

# 1 Computational\_Geometry

## 1.1 Geometry.cpp

```

1 const double PI=atan2(0.0,-1.0);
2 template<typename T>
3 struct point{
4     T x,y;
5     point(){
6     point(const T&x,const T&y):x(x),y(y){
7     point operator+(const point &b)const{
8         return point(x+b.x,y+b.y);
9     point operator-(const point &b)const{
10        return point(x-b.x,y-b.y);
11    point operator*(const T &b)const{
12        return point(x*b,y*b);
13    point operator/(const T &b)const{
14        return point(x/b,y/b);
15    bool operator==(const point &b)const{
16        return x==b.x&&y==b.y;
17    T dot(const point &b)const{
18        return x*b.x+y*b.y;
19    T cross(const point &b)const{
20        return x*b.y-y*b.x;
21    point normal()const{//求法向量
22        return point(-y,x);
23    T abs2()const{//向量長度的平方
24        return dot(*this);
25    }
26    T rad(const point &b)const{//兩向量的弧度
27        return fabs(atan2(fabs(cross(b)),dot(b)));
28    }
29    T getA()const{//對x軸的弧度
30        T A=atan2(y,x);{//超過180度會變負的
31        if(A<=-PI/2)A+=PI*2;
32        return A;
33    }
34 };
35 template<typename T>
36 struct line{
37     line(){
38     point<T> p1,p2;
39     T a,b,c;//ax+by+c=0
40     line(const point<T>&x,const point<T>&y):p1(
41         x),p2(y){
42     void pton()const{//轉成一般式
43         a=p1.y-p2.y;
44         b=p2.x-p1.x;
45         c=-a*p1.x-b*p1.y;
46     }
47     T cross(const point<T> &p)const{//點和有向
48         直線的關係，>0左邊、=0在線上、<0右邊
49         return (p2-p1).cross(p-p1);
50     }
51     bool point_on_segment(const point<T>&p)
52         const{//點是否線段上
53         return cross(p)==0&&(p1-p).dot(p2-p)<=0;
54     }
55     T dis2(const point<T> &p,bool is_segment
56         =0)const{//點跟直線/線段的距離平方
57     point<T> v=p2-p1,v1=p-p1;
58     if(is_segment){
59         point<T> v2=p-p2;
60         if(v.dot(v1)<=0)return v1.abs2();
61         if(v.dot(v2)>=0)return v2.abs2();
62     }
63     T tmp=v.cross(v1);
64     return tmp*tmp/v.abs2();
65 }
66 T seg_dis2(const line<T> &l)const{//兩線段
67     距離平方
68     return min({dis2(l.p1,1),dis2(l.p2,1),l.
69         dis2(p1,1),l.dis2(p2,1)});
70 }
71 point<T> projection(const point<T> &p)
72     const{//點對直線的投影
73     point<T> n=(p2-p1).normal();
74     return p-n*(p-p1).dot(n)/n.abs2();
75 }
76 point<T> mirror(const point<T> &p)const{//
77     點對直線的鏡射
78     //要先呼叫pton轉成一般式
79     point<T> ans;
80     T d=a*a+b*b;
81     ans.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/
82         d;
83     ans.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/
84         d;
85     return ans;
86 }
87 bool equal(const line &l)const{//直線相等
88     return cross(l.p1)==0&&cross(l.p2)==0;
89 }
90 bool parallel(const line &l)const{//直線平
91     行
92     return (p1-p2).cross(l.p1-l.p2)==0;
93 }
94 bool cross_seg(const line &l)const{//直線
95     是否交線段
96     return (p2-p1).cross(l.p1-p1)*(p2-p1).
97         cross(l.p2-p1)<=0;
98 }
99 char line_intersect(const line &l)const{//
100     直線相交情況，-1無限多點、1交於一點、0
101     不相交
102     return parallel(l)?(cross(l.p1)==0?-1:0)
103         :1;
104 }
105 char seg_intersect(const line &l)const{//
106     線段相交情況，-1無限多點、1交於一點、0
107     不相交
108     T c1=(p2-p1).cross(l.p1-p1);
109     T c2=(p2-p1).cross(l.p2-p1);
110     T c3=(l.p2-l.p1).cross(p1-l.p1);
111     T c4=(l.p2-l.p1).cross(p2-l.p1);
112     if(c1==0&&c2==0){
113         if(p1==l.p1&&(p2-p1).dot(l.p2)<=0)
114             return 1;
115         if(p1==l.p2&&(p2-p1).dot(l.p1)<=0)
116             return 1;
117         if(p2==l.p1&&(p1-p2).dot(l.p2)<=0)
118             return 1;
119         if(p2==l.p2&&(p1-p2).dot(l.p1)<=0)
120             return 1;
121     }
122     return 0;
123 }
124 point<T> line_intersection(const line &l)
125     const{//直線交點
126     T c1=(p2-p1).cross(l.p1-p1);
127     T c2=(p2-p1).cross(l.p2-p1);
128     T c3=(l.p2-l.p1).cross(p1-l.p1);
129     T c4=(l.p2-l.p1).cross(p2-l.p1);
130     if(c1==0&&c2==0){
131         if(p1==l.p1&&(p2-p1).dot(l.p2)<=0)
132             return p1;
133         if(p1==l.p2&&(p2-p1).dot(l.p1)<=0)
134             return p1;
135         if(p2==l.p1&&(p1-p2).dot(l.p2)<=0)
136             return p2;
137         if(p2==l.p2&&(p1-p2).dot(l.p1)<=0)
138             return p2;
139     }
140     else if(c1*c2<=0&&c3*c4<=0)return
141         line_intersection(l);
142     //return INF;
143 }
144 };
145 template<typename T>
146 struct polygon{
147     polygon(){
148     vector<point<T>> p;//逆時針順序
149     T area()const{//面積
150     T ans=0;
151     for(int i=p.size()-1,j=0;j<(int)p.size()
152         ;i=j++){
153         ans+=p[i].cross(p[j]);
154     }
155     return ans/2;
156 }
157 point<T> center_of_mass()const{//重心
158     T cx=0,cy=0,w=0;
159     for(int i=p.size()-1,j=0;j<(int)p.size()
160         ;i=j++){
161         T a=p[i].cross(p[j]);
162         cx+=(p[i].x+p[j].x)*a;
163         cy+=(p[i].y+p[j].y)*a;
164         w+=a;
165     }
166     return point<T>(cx/3/w,cy/3/w);
167 }
168 }
169 char ahas(const point<T>&t)const{//點是否
170     在簡單多邊形內，是的話回傳1、在邊上回
171     傳-1、否則回傳0
172     bool c=0;
173     for(int i=0,j=p.size()-1;i<p.size();i=
174         ++j){
175         if(line<T>(p[i],p[j]).point_on_segment
176             (t))return -1;
177         else if((p[i].y>t.y)!=p[j].y>t.y)&&
178             t.x<(p[j].x+p[i].x)*(t.y-p[i].y)/(p[j]
179                 .y-p[i].y)+p[i].x)
180             c=!c;
181     }
182     return c;
183 }
184 char point_in_convex(const point<T>&x)
185     const{
186     int l=1,r=(int)p.size()-2;
187     while(l<=r){//點是否在凸多邊形內，是的話
188         回傳1、在邊上回傳-1、否則回傳0
189         int mid=(l+r)/2;
190         T a1=(p[mid]-p[0]).cross(x-p[0]);
191         T a2=(p[mid+1]-p[0]).cross(x-p[0]);
192         if(a1>=0&&a2<=0){
193             T res=(p[mid+1]-p[mid]).cross(x-p[
194                 mid]);
195             return res>0?1:(res>=0?-1:0);
196         }
197         else if(a1<0)r=mid-1;
198         else l=mid+1;
199     }
200     return 0;
201 }
202 vector<T> getA()const{//凸包邊對x軸的夾角
203     vector<T> res;//一定是遞增的
204     for(size_t i=0;i<p.size();i++){
205         res.push_back((p[(i+1)%p.size()]-p[i])
206             .getA());
207     }
208     return res;
209 }
210 bool line_intersect(const vector<T>&A,
211     const line<T> &l)const{//O(LogN)
212     int f1=upper_bound(A.begin(),A.end(),(l.
213         p1-l.p2).getA())-A.begin();
214     int f2=upper_bound(A.begin(),A.end(),(l.
215         p2-l.p1).getA())-A.begin();
216     return l.cross_seg(line<T>(p[f1],p[f2]));
217 }
218 polygon cut(const line<T> &l)const{//凸包
219     對直線切割，得到直線L左側的凸包
220     polygon ans;
221     for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
222         if(l.cross(p[i])>=0){
223             ans.p.push_back(p[i]);
224             if(l.cross(p[j])<0)
225                 ans.p.push_back(l.
226                     line_intersection(line<T>(p[i]
227                         ],p[j])));
228         }
229         else if(l.cross(p[j])>=0)
230             ans.p.push_back(l.line_intersection(
231                 line<T>(p[i],p[j]));
232     }
233     return ans;
234 }
235 static bool graham_cmp(const point<T>&a,
236     const point<T>&b){
237     return (a.x<b.x)||((a.x==b.x&&a.y<b.y));
238 }
239 void graham(vector<point<T>> &s){{//凸包
240     sort(s.begin(),s.end(),graham_cmp);
241     p.resize(s.size()+1);
242     int m=0;
243     for(int i=0;i<(int)s.size();i++){
244         while(m>2&&(p[m-1]-p[m-2]).cross(s[i]
245             -p[m-2])<=0)-m;
246         p[m++]=s[i];
247     }
248 }

```

```

198 }
199 for(int i=s.size()-2,t=m+1;i>=0;--i){
200     while(m>=t&&(p[m-1]-p[m-2]).cross(s[i]
201         ]-p[m-2])<=0)--m;
202     p[m++]=s[i];
203 }
204 if(s.size()>1)--m;
205 p.resize(m);
206 }
207 T diam(){//直徑
208     int n=p.size(),t=1;
209     T ans=0;p.push_back(p[0]);
210     for(int i=0;i<n;i++){
211         point<T> now=p[i+1]-p[i];
212         while(now.cross(p[t+1]-p[i])>now.cross
213             (p[t]-p[i]))t=(t+1)%n;
214         ans=max(ans,max((p[i]-p[t]).abs2(),(p[
215             i+1]-p[t+1]).abs2()));
216     }
217     return p.pop_back(),ans;
218 }
219 T min_cover_rectangle(){//最小覆蓋矩形
220     int n=p.size(),t=1,r=1,l;
221     if(n<3)return 0;//也可以做最小周長矩形
222     T ans=1e99;p.push_back(p[0]);
223     for(int i=0;i<n;i++){
224         point<T> now=p[i+1]-p[i];
225         while(now.cross(p[t+1]-p[i])>now.cross
226             (p[t]-p[i]))t=(t+1)%n;
227         while(now.dot(p[r+1]-p[i])>now.dot(p[r
228             ]-p[i]))r=(r+1)%n;
229         if(l==r;
230         while(now.dot(p[l+1]-p[i])<=now.dot(p[
231             l]-p[i]))l=(l+1)%n;
232         T d=now.abs2();
233         T tmp=now.cross(p[t]-p[i])*(now.dot(p[
234             r]-p[i])-now.dot(p[l]-p[i]))/d;
235         ans=min(ans,tmp);
236     }
237     return p.pop_back(),ans;
238 }
239 T max_triangle(){//最大內接三角形
240     int n=p.size(),a=1,b=2;
241     if(n<3)return 0;
242     T ans=0,tmp;p.push_back(p[0]);
243     for(int i=0;i<n;i++){
244         while((p[a]-p[i]).cross(p[b+1]-p[i])>(
245             tmp=(p[a]-p[i]).cross(p[b]-p[i]))
246             b=(b+1)%n;
247         ans=max(ans,tmp);
248         while((p[a+1]-p[i]).cross(p[b]-p[i])>(
249             tmp=(p[a]-p[i]).cross(p[b]-p[i]))
250             a=(a+1)%n;
251         ans=max(ans,tmp);
252     }
253     return p.pop_back(),ans/2;
254 }
255 T dis2(polygon &p1){//凸包最近距離平方
256     vector<point<T> > &P=p,Q=p1.p;
257     int n=P.size(),m=Q.size(),l=0,r=0;
258     for(int i=0;i<n;i++){
259         if(P[i].y<P[l].y)l=i
260     }
261     for(int i=0;i<m;i++){
262         if(Q[i].y<Q[r].y)r=i
263     }
264     P.push_back(P[0]),Q.push_back(Q[0]);
265
266     T ans=1e99;
267     for(int i=0;i<n;i++){
268         while((P[i]-P[l+1]).cross(Q[r+1]-Q[r])
269             <0)r=(r+1)%m;
270         ans=min(ans,line<T>(P[i],P[l+1]).
271             seg_dis2(line<T>(Q[r],Q[r+1])));
272         l=(l+1)%n;
273     }
274     return P.pop_back(),Q.pop_back(),ans;
275 }
276 static char sign(const point<T>&t){
277     return (t.y==0?t.x:t.y)<0;
278 }
279 static bool angle_cmp(const line<T>& A,
280     const line<T>& B){
281     point<T> a=A.p2-A.p1,b=B.p2-B.p1;
282     return sign(a)<sign(b)||((sign(a)==sign(b)
283         )&&a.cross(b)>0);
284 }
285 int halfplane_intersection(vector<line<T>
286     > &s){//半平面交
287     sort(s.begin(),s.end(),angle_cmp);
288     //左側為該線段半平面
289     int L,R,n=s.size();
290     vector<point<T> > px(n);
291     vector<line<T> > q(n);
292     q[L=R=0]=s[0];
293     for(int i=1;i<n;i++){
294         while(L<R&&s[i].cross(px[R-1])<=0)--R;
295         while(L<R&&s[i].cross(px[L])<=0)+L;
296         q[++R]=s[i];
297         if(q[R].parallel(q[R-1])){
298             --R;
299             if(q[R].cross(s[i].p1)>0)q[R]=s[i];
300         }
301         if(L<R)px[R-1]=q[R-1].
302             line_intersection(q[R]);
303     }
304     while(L<R&&q[L].cross(px[R-1])<=0)--R;
305     p.clear();
306     if(R-L==1)return 0;
307     px[R]=q[R].line_intersection(q[L]);
308     for(int i=L;i<R;i++)p.push_back(px[i]);
309     return R-L+1;
310 }
311 template<typename T>
312 struct triangle{
313     point<T> a,b,c;
314     triangle(){
315         triangle(const point<T> &a,const point<T>
316             &b,const point<T> &c):a(a),b(b),c(c){
317             T area(){const
318                 T t=(b-a).cross(c-a)/2;
319                 return t>0?t:-t;
320             }
321     }
322     point<T> barycenter(){const{//重心
323         return (a+b+c)/3;
324     }
325     point<T> circumcenter(){const{//外心
326         static line<T> u,v;
327         u.p1=(a+b)/2;
328         u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
329             b.x);
330         v.p1=(a+c)/2;
331         v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
332             c.x);
333         return u.line_intersection(v);
334     }
335     point<T> incenter(){const{//內心
336         T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
337             ()),C=sqrt((a-b).abs2());
338         return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
339             B*b.y+C*c.y)/(A+B+C);
340     }
341     point<T> perpercenter(){const{//垂心
342         return barycenter()*3-circumcenter()*2;
343     }
344 };
345 template<typename T>
346 struct point3D{
347     T x,y,z;
348     point3D(){
349         point3D(const T&x,const T&y,const T&z):x(x
350             ),y(y),z(z){
351         point3D operator+(const point3D &b)const{
352             return point3D(x+b.x,y+b.y,z+b.z);
353         }
354         point3D operator-(const point3D &b)const{
355             return point3D(x-b.x,y-b.y,z-b.z);
356         }
357         point3D operator*(const T &b)const{
358             return point3D(x*b,y*b,z*b);
359         }
360         point3D operator/(const T &b)const{
361             return point3D(x/b,y/b,z/b);
362         }
363         bool operator==(const point3D &b)const{
364             return x==b.x&&y==b.y&&z==b.z;
365         }
366         T dot(const point3D &b)const{
367             return x*b.x+y*b.y+z*b.z;
368         }
369         point3D cross(const point3D &b)const{
370             return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
371                 *b.y-y*b.x);
372         }
373         T abs2(){const{//向量長度的平方
374             return dot(*this);
375         }
376         T area2(const point3D &b)const{//和b、原點
377             圍成面積的平方
378             return cross(b).abs2()/4;
379         }
380 };
381 template<typename T>
382 struct line3D{
383     point3D<T> p1,p2;
384     line3D(){
385         line3D(const point3D<T> &p1,const point3D<
386             T> &p2):p1(p1),p2(p2){
387         T dis2(const point3D<T> &p,bool is_segment
388             =0)const{//點跟直線/線段的距離平方
389             point3D<T> v=p2-p1,v1=p-p1;
390             if(is_segment){
391                 point3D<T> v2=p-p2;
392                 if(v.dot(v1)<=0)return v1.abs2();
393                 if(v.dot(v2)>=0)return v2.abs2();
394             }
395             point3D<T> tmp=v.cross(v1);
396             return tmp.abs2()/v.abs2();
397         }
398     }
399     pair<point3D<T>,point3D<T> > closest_pair(
400         const line3D<T> &l)const{
401         point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
402         point3D<T> N=v1.cross(v2),ab(p1-l.p1);
403         //if(N.abs2()==0)return NULL;平行或重合
404
405         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
406         最近點對距離
407         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
408             cross(d2);
409         T t1=((l.p1-p1).cross(d2)).dot(D)/D.abs2
410             ();
411         T t2=((l.p1-p1).cross(d1)).dot(D)/D.abs2
412             ();
413         return make_pair(p1+d1*t1,l.p1+d2*t2);
414     }
415     bool same_side(const point3D<T> &a,const
416         point3D<T> &b)const{
417         return (p2-p1).cross(a-p1).dot((p2-p1).
418             cross(b-p1))>0;
419     }
420 };
421 template<typename T>
422 struct plane{
423     point3D<T> p0,n;//平面上的點和法向量
424     plane(){
425         plane(const point3D<T> &p0,const point3D<T>
426             &n):p0(p0),n(n){
427         T dis2(const point3D<T> &p)const{//點到平
428             面距離的平方
429             T tmp=(p-p0).dot(n);
430             return tmp*tmp/n.abs2();
431         }
432         point3D<T> projection(const point3D<T> &p)
433             const{
434             return p-n*(p-p0).dot(n)/n.abs2();
435         }
436         point3D<T> line_intersection(const line3D<
437             T> &l)const{
438             T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
439             重合該平面
440             return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
441                 tmp);
442         }
443         line3D<T> plane_intersection(const plane &
444             pl)const{
445             point3D<T> e=n.cross(pl.n),v=n.cross(e);
446             T tmp=pl.n.dot(v);//等於0表示平行或重合
447             該平面
448             point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/
449                 tmp);
450             return line3D<T>(q,q+e);
451         }
452     };
453     template<typename T>
454     struct triangle3D{
455         point3D<T> a,b,c;
456         triangle3D(){
457             triangle3D(const point3D<T> &a,const
458                 point3D<T> &b,const point3D<T> &c):a(a
459                     ),b(b),c(c){
460             bool point_in(const point3D<T> &p)const{//
461                 點在該平面上的投影在三角形中
462                 return line3D<T>(b,c).same_side(p,a)&&
463                     line3D<T>(a,c).same_side(p,b)&&
464                     line3D<T>(a,b).same_side(p,c);
465             }
466         };
467     };
468     template<typename T>
469     struct tetrahedron{//四面體
470

