps;
4 }
5
6 double get(const vector<double>&coef, double
7     x){
8     double e = 1, s = 0;
9     for(auto i : coef) s += i*e, e *= x;
10    return s;
11 }
12 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]/
35             coef[1]);
36         return res;
37     }
38     vector<double>dcoef(n);
39     for(int i = 0; i < n; ++i) dcoef[i] =
40         coef[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 }

```

```

44 int main () {
45     vector<double>ve;
46     vector<double>ans = cal(ve, n);
47     // 視情況把答案 +eps · 避免 -0
48 }

```

## 8.10 formula.tex

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

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

$$g(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d) * g(n/d)$$

$$\text{Catalan number} : (2n)!/n!/(n+1)$$

$$\text{Harmonic series } H_n = \ln(n) + \gamma + 1/(2n) - 1/(12n^2) + 1/(120n^4) - \dots$$

$$\gamma = 0.577215664901532860606512090082402431042187590962$$

$$i - \text{th gray code} : i^{(i>1)}$$

$$SG(A+B) = SG(A) \oplus SG(B)$$

## 8.11 Gauss\_Elimination.cpp

```

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

```

## 8.12 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         m[i] /= d;

```

```

7         b[i] = LLMul(b[i]/d, x, m[i]);
8     }
9     LL lastb = b[0], lastm = m[0];
10    for(int i=1; i<n; i++) {
11        LL x, y, d = extgcd(m[i], lastm, x, y);
12        if((lastb-b[i])%d!=0) return
13            make_pair(-1LL, 0LL);
14        lastb = LLMul((lastb-b[i])/d, x, (
15            lastm/d))*m[i];
16        lastm = (lastm/d)*m[i];
17        lastb = (lastb+b[i])%lastm;
18    }
19    return make_pair(lastb<0?lastb+lastm:
20        lastb, lastm);

```

## 8.13 Lucas.cpp

```

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

```

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

```

```

70    for(int i=0; i<r; i++){
71        det = det*m[i][i];
72        det = det/lazy[i];
73        for(auto &j:m[i]) j/=lazy[i];
74    }
75    return det;
76 }
77 };

```

## 8.15 NTT.cpp

```

1 26150536055667*(2^18)+1,3
2 15*(2^27)+1,31
3 479*(2^21)+1,3
4 7*17*(2^23)+1,3
5 3*3*211*(2^19)+1,5
6 25*(2^22)+1,3
7 template<typename T, typename VT=std::vector<
8     T> >
9 struct NTT {
10     const T P,G;
11     NTT(T p=(1<<23)*7*17+1, T g=3):P(p),G(g){}
12     unsigned int bit_reverse(unsigned int a,
13         int len){
14         a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)
15             >>1);
16         a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)
17             >>2);
18         a=((a&0x0F0F0F0FU)<<4)|((a&0xFF0F0F0FU)
19             >>4);
20         a=((a&0xFF0F0FFU)<<8)|((a&0xFF0F0F0U)
21             >>8);
22         a=((a&0x000FFFFU)<<16)|((a&0xFFFF000U)
23             >>16);
24         return a>>(32-len);
25     }
26     T pow_mod(T n, T k, T m) {
27         T ans=1;
28         for(n=(n>=m?n%m:n); k>=1){
29             if(k&1) ans=ans*n%m;
30             n=n*n%m;
31         }
32         return ans;
33     }
34     void ntt(bool is_inv, VT &in, VT &out, int N) {
35         int bitlen=std::lg(N);
36         for(int i=0; i<N; i++) out[bit_reverse(i,
37             bitlen)]=in[i];
38         for(int step=2, id=1; step<=N; step<<=1, ++
39             id){
40             T wn=pow_mod(G, (P-1)>>id, P), wi=1, u, t;
41             const int mh=step>>1;
42             for(int i=0; i<mh; i++){
43                 for(int j=i; j<N; j+=step){
44                     u=out[j]; t=wi*out[j+mh]%P;
45                     out[j]=u+t;
46                     out[j+mh]=u-t;
47                     if(out[j]>=P) out[j]-=P;
48                     if(out[j+mh]<0) out[j+mh]+=P;
49                 }
50                 wi=wi*wn%P;
51             }
52         }
53     }

```

## 8.16 random.cpp

```

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

```

## 8.17 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
21     ,long long mod){ //a^b%mod
22     long long ans=1;
23     for(;b;a=a*a%mod,b>>=1)
24         if(b&1) ans=ans*a%mod;
25     return ans;
26 }
27 bool isless(long long *a,int n,int k){
28     if(*a==1) return k>1;
29     if(--n==0) return *a<k;
30     int next=0;
31     for(long long b=1;b<k;next++)
32         b*=a;
33     return isless(a+1,n,next);
34 }
35 long long high_pow(long long *a,int n,long
36     long mod){

