

1 Computational_Geometry

1.1 Geometry

```

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
39         (x),p2(y){ }
40     void pton(){ //轉成一般式
41         a=p1.y-p2.y;
42         b=p2.x-p1.x;
43         c=-a*p1.x-b*p1.y;
44     }
45     T ori(const point<T> &p)const{ //點和有向直
46         線的關係 · >0左邊 · =0在線上<0右邊
47         return (p2-p1).cross(p-p1);
48     }
49     T btw(const point<T> &p)const{ //點投影落在
50         線段上<=0
51         return (p1-p).dot(p2-p);
52     }
53     bool point_on_segment(const point<T>&p)
54         const{ //點是否在線段上
55         return ori(p)==0&&btw(p)<=0;
56     }
57     T dis2(const point<T> &p,bool is_segment
58         =0)const{ //點跟直線/線段的距離平方
59     point<T> v=p2-p1,v1=p-p1;
60     if(is_segment){
61         point<T> v2=p-p2;
62         if(v.dot(v1)<=0)return v1.abs2();
63         if(v.dot(v2)>=0)return v2.abs2();
64     }
65     T tmp=v.cross(v1);
66     return tmp*tmp/v.abs2();
67 }
68 T seg_dis2(const line<T> &l)const{ //兩線段
69     距離平方
70     return min({dis2(l.p1,1),dis2(l.p2,1),l.
71         dis2(p1,1),l.dis2(p2,1)});
72 }
73 point<T> projection(const point<T> &p)
74     const{ //點對直線的投影
75     point<T> n=(p2-p1).normal();
76     return p-n*(p-p1).dot(n)/n.abs2();
77 }
78 point<T> mirror(const point<T> &p)const{
79     //點對直線的鏡射 · 要先呼叫pton轉成一般式
80     point<T> R;
81     T d=a*a+b*b;
82     R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
83     R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
84     return R;
85 }
86 bool equal(const line &l)const{ //直線相等
87     return ori(l.p1)==0&&ori(l.p2)==0;
88 }
89 bool parallel(const line &l)const{
90     return (p1-p2).cross(l.p1-l.p2)==0;
91 }
92 bool cross_seg(const line &l)const{
93     return (p2-p1).cross(l.p1-p1)*(p2-p1).
94         cross(l.p2-p1)<=0; //直線是否交線段
95 }
96 int line_intersect(const line &l)const{ //
97     直線相交情況 · -1無限多點 · 1交於一點 · 0
98     不相交
99     return parallel(l)?(ori(l.p1)==0?-1:0)
100         :1;
101 }
102 int seg_intersect(const line &l)const{
103     T c1=ori(l.p1), c2=ori(l.p2);
104     T c3=l.ori(p1), c4=l.ori(p2);
105     if(c1==0&&c2==0){ //共線
106         bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
107         T a3=l.btw(p1),a4=l.btw(p2);
108         if(b1&&b2&&a3==0&&a4==0) return 2;
109         if(b1&&b2&&a3>0&&a4==0) return 3;
110         if(b1&&b2&&a3>0&&a4>0) return 0;
111         return -1; //無限交點
112     }else if(c1*c2<0&&c3*c4<0)return 1;
113     return 0; //不相交
114 }
115 point<T> line_intersection(const line &l)
116     const{ //直線交點*/
117     point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
118     //if(a.cross(b)==0)return INF;
119     return p1+a*(s.cross(b)/a.cross(b));
120 }
121 point<T> seg_intersection(const line &l)
122     const{ //線段交點
123     int res=seg_intersect(l);
124     if(res<=0) assert(0);
125     if(res==2) return p1;
126     if(res==3) return p2;
127     return line_intersection(l);
128 }
129 template<typename T>
130 struct polygon{
131     polygon(){ }
132     vector<point<T> > p; //逆時針順序
133     T area()const{ //面積
134     T ans=0;
135     for(int i=p.size()-1,j=0;j<(int)p.size()
136         ;i=j++)
137         ans+=p[i].cross(p[j]);
138     return ans/2;
139 }
140 point<T> center_of_mass()const{ //重心
141     T cx=0,cy=0,w=0;
142     for(int i=p.size()-1,j=0;j<(int)p.size()
143         ;i=j++){
144         T a=p[i].cross(p[j]);
145         cx+=(p[i].x+p[j].x)*a;
146         cy+=(p[i].y+p[j].y)*a;
147         w+=a;
148     }
149     return point<T>(cx/3/w,cy/3/w);
150 }
151 char ahas(const point<T>&t)const{ //點是否
152     在簡單多邊形內 · 是的話回傳1 · 在邊上回
153     傳-1 · 否則回傳0
154     bool c=0;
155     for(int i=0,j=p.size()-1;i<p.size();j=i
156         ++){
157         if(line<T>(p[i],p[j]).point_on_segment
158             (t))return -1;
159         else if((p[i].y>t.y)!(p[j].y>t.y)&&
160             t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j]
161                 .y-p[i].y)+p[i].x)
162             c=!c;
163     }
164     return c;
165 }
166 char point_in_convex(const point<T>&x)
167     const{
168     int l=1,r=(int)p.size()-2;
169     while(l<r){ //點是否在凸多邊形內 · 是的話
170         回傳1 · 在邊上回傳-1 · 否則回傳0
171         int mid=(l+r)/2;
172         T a1=(p[mid]-p[0]).cross(x-p[0]);
173         T a2=(p[mid+1]-p[0]).cross(x-p[0]);
174         if(a1>0&&a2<=0){
175             T res=(p[mid+1]-p[mid]).cross(x-p[
176                 mid]);
177             return res>0?1:(res==0?-1:0);
178         }else if(a1<0)r=mid-1;
179         else l=mid+1;
180     }
181     return 0;
182 }
183 vector<T> getA()const{ //凸包邊對x軸的夾角
184     vector<T>res; //一定是遞增的
185     for(size_t i=0;i<p.size();++i)
186         res.push_back((p[(i+1)%p.size()]-p[i])
187             .getA());
188     return res;
189 }
190 bool line_intersect(const vector<T>&A,
191     const line<T> &l)const{ //O(LogN)
192     int f1=upper_bound(A.begin(),A.end(),(l.
193         p1-l.p2).getA())-A.begin();
194     int f2=upper_bound(A.begin(),A.end(),(l.
195         p2-l.p1).getA())-A.begin();
196     return l.cross_seg(line<T>(p[f1],p[f2]))
197         ;
198 }
199 polygon cut(const line<T> &l)const{ //凸包
200     對直線切割 · 得到直線L左側的凸包
201     polygon ans;
202     for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
203         if(l.ori(p[i])>=0){
204             ans.p.push_back(p[i]);
205             if(l.ori(p[j])<0)
206                 ans.p.push_back(l.
207                     line_intersection(line<T>(p[i]
208                         ],p[j])));
209         }else if(l.ori(p[j])>0)
210             ans.p.push_back(l.
211                 line_intersection(line<T>(p[i],p[j])));
212     }
213     return ans;
214 }
215 static bool graham_cmp(const point<T>&a,
216     const point<T>&b){ //凸包排序函數
217     return (a.x<b.x)|| (a.x==b.x&&a.y<b.y);
218 }
219 void graham(vector<point<T> > &s){ //凸包
220     sort(s.begin(),s.end(),graham_cmp);
221     p.resize(s.size()+1);
222     int m=0;
223     for(size_t i=0;i<s.size();++i){
224         while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]
225             ]-p[m-2])<=0)--m;
226         p[m++]=s[i];
227     }
228     for(int i=s.size()-2,t=m+1;i>=0;--i){
229         while(m>=t&&(p[m-1]-p[m-2]).cross(s[i]
230             ]-p[m-2])<=0)--m;
231         p[m++]=s[i];
232     }
233     if(s.size()>1)--m;
234     p.resize(m);
235 }
236 T diam(){ //直徑
237     int n=p.size(),t=1;
238     T ans=0;p.push_back(p[0]);
239     for(int i=0;i<n;i++){
240         point<T> now=p[i+1]-p[i];
241         while(now.cross(p[t+1]-p[i])>now.cross
242             (p[t]-p[i]))t=(t+1)%n;
243         ans=max(ans,(p[i]-p[t]).abs2());
244     }
245     return p.pop_back(),ans;
246 }
247 T min_cover_rectangle(){ //最小覆蓋矩形
248     int n=p.size(),t=1,r=1,l=1;