```

```

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

```

```

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

```

## 1.2 SmallestCircle.cpp

```

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

```

```

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

```

## 1.3 最近點對.cpp

```

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

```

```

9     t.clear();
10     for(int i=l;i<=r;++i)
11         if((v[i].x-v[mid].x)*(v[i].x-v[mid].x)<
            dis)t.push_back(v[i]);
12     sort(t.begin(),t.end(),point<T>::y_cmp);/*
        如果用merge_sort的方式可以O(n)*/
13     for(int i=0;i<(int)t.size();++i)
14         for(int j=1;j<=3&&i+j<(int)t.size();++j)
15             if((tmd=(t[i]-t[i+j]).abs2())<dis)dis=
                tmd;
16     return dis;
17 }
18 template<typename T>
19 inline T closest_pair(vector<point<T> >&v){
20     vector<point<T> >t;
21     sort(v.begin(),v.end(),point<T>::x_cmp);
22     return closest_pair(v,t,0,v.size()-1);/*最
        近點對距離*/
23 }

```

## 1.4 浮點數誤差模板.cpp

```

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

```

## 2 Data\_Structure

### 2.1 DLX.cpp

```

1 #define MAXN 4100
2 #define MAXM 1030
3 #define MAXND 16390
4 struct DLX{
5     int n,m,sz,ansd;//高是n 寬是m的稀疏矩陣
6     int S[MAXN],H[MAXN];
7     int row[MAXN],col[MAXND];//每個節點代表的
        列行
8     int L[MAXN],R[MAXN],U[MAXN],D[MAXND];
9     vector<int> ans,ans1;
10    void init(int _n,int _m){
11        n=_n,m=_m;

```

```

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

```

```

64     DFOR(j,L,i)restore(col[j]);
65 }
66 restore(c);
67 return 0;
68 }
69 void dfs2(int d){ //for最小重複覆蓋問題
70     if(d+h()==ansd)return;
71     if(!R[0]){ansd=d;ans=ans;return;}
72     int c=R[0];
73     DFOR(i,R,0)if(S[i]<S[c])c=i;
74     DFOR(i,D,c){
75         anst.push_back(row[i]);
76         remove2(i);
77         DFOR(j,R,i)remove2(j),--S[col[j]];
78         dfs2(d+1);
79         anst.pop_back();
80         DFOR(j,L,i)restore2(j),++S[col[j]];
81         restore2(i);
82     }
83 }
84 bool exact_cover(){ //解精確覆蓋問題
85     ans.clear(); //答案
86     return dfs(0);
87 }
88 void min_cover(){ //解最小重複覆蓋問題
89     anst.clear(); //暫存用，答案還是存在ans裡
90     dfs2(0);
91 }
92 #undef DFOR
93 };

```

## 2.2 Dynamic\_KD\_tree.cpp

```

1 template<typename T,size_t kd> //有kd個維度
2 class kd_tree{
3 public:
4     struct point{
5         T d[kd];
6         T dist(const point &x)const{
7             T ret=0;
8             for(size_t i=0;i<kd;++i)ret+=std::abs(d[i]-x.d[i]);
9             return ret;
10        }
11        bool operator==(const point &p){
12            for(size_t i=0;i<kd;++i)
13                if(d[i]!=p.d[i])return 0;
14            return 1;
15        }
16        bool operator<(const point &b)const{
17            return d[0]<b.d[0];
18        }
19    };
20 private:
21     struct node{
22         node *l,*r;
23         point pid;
24         int s;
25         node(const point &p):l(0),r(0),pid(p),s(1){}
26         ~node(){delete l;delete r;}
27         void up(){s=(l?l->s:0)+1+(r?r->s:0);}

```

```

28     }*root;
29     const double alpha,loga;
30     const T INF; //記得要給INF，表示極大值
31     int maxn;
32     struct __cmp{
33         int sort_id;
34         bool operator()(const node*x,const node*y)const{
35             return operator()(x->pid,y->pid);
36         }
37         bool operator()(const point &x,const point &y)const{
38             if(x.d[sort_id]!=y.d[sort_id])
39                 return x.d[sort_id]<y.d[sort_id];
40             for(size_t i=0;i<kd;++i)
41                 if(x.d[i]!=y.d[i])return x.d[i]<y.d[i];
42             return 0;
43         }
44     };
45     int size(node *o){return o?o->s:0;}
46     std::vector<node*> A;
47     node* build(int k,int l,int r){
48         if(l>r)return 0;
49         if(k==kd)k=0;
50         int mid=(l+r)/2;
51         cmp.sort_id=k;
52         std::nth_element(A.begin()+l,A.begin()+mid,A.begin()+r+1,cmp);
53         node *ret=A[mid];
54         ret->l=build(k+1,l,mid-1);
55         ret->r=build(k+1,mid+1,r);
56         ret->up();
57         return ret;
58     }
59     bool isbad(node*o){
60         return size(o->l)>alpha*o->s||size(o->r)>alpha*o->s;
61     }
62     void flatten(node *u,typename std::vector<node*>::iterator &it){
63         if(!u)return;
64         flatten(u->l,it);
65         *it=u;
66         flatten(u->r,++it);
67     }
68     void rebuild(node*&u,int k){
69         if((int)A.size()<u->s)A.resize(u->s);
70         typename std::vector<node*>::iterator it=A.begin();
71         flatten(u,it);
72         u=build(k,0,u->s-1);
73     }
74     bool insert(node*&u,int k,const point &x,int dep){
75         if(!u){
76             u=new node(x);
77             return dep<=0;
78         }
79         ++u->s;
80         cmp.sort_id=k;
81         if(insert(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x,dep-1)){
82             if(!isbad(u))return 1;
83             rebuild(u,k);

```

```

84         }
85         return 0;
86     }
87     node *findmin(node*o,int k){
88         if(!o)return 0;
89         if(cmp.sort_id==k)return o->l?findmin(o->l,(k+1)%kd):o;
90         node *l=findmin(o->l,(k+1)%kd);
91         node *r=findmin(o->r,(k+1)%kd);
92         if(l&&!r)return cmp(l,o)?l:o;
93         if(!l&&r)return cmp(r,o)?r:o;
94         if(!l&&!r)return o;
95         if(cmp(l,r))return cmp(l,o)?l:o;
96         return cmp(r,o)?r:o;
97     }
98     bool erase(node *&u,int k,const point &x){
99         if(!u)return 0;
100         if(u->pid==x){
101             if(u->r){
102                 else if(u->l){
103                     u->r=u->l;
104                     u->l=0;
105                 }else{
106                     delete u;
107                     u=0;
108                     return 1;
109                 }
110             }
111             cmp.sort_id=k;
112             u->pid=findmin(u->r,(k+1)%kd)->pid;
113             return erase(u->r,(k+1)%kd,u->pid);
114         }
115         cmp.sort_id=k;
116         if(erase(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x)){
117             --u->s;return 1;
118         }else return 0;
119     }
120     T heuristic(const T h[])const{
121         T ret=0;
122         for(size_t i=0;i<kd;++i)ret+=h[i];
123         return ret;
124     }
125     int qM;
126     std::priority_queue<std::pair<T,point >>pQ;
127     void nearest(node *u,int k,const point &x,T *h,T &mndist){
128         if(u==0||heuristic(h)>=mndist)return;
129         T dist=u->pid.dist(x),old=h[k];
130         /*mndist=std::min(mndist,dist);*/
131         if(dist<mndist){
132             pQ.push(std::make_pair(dist,u->pid));
133             if((int)pQ.size()==qM+1)
134                 mndist=pQ.top().first,pQ.pop();
135         }
136         if(x.d[k]<u->pid.d[k]){
137             nearest(u->l,(k+1)%kd,x,h,mndist);
138             h[k]=std::abs(x.d[k]-u->pid.d[k]);
139             nearest(u->r,(k+1)%kd,x,h,mndist);
140         }else{
141             nearest(u->r,(k+1)%kd,x,h,mndist);
142             h[k]=std::abs(x.d[k]-u->pid.d[k]);
143             nearest(u->l,(k+1)%kd,x,h,mndist);

```



## 2.3 kd\_tree\_replace\_segment

```

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

```

/\*kd樹代替高維線段樹\*/

```

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

```

## 2.3 kd\_tree\_replace\_segment

```

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

```

## 2.4 persistent\_segment\_tree.cpp

```

1 #include<bits/stdc++.h> //POJ 2104
2 using namespace std;
3 struct node{
4     int l,r;
5     int data;

```

```

6     node(int l, int r, int d): l(l), r(r), data(d)
7     {}
8 };
9 vector<node> nds;
10 inline void up(int o, int l, int r){
11     nds[o].data=nds[l].data+nds[r].data;
12 }
13 inline int new_node(int l, int r, int d){
14     nds.push_back(node(l, r, d));
15     return nds.size()-1;
16 }
17 inline int new_node(const node &nd){
18     nds.push_back(nd);
19     return nds.size()-1;
20 }
21 int build_tree(int l, int r){
22     int nd=new_node(-1, -1, 0);
23     if(l==r) return nd;
24     int mid=(l+r)/2;
25     int L=build_tree(l, mid); //執行時vector會被
26     重構
27     int R=build_tree(mid+1, r); //一定要這樣寫
28     nds[nd].l=L;
29     nds[nd].r=R;
30     //up(nd, L, R);
31     return nd;
32 }
33 int insert(int l, int r, int rt, int x, int d){
34     if(x<l || r<x) return rt;
35     int nd=new_node(nds[rt]);
36     if(l==r && l==x) nds[nd].data+=d;
37     else{
38         int mid=(l+r)/2;
39         int L=insert(l, mid, nds[nd].l, x, d);
40         int R=insert(mid+1, r, nds[nd].r, x, d);
41         nds[nd].l=L;
42         nds[nd].r=R;
43         up(nd, L, R);
44     }
45     return nd;
46 }
47 inline int cal(int L, int R){
48     return nds[R].data-nds[L].data;
49 }
50 int find(int l, int r, int L, int R, int k){
51     if(l==r) return l;
52     int mid=(l+r)/2;
53     int add=cal(nds[L].l, nds[R].l);
54     if(k<=add) return find(l, mid, nds[L].l, nds[R
55     ].l, k);
56     return find(mid+1, r, nds[L].r, nds[R].r, k-
57     add);
58 }
59 int n,m;
60 int s[100005];
61 int root[100005];
62 int main(){
63     while(~scanf("%d%d", &n, &m)){
64         nds.clear();
65         vector<int> lsh;
66         for(int i=1; i<=n; ++i){
67             scanf("%d", &s[i]);
68             lsh.push_back(s[i]);
69         }
70         sort(lsh.begin(), lsh.end());

```

```

67 lsh.resize(unique(lsh.begin(),lsh.end())
68           -lsh.begin());
69 int N=(int)lsh.size()-1;
70 root[0]=build_tree(0,N);
71 for(int i=1;i<=n;++i){
72     s[i]=lower_bound(lsh.begin(),lsh.end())
73         ,s[i])-lsh.begin();
74     root[i]=insert(0,N,root[i-1],s[i],1);
75 }
76 while(m--){
77     int a,b,k;
78     scanf("%d%d%d",&a,&b,&k);
79     int res=find(0,N,root[a-1],root[b],k);
80     printf("%d\n",lsh[res]);
81 }
82 return 0;