```

```

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)
38         ;++k)
39         tma=tma*(*a)%mod;
40     if(isless(a+1,n,k)) return pow(*a,high_pow(
41         a+1,n,k),mod);
42     int tmd=high_pow(a+1,n,r);
43     int t=(tmd-k+r)%r;
44     return pow(*a,k+t,mod);
45 }
46 long long a[1000005];
47 int t,mod;
48 int main(){
49     init_euler();
50     scanf("%d",&t);
51     #define n 4
52     while(t--){
53         for(int i=0;i<n;i++) scanf("%lld",&a[i]);
54         scanf("%d",&mod);
55         printf("%lld\n",high_pow(a,n,mod));
56     }

```

## 8.18 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     }
7     mod_t pow(T b)const{
8         mod_t ans(1);
9         for(mod_t now=*this;b;now=now*b,b/=2)
10             if(b%2) ans=ans*now;
11         return ans;
12 }
13 mod_t operator-(int) const{
14     return mod_t(mod-data);
15 }
16 mod_t operator+(const mod_t &b) const{
17     return mod_t((data+b.data)%mod);
18 }
19 mod_t operator-(const mod_t &b) const{
20     return mod_t((data-b.data+mod)%mod);
21 }
22 mod_t operator*(const mod_t &b) const{
23     return mod_t((data*b.data)%mod);
24 }
25 mod_t operator/(const mod_t &b) const{
26     return *this*b.pow(mod-2); // *this *
27     Inverse(b)
28 }
29 operator T() const{ return data; }
30 friend istream &operator>>(istream &i,
31     mod_t &b){
32     T d;
33     i>>d;
34     b=mod_t(d);
35     return i;
36 }

```

## 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         }
53     }
54     /*DP出每個前綴在字串s出現的次數並傳回所
55     有字串被s匹配成功的次數O(N*M)*/
56     int match_0(const char *s){
57         int ans=0,id,p=0,i;
58         for(i=0;s[i];++i){

```

```

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

```



```

110     if(S[p].next[i]==0)S[p].next[i]=S[
111         S[p].fail].next[i];
112     }
113 };

```

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

```

## 9.3 KMP.cpp

```

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

```

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

```

## 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
   置
14 }

```

## 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
   >=len||tmp[sa[i-1]+k]!=tmp[sa[i]+k
   ])+id;
26        rank[sa[i]]=id;
27    }
28    A=id+1;
29 }
30 }
31 #undef radix_sort
32 //h:高度數組 sa:後綴數組 rank:排名
33 inline void suffix_array_lcp(const char *s,
   int len,int *h,int *sa,int *rank){
34     for(int i=0;i<len;++i)rank[sa[i]]=i;
35     for(int i=0,k=0;i<len;++i){
36         if(rank[i]==0)continue;
37         if(k)--k;
38         while(s[i+k]==s[sa[rank[i]-1]+k])++k;
39         h[rank[i]]=k;
40     }
41     h[0]=0;
42 }

```

## 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 tnfnshb017\_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*MAXN)
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[MXN2];
15 bool vis[MXN2]={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
6         (0){}
7 };
8 vector<edge> E;
9 vector<int> G[N];// 1-base
10 int low[N],vis[N],Time;
11 int bcc_id[N],bridge_cnt,bcc_cnt;// 1-base
12 int st[N],top;//BCC用
13 inline void add_edge(int u,int v){
14     G[u].push_back(E.size());
15     E.push_back(edge(u,v));
16     G[v].push_back(E.size());
17     E.push_back(edge(v,u));
18 }
19 void dfs(int u,int re=-1){//u當前點，re為u連
20     接前一個點的邊
21     int v;
22     low[u]=vis[u]=++Time;
23     st[top++]=u;
24     for(size_t i=0;i<G[u].size();++i){
25         int e=G[u][i];v=E[e].v;
26         if(!vis[v]){
27             dfs(v,e^1);//e^1反向邊
28             low[u]=min(low[u],low[v]);
29             if(vis[u]<low[v]){
30                 E[e].is_bridge=E[e^1].is_bridge=1;
31                 ++bridge_cnt;
32             }
33         }else if(vis[v]<vis[u]&&v!=re){
34             low[u]=min(low[u],vis[v]);
35         }
36         if(vis[u]==low[u]){//處理BCC
37             ++bcc_cnt;// 1-base
38             do bcc_id[v=st[--top]]=bcc_cnt;//每個點
39                 所在的BCC
40             while(v!=u);
41         }
42     }
43     inline void bcc_init(int n){
44         Time=bcc_cnt=bridge_cnt=top=0;
45         E.clear();
46         for(int i=1;i<=n;++i){
47             G[i].clear();
48             vis[i]=bcc_id[i]=0;
49         }
50     }

```

### 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
31        的根要特判
32    }
33    inline void bcc_init(int n){
34        Time=bcc_cnt=top=0;
35        for(int i=1;i<=n;++i){
36            G[i].clear();
37            is_cut[i]=vis[i]=bcc_id[i]=0;
38        }
39    }