```

```

212 if(n<3)return 0;//也可以做最小周長矩形
213 T ans=1e99;p.push_back(p[0]);
214 for(int i=0;i<n;i++){
215     point<T> now=p[i+1]-p[i];
216     while(now.cross(p[t+1]-p[i])>now.cross
217         (p[t]-p[i]))t=(t+1)%n;
218     while(now.dot(p[r+1]-p[i])>now.dot(p[r
219         ]-p[i]))r=(r+1)%n;
220     if(!i)l=r;
221     while(now.dot(p[l+1]-p[i])<now.dot(p[
222         l]-p[i]))l=(l+1)%n;
223     T d=now.abs2();
224     T tmp=now.cross(p[t]-p[i])*(now.dot(p[
225         r]-p[i])-now.dot(p[l]-p[i]))/d;
226     ans=min(ans,tmp);
227 }
228 return p.pop_back(),ans;
229 }
230 T dis2(polygon &p1){//凸包最近距離平方
231     vector<point<T> > &P=p,&Q=p1.p;
232     int n=P.size(),m=Q.size(),l=0,r=0;
233     for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
234     for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
235     P.push_back(P[0]),Q.push_back(Q[0]);
236     T ans=1e99;
237     for(int i=0;i<n;++i){
238         while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])
239             <0)r=(r+1)%m;
240         ans=min(ans,line<T>(P[l],P[l+1]).
241             seg_dis2(line<T>(Q[r],Q[r+1])));
242         l=(l+1)%n;
243     }
244     return P.pop_back(),Q.pop_back(),ans;
245 }
246 static char sign(const point<T>&t){
247     return (t.y==0?t.x:t.y)<0;
248 }
249 static bool angle_cmp(const line<T>& A,
250     const line<T>& B){
251     point<T> a=A.p2-A.p1,b=B.p2-B.p1;
252     return sign(a)<sign(b)||(!sign(a)==sign(b)
253         )&&a.cross(b)>0;
254 }
255 int halfplane_intersection(vector<line<T>
256     > &s){//半平面交
257     sort(s.begin(),s.end(),angle_cmp);//線段
258     左側為該線段半平面
259     int L,R,n=s.size();
260     vector<point<T> > px(n);
261     vector<line<T> > q(n);
262     q[L=R=0]=s[0];
263     for(int i=1;i<n;++i){
264         while(L<R&&s[i].ori(px[R-1])<=0)--R;
265         while(L<R&&s[i].ori(px[L])<=0)+L;
266         q[++R]=s[i];
267         if(q[R].parallel(q[R-1])){
268             --R;
269             if(q[R].ori(s[i].p1)>0)q[R]=s[i];
270         }
271         if(L<R)px[R-1]=q[R-1].
272             line_intersection(q[R]);
273     }
274     while(L<R&&q[L].ori(px[R-1])<=0)--R;
275     p.clear();
276     if(R-L<=1)return 0;
277     px[R]=q[R].line_intersection(q[L]);
278     for(int i=L;i<=R;++i)p.push_back(px[i]);
279     return R-L+1;
280 }
281 template<typename T>
282 struct triangle{
283     point<T> a,b,c;
284     triangle(){}
285     triangle(const point<T> &a,const point<T>
286         &b,const point<T> &c):a(a),b(b),c(c){}
287     T area()const{
288         T t=(b-a).cross(c-a)/2;
289         return t>0?t:-t;
290     }
291     point<T> barycenter()const{//重心
292         return (a+b+c)/3;
293     }
294     point<T> circumcenter()const{//外心
295         static line<T> u,v;
296         u.p1=(a+b)/2;
297         u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
298             b.x);
299         v.p1=(a+c)/2;
300         v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
301             c.x);
302         return u.line_intersection(v);
303     }
304     point<T> incenter()const{//內心
305         T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
306             ()),C=sqrt((a-b).abs2());
307         return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
308             B*b.y+C*c.y)/(A+B+C);
309     }
310     point<T> perpcenter()const{//垂心
311         return barycenter()*3-circumcenter()*2;
312     }
313 };
314 template<typename T>
315 struct point3D{
316     T x,y,z;
317     point3D(){}
318     point3D(const T&x,const T&y,const T&z):x(x
319         ),y(y),z(z){}
320     point3D operator+(const point3D &b)const{
321         return point3D(x+b.x,y+b.y,z+b.z);}
322     point3D operator-(const point3D &b)const{
323         return point3D(x-b.x,y-b.y,z-b.z);}
324     point3D operator*(const T &b)const{
325         return point3D(x*b,y*b,z*b);}
326     point3D operator/(const T &b)const{
327         return point3D(x/b,y/b,z/b);}
328     bool operator==(const point3D &b)const{
329         return x==b.x&&y==b.y&&z==b.z;}
330     T dot(const point3D &b)const{
331         return x*b.x+y*b.y+z*b.z;}
332     point3D cross(const point3D &b)const{
333         return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
334             *b.y-y*b.x);}
335     T abs2()const{//向量長度的平方
336         return dot(*this);}
337     T area2(const point3D &b)const{//和b、原點
338         圍成面積的平方
339         return cross(b).abs2()/4;}
340 };
341 template<typename T>
342 struct line3D{
343     point3D<T> p1,p2;
344     line3D(){}
345     line3D(const point3D<T> &p1,const point3D<
346         T> &p2):p1(p1),p2(p2){}
347     T dis2(const point3D<T> &p,bool is_segment
348         =0)const{//點跟直線/線段的距離平方
349         point3D<T> v=p2-p1,v1=p-p1;
350         if(is_segment){
351             point3D<T> v2=p-p2;
352             if(v.dot(v1)<=0)return v1.abs2();
353             if(v.dot(v2)>=0)return v2.abs2();
354         }
355         point3D<T> tmp=v.cross(v1);
356         return tmp.abs2()/v.abs2();
357     }
358     pair<point3D<T>,point3D<T> > closest_pair(
359         const line3D<T> &l)const{
360         point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
361         point3D<T> N=v1.cross(v2),ab(p1-l.p1);
362         //if(N.abs2()==0)return NULL; //平行或重合
363         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
364         最近點對距離
365         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
366             cross(d2),G=l.p1-p1;
367         T t1=(G.cross(d2)).dot(D)/D.abs2();
368         T t2=(G.cross(d1)).dot(D)/D.abs2();
369         return make_pair(p1+d1*t1,l.p1+d2*t2);
370     }
371     bool same_side(const point3D<T> &a,const
372         point3D<T> &b)const{
373         return (p2-p1).cross(a-p1).dot((p2-p1).
374             cross(b-p1))>0;
375     }
376 };
377 template<typename T>
378 struct plane{
379     point3D<T> p0,n;//平面上的點和法向量
380     plane(){}
381     plane(const point3D<T> &p0,const point3D<T>
382         &n):p0(p0),n(n){}
383     T dis2(const point3D<T> &p)const{//點到平
384         面距離的平方
385         T tmp=(p-p0).dot(n);
386         return tmp*tmp/n.abs2();
387     }
388     point3D<T> projection(const point3D<T> &p)
389         const{
390         return p-n*(p-p0).dot(n)/n.abs2();
391     }
392     point3D<T> line_intersection(const line3D<
393         T> &l)const{
394         T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
395         重合該平面
396         return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
397             tmp);
398     }
399     line3D<T> plane_intersection(const plane &
400         p1)const{
401         point3D<T> e=n.cross(p1.n),v=n.cross(e);
402         T tmp=p1.n.dot(v);//等於0表示平行或重合
403         該平面
404         point3D<T> q=p0+(v*(p1.n.dot(p1.p0-p0))/
405             tmp);
406         return line3D<T>(q,q+e);
407     }
408 };
409 template<typename T>
410 struct triangle3D{
411     point3D<T> a,b,c;
412     triangle3D(){}
413     triangle3D(const point3D<T> &a,const
414         point3D<T> &b,const point3D<T> &c):a(a)
415         ,b(b),c(c){}
416     bool point_in(const point3D<T> &p)const{//
417         點在該平面上的投影在三角形中
418         return line3D<T>(b,c).same_side(p,a)&&
419             line3D<T>(a,c).same_side(p,b)&&
420             line3D<T>(a,b).same_side(p,c);
421     }
422 };
423 template<typename T>
424 struct tetrahedron{//四面體
425     point3D<T> a,b,c,d;
426     tetrahedron(){}
427     tetrahedron(const point3D<T> &a,const
428         point3D<T> &b,const point3D<T> &c,
429         const point3D<T> &d):a(a),b(b),c(c),d(
430             d){}
431     T volume6()const{//體積的六倍
432         return (d-a).dot((b-a).cross(c-a));
433     }
434     point3D<T> centroid()const{
435         return (a+b+c+d)/4;
436     }
437     bool point_in(const point3D<T> &p)const{
438         return triangle3D<T>(a,b,c).point_in(p)
439             &&triangle3D<T>(c,d,a).point_in(p);
440     }
441 };
442 template<typename T>
443 struct convexhull3D{
444     static const int MAXN=1005;
445     struct face{
446         int a,b,c;
447         face(int a,int b,int c):a(a),b(b),c(c){}
448     };
449     vector<point3D<T> > pt;
450     vector<face> ans;
451     int fid[MAXN][MAXN];
452     void build(){
453         int n=pt.size();
454         ans.clear();
455         memset(fid,0,sizeof(fid));
456         ans.emplace_back(0,1,2);//注意不能共線
457         ans.emplace_back(2,1,0);
458         int ftop = 0;
459         for(int i=3; ftop=1; i<n; ++i,++ftop){
460             vector<face> next;
461             for(auto &f:ans){
462                 T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[
463                     f.a]).cross(pt[f.c]-pt[f.a]));
464                 if(d<=0) next.push_back(f);
465                 int ff=0;
466                 if(d>0) ff=ftop;
467                 else if(d<0) ff=-ftop;
468             }
469         }
470     }

```

```

424     fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c
        ][f.a]=ff;
425 }
426 for(auto &f:ans){
427     if(fid[f.a][f.b]>0 && fid[f.a][f.b
        ]!=fid[f.b][f.a])
428         next.emplace_back(f.a,f.b,i);
429     if(fid[f.b][f.c]>0 && fid[f.b][f.c
        ]!=fid[f.c][f.b])
430         next.emplace_back(f.b,f.c,i);
431     if(fid[f.c][f.a]>0 && fid[f.c][f.a
        ]!=fid[f.a][f.c])
432         next.emplace_back(f.c,f.a,i);
433 }
434 ans=next;
435 }
436 }
437 point3D<T> centroid()const{
438     point3D<T> res(0,0,0);
439     T vol=0;
440     for(auto &f:ans){
441         T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c
            ]));
442         res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
443         vol+=tmp;
444     }
445     return res/(vol*4);
446 }
447 };

```

1.2 SmallestCircle

```

1 using PT=point<T>; using CPT=const PT;
2 PT circumcenter(CPT &a,CPT &b,CPT &c){
3     PT u=b-a, v=c-a;
4     T c1=u.abs2()/2, c2=v.abs2()/2;
5     T d=u.cross(v);
6     return PT(a.x+(v.y*c1-u.y*c2)/d,a.y+(u.x*
            c2-v.x*c1)/d);
7 }
8 void solve(PT p[],int n,PT &c,T &r2){
9     random_shuffle(p,p+n);
10    c=p[0]; r2=0; // c,r2 = 圓心,半徑平方
11    for(int i=1;i<n;i++){if((p[i]-c).abs2())>r2){
12        c=p[i]; r2=0;
13    }
14    for(int j=0;j<i;j++){if((p[j]-c).abs2())>r2){
15        c.x=(p[i].x+p[j].x)/2;
16        c.y=(p[i].y+p[j].y)/2;
17        r2=(p[j]-c).abs2();
18    }
19    for(int k=0;k<j;k++){if((p[k]-c).abs2())>r2){
20        c=circumcenter(p[i],p[j],p[k]);
21        r2=(p[i]-c).abs2();
22    }
23 }

```

1.3 最近點對

```

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(dx*dx+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

```

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[MAXN]; //每個節點代表的
        //列跟行
6     int L[MAXN],R[MAXN],U[MAXN],D[MAXN];
7     vector<int> ans,ans2;
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)remove(col[j]);
60         if(dfs(d+1))return 1;
61         ans.pop_back();
62         DFOR(j,L,i)restore(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=ans2;return;}
70     int c=R[0];
71     DFOR(i,R,0)if(S[i]<S[c])c=i;
72     DFOR(i,D,c){
73         ans.push_back(row[i]);
74         remove2(i);
75         DFOR(j,R,i)remove2(j),--S[col[j]];
76         dfs2(d+1);
77         ans.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     ans.clear(); //暫存用,答案還是存在ans裡
87     dfs2(0);
88 }
89 #undef DFOR
90 };

```

2.2 Dynamic_KD_tree

