

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

```

```

205 int n=p.size(),t=1;
206 T ans=0;p.push_back(p[0]);
207 for(int i=0;i<n;i++){
208     point<T> now=p[i+1]-p[i];
209     while(now.cross(p[t+1]-p[i])>now.cross
210         (p[t]-p[i]))t=(t+1)%n;
211     ans=max(ans,max((p[i]-p[t]).abs2()),(p[
212         i+1]-p[t+1]).abs2()));
213 }
214 return p.pop_back(),ans;
215 }
216 min_cover_rectangle(){//最小覆蓋矩形
217 int n=p.size(),t=1,r=1,l;
218 if(n<3)return 0;//也可以做最小周長矩形
219 T ans=1e99;p.push_back(p[0]);
220 for(int i=0;i<n;i++){
221     point<T> now=p[i+1]-p[i];
222     while(now.cross(p[t+1]-p[i])>now.cross
223         (p[t]-p[i]))t=(t+1)%n;
224     while(now.dot(p[r+1]-p[i])>now.dot(p[r
225         ]-p[i]))r=(r+1)%n;
226     if(!l)l=r;
227     while(now.dot(p[l+1]-p[i])<now.dot(p[
228         l]-p[i]))l=(l+1)%n;
229     T d=now.abs2();
230     T tmp=now.cross(p[t]-p[i])*(now.dot(p[
231         r]-p[i])-now.dot(p[l]-p[i]))/d;
232     ans=min(ans,tmp);
233 }
234 return p.pop_back(),ans;
235 }
236 max_triangle(){//最大內接三角形
237 int n=p.size(),a=1,b=2;
238 if(n<3)return 0;
239 T ans=0,tmp;p.push_back(p[0]);
240 for(int i=0;i<n;++i){
241     while((p[a]-p[i]).cross(p[b+1]-p[i])>
242         tmp=(p[a]-p[i]).cross(p[b]-p[i]))
243         b=(b+1)%n;
244     ans=max(ans,tmp);
245     while((p[a+1]-p[i]).cross(p[b]-p[i])>
246         tmp=(p[a]-p[i]).cross(p[b]-p[i]))
247         a=(a+1)%n;
248     ans=max(ans,tmp);
249 }
250 return p.pop_back(),ans/2;
251 }
252 dis2(polygon &p1){//凸包最近距離平方
253 vector<point<T>> &P=p,&Q=p1.p;
254 int n=P.size(),m=Q.size(),l=0,r=0;
255 for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
256 for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
257 P.push_back(P[0]),Q.push_back(Q[0]);
258 T ans=1e99;
259 for(int i=0;i<n;++i){
260     while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])
261         <0)r=(r+1)%m;
262     ans=min(ans,line<T>(P[l],P[l+1]).
263         seg_dis2(line<T>(Q[r],Q[r+1])));
264     l=(l+1)%n;
265 }
266 return P.pop_back(),Q.pop_back(),ans;
267 }
268 static char sign(const point<T>&t){
269     return (t.y==0?t.x:t.y)<0;
270 }
271 }
272 static bool angle_cmp(const line<T>& A,
273     const line<T>& B){
274     point<T> a=A.p2-A.p1,b=B.p2-B.p1;
275     return sign(a)<sign(b)||((sign(a)==sign(b)
276         )&&a.cross(b)>0);
277 }
278 int halfplane_intersection(vector<line<T>
279     > &s){//半平面交
280     sort(s.begin(),s.end(),angle_cmp);//線段
281     //左側為該線段半平面
282     int L,R,n=s.size();
283     vector<point<T>> > px(n);
284     vector<line<T>> > q(n);
285     q[L=R=0]=s[0];
286     for(int i=1;i<n;++i){
287         while(L<R&&s[i].cross(px[R-1])<=0)--R;
288         while(L<R&&s[i].cross(px[L])<=0)++L;
289         q[++R]=s[i];
290         if(q[R].parallel(q[R-1])){
291             --R;
292             if(q[R].cross(s[i].p1)>0)q[R]=s[i];
293         }
294         if(L<R)px[R-1]=q[R-1].
295             line_intersection(q[R]);
296     }
297     while(L<R&&q[L].cross(px[R-1])<=0)--R;
298     p.clear();
299     if(R-L<=1)return 0;
300     px[R]=q[R].line_intersection(q[L]);
301     for(int i=L;i<R;++i)p.push_back(px[i]);
302     return R-L+1;
303 }
304 }
305 template<typename T>
306 struct triangle{
307     point<T> a,b,c;
308     triangle(){
309         triangle(const point<T> &a,const point<T>
310             &b,const point<T> &c):a(a),b(b),c(c){}
311     }
312     T area()const{
313         T t=(b-a).cross(c-a)/2;
314         return t>0?t:-t;
315     }
316     point<T> barycenter()const{//重心
317         return (a+b+c)/3;
318     }
319     point<T> circumcenter()const{//外心
320         static line<T> u,v;
321         u.p1=(a+b)/2;
322         u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
323             b.x);
324         v.p1=(a+c)/2;
325         v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
326             c.x);
327         return u.line_intersection(v);
328     }
329     point<T> incenter()const{//內心
330         T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
331             ()),C=sqrt((a-b).abs2());
332         return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
333             B*b.y+C*c.y)/(A+B+C);
334     }
335     point<T> perpcenter()const{//垂心
336         return barycenter()*3-circumcenter()*2;
337     }
338 }
339 template<typename T>
340 struct point3D{
341     T x,y,z;
342     point3D(){
343         point3D(const T&x,const T&y,const T&z):x(x
344             ),y(y),z(z){}
345     }
346     point3D operator+(const point3D &b)const{
347         return point3D(x+b.x,y+b.y,z+b.z);
348     }
349     point3D operator-(const point3D &b)const{
350         return point3D(x-b.x,y-b.y,z-b.z);
351     }
352     point3D operator*(const T &b)const{
353         return point3D(x*b,y*b,z*b);
354     }
355     point3D operator/(const T &b)const{
356         return point3D(x/b,y/b,z/b);
357     }
358     bool operator==(const point3D &b)const{
359         return x==b.x&&y==b.y&&z==b.z;
360     }
361     T dot(const point3D &b)const{
362         return x*b.x+y*b.y+z*b.z;
363     }
364     point3D cross(const point3D &b)const{
365         return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
366             *b.y-y*b.x);
367     }
368     T abs2()const{//向量长度的平方
369         return dot(*this);
370     }
371     T area2(const point3D &b)const{//和b、原點
372         //圍成面積的平方
373         return cross(b).abs2()/4;
374     }
375 };
376 template<typename T>
377 struct line3D{
378     point3D<T> p1,p2;
379     line3D(){
380         line3D(const point3D<T> &p1,const point3D<
381             T> &p2):p1(p1),p2(p2){}
382     }
383     T dis2(const point3D<T> &p,bool is_segment
384         =0)const{//點跟直線/線段的距離平方
385         point3D<T> v=p2-p1,v1=p-p1;
386         if(is_segment){
387             point3D<T> v2=p-p2;
388             if(v.dot(v1)<=0)return v1.abs2();
389             if(v.dot(v2)>=0)return v2.abs2();
390         }
391         point3D<T> tmp=v.cross(v1);
392         return tmp.abs2()/v.abs2();
393     }
394     pair<point3D<T>,point3D<T>> closest_pair(
395         const line3D<T> &l1)const{
396         point3D<T> v1=(p1-p2),v2=(l1.p1-l1.p2);
397         point3D<T> N=v1.cross(v2),ab(p1-l1.p1);
398         //if(N.abs2()==0)return NULL;平行或重合
399         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
400         //最近點距離
401         point3D<T> d1=p2-p1,d2=l1.p1-l1.p1,D=d1.
402             cross(d2),G=l1.p1-p1;
403         T t1=(G.cross(d2)).dot(D)/D.abs2();
404         T t2=(G.cross(d1)).dot(D)/D.abs2();
405         return make_pair(p1+d1*t1,l1.p1+d2*t2);
406     }
407     bool same_side(const point3D<T> &a,const
408         point3D<T> &b)const{
409         return (p2-p1).cross(a-p1).dot((p2-p1).
410             cross(b-p1))>0;
411     }
412 };
413 template<typename T>
414 struct plane{
415     point3D<T> p0,n;//平面上的點和法向量
416     plane(){
417         plane(const point3D<T> &p0,const point3D<T>
418             &n):p0(p0),n(n){}
419     }
420     T dis2(const point3D<T> &p)const{//點到平
421         //面距離的平方
422         T tmp=(p-p0).dot(n);
423         return tmp*tmp/n.abs2();
424     }
425     point3D<T> projection(const point3D<T> &p)
426         const{
427         return p-n*(p-p0).dot(n)/n.abs2();
428     }
429     point3D<T> line_intersection(const line3D<
430         T> &l1)const{
431         T tmp=n.dot(l1.p2-l1.p1);//等於0表示平行或
432         //重合該平面
433         return l1.p1+(l1.p2-l1.p1)*(n.dot(p0-l1.p1)/
434             tmp);
435     }
436     line3D<T> plane_intersection(const plane &
437         p1)const{
438         point3D<T> e=n.cross(p1.n),v=n.cross(e);
439         T tmp=p1.n.dot(v);//等於0表示平行或重合
440         //該平面
441         point3D<T> q=p0+(v*(p1.n.dot(p1.p0-p0))/
442             tmp);
443         return line3D<T>(q,q+e);
444     }
445 };
446 template<typename T>
447 struct triangle3D{
448     point3D<T> a,b,c;
449     triangle3D(){
450         triangle3D(const point3D<T> &a,const
451             point3D<T> &b,const point3D<T> &c):a(a
452             ),b(b),c(c){}
453     }
454     bool point_in(const point3D<T> &p)const{//
455         //點在該平面上的投影在三角形中
456         return line3D<T>(b,c).same_side(p,a)&&
457             line3D<T>(a,c).same_side(p,b)&&
458             line3D<T>(a,b).same_side(p,c);
459     }
460 };
461 template<typename T>
462 struct tetrahedron{//四面體
463     point3D<T> a,b,c,d;
464     tetrahedron(){
465         tetrahedron(const point3D<T> &a,const
466             point3D<T> &b,const point3D<T> &c,
467             const point3D<T> &d):a(a),b(b),c(c),d(
468             d){}
469     }
470     T volume6()const{//體積的六倍
471         return (d-a).dot((b-a).cross(c-a));
472     }
473     point3D<T> centroid()const{
474         return (a+b+c+d)/4;
475     }
476     bool point_in(const point3D<T> &p)const{
477         return triangle3D<T>(a,b,c).point_in(p)
478             &&triangle3D<T>(c,d,a).point_in(p);
479     }
480 };

```

```

413 }
414 };
415 template<typename T>
416 struct convexhull3D{
417     static const int MAXN=1005;
418     struct face{
419         int a,b,c;
420         face(int a,int b,int c):a(a),b(b),c(c){}
421     };
422     vector<point3D<T>> pt;
423     vector<face> ans;
424     int fid[MAXN][MAXN];
425     void build(){
426         int n=pt.size();
427         ans.clear();
428         memset(fid,0,sizeof(fid));
429         ans.emplace_back(0,1,2); //注意不能共線
430         ans.emplace_back(2,1,0);
431         int ftop = 0;
432         for(int i=3, ftop=1; i<n; ++i, ++ftop){
433             vector<face> next;
434             for(auto &f:ans){
435                 T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[f.a]).cross(pt[f.c]-pt[f.a]));
436                 if(d<=0) next.push_back(f);
437                 int ff=0;
438                 if(d>0) ff=ftop;
439                 else if(d<0) ff=-ftop;
440                 fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c][f.a]=ff;
441             }
442             for(auto &f:ans){
443                 if(fid[f.a][f.b]>0 && fid[f.a][f.b]!=fid[f.b][f.a])
444                     next.emplace_back(f.a,f.b,i);
445                 if(fid[f.b][f.c]>0 && fid[f.b][f.c]!=fid[f.c][f.b])
446                     next.emplace_back(f.b,f.c,i);
447                 if(fid[f.c][f.a]>0 && fid[f.c][f.a]!=fid[f.a][f.c])
448                     next.emplace_back(f.c,f.a,i);
449             }
450             ans=next;
451         }
452     }
453     point3D<T> centroid()const{
454         point3D<T> res(0,0,0);
455         T vol=0;
456         for(auto &f:ans){
457             T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c]));
458             res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
459             vol+=tmp;
460         }
461         return res/(vol*4);
462     }
463 };

```

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