```

## 11 Tree\_problem

### 11.1 HeavyLight.cpp

```

1 #include<vector>
2 #define MXN 100005
3 typedef std::vector<int>::iterator VIT;
4 int siz[MXN],max_son[MXN],pa[MXN],dep[
5     MXN];
6 int link_top[MXN],link[MXN],cnt;
7 std::vector<int> >G[MXN];
8 void find_max_son(int x){
9     siz[x]=1;
10    max_son[x]=-1;
11    for(VIT i=G[x].begin();i!=G[x].end();++i){
12        if(*i==pa[x])continue;
13        pa[*i]=x;
14        dep[*i]=dep[x]+1;
15        find_max_son(*i);
16        if(max_son[x]==-1||siz[*i]>siz[max_son[x]
17            ])max_son[x]=*i;
18        siz[x]+=siz[*i];
19    }
20 }
21 void build_link(int x,int top){
22     link[x]=++cnt;
23     link_top[x]=top;
24     if(max_son[x]==-1)return;
25     build_link(max_son[x],top);
26     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
42     次區間處理
43     return dep[a]<dep[b]?a:b;
44 }

```

### 11.2 LCA.cpp

```

1 #define MXN 100000
2 #define MAX_LOG 17
3 int pa[MAX_LOG+1][MXN+5];
4 int dep[MXN+5];
5 vector<int>G[MXN+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[
9         i][pa[i][x]];
10    for(auto &i:G[x]){
11        if(i==p)continue;
12        dep[i]=dep[x]+1;
13        dfs(i,x);
14    }
15 }
16 inline int jump(int x,int d){
17     for(int i=0;i<d;++i)if((x>>i)&1)x=pa[i][x];
18     return x;
19 }
20 inline int find_lca(int a,int b){
21     if(dep[a]>dep[b])swap(a,b);
22     b=jump(b,dep[b]-dep[a]);
23     if(a==b)return a;
24     for(int i=MAX_LOG;i>=0;--i){
25         if(pa[i][a]!=pa[i][b]){
26             a=pa[i][a];
27             b=pa[i][b];
28         }
29     }
30     return pa[0][a];
31 }

```

### 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[0]=ch[1]=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
11     tree的根
12     return node[node[x].pa].ch[0]!=x&&node[
13         node[x].pa].ch[1]!=x;
14 }
15 void down(int x){//懶惰標記下推
16     if(node[x].rev){
17         if(node[x].ch[0])node[node[x].ch[0]].rev
18             ^=1;
19         if(node[x].ch[1])node[node[x].ch[1]].rev
20             ^=1;
21         std::swap(node[x].ch[0],node[x].ch[1]);
22         node[x].rev^=1;
23     }
24 }
25 void push_down(int x){//將所有祖先的懶惰標記
26     下推
27     if(!isroot(x))push_down(node[x].pa);
28     down(x);
29 }
30 void up(int x){//將子節點的資訊向上更新
31 void rotate(int x){//旋轉，會自行判斷轉的方
32     向
33     int y=node[x].pa,z=node[y].pa,d=(node[y].
34         ch[1]==x);
35     node[x].pa=z;
36     if(!isroot(y))node[z].ch[node[z].ch[1]==y
37         ]=x;
38     node[y].ch[d]=node[x].ch[d^1];
39     node[node[y].ch[d]].pa=y;
40     node[y].pa=x,node[x].ch[d^1]=y;
41     up(y);
42     up(x);
43 }
44 void splay(int x){//將節點x伸展到所在splay
45     tree的根
46     push_down(x);
47     while(!isroot(x)){
48         int y=node[x].pa;
49         if(!isroot(y)){
50             int z=node[y].pa;
51             if((node[z].ch[0]==y)^(node[y].ch[0]==
52                 x))rotate(y);
53             else rotate(x);
54         }
55         rotate(x);
56     }
57 }
58 int access(int x){
59     int last=0;
60     while(x){
61         splay(x);
62         node[x].ch[1]=last;
63         up(x);
64         last=x;
65     }
66 }

```