```

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

```

```

46 node* build(int k,int l,int r){
47     if(l>r) return 0;
48     if(k==kd) k=0;
49     int mid=(l+r)/2;
50     cmp.sort_id = k;
51     nth_element(A.begin()+l,A.begin()+mid,A.
52         begin()+r+1,cmp);
53     node *ret=A[mid];
54     ret->l = build(k+1,l,mid-1);
55     ret->r = build(k+1,mid+1,r);
56     ret->up();
57     return ret;
58 }
59 bool isbad(node*o){
60     return size(o->l)>alpha*o->s||size(o->r)
61         >alpha*o->s;
62 }
63 void flatten(node *u,typename vector<node
64     *>::iterator &it){
65     if(!u)return;
66     flatten(u->l,it);
67     *it=u;
68     flatten(u->r,++it);
69 }
70 void rebuild(node*&u,int k){
71     if((int)A.size()<u->s)A.resize(u->s);
72     auto it=A.begin();
73     flatten(u,it);
74     u=build(k,0,u->s-1);
75 }
76 bool insert(node*&u,int k,const point &x,
77     int dep){
78     if(!u) return u=new node(x), dep<=0;
79     ++u->s;
80     cmp.sort_id=k;
81     if(insert(cmp(x,u->pid)?u->l:u->r,(k+1)%
82         kd,x,dep-1)){
83         if(!isbad(u))return 1;
84         rebuild(u,k);
85     }
86     return 0;
87 }
88 node *findmin(node*o,int k){
89     if(!o)return 0;
90     if(cmp.sort_id==k)return o->l?findmin(o
91         ->l,(k+1)%kd):o;
92     node *l=findmin(o->l,(k+1)%kd);
93     node *r=findmin(o->r,(k+1)%kd);
94     if(l&&!r)return cmp(l,o)?l:o;
95     if(!l&&r)return cmp(r,o)?r:o;
96     if(!l&&!r)return o;
97     if(cmp(l,r))return cmp(l,o)?l:o;
98     return cmp(r,o)?r:o;
99 }
100 bool erase(node *&u,int k,const point &x){
101     if(!u)return 0;
102     if(u->pid==x){
103         if(u->r);
104         else if(u->l) u->r=u->l, u->l=0;
105         else return delete(u),u=0, 1;
106         --u->s;
107         cmp.sort_id=k;
108         u->pid=findmin(u->r,(k+1)%kd)->pid;
109         return erase(u->r,(k+1)%kd,u->pid);
110     }
111     cmp.sort_id=k;
112     if(erase(cmp(x,u->pid)?u->l:u->r,(k+1)%
113         kd,x))
114         return --u->s, 1;
115     return 0;
116 }
117 T heuristic(const T h[])const{
118     T ret=0;
119     for(set_t i=0;i<kd;++i)ret+=h[i];
120     return ret;
121 }
122 int qM;
123 priority_queue<pair<T,point>> pQ;
124 void nearest(node *u,int k,const point &x,
125     T *h,T &mndist){
126     if(u==0||heuristic(h)==mndist)return;
127     T dist=u->pid.dist(x),old=h[k];
128     /*mndist=std::min(mndist,dist);*/
129     if(dist<mndist){
130         pQ.push(std::make_pair(dist,u->pid));
131         if((int)pQ.size()==qM+1)
132             mndist=pQ.top().first,pQ.pop();
133     }
134     if(x.d[k]<u->pid.d[k]){
135         nearest(u->l,(k+1)%kd,x,h,mndist);
136         h[k] = abs(x.d[k]-u->pid.d[k]);
137         nearest(u->r,(k+1)%kd,x,h,mndist);
138     }else{
139         nearest(u->r,(k+1)%kd,x,h,mndist);
140         h[k] = abs(x.d[k]-u->pid.d[k]);
141         nearest(u->l,(k+1)%kd,x,h,mndist);
142     }
143     h[k]=old;
144 }
145 vector<point>in_range;
146 void range(node *u,int k,const point&mi,
147     const point&ma){
148     if(!u)return;
149     bool is=1;
150     for(int i=0;i<kd;++i)
151         if(u->pid.d[i]<mi.d[i]||ma.d[i]<u->pid
152             .d[i])
153             { is=0;break; }
154     if(is) in_range.push_back(u->pid);
155     if(mi.d[k]<u->pid.d[k])range(u->l,(k+1)
156         %kd,mi,ma);
157     if(ma.d[k]>u->pid.d[k])range(u->r,(k+1)
158         %kd,mi,ma);
159 }
160 public:
161 kd_tree(const T &INF,double a=0.75):
162     root(0),alpha(a),loga(log2(1.0/a)),INF(INF
163         ),maxn(1){}
164 ~kd_tree(){delete root;}
165 void clear(){delete root,root=0,maxn=1;}
166 void build(int n,const point *p){
167     delete root,A.resize(maxn);
168     for(int i=0;i<n;++i)A[i]=new node(p[i]);
169     root=build(0,0,n-1);
170 }
171 void insert(const point &x){
172     insert(root,0,x,__lg(size(root))/loga);
173     if(root->s>maxn)maxn=root->s;
174 }
175 bool erase(const point &p){
176     bool d=erase(root,0,p);
177     if(root&&root->s<alpha*maxn)rebuild();
178 }
179 if(erase(cmp(x,u->pid)?u->l:u->r,(k+1)%
180     kd,x))
181     return --u->s, 1;
182     return 0;
183 }
184 int size(){return root?root->s:0;}
185 };
186 }
187 void rebuild(){
188     if(root)rebuild(root,0);
189     maxn=root->s;
190 }
191 T nearest(const point &x,int k){
192     qM=k;
193     T mndist=INF,h[k]={};
194     nearest(root,0,x,h,mndist);
195     mndist=pQ.top().first;
196     pQ = priority_queue<pair<T,point>>();
197     return mndist; //回傳離x第k近的點的距離
198 }
199 const vector<point> &range(const point&mi,
200     const point&ma){
201     in_range.clear();
202     range(root,0,mi,ma);
203     return in_range; //回傳介於mi到ma之間的點
204     vector
205 }
206 }
207 int size(){return root?root->s:0;}
208 };
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

2.3 kd_tree_replace_segment

```

1 struct node{//kd樹代替高維線段樹
2     node *l,*r;
3     point pid,mi,ma;
4     int s, data;
5     node(const point &p,int d):l(0),r(0),pid(p
6         ),mi(p),ma(p),s(1),data(d),dmin(d),
7         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            s+=l->s;
17        }
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        void up2(){/*其他懶惰標記向上更新*/}
26        void down(){/*其他懶惰標記下推*/}
27    }*root;
28    //檢查區間包含用的函數
29    bool range_include(node *o,const point &L,
30        const point &R){
31        for(int i=0;i<kd;++i){
32            if(L.d[i]>o->ma.d[i]||R.d[i]<o->mi.d[i])
33                return 0;
34        }
35        return 1;
36    }
37    bool range_in_range(node *o,const point &L,
38        const point &R){
39        for(int i=0;i<kd;++i){
40            if(L.d[i]>o->mi.d[i]||R.d[i]<o->ma.d[i])
41                return 0;
42        }
43        return 1;
44    }
45    //單點修改 · 以單點改值為例
46    void update(node *u,const point &x,int data,
47        int k=0){
48        if(!u)return;
49        u->down();
50        if(u->pid==x){
51            u->data=data;
52            u->up2();
53            return;
54        }
55        cmp.sort_id=k;
56        update(cmp(x,u->pid)?u->l:u->r,x,data,(k
57            +1)%kd);
58        u->up2();
59    }
60    //區間修改
61    void update(node *o,const point &L,const
62        point &R,int data){
63        if(!o)return;
64        o->down();
65        if(range_in_range(o,L,R)){
66            //區間懶惰標記修改
67            o->down();
68            return;
69        }
70        if(point_in_range(o,L,R)){
71            //這個點在(L,R)區間 · 但是他的左右子樹不
72            //一定在區間中
73            //單點懶惰標記修改
74            if(o->l&&range_include(o->l,L,R))update(o
75                ->l,L,R,data);
76            if(o->r&&range_include(o->r,L,R))update(o
77                ->r,L,R,data);
78            o->up2();
79        }
80        //區間查詢 · 以總和為例
81        int query(node *o,const point &L,const point
82            &R){
83            if(!o)return 0;
84            o->down();
85            if(range_in_range(o,L,R))return o->sum;
86            int ans=0;
87            if(point_in_range(o,L,R))ans+=o->data;
88            if(o->l&&range_include(o->l,L,R))ans+=
89                query(o->l,L,R);
90            if(o->r&&range_include(o->r,L,R))ans+=
91                query(o->r,L,R);
92            return ans;
93        }
94    }
95    //區間查詢 · 以總和為例
96    int query(node *o,const point &L,const point
97        &R){
98        if(!o)return 0;
99        o->down();
100        if(range_in_range(o,L,R))return o->sum;
101        int ans=0;
102        if(point_in_range(o,L,R))ans+=o->data;
103        if(o->l&&range_include(o->l,L,R))ans+=
104            query(o->l,L,R);
105        if(o->r&&range_include(o->r,L,R))ans+=
106            query(o->r,L,R);
107        return ans;
108    }
109 }

```



```

84 if(o->r&&range_include(o->r,L,R))ans+=
    query(o->r,L,R);
85 return ans;
86 }

```

2.4 reference_point

```

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

```

1 node *merge(node *a,node *b){
2     if(!a||!b) return 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

```

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 }dfs;

```

2.7 整體二分

```

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 Flow

3.1 dinic

```

1 template<typename T>
2 struct DINIC{
3     static const int MAXN=105;
4     static const T INF=INT_MAX;
5     int n, LV[MAXN], cur[MAXN];
6     struct edge{
7         int v,pre;
8         T cap,r;
9         edge(int v,int pre,T cap):v(v),pre(pre),
10            cap(cap),r(cap){}
11     };
12     int g[MAXN];
13     vector<edge> e;

```