```

1.3 最近點對.cpp

```

1 template<typename _IT=point<T>* >
2 T closest_pair(_IT L, _IT R){
3     if(R-L <= 1) return INF;
4     _IT mid = L+(R-L)/2;
5     T x = mid->x;

```

```

6     T d = min(closest_pair(L,mid),closest_pair(mid,R));
7     inplace_merge(L, mid, R, ycmp);
8     static vector<point> b; b.clear();
9     for(auto u=L;u<R;++u){
10         if((u->x-x)*(u->x-x)>=d) continue;
11         for(auto v=b.rbegin();v!=b.rend();++v){
12             T dx=u->x-v->x, dy=u->y-v->y;
13             if(dy*dy>=d) break;
14             d=min(d,dx*dx+dy*dy);
15         }
16         b.push_back(*u);
17     }
18     return d;
19 }
20 T closest_pair(vector<point<T>> &v){
21     sort(v.begin(),v.end(),xcmp);
22     return closest_pair(v.begin(),v.end());
23 }

```

2 Data_Structure

2.1 DLX.cpp

```

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

```

```

29 }
30 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
31     DFOR(i,U,c)DFOR(j,L,i){++S[col[j]],U[D[j]]=j,D[U[j]]=j;
32     L[R[c]]=c,R[L[c]]=c;
33 }
34 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
35     DFOR(i,D,nd)L[R[i]]=L[i],R[L[i]]=R[i];
36 }
37 void restore2(int nd){ //刪除nd所在的行當前所有點 為remove2的逆操作
38     DFOR(i,U,nd)L[R[i]]=R[L[i]]=i;
39 }
40 bool vis[MAXN];
41 int h(){ //估價函數 for IDA*
42     int res=0;
43     memset(vis,0,sizeof(vis));
44     DFOR(i,R,0)if(!vis[i]){
45         vis[i]=1;
46         ++res;
47         DFOR(j,D,i)DFOR(k,R,j)vis[col[k]]=1;
48     }
49     return res;
50 }
51 bool dfs(int d){ //for精確覆蓋問題
52     if(d+h()>=ansd)return 0; //找最佳解用 找任意解可以刪掉
53     if(!R[0]){ansd=d;return 1;}
54     int c=R[0];
55     DFOR(i,R,0)if(S[i]<S[c])c=i;
56     remove(c);
57     DFOR(i,D,c){
58         ans.push_back(row[i]);
59         DFOR(j,R,i)remove2(col[j]);
60         if(dfs(d+1))return 1;
61         ans.pop_back();
62         DFOR(j,L,i)restore2(col[j]);
63     }
64     restore(c);
65     return 0;
66 }
67 void dfs2(int d){ //for最小重複覆蓋問題
68     if(d+h()>=ansd)return;
69     if(!R[0]){ansd=d;ans=anst;return;}
70     int c=R[0];
71     DFOR(i,R,0)if(S[i]<S[c])c=i;
72     DFOR(i,D,c){
73         anst.push_back(row[i]);
74         remove2(i);
75         DFOR(j,R,i)remove2(j),--S[col[j]];
76         dfs2(d+1);
77         anst.pop_back();
78         DFOR(j,L,i)restore2(j),++S[col[j]];
79         restore2(i);
80     }
81 }
82 bool exact_cover(){ //解精確覆蓋問題
83     return ans.clear(), dfs(0);
84 }
85 void min_cover(){ //解最小重複覆蓋問題
86     anst.clear(); //暫存用 答案還是存在ans裡

```

2.2 Dynamic_KD_tree.cpp

```

87     dfs2(0);
88 }
89 #undef DFOR
90 };

template<typename T, size_t kd> //有kd個維度
struct kd_tree{
    struct point{
        T d[kd];
        T dist(const point &x) const{
            T ret=0;
            for(size_t i=0; i<kd; ++i) ret+=std::abs(
                d[i]-x.d[i]);
            return ret;
        }
        bool operator==(const point &p){
            for(size_t i=0; i<kd; ++i)
                if(d[i]!=p.d[i]) return 0;
            return 1;
        }
        bool operator<(const point &b) const{
            return d[0]<b.d[0];
        }
    };
private:
    struct node{
        node *l, *r;
        point pid;
        int s;
        node(const point &p): l(0), r(0), pid(p), s
            (1){}
        ~node(){ delete l; delete r; }
        void up(){ s=(l?l->s:0)+1+(r?r->s:0); }
    } *root;
    const double alpha, loga;
    const T INF; //記得要給INF，表示極大值
    int maxn;
    struct __cmp{
        int sort_id;
        bool operator()(const node *x, const node *
            y) const{
            return operator()(x->pid, y->pid);
        }
        bool operator()(const point &x, const
            point &y) const{
            if(x.d[sort_id]!=y.d[sort_id])
                return x.d[sort_id]<y.d[sort_id];
            for(size_t i=0; i<kd; ++i)
                if(x.d[i]!=y.d[i]) return x.d[i]<y.d[
                    i];
            return 0;
        }
    } cmp;
    int size(node *o){ return o?o->s:0; }
    std::vector<node*> A;
    node* build(int k, int l, int r){
        if(l>r) return 0;
        if(k==kd) k=0;
        int mid=(l+r)/2;
        cmp.sort_id = k;

```

```

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

```

```

109     if(erase(cmp(x, u->pid)?u->l:u->r, (k+1)%
        kd, x))
110         return --u->s, 1;
111     return 0;
112 }
113 T heuristic(const T h[]) const{
114     T ret=0;
115     for(size_t i=0; i<kd; ++i) ret+=h[i];
116     return ret;
117 }
118 int qM;
119 std::priority_queue<std::pair<T, point>>
    pQ;
120 void nearest(node *u, int k, const point &x,
    T *h, T &mndist){
121     if(u==0||heuristic(h)>=mndist) return;
122     T dist=u->pid.dist(x), old=h[k];
123     /*mndist=std::min(mndist, dist);*/
124     if(dist<mndist){
125         pQ.push(std::make_pair(dist, u->pid));
126         if((int)pQ.size()==qM+1)
127             mndist=pQ.top().first, pQ.pop();
128     }
129     if(x.d[k]<u->pid.d[k]){
130         nearest(u->l, (k+1)%kd, x, h, mndist);
131         h[k]=std::abs(x.d[k]-u->pid.d[k]);
132         nearest(u->r, (k+1)%kd, x, h, mndist);
133     } else{
134         nearest(u->r, (k+1)%kd, x, h, mndist);
135         h[k]=std::abs(x.d[k]-u->pid.d[k]);
136         nearest(u->l, (k+1)%kd, x, h, mndist);
137     }
138     h[k]=old;
139 }
140 std::vector<point> in_range;
141 void range(node *u, int k, const point &mi,
    const point &ma){
142     if(!u) return;
143     bool is=1;
144     for(int i=0; i<kd; ++i)
145         if(u->pid.d[i]<mi.d[i]||ma.d[i]<u->pid
            .d[i]){
146             is=0; break;
147         }
148     if(is) in_range.push_back(u->pid);
149     if(mi.d[k]<u->pid.d[k]) range(u->l, (k+1)
        %kd, mi, ma);
150     if(ma.d[k]>u->pid.d[k]) range(u->r, (k+1)
        %kd, mi, ma);
151 }
152 public:
153 kd_tree(const T &INF, double a=0.75): root
    (0), alpha(a), loga(log2(1.0/a)), INF(INF
        ), maxn(1){}
154 ~kd_tree(){ delete root; }
155 void clear(){ delete root; root=0, maxn=1; }
156 void build(int n, const point *p){
157     delete root; A.resize(maxn=n);
158     for(int i=0; i<n; ++i) A[i]=new node(p[i]);
159     root=build(0, 0, n-1);
160 }
161 void insert(const point &x){
162     insert(root, 0, x, __lg(size(root))/loga);
163     if(root->s>maxn) maxn=root->s;
164 }
165 bool erase(const point &p){

```

```

166     bool d=erase(root, 0, p);
167     if(root&&root->s<alpha*maxn) rebuild();
168     return d;
169 }
170 void rebuild(){
171     if(root) rebuild(root, 0);
172     maxn=root->s;
173 }
174 T nearest(const point &x, int k){
175     qM=k;
176     T mndist=INF, h[kd]={};
177     nearest(root, 0, x, h, mndist);
178     mndist=pQ.top().first;
179     pQ=std::priority_queue<std::pair<T, point
        >>();
180     return mndist; //回傳離x第k近的點的距離
181 }
182 const std::vector<point> &range(const
    point &mi, const point &ma){
183     in_range.clear();
184     range(root, 0, mi, ma);
185     return in_range; //回傳介於mi到ma之間的點
        vector
186 }
187 int size(){ return root?root->s:0; }
188 };

```

2.3 kd_tree_replace_segment_tr

```

1 /*kd樹代替高維線段樹*/
2 struct node{
3     node *l, *r;
4     point pid, mi, ma;
5     int s;
6     int data;
7     node(const point &p, int d): l(0), r(0), pid(p
        ), mi(p), ma(p), s(1), data(d), dmin(d),
        dmax(d){}
8     void up(){
9         mi=ma=pid;
10        s=1;
11        if(l){
12            for(int i=0; i<kd; ++i){
13                mi.d[i]=min(mi.d[i], l->mi.d[i]);
14                ma.d[i]=max(ma.d[i], l->ma.d[i]);
15            }
16        }
17        s+=l->s;
18        if(r){
19            for(int i=0; i<kd; ++i){
20                mi.d[i]=min(mi.d[i], r->mi.d[i]);
21                ma.d[i]=max(ma.d[i], r->ma.d[i]);
22            }
23            s+=r->s;
24        }
25    }
26    void up2(){
27        //其他懶惰標記向上更新
28    }
29    void down(){
30        //其他懶惰標記下推
31    }

```



```

32 }*root;
33
34 /*檢查區間包含用的函數*/
35 inline bool range_include(node *o,const
    point &L,const point &R){
36     for(int i=0;i<kd;++i){
37         if(L.d[i]>o->ma.d[i]||R.d[i]<o->mi.d[i])
            return 0;
38     }//只要(L,R)區間有和o的區間有交集就回傳
        true
39     return 1;
40 }
41 inline bool range_in_range(node *o,const
    point &L,const point &R){
42     for(int i=0;i<kd;++i){
43         if(L.d[i]>o->mi.d[i]||o->ma.d[i]>R.d[i])
            return 0;
44     }//如果(L,R)區間完全包含o的區間就回傳true
45     return 1;
46 }
47 inline bool point_in_range(node *o,const
    point &L,const point &R){
48     for(int i=0;i<kd;++i){
49         if(L.d[i]>o->pid.d[i]||R.d[i]<o->pid.d[i])
            return 0;
50     }//如果(L,R)區間完全包含o->pid這個點就回傳
        true
51     return 1;
52 }
53
54 /*單點修改 · 以單點改值為例*/
55 void update(node *u,const point &x,int data,
    int k=0){
56     if(!u)return;
57     u->down();
58     if(u->pid==x){
59         u->data=data;
60         u->up2();
61         return;
62     }
63     cmp.sort_id=k;
64     update(cmp(x,u->pid)?u->l:u->r,x,data,(k
        +1)%kd);
65     u->up2();
66 }
67
68 /*區間修改*/
69 void update(node *o,const point &L,const
    point &R,int data){
70     if(!o)return;
71     o->down();
72     if(range_in_range(o,L,R)){
73         //區間懶惰標記修改
74         o->down();
75         return;
76     }
77     if(point_in_range(o,L,R)){
78         //這個點在(L,R)區間 · 但是他的左右子樹不
            一定在區間中
79         //單點懶惰標記修改
80     }
81     if(o->l&&range_include(o->l,L,R))update(o
        ->l,L,R,data);