```

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
60     的access
61     int last=0;
62     while(x){
63         splay(x);
64         if(is&&!node[x].pa){
65             //printf("%d\n",max(node[last].ma,node
66                 [node[x].ch[1]].ma));
67         }
68         node[x].ch[1]=last;
69         up(x);
70         last=x;
71         x=node[x].pa;
72     }
73 void query_edge(int u,int v){
74     access(u);
75     access(v,1);
76 }
77 void make_root(int x){
78     access(x),splay(x);
79     node[x].rev^=1;
80 }
81 void make_root(int x){
82     node[access(x)].rev^=1;
83     splay(x);
84 }
85 void cut(int x,int y){
86     make_root(x);
87     access(y);
88     splay(y);
89     node[y].ch[0]=0;
90     node[x].pa=0;
91 }
92 void cut_parents(int x){
93     access(x);
94     splay(x);
95     node[node[x].ch[0]].pa=0;
96     node[x].ch[0]=0;
97 }
98 void link(int x,int y){
99     make_root(x);
100    node[x].pa=y;
101 }
102 int find_root(int x){
103    x=access(x);
104    while((node[x].ch[0])x=node[x].ch[0];
105    splay(x);
106    return x;
107 }
108 int query(int u,int v){
109     //傳回uv路徑splay tree的根結點
110     //這種寫法無法求LCA
111     make_root(u);
112     return access(v);
113 }
114 int query_lca(int u,int v){
115     //假設求鏈上點權的總和，sum是子樹的權重和，
116     data是節點的權重

```

```

115     access(u);
116     int lca=access(v);
117     splay(u);
118     if(u==lca){
119         //return node[lca].data+node[node[lca].
120             ch[1]].sum
121     }else{
122         //return node[lca].data+node[node[lca].
123             ch[1]].sum+node[u].sum
124     }
125 }
126 struct EDGE{
127     int a,b,w;
128     e[10005];
129     int n;
130     vector<pair<int,int>> >G[10005];
131     //first表示子節點，second表示邊的編號
132     int pa[10005],edge_node[10005];
133     //pa是父母節點，暫存用的，edge_node是每個編
134     被存在哪個點裡面的陣列
135 void bfs(int root){
136     //在建構的時候把每個點都設成一個splay tree，
137     不會壞掉
138     queue<int> q;
139     for(int i=1;i<=n;++i)pa[i]=0;
140     q.push(root);
141     while(q.size()){
142         int u=q.front();
143         q.pop();
144         for(int i=0;i<(int)G[u].size();++i){
145             int v=G[u][i].first;
146             if(v!=pa[u]){
147                 pa[v]=u;
148                 node[v].pa=u;
149                 node[v].data=e[G[u][i].second].w;
150                 edge_node[G[u][i].second]=v;
151                 up(v);
152                 q.push(v);
153             }
154         }
155     }
156 void change(int x,int b){
157     splay(x);
158     //node[x].data=b;
159     up(x);
160 }

```

## 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 }
22 vector<int> dis;//這東西如果放在函數裡會TLE
23 int cal(int u,int d){
24     dis.clear();
25     get_dis(dis,u,-1,d);
26     sort(dis.begin(),dis.end());
27     int l=0,r=dis.size()-1,res=0;
28     while(l<r){
29         while(l<r&&dis[l]+dis[r]>k)--r;
30         res+=r-(l+1);
31     }
32     return res;
33 }
34 pair<int,int> tree_centroid(int u,int pa,
35     const int sz){
36     size[u]=1;//找樹重心，second是重心
37     pair<int,int> res(INT_MAX,-1);
38     int ma=0;
39     for(size_t i=0;i<g[u].size();++i){
40         int v=g[u][i].first;
41         if(v==pa||vis[v])continue;
42         res=min(res,tree_centroid(v,u,sz));
43         size[u]+=size[v];
44         ma=max(ma,size[v]);
45     }
46     ma=max(ma,sz-size[u]);
47     return min(res,make_pair(ma,u));
48 }
49 int tree_DC(int u,int sz){
50     int center=tree_centroid(u,-1,sz).second;
51     int ans=cal(center,0);
52     vis[center]=1;
53     for(size_t i=0;i<g[center].size();++i){
54         int v=g[center][i].first,w=g[center][i].
55             second;
56         if(vis[v])continue;
57         ans-=cal(v,w);
58         ans+=tree_DC(v,size[v]);
59     }
60     return ans;
61 }
62 int main(){
63     while(scanf("%d",&n,&k),n||k){
64         init();
65         for(int i=1;i<=n;++i){
66             int u,v,w;
67             scanf("%d%d%d",&u,&v,&w);
68             g[u].push_back(make_pair(v,w));
69             g[v].push_back(make_pair(u,w));
70         }
71         printf("%d\n",tree_DC(1,n));
72     }
73     return 0;
74 }

```

# ACM ICPC Team

## Reference - NTHU

### Jinkela

#### Contents

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