```

13 void init(int _n){
14     memset(g,-1,sizeof(int)*((n=_n)+1));
15     e.clear();
16 }
17 void add_edge(int u,int v,T cap,bool
    directed=false){
18     e.push_back(edge(v,g[u],cap));
19     g[u]=e.size()-1;
20     e.push_back(edge(u,g[v],directed?0:cap))
21     ;
22     g[v]=e.size()-1;
23 }
24 int bfs(int s,int t){
25     memset(LV,0,sizeof(int)*(n+1));
26     memcpy(cur,g,sizeof(int)*(n+1));
27     queue<int> q;
28     q.push(s);
29     LV[s]=1;
30     while(q.size()){
31         int u=q.front();q.pop();
32         for(int i=g[u];~i;i=e[i].pre){
33             if(!LV[e[i].v]&&e[i].r){
34                 LV[e[i].v]=LV[u]+1;
35                 q.push(e[i].v);
36                 if(e[i].v==t) return 1;
37             }
38         }
39     }
40     return 0;
41 }
42 T dfs(int u,int t,T CF=INF){
43     if(u==t) return CF;
44     T df;
45     for(int &i=cur[u];~i;i=e[i].pre){
46         if(LV[e[i].v]==LV[u]+1&&e[i].r){
47             if(df=dfs(e[i].v,t,min(CF,e[i].r)))
48                 e[i].r-=df;
49             e[i]^1.r+=df;
50             return df;
51         }
52     }
53     return LV[u]=0;
54 }
55 T dinic(int s,int t,bool clean=true){
56     if(clean)for(size_t i=0;i<e.size();++i)
57         e[i].r=e[i].cap;
58     T ans=0, f=0;
59     while(bfs(s,t))while(f=dfs(s,t))ans+=f;
60     return ans;
61 }
62 };

```

3.2 Gomory_Hu

```

1 //最小割樹+求任兩點間最小割
2 //0-base, root=0
3 LL e[MAXN][MAXN]; //任兩點間最小割
4 int p[MAXN]; //parent
5 ISAP D; // original graph
6 void gomory_hu(){
7     fill(p, p+n, 0);

```

```

8     fill(e[0], e[n], INF);
9     for( int s = 1; s < n; ++s ) {
10         int t = p[s];
11         ISAP F = D;
12         LL tmp = F.min_cut(s, t);
13         for( int i = 1; i < s; ++i )
14             e[s][i] = e[i][s] = min(tmp, e[t][i]);
15         for( int i = s+1; i <= n; ++i )
16             if( p[i] == t && F.vis[i] ) p[i] = s;
17     }
18 }

```

3.3 ISAP_with_cut

```

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,r;
10        edge(int v,int pre,T cap):v(v),pre(pre),
11            cap(cap),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
        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        ;
24        g[v]=e.size()-1;
25    }
26    T dfs(int u,int s,int t,T CF=INF){
27        if(u==t) return CF;
28        T tf=CF,df;
29        for(int &i=cur[u];~i;i=e[i].pre){
30            if(e[i].r&&d[u]==d[e[i].v]+1){
31                df=dfs(e[i].v,s,t,min(tf,e[i].r));
32                e[i].r-=df;
33                e[i]^1.r+=df;
34                if(!(tf-=df)||d[s]==n) return CF-tf;
35            }
36        }
37        int mh=n;
38        for(int i=cur[u]=g[u];~i;i=e[i].pre){
39            if(e[i].r&&d[e[i].v]<mh)mh=d[e[i].v];
40        }
41        if(!--gap[d[u]])d[s]=n;
42        else ++gap[d[u]=++mh];
43        return CF-tf;
44    }
45    T isap(int s,int t,bool clean=true){
46        memset(d,0,sizeof(int)*(n+1));
47        memset(gap,0,sizeof(int)*(n+1));
48        memcpy(cur,g,sizeof(int)*(n+1));

```

```

47 if(clean) for(size_t i=0;i<e.size();++i)
48   e[i].r=e[i].cap;
49   T MF=0;
50   for(gap[0]=n;d[s]<n;)MF+=dfs(s,s,t);
51   return MF;
52 }
53 vector<int> cut_e;//最小割邊集
54 bool vis[MAXN];
55 void dfs_cut(int u){
56   vis[u]=1;//表示u屬於source的最小割集
57   for(int i=g[u];~i;i=e[i].pre)
58     if(e[i].r>0&&!vis[e[i].v])dfs_cut(e[i].v);
59 }
60 T min_cut(int s,int t){
61   T ans=isap(s,t);
62   memset(vis,0,sizeof(bool)*(n+1));
63   dfs_cut(s); cut_e.clear();
64   for(int u=0;u<n;++u)if(vis[u])
65     for(int i=g[u];~i;i=e[i].pre)
66       if(!vis[e[i].v])cut_e.push_back(i);
67   return ans;
68 }
69 };

```

3.4 MinCostMaxFlow

```

1 template<typename TP>
2 struct MCMF{
3   static const int MAXN=440;
4   static const TP INF=999999999;
5   struct edge{
6     int v,pre;
7     TP r,cost;
8     edge(int v,int pre,TP r,TP cost):v(v),
9       pre(pre),r(r),cost(cost){}
10 };
11 int n,S,T;
12 TP 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,TP r,TP cost,
21   bool directed=false){
22   e.push_back(edge(v,g[u],r,cost));
23   g[u]=e.size()-1;
24   e.push_back(
25     edge(u,g[v],directed?0:r,-cost));
26   g[v]=e.size()-1;
27 }
28 TP augment(int u,TP CF){
29   if(u==T||!CF)return ans+=PIS*CF,CF;
30   vis[u]=1;
31   TP r=CF,d;
32   for(int i=g[u];~i;i=e[i].pre){
33     if(e[i].r&&!e[i].cost&&vis[e[i].v]){
34       d=augment(e[i].v,min(r,e[i].r));
35       e[i].r-=d;

```

```

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

```

4 Graph

4.1 Augmenting_Path

```

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 0-base
6 bool vis[MAXN2];//是否走訪過
7 bool dfs(int u){
8   for(int v:g[u]){
9     if(vis[v]) continue;
10    vis[v]=1;
11    if(match[v]==-1||dfs(match[v]))
12      return match[v]=u, 1;
13  }
14  return 0;
15 }
16 int max_match(){
17   int ans=0;
18   memset(match,-1,sizeof(int)*n2);

```

```

19 for(int i=0;i<n1;++i){
20   memset(vis,0,sizeof(bool)*n2);
21   if(dfs(i)) ++ans;
22 }
23 return ans;
24 }

```

4.2 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 0-base
6 size_t c[MAXN2];//每個屬於n2點最多可以接受幾
7   條匹配邊
8 vector<int> matchs[MAXN2];//每個屬於n2的點匹
9   配了那些點
10 bool vis[MAXN2];
11 bool dfs(int u){
12   for(int v:g[u]){
13     if(vis[v])continue;
14     vis[v]=1;
15     if(matchs[v].size()<c[v]){
16       return matchs[v].push_back(u), 1;
17     }else for(size_t j=0;j<matchs[v].size()
18       ;++j){
19       if(dfs(matchs[v][j]))
20         return matchs[v][j]=u, 1;
21     }
22   }
23   return 0;
24 }
25 int max_match(){
26   for(int i=0;i<n2;++i) matchs[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 }

```

4.3 blossom_matching

```

1 #define MAXN 505
2 int n; //1-base
3 vector<int> g[MAXN];
4 int MH[MAXN]; //output MH
5 int pa[MAXN],st[MAXN],S[MAXN],v[MAXN],t;
6 int lca(int x,int y){
7   for(++t;swap(x,y)){
8     if(!x) continue;
9     if(v[x]==t) return x;
10    v[x]=t;
11    x=st[pa[MH[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   while(st[x]!=1){
17     pa[x]=y;
18     if(S[y==MH[x]]==1)qpush(y);
19     st[x]=st[y]=1, x=pa[y];
20   }
21 }
22 bool bfs(int x){
23   iota(st+1, st+n+1, 1);
24   memset(S+1,-1,sizeof(int)*n);
25   queue<int>q; qpush(x);
26   while(q.size()){
27     x=q.front(),q.pop();
28     for(int y:g[x]){
29       if(S[y]==-1){
30         pa[y]=x,S[y]=1;
31         if(!MH[y]){
32           for(int lst;x=y=lst,x=pa[y])
33             lst=MH[lst],MH[x]=y,MH[y]=x;
34           return 1;
35         }
36         qpush(MH[y]);
37       }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 int blossom(){
46   memset(MH+1,0,sizeof(int)*n);
47   int ans=0;
48   for(int i=1;i<n;++i)
49     if(!MH[i]&&bfs(i)) ++ans;
50   return ans;
51 }

```

4.4 graphISO

```

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]*AP;
20       for(int j:g[i])f[t][i]=(f[t][i]+f[t-1][j]*BP)%P;
21     }
22   }
23 }
24 for(int j:rg[i])f[t][i]=(f[t][i]+f[t-1][j]*CP)%P;

```

```

21     if(i==u)f[t][i]+=D;//如果圖太大的話・
        把這行刪掉・執行一次後f[K]就會是所
        有點的答案
22     f[t][i]%=P;
23 }
24 }
25 return f[K][u];
26 }
27 vector<long long> graph_hash(){
28     vector<long long> ans;
29     for(int i=0;i<n;++i)ans.push_back(
        point_hash(i));//O(N^2)
30     sort(ans.begin(),ans.end());
31     return ans;
32 }

```

4.5 KM

```

1 #define MAXN 405
2 #define INF 0x3f3f3f3f3f3f3f3f
3 int n;// 1-base・0表示沒有匹配
4 LL g[MAXN][MAXN]; //input graph
5 int My[MAXN],Mx[MAXN]; //output match
6 LL lx[MAXN],ly[MAXN],pa[MAXN],Sy[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=Mx[x];
11         My[y]=x,Mx[x]=y;
12     }
13 }
14 void bfs(int st){
15     for(int i=1;i<=n;++i)
16         Sy[i] = INF, vx[i]=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                 LL t = lx[x]+ly[y]-g[x][y];
24                 if(t==0){
25                     pa[y]=x;
26                     if(!My[y]){augment(y);return;}
27                     vy[y]=1,q.push(My[y]);
28                 }else if(Sy[y]>t) pa[y]=x,Sy[y]=t;
29             }
30         }
31         LL cut = INF;
32         for(int y=1;y<=n;++y)
33             if(!vy[y]&&cut>Sy[y]) cut=Sy[y];
34         for(int j=1;j<=n;++j){
35             if(vx[j]) lx[j] -= cut;
36             if(vy[j]) ly[j] += cut;
37             else Sy[j] -= cut;
38         }
39         for(int y=1;y<=n;++y){
40             if(!vy[y]&&Sy[y]==0){
41                 if(!My[y]){augment(y);return;}
42                 vy[y]=1, q.push(My[y]);
43             }
44         }

```

```

45     }
46 }
47 LL KM(){
48     memset(My,0,sizeof(int)*(n+1));
49     memset(Mx,0,sizeof(int)*(n+1));
50     memset(ly,0,sizeof(LL)*(n+1));
51     for(int x=1;x<=n;++x){
52         lx[x] = -INF;
53         for(int y=1;y<=n;++y)
54             lx[x] = max(lx[x],g[x][y]);
55     }
56     for(int x=1;x<=n;++x) bfs(x);
57     LL ans = 0;
58     for(int y=1;y<=n;++y) ans+=g[My[y]][y];
59     return ans;
60 }