```

```

82     if(o->r&&range_include(o->r,L,R))update(o
        ->r,L,R,data);
83     o->up2();
84 }
85
86 /*區間查詢 · 以總和為例*/
87 int query(node *o,const point &L,const point
    &R){
88     if(!o)return 0;
89     o->down();
90     if(range_in_range(o,L,R))return o->sum;
91     int ans=0;
92     if(point_in_range(o,L,R))ans+=o->data;
93     if(o->l&&range_include(o->l,L,R))ans+=
        query(o->l,L,R);
94     if(o->r&&range_include(o->r,L,R))ans+=
        query(o->r,L,R);
95     return ans;
96 }

```

2.4 reference_point.cpp

```

1 template<typename T>
2 struct _RefC{
3     T data;
4     int ref;
5     _RefC(const T&d=0):data(d),ref(0){}
6 };
7 template<typename T>
8 struct _rp{
9     _RefC<T> *p;
10    T *operator->(){return &p->data;}
11    T &operator*(){return p->data;}
12    operator _RefC<T>*(){return p;}
13    _rp &operator=(const _rp &t){
14        if(p&&!--p->ref)delete p;
15        p=t.p,p&&+p->ref;
16        return *this;
17    }
18    _rp(_RefC<T> *t=0):p(t){p&&+p->ref;}
19    _rp(const _rp &t):p(t.p){p&&+p->ref;}
20    ~_rp(){if(p&&!--p->ref)delete p;}
21 };
22 template<typename T>
23 inline _rp<T> new_rp(const T&nd){
24     return _rp<T>(new _RefC<T>(nd));
25 }

```

2.5 skew_heap.cpp

```

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

```

2.6 undo_disjoint_set.cpp

```

1 struct DisjointSet {
2     // save() is like recursive
3     // undo() is like return
4     int n, fa[MXN], sz[MXN];
5     vector<pair<int*,int>> h;
6     vector<int> sp;
7     void init(int tn) {
8         n=tn;
9         for (int i=0; i<n; i++) sz[fa[i]=i]=1;
10        sp.clear(); h.clear();
11    }
12    void assign(int *k, int v) {
13        h.PB({k, *k});
14        *k=v;
15    }
16    void save() { sp.PB(SZ(h)); }
17    void undo() {
18        assert(!sp.empty());
19        int last=sp.back(); sp.pop_back();
20        while (SZ(h)!=last) {
21            auto x=h.back(); h.pop_back();
22            *x.F=x.S;
23        }
24    }
25    int f(int x) {
26        while (fa[x]!=x) x=fa[x];
27        return x;
28    }
29    void uni(int x, int y) {
30        x=f(x); y=f(y);
31        if (x==y) return ;
32        if (sz[x]<sz[y]) swap(x, y);
33        assign(&sz[x], sz[x]+sz[y]);
34        assign(&fa[y], x);
35    }
36 }djs;

```

2.7 整體二分.cpp

```

1 void totBS(int L, int R, vector<Item> M){
2     if(Q.empty()) return; //維護全域B陣列
3     if(L==R) 整個M的答案=r, return;
4     int mid = (L+R)/2;
5     vector<Item> mL, mR;
6     do_modify_B_with_divide(mid,M);
7     //讓B陣列在遞迴的時候只會保留[L~mid]的資訊
8     undo_modify_B(mid,M);
9     totBS(L,mid,mL);
10    totBS(mid+1,R,mR);
11 }

```

3 default

3.1 debug.cpp

```

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

```

3.2 ext.cpp

```

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

```

3.3 IncStack.cpp

```

1 //Magic
2 #pragma GCC optimize "Ofast"
3 //stack resize,change esp to rsp if 64-bit
    system
4 asm("mov %0,%esp\n" :: "g"(mem+10000000));
5 -Wl,--stack,214748364 -trigraphs
6 //linux stack resize
7 #include<sys/resource.h>
8 void increase_stack(){
9     const rlim_t ks=64*1024*1024;
10    struct rlimit rl;
11    int res=getrlimit(RLIMIT_STACK,&rl);
12    if(!res&&rl.rlim_cur<ks){
13        rl.rlim_cur=ks;
14        res=setrlimit(RLIMIT_STACK,&rl);
15    }
16 }

```

3.4 input.cpp

```

1 inline int read(){
2     int x=0; bool f=0; char c=getchar();
3     while(ch<'0' || '9'<ch) f|=ch=='-' ,ch=getchar
4     ();
5     while('0'<=ch&&ch<='9') x=x*10-'0'+ch,ch=
6     getchar();
7     return f?-x:x;
8 // #!/bin/bash
9 // g++ -std=c++11 -O2 -Wall -Wextra -Wno-
  unused-result -DDEBUG $1 && ./a.out
  // -fsanitize=address -fsanitize=undefined
  -fsanitize=return

```

4 Flow

4.1 dinic.cpp

```

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

```

```

38 }
39 return 0;
40 }
41 T dfs(int u,int t,T cur_flow=INF){
42     if(u==t) return cur_flow;
43     T df;
44     for(int &i=cur[u];~i;i=e[i].pre){
45         if(level[e[i].v]==level[u]+1&&e[i].r){
46             if(df=dfs(e[i].v,t,min(cur_flow,e[i]
47             ].r))){
48                 e[i].flow+=df;
49                 e[i^1].flow-=df;
50                 e[i].r-=df;
51                 e[i^1].r+=df;
52                 return df;
53             }
54         }
55     }
56     return level[u]=0;
57 }
58 T dinic(int s,int t,bool clean=true){
59     if(clean){
60         for(size_t i=0;i<e.size();++i){
61             e[i].flow=0;
62             e[i].r=e[i].cap;
63         }
64     }
65     T ans=0, mf=0;
66     while(bfs(s,t)) while(mf=dfs(s,t)) ans+=mf;
67     return ans;
68 };

```

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 df;
29     for(int &i=cur[u];~i;i=e[i].pre){
30         if(level[e[i].v]==level[u]+1&&e[i].r){
31             if(df=dfs(e[i].v,t,min(cur_flow,e[i]
32             ].r))){
33                 e[i].flow+=df;
34                 e[i^1].flow-=df;
35                 e[i].r-=df;
36                 e[i^1].r+=df;
37                 return df;
38             }
39         }
40     }
41     return level[u]=0;
42 }
43 T isap(int s,int t,bool clean=true){
44     memset(d,0,sizeof(int)*(n+1));
45     memset(gap,0,sizeof(int)*(n+1));
46     memcpy(cur,g,sizeof(int)*(n+1));
47     if(clean) for(size_t i=0;i<e.size();++i){
48         e[i].flow=0;
49         e[i].r=e[i].cap;
50     }
51     T max_flow=0;
52     for(gap[0]=n;d[s]<n;d[s]--){
53         max_flow+=dfs(s,s,t);
54     }
55     return max_flow;
56 }
57 vector<int> cut_e; //最小割邊集
58 bool vis[MAXN];
59 void dfs_cut(int u){
60     vis[u]=1; //表示u屬於source的最小割集
61     for(int i=g[u];~i;i=e[i].pre){
62         if(e[i].flow<e[i].cap&&vis[e[i].v]){
63             dfs_cut(e[i].v);
64         }
65     }
66     T min_cut(int s,int t){
67         T ans=isap(s,t);
68         memset(vis,0,sizeof(bool)*(n+1));
69         dfs_cut(s); cut_e.clear();
70         for(int u=0;u<n;u++){
71             if(vis[u]) for(int i=g[u];~i;i=e[i].pre){
72                 if(!vis[e[i].v]) cut_e.push_back(i);
73             }
74         }
75     }
76 }
77 T min_cost_max_flow(int s,int t,T cur_flow=INF){
78     T ans=isap(s,t);
79     min_cut(s,t);
80     return ans;
81 }

```

```

25 if(u==t) return cur_flow;
26 T tf=cur_flow,df;
27 for(int &i=cur[u];~i;i=e[i].pre){
28     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
35         cur_flow-tf;
36     }
37 }
38 int mh=n;
39 for(int i=cur[u]=g[u];~i;i=e[i].pre){
40     if(e[i].r&&d[e[i].v]<mh) mh=d[e[i].v];
41 }
42 if(!-gap[d[u]]) d[s]=n;
43 else ++gap[d[u]]=mh;
44 return cur_flow-tf;
45 }
46 T isap(int s,int t,bool clean=true){
47     memset(d,0,sizeof(int)*(n+1));
48     memset(gap,0,sizeof(int)*(n+1));
49     memcpy(cur,g,sizeof(int)*(n+1));
50     if(clean) for(size_t i=0;i<e.size();++i){
51         e[i].flow=0;
52         e[i].r=e[i].cap;
53     }
54     T max_flow=0;
55     for(gap[0]=n;d[s]<n;d[s]--){
56         max_flow+=dfs(s,s,t);
57     }
58     return max_flow;
59 }
60 vector<int> cut_e; //最小割邊集
61 bool vis[MAXN];
62 void dfs_cut(int u){
63     vis[u]=1; //表示u屬於source的最小割集
64     for(int i=g[u];~i;i=e[i].pre){
65         if(e[i].flow<e[i].cap&&vis[e[i].v]){
66             dfs_cut(e[i].v);
67         }
68     }
69     T min_cut(int s,int t){
70         T ans=isap(s,t);
71         memset(vis,0,sizeof(bool)*(n+1));
72         dfs_cut(s); cut_e.clear();
73         for(int u=0;u<n;u++){
74             if(vis[u]) for(int i=g[u];~i;i=e[i].pre){
75                 if(!vis[e[i].v]) cut_e.push_back(i);
76             }
77         }
78     }
79 }
80 T min_cost_max_flow(int s,int t,T cur_flow=INF){
81     T ans=isap(s,t);
82     min_cut(s,t);
83     return ans;
84 }

```

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             d=augment(e[i].v,min(r,e[i].cap));
36             e[i].cap-=d;
37             e[i^1].cap+=d;
38             if(!(r-=d)) break;
39         }
40     }
41     return cur_flow-r;
42 }
43 bool modlabel(){
44     for(int u=0;u<n;u++) dis[u]=INF;
45     static deque<int> q;
46     dis[t]=0,q.push_back(t);
47     while(q.size()){
48         int u=q.front();q.pop_front();
49         _T dt;
50         for(int i=g[u];~i;i=e[i].pre){
51             if(e[i^1].cap&&(dt=dis[u]-e[i].cost)
52             <dis[e[i].v]){
53                 if((dis[e[i].v]=dt)<=dis[q.size()])
54                 q.front():S){}
55                 q.push_front(e[i].v);
56             } else q.push_back(e[i].v);
57         }
58     }
59 }
60 for(int u=0;u<n;u++){
61     for(int i=g[u];~i;i=e[i].pre)
62         e[i].cost+=dis[e[i].v]-dis[u];
63     return piS+=dis[S], dis[S]<INF;
64 }
65 _T mincost(int s,int t){
66     S=s,T=t;
67     piS=ans=0;
68     while(modlabel()){
69         do memset(vis,0,sizeof(bool)*(n+1));
70         while(augment(S,INF));
71     }
72 }

```

```

65     }return ans;
66 }
67 };