```

## 2.5 reference\_point.cpp

```

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

```

## 2.6 skew\_heap.cpp

```

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

```

## 2.7 split\_merge.cpp

```

1 void split(node *o,node *a,node *b,int k){
2     if(!o)a=b=0;
3     else{
4         //o=new node(*o);
5         o->down();
6         if(k<=size(o->l)){
7             b=0;
8             split(o->l,a,b->l,k);
9         }else{
10            a=0;
11            split(o->r,a->r,b,k-size(o->l)-1);
12        }
13        o->up();
14    }
15 }
16 node *merge(node *a,node *b){
17     if(!a||!b)return a?a:b;
18     static int x;
19     if(x++%(a->s+b->s)<a->s){
20         //a=new node(*a);
21         a->down();
22         a->r=merge(a->r,b);
23         a->up();
24         return a;
25     }else{
26         //b=new node(*b);
27         b->down();
28         b->l=merge(a,b->l);
29         b->up();
30         return b;
31     }
32 }

```

## 2.8 treap.cpp

```

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

```

```

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

```

```

87     else k=o->ch[0]->s+1,o=o->ch[1];
88 }
89 const T&operator[](int k){
90     return kth(k);
91 }
92 const T&preorder(const T&data){
93     node *x=root,*y=0;
94     while(x->s)
95         if(x->data<data)y=x,x=x->ch[1];
96     else x=x->ch[0];
97     if(y)return y->data;
98     return data;
99 }
100 const T&successor(const T&data){
101     node *x=root,*y=0;
102     while(x->s)
103         if(data<x->data)y=x,x=x->ch[0];
104     else x=x->ch[1];
105     if(y)return y->data;
106     return data;
107 }
108 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=l;////////
4     int mid=(l+r)/2;
5     do_thing(l,mid);//做答案<=mid會做的事
6     vector<Item> left=vs裡滿足的;
7     vector<Item> right=vs-left;
8     undo_thing(l,mid);
9     BS(l,mid,left);
10    do_thing(l,mid);
11    BS(mid+1,r,right);////////
12 }

```

## 3 default

### 3.1 debug.cpp

```

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

```

### 3.2 ext.cpp

```

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

```

### 3.3 IncStack.cpp

```

1 //Magic
2 #pragma GCC optimize "Ofast"
3 //stack resize,change esp to rsp if 64-bit
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    }

```

### 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 // !/bin/bash
10 // g++ -std=c++11 -O2 -Wall -Wextra -Wno-
11     unused-variable -DDEBUG $1 && ./a.out

```

## 4 Flow

### 4.1 dinic.cpp

```

1 template<typename T>
2 struct DINIC{
3     static const int MAXN=105;
4     static const T INF=INT_MAX;
5     int n;//點數
6     int level[MAXN],cur[MAXN];
7     struct edge{
8         int v,pre;
9         T cap,flow,r;
10    } edge(int v,int pre,T cap):v(v),pre(pre),
11        cap(cap),flow(0),r(cap){}
12 };
13 int g[MAXN];
14 vector<edge> e;
15 void init(int _n){
16     memset(g,-1,sizeof(int)*((n=_n)+1));
17     e.clear();
18 }
19 void add_edge(int u,int v,T cap,bool
20     directed=false){
21     e.push_back(edge(v,g[u],cap));
22     g[u]=e.size()-1;
23     e.push_back(edge(u,g[v],directed?0:cap));
24     g[v]=e.size()-1;
25 }
26 int bfs(int s,int t){
27     memset(level,0,sizeof(int)*(n+1));
28     memcpy(cur,g,sizeof(int)*(n+1));
29     queue<int> q;
30     q.push(s);
31     level[s]=1;
32     while(q.size()){
33         int u=q.front();q.pop();
34         for(int i=g[u];~i;i=e[i].pre){
35             if(!level[e[i].v]&&e[i].r){
36                 level[e[i].v]=level[u]+1;
37                 q.push(e[i].v);
38                 if(e[i].v==t)return 1;
39             }
40         }
41     }
42     return 0;
43 }
44 T dfs(int u,int t,T cur_flow=INF){

```

```

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

```

### 4.2 ISAP\_with\_cut.cpp

```

1 template<typename T>
2 struct ISAP{
3     static const int MAXN=105;
4     static const T INF=INT_MAX;
5     int n;//點數
6     int d[MAXN],gap[MAXN],cur[MAXN];
7     struct edge{
8         int v,pre;
9         T cap,flow,r;
10    } edge(int v,int pre,T cap):v(v),pre(pre),
11        cap(cap),flow(0),r(cap){}
12 };
13 int g[MAXN];
14 vector<edge> e;
15 void init(int _n){
16     memset(g,-1,sizeof(int)*(n=_n)+1);
17     e.clear();
18 }
19 void add_edge(int u,int v,T cap,bool
20     directed=false){
21     e.push_back(edge(v,g[u],cap));
22     g[u]=e.size()-1;
23     e.push_back(edge(u,g[v],directed?0:cap));
24     g[v]=e.size()-1;
25 }
26 T dfs(int u,int s,int t,T cur_flow=INF){
27     if(u==t)return cur_flow;
28     T tf=cur_flow,df;
29     for(int &i=cur[u];~i;i=e[i].pre){
30         if(e[i].r&&d[u]==d[e[i].v]+1){

```

```

29     df=dfs(e[i].v,s,t,min(tf,e[i].r));
30     e[i].flow+=df;
31     e[i^1].flow-=df;
32     e[i].r-=df;
33     e[i^1].r+=df;
34     if(!(tf==df)||d[s]==n)return
        cur_flow-tf;
35 }
36 }
37 int mh=n;
38 for(int i=cur[u]=g[u];~i;i=e[i].pre){
39     if(e[i].r&&d[e[i].v]<mh)mh=d[e[i].v];
40 }
41 if(!--gap[d[u]])d[s]=n;
42 else ++gap[d[u]]=mh;
43 return cur_flow-tf;
44 }
45 T isap(int s,int t,bool clean=true){
46     memset(d,0,sizeof(int)*(n+1));
47     memset(gap,0,sizeof(int)*(n+1));
48     memcpy(cur,g,sizeof(int)*(n+1));
49     if(clean){
50         for(size_t i=0;i<e.size();++i){
51             e[i].flow=0;
52             e[i].r=e[i].cap;
53         }
54     }
55     T max_flow=0;
56     for(gap[0]=n;d[s]<n;max_flow+=dfs(s,s,t))
57         return max_flow;
58 }
59 vector<int> cut_e;//最小割邊集
60 bool vis[MAXN];
61 void dfs_cut(int u){
62     vis[u]=1;//表示u屬於source的最小割集
63     for(int i=g[u];~i;i=e[i].pre){
64         if(e[i].flow<e[i].cap&&!vis[e[i].v])
65             dfs_cut(e[i].v);
66     }
67 }
68 T min_cut(int s,int t){
69     T ans=isap(s,t);
70     memset(vis,0,sizeof(bool)*(n+1));
71     dfs_cut(s),cut_e.clear();
72     for(int u=0;u<n;++u){
73         if(vis[u])for(int i=g[u];~i;i=e[i].pre)
74             if(!vis[e[i].v])cut_e.push_back(i);
75     }
76     return ans;
77 }
78 };

```

### 4.3 MinCostMaxFlow.cpp

```

1 template<typename _T>
2 struct MCMF{
3     static const int MAXN=440;
4     static const _T INF=999999999;
5     struct edge{

```

```

6         int v,pre;
7         _T cap,cost;
8         edge(int v,int pre,_T cap,_T cost):v(v),
9             pre(pre),cap(cap),cost(cost){}
10    };
11    int n,S,T;
12    _T dis[MAXN],piS,ans;
13    bool vis[MAXN];
14    vector<edge> e;
15    int g[MAXN];
16    void init(int _n){
17        memset(g,-1,sizeof(int)*((n=_n)+1));
18        e.clear();
19    }
20    void add_edge(int u,int v,_T cap,_T cost,
21        bool directed=false){
22        e.push_back(edge(v,g[u],cap,cost));
23        g[u]=e.size()-1;
24        e.push_back(edge(u,g[v],directed?0:cap,-
25            cost));
26        g[v]=e.size()-1;
27    }
28    _T augment(int u,_T cur_flow){
29        if(u==T||!cur_flow)return ans+=piS*
30            cur_flow,cur_flow;
31        vis[u]=1;
32        _T r=cur_flow,d;
33        for(int i=g[u];~i;i=e[i].pre){
34            if(e[i].cap&&!e[i].cost&&!vis[e[i].v])
35                {
36                    d=augment(e[i].v,min(r,e[i].cap));
37                    e[i].cap-=d;
38                    e[i^1].cap+=d;
39                    if(!r==d)break;
40                }
41        }
42        return cur_flow-r;
43    }
44    bool modlabel(){
45        for(int u=0;u<n;++u)dis[u]=INF;
46        static deque<int>q;
47        dis[T]=0,q.push_back(T);
48        while(q.size()){
49            int u=q.front();q.pop_front();
50            _T dt;
51            for(int i=g[u];~i;i=e[i].pre){
52                if(e[i^1].cap&&(dt=dis[u]-e[i].cost)
53                    <dis[e[i].v]){
54                    if((dis[e[i].v]=dt)<=dis[q.size()])
55                        q.front():S){}
56                    q.push_front(e[i].v);
57                }else q.push_back(e[i].v);
58            }
59        }
60    }
61    T mincost(int s,int t){
62        S=s,T=t;
63        piS=ans=0;
64        while(modlabel()){
65            do memset(vis,0,sizeof(bool)*(n+1));

```

```

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

```

## 5 Graph

### 5.1 Augmenting\_Path.cpp

```

1 #define MAXN1 505
2 #define MAXN2 505
3 int n1,n2;//n1個點連向n2個點
4 int match[MAXN2];//屬於n2的點匹配了哪個點
5 vector<int> g[MAXN1];//圖
6 bool vis[MAXN2];//是否走訪過
7 bool dfs(int u){
8     for(size_t i=0;i<g[u].size();++i){
9         int v=g[u][i];
10        if(vis[v])continue;
11        vis[v]=1;
12        if(match[v]==-1||dfs(match[v])){
13            match[v]=u;
14            return 1;
15        }
16    }
17    return 0;
18 }
19 inline int max_match(){
20     int ans=0;
21     memset(match,-1,sizeof(int)*n2);
22     for(int i=0;i<n1;++i){
23         memset(vis,0,sizeof(bool)*n2);
24         if(dfs(i))++ans;
25     }
26     return ans;
27 }

```

### 5.2 Augmenting\_Path\_multiple

```

1 #define MAXN1 1005
2 #define MAXN2 505
3 int n1,n2;//n1個點連向n2個點，其中n2個點可以
4     匹配很多邊
5 vector<int> g[MAXN1];//圖
6 int c[MAXN2];//每個屬於n2點最多可以接受幾條
7     匹配邊
8 vector<int> match_list[MAXN2];//每個屬於n2的
9     點匹配了那些點
10 bool vis[MAXN2];//是否走訪過
11 bool dfs(int u){
12     for(size_t i=0;i<g[u].size();++i){
13         int v=g[u][i];
14         if(vis[v])continue;
15         vis[v]=true;
16         if((int)match_list[v].size()<c[v]){

```

```

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

```

### 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 inline int lca(int x,int y){
6     for(++t;swap(x,y)){
7         if(x==0)continue;
8         if(v[x]==t)return x;
9         v[x]=t;
10        x=st[pa[match[x]]];
11    }
12 }
13 #define qpush(x) q.push(x),S[x]=0
14 inline void flower(int x,int y,int l,queue<
15     int> &q){
16     while(st[x]!=1){
17         pa[x]=y;
18         if(S[y==match[x]]==1)qpush(y);
19         st[x]=st[y]=1,x=pa[y];
20     }
21 }
22 inline bool bfs(int x){
23     for(int i=1;i<=n;++i)st[i]=i;
24     memset(S+1,-1,sizeof(int)*n);
25     queue<int>q;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])

```



```

33     lst=match[x],match[x]=y,match[y
34         ]=x;
35     return 1;
36     }
37     qpush(match[y]);
38 }else if(!S[y]&&st[y]!=st[x]){
39     int l=lca(y,x);
40     flower(y,x,l,q),flower(x,y,l,q);
41 }
42 }
43 return 0;
44 }
45 inline int blossom(){
46     int ans=0;
47     for(int i=1;i<=n;++i)
48         if(!match[i]&&bfs(i))++ans;
49     return ans;
50 }

```

## 5.4 graphISO.cpp

```

1  const int MAXN=1005,K=30; //K要夠大
2  const long long A=3,B=11,C=2,D=19,P=0
3  xdefaced;
4  long long f[K+1][MAXN];
5  vector<int> g[MAXN],rg[MAXN];
6  int n;
7  inline void init(){
8      for(int i=0;i<n;++i){
9          f[0][i]=1;
10         g[i].clear();
11         rg[i].clear();
12     }
13 }
14 inline void add_edge(int u,int v){
15     g[u].push_back(v);
16     rg[v].push_back(u);
17 }
18 inline long long point_hash(int u){//O(N)
19     for(int t=1;t<=K;++t){
20         for(int i=0;i<n;++i){
21             f[t][i]=f[t-1][i]*A%P;
22             for(int j:g[i])f[t][i]=(f[t][i]+f[t-1][j]*B%P)%P;
23             for(int j:rg[i])f[t][i]=(f[t][i]+f[t-1][j]*C%P)%P;
24             if(i==u)f[t][i]+=D; //如果圖太大的話，
25             //把這行刪掉，執行一次後f[K]就會是所
26             //有點的答案
27             f[t][i]%=P;
28         }
29     }
30     return f[K][u];
31 }
32 inline vector<long long> graph_hash(){
33     vector<long long> ans;
34     for(int i=0;i<n;++i)ans.push_back(
35         point_hash(i)); //O(N^2)
36     sort(ans.begin(),ans.end());
37     return ans;
38 }

```

## 5.5 KM.cpp

```

1  #define MAXN 100
2  int n;
3  int g[MAXN][MAXN],lx[MAXN],ly[MAXN],slack_y[
4  MAXN];
5  int match_y[MAXN];
6  bool vx[MAXN],vy[MAXN]; //要保證g是完全二分圖
7  bool dfs(int x,bool adjust=1){ //DFS找增廣
8  路 · is=1表示要交換邊
9      if(vx[x])return 0;
10     vx[x]=1;
11     for(int y=0;y<n;++y){
12         if(vy[y])continue;
13         int t=lx[x]+ly[y]-g[x][y];
14         if(t==0){
15             vy[y]=1;
16             if(match_y[y]==-1||dfs(match_y[y],
17                 adjust)){
18                 if(adjust)match_y[y]=x;
19                 return 1;
20             }
21         }else if(slack_y[y]>t)slack_y[y]=t;
22     }
23     return 0;
24 }
25 inline int km(){
26     memset(ly,0,sizeof(int)*n);
27     memset(match_y,-1,sizeof(int)*n);
28     for(int x=0;x<n;++x){
29         lx[x]=0;
30         for(int y=0;y<n;++y){
31             lx[x]=max(lx[x],g[x][y]);
32         }
33     }
34     for(int x=0;x<n;++x){
35         for(int y=0;y<n;++y)slack_y[y]=INT_MAX;
36         memset(vx,0,sizeof(bool)*n);
37         memset(vy,0,sizeof(bool)*n);
38         if(dfs(x))continue;
39         bool flag=1;
40         while(flag){
41             int cut=INT_MAX;
42             for(int y=0;y<n;++y){
43                 if(!vy[y]&&slack_y[y]==0){
44                     vy[y]=1;
45                     if(match_y[y]==-1||dfs(match_y[y],
46                         1,0)){
47                         flag=0; //測試成功 · 有增廣路
48                         break;
49                     }
50                 }
51             }
52         }
53     }
54     return 0;
55 }
56 }
57 memset(vx,0,sizeof(bool)*n);