```

4.6 MaximumClique

```

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

```

```

42 }
43 };

```

4.7 MinimumMeanCycle

```

1 #include<cstdio> //for DBL_MAX
2 int dp[MAXN][MAXN]; // 1-base,0(NM)
3 vector<tuple<int,int,int>> edge;
4 double mmc(int n){//allow negative weight
5     const int INF=0x3f3f3f3f;
6     for(int t=0;t<n;++t){
7         memset(dp[t+1],0,sizeof(dp[t+1]));
8         for(const auto &e:edge){
9             int u,v,w;
10            tie(u,v,w) = e;
11            dp[t+1][v]=min(dp[t+1][v],dp[t][u]+w);
12        }
13    }
14    double res = DBL_MAX;
15    for(int u=1;u<=n;++u){
16        if(dp[n][u]==INF) continue;
17        double val = -DBL_MAX;
18        for(int t=0;t<n;++t)
19            val=max(val,(dp[n][u]-dp[t][u])*1.0/(n-t));
20        res=min(res,val);
21    }
22    return res;
23 }

```

4.8 Rectilinear_MST

```

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),
        id(id){}

```

```

27 };
28 vector<bit_node> bit;
29 void bit_update(int i,const T&data,int id){
30     for(;i;i=i&(-i)){
31         if(data<bit[i].mi)bit[i]=bit_node(data,
            id);
32     }
33 }
34 int bit_find(int i,int m){
35     bit_node x;
36     for(;i<=m;i=i&(-i)) if(bit[i].mi<x.mi)x=
        bit[i];
37     return x.id;
38 }
39 vector<edge> build_graph(int n,point p[]){
40     vector<edge> e;//edge for MST
41     for(int dir=0;dir<4;++dir){//4種座標變換
42         if(dir%2) for(int i=0;i<n;++i) swap(p[i]
            .x,p[i].y);
43         else if(dir==2) for(int i=0;i<n;++i) p[i]
            .x=-p[i].x;
44         sort(p,p+n,cmpx);
45         vector<T> ga(n), gb;
46         for(int i=0;i<n;++i)ga[i]=p[i].y-p[i].x;
47         gb=ga, sort(gb.begin(),gb.end());
48         gb.erase(unique(gb.begin(),gb.end()),gb.
            end());
49         int m=gb.size();
50         bit=vector<bit_node>(m+1);
51         for(int i=n-1;i>=0;--i){
52             int pos=lower_bound(gb.begin(),gb.end
                (),ga[i])-gb.begin()+1;
53             int ans=bit_find(pos,m);
54             if(~ans)e.push_back(edge(p[i].id,p[ans
                ].id,p[i].dist(p[ans])));
55             bit_update(pos,p[i].x+p[i].y,i);
56         }
57     }
58     return e;
59 }

```

4.9 treeISO

```

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 }

```

4.10 一般圖最小權完美匹配

```

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[v]){
23                int m = match[v];
24                if (dis[m] > dis[u] - edge[v][m] +
25                    edge[u][v]){
26                    dis[m] = dis[u] - edge[v][m] +
27                        edge[u][v];
28                    onstk[v] = 1;
29                    stk.push_back(v);
30                    if (SPFA(m)) return true;
31                    stk.pop_back();
32                    onstk[v] = 0;
33                }
34            }
35            onstk[u] = 0;
36            stk.pop_back();
37            return false;
38        }
39    }
40    int solve() {
41        // find a match
42        for (int i=0; i<n; i+=2){
43            match[i] = i+1, match[i+1] = i;
44        }
45        for(;;){
46            int found = 0;
47            for (int i=0; i<n; i++) dis[i] = onstk[i] = 0;
48            for (int i=0; i<n; i++){
49                stk.clear();
50                if (!onstk[i] && SPFA(i)){
51                    found = 1;
52                    while (stk.size())>=2){

```

```

50        int u = stk.back(); stk.pop_back();
51        int v = stk.back(); stk.pop_back();
52        match[u] = v;
53        match[v] = u;
54    }
55    }
56    }
57    if (!found) break;
58    }
59    int ret = 0;
60    for (int i=0; i<n; i++)
61        ret += edge[i][match[i]];
62    ret /= 2;
63    return ret;
64    }
65 }graph;

```

4.11 全局最小割

```

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

```

4.12 平面圖判定

```

1 static const int MAXN = 20;
2 struct Edge{
3     int u, v;
4     Edge(int s, int d) : u(s), v(d) {}
5 };
6 bool isK33(int n, int degree[]){
7     int t = 0, z = 0;
8     for(int i=0; i<n; ++i){
9         if(degree[i] == 3)++t;
10        else if(degree[i] == 0)++z;
11        else return false;
12    }
13    return t == 6 && t + z == n;
14 }
15 bool isK5(int n, int degree[]){
16     int f = 0, z = 0;
17     for(int i=0; i<n; ++i){
18         if(degree[i] == 4)++f;
19         else if(degree[i] == 0)++z;
20         else return false;
21    }
22    return f == 5 && f + z == n;
23 }
24 // it judge a given graph is Homeomorphic
25 with K33 or K5
26 bool isHomeomorphic(bool G[MAXN][MAXN],
27     const int n){
28     for(;;){
29         int cnt = 0;
30         for(int i=0; i<n; ++i){
31             vector<Edge> E;
32             for(int j=0; j<n && E.size()<3; ++j)
33                 if(G[i][j] && i != j)
34                     E.push_back(Edge(i, j));
35             if(E.size() == 1){
36                 G[i][E[0].v] = G[E[0].v][i] = false;
37             }else if(E.size() == 2){
38                 G[i][E[0].v] = G[E[0].v][i] = false;
39                 G[i][E[1].v] = G[E[1].v][i] = false;
40                 G[E[0].v][E[1].v] = G[E[1].v][E[0].v] = true;
41             }
42             ++cnt;
43         }
44         if(cnt == 0)break;
45     }
46     static int degree[MAXN];
47     fill(degree, degree + n, 0);
48     for(int i=0; i<n; ++i){
49         for(int j=i+1; j<n; ++j){
50             if(!G[i][j])continue;
51             ++degree[i];
52             ++degree[j];
53         }
54     }
55     return !(isK33(n, degree) || isK5(n, degree));
56 }

```

4.13 弦圖完美消除序列

```

1 struct chordal{
2     static const int MAXN=1005;

```

```

3     int n; // 0-base
4     vector<int> G[MAXN];
5     int rank[MAXN], label[MAXN];
6     bool mark[MAXN];
7     void init(int _n){n=_n;
8         for(int i=0; i<n; ++i)G[i].clear();
9     }
10    void add_edge(int u, int v){
11        G[u].push_back(v);
12        G[v].push_back(u);
13    }
14    vector<int> MCS(){
15        memset(rank, -1, sizeof(int)*n);
16        memset(label, 0, sizeof(int)*n);
17        priority_queue<pair<int, int>> pq;
18        for(int i=0; i<n; ++i)pq.push(make_pair(0, i));
19        for(int i=n-1; i>=0; --i)for(;;){
20            int u=pq.top().second; pq.pop();
21            if(~rank[u])continue;
22            rank[u]=i;
23            for(auto v:G[u])if(rank[v]==-1){
24                pq.push(make_pair(++label[v], v));
25            }
26            break;
27        }
28        vector<int> res(n);
29        for(int i=0; i<n; ++i)res[rank[i]]=i;
30        return res;
31    }
32    bool check(vector<int> ord){ // 弦圖判定
33        for(int i=0; i<n; ++i)rank[ord[i]]=i;
34        memset(mark, 0, sizeof(bool)*n);
35        for(int i=0; i<n; ++i){
36            vector<pair<int, int>> tmp;
37            for(auto u:G[ord[i]])if(!mark[u])
38                tmp.push_back(make_pair(rank[u], u));
39            sort(tmp.begin(), tmp.end());
40            if(tmp.size()){
41                int u=tmp[0].second;
42                set<int> S;
43                for(auto v:G[u])S.insert(v);
44                for(size_t j=1; j<tmp.size(); ++j)
45                    if(!S.count(tmp[j].second))return 0;
46            }
47            mark[ord[i]]=1;
48        }
49        return 1;
50    }
51 };

```

4.14 最小斯坦納樹 DP

```

1 //n個點，其中r個要構成斯坦納樹
2 //答案在max(dp[(1<r)-1][k]) k=0~n-1
3 //p表示要構成斯坦納樹的點集
4 //O( n^3 + n*3^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     }
29 }

```

4.15 最小樹形圖 — 朱劉

```

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(w),tag(0),l(0),r(0){}
9     }
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     }
25     node *merge(node *a,node *b){ //skew heap
26         if(!a||!b)return a?b:
27         a->down(),b->down();
28         if(b->w<a->w)return merge(b,a);
29         swap(a->l,a->r);
30         a->l=merge(b,a->l);
31         return a;
32     }
33     void add_edge(int u,int v,T w){
34         if(u!=v)pq[v]=merge(pq[v],&(mem[m++]=
35             node(u,v,w)));
36     }
37     int find(int x,int *st){
38         return st[x]==x?x:st[x]=find(st[x],st);
39     }

```

```

37 T build(int root,int n){
38     T ans=0;int N=n,all=n;
39     for(int i=1;i<=N;++i){
40         if(i==root||!pq[i])continue;
41         while(pq[i]){
42             pq[i]->down(),E[i]=pq[i];
43             pq[i]=merge(pq[i]->l,pq[i]->r);
44             if(find(E[i]->u,id)!=find(i,id))
45                 break;
46         }
47         if(find(E[i]->u,id)==find(i,id))
48             continue;
49         ans+=E[i]->w;
50         if(find(E[i]->u,st)==find(i,st)){
51             if(pq[i])pq[i]->tag-=E[i]->w;
52             pq[++N]=pq[i];id[N]=N;
53             for(int u=find(E[i]->u,id);u!=i;u=
54                 find(E[u]->u,id)){
55                 if(pq[u])pq[u]->tag-=E[u]->w;
56                 id[find(u,id)]=N;
57                 pq[N]=merge(pq[N],pq[u]);
58             }
59             st[N]=find(i,st);
60             id[find(i,id)]=N;
61             }else st[find(i,st)]=find(E[i]->u,st);
62             ,--all;
63         }
64     }
65     return all==1?ans:-INT_MAX; //圖不連通就無解
66 }

```

4.16 穩定婚姻模板

```

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加入科系錄取名單中;
16            break;
17        }
18        if ( 目前科系已額滿 ) {
19            if ( 此考生成績比最低分數還高 ) {
20                依加權後分數高低順序將考生id加入科系錄取名單;
21                Q.push(被踢出的考生);
22            }
23        }
24    }
25 }