```

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            return match[v]=u, 1;
14    }
15    return 0;
16 }
17 inline int max_match(){
18     int ans=0;
19     memset(match,-1,sizeof(int)*n2);
20     for(int i=0;i<n1;++i){
21         memset(vis,0,sizeof(bool)*n2);
22         if(dfs(i))++ans;
23     }
24     return ans;
25 }

```

5.2 Augmenting_Path_multiple

```

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

```

```

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

```

5.3 blossom_matching.cpp

```

1 #define MAXN 505
2 vector<int> g[MAXN];
3 int pa[MAXN],match[MAXN],st[MAXN],S[MAXN],v[
4     MAXN];
5 int t,n;
6 int lca(int x,int y){
7     for(++t;swap(x,y)){
8         if(x==0)continue;
9         if(v[x]==t)return x;
10        v[x]=t;
11        x=st[pa[match[x]]];
12    }
13 }
14 #define qpush(x) q.push(x),S[x]=0
15 void flower(int x,int y,int l,queue<int> &q)
16 {
17     while(st[x]!=1){
18         pa[x]=y;
19         if(S[y==match[x]]==1)qpush(y);
20         st[x]=st[y]=1, x=pa[y];
21     }
22 }
23 bool bfs(int x){
24     for(int i=1;i<n;++i)st[i]=i;
25     memset(S+1,-1,sizeof(int)*n);
26     queue<int> q; qpush(x);
27     while(q.size()){
28         x=q.front(),q.pop();
29         for(size_t i=0;i<g[x].size();++i){
30             int y=g[x][i];
31             if(S[y]==-1){
32                 pa[y]=x,S[y]=1;
33                 if(!match[y]){
34                     for(int l=st[x],x=pa[y])
35                         l=match[x],match[x]=y,match[y]
36                             =x;
37                     return 1;
38                 }
39                 qpush(match[y]);
40             }else if(!S[y]&&st[y]!=st[x]){

```

```

38         int l=lca(y,x);
39         flower(y,x,l,q),flower(x,y,l,q);
40     }
41 }
42 }
43 return 0;
44 }
45 inline 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 void init(){
8     for(int i=0;i<n;++i){
9         f[0][i]=1;
10        g[i].clear(), rg[i].clear();
11    }
12 }
13 void add_edge(int u,int v){
14     g[u].push_back(v), rg[v].push_back(u);
15 }
16 long long point_hash(int u){ //O(N)
17     for(int t=1;t<=K;++t){
18         for(int i=0;i<n;++i){
19             f[t][i]=f[t-1][i]*A%P;
20             for(int j:g[i])f[t][i]=(f[t][i]+f[t-1][j]*B%P)%P;
21             for(int j:rg[i])f[t][i]=(f[t][i]+f[t-1][j]*C%P)%P;
22             if(i==u)f[t][i]+=D; //如果圖太大的話，
23                 把這行刪掉，執行一次後f[K]就會是所
24                 有點的答案
25             f[t][i]%=P;
26         }
27     }
28     return f[K][u];
29 }
30 vector<long long> graph_hash(){
31     vector<long long> ans;
32     for(int i=0;i<n;++i)ans.push_back(
33         point_hash(i)); //O(N^2)
34     sort(ans.begin(),ans.end());
35     return ans;
36 }

```

5.5 KM.cpp

```

1 #define MAXN 405
2 #define INF 0x3f3f3f3f

```

```

3 int n;// 1-base，0表示沒有匹配
4 int g[MAXN][MAXN],lx[MAXN],ly[MAXN],pa[MAXN],
5     slack_y[MAXN];
6 int match_y[MAXN],match_x[MAXN];
7 bool vx[MAXN],vy[MAXN];
8 void augment(int y){
9     for(int x,z;y=y=z){
10        x=pa[y],z=match_x[x];
11        match_y[y]=x,match_x[x]=y;
12    }
13 }
14 void bfs(int st){
15     for(int i=1;i<n;++i)slack_y[i]=INF,vx[i]=
16         vy[i]=0;
17     queue<int> q;q.push(st);
18     for(;;){
19         while(q.size()){
20             int x=q.front(),q.pop();
21             vx[x]=1;
22             for(int y=1;y<n;++y)if(!vy[y]){
23                 int t=lx[x]+ly[y]-g[x][y];
24                 if(t==0){
25                     pa[y]=x;
26                     if(!match_y[y]){augment(y);return}
27                     vy[y]=1,q.push(match_y[y]);
28                 }else if(slack_y[y]>t)pa[y]=x,
29                     slack_y[y]=t;
30             }
31         }
32         int cut=INF;
33         for(int y=1;y<n;++y){
34             if(!vy[y]&&cut>slack_y[y])cut=slack_y[
35                 y];
36         }
37         for(int y=1;y<n;++y){
38             if(!vy[y]&&slack_y[y]==0){
39                 if(!match_y[y]){augment(y);return;}
40                 vy[y]=1,q.push(match_y[y]);
41             }
42         }
43     }
44 }
45 }
46 long long KM(){
47     memset(match_y,0,sizeof(int)*(n+1));
48     memset(ly,0,sizeof(int)*(n+1));
49     for(int x=1;x<n;++x){
50         lx[x]=-INF;
51         for(int y=1;y<n;++y)
52             lx[x]=max(lx[x],g[x][y]);
53     }
54     for(int x=1;x<n;++x)bfs(x);
55     long long ans=0;
56     for(int y=1;y<n;++y)ans+=g[match_y[y]][y];
57     return ans;
58 }

```

5.6 MaximumClique.cpp

```

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

```

5.7 MinimumMeanCycle.cpp

```

1 #include<cstdio>//for DBL_MAX
2 int dp[maxN+1][maxN+1];
3 double mnc(int n){
4     int u,v,w;
5     const int inf=0x7f7f7f7f;
6     memset(dp,0x7f,sizeof(dp));
7     memset(dp[0],0,sizeof(dp[0]));
8     for(int i=0;i<n;++i){
9         for(auto e:E){
10             tie(u,v,w)=e;
11             if(dp[i][u]!=inf)

```

```

12         dp[i+1][v]=min(dp[i+1][v],dp[i][u]+w);
13     }
14     double res = DBL_MAX;
15     for(int i=1;i<n;++i){
16         double val = DBL_MIN;
17         for(int j=0;j<n;++j)
18             val=max(val,double(dp[n][i]-dp[i][j]
19             ))/(n-j));
20         res=min(res,val);
21     }
22     return res;
23 }

```

5.8 Rectilinear_MST.cpp

```

1 //平面曼哈頓最小生成樹構造圖(去除非必要邊)
2 #define T int
3 #define INF 0x3f3f3f3f
4 struct point{
5     T x,y;
6     int id;//從0開始編號
7     point(){}
8     T dist(const point &p)const{
9         return abs(x-p.x)+abs(y-p.y);
10    }
11 };
12 bool cmpx(const point &a,const point &b){
13     return a.x<b.x||(a.x==b.x&&a.y<b.y);
14 }
15 struct edge{
16     int u,v;
17     T cost;
18     edge(int u,int v,T c):u(u),v(v),cost(c){}
19     bool operator<(const edge&e)const{
20         return cost<e.cost;
21     }
22 };
23 struct bit_node{
24     T mi;
25     int id;
26     bit_node(const T&mi=INF,int id=-1):mi(mi),
27         id(id){}
28 };
29 vector<bit_node> bit;
30 void bit_update(int i,const T&data,int id){
31     for(;i=i&(-i)){
32         if(data<bit[i].mi)bit[i]=bit_node(data,
33             id);
34     }
35 }
36 int bit_find(int i,int m){
37     bit_node x;
38     for(;i<m;i+=i&(-i)) if(bit[i].mi<x.mi)x=
39         bit[i];
40     return x.id;
41 }
42 vector<edge> build_graph(int n,point p[]){
43     vector<edge> e;//edge for MST
44     for(int dir=0;dir<4;++dir){//4種座標變換
45         if(dir%2) for(int i=0;i<n;++i) swap(p[i]
46             ].x,p[i].y);

```

```

47     else if(dir==2) for(int i=0;i<n;++i) p[i]
48         ].x=-p[i].x;
49     sort(p,p+n,cmpx);
50     vector<T> ga(n), gb;
51     for(int i=0;i<n;++i)ga[i]=p[i].y-p[i].x;
52     gb=ga, sort(gb.begin(),gb.end());
53     gb.erase(unique(gb.begin(),gb.end()),gb.
54         end());
55     int m=gb.size();
56     bit=vector<bit_node>(m+1);
57     for(int i=n-1;i>=0;--i){
58         int pos=lower_bound(gb.begin(),gb.end
59             (),ga[i])-gb.begin()+1;
60         int ans=bit_find(pos,m);
61         if(~ans)e.push_back(edge(p[i].id,p[ans]
62             ].id,p[i].dist(p[ans])));
63         bit_update(pos,p[i].x+p[i].y,i);
64     }
65     return e;
66 }

```

5.9 treeISO.cpp

```

1 const int MAXN=100005;
2 const long long X=12327,P=0xdefaced;
3 vector<int> g[MAXN];
4 bool vis[MAXN];
5 long long dfs(int u){//hash ver
6     vis[u]=1;
7     vector<long long> tmp;
8     for(auto v:g[u])if(!vis[v])tmp.pb(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 }

```

5.10 一般圖最小權完美匹配.cpp

```

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                    onstk[v] = 0;
34                }
35            }
36        }
37        onstk[u] = 0;
38        stk.pop_back();
39        return false;
40    }
41    int solve() {
42        // find a match
43        for (int i=0; i<n; i+=2){
44            match[i] = i+1, match[i+1] = i;
45        }
46        for(;;){
47            int found = 0;
48            for (int i=0; i<n; i++) dis[i] = onstk
49                [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_ba
56                            ck();
57                        int v = stk.back(); stk.pop_ba
58                            ck();
59                        match[u] = v;
60                        match[v] = u;
61                    }
62                }
63            }
64            if (!found) break;
65        }
66        int ret = 0;
67        for (int i=0; i<n; i++)
68            ret += edge[i][match[i]];
69        ret /= 2;
70        return ret;
71    }
72 }graph;

```


5.15 最小樹形圖 __ 朱劉.cpp

```

1 template<typename T>
2 struct zhu_liu{
3     static const int MAXN=110,MAXM=10005;
4     struct node{
5         int u,v;
6         T w,tag;
7         node *l,*r;
8         node(int u=0,int v=0,T w=0):u(u),v(v),w(
9             w),tag(0),l(0),r(0){}
10        void down(){
11            w+=tag;
12            if(l)l->tag+=tag;
13            if(r)r->tag+=tag;
14            tag=0;
15        }
16    }mem[MAXN];//靜態記憶體
17    node *pq[MAXN*2],*E[MAXN*2];
18    int st[MAXN*2],id[MAXN*2],m;
19    void init(int n){
20        for(int i=1;i<=n;++i){
21            pq[i]=E[i]=0, st[i]=id[i]=i;
22        }m=0;
23    }
24    node *merge(node *a,node *b){//skew heap
25        if(!a||!b)return a?a:b;
26        a->down(),b->down();
27        if(b->w<a->w)return merge(b,a);
28        swap(a->l,a->r);
29        a->l=merge(b,a->l);
30        return a;
31    }
32    void add_edge(int u,int v,T w){
33        if(u!=v)pq[v]=merge(pq[v],&(mem[m++]=
34            node(u,v,w)));
35    }
36    int find(int x,int *st){
37        return st[x]==x?x:st[x]=find(st[x],st);
38    }
39    T build(int root,int n){
40        T ans=0;int N=n,all=n;
41        for(int i=1;i<=N;++i){
42            if(i==root||!pq[i])continue;
43            while(pq[i]){
44                pq[i]->down(),E[i]=pq[i];
45                pq[i]=merge(pq[i]->l,pq[i]->r);
46                if(find(E[i]->u,id)!=find(i,id))
47                    break;
48            }
49            if(find(E[i]->u,id)==find(i,id))
50                continue;
51            ans+=E[i]->w;
52            if(find(E[i]->u,st)==find(i,st)){
53                if(pq[i])pq[i]->tag-=E[i]->w;
54                pq[++N]=pq[i];id[N]=N;
55                for(int u=find(E[i]->u,id);u!=i;u=
56                    find(E[u]->u,id)){
57                    if(pq[u])pq[u]->tag-=E[u]->w;
58                    id[find(u,id)]=N;
59                    pq[N]=merge(pq[N],pq[u]);

```

5.14 最小斯坦納樹 DP.cpp