```

```

58     memset(vy,0,sizeof(bool)*n);
59     dfs(x); //最後要記得將邊翻反轉
60 }
61 int ans=0;
62 for(int y=0;y<n;++y)ans+=g[match_y[y]][y];
63 return ans;
64 }

```

## 5.6 MaximumClique.cpp

```

1  struct MaxClique{
2      static const int MAXN=105;
3      int N,ans;
4      int g[MAXN][MAXN],dp[MAXN],stk[MAXN][MAXN
5      ];
6      int sol[MAXN],tmp[MAXN]; //sol[0~ans-1]為答
7      案
8      void init(int n){
9          N=n; //0-base
10         memset(g,0,sizeof(g));
11     }
12     void add_edge(int u,int v){
13         g[u][v]=g[v][u]=1;
14     }
15     int dfs(int ns,int dep){
16         if(!ns){
17             if(dep>ans){
18                 ans=dep;
19                 memcpy(sol,tmp,sizeof tmp);
20                 return 1;
21             }else return 0;
22         }
23         for(int i=0;i<ns;++i){
24             if(dep+ns-i==ans)return 0;
25             int u=stk[dep][i],cnt=0;
26             if(dep+dp[u]<=ans)return 0;
27             for(int j=i+1;j<ns;++j){
28                 int v=stk[dep][j];
29                 if(g[u][v])stk[dep+1][cnt++]=v;
30             }
31             tmp[dep]=u;
32             if(dfs(cnt,dep+1))return 1;
33         }
34         return 0;
35     }
36     int clique(){
37         int u,v,ns;
38         for(ans=0,u=N-1;u>0;--u){
39             for(ns=0,tmp[0]=u,v=u+1;v<N;v++){
40                 if(g[u][v])stk[1][ns++]=v;
41             }
42             dfs(ns,1),dp[u]=ans;
43         }
44     }
45 }

```

## 5.7 MinimumMeanCycle.cpp

```

1  #include<cstdint> //for DBL_MAX
2  int dp[100000][100000];
3  double mnc(int n){
4      int u,v,w;
5      const int inf=0x7f7f7f7f;
6      memset(dp,0x7f,sizeof(dp));
7      memset(dp[0],0,sizeof(dp[0]));
8      for(int i=0;i<n;++i){
9          for(auto e:E){ //tuple<int,int,int>
10             of u,v,w
11             tie(u,v,w)=e;
12             if(dp[i][u]!=inf)
13                 dp[i+1][v]=min(dp[i+1][v],dp
14                     [i][u]+w);
15             }
16         }
17         double res = DBL_MAX;
18         for(int i=1;i<=n;++i){
19             double val = DBL_MIN;
20             for(int j=0;j<n;++j)
21                 val=max(val,double(dp[n][i]-
22                     dp[i][j])/(n-j));
23             res=min(res,val);
24         }
25     }
26     return res;
27 }

```

## 5.8 Minimum\_General\_Weighted

```

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

```

```

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

```

## 5.9 Rectilinear\_Steiner\_tree.cpp

```

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

```

```

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

```

## 5.10 treeISO.cpp

```

1 const int MAXN=100005;
2 const long long X=12327,P=0xdefaced;
3 vector<int> g[MAXN];
4 bool vis[MAXN];
5 long long dfs(int u){//hash ver
6     vis[u]=1;
7     vector<long long> tmp;
8     for(auto v:g[u])if(!vis[v])tmp.push_back(dfs(v));
9     if(tmp.empty())return 177;
10    long long ret=4931;
11    sort(tmp.begin(),tmp.end());
12    for(auto v:tmp)ret=((ret*X)^v)%P;
13    return ret;
14 }
15
16 string dfs(int x,int p){
17     vector<string> c;
18     for(int y:g[x])
19         if(y!=p)c.emplace_back(dfs(y,x));
20     sort(c.begin(),c.end());
21     string ret("");
22     for(auto &s:c)ret+=s;
23     ret+=")";
24     return ret;
25 }

```

## 5.11 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*2+1],pa[MAXN*2+1];
14 int flower_from[MAXN*2+1][MAXN+1],S[MAXN*2+1],vis[MAXN*2+1];
15 vector<int> flower[MAXN*2+1];
16 queue<int> q;
17 int e_delta(const edge &e){ // does not work inside blossoms
18     return lab[e.u]+lab[e.v]-g[e.u][e.v].w*2;
19 }
20 void update_slack(int u,int x){
21     if(!slack[x]||e_delta(g[u][x])<e_delta(g[slack[x]][x]))slack[x]=u;
22 }
23 void set_slack(int x){
24     slack[x]=0;
25     for(int u=1;u<=n;u++){
26         if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)update_slack(u,x);

```

```

27 }
28 void q_push(int x){
29     if(x<=n)q.push(x);
30     else for(size_t i=0;i<flower[x].size();i++)q_push(flower[x][i]);
31 }
32 void set_st(int x,int b){
33     st[x]=b;
34     if(x>n)for(size_t i=0;i<flower[x].size();i++){
35         set_st(flower[x][i],b);
36     }
37 int get_pr(int b,int xr){
38     int pr=find(flower[b].begin(),flower[b].end(),xr)-flower[b].begin();
39     if(pr%2==1){//檢查他在前一層是奇點還是偶點
40         reverse(flower[b].begin()+1,flower[b].end());
41         return (int)flower[b].size()-pr;
42     }else return pr;
43 }
44 void set_match(int u,int v){
45     match[u]=g[u][v].v;
46     if(u>n){
47         edge e=g[u][v];
48         int xr=flower_from[u][e.u],pr=get_pr(u,xr);
49         for(int i=0;i<pr;i++)set_match(flower[u][i],flower[u][i^1]);
50         set_match(xr,v);
51         rotate(flower[u].begin(),flower[u].begin()+pr,flower[u].end());
52     }
53 }
54 void augment(int u,int v){
55     for(;;){
56         int xnv=st[match[u]];
57         set_match(u,v);
58         if(!xnv)return;
59         set_match(xnv,st[pa[xnv]]);
60         u=st[pa[xnv]],v=xnv;
61     }
62 }
63 int get_lca(int u,int v){
64     static int t=0;
65     for(++t;u|v;swap(u,v)){
66         if(u==0)continue;
67         if(vis[u]==t)return u;
68         vis[u]=t;//這種方法可以不用清空v陣列
69         u=st[match[u]];
70         if(u==0)u=st[pa[u]];
71     }
72     return 0;
73 }
74 void add_blossom(int u,int lca,int v){
75     int b=n+1;
76     while(b<=n_x&&st[b]==b){
77         if(b>n_x)b++;
78         lab[b]=0,S[b]=0;
79         match[b]=match[lca];
80         flower[b].clear();
81         flower[b].push_back(lca);
82         for(int x=u,y;lca;x=st[pa[y]])
83             flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);

```

```

84 reverse(flower[b].begin()+1, flower[b].end 143
85 ()); 144
86 for(int x=v, y; x!=lca; x=st[pa[y]]) 145
87 flower[b].push_back(x), flower[b]. 146
88 push_back(y=st[match[x]]), q_push(y); 147
89 set_st(b, b); 148
90 for(int x=1; x<=n_x; ++x) g[b][x].w=g[x][b].w 149
91 =0; 150
92 for(int x=1; x<=n; ++x) flower_from[b][x]=0; 151
93 for(size_t i=0; i<flower[b].size(); ++i){ 152
94 int xs=flower[b][i]; 153
95 for(int x=1; x<=n_x; ++x) 154
96 if(g[b][x].w==0 || e_delta(g[xs][x])< 155
97 e_delta(g[b][x])) 156
98 g[b][x]=g[xs][x], g[x][b]=g[x][xs]; 157
99 for(int x=1; x<=n; ++x) 158
100 if(flower_from[xs][x]) flower_from[b][x 159
101 ]=xs; 160
102 set_slack(b); 161
103 } 162
104 void expand_blossom(int b){ // S[b] == 1 163
105 for(size_t i=0; i<flower[b].size(); ++i) 164
106 set_st(flower[b][i], flower[b][i]); 165
107 int xr=flower_from[b][g[b][pa[b]].u], pr= 166
108 get_pr(b, xr); 167
109 for(int i=0; i<pr; i+=2){ 168
110 int xs=flower[b][i], xns=flower[b][i+1]; 169
111 pa[xs]=g[xns][xs].u; 170
112 S[xs]=1, S[xns]=0; 171
113 slack[xs]=0, set_slack(xns); 172
114 q_push(xns); 173
115 } 174
116 S[xr]=1, pa[xr]=pa[b]; 175
117 for(size_t i=pr+1; i<flower[b].size(); ++i){ 176
118 int xs=flower[b][i]; 177
119 S[xs]=-1, set_slack(xs); 178
120 } 179
121 st[b]=0; 180
122 } 181
123 bool on_found_edge(const edge &e){ 182
124 int u=st[e.u], v=st[e.v]; 183
125 if(S[v]==-1){ 184
126 pa[v]=e.u, S[v]=1; 185
127 int nu=st[match[v]]; 186
128 slack[v]=slack[nu]=0; 187
129 S[nu]=0, q_push(nu); 188
130 }else if(S[v]==0){ 189
131 int lca=get_lca(u, v); 190
132 if(!lca){ 191
133 augment(u, v), augment(v, u); 192
134 return true; 193
135 }else add_blossom(u, lca, v); 194
136 } 195
137 return false; 196
138 } 197
139 bool matching(){ 198
140 memset(S+1, -1, sizeof(int)*n_x); 199
141 memset(slack+1, 0, sizeof(int)*n_x); 200
142 q=queue<int>(); 201
143 for(int x=1; x<=n_x; ++x) 202
144 if(st[x]==x&&!match[x]) pa[x]=0, S[x]=0, 203
145 q_push(x); 204
146 if(q.empty()) return false; 205
147 for(;;){ 206
148 while(q.size()){ 207

```

```

143 int u=q.front(); q.pop(); 201
144 if(S[st[u]]==1) continue; 202
145 for(int v=1; v<=n; ++v) 203
146 if(g[u][v].w>0&&st[u]!=st[v]){ 204
147 if(e_delta(g[u][v])==0){ 205
148 if(on_found_edge(g[u][v])) return 206
149 true; 207
150 }else update_slack(u, st[v]); 208
151 } 209
152 } 210
153 int d=INF; 211
154 for(int b=n+1; b<=n_x; ++b) 212
155 if(st[b]==b&&S[b]==1) d=min(d, lab[b]/2) 213
156 ; 214
157 for(int x=1; x<=n_x; ++x) 215
158 if(st[x]==x&&slack[x]){ 216
159 if(S[x]==-1) d=min(d, e_delta(g[slack[ 217
160 x]][x])); 218
161 else if(S[x]==0) d=min(d, e_delta(g[ 219
162 slack[x]][x])/2); 220
163 } 221
164 for(int u=1; u<=n; ++u){ 222
165 if(S[st[u]]==0){ 223
166 if(lab[u]<=d) return 0; 224
167 lab[u]=-d; 225
168 }else if(S[st[u]]==1) lab[u]+=d; 226
169 } 227
170 for(int b=n+1; b<=n_x; ++b) 228
171 if(st[b]==b){ 229
172 if(S[st[b]]==0) lab[b]+d*2; 230
173 else if(S[st[b]]==1) lab[b]-d*2; 231
174 } 232
175 q=queue<int>(); 233
176 for(int x=1; x<=n_x; ++x) 234
177 if(st[x]==x&&slack[x]&&st[slack[x]]!=x 235
178 &&e_delta(g[slack[x]][x])==0) 236
179 if(on_found_edge(g[slack[x]][x])) 237
180 return true; 238
181 for(int b=n+1; b<=n_x; ++b) 239
182 if(st[b]==b&&S[b]==1&&lab[b]==0) 240
183 expand_blossom(b); 241
184 } 242
185 pair<long long, int> weight_blossom(){ 243
186 memset(match+1, 0, sizeof(int)*n); 244
187 n_x=n; 245
188 int n_matches=0; 246
189 long long tot_weight=0; 247
190 for(int u=0; u<=n; ++u) st[u]=u, flower[u]. 248
191 clear(); 249
192 int w_max=0; 250
193 for(int u=1; u<=n; ++u) 251
194 for(int v=1; v<=n; ++v){ 252
195 flower_from[u][v]=(u==v?u:0); 253
196 w_max=max(w_max, g[u][v].w); 254
197 } 255
198 for(int u=1; u<=n; ++u) lab[u]=w_max; 256
199 while(matching()) ++n_matches; 257
200 for(int u=1; u<=n; ++u) 258
201 if(match[u]&&match[u]<u) 259
202 tot_weight+=g[u][match[u]].w; 260
203 return make_pair(tot_weight, n_matches); 261
204 } 262
205 void init_weight_graph(){ 263
206 for(int u=1; u<=n; ++u) 264

```

```

201 for(int v=1; v<=n; ++v) 201
202 g[u][v]=edge(u, v, 0); 202
203 } 203
204 int main(){ 204
205 int m; 205
206 scanf("%d%d", &n, &m); 206
207 init_weight_graph(); 207
208 for(int i=0; i<=m; ++i){ 208
209 int u, v, w; 209
210 scanf("%d%d%d", &u, &v, &w); 210
211 g[u][v].w=g[v][u].w=w; 211
212 } 212
213 printf("%lld\n", weight_blossom().first); 213
214 for(int u=1; u<=n; ++u) printf("%d ", match[u 214
215 ]); puts(""); 215
216 return 0; 216
217 } /*7 20 217
218 5 7 9 3 7 4 3 6 6 2 5 8 5 1 9 1 3 6 6 5 1 218
219 2 7 4 2 3 5 6 4 2 7 1 5 5 4 4 1 3 5 3 9 219
220 7 6 4 2 1 3 4 3 9 6 2 7 4 2 8 6 1 10 220
221 ----- 221
222 6 0 4 3 7 1 5 */ 222

```

## 5.12 全局最小割.cpp

```

1 const int INF=0x3f3f3f3f; 1
2 template<typename T> 2
3 struct stoer_wagner{// 0-base 3
4 static const int MAXN=150; 4
5 T g[MAXN][MAXN], dis[MAXN]; 5
6 int nd[MAXN], n, s, t; 6
7 void init(int _n){ 7
8 n=_n; 8
9 for(int i=0; i<=n; ++i) 9
10 for(int j=0; j<=n; ++j) g[i][j]=0; 10
11 } 11
12 void add_edge(int u, int v, T w){ 12
13 g[u][v]=g[v][u]+w; 13
14 } 14
15 T min_cut(){ 15
16 T ans=INF; 16
17 for(int i=0; i<=n; ++i) nd[i]=i; 17
18 for(int ind, tn=n; tn>1; --tn){ 18
19 for(int i=1; i<tn; ++i) dis[nd[i]]=0; 19
20 for(int i=1; i<tn; ++i){ 20
21 ind=i; 21
22 for(int j=i; j<tn; ++j){ 22
23 dis[nd[j]]+=g[nd[i-1]][nd[j]]; 23
24 if(dis[nd[ind]]<dis[nd[j]]) ind=j; 24
25 } 25
26 swap(nd[ind], nd[i]); 26
27 } 27
28 if(ans>dis[nd[ind]]) ans=dis[t=nd[ind 28
29 ]], s=nd[ind-1]; 29
30 for(int i=0; i<tn; ++i) 30
31 g[nd[ind-1]][nd[i]]=g[nd[i]][nd[ind 31
32 -1]]+g[nd[i]][nd[ind]]; 32
33 } 33
34 return ans; 34

```

## 5.13 最小樹形圖 \_\_ 朱劉.cpp