```

```

24 }

```

5 Linear_Programming

5.1 simplex

```

1 /*target:
2     max \sum_{j=1}^n A_{0,j}*x_j
3     condition:
4         \sum_{j=1}^n A_{i,j}*x_j <= A_{i,0} | i=1~m
5     x_j >= 0 | j=1~n
6 VDB = vector<double>*/
7 template<class VDB>
8 VDB simplex(int m,int n,vector<VDB> a){
9     vector<int> left(m+1), up(n+1);
10     iota(left.begin(), left.end(), n);
11     iota(up.begin(), up.end(), 0);
12     auto pivot = [&](int x, int y){
13         swap(left[x], up[y]);
14         auto k = a[x][y]; a[x][y] = 1;
15         vector<int> pos;
16         for(int j = 0; j <= n; ++j){
17             a[x][j] /= k;
18             if(a[x][j] != 0) pos.push_back(j);
19         }
20         for(int i = 0; i <= m; ++i){
21             if(a[i][y]==0 || i == x) continue;
22             k = a[i][y], a[i][y] = 0;
23             for(int j : pos) a[i][j] -= k*a[x][j];
24         }
25     };
26     for(int x,y;){
27         for(int i=x=1; i <= m; ++i)
28             if(a[i][0]<a[x][0]) x = i;
29         if(a[x][0]>=0) break;
30         for(int j=y=1; j <= n; ++j)
31             if(a[x][j]<a[x][y]) y = j;
32         if(a[x][y]>=0) return VDB();//infeasible
33         pivot(x, y);
34     }
35     for(int x,y;){
36         for(int j=y=1; j <= n; ++j)
37             if(a[0][j] > a[0][y]) y = j;
38         if(a[0][y]<=0) break;
39         x = -1;
40         for(int i=1; i<=m; ++i) if(a[i][y] > 0)
41             if(x == -1 || a[i][0]/a[i][y]
42                 < a[x][0]/a[x][y]) x = i;
43         if(x == -1) return VDB();//unbounded
44         pivot(x, y);
45     }
46     VDB ans(n + 1);
47     for(int i = 1; i <= m; ++i)
48         if(left[i] <= n) ans[left[i]] = a[i][0];
49     ans[0] = -a[0][0];
50     return ans;
51 }

```

6 Number_Theory

6.1 basic

```

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

```

```

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;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
154 //java code
155 //求sqrt(N)的連分數
156 public static void Pell(int n){
157     BigInteger N,p1,p2,q1,q2,a0,a1,a2,g1,g2,h1
        ,h2,p,q;
158     g1=q2=p1=BigInteger.ZERO;
159     h1=q1=p2=BigInteger.ONE;
160     a0=a1=BigInteger.valueOf((int)Math.sqrt
        (1.0*n));
161     BigInteger ans=a0.multiply(a0);
162     if(ans.equals(BigInteger.valueOf(n))){
163         System.out.println("No solution!");
164         return ;
165     }
166     while(true){
167         g2=a1.multiply(h1).subtract(g1);
168         h2=N.subtract(g2.pow(2)).divide(h1);
169         a2=g2.add(a0).divide(h2);
170         p=a1.multiply(p2).add(p1);
171         q=a1.multiply(q2).add(q1);
172         if(p.pow(2).subtract(N.multiply(q.pow
            (2))).compareTo(BigInteger.ONE)==0)
            break;
173         g1=g2;h1=h2;a1=a2;
174         p1=p2;p2=p;

```

```

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

```

6.2 bit_set

```

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 }

```

6.3 cantor_expansion

```

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

```

6.4 FFT

```

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&0x0F0F0F0FU)<<4)|((a&0xFF0F0F0FU)>>4);
9         a=((a&0x00FF00FFU)<<8)|((a&0xFFFF0000U)>>8);
10        a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)
            >>16);
11        return a>>(32-len);
12    }
13    void fft(bool is_inv,VT &in,VT &out,int N)
        {
14        int bitlen=__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    }
30 };

```

6.5 find_real_root

```

1 // an*x^n + ... + a1x + a0 = 0;
2 int sign(double x){
3     return x < -eps ? -1 : x > eps;
4 }
5
6 double get(const vector<double>&coef, double
    x){
7     double e = 1, s = 0;
8     for(auto i : coef) s += i*e, e *= x;
9     return s;
10 }
11
12 double find(const vector<double>&coef, int n
    , double lo, double hi){
13     double sign_lo, sign_hi;
14     if( !(sign_lo = sign(get(coef,lo))) )
15         return lo;
16     if( !(sign_hi = sign(get(coef,hi))) )
17         return hi;
18     if(sign_lo * sign_hi > 0) return INF;
19     for(int stp = 0; stp < 100 && hi - lo >
        eps; ++stp){

```

```

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

```

6.6 FWT

```

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

```

```

21         f[j+k+(1<<i)] = u-v, f[j+k] = u+v;
22     }
23     if(inverse) for(auto &a:f) a/=f.size();
24     return f;
25 }

```

6.7 LinearCongruence

```

1 pair<LL,LL> LinearCongruence(LL a[],LL b[],
    LL m[],int n) {
2     // a[i]*x = b[i] ( mod m[i] )
3     for(int i=0;i<n;++i) {
4         LL x, y, d = extgcd(a[i],m[i],x,y);
5         if(b[i]%d!=0) return make_pair(-1LL,0LL)
6             ;
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 }

```

6.8 Lucas

```

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 }

```

6.9 Matrix

```

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    }
51    T gas(){
52        vector<T> lazy(r,1);
53        bool sign=false;
54        for(int i=0;i<r;++i){
55            if( m[i][i]==0 ){
56                int j=i+1;
57                while(j<r&&!m[j][i])j++;
58                if(j==r)continue;
59                m[i].swap(m[j]);
60                sign=!sign;
61            }
62            for(int j=0;j<r;++j){
63                if(i==j)continue;
64                lazy[j]=lazy[j]*m[i][i];
65                T mx=m[j][i];
66                for(int k=0;k<c;++k)
67                    m[j][k]=m[j][k]*m[i][i]-m[i][k]*mx
68                    ;
69            }
70        }
71    }

```

```

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 };

```

6.10 MillerRobin

```

1 LL LLmul(LL a, LL b, const LL &mod) {
2     LL ans=0;
3     while(b) {
4         if(b&1) {
5             ans+=a;
6             if(ans>=mod) ans-=mod;
7         }
8         a<<=1, b>>=1;
9         if(a>=mod) a-=mod;
10    }
11    return ans;
12 }
13 LL mod_mul(LL a,LL b,LL m){
14     a%=m,b%=m; /* fast for m < 2^58 */
15     LL y=(LL)((double)a*b/m+0.5);
16     LL r=(a*b-y*m)%m;
17     return r<0?r+m:r;
18 }
19 template<typename T>
20 T pow(T a,T b,T mod){/*a^b%mod
21     T ans=1;
22     for(;b;a=mod_mul(a,a,mod),b>>=1)
23         if(b&1)ans=mod_mul(ans,a,mod);
24     return ans;
25 }
26 int sprp[3]={2,7,61};/*int範圍可解
27 int llsprr
    [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;u++)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 }

```

6.11 NTT

```

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;
36                     if(out[j]>=P)out[j]-=P;
37                     if(out[j+mh]<0)out[j+mh]+=P;
38                 }
39                 wi=wi*wn%P;
40             }
41             if(is_inv){
42                 for(int i=1;i<N/2;++i)swap(out[i],out[
43                     N-i]);
44                 T invn=pow_mod(N,P-2,P);
45                 for(int i=0;i<N;++i)out[i]=out[i]*invn
46                     %P;
47             }
48         }
49     };

```

6.12 Simpson

```

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;

```

```

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

```

6.13 外星模運算

```

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; //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]--;
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*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));

```

```

51 }
52 return 0;
53 }

```

6.14 數位統計

```

1 ll d[65], dp[65][2]; //up 區間是不是完整
2 ll dfs(int p,bool is8,bool up){
3     if(!p)return 1; // 回傳0是不是答案
4     if(!up&&dp[p][is8])return dp[p][is8];
5     int mx = up?d[p]:9; //可以用的有那些
6     ll ans=0;
7     for(int i=0;i<=mx;++i){
8         if( is8&&i==7 )continue;
9         ans += dfs(p-1,i==8,up&&i==mx);
10    }
11    if(!up)dp[p][is8]=ans;
12    return ans;
13 }
14 ll f(ll N){
15     int k=0;
16     while(N){ // 把數字先分解到陣列
17         d[++k] = N%10;
18         N/=10;
19     }
20     return dfs(k,false,true);
21 }

```

6.15 質因數分解

```

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 pollorroho(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; 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; 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     }
39     if(n!=1) v.push_back(n);
40 } }
41 }
42 void comfactor(const LL &n, vector<LL> &v) {
43     if(n<1e9) {
44         smallfactor(n,v);
45         return;
46     }
47     if(Isprime(n)) {
48         v.push_back(n);
49         return;
50     }
51     LL d;
52     for(int c=3; c<=n; c++) {
53         d = pollorroho(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 }

```


7 String

7.1 AC 自動機

```

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             ++S[o].ed;
30         }
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].fail;
44                 S[t].fail=~id?S[id].next[i]:0;
45                 S[t].efl=S[S[t].fail].ed?S[t].fail:S[t].fail].efl;
46                 q.push_back(t);
47                 ++qe;
48             }
49         }
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             if(!S[p].next[id])continue;
58             p=S[p].next[id];
59             ++S[p].cnt_dp; /*匹配成功則它所有後綴都
60             可以被匹配(DP計算)*/
61         }
62         for(i=qe-1;i>=0;--i){
63             ans+=S[q[i]].cnt_dp*S[q[i]].ed;
64             if(~S[q[i]].fail)S[q[i]].fail].cnt_dp+=S[q[i]].cnt_dp;
65         }
66         return ans;
67     }
68     /*多串匹配走efl邊並傳回所有字串被s匹配成功
69     的次數O(N*M^1.5)*/
70     int match_1(const char *s)const{
71         int ans=0,id,p=0,t;
72         for(int i=0;s[i];++i){
73             id=s[i]-L;
74             while(!S[p].next[id]&&p=S[p].fail;
75             if(!S[p].next[id])continue;
76             p=S[p].next[id];
77             if(S[p].ed)ans+=S[p].ed;
78             for(t=S[p].efl;~t;t=S[t].efl){
79                 ans+=S[t].ed; /*因為都走efl邊所以保證
80                 匹配成功*/
81             }
82             return ans;
83         }
84     }
85     /*枚舉(s的子字串nA)的所有相異字串各恰一次
86     並傳回次數O(N*M^(1/3))*/
87     int match_2(const char *s){
88         int ans=0,id,p=0,t;
89         ++vt;
90         /*把戳記vt+=1,只要vt沒溢位,所有S[p].
91         vis=vt就會變成false
92         這種利用vt的方法可以O(1)歸零vis陣列*/
93         for(int i=0;s[i];++i){
94             id=s[i]-L;
95             while(!S[p].next[id]&&p=S[p].fail;
96             if(!S[p].next[id])continue;
97             p=S[p].next[id];
98             if(S[p].ed&&S[p].vis!=vt){
99                 S[p].vis=vt;
100                 ans+=S[p].ed;
101             }
102             for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t].efl){
103                 S[t].vis=vt;
104                 ans+=S[t].ed; /*因為都走efl邊所以保證
105                 匹配成功*/
106             }
107             return ans;
108         }
109     }
110     /*把AC自動機變成真的自動機*/
111     void evolution(){
112         for(qs=1;qs!=qe;){
113             int p=q[qs++];
114             for(int i=0;i<R-L;++i)
115                 if(S[p].next[i]==0)S[p].next[i]=S[S[p].fail].next[i];