```

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

```

```

55     }
56     st[N]=find(i,st);
57     id[find(i,id)]=N;
58     }else st[find(i,st)]=find(E[i]->u,st)
59     ,--all;
60 }
61 return all==1?ans:-INT_MAX;//圖不連通就
62 無解
63 }
64 }

```

5.16 穩定婚姻模板.cpp

```

1 queue<int> Q;
2 for ( i : 所有考生 ) {
3     設定在第0志願;
4     Q.push(考生i);
5 }
6 while(Q.size()){
7     當前考生=Q.front();Q.pop();
8     while ( 此考生未分發 ) {
9         指標移到下一志願;
10        if ( 已經沒有志願 or 超出志願總數 )
11            break;
12        計算該考生在該科系加權後的總分;
13        if ( 不符合科系需求 ) continue;
14        if ( 目前科系有餘額 ) {
15            依加權後分數高低順序將考生id加入科系錄
16            取名單中;
17            break;
18        }
19        if ( 目前科系已額滿 ) {
20            if ( 此考生成績比最低分數還高 ) {
21                依加權後分數高低順序將考生id加入科系
22                錄取名單;
23                Q.push(被踢出的考生);
24            }
25        }
26    }
27 }

```

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

```

```

9 map<char,int> rule;//每個字元對應到的規則·
10 小寫字母為終端字符
11 vector<CNF> cnf;
12 void init(){
13     state=0;
14     rule.clear();
15     cnf.clear();
16 }
17 void add_to_cnf(char s,const string &p,int
18 cost){
19     //加入一個s -> <p>的文法·代價為cost
20     if(rule.find(s)==rule.end())rule[s]=state
21     ++;
22     for(auto c:p)if(rule.find(c)==rule.end())
23     rule[c]=state++;
24     if(p.size()==1){
25         cnf.push_back(CNF(rule[s],rule[p[0]],-1,
26 cost));
27     }else{
28         int left=rule[s];
29         int sz=p.size();
30         for(int i=0;i<sz-2;++i){
31             cnf.push_back(CNF(left,rule[p[i]],
32 state,0));
33             left=state++;
34         }
35         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 void relax(int l,int r,const CNF &c,long
43 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 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]+c
56 .cost,k==n);
57 }
58 void cyk(const vector<int> &tok){
59     for(int i=0;i<(int)tok.size();++i){
60         for(int j=0;j<(int)tok.size();++j){
61             dp[i][j]=vector<long long>(state+1,
62 INT_MAX);
63             neg_INF[i][j]=vector<bool>(state+1,
64 false);
65         }
66         dp[i][i][tok[i]]=0;
67         bellman(i,i,tok.size());
68     }
69     for(int r=1;r<(int)tok.size();++r){
70         for(int l=r-1;l>=0;--l){
71             for(int k=1;k<r;++k)
72             for(auto c:cnf)

```

```

60     if(~c.y)relax(l,r,c,dp[l][k][c.x]+
61 dp[k+1][r][c.y]+c.cost);
62     bellman(l,r,tok.size());
63 }
64 }

```

7 Linear_Programming

7.1 最大密度子圖.cpp

```

1 typedef double T;//POJ 3155
2 const int MAXN=105;
3 struct edge{
4     int u,v;
5     T w;
6     edge(int u=0,int v=0,T w=0):u(u),v(v),w(w)
7     {}
8 }
9 vector<edge> E;
10 int n,m;// 1-base
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     for(int v=1;v<=n;++v){
34         isap.add_edge(s,v,U);
35         isap.add_edge(v,t,U+2*L-de[v]-2*pv[v]);
36     }
37 }
38 int main(){
39     while(~scanf("%d%d",&n,&m)){
40         if(!m){
41             puts("1\n1");
42             continue;
43         }
44         init();
45         int u,v;
46         for(int i=0;i<m;++i){
47             scanf("%d%d",&u,&v);
48             add_edge(u,v,1);
49         }

```

```

48 get_U();
49 s=n+1,t=n+2;
50 T l=0,r=U,k=1.0/(n*n);
51 while(r-l>k){//二分搜最大值
52     T mid=(l+r)/2;
53     build(mid);
54     T res=(U*n-isap.isap(s,t))/2;
55     if(res>0)l=mid;
56     else r=mid;
57 }
58 build(l);
59 isap.min_cut(s,t);
60 vector<int> ans;
61 for(int i=1;i<=n;++i)
62     if(isap.vis[i])ans.push_back(i);
63     printf("%d\n",ans.size());
64     for(size_t i=0;i<ans.size();++i)
65         printf("%d\n",ans[i]);
66 }
67 return 0;
68 }

```

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 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];

```

```

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

```

```

94 LL Tonelli_Shanks(const LL &n, const LL &p)
    {
95     // x^2 = n ( mod p )
96     if(n==0) return 0;
97     if(Legendre(n,p)!=1) while(1) { puts("SQRT
        ROOT does not exist"); }
98     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>=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     }
150     //如果m[i]是質數 · Euler(m[i])-1=m[i]-2 ·
        就不用算Euler了*/
151     return ans;
152 }
153 //java code

```

```

154 //求sqrt(N)的連分數
155 public static void Pell(int n){
156     BigInteger N,p1,p2,q1,q2,a0,a1,a2,g1,g2,h1
        ,h2,p,q;
157     g1=q2=p1=BigInteger.ZERO;
158     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 }

```

```

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=vector<
    complex<T> > >
2 struct FFT{
3     const T pi;
4     FFT(const T pi=acos((T)-1)):pi(pi){}
5     unsigned bit_reverse(unsigned a,int len){
6         a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)
            >>1);
7         a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)
            >>2);
8         a=((a&0xF0F0F0F0U)<<4)|((a&0xF0F0F0F0U)
            >>4);
9         a=((a&0xFF0FF0FFU)<<8)|((a&0xFF0FF0FFU)
            >>8);
10        a=((a&0x000FFFFFU)<<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=__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                complex<T> wi=exp(complex<T>(0,i*num
                    *pi/mh));
20                for(int j=i;j<N;j+=step){
21                    int k=j+mh;
22                    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    }

```

8.2 bit_set.cpp

```

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

```

8.3 cantor_expansion.cpp

```

1 int factorial[MAXN];
2 void init(){
3     factorial[0]=1;
4     for(int i=1;i<=MAXN;++i)factorial[i]=
        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;

```

30| };

8.6 FWT.cpp

8.5 find_real_root.cpp

```

1 //  $an^x + \dots + a_1x + a_0 = 0$ ;
2 int sign(double x){
3     return x < -eps ? -1 : x > eps;
4 }
5
6 double get(const vector<double>&coef, double
7     x){
8     double e = 1, s = 0;
9     for(auto i : coef) s += i*e, e *= x;
10    return s;
11 }
12
13 double find(const vector<double>&coef, int n
14     , double lo, double hi){
15     double sign_lo, sign_hi;
16     if( !(sign_lo = sign(get(coef,lo))) )
17         return lo;
18     if( !(sign_hi = sign(get(coef,hi))) )
19         return hi;
20     if(sign_lo * sign_hi > 0) return INF;
21     for(int stp = 0; stp < 100 && hi - lo >
22         eps; ++stp){
23         double m = (lo+hi)/2.0;
24         int sign_mid = sign(get(coef,m));
25         if(!sign_mid) return m;
26         if(sign_lo*sign_mid < 0) hi = m;
27         else lo = m;
28     }
29     return (lo+hi)/2.0;
30 }
31
32 vector<double> cal(vector<double>coef, int n
33 ){
34     vector<double>res;
35     if(n == 1){
36         if(sign(coef[1])) res.pb(-coef[0]/coef
37             [1]);
38         return res;
39     }
40     vector<double>dcoef(n);
41     for(int i = 0; i < n; ++i) dcoef[i] = coef
42         [i+1]*(i+1);
43     vector<double>droot = cal(dcoef, n-1);
44     droot.insert(droot.begin(), -INF);
45     droot.pb(INF);
46     for(int i = 0; i+1 < droot.size(); ++i){
47         double tmp = find(coef, n, droot[i],
48             droot[i+1]);
49         if(tmp < INF) res.pb(tmp);
50     }
51     return res;
52 }
53
54 int main () {
55     vector<double>ve;
56     vector<double>ans = cal(ve, n);
57     // 視情況把答案 +eps，避免 -0
58 }

```

```

1 vector<int> F_OR_T(vector<int> f, bool
2     inverse){
3     for(int i=0; (2<<i)<=f.size(); ++i)
4         for(int j=0; j<f.size(); j+=2<<i)
5             for(int k=0; k<(1<<i); ++k)
6                 f[j+k+(1<<i)] += f[j+k]*(inverse
7                     ?-1:1);
8     return f;
9 }
10 vector<int> rev(vector<int> A) {
11     for(int i=0; i<A.size(); i+=2)
12         swap(A[i],A[i^(A.size()-1)]);
13     return A;
14 }
15 vector<int> F_AND_T(vector<int> f, bool
16     inverse){
17     return rev(F_OR_T(rev(f), inverse));
18 }
19 vector<int> F_XOR_T(vector<int> f, bool
20     inverse){
21     for(int i=0; (2<<i)<=f.size(); ++i)
22         for(int j=0; j<f.size(); j+=2<<i)
23             for(int k=0; k<(1<<i); ++k){
24                 int u=f[j+k], v=f[j+k+(1<<i)];
25                 f[j+k+(1<<i)] = u-v, f[j+k] = u+v;
26             }
27     if(inverse) for(auto &a:f) a/=f.size();
28     return f;
29 }

```

8.7 LinearCongruence.cpp

```

1 pair<LL,LL> LinearCongruence(LL a[],LL b[],
2     LL m[],int n) {
3     //  $a[i]*x = b[i] \pmod{m[i]}$ 
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
14            (-1LL,0LL);
15        lastb = LLmul((lastb-b[i])/d,x,(lastm/d)
16            )*m[i];
17        lastm = (lastm/d)*m[i];
18        lastb = (lastb+b[i])%lastm;
19    }
20    return make_pair(lastb<0?lastb+lastm:lastb
21        ,lastm);
22 }

```

8.8 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 }

```

8.9 Matrix.cpp

```

1 template<typename T>
2 struct Matrix{
3     using rt = std::vector<T>;
4     using mt = std::vector<rt>;
5     using matrix = Matrix<T>;
6     int r,c;
7     mt m;
8     Matrix(int r,int c):r(r),c(c),m(r,rt(c)){}
9     rt& operator[](int i){return m[i];}
10    matrix operator+(const matrix &a){
11        matrix rev(r,c);
12        for(int i=0;i<r;++i)
13            for(int j=0;j<c;++j)
14                rev[i][j]=m[i][j]+a.m[i][j];
15        return rev;
16 }
17 matrix operator-(const matrix &a){
18     matrix rev(r,c);
19     for(int i=0;i<r;++i)
20         for(int j=0;j<c;++j)
21             rev[i][j]=m[i][j]-a.m[i][j];
22     return rev;
23 }
24 matrix operator*(const matrix &a){
25     matrix rev(r,a.c);
26     matrix tmp(a.c,a.r);
27     for(int i=0;i<a.r;++i)
28         for(int j=0;j<a.c;++j)
29             tmp[j][i]=a.m[i][j];
30     for(int i=0;i<r;++i)
31         for(int j=0;j<a.c;++j)
32             for(int k=0;k<c;++k)
33                 rev.m[i][j]+=m[i][k]*tmp[j][k];
34     return rev;
35 }
36 bool inverse(){
37     Matrix t(r,r+c);
38     for(int y=0;y<r;y++){
39         t.m[y][c+y] = 1;
40         for(int x=0;x<c;++x)
41             t.m[y][x]=m[y][x];
42     }
43     if( !t.gas() )
44         return false;