```

1 #define INF 0x3f3f3f3f 1
2 template<typename T> 2
3 struct zhu_liu{ 3
4 static const int MAXN=110; 4
5 struct edge{ 5
6 int u, v; 6
7 T w; 7
8 edge(int u=0, int v=0, T w=0):u(u), v(v), w( 8
9 w){} 9
10 vector<edge> E; // 0-base 10
11 int pe[MAXN], id[MAXN], vis[MAXN]; 11
12 T in[MAXN]; 12
13 void init(){ E.clear(); } 13
14 void add_edge(int u, int v, T w){ 14
15 if(u!=v) E.push_back(edge(u, v, w)); 15
16 } 16
17 T build(int root, int n){ 17
18 T ans=0; int N=n; 18
19 for(;;){ 19
20 for(int u=0; u<=n; ++u) in[u]=INF; 20
21 for(size_t i=0; i<E.size(); ++i) 21
22 if(E[i].u!=E[i].v&&E[i].w<in[E[i].v]) 22
23 pe[E[i].v]=i, in[E[i].v]=E[i].w; 23
24 for(int u=0; u<=n; ++u) //uL, 24
25 if(u!=root&&in[u]==INF) return -INF; 25
26 int cntnode=0; 26
27 memset(id, -1, sizeof(int)*N); 27
28 memset(vis, -1, sizeof(int)*N); 28
29 for(int u=0; u<=n; ++u){ 29
30 if(u!=root) ans+=in[u]; 30
31 int v=u; 31
32 for(;; vis[v]!=u&&id[v]==-1&&v!=root; v 32
33 =E[pe[v]].u) 33
34 vis[v]=u; 34
35 if(v!=root&&id[v]==-1){ 35
36 for(int x=E[pe[v]].u; x!=v; x=E[pe[x 36
37 ]].u) 37
38 id[x]=cntnode; 38
39 id[v]=cntnode++; 39
40 } 40
41 if(!cntnode) break; //uL 41
42 for(int u=0; u<=n; ++u) if(id[u]==-1) id[u 42
43 ]=cntnode++; 43
44 for(size_t i=0; i<E.size(); ++i){ 44
45 int v=E[i].v; 45
46 E[i].u=id[E[i].u]; 46
47 E[i].v=id[E[i].v]; 47
48 if(E[i].u!=E[i].v) E[i].w-=in[v]; 48
49 } 49
50 n=cntnode; 50
51 root=id[root]; 51
52 } 52
53 return ans; 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)
7     ,cost(c){}
8 };
9 int state;//規則數量
10 map<char,int> rule;//每個字元對應到的規則
11 //小寫字母為終端字符
12 vector<CNF> cnf;
13 inline void init(){
14     state=0;
15     rule.clear();
16     cnf.clear();
17 }
18 inline void add_to_cnf(char s,const string &
19 p,int cost){
20     //加入一個s -> <p>的文法，代價為cost
21     if(rule.find(s)==rule.end())rule[s]=state++;
22     for(auto c:p)if(rule.find(c)==rule.end())
23         rule[c]=state++;
24     if(p.size()==1){
25         cnf.push_back(CNF(rule[s],rule[p[0]],-1,
26             cost));
27     }else{
28         int left=rule[s];
29         int sz=p.size();
30         for(int i=0;i<sz-2;++i){
31             cnf.push_back(CNF(left,rule[p[i]],
32                 state,0));
33             left=state++;
34         }
35         cnf.push_back(CNF(left,rule[p[sz-2]],
36             rule[p[sz-1]],cost));
37     }
38 }
39 vector<long long> dp[MAXN][MAXN];
40 vector<bool> neg_INF[MAXN][MAXN];//如果花費
41 //是負的可能會有無限小的情形
42 inline void relax(int l,int r,const CNF &c,
43     long long cost,bool neg_c=0){
44     if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x]
45         ||cost<dp[l][r][c.s])){
46         if(neg_c||neg_INF[l][r][c.x]){
47             dp[l][r][c.s]=0;
48             neg_INF[l][r][c.s]=true;
49         }else dp[l][r][c.s]=cost;
50     }
51 }
52 inline void bellman(int l,int r,int n){
53     for(int k=1;k<=state;++k)
54         for(auto c:cnf)
55             if(c.y== -1)relax(l,r,c,dp[l][r][c.x]+
56                 c.cost,k=n);
57 }

```

```

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,
51                 INT_MAX);
52             neg_INF[i][j]=vector<bool>(state+1,
53                 false);
54         }
55         dp[i][i][tok[i]]=0;
56         bellman(i,i,tok.size());
57     }
58     for(int r=1;r<(int)tok.size();++r){
59         for(int l=r-1;l>=0;--l){
60             for(int k=1;k<r;++k)
61                 for(auto c:cnf)
62                     if(~c.y)relax(l,r,c,dp[l][k][c.x]+
63                         dp[k+1][r][c.y]+c.cost);
64             bellman(l,r,tok.size());
65         }
66     }
67 }

```

### 6.2 earley.cpp

```

1 struct Rule{
2     vector<vector<Rule*> > p;
3     void add(const vector<Rule*> &l){
4         p.push_back(l);
5     }
6 };
7 map<string,Rule*> NameRule;
8 map<Rule*,string> RuleName;
9 inline void init_Rule(){
10     for(auto r:RuleName)delete r.first;
11     RuleName.clear();
12     NameRule.clear();
13 }
14 inline Rule *add_rule(const string &s){
15     if(NameRule.find(s)!=NameRule.end())return
16         NameRule[s];
17     Rule *r=new Rule();
18     RuleName[r]=s;
19     NameRule[s]=r;
20     return r;
21 }
22 typedef vector<Rule*> production;
23 struct State{
24     Rule *r;
25     int rid,dot_id,start,end;
26     State(Rule *r,int rid,int dot_id,int start):r(
27         r),rid(rid),dot_id(dot_id),start(start),
28         end(-1){}
29     State(Rule *r=0,int col=0):r(r),rid(-1),
30         dot_id(-1),start(-1),end(col){}
31     bool completed()const{
32         return rid== -1||dot_id>=(int)r->p[rid].
33             size();
34     }
35     Rule *next_term()const{
36         if(completed())return 0;
37         return r->p[rid][dot_id];
38     }
39     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.
37         dot_id;
38     if(r!=b.r)return r<b.r;
39     return rid<b.rid;
40 }
41 void print()const{
42     cout<<RuleName[r]<<"->";
43     if(rid!= -1)for(size_t i=0;++i){
44         if((int)i==dot_id)cout<<" "<<"$";
45         if(i>=r->p[rid].size())break;
46         cout<<" "<<RuleName[r->p[rid][i]];
47     }
48     cout<<" "<<"["<<start<<" "<<end<<""]<<
49         endl;
50 }
51 struct Column{
52     Rule *term;
53     string value;
54     vector<State> s;
55     map<State,set<pair<State,State> > > div;
56     //div比較像一棵左兒右子的樹
57     Column(Rule *r,const string &s):term(r),
58         value(s){}
59     Column(){}
60     bool add(const State &st,int col){
61         if(div.find(st)==div.end()){
62             div[st].push_back(st);
63             s.back().end=col;
64             return true;
65         }else return false;
66     }
67     inline vector<Column> lexer(string text){
68         //tokenize，要自己寫，以下為範例
69         //他會把 input stream 變成 token stream
70         //就是 (terminal,value)pair
71         vector<Column> token;
72         replace(text.begin(),text.end(),',',' ');
73         stringstream ss(text);
74         while(ss>>text){
75             if(text=="a"||text=="of")continue;
76             if(text=="List"){
77                 token.push_back(Column(NameRule["("],
78                     ""));
79             }else if(text=="and"){
80                 token.push_back(Column(NameRule["("],
81                     ""));
82             }else token.push_back(Column(NameRule["T
83                 "],text));
84         }
85         return token;
86     }
87     vector<Column> table;
88     inline void predict(int col,Rule *rul){
89         for(size_t i=0;i<rul->p.size();++i){
90             table[col].add(State(rul,i,0,col),col);
91         }
92     }
93     inline void scan(int col,const State &s,Rule
94         *r){
95         if(r!=table[col].term)return;
96         State ns(s.r,s.rid,s.dot_id+1,s.start);

```

```

92     table[col].add(ns,col);
93     table[col].div[ns].insert(make_pair(s,
94         State(r,col)));
95 }
96 inline void complete(int col,const State &s)
97 {
98     for(size_t i=0;i<table[s.start].s.size()
99         ;++i){
100         State &st=table[s.start].s[i];
101         Rule *term=st.next_term();
102         if(!term||term->p.size()==0)continue;
103         if(term==s.r){
104             State nst(st.r,st.rid,st.dot_id+1,st.
105                 start);
106             table[col].add(nst,col);
107             table[col].div[nst].insert(make_pair(
108                 st,s));
109         }
110     }
111 }
112 inline pair<bool,State> parse(Rule *GAMMA,
113     const vector<Column> &token){
114     table.resize(token.size()+1);
115     for(size_t i=0;i<token.size();++i)table[i
116         +1]=Column(token[i]);
117     table[0]=Column();
118     table[0].add(State(GAMMA,0,0,0),0);
119     for(size_t i=0;i<table.size();++i){
120         for(size_t j=0;j<table[i].s.size();++j){
121             State state=table[i].s[j];
122             if(state.completed())complete(i,state)
123                 ;
124             else{
125                 Rule *term=state.next_term();
126                 if(term->p.size())predict(i,term);
127                 else if(i+1<table.size())scan(i+1,
128                     state,term);
129             }
130         }
131     }
132     for(size_t i=0;i<table.back().s.size();++i)
133     {
134         if(table.back().s[i].r==GAMMA&&table.
135             back().s[i].completed()){
136             return make_pair(true,table.back().s[i]
137                 );
138         }
139     }
140     return make_pair(false,State(0,-1));
141 }
142 struct node{//語法樹的節點
143     State s;
144     vector<vector<node*> > child;//vector<node
145         *>.size()>1表示ambiguous
146     node(const State &s):s(s){}
147     node(){}
148 };
149 struct State_end_cmp{
150     bool operator()(const State &a,const State
151         &b)const{
152         return a.end<b.end||(a.end==b.end&&a<b);
153     }
154 };
155 map<State,node*,State_end_cmp> cache;
156 vector<node*> node_set;

```



## 7 Linear\_Programming

### 7.1 最大密度子圖.cpp

```

143 inline void init_cache(){
144     for(auto d:node_set)delete d;
145     cache.clear();
146     node_set.clear();
147 }
148 void build_tree(const State &s,node *pa,
149     bool amb=0){
150     if(cache.find(s)!=cache.end()){
151         pa->child.push_back(vector<node*>(1,
152             cache[s]));
153         return;
154     }
155     node *o;
156     if(s.completed()){
157         o=new node(s);
158         if(amb)pa->child.back().push_back(o);
159         else pa->child.push_back(vector<node
160             *>(1,o));
161     }else o=pa->child.back().back();
162     amb=0;
163     for(auto div:table[s.end].div[s]){
164         if(!amb)_build_tree(div.first,pa);
165         _build_tree(div.second,o,amb);
166         amb=1;
167     }
168     if(s.completed())cache[s]=o;
169 }
170 inline node *build_tree(const State &s){
171     init_cache();
172     node o;
173     _build_tree(s,&o);
174     assert(o.child.size()==1);
175     assert(o.child.back().size()==1);
176     return o.child.back().back();
177 }
178 void print_tree(node *o,int dep=0){
179     cout<<string(dep,' '),o->s.print();
180     for(auto div:o->child){
181         for(auto nd:div){
182             print_tree(nd,dep+2);
183         }
184     }
185 }
186 //開始寫code:以下為加入語法的範例
187 inline Rule *get_my_Rule(){
188     Rule *S=add_rule("S"),*E=add_rule("E"),*L=
189         add_rule("L");
190     Rule *list=add_rule("("),*AND=add_rule("&")
191         ,*T=add_rule("T");
192     S->add({list,E});
193     S->add({list,L});
194     L->add({E,L});
195     L->add({E,AND,E});
196     E->add({T});
197     E->add({S});
198     Rule *GAMMA=add_rule("GAMMA");//一定要有
199         gamma_rule當作是最上層的語法
200     GAMMA->add({S});
201     return GAMMA;
202 }

```

```

1 typedef double T; //POJ 3155
2 const int MAXN=105;
3 struct edge{
4     int u,v;
5     T w;
6     edge(int u=0,int v=0,T w=0):u(u),v(v),w(w)
7     {}
8 };
9 vector<edge> E;
10 int n,m; // 1-base
11 T de[MAXN],pv[MAXN]; //每個點的邊權和和點權(
12     有些題目會給)
13 void init(){
14     E.clear();
15     for(int i=1;i<=n;++i)de[i]=pv[i]=0;
16 }
17 void add_edge(int u,int v,T w){
18     E.push_back(edge(u,v,w));
19     de[u]+=w,de[v]+=w;
20 }
21 T U; //二分搜的最大值
22 void get_U(){
23     U=0;
24     for(int i=1;i<=n;++i)U+=2*pv[i];
25     for(size_t i=0;i<E.size();++i)U+=E[i].w;
26 }
27 ISAP<T> isap; //網路流
28 int s,t; //原匯點
29 void build(T L){
30     isap.init(n+2);
31     for(size_t i=0;i<E.size();++i){
32         isap.add_edge(E[i].u,E[i].v,E[i].w);
33     }
34     for(int v=1;v<=n;++v){
35         isap.add_edge(s,v,U);
36         isap.add_edge(v,t,U+2*L-de[v]-2*pv[v]);
37     }
38 }
39 int main(){
40     while(~scanf("%d%d",&n,&m)){
41         if(!m){
42             puts("1\n1");
43             continue;
44         }
45         init();
46         int u,v;
47         for(int i=0;i<=m;++i){
48             scanf("%d%d",&u,&v);
49             add_edge(u,v,1);
50         }
51         get_U();
52         s=n+1,t=n+2;
53         T l=0,r=U,k=1.0/(n*n);
54         while(r-l>k){ //二分搜最大值
55             T mid=(l+r)/2;
56             build(mid);
57             T res=(U*n-isap.isap(s,t))/2;
58             if(res>0)l=mid;

```

```

57     else r=mid;
58 }
59 build(1);
60 isap.min_cut(s,t);
61 vector<int> ans;
62 for(int i=1;i<=n;++i){
63     if(isap.vis[i])ans.push_back(i);
64 }
65 printf("%d\n",ans.size());
66 for(size_t i=0;i<ans.size();++i){
67     printf("%d\n",ans[i]);
68 }
69 }
70 return 0;
71 }

```

## 8 Number\_Theory

### 8.1 basic.cpp