```

7.2 hash

```

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    T get_hash(int l,int r){ /*閉區間寫法,設編號
15        為0 ~ len-1*/
16        return (h[r+1]-(h[l]*h_base[r-l+1])%mod+
17            mod)%mod;

```

7.3 KMP

```

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;

```

7.4 manacher

```

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 }

```

7.5 minimal_string_rotation

```

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 }

```

7.6 reverseBWT

```

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(tots));
6     for(size_t i=0;i<bw.size();++i)
7         ranks[i] = tots[int(bw[i])]+1;
8 }
9 void firstCol(){
10    memset(first,0,sizeof(first));
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[int(c)] + ranks[i];
26     }while( i != begin );
27     return res;
28 }

```

7.7 suffix_array_lcp

```

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[tmp[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)--k;
31       while(s[i+k]==s[sa[rank[i]-1]+k])++k;
32       h[rank[i]]=k;
33     }
34     h[0]=0; // h[k]=Lcp(sa[k],sa[k-1]);

```

7.8 Z

```

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

```

8 Tarjan

8.1 dominator_tree

```

1 struct dominator_tree{
2   static const int MAXN=5005;
3   int n; // 1-base
4   vector<int> suc[MAXN],pre[MAXN];
5   int fa[MAXN],dfn[MAXN],id[MAXN],Time;
6   int semi[MAXN],idom[MAXN];
7   int anc[MAXN],best[MAXN]; // disjoint set
8   vector<int> dom[MAXN]; // dominator_tree
9   void init(int _n){
10     n=_n;
11     for(int i=1;i<=n;++i)suc[i].clear(),pre[
12     i].clear();
13   }
14   void add_edge(int u,int v){
15     suc[u].push_back(v);
16     pre[v].push_back(u);
17   }
18   void dfs(int u){
19     dfn[u]=++Time,id[Time]=u;
20     for(auto v:suc[u]){
21       if(dfn[v])continue;
22       dfs(v),fa[dfn[v]]=dfn[u];
23     }
24   }
25   int find(int x){
26     if(x==anc[x])return x;
27     int y=find(anc[x]);
28     if(semi[best[x]]>semi[best[anc[x]]])best
29     [x]=best[anc[x]];
30     return anc[x]=y;
31   }
32   void tarjan(int r){
33     Time=0;
34     for(int t=1;t<=n;++t){
35       dfn[t]=idom[t]=0; // u=r 或是 u無法到達r時
36       idom[id[u]]=0
37     }
38     dom[t].clear();
39     anc[t]=best[t]=semi[t]=t;
40     dfs(r);
41     for(int y=Time;y>=2;--y){
42       int x=fa[y],idy=id[y];
43       for(auto z:pre[idy]){
44         if(!z=dfn[z])continue;
45         find(z);
46         semi[y]=min(semi[y],semi[best[z]]);
47       }
48       dom[semi[y]].push_back(y);
49       anc[y]=x;
50       for(auto z:dom[x]){
51         find(z);
52         idom[z]=semi[best[z]]<x?best[z]:x;
53       }
54       dom[x].clear();
55     }
56     for(int u=2;u<=Time;++u){
57       if(idom[u]!=semi[u])idom[u]=idom[idom[
58       u]];
59       dom[id[idom[u]]].push_back(id[u]);
60     }
61   }
62   }
63   }
64   }
65   }
66   }
67   }
68   }
69   }
70   }
71   }
72   }
73   }
74   }
75   }
76   }
77   }
78   }
79   }
80   }
81   }
82   }
83   }
84   }
85   }
86   }
87   }
88   }
89   }
90   }
91   }
92   }
93   }
94   }
95   }
96   }
97   }
98   }
99   }
100  }

```

8.2 tnfsb017_2_sat

```

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 }

```

8.3 橋連通分量

```

1 #define N 1005
2 struct edge{
3   int u,v;
4   bool is_bridge;
5   edge(int u=0,int v=0):u(u),v(v),is_bridge
6   (0){}
7 };
8 vector<edge> E;
9 vector<int> G[N]; // 1-base
10 int low[N],vis[N],Time;
11 int bcc_id[N],bridge_cnt,bcc_cnt; // 1-base
12 int st[N],top; // BCC用
13 void add_edge(int u,int v){
14   G[u].push_back(E.size());
15   E.emplace_back(u,v);
16   G[v].push_back(E.size());
17   E.emplace_back(v,u);
18 }
19 void dfs(int u,int re=-1){ // u當前點 · re為u連
20   接前一個點的邊
21   int v;
22   low[u]=vis[u]=++Time;
23   st[top++]=u;
24   for(int e:G[u]){
25     v=E[e].v;
26     if(!vis[v]){
27       dfs(v,e^1); // e^1 反向邊
28       low[u]=min(low[u],low[v]);
29       if(vis[u]<low[v]){
30         E[e].is_bridge=1;
31         ++bridge_cnt;
32       }
33     }else if(vis[v]<vis[u]&&e!=re){
34       low[u]=min(low[u],vis[v]);
35     }
36   }
37   if(vis[u]==low[u]){ // 處理BCC
38     ++bcc_cnt; // 1-base
39     do bcc_id[v=st[--top]]=bcc_cnt; // 每個點
40     所在的BCC
41     while(v!=u);
42   }
43 }
44 void bcc_init(int n){
45   Time=bcc_cnt=bridge_cnt=top=0;
46   E.clear();
47   for(int i=1;i<=n;++i){
48     G[i].clear();
49     vis[i]=bcc_id[i]=0;
50   }
51 }

```

8.4 雙連通分量 & 割點

```

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 t, child=0;
10    low[u]=vis[u]=++Time;
11    st[top++]=u;
12    for(int v:G[u]){
13        if(!vis[v]){
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                do{
20                    bcc_id[t=st[--top]]=bcc_cnt;
21                    bcc[bcc_cnt].push_back(t);
22                }while(t!=v);
23                bcc_id[u]=bcc_cnt;
24                bcc[bcc_cnt].push_back(u);
25            }
26        }else if(vis[v]<vis[u]&&v!=pa) // 反向邊
27            low[u]=min(low[u], vis[v]);
28    } // u是dfs樹的根要特判
29    if(pa!=-1&&child<2) is_cut[u]=0;
30 }
31 void bcc_init(int n){
32     Time=bcc_cnt=top=0;
33     for(int i=1; i<=n; ++i){
34         G[i].clear();
35         is_cut[i]=vis[i]=bcc_id[i]=0;
36     }
37 }

```

9 Tree_problem

9.1 HeavyLight

```

1 #include<vector>
2 #define MAXN 100005
3 int siz[MAXN], max_son[MAXN], pa[MAXN], dep[
4     MAXN];
5 int link_top[MAXN], link[MAXN], cnt;
6 vector<int> G[MAXN];
7 void find_max_son(int u){
8     siz[u]=1;
9     max_son[u]=-1;
10    for(auto v:G[u]){
11        if(v==pa[u]) continue;
12        pa[v]=u;
13        dep[v]=dep[u]+1;
14        find_max_son(v);

```

```

14        if(max_son[u]==-1 || siz[v]>siz[max_son[u]
15            ]) max_son[u]=v;
16        siz[u]+=siz[v];
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
41        // 次區間處理
42        return dep[a]<dep[b]?a:b;

```

9.2 LCA

```

1 const int MAXN=100000; // 1-base
2 const int MLG=17; // Log2(MAXN)+1;
3 int pa[MLG+2][MAXN+5];
4 int dep[MAXN+5];
5 vector<int> G[MAXN+5];
6 void dfs(int x, int p=0) { // dfs(root);
7     pa[0][x]=p;
8     for(int i=0; i<=MLG; ++i)
9         pa[i+1][x]=pa[i][pa[i][x]];
10    for(auto &i:G[x]){
11        if(i==p) continue;
12        dep[i]=dep[x]+1;
13        dfs(i, x);
14    }
15 }
16 inline int jump(int x, int d){
17     for(int i=0; i<=MLG; ++i)
18         if((d>>i)&1) x=pa[i][x];
19     return x;
20 }
21 inline int find_lca(int a, int b){
22     if(dep[a]>dep[b]) swap(a, b);
23     b=jump(b, dep[b]-dep[a]);
24     if(a==b) return a;
25     for(int i=MLG; i>=0; --i){
26         if(pa[i][a]!=pa[i][b]){
27             a=pa[i][a];
28             b=pa[i][b];