```

```

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

```

8.10 MillerRobin.cpp

```

1 LL LLmul(LL a, LL b, const LL &mod) {
2     LL ans=0;
3     while(b) {
4         if(b&1) {
5             ans+=a;
6             if(ans>=mod) ans-=mod;
7         }
8         a<<=1, b>>=1;
9         if(a>=mod) a-=mod;
10    }
11    return ans;
12 }
13 LL mod_mul(LL a,LL b,LL m){
14     a%=m,b%=m; /* fast for  $m < 2^{58}$  */
15     LL y=(LL)((double)a*b/m+.5);
16     LL r=(a*b-y*m)%m;
17     return r<0?r+m:r;
18 }
19 template<typename T>
20 T pow(T a,T b,T mod){/* $a^b \pmod{mod}$ 
21     T ans=1;
22     for(;b;a=mod_mul(a,a,mod),b>>=1)
23         if(b&1)ans=mod_mul(ans,a,mod);
24     return ans;
25 }
26 int sprp[3]={2,7,61};/*int範圍可解

```



```

27 int llsprp
   [7]={2,325,9375,28178,450775,9780504,
28 1795265022}; //至少 unsigned long long 範圍
29 template<typename T>
30 bool isprime(T n,int *sprp,int num){
31     if(n==2)return 1;
32     if(n<2||n%2==0)return 0;
33     int t=0;
34     T u=n-1;
35     for(;u%2==0;++t)u>>=1;
36     for(int i=0;i<num;++i){
37         T a=sprp[i]%n;
38         if(a==0||a==1||a==n-1)continue;
39         T x=pow(a,u,n);
40         if(x==1||x==n-1)continue;
41         for(int j=0;j<t;++j){
42             x=mod_mul(x,x,n);
43             if(x==1)return 0;
44             if(x==n-1)break;
45         }
46         if(x==n-1)continue;
47         return 0;
48     }
49     return 1;
50 }

```

8.11 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=vector<T>>
8 struct NTT{
9     const T P,G;
10     NTT(T p=(1<<23)*7*17+1,T g=3):P(p),G(g){}
11     unsigned bit_reverse(unsigned a,int len){
12         //Look FFT.cpp
13     }
14     T pow_mod(T n,T k,T m){
15         T ans=1;
16         for(n=(n>=m?n%m:n);k>>=1){
17             if(k&1)ans=ans*n%m;
18             n=n*n%m;
19         }
20         return ans;
21     }
22     void ntt(bool is_inv,VT &in,VT &out,int N)
23     {
24         int bitlen=__lg(N);
25         for(int i=0;i<N;++i)out[bit_reverse(i,
26             bitlen)]=in[i];
27         for(int step=2,id=1;step<=N;step<=1,++
28             id){
29             T wn=pow_mod(G,(P-1)>>id,P),wi=1,u,t;
30             const int mh=step>>1;
31             for(int i=0;i<mh;++i){
32                 for(int j=i;j<N;j+=step){
33                     u=out[j],t=wi*out[j+mh]%P;
34                     out[j]=u+t;
35                     out[j+mh]=u-t;

```

```

33         if(out[j]>=P)out[j]-=P;
34         if(out[j+mh]<0)out[j+mh]+=P;
35     }
36     wi=wi*wn%P;
37 }
38 }
39 if(is_inv){
40     for(int i=1;i<N/2;++i)swap(out[i],out[
41         N-i]);
42     T invn=pow_mod(N,P-2,P);
43     for(int i=0;i<N;++i)out[i]=out[i]*invn
44         %P;
45 }

```

8.12 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 }

```

8.13 外星模運算.cpp

```

1 //a[0]^(a[1]^a[2]^...)
2 #define maxn 1000000
3 int euler[maxn+5];
4 bool is_prime[maxn+5];
5 void init_euler(){
6     is_prime[1]=1; //一不是質數
7     for(int i=1;i<=maxn;i++)euler[i]=i;
8     for(int i=2;i<=maxn;i++){
9         if(!is_prime[i]){//是質數
10             euler[i]-=1;
11             for(int j=i<1;j<=maxn;j+=i){
12                 is_prime[j]=1;
13                 euler[j]=euler[j]/i*(i-1);
14             }
15         }
16     }
17 }
18 LL pow(LL a,LL b,LL mod){//a^b%mod
19     LL ans=1;
20     for(;b;a=a%mod,b>>=1)
21         if(b&1)ans=ans*a%mod;
22     return ans;
23 }

```

```

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

```

8.14 質因數分解.cpp

```

1 LL func(const LL n,const LL mod,const int c)
2 {
3     return (LLmul(n,n,mod)+c+mod)%mod;
4 }
5 LL pollorro(const LL n, const int c) { //循環
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     } else {
32         for(int i=0;i<primecnt&&prime[i]*prime[i]
33             <=n;++i) {
34             while(n%prime[i]==0) {
35                 v.push_back(prime[i]);
36                 n/=prime[i];
37             }
38             if(n!=1) v.push_back(n);
39         }
40     }
41 }
42 void comfactor(const LL &n, vector<LL> &v) {
43     if(n<1e9) {
44         smallfactor(n,v);
45         return;
46     }
47     if(Isprime(n)) {
48         v.push_back(n);
49         return;
50     }
51     LL d;
52     for(int c=3;++c) {
53         d = pollorro(n,c);
54         if(d!=n) break;
55     }
56     comfactor(d,v);
57     comfactor(n/d,v);
58 }
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 }
69 void AllFactor(const LL &n,vector<LL> &v) {
70     vector<LL> tmp;
71     Factor(n,tmp);
72     v.clear();
73     v.push_back(1);
74     int len;
75     LL now=1;
76     for(int i=0;i<tmp.size();++i) {
77         if(i==0 || tmp[i]!=tmp[i-1]) {
78             len = v.size();
79             now = 1;
80         }
81         now*=tmp[i];
82         for(int j=0;j<len;++j)
83             v.push_back(v[j]*now);
84     }
85 }
86 }

```

9 other

9.1 WhatDay.cpp

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

9.2 上下最大正方形.cpp

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

9.3 最大矩形.cpp

```
1 LL max_rectangle(vector<int> s){
2     stack<pair<int,int> > st;
3     st.push(make_pair(-1,0));
4     s.push_back(0);
5     LL ans=0;
6     for(size_t i=0;i<s.size();++i){
7         int h=s[i];
8         pair<int,int> now=make_pair(h,i);
9         while(h<st.top().first){
10            now=st.top();
11            st.pop();
12            ans=max(ans,(LL)(i-now.second)*now.
13                first);
14        }
15        if(h>st.top().first){
```

```
15         st.push(make_pair(h,now.second));
16     }
17 }
18 return ans;
19 }
```

10 String

10.1 AC 自動機.cpp

```
1 template<char L='a',char R='z'>
2 class ac_automaton{
3     struct joe{
4         int next[R-L+1],fail,efl,ed,cnt_dp,vis;
5         joe():ed(0),cnt_dp(0),vis(0){
6             for(int i=0;i<=R-L;++i)next[i]=0;
7         }
8     };
9     public:
10        std::vector<joe> S;
11        std::vector<int> q;
12        int qs,qe,vt;
13        ac_automaton():S(1),qs(0),qe(0),vt(0){
14            void clear(){
15                q.clear();
16                S.resize(1);
17                for(int i=0;i<=R-L;++i)S[0].next[i]=0;
18                S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
19            }
20            void insert(const char *s){
21                int o=0;
22                for(int i=0,id;s[i];++i){
23                    id=s[i]-L;
24                    if(!S[o].next[id]){
25                        S.push_back(joe());
26                        S[o].next[id]=S.size()-1;
27                    }
28                    o=S[o].next[id];
29                }
30                ++S[o].ed;
31            }
32            void build_fail(){
33                S[0].fail=S[0].efl=-1;
34                q.clear();
35                q.push_back(0);
36                ++qe;
37                while(qs!=qe){
38                    int pa=q[qs++],id,t;
39                    for(int i=0;i<=R-L;++i){
40                        t=S[pa].next[i];
41                        if(!t)continue;
42                        id=S[pa].fail;
43                        while(~id&&!S[id].next[i])id=S[id].
44                            fail;
45                        S[t].fail=~id?S[id].next[i]:0;
46                        S[t].efl=S[t].fail!.ed?S[t].fail:S
47                            [S[t].fail].efl;
48                        q.push_back(t);
49                    }
50                }
51            }
```

```
50 }
51 /*DP出每個前綴在字串s出現的次數並傳回所有
52 字串被s匹配成功的次數O(N*M)*/
53 int match_0(const char *s){
54     int ans=0,id,p=0,i;
55     for(i=0;s[i];++i){
56         id=s[i]-L;
57         while(!S[p].next[id]&&p)p=S[p].fail;
58         if(!S[p].next[id])continue;
59         p=S[p].next[id];
60         ++S[p].cnt_dp; /*匹配成功則它所有後綴都
61             可以被匹配(DP計算)*/
62     }
63     for(i=qe-1;i>=0;--i){
64         ans+=S[q[i]].cnt_dp*S[q[i]].ed;
65         if(~S[q[i]].fail)S[S[q[i]].fail].
66             cnt_dp+=S[q[i]].cnt_dp;
67     }
68     return ans;
69 }
70 /*多串匹配走efl邊並傳回所有字串被s匹配成功
71 的次數O(N*M^1.5)*/
72 int match_1(const char *s)const{
73     int ans=0,id,p=0,t;
74     for(int i=0;s[i];++i){
75         id=s[i]-L;
76         while(!S[p].next[id]&&p)p=S[p].fail;
77         if(!S[p].next[id])continue;
78         p=S[p].next[id];
79         if(S[p].ed)ans+=S[p].ed;
80         for(t=S[p].efl;~t;t=S[t].efl){
81             ans+=S[t].ed; /*因為都走efl邊所以保證
82                 匹配成功*/
83         }
84     }
85     return ans;
86 }
87 /*枚舉(s的子字串nA)的所有相異字串各恰一次
88 並傳回次數O(N*M^(1/3))*/
89 int match_2(const char *s){
90     int ans=0,id,p=0,t;
91     ++vt;
92     /*把戳記vt+=1,只要vt沒溢位,所有S[p].
93     vis==vt就會變成false
94     這種利用vt的方法可以O(1)歸零vis陣列*/
95     for(int i=0;s[i];++i){
96         id=s[i]-L;
97         while(!S[p].next[id]&&p)p=S[p].fail;
98         if(!S[p].next[id])continue;
99         p=S[p].next[id];
100         if(S[p].ed&&S[p].vis!=vt){
101             S[p].vis=vt;
102             ans+=S[p].ed;
103         }
104         for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t].
105             efl){
106             S[t].vis=vt;
107             ans+=S[t].ed; /*因為都走efl邊所以保證
108                 匹配成功*/
109         }
110     }
111     return ans;
112 }
```

/*把AC自動機變成真的自動機*/

```
104 void evolution(){
105     for(qs=1;qs!=qe;){
106         int p=q[qs++];
107         for(int i=0;i<=R-L;++i)
108             if(S[p].next[i]==0)S[p].next[i]=S[S[
109                 p].fail].next[i];
110     }
111 }
112 ;
```