```

1 template<typename T>
2 void gcd(const T &a,const T &b,T &d,T &x,T &
3     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 long long int phi[N+1];
8 void phiTable(){
9     for(int i=1;i<=N;++i)phi[i]=i;
10    for(int i=1;i<=N;++i)for(x=i*2;x<=N;x+=i)
11        phi[x]-=phi[i];
12 }
13 void all_divdown(const LL &n){ // all n/x
14     for(LL a=1;a<=n;a=n/(n/(a+1)))
15         // dosomething;
16 }
17 const int MAXPRIME = 1000000;
18 int iscom[MAXPRIME], prime[MAXPRIME],
19     primecnt;
20 int phi[MAXPRIME], mu[MAXPRIME];
21 void sieve(void){
22     memset(iscom,0,sizeof(iscom));
23     primecnt = 0;
24     phi[1] = mu[1] = 1;
25     for(int i=2;i<MAXPRIME;++i){
26         if(!iscom[i]){
27             prime[primecnt++] = i;
28             mu[i] = -1;
29             phi[i] = i-1;
30         }
31         for(int j=0;j<primecnt;++j){
32             int k = i * prime[j];
33             if(k>=MAXPRIME) break;
34             iscom[k] = prime[j];
35             if(i%prime[j]==0){
36                 mu[k] = 0;
37                 phi[k] = phi[i] * prime[j];
38                 break;
39             } else {
40                 mu[k] = -mu[i];
41                 phi[k] = phi[i] * (prime[j]-1);
42             }
43         }
44     }
45 }

```

```

39     phi[k] = phi[i] * (prime[j]-1);
40 }
41 }
42 }
43 }
44 }
45 bool g_test(const LL &g, const LL &p, const
46     vector<LL> &v) {
47     for(int i=0;i<v.size();++i)
48         if(modexp(g,(p-1)/v[i],p)==1)
49             return false;
50     return true;
51 }
52 LL primitive_root(const LL &p) {
53     if(p==2) return 1;
54     vector<LL> v;
55     Factor(p-1,v);
56     v.erase(unique(v.begin(), v.end()), v.end()
57         ());
58     for(LL g=2;g<p;++g)
59         if(g_test(g,p,v))
60             return g;
61     puts("primitive_root NOT FOUND");
62     return -1;
63 }
64 int Legendre(const LL &a, const LL &p) {
65     return modexp(a%p,(p-1)/2,p);
66 }
67 LL inv(const LL &a, const LL &n) {
68     LL d,x,y;
69     gcd(a,n,d,x,y);
70     return d==1 ? (x+n)%n : -1;
71 }
72 int inv[maxn];
73 LL invtable(int n,LL P){
74     inv[1]=1;
75     for(int i=2;i<=n;++i)
76         inv[i]=(P-(P/i))*inv[P%i]%P;
77 }
78 LL log_mod(const LL &a, const LL &b, const
79     LL &p) {
80     // a ^ x = b ( mod p )
81     int m=sqrt(p+.5), e=1;
82     LL v=inv(modexp(a,m,p), p);
83     map<LL,int> x;
84     x[1]=0;
85     for(int i=1;i<=m;++i){
86         e = LLMul(e,a,p);
87         if(!x.count(e)) x[e] = i;
88     }
89     for(int i=0;i<=m;++i){
90         if(x.count(b)) return i*m + x[b];
91         b = LLMul(b,v,p);
92     }
93     return -1;
94 }
95 LL Tonelli_Shanks(const LL &n, const LL &p)
96 {
97     // x^2 = n ( mod p )
98     if(n==0) return 0;
99     if(Legendre(n,p)!=1) while(1) { puts("SQRT
100         ROOT does not exist"); }
101     int S = 0;

```

```

99 LL Q = p-1;
100 while( !(Q&1) ) { Q>>=1; ++S; }
101 if(S==1) return modexp(n%p,(p+1)/4,p);
102 LL z = 2;
103 for(; Legendre(z,p)!=-1; ++z)
104 LL c = modexp(z,Q,p);
105 LL R = modexp(n%p,(Q+1)/2,p), t = modexp(n
    %p,Q,p);
106 int M = S;
107 while(1) {
108     if(t==1) return R;
109     LL b = modexp(c,1L<<(M-i-1),p);
110     R = Llmul(R,b,p);
111     t = Llmul( Llmul(b,b,p), t, p);
112     c = Llmul(b,b,p);
113     M = i;
114 }
115 return -1;
116 }
117
118 template<typename T>
119 T Euler(T n){
120     T ans=n;
121     for(T i=2;i*i<=n;++i){
122         if(n%i==0){
123             ans=ans/i*(i-1);
124             while(n%i==0)n/=i;
125         }
126     }
127     if(n>1)ans=ans/n*(n-1);
128     return ans;
129 }
130
131 //Chinese_remainder_theorem
132 template<typename T>
133 T pow_mod(T n,T k,T m){
134     T ans=1;
135     for(n=(n>=m?n%m:n);k;k>>=1){
136         if(k&1)ans=ans*n%m;
137         n=n*n%m;
138     }
139     return ans;
140 }
141
142 template<typename T>
143 T crt(vector<T> &m,vector<T> &a){
144     T M=1,tM,ans=0;
145     for(int i=0;i<(int)m.size();++i)M*=m[i];
146     for(int i=0;i<(int)a.size();++i){
147         tM=M/m[i];
148         ans=(ans+(a[i]*tM%M)*pow_mod(tM,Euler(m[
            i])-1,m[i])%M)%M;
149         /*如果m[i]是質數·Euler(m[i])-1=m[i]-2·
            就不用算Euler了*/
150     }
151     return ans;
152 }
153
154 //java code
155 //求sqrt(N)的連分數
156 public static void Pell(int n){
157     BigInteger N,p1,p2,q1,q2,a0,a1,a2,g1,g2,h1
        ,h2,p,q;
158     g1=q2=p1=BigInteger.ZERO;
159     h1=q1=p2=BigInteger.ONE;

```

```

159     a0=a1=BigInteger.valueOf((int)Math.sqrt
        (1.0*n));
160     BigInteger ans=a0.multiply(a0);
161     if(ans.equals(BigInteger.valueOf(n))){
162         System.out.println("No solution!");
163         return ;
164     }
165     while(true){
166         g2=a1.multiply(h1).subtract(g1);
167         h2=N.subtract(g2.pow(2)).divide(h1);
168         a2=g2.add(a0).divide(h2);
169         p=a1.multiply(p2).add(p1);
170         q=a1.multiply(q2).add(q1);
171         if(p.pow(2).subtract(N.multiply(q.pow
            (2))).compareTo(BigInteger.ONE)==0)
            break;
172         g1=g2;h1=h2;a1=a2;
173         p1=p2;p2=p;
174         q1=q2;q2=q;
175     }
176     System.out.println(p+" "+q);
177 }

```

## 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的子集的處理
12        int x=comb&-comb,y=comb+x;
13        comb=((comb&~y)/x>>1)|y;
14    }
15 }

```

## 8.3 cantor\_expansion.cpp

```

1 int factorial[MAXN];
2 void init(){
3     factorial[0]=1;
4     for(int i=1;i<=MAXN;++i)factorial[i]=
        factorial[i-1]*i;
5 }
6 int encode(const vector<int> &s){
7     int n=s.size(),res=0;
8     for(int i=0;i<n;++i){
9         int t=0;
10        for(int j=i+1;j<n;++j)
11            if(s[j]<s[i])++t;
12        res+=t*factorial[n-i-1];
13    }
14    return res;
15 }

```

```

16 vector<int> decode(int a,int n){
17     vector<int> res;
18     vector<bool> vis(n,0);
19     for(int i=n-1;i>=0;--i){
20         int t=a/factorial[i],j;
21         for(j=0;j<n;++j)
22             if(!vis[j]){
23                 if(t==0)break;
24                 --t;
25             }
26         res.push_back(j);
27         vis[j]=1;
28         a%=factorial[i];
29     }
30     return res;
31 }

```

## 8.4 FFT.cpp

```

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

```

## 8.5 find\_real\_root.cpp

```

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

```

## 8.6 LinearCongruence.cpp

```

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

```

## 8.7 Lucas.cpp

```

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

```

## 8.8 Matrix.cpp

```

1 template<typename T>
2 struct Matrix{
3   using rt = std::vector<T>;
4   using mt = std::vector<rt>;
5   using matrix = Matrix<T>;
6   int r,c;
7   mt m;
8   Matrix(int r,int c):r(r),c(c),m(r,rt(c)){}
9   rt& operator[](int i){return m[i];}
10  matrix operator+(const matrix &a){
11    matrix rev(r,c);
12    for(int i=0;i<r;++i)

```

```

13    for(int j=0;j<c;++j)
14      rev[i][j]=m[i][j]+a.m[i][j];
15    return rev;
16  }
17  matrix operator-(const matrix &a){
18    matrix rev(r,c);
19    for(int i=0;i<r;++i)
20      for(int j=0;j<c;++j)
21        rev[i][j]=m[i][j]-a.m[i][j];
22    return rev;
23  }
24  matrix operator*(const matrix &a){
25    matrix rev(r,a.c);
26    matrix tmp(a.c,a.r);
27    for(int i=0;i<a.r;++i)
28      for(int j=0;j<a.c;++j)
29        tmp[j][i]=a.m[i][j];
30    for(int i=0;i<r;++i)
31      for(int j=0;j<a.c;++j)
32        for(int k=0;k<c;++k)
33          rev.m[i][j]=m[i][k]*tmp[j][k];
34    return rev;
35  }
36  bool inverse(){
37    Matrix t(r,r+c);
38    for(int y=0;y<r;y++){
39      t.m[y][c+y] = 1;
40      for(int x=0;x<c;++x)
41        t.m[y][x]=m[y][x];
42    }
43    if(!t.gas())
44      return false;
45    for(int y=0;y<r;y++){
46      for(int x=0;x<c;++x)
47        m[y][x]=t.m[y][c+x]/t.m[y][y];
48    }
49    return true;
50  }
51  T gas(){
52    vector<T> lazy(r,1);
53    bool sign=false;
54    for(int i=0;i<r;++i){
55      if(m[i][i]==0){
56        int j=i+1;
57        while(j<r&&!m[j][i])j++;
58        if(j==r) continue;
59        m[i].swap(m[j]);
60        sign=!sign;
61      }
62      for(int j=0;j<r;++j){
63        if(i==j) continue;
64        lazy[j]=lazy[j]*m[i][i];
65        T mx=m[j][i];
66        for(int k=0;k<c;++k)
67          m[j][k]=m[j][k]*m[i][i]-mx*m[i][k];
68      }
69    }
70    T det=sign?-1:1;
71    for(int i=0;i<r;++i){
72      det = det*m[i][i];
73      for(auto &j:m[i])j/=lazy[i];
74    }
75    return det;
76  }
77 }

```

## 8.9 MillerRobin.cpp

```

1 LL LLmul(LL a, LL b, const LL &mod) {
2   LL ans=0;
3   while(b) {
4     if(b&1) {
5       ans+=a;
6       if(ans>=mod) ans-=mod;
7     }
8     a<<=1, b>>=1;
9     if(a>=mod) a-=mod;
10  }
11  return ans;
12 }
13 long long mod_mul(long long a,long long b,
14   long long m){
15   a%=m,b%=m;
16   long long y=(long long)((double)a*b/m+0.5);
17   /* fast for m < 2^58 */
18   long long r=(a*b-y*m)%m;
19   return r<0?r+m:r;
20 }
21 template<typename T>
22 T pow(T a,T b,T mod){//a^b%mod
23   T ans=1;
24   for(;b;a=mod_mul(a,a,mod),b>>=1)
25     if(b&1) ans=mod_mul(ans,a,mod);
26   return ans;
27 }
28 int sprp[3]={2,7,61}; //int%d^3%i,0
29 int llsprp
30 [7]={2,325,9375,28178,450775,9780504,179526};
31 //!00unsigned long long%d^30
32 template<typename T>
33 bool isprime(T n,int *sprp,int num){
34   if(n==2) return 1;
35   if(n<2||n%2==0) return 0;
36   int t=n-1;
37   T u=n-1;
38   for(;u%2==0;++t)u>>=1;
39   for(int i=0;i<num;++i){
40     T a=sprp[i]%n;
41     if(a==0||a==1||a==n-1) continue;
42     T x=pow(a,u,n);
43     if(x==1||x==n-1) continue;
44     for(int j=0;j<t;++j){
45       x=mod_mul(x,x,n);
46       if(x==1) return 0;
47       if(x==n-1) break;
48     }
49     if(x==n-1) continue;
50     return 0;
51   }
52   return 1;
53 }

```

## 8.10 NTT.cpp

```

1 2615053605667*(2^18)+1,3
2 15*(2^27)+1,31
3 479*(2^21)+1,3
4 7*17*(2^23)+1,3

```

```

5 3*3*211*(2^19)+1,5
6 25*(2^22)+1,3
7 template<typename T,typename VT=std::vector<
8   T>>
9 struct NTT{
10  const T P,G;
11  NTT(T p=(1<<23)*7*17+1,T g=3):P(p),G(g){}
12  unsigned int bit_reverse(unsigned int a,
13    int len){
14    a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)
15      >>1);
16    a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)
17      >>2);
18    a=((a&0x0F0F0F0FU)<<4)|((a&0xFF0FF0FFU)
19      >>4);
20    a=((a&0x00FF00FFU)<<8)|((a&0xFFFF0000U)
21      >>8);
22    a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)
23      >>16);
24    return a>>(32-len);
25  }
26  T pow_mod(T n,T k,T m){
27    T ans=1;
28    for(n=(n>=m?n%m:n);k>>=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  {
36    int bitlen=std::lg(N);
37    for(int i=0;i<N;++i) out[bit_reverse(i,
38      bitlen)]=in[i];
39    for(int step=2,id=1;step<=N;step<<=1,++
40      id){
41      T wn=pow_mod(G,(P-1)>>id,P),wi=1,u,t;
42      const int mh=step>>1;
43      for(int i=0;i<N;i+=step){
44        for(int j=i;j<N;j+=step){
45          u=out[j];t=wi*out[j+mh]%P;
46          out[j]=u+t;
47          out[j+mh]=u-t;
48          if(out[j]>=P) out[j]-=P;
49          if(out[j+mh]<0) out[j+mh]+=P;
50        }
51        wi=wi*wn%P;
52      }
53    }
54    if(is_inv){
55      for(int i=1;i<N/2;++i) std::swap(out[i],
56        out[N-i]);
57      T invn=pow_mod(N,P-2,P);
58      for(int i=0;i<N;++i) out[i]=out[i]*invn
59        %P;
60    }
61  }
62 }

```

## 8.11 Simpson.cpp

```

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

```

## 8.12 WhatDay.cpp

```

1 int whatday(int y, int m, int d){
2     if(m<=2)m+=12, --y;
3     if(y<1752||y==1752&&m<9||y==1752&&m==9&&
4         d<3)
5         return (d+2*m+3*(m+1)/5+y+y/4+y/5)%7;
6     return (d+2*m+3*(m+1)/5+y+y/4-y/10+y/400)%7;
7 }

```

## 8.13 外星模運算.cpp

```

1 //a[0]^(a[1]^a[2]^...)
2 #include<bits/stdc++.h>
3 using namespace std;
4 #define maxn 1000000
5 int euler[maxn+5];
6 bool is_prime[maxn+5];
7 inline void init_euler(){
8     is_prime[1]=1; //不是質數
9     for(int i=1; i<=maxn; i++) euler[i]=i;
10    for(int i=2; i<=maxn; i++){
11        if(!is_prime[i]){ //是質數
12            euler[i]--;
13            for(int j=i<1; j<=maxn; j+=i){
14                is_prime[j]=1;
15                euler[j]=euler[j]/i*(i-1);
16            }
17        }
18    }
19 }
20 inline long long pow(long long a, long long b,
21     long long mod){ //a^b%mod
22     long long ans=1;
23     for(; b; a=a*a%mod, b>>=1)
24         if(b&1) ans=ans*a%mod;
25     return ans;
26 }