10.2 hash.cpp

```
1 #define MAXN 1000000
2 #define mod 1073676287
3 /*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)%mod*/
8 void hash_init(int len,T prime){
9     h_base[0]=1;
10    for(int i=1;i<=len;++i){
11        h[i]=(h[i-1]*prime+s[i-1])%mod;
12        h_base[i]=(h_base[i-1]*prime)%mod;
13    }
14 }
15 T get_hash(int l,int r){/*閉區間寫法,設編號
16    為0 ~ Len-1*/
17    return (h[r+1]-(h[l]*h_base[r-l+1])%mod+
18        mod)%mod;
19 }
```

10.3 KMP.cpp

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

10.4 manacher.cpp

```

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

```

10.5 minimal_string_rotation.cpp

```

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

```

10.6 reverseBWT.cpp

```

1 const int MAXN = 305, MAXC = 'Z';
2 int ranks[MAXN], tots[MAXC], first[MAXC];
3 void rankBWT(const string &bw){
4     memset(ranks, 0, sizeof(int)*bw.size());
5     memset(tots, 0, sizeof(int)*MAXC);
6     for(size_t i=0; i<bw.size(); ++i)
7         ranks[i] = tots[ bw[i] ]++;
8 }
9 void firstCol(){
10    memset(first, 0, sizeof(int)*MAXC);
11    int totc = 0;
12    for(int c='A'; c<='Z'; ++c){
13        if(!tots[c]) continue;
14        first[c] = totc;
15        totc += tots[c];
16    }
17 }
18 string reverseBwt(string bw, int begin){
19     rankBWT(bw, firstCol());
20     int i = begin; //原字串最後一個元素的位置
21     string res;
22     do{
23         char c = bw[i];
24         res = c + res;
25         i = first[ bw[i] ] + ranks[i];
26     } while( i != begin );

```

```

27     return res;
28 }

```

10.7 suffix_array_lcp.cpp

```

1 #define radix_sort(x, y){\
2     for(i=0; i<A; ++i)c[i]=0;\
3     for(i=0; i<n; ++i)c[x[y[i]]]++;\
4     for(i=1; i<A; ++i)c[i]+=c[i-1];\
5     for(i=n-1; ~i; --i)sa[--c[x[y[i]]]]=y[i];\
6 }
7 #define AC(r, a, b)\
8     r[a]!=r[b]||a+k>n||r[a+k]!=r[b+k]
9 void suffix_array(const char *s, int n, int *
10     sa, int *rank, int *tmp, int *c){
11     int A='Z'+1, i, k, id=0;
12     for(i=0; i<n; ++i)rank[i]=s[i];
13     radix_sort(rank, tmp);
14     for(k=1; id<n-1; k<=1){
15         for(id=0, i=n-k; i<n; ++i)tmp[id++]=i;
16         for(i=0; i<n; ++i)
17             if(sa[i]>k)tmp[id++] = sa[i]-k;
18         radix_sort(rank, tmp);
19         swap(rank, tmp);
20         for(rank[sa[0]]=id=0, i=1; i<n; ++i)
21             rank[sa[i]] = id++ + AC(tmp, sa[i-1], sa[i]);
22         A=id+1;
23     }
24     //h: 高度數組 sa: 後綴數組 rank: 排名
25 void suffix_array_lcp(const char *s, int len,
26     int *h, int *sa, int *rank){
27     for(int i=0; i<len; ++i)rank[sa[i]]=i;
28     for(int i=0, k=0; i<len; ++i){
29         if(rank[i]==0)continue;
30         if(k--<0)
31             while(s[i+k]==s[sa[rank[i]-1]+k])++k;
32         h[rank[i]]=k;
33     }
34 }

```

10.8 Z.cpp

```

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

```

11 Tarjan

```

56     }
57 }
58 }dom;

```

11.1 dominator_tree.cpp

```

1 struct dominator_tree{
2     static const int MAXN=5005;
3     int n; // 1-base
4     vector<int> suc[MAXN], pre[MAXN];
5     int fa[MAXN], dfn[MAXN], id[MAXN], Time;
6     int semi[MAXN], idom[MAXN];
7     int anc[MAXN], best[MAXN]; //disjoint set
8     vector<int> dom[MAXN]; //dominator_tree
9     void init(int _n){
10         n=_n;
11         for(int i=1; i<=n; ++i)suc[i].clear(), pre[i].clear();
12     }
13     void add_edge(int u, int v){
14         suc[u].push_back(v);
15         pre[v].push_back(u);
16     }
17     void dfs(int u){
18         dfn[u]++Time, id[Time]=u;
19         for(auto v:suc[u]){
20             if(dfn[v])continue;
21             dfs(v), fa[dfn[v]]=dfn[u];
22         }
23     }
24     int find(int x){
25         if(x==anc[x])return x;
26         int y=find(anc[x]);
27         if(semi[best[x]]>semi[best[anc[x]]])best[x]=best[anc[x]];
28         return anc[x]=y;
29     }
30     void tarjan(int r){
31         Time=0;
32         for(int t=1; t<=n; ++t){
33             dfn[t]=idom[t]=0; //u=r或是u無法到達r時
34             idom[id[u]]=0
35             dom[t].clear();
36             anc[t]=best[t]=semi[t]=t;
37         }
38         dfs(r);
39         for(int y=Time; y>=2; --y){
40             int x=fa[y], idy=id[y];
41             for(auto z:pre[idy]){
42                 if(!z=dfn[z])continue;
43                 find(z);
44                 semi[y]=min(semi[y], semi[best[z]]);
45             }
46             dom[semi[y]].push_back(y);
47             anc[y]=x;
48             for(auto z:dom[x]){
49                 find(z);
50                 idom[z]=semi[best[z]]<x?best[z]:x;
51             }
52             dom[x].clear();
53         }
54         for(int u=2; u<=Time; ++u){
55             if(idom[u]!=semi[u])idom[u]=idom[idom[u]];
56             dom[id[idom[u]]].push_back(id[u]);
57         }

```

11.2 tnfshb017_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], rv[MAXN2], vis_t;
7 int N, M;
8 void addedge(int s, int e){
9     v[s].push_back(e);
10    rv[e].push_back(s);
11 }
12 int scc[MAXN2];
13 bool vis[MAXN2]={false};
14 void dfs(vector<int> *uv, int n, int k=-1){
15     vis[n]=true;
16     for(int i=0; i<uv[n].size(); ++i)
17         if(!vis[uv[n][i]])
18             dfs(uv, uv[n][i], k);
19     if(uv==v)vis_t.push_back(n);
20     scc[n]=k;
21 }
22 void solve(){
23     for(int i=1; i<=N; ++i){
24         if(!vis[i])dfs(v, i);
25         if(!vis[n(i)])dfs(v, n(i));
26     }
27     memset(vis, 0, sizeof(vis));
28     int c=0;
29     for(int i=vis_t.size()-1; i>=0; --i)
30         if(!vis[vis_t[i]])
31             dfs(rv, vis_t[i], c++);
32 }
33 int main(){
34     int a, b;
35     scanf("%d%d", &N, &M);
36     for(int i=1; i<=N; ++i){
37         // (A or B) & (!A & !B) A^B
38         a=i*2-1;
39         b=i*2;
40         addedge(n(a), b);
41         addedge(n(b), a);
42         addedge(a, n(b));
43         addedge(b, n(a));
44     }
45     while(M--){
46         scanf("%d%d", &a, &b);
47         a = a>0?a*2-1:-a*2;
48         b = b>0?b*2-1:-b*2;
49         // A or B
50         addedge(n(a), b);
51         addedge(n(b), a);
52     }
53     solve();
54     bool check=true;
55     for(int i=1; i<=2*N; ++i)
56         if(scc[i]==scc[n(i)])
57             check=false;

```

```

58 if(check){
59     printf("%d\n",N);
60     for(int i=1;i<=2*N;i+=2){
61         if(scc[i]>scc[i+2*N]) putchar('+');
62         else putchar('-');
63     }
64     puts("");
65 }else puts("0");
66 return 0;
67 }

```

11.3 橋連通分量.cpp

```

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

```

```

47 }

```

11.4 雙連通分量 & 割點.cpp

```

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

```

12 Tree_problem

12.1 HeavyLight.cpp

```

1 #include<vector>
2 #define MAXN 100005
3 int siz[MAXN],max_son[MAXN],pa[MAXN],dep[
    MAXN];
4 int link_top[MAXN],link[MAXN],cnt;
5 vector<int> G[MAXN];
6 void find_max_son(int u){
7     siz[u]=1;
8     max_son[u]=-1;

```

```

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

```

12.2 LCA.cpp

```

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

```

```

23     for(int i=MAX_LOG;i>=0;--i){
24         if(pa[i][a]!=pa[i][b]){
25             a=pa[i][a];
26             b=pa[i][b];
27         }
28     }
29     return pa[0][a];
30 }

```

12.3 link_cut_tree.cpp

```

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

```



```

46 int access(int x){
47     int last=0;
48     while(x){
49         splay(x);
50         nd[x].ch[1]=last;
51         up(x);
52         last=x;
53         x=nd[x].pa;
54     }
55     return last; //access後splay tree的根
56 }
57 void access(int x, bool is=0) { //is=0就是一般
    的access
58     int last=0;
59     while(x){
60         splay(x);
61         if(is&&!nd[x].pa){
62             //printf("%d\n", max(nd[last].ma, nd[nd[
                x].ch[1]].ma));
63         }
64         nd[x].ch[1]=last;
65         up(x);
66         last=x;
67         x=nd[x].pa;
68     }
69 }
70 void query_edge(int u, int v){
71     access(u);
72     access(v, 1);
73 }
74 void make_root(int x){
75     access(x), splay(x);
76     nd[x].rev ^= 1;
77 }
78 void make_root(int x){
79     nd[access(x)].rev ^= 1;
80     splay(x);
81 }
82 void cut(int x, int y){
83     make_root(x);
84     access(y);
85     splay(y);
86     nd[y].ch[0]=0;
87     nd[x].pa=0;
88 }
89 void cut_parents(int x){
90     access(x);
91     splay(x);
92     nd[nd[x].ch[0]].pa=0;
93     nd[x].ch[0]=0;
94 }
95 void link(int x, int y){
96     make_root(x);
97     nd[x].pa=y;
98 }
99 int find_root(int x){
100     x=access(x);
101     while(nd[x].ch[0]) x=nd[x].ch[0];
102     splay(x);
103     return x;
104 }
105 int query(int u, int v){
106     //傳回uv路徑splay tree的根結點
107     //這種寫法無法求LCA
108     make_root(u);
109     return access(v);
110 }
111 int query_lca(int u, int v){
112     //假設求鏈上點權的總和，sum是子樹的權重和，
    data是節點的權重
113     access(u);
114     int lca=access(v);
115     splay(u);
116     if(u==lca){
117         //return nd[lca].data+nd[nd[lca].ch[1]].
            sum
118     } else {
119         //return nd[lca].data+nd[nd[lca].ch[1]].
            sum+nd[u].sum
120     }
121 }
122 struct EDGE{
123     int a, b, w;
124 } e[10005];
125 int n;
126 vector<pair<int, int>> G[10005];
127 //first表示子節點，second表示邊的編號
128 int pa[10005], edge_node[10005];
129 //pa是父母節點，暫存用的，edge_node是每個編
    被存在哪個點裡面的陣列
130 void bfs(int root){
131     //在建構的時候把每個點都設成一個splay tree
132     queue<int> q;
133     for(int i=1; i<=n; ++i) pa[i]=0;
134     q.push(root);
135     while(q.size()){
136         int u=q.front();
137         q.pop();
138         for(auto P:G[u]){
139             int v=P.first;
140             if(v!=pa[u]){
141                 pa[v]=u;
142                 nd[v].pa=u;
143                 nd[v].data=e[P.second].w;
144                 edge_node[P.second]=v;
145                 up(v);
146                 q.push(v);
147             }
148         }
149     }
150 }
151 void change(int x, int b){
152     splay(x);
153     //nd[x].data=b;
154     up(x);
155 }

```

12.4 POJ_tree.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define MAXN 10005
4 int n, k;
5 vector<pair<int, int>> g[MAXN];
6 int size[MAXN];
7 bool vis[MAXN];
8 inline void init(){