```

```

26 bool isless(long long *a, int n, int k){
27     if(*a==1) return k>1;
28     if(--n==0) return *a<k;
29     int next=0;
30     for(long long b=1; b<k; ++next)
31         b*=*a;
32     return isless(a+1, n, next);
33 }
34 long long high_pow(long long *a, int n, long
35     long mod){
36     if(*a==1||--n==0) return *a%mod;
37     int k=0, r=euler[mod];
38     for(long long tma=1; tma!=pow(*a, k+r, mod);
39         ++k)
40         tma=tma*(*a)%mod;
41     if(isless(a+1, n, k)) return pow(*a, high_pow(
42         a+1, n, k, mod));
43     int tmd=high_pow(a+1, n, r);
44     int t=(tmd-k+r)%r;
45     return pow(*a, k+t, mod);
46 }
47 long long a[1000005];
48 int t, mod;
49 int main(){
50     init_euler();
51     scanf("%d", &t);
52     #define n 4
53     while(t--){
54         for(int i=0; i<n; ++i) scanf("%lld", &a[i]);
55         scanf("%d", &mod);
56         printf("%lld\n", high_pow(a, n, mod));
57     }
58     return 0;
59 }

```

## 8.14 模運算模板.cpp

```

1 template<typename T, long long mod>
2 struct mod_t{ //mod 只能是質數
3     T data;
4     mod_t(){ }
5     mod_t(const T &d): data((d%mod+mod)%mod){ }
6     mod_t pow(T b) const{
7         mod_t ans(1);
8         for(mod_t now=*this; b; now=now*now, b/2)
9             if(b%2) ans=ans*now;
10        return ans;
11    }
12    mod_t operator-(int) const{
13        return mod_t(mod-data);
14    }
15    mod_t operator+(const mod_t &b) const{
16        return mod_t((data+b.data)%mod);
17    }
18    mod_t operator-(const mod_t &b) const{
19        return mod_t((data-b.data+mod)%mod);
20    }
21    mod_t operator*(const mod_t &b) const{
22        return mod_t((data*b.data)%mod);
23    }
24    mod_t operator/(const mod_t &b) const{
25        return *this*b.pow(mod-2); // *this *
26        Inverse(b)
27    }
28 }

```

```

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

```

## 8.15 質因數分解.cpp

```

1 LL func(const LL n, const LL mod, const int c)
2 {
3     return (LLmul(n, n, mod)+c+mod)%mod;
4 }
5 LL pollorro(const LL n, const int c) { // 循
6     環 節 長 度
7     LL a=1, b=1;
8     a=func(a, n, c)%n;
9     b=func(b, n, c)%n;
10    while(gcd(abs(a-b), n)==1) {
11        a=func(a, n, c)%n;
12        b=func(b, n, c)%n;
13    }
14    return gcd(abs(a-b), n);
15 }
16 void prefactor(LL &n, vector<LL> &v) {
17     for(int i=0; i<12; ++i) {
18         while(n%prime[i]==0) {
19             v.push_back(prime[i]);
20             n/=prime[i];
21         }
22     }
23 }
24 void smallfactor(LL n, vector<LL> &v) {
25     if(n<MAXPRIME) {
26         while(isp[(int)n]) {
27             v.push_back(isp[(int)n]);
28             n/=isp[(int)n];
29         }
30         v.push_back(n);
31     }
32     else {
33         for(int i=0; i<primecnt&&prime[i]*prime[i]
34             ]<=n; ++i) {
35             while(n%prime[i]==0) {
36                 v.push_back(prime[i]);
37                 n/=prime[i];
38             }
39             if(n!=1) v.push_back(n);
40         }
41     }
42 }
43 void comfactor(const LL &n, vector<LL> &v) {
44     if(n<1e9) {
45         smallfactor(n, v);
46         return;
47     }
48     if(Isprime(n)) {
49         v.push_back(n);
50         return;
51     }
52     LL d;
53     for(int c=3; ++c) {
54         d = pollorro(n, c);
55         if(d!=n) break;
56     }
57     comfactor(d, v);
58     comfactor(n/d, v);
59 }
60 void Factor(const LL &x, vector<LL> &v) {
61     LL n = x;
62     if(n==1) { puts("Factor 1"); return; }
63     prefactor(n, v);
64     if(n==1) return;
65     comfactor(n, v);
66     sort(v.begin(), v.end());
67 }
68 void AllFactor(const LL &n, vector<LL> &v) {
69     vector<LL> tmp;
70     Factor(n, tmp);
71     v.clear();
72     v.push_back(1);
73     int len;
74     LL now=1;
75     for(int i=0; i<tmp.size(); ++i) {
76         if(i==0 || tmp[i]!=tmp[i-1]) {
77             len = v.size();
78             now = 1;
79         }
80         now*=tmp[i];
81         for(int j=0; j<len; ++j)
82             v.push_back(v[j]*now);
83     }
84 }

```

## 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         joe(): ed(0), cnt_dp(0), vis(0){
7             for(int i=0; i<=R-L; ++i) next[i]=0;
8         }
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     /*DP出每個前綴在字串s出現的次數並傳回所
54     有字串被s匹配成功的次數O(N*M)*/
55     int match_0(const char *s){
56         int ans=0,id,p=0,i;
57         for(i=0;s[i];++i){
58             id=s[i]-L;
59             while(!S[p].next[id]&&p=S[p].fail;
60                 if(!S[p].next[id])continue;
61                 p=S[p].next[id];
62             ++S[p].cnt_dp; /*匹配成功則它所有後綴
63             都可以被匹配(DP計算)*/
64         }
65         for(i=qe-1;i>=0;--i){
66             ans+=S[q[i]].cnt_dp*S[q[i]].ed;
67             if(~S[q[i]].fail)S[q[i]].fail.
68                 cnt_dp+=S[q[i]].cnt_dp;
69         }
70         return ans;
71     }
72     /*多串匹配走efl邊並傳回所有字串被s匹配成
73     功的次數O(N*M^1.5)*/
74     int match_1(const char *s)const{
75         int ans=0,id,p=0,t;
76         for(int i=0;s[i];++i){
77             id=s[i]-L;
78             while(!S[p].next[id]&&p=S[p].fail;
79                 if(!S[p].next[id])continue;
80                 p=S[p].next[id];
81             if(S[p].ed)ans+=S[p].ed;
82             for(t=S[p].efl;~t;t=S[t].efl){
83                 ans+=S[t].ed; /*因為都走efl邊所以保
84                 證匹配成功*/
85             }
86         }
87         return ans;
88     }
89     /*枚舉(s的子字串nA)的所有相異字串各恰一
90     次並傳回次數O(N*M^(1/3))*/
91     int match_2(const char *s){
92         int ans=0,id,p=0,t;
93         ++vt;
94         /*把戳記vt+=1, 只要vt沒溢位, 所有S[p].
95         vis==vt就會變成false
96         這種利用vt的方法可以O(1)歸零vis陣列*/
97         for(int i=0;s[i];++i){
98             id=s[i]-L;
99             while(!S[p].next[id]&&p=S[p].fail;
100                 if(!S[p].next[id])continue;
101                 p=S[p].next[id];
102             if(S[p].ed&&S[p].vis!=vt){
103                 S[p].vis=vt;
104                 ans+=S[p].ed;
105             }
106             for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t]
107                 .efl){
108                 S[t].vis=vt;
109                 ans+=S[t].ed; /*因為都走efl邊所以保
110                 證匹配成功*/
111             }
112         }
113     }
114 }

```

## 9.2 hash.cpp

```

1 #define MAXN 1000000
2 #define prime_mod 1073676287
3 /*prime_mod 必須要是質數*/
4 typedef long long T;
5 char s[MAXN+5];
6 T h[MAXN+5]; /*hash陣列*/
7 T h_base[MAXN+5]; /*h_base[n]=(prime^n)%
8     prime_mod*/
9 inline void hash_init(int len,T prime=0
10     xdefaced){
11     h_base[0]=1;
12     for(int i=1;i<len;++i){
13         h[i]=(h[i-1]*prime+s[i-1])%prime_mod;

```

```

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
16     法, 設編號為0 ~ len-1*/
17     return (h[r+1]-(h[l]*h_base[r-l+1])%
18         prime_mod+prime_mod)%prime_mod;

```

## 9.3 KMP.cpp

```

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

```

## 9.4 manacher.cpp

```

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

```

## 9.5 minimal\_string\_rotation.cpp

```

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

```

## 9.6 suffix\_array\_lcp.cpp

```

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

```

## 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[
12             i].clear();
13     }
14     void add_edge(int u, int v){
15         suc[u].push_back(v);
16         pre[v].push_back(u);
17     }
18     void dfs(int u){
19         dfn[u]=++Time, id[Time]=u;
20         for(auto v:suc[u]){
21             if(dfn[v])continue;
22             dfs(v), fa[dfn[v]]=dfn[u];
23         }
24     }
25     int find(int x){
26         if(x==anc[x])return x;
27         int y=find(anc[x]);
28         if(semi[best[x]]>semi[best[anc[x]]])best
29             [x]=best[anc[x]];
30         return anc[x]=y;
31     }
32     void tarjan(int r){
33         Time=0;
34         for(int t=1; t<=n; ++t){
35             dfn[t]=idom[t]=0; // u=r 或是 u 無法到達 r 時
36             idom[id[u]]=0

```

```

34     dom[t].clear();
35     anc[t]=best[t]=semi[t]=t;
36 }
37 dfs(r);
38 for(int y=Time; y>=2; --y){
39     int x=fa[y], idy=id[y];
40     for(auto z:pre[idy]){
41         if(! (z=dfn[z]))continue;
42         find(z);
43         semi[y]=min(semi[y], semi[best[z]]);
44     }
45     dom[semi[y]].push_back(y);
46     anc[y]=x;
47     for(auto z:dom[x]){
48         find(z);
49         idom[z]=semi[best[z]<x?best[z]:x];
50     }
51     dom[x].clear();
52 }
53 for(int u=2; u<=Time; ++u){
54     if(idom[u]!=semi[u])idom[u]=idom[idom[
55         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*N)
6 vector<int> v[MAXN2];
7 vector<int> rv[MAXN2];
8 vector<int> vis_t;
9 int N, M;
10 void addedge(int s, int e){
11     v[s].push_back(e);
12     rv[e].push_back(s);
13 }
14 int scc[MAXN2];
15 bool vis[MAXN2]={false};
16 void dfs(vector<int> *uv, int n, int k=-1){
17     vis[n]=true;
18     for(int i=0; i<uv[n].size(); ++i)
19         if(!vis[uv[n][i]])
20             dfs(uv, uv[n][i], k);
21     if(uv==v)vis_t.push_back(n);
22     scc[n]=k;
23 }
24 void solve(){
25     for(int i=1; i<=N; ++i){
26         if(!vis[i])dfs(v, i);
27         if(!vis[n(i)])dfs(v, n(i));
28     }
29     memset(vis, 0, sizeof(vis));
30     int c=0;
31     for(int i=vis_t.size()-1; i>=0; --i)
32         if(!vis[vis_t[i]])
33             dfs(rv, vis_t[i], c++);
34 }

```

```

35 int main(){
36     int a, b;
37     scanf("%d%d", &N, &M);
38     for(int i=1; i<=N; ++i){
39         // (A or B) & (!A & !B) A^B
40         a=i*2-1;
41         b=i*2;
42         addedge(n(a), b);
43         addedge(n(b), a);
44         addedge(a, n(b));
45         addedge(b, n(a));
46     }
47     while(M--){
48         scanf("%d%d", &a, &b);
49         a = a>0?a*2-1:-a*2;
50         b = b>0?b*2-1:-b*2;
51         // A or B
52         addedge(n(a), b);
53         addedge(n(b), a);
54     }
55     solve();
56     bool check=true;
57     for(int i=1; i<=2*N; ++i)
58         if(scc[i]==scc[n(i)])
59             check=false;
60     if(check){
61         printf("%d\n", N);
62         for(int i=1; i<=2*N; i+=2){
63             if(scc[i]>scc[i+2*N])
64                 putchar('+');
65             else
66                 putchar('-');
67         }
68         putchar('\n');
69     }else puts("0");
70     return 0;
71 }

```

### 10.3 橋連通分量.cpp

```

1 #define N 1005
2 struct edge{
3     int u, v;
4     bool is_bridge;
5     edge(int u=0, int v=0):u(u), v(v), is_bridge
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         if(!vis[v=G[u][i]]){
26             dfs(v, u), ++child;
27             low[u]=min(low[u], low[v]);
28             if(vis[u]<=low[v]){
29                 is_cut[u]=1;
30                 bcc[++bcc_cnt].clear();
31                 int t;
32                 do{
33                     bcc_id[t=st[--top]]=bcc_cnt;
34                     bcc[bcc_cnt].push_back(t);
35                 }while(t!=v);
36                 bcc_id[u]=bcc_cnt;
37                 bcc[bcc_cnt].push_back(u);
38             }
39             else if(vis[v]<vis[u]&&v!=pa) // 反向邊
40                 low[u]=min(low[u], vis[v]);
41         }
42     }
43 }

```

```

21 st[top++]=u;
22 for(size_t i=0; i<G[u].size(); ++i){
23     int e=G[u][i]; v=E[e].v;
24     if(!vis[v]){
25         dfs(v, e^1); // e^1 反向邊
26         low[u]=min(low[u], low[v]);
27         if(vis[u]<low[v]){
28             E[e].is_bridge=E[e^1].is_bridge=1;
29             ++bridge_cnt;
30         }
31     }else if(vis[v]<vis[u]&&v!=re)
32         low[u]=min(low[u], vis[v]);
33 }
34 if(vis[u]==low[u]) // 處理 BCC
35     ++bcc_cnt; // 1-base
36     do bcc_id[v=st[--top]]=bcc_cnt; // 每個點
37         所在的 BCC
38     while(v!=u);
39 }
40 inline void bcc_init(int n){
41     Time=bcc_cnt=bridge_cnt=top=0;
42     E.clear();
43     for(int i=1; i<=n; ++i){
44         G[i].clear();
45         vis[i]=bcc_id[i]=0;
46     }
47 }

```

### 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     }
31 }

```

```

30 if(pa== -1&&child<2)is_cut[u]=0; //u是dfs樹的根要特判
31 }
32 inline void bcc_init(int n){
33     Time=bcc_cnt=top=0;
34     for(int i=1;i<=n;++i){
35         G[i].clear();
36         is_cut[i]=vis[i]=bcc_id[i]=0;
37     }
38 }

```

## 11 Tree\_problem

### 11.1 HeavyLight.cpp

```

1 #include<vector>
2 #define MAXN 100005
3 typedef std::vector<int>::iterator VIT;
4 int siz[MAXN],max_son[MAXN],pa[MAXN],dep[
    MAXN];
5 int link_top[MAXN],link[MAXN],cnt;
6 std::vector<int> >G[MAXN];
7 void find_max_son(int x){
8     siz[x]=1;
9     max_son[x]=-1;
10    for(VIT i=G[x].begin();i!=G[x].end();++i){
11        if(*i==pa[x])continue;
12        pa[*i]=x;
13        dep[*i]=dep[x]+1;
14        find_max_son(*i);
15        if(max_son[x]==-1||siz[*i]>siz[max_son[x]
            ]){max_son[x]=*i;
16            siz[x]+=siz[*i];
17        }
18    }
19    void build_link(int x,int top){
20        link[x]=++cnt;
21        link_top[x]=top;
22        if(max_son[x]==-1)return;
23        build_link(max_son[x],top);
24        for(VIT i=G[x].begin();i!=G[x].end();++i){
25            if(*i==max_son[x]||*i==pa[x])continue;
26            build_link(*i,*i);
27        }
28    }
29    inline int find_lca(int a,int b){
30        //求LCA · 可以在過程中對區間進行處理
31        int ta=link_top[a],tb=link_top[b];
32        while(ta!=tb){
33            if(dep[ta]<dep[tb]){
34                std::swap(ta,tb);
35                std::swap(a,b);
36            }
37            //這裡可以對a所在的鏈做區間處理
38            //區間為(Link[ta],Link[a])
39            ta=link_top[a=pa[ta]];
40        }
41        //最後a,b會在同一條鏈 · 若a!=b還要在進行一
        次區間處理
42        return dep[a]<dep[b]?a:b;

```

### 11.2 LCA.cpp

```

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

```

### 11.3 link\_cut\_tree.cpp

```

1 #include<vector>
2 struct splay_tree{
3     int ch[2],pa; //子節點跟父母
4     bool rev; //反轉的懶惰標記
5     splay_tree():pa(0),rev(0){ch[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
    tree的根
11    return node[node[x].pa].ch[0]!=x&&node[
        node[x].pa].ch[1]!=x;
12 }
13 void down(int x){ //懶惰標記下推
14    if(node[x].rev){
15        if(node[x].ch[0])node[node[x].ch[0]].rev
            ^=1;
16        if(node[x].ch[1])node[node[x].ch[1]].rev
            ^=1;

```

```

17    std::swap(node[x].ch[0],node[x].ch[1]);
18    node[x].rev^=1;
19 }
20 }
21 void push_down(int x){ //將所有祖先的懶惰標記
    下推
22    if(!isroot(x))push_down(node[x].pa);
23    down(x);
24 }
25 void up(int x){ //將子節點的資訊向上更新
26 void rotate(int x){ //旋轉 · 會自行判斷轉的方
    向
27    int y=node[x].pa,z=node[y].pa,d=(node[y].
        ch[1]==x);
28    node[x].pa=z;
29    if(!isroot(y))node[z].ch[node[z].ch[1]==y
        ]=x;
30    node[y].ch[d]=node[x].ch[d^1];
31    node[node[y].ch[d]].pa=y;
32    node[y].pa=x,node[x].ch[d^1]=y;
33    up(y);
34    up(x);
35 }
36 void splay(int x){ //將節點x伸展到所在splay
    tree的根
37    push_down(x);
38    while(!isroot(x)){
39        int y=node[x].pa;
40        if(!isroot(y)){
41            int z=node[y].pa;
42            if((node[z].ch[0]==y)^(node[y].ch[0]==
                x))rotate(y);
43            else rotate(x);
44        }
45        rotate(x);
46    }
47 }
48 int access(int x){
49    int last=0;
50    while(x){
51        splay(x);
52        node[x].ch[1]=last;
53        up(x);
54        last=x;
55        x=node[x].pa;
56    }
57    return last; //回傳access後splay tree的根
58 }
59 void access(int x,bool is=0){ //is=0就是一般
    的access
60    int last=0;
61    while(x){
62        splay(x);
63        if(is&&!node[x].pa){
64            //printf("%d\n",max(node[last].ma,node
                [node[x].ch[1]].ma));
65        }
66        node[x].ch[1]=last;
67        up(x);
68        last=x;
69        x=node[x].pa;
70    }
71 }
72 void query_edge(int u,int v){

```

```

73    access(u);
74    access(v,1);
75 }
76 void make_root(int x){
77    access(x),splay(x);
78    node[x].rev^=1;
79 }
80 void make_root(int x){
81    node[access(x)].rev^=1;
82    splay(x);
83 }
84 void cut(int x,int y){
85    make_root(x);
86    access(y);
87    splay(y);
88    node[y].ch[0]=0;
89    node[x].pa=0;
90 }
91 void cut_parents(int x){
92    access(x);
93    splay(x);
94    node[node[x].ch[0]].pa=0;
95    node[x].ch[0]=0;
96 }
97 void link(int x,int y){
98    make_root(x);
99    node[x].pa=y;
100 }
101 int find_root(int x){
102    x=access(x);
103    while(node[x].ch[0])x=node[x].ch[0];
104    splay(x);
105    return x;
106 }
107 int query(int u,int v){
108    //傳回uv路徑splay tree的根結點
109    //這種寫法無法求LCA
110    make_root(u);
111    return access(v);
112 }
113 int query_lca(int u,int v){
114    //假設求鏈上點權的總和 · sum是子樹的權重和 ·
        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].
            ch[1]].sum
120    }else{
121        //return node[lca].data+node[node[lca].
            ch[1]].sum+node[u].sum
122    }
123 }
124 struct EDGE{
125     int a,b,w;
126 }e[10005];
127 int n;
128 vector<pair<int,int>> >G[10005];
129 //first表示子節點 · second表示邊的編號
130 int pa[10005],edge_node[10005];
131 //pa是父母節點 · 暫存用的 · edge_node是每個編
        被存在哪個點裡面的陣列
132 void bfs(int root){

```

```

133 //在建構的時候把每個點都設成一個splay tree · 不會壞掉
134 queue<int> q;
135 for(int i=1;i<=n;++i)pa[i]=0;
136 q.push(root);
137 while(q.size()){
138     int u=q.front();
139     q.pop();
140     for(int i=0;i<(int)G[u].size();++i){
141         int v=G[u][i].first;
142         if(v!=pa[u]){
143             pa[v]=u;
144             node[v].pa=u;
145             node[v].data=e[G[u][i].second].w;
146             edge_node[G[u][i].second]=v;
147             up(v);
148             q.push(v);
149         }
150     }
151 }
152 }
153 void change(int x,int b){
154     splay(x);
155     //node[x].data=b;
156     up(x);
157 }

```

```

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

```

## 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,
    int d){
15     dis.push_back(d);
16     for(size_t i=0;i<g[u].size();++i){
17         int v=g[u][i].first,w=g[u][i].second;
18         if(v!=pa&&vis[v])get_dis(dis,v,u,d+w);
19     }
20 }
21 vector<int> dis;//這東西如果放在函數裡會TLE
22 int cal(int u,int d){
23     dis.clear();
24     get_dis(dis,u,-1,d);
25     sort(dis.begin(),dis.end());
26     int l=0,r=dis.size()-1,res=0;
27     while(l<r){
28         while(l<r&&dis[l]+dis[r]>k)--r;
29         res+=r-l++;
30     }
31     return res;
32 }

```

```

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

```

## 12 zformula

### 12.1 formula.tex

#### 12.1.1 Pick 公式

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

#### 12.1.2 圖論

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

$$(b) \frac{I(G)+C_v(G)}{M(G)+C_e(G)} = \frac{|V|}{|V|}$$

- 對於連通二分圖：

$$(a) \frac{I(G)}{M(G)} = \frac{C_v(G)}{C_e(G)}$$

- 最大權閉合圖：

$$(a) \frac{C(u,v)}{C(v,T)} = \frac{\infty, (u,v) \in E}{\frac{1}{U+2g-d_v}, u,v \in V}$$

- 最大密度子圖：

$$(a) \frac{C(u,v)}{C(v,T)} = \frac{1}{U+2g-d_v}, (u,v) \in E$$

- 弦圖：

- 完美消除序列從後往前依次給每個點染色 · 給每個點染上可以染的最小顏色
- 最大獨立集 = 色數
- 最大獨立集 = 完美消除序列從前往後能選就選
- 最小圖覆蓋 = 完美消除序列從前往後能選就選
- 區間圖是弦圖
- 區間圖的完美消除序列：將區間按造又端點由小到大的排序
- 區間圖染色：用線段樹做

```

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

```

#### 12.1.3 學長公式

- $\sum_{d|n} \phi(d) = n$
- $g(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d) * g(n/d)$
- $Harmonicseries H_n = \ln(n) + \gamma + 1/(2n) - 1/(12n^2) + 1/(120n^4)$
- $\gamma = 0.57721566490153286060651209008240243104215933593992$
- 格雷碼 =  $n \oplus (n >> 1)$
- $SG(A+B) = SG(A) \oplus SG(B)$
- 選轉矩陣  $M(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

#### 12.1.4 基本數論

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

#### 12.1.5 排組公式

- k 卡特蘭  $\frac{C_n^{kn}}{n(k-1)+1} \cdot C_m^n = \frac{n!}{m!(n-m)!}$
- $H(n,m) \cong x_1 + x_2 \dots + x_n = k, num = C_k^{n+k-1}$
- Stirling number of  $2^{2^d}$  ·  $n$  人分  $k$  組方法數目
- Bell number,  $n$  人分任意多組方法數目

- $B_0 = 1$
- $B_n = \sum_{i=0}^n S(n,i)$
- $B_{n+1} = \sum_{k=0}^n C_k^n B_k$
- $B_{p+n} \equiv B_n + B_{n+1} \pmod{p}$ ,  $p$  is prime
- $B_{p^m+n} \equiv mB_n + B_{n+1} \pmod{p}$ ,  $p$  is prime
- From B0:1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975

- Derangement, 錯排, 沒有人在自己位置上

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

#### 12.1.6 冪次, 冪次和

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

#### 12.1.7 Burnside's lemma

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



12.1.8 Count on a tree

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

(a) 完全圖  $n^n - 2$

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

12.1.9 積分表

- 1.  $\int \frac{1}{x} dx = \ln|x|$
- 2.  $\int u dv = uv - \int v du$
- 3.  $\int a^x dx = \frac{1}{\ln a} a^x$
- 4.  $\int \ln x dx = x \ln x - x$
- 5.  $\int \tan x dx = \ln|\sec x|$
- 6.  $\int \sec x dx = \ln|\sec x + \tan x|$
- 7.  $\int \sec^2 x dx = \tan x$
- 8.  $\int \sec x \tan x dx = \sec x$

- 9.  $\int \frac{a}{a^2+x^2} dx = \tan^{-1} \frac{x}{a}$
- 10.  $\int \frac{a}{a^2-x^2} dx = \frac{1}{2} \ln|\frac{x+a}{x-a}|$
- 11.  $\int \frac{1}{\sqrt{a^2-x^2}} dx = \sin^{-1} \frac{x}{a}$
- 12.  $\int \frac{a}{x\sqrt{x^2-a^2}} dx = \sec^{-1} \frac{x}{a}$
- 13.  $\int \frac{1}{\sqrt{x^2-a^2}} dx = \cosh^{-1} \frac{x}{a} = \ln(x + \sqrt{x^2-a^2})$
- 14.  $\int \frac{1}{\sqrt{x^2+a^2}} dx = \sinh^{-1} \frac{x}{a} = \ln(x + \sqrt{x^2+a^2})$

# ACM ICPC TEAM

## REFERENCE - NTHU

### JINKELA

#### Contents

<b>1 Computational_Geometry</b>	<b>1</b>	<b>4 Flow</b>	<b>7</b>	8.11 Simpson.cpp . . . . .	16
1.1 Geometry.cpp . . . . .	1	4.1 dinic.cpp . . . . .	7	8.12 WhatDay.cpp . . . . .	16
1.2 SmallestCircle.cpp . . . . .	3	4.2 ISAP_with_cut.cpp . . . . .	7	8.13 外星模運算.cpp . . . . .	16
1.3 最近點對.cpp . . . . .	3	4.3 MinCostMaxFlow.cpp . . . . .	8	8.14 模運算模板.cpp . . . . .	16
1.4 浮點數誤差模板.cpp . . . . .	3	<b>5 Graph</b>	<b>8</b>	8.15 質因數分解.cpp . . . . .	16
<b>2 Data_Structure</b>	<b>3</b>	5.1 Augmenting_Path.cpp . . . . .	8	<b>9 String</b>	<b>16</b>
2.1 DLX.cpp . . . . .	3	5.2 Augmenting_Path_multiple.cpp . . . . .	8	9.1 AC 自動機.cpp . . . . .	16
2.2 Dynamic_KD_tree.cpp . . . . .	4	5.3 blossom_matching.cpp . . . . .	8	9.2 hash.cpp . . . . .	17
2.3 kd_tree_replace_segment_tree.cpp . . . . .	5	5.4 graphISO.cpp . . . . .	9	9.3 KMP.cpp . . . . .	17
2.4 persistent_segment_tree.cpp . . . . .	5	5.5 KM.cpp . . . . .	9	9.4 manacher.cpp . . . . .	17
2.5 reference_point.cpp . . . . .	6	5.6 MaximumClique.cpp . . . . .	9	9.5 minimal_string_rotation.cpp . . . . .	17
2.6 skew_heap.cpp . . . . .	6	5.7 MinimumMeanCycle.cpp . . . . .	9	9.6 suffix_array_lcp.cpp . . . . .	17
2.7 split_merge.cpp . . . . .	6	5.8 Minimum_General_Weighted_Matching.cpp . . . . .	9	9.7 Z.cpp . . . . .	18
2.8 treap.cpp . . . . .	6	5.9 Rectilinear_Steiner_tree.cpp . . . . .	10	<b>10 Tarjan</b>	<b>18</b>
2.9 操作分治.cpp . . . . .	7	5.10 treeISO.cpp . . . . .	10	10.1 dominator_tree.cpp . . . . .	18
2.10 整體二分.cpp . . . . .	7	5.11 一般圖最大權匹配.cpp . . . . .	10	10.2 tnfsb017_2_sat.cpp . . . . .	18
<b>3 default</b>	<b>7</b>	5.12 全局最小割.cpp . . . . .	11	10.3 橋連通分量.cpp . . . . .	18
3.1 debug.cpp . . . . .	7	5.13 最小樹形圖_朱劉.cpp . . . . .	11	10.4 雙連通分量 & 割點.cpp . . . . .	18
3.2 ext.cpp . . . . .	7	<b>6 language</b>	<b>12</b>	<b>11 Tree_problem</b>	<b>19</b>
3.3 IncStack.cpp . . . . .	7	6.1 CNF.cpp . . . . .	12	11.1 HeavyLight.cpp . . . . .	19
3.4 input.cpp . . . . .	7	6.2 earley.cpp . . . . .	12	11.2 LCA.cpp . . . . .	19
		<b>7 Linear_Programming</b>	<b>13</b>	11.3 link_cut_tree.cpp . . . . .	19
		7.1 最大密度子圖.cpp . . . . .	13	11.4 POJ_tree.cpp . . . . .	20
		<b>8 Number_Theory</b>	<b>13</b>	<b>12 zformula</b>	<b>20</b>
		8.1 basic.cpp . . . . .	13	12.1 formula.tex . . . . .	20
		8.2 bit_set.cpp . . . . .	14	12.1.1 Pick 公式 . . . . .	20
		8.3 cantor_expansion.cpp . . . . .	14	12.1.2 圖論 . . . . .	20
		8.4 FFT.cpp . . . . .	14	12.1.3 學長公式 . . . . .	20
		8.5 find_real_root.cpp . . . . .	14	12.1.4 基本數論 . . . . .	20
		8.6 LinearCongruence.cpp . . . . .	15	12.1.5 排組公式 . . . . .	20
		8.7 Lucas.cpp . . . . .	15	12.1.6 冪次, 冪次和 . . . . .	20
		8.8 Matrix.cpp . . . . .	15	12.1.7 Burnside's lemma . . . . .	20
		8.9 MillerRobin.cpp . . . . .	15	12.1.8 Count on a tree . . . . .	20
		8.10 NTT.cpp . . . . .	15	12.1.9 積分表 . . . . .	21