```

```

9     for(int i=0; i<=n; ++i){
10         g[i].clear();
11         vis[i]=0;
12     }
13 }
14 void get_dis(vector<int> &dis, int u, int pa,
    int d){
15     dis.push_back(d);
16     for(size_t i=0; i<g[u].size(); ++i){
17         int v=g[u][i].first, w=g[u][i].second;
18         if(v!=pa&&!vis[v]) get_dis(dis, v, u, d+w);
19     }
20 }
21 vector<int> dis; //這東西如果放在函數裡會TLE
22 int cal(int u, int d){
23     dis.clear();
24     get_dis(dis, u, -1, d);
25     sort(dis.begin(), dis.end());
26     int l=0, r=dis.size()-1, res=0;
27     while(l<r){
28         while(l<r&&dis[l]+dis[r]>k) --r;
29         res+=r-(l++);
30     }
31     return res;
32 }
33 pair<int, int> tree_centroid(int u, int pa,
    const int sz){
34     size[u]=1; //找樹重心，second是重心
35     pair<int, int> res(INT_MAX, -1);
36     int ma=0;
37     for(size_t i=0; i<g[u].size(); ++i){
38         int v=g[u][i].first;
39         if(v==pa||vis[v]) continue;
40         res=min(res, tree_centroid(v, u, sz));
41         size[u]+=size[v];
42         ma=max(ma, size[v]);
43     }
44     ma=max(ma, sz-size[u]);
45     return min(res, make_pair(ma, u));
46 }
47 int tree_DC(int u, int sz){
48     int center=tree_centroid(u, -1, sz).second;
49     int ans=cal(center, 0);
50     vis[center]=1;
51     for(size_t i=0; i<g[center].size(); ++i){
52         int v=g[center][i].first, w=g[center][i].
            second;
53         if(vis[v]) continue;
54         ans+=cal(v, w);
55         ans+=tree_DC(v, size[v]);
56     }
57     return ans;
58 }
59 int main(){
60     while(scanf("%d%d", &n, &k), n||k){
61         init();
62         for(int i=1; i<=n; ++i){
63             int u, v, w;
64             scanf("%d%d%d", &u, &v, &w);
65             g[u].push_back(make_pair(v, w));
66             g[v].push_back(make_pair(u, w));
67         }
68         printf("%d\n", tree_DC(1, n));
69     }
70     return 0;
71 }

```

13 zformula

13.1 formula.tex

13.1.1 Pick 公式

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

13.1.2 圖論

1. $V - E + F = 2$
2. 對於平面圖， $F = E - V + n + 1$ ， n 是連通分量
3. 對於平面圖， $E < 3V - 6$
4. 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：

$$\begin{aligned} (a) \quad & I(G) + C_v(G) = |V| \\ (b) \quad & M(G) + C_e(G) = |V| \end{aligned}$$

5. 對於連通二分圖：

$$\begin{aligned} (a) \quad & I(G) = C_v(G) \\ (b) \quad & M(G) = C_e(G) \end{aligned}$$

6. 最大權閉合圖：

$$\begin{aligned} (a) \quad & C(u, V) = \infty, (u, v) \in E \\ (b) \quad & C(S, v) = W_v, W_v > 0 \\ (c) \quad & C(v, T) = -W_v, W_v < 0 \end{aligned}$$

7. 最大密度子圖：

$$\begin{aligned} (a) \quad & C(u, v) = 1, (u, v) \in E \\ (b) \quad & C(S, v) = U_v, v \in V \\ (c) \quad & C(v, T) = U + 2g - d_v, v \in V \end{aligned}$$

8. 弦圖：

- (a) 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
- (b) 最大團大小 = 色數
- (c) 最大獨立集：完美消除序列從前往後能選就選
- (d) 最小點覆蓋：最大獨立集的點和他延伸的邊構成
- (e) 區間圖是弦圖
- (f) 區間圖的完美消除序列：將區間按造又端點由小到大排序
- (g) 區間圖染色：用線段樹做

```

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

```

13.1.3 學長公式

- $\sum_{d|n} \phi(n) = n$
- $g(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d) \times g(n/d)$
- Harmonic series $H_n = \ln(n) + \gamma + 1/(2n) - 1/(12n^2) + 1/(120n^4)$
- $\gamma = 0.57721566490153286060651209008240243104215$
- 格雷碼 $= n \oplus (n >> 1)$
- $SG(A+B) = SG(A) \oplus SG(B)$
- 選轉矩陣 $M(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

13.1.4 基本數論

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

13.1.5 排組公式

- k 卡特蘭 $\frac{C_n^{kn}}{n(k-1)+1} \cdot C_m^n = \frac{n!}{m!(n-m)!}$
- $H(n, m) \cong x_1 + x_2 \dots + x_n = k, num = C_k^{n+k-1}$
- Stirling number of 2^{nd} , n 人分 k 組方法數目
 - $S(0, 0) = S(n, n) = 1$
 - $S(n, 0) = 0$
 - $S(n, k) = kS(n-1, k) + S(n-1, k-1)$
- Bell number, n 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 $B_0 : 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975$
- 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 $D_0 : 1, 0, 1, 2, 9, 44, 265, 1854, 14833, 133496$
- Binomial Equality
 - $\sum_k \binom{r}{m+k} \binom{s}{n-k} = \binom{r+s}{m+n}$
 - $\sum_k \binom{l}{m+k} \binom{s}{n-k} = \binom{l+s}{l-m+n}$
 - $\sum_k \binom{l}{m+k} \binom{s+k}{n} (-1)^k = (-1)^{l+m} \binom{s-m}{n-l}$
 - $\sum_{k \leq l} \binom{l-k}{m} \binom{s}{k-n} (-1)^k = (-1)^{l+m} \binom{s-m-1}{l-n-m}$
 - $\sum_{0 \leq k \leq l} \binom{l-k}{m} \binom{q+k}{n} = \binom{l+q+1}{m+n+1}$
 - $\binom{r}{k} = (-1)^k \binom{k-r-1}{k}$

- $\binom{r}{m} \binom{m}{k} = \binom{r}{m-k} \binom{r-k}{m-k}$
- $\sum_{k \leq n} \binom{r+k}{k} = \binom{r+n+1}{n}$
- $\sum_{0 \leq k \leq n} \binom{k}{m} = \binom{n+1}{m+1}$
- $\sum_{k \leq m} \binom{m+r}{k} x^k y^{m-k} = \sum_{k \leq m} \binom{-r}{k} (-x)^k (x+y)^{m-k}$

13.1.6 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
- $B_2 = 1/6, B_4 = -1/30, B_6 = 1/42, B_8 = -1/30, B_{10} = 5/66, B_{12} = -691/2730, B_{14} = 7/6, B_{16} = -3617/510, B_{18} = 43867/798, B_{20} = -174611/330,$

13.1.7 Burnside's lemma

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

13.1.8 Count on a tree

- Rooted tree: $s_{n+1} = \frac{1}{n} \sum_{i=1}^n (i \times a_i \times \sum_{j=1}^{\lfloor n/i \rfloor} a_{n+1-i \times j})$
- Unrooted tree:
 - Odd: $a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$
 - Even: $Odd + \frac{1}{2} a_{n/2} (a_{n/2} + 1)$
- Spanning Tree
 - 完全圖 $n^n - 2$
 - 一般圖 (Kirchhoff's theorem) $M[i][i] = \text{degree}(V_i), M[i][j] = -1, \text{if have } E(i, j), 0 \text{ if no edge. delete any one row and col in } A, \text{ans} = \det(A)$

13.2 java.tex

13.2.1 文件操作

```

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

```

13.2.2 优先队列

```

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

```

13.2.3 Map

```

1 Map map = new HashMap();
2 map.put("sa", "dd");
3 String str = map.get("sa").toString();
4
5 for(Object obj : map.keySet()){
6     Object value = map.get(obj);
7 }

```

13.2.4 sort

```

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

```

ACM ICPC TEAM REFERENCE - MADE IN ABYSS

Contents

1 Computational_Geometry	1	3.2 ext.cpp	5	7 Linear_Programming	10	10.7 suffix_array_lcp.cpp	15
1.1 Geometry.cpp	1	3.3 IncStack.cpp	5	7.1 最大密度子圖.cpp	10	10.8 Z.cpp	15
1.2 SmallestCircle.cpp	3	3.4 input.cpp	6	8 Number_Theory	10	11 Tarjan	15
1.3 最近點對.cpp	3	4 Flow	6	8.1 basic.cpp	10	11.1 dominator_tree.cpp	15
2 Data_Structure	3	4.1 dinic.cpp	6	8.2 bit_set.cpp	11	11.2 tnfsb017_2_sat.cpp	15
2.1 DLX.cpp	3	4.2 ISAP_with_cut.cpp	6	8.3 cantor_expansion.cpp	11	11.3 橋連通分量.cpp	16
2.2 Dynamic_KD_tree.cpp	4	4.3 MinCostMaxFlow.cpp	6	8.4 FFT.cpp	11	11.4 雙連通分量 & 割點.cpp	16
2.3 kd_tree_replace_segment_tree.cpp	4	5 Graph	7	8.5 find_real_root.cpp	12	12 Tree_problem	16
2.4 reference_point.cpp	5	5.1 Augmenting_Path.cpp	7	8.6 FWT.cpp	12	12.1 HeavyLight.cpp	16
2.5 skew_heap.cpp	5	5.2 Augmenting_Path_multiple.cpp	7	8.7 LinearCongruence.cpp	12	12.2 LCA.cpp	16
2.6 undo_disjoint_set.cpp	5	5.3 blossom_matching.cpp	7	8.8 Lucas.cpp	12	12.3 link_cut_tree.cpp	16
2.7 整體二分.cpp	5	5.4 graphISO.cpp	7	8.9 Matrix.cpp	12	12.4 POJ_tree.cpp	17
3 default	5	5.5 KM.cpp	7	8.10 MillerRobin.cpp	12	13 zformula	17
3.1 debug.cpp	5	5.6 MaximumClique.cpp	8	8.11 NTT.cpp	13	13.1 formula.tex	17
		5.7 MinimumMeanCycle.cpp	8	8.12 Simpson.cpp	13	13.1.1 Pick 公式	17
		5.8 Rectilinear_MST.cpp	8	8.13 外星模運算.cpp	13	13.1.2 圖論	17
		5.9 treeISO.cpp	8	8.14 質因數分解.cpp	13	13.1.3 學長公式	18
		5.10 一般圖最小權完美匹配.cpp	8	9 other	14	13.1.4 基本數論	18
		5.11 全局最小割.cpp	9	9.1 WhatDay.cpp	14	13.1.5 排組公式	18
		5.12 平面圖判定.cpp	9	9.2 上下最大正方形.cpp	14	13.1.6 冪次, 冪次和	18
		5.13 弦圖完美消除序列.cpp	9	9.3 最大矩形.cpp	14	13.1.7 Burnside's lemma	18
		5.14 最小斯坦納樹 DP.cpp	9	10 String	14	13.1.8 Count on a tree	18
		5.15 最小樹形圖 _ 朱劉.cpp	9	10.1 AC 自動機.cpp	14	13.2 java.tex	18
		5.16 穩定婚姻模板.cpp	10	10.2 hash.cpp	14	13.2.1 文件操作	18
		6 language	10	10.3 KMP.cpp	14	13.2.2 優先队列	18
		6.1 CNF.cpp	10	10.4 manacher.cpp	15	13.2.3 Map	18
				10.5 minimal_string_rotation.cpp	15	13.2.4 sort	18
				10.6 reverseBWT.cpp	15		