```

```

29         }
30     }
31     return pa[0][a];
32 }

```

9.3 link_cut_tree

```

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
10    // tree的根
11    return nd[nd[x].pa].ch[0]!=x&&nd[nd[x].pa]
12        .ch[1]!=x;
13 }
14 void down(int x) { // 懶惰標記下推
15     if(nd[x].rev){
16         if(nd[x].ch[0]nd[nd[x].ch[0]].rev^=1;
17         if(nd[x].ch[1]nd[nd[x].ch[1]].rev^=1;
18         swap(nd[x].ch[0], nd[x].ch[1]);
19         nd[x].rev=0;
20     }
21 }
22 void push_down(int x) { // 所有祖先懶惰標記下推
23     if(!isroot(x)) push_down(nd[x].pa);
24     down(x);
25 }
26 void up(int x) { // 將子節點的資訊向上更新
27 }
28 void rotate(int x) { // 旋轉, 會自行判斷轉的方
29    // 向
30    int y=nd[x].pa, z=nd[y].pa, d=(nd[y].ch[1]==
31        x);
32    nd[x].pa=z;
33    if(!isroot(y)) nd[z].ch[nd[z].ch[1]==y]=x;
34    nd[y].ch[d]=nd[x].ch[d^1];
35    nd[nd[y].ch[d]].pa=y;
36    nd[y].pa=x, nd[x].ch[d^1]=y;
37    up(y), up(x);
38 }
39 void splay(int x) { // 將x伸展到splay tree的根
40    push_down(x);
41    while(!isroot(x)){
42        int y=nd[x].pa;
43        if(!isroot(y)){
44            int z=nd[y].pa;
45            if((nd[z].ch[0]==y)^(nd[y].ch[0]==x))
46                rotate(y);
47            else rotate(x);
48        }
49        rotate(x);
50    }
51 }
52 int access(int x){
53     int last=0;
54     while(x){
55         nd[x].ch[1]=last;
56         up(x);
57         last=x;
58         x=nd[x].pa;
59     }
60     return last; // access後splay tree的根
61 }
62 void access(int x, bool is=0) { // is=0就是一般
63    // 的access
64    int last=0;
65    while(x){
66        splay(x);
67        if(is&&nd[x].pa){
68            // printf("%d\n", max(nd[last].ma, nd[nd[
69                x].ch[1]].ma));
70        }
71        nd[x].ch[1]=last;
72        up(x);
73        last=x;
74        x=nd[x].pa;
75    }
76 }
77 void query_edge(int u, int v){
78     access(u);
79     access(v, 1);
80 }
81 void make_root(int x){
82     access(x), splay(x);
83     nd[x].rev^=1;
84 }
85 void make_root(int x){
86     nd[access(x)].rev^=1;
87     splay(x);
88 }
89 void cut(int x, int y){
90     make_root(x);
91     access(y);
92     splay(y);
93     nd[y].ch[0]=0;
94     nd[x].pa=0;
95 }
96 void cut_parents(int x){
97     access(x);
98     splay(x);
99     nd[nd[x].ch[0]].pa=0;
100    nd[x].ch[0]=0;
101 }
102 void link(int x, int y){
103     make_root(x);
104     nd[x].pa=y;
105 }
106 int find_root(int x){
107     x=access(x);
108     while(nd[x].ch[0]) x=nd[x].ch[0];
109     splay(x);
110     return x;
111 }
112 int query(int u, int v){
113     // 傳回uv路徑splay tree的根結點
114     // 這種寫法無法求LCA
115     make_root(u);
116     return access(v);
117 }
118 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，
    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 }

```

9.4 POJ_tree

```

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+1);
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 }

```

10 default

10.1 debug

```

1 //volatile
2 #ifndef 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

```

10.2 ext

```

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);

```

10.3 IncStack

```

1 //Magic
2 #pragma GCC optimize "Ofast"
3 //stack resize,change esp to rsp if 64-bit
    system
4 asm("mov %0,%%esp\n" :: "g"(mem+1000000));
5 -Wl,--stack,214748364 -trigraphs
6 #pragma comment(linker, "/STACK
    :1024000000,1024000000")
7 //linux stack resize
8 #include<sys/resource.h>
9 void increase_stack(){
10     const rlim_t ks=64*1024*1024;
11     struct rlimit rl;
12     int res=getrlimit(RLIMIT_STACK,&rl);
13     if(!res&&rl.rlim_cur<ks){

```

```

14         rlim_t cur=rl.rlim_cur;
15         res=setrlimit(RLIMIT_STACK,&rl);
16     }
17 }

```

10.4 input

```

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

```

11 language

11.1 CNF

```

1 #define MAXN 55
2 struct CNF{
3     int s,x,y;//s->xy | s->x, if y==-1
4     int cost;
5     CNF(){}
6     CNF(int s,int x,int y,int c):s(s),x(x),y(y)
        ,cost(c){}
7 };
8 int state;//規則數量
9 map<char,int> rule;//每個字元對應到的規則，
    小寫字母為終端字符
10 vector<CNF> cnf;
11 void init(){
12     state=0;
13     rule.clear();
14     cnf.clear();
15 }
16 void add_to_cnf(char s,const string &p,int
    cost){
17     //加入一個s -> <p>的文法，代價為cost
18     if(rule.find(s)==rule.end())rule[s]=state
        ++;
19     for(auto c:p)if(rule.find(c)==rule.end())
        rule[c]=state++;
20     if(p.size()==1){
21         cnf.push_back(CNF(rule[s],rule[p[0]],-1,
            cost));
22     }else{
23         int left=rule[s];
24         int sz=p.size();
25         for(int i=0;i<sz-2;++i){

```



```

26     cnf.push_back(CNF(left,rule[p[i]],
27         state,0));
28     left=state++;
29     cnf.push_back(CNF(left,rule[p[sz-2]],
30         rule[p[sz-1]],cost));
31 }
32 vector<long long> dp[MAXN][MAXN];
33 vector<bool> neg_INF[MAXN][MAXN]; //如果花費
34 //是負的可能會有無限小的情形
35 void relax(int l,int r,const CNF &c,long
36     long cost,bool neg_c=0){
37     if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x]
38         ||cost<dp[l][r][c.s])){
39         if(neg_c||neg_INF[l][r][c.x]){
40             dp[l][r][c.s]=0;
41             neg_INF[l][r][c.s]=true;
42         }else dp[l][r][c.s]=cost;
43     }
44 }
45 void bellman(int l,int r,int n){
46     for(int k=1;k<=state;++k)
47         for(auto c:cnf)
48             if(c.y==-1)relax(l,r,c,dp[l][r][c.x]+c
49                 .cost,k==n);
50 }
51 void cyk(const vector<int> &tok){
52     for(int i=0;i<(int)tok.size();++i){
53         for(int j=0;j<(int)tok.size();++j){
54             dp[i][j]=vector<long long>(state+1,
55                 INT_MAX);
56             neg_INF[i][j]=vector<bool>(state+1,
57                 false);
58         }
59         dp[i][i][tok[i]]=0;
60         bellman(i,i,tok.size());
61     }
62     for(int r=1;r<(int)tok.size();++r){
63         for(int l=r-1;l=0;--l){
64             for(int k=1;k<r;++k)
65                 for(auto c:cnf)
66                     if(~c.y)relax(l,r,c,dp[l][k][c.x]+
67                         dp[k+1][r][c.y]+c.cost);
68             bellman(l,r,tok.size());
69         }
70     }
71 }
72 }
73 }

```

12 other

12.1 WhatDay

```

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 }

```

12.2 上下最大正方形

```

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+l>d;++l){
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             ans=max(ans,r-l+1);
21         }
22     }
23     printf("%d\n",ans);
24 }

```

12.3 最大矩形

```

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){
16             st.push(make_pair(h,now.second));
17         }
18     }
19     return ans;
20 }

```

13 zformula

13.1 formula

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

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

$$(a) \quad I(G) + C_v(G) = |V|$$

$$(b) \quad M(G) + C_e(G) = |V|$$

- 對於連通二分圖：

$$(a) \quad I(G) = C_v(G)$$

$$(b) \quad M(G) = C_e(G)$$

- 最大權閉合圖：

$$(a) \quad C(u, v) = \infty, (u, v) \in E$$

$$(b) \quad C(S, v) = W_v, W_v > 0$$

$$(c) \quad C(v, T) = -W_v, W_v < 0$$

$$(d) \quad ans = \sum_{W_v > 0} W_v - flow(S, T)$$

- 最大密度子圖：

$$(a) \quad \text{求 } \max \left(\frac{W_e + W_v}{|V'|} \right), e \in E', v \in V'$$

$$(b) \quad U = \sum_{v \in V} 2W_v + \sum_{e \in E} W_e$$

$$(c) \quad C(u, v) = W_{(u,v)}, (u, v) \in E \cdot \text{雙向邊}$$

$$(d) \quad C(S, v) = U, v \in V$$

$$(e) \quad D_u = \sum_{(u,v) \in E} W_{(u,v)}$$

$$(f) \quad C(v, T) = U + 2g - D_v - 2W_v, v \in V$$

$$(g) \quad \text{二分搜 } g:$$

$$l = 0, r = U, eps = 1/n^2$$

$$\text{if}((U \times |V| - flow(S, T))/2 > 0) \quad l = mid$$

$$\text{else } r = mid$$

$$(h) \quad ans = \min_cut(S, T)$$

$$(i) \quad |E| = 0 \text{ 要特殊判斷}$$

- 弦圖：

- 點數大於 3 的環都要有一條弦
- 完美消除序列從後往前依次給每個點染色 \cdot 給每個點染上可以染的最小顏色
- 最大團大小 = 色數
- 最大獨立集：完美消除序列從前往後能選就選
- 最小團覆蓋：最大獨立集的點和他延伸的邊構成
- 區間圖是弦圖
- 區間圖的完美消除序列：將區間按造又端點由小到大排序
- 區間圖染色：用線段樹做

13.1.3 dinic 特殊圖複雜度

- 單位流： $O\left(\min\left(V^{3/2}, E^{1/2}\right)E\right)$
- 二分圖： $O\left(V^{1/2}E\right)$

13.1.4 0-1 分數規劃

$x_i = \{0, 1\} \cdot x_i$ 可能會有其他限制 \cdot 求 $\max \left(\frac{\sum B_i x_i}{\sum C_i x_i} \right)$

- $D(i, g) = B_i - g \times C_i$
- $f(g) = \sum D(i, g) x_i$
- $f(g) = 0$ 時 g 為最佳解 $\cdot f(g) < 0$ 沒有意義
- 因為 $f(g)$ 單調可以二分搜 g
- 或用 Dinkelbach 通常比較快

```

1 binary_search(){
2     while(r-l>eps){
3         g=(l+r)/2;
4         for(i:所有元素)D[i]=B[i]-g*C[i]; //D(i,g)
5         找出一組合法x[i]使f(g)最大;
6         if(f(g)>0) l=g;
7         else r=g;
8     }
9     Ans = r;
10 }
11 Dinkelbach(){
12     g=任意狀態(通常設為0);
13     do{
14         Ans=g;
15         for(i:所有元素)D[i]=B[i]-g*C[i]; //D(i,g)
16         找出一組合法x[i]使f(g)最大;
17         p=0,q=0;
18         for(i:所有元素)
19             if(x[i])p+=B[i],q+=C[i];
20         g=p/q; //更新解, 注意q=0的情況
21     }while(abs(Ans-g)>EPS);
22     return Ans;
23 }

```

13.1.5 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 $H_n = \ln(n) + \gamma + 1/(2n) - 1/(12n^2) + 1/(120n^4)$
- $\gamma = 0.57721566490153286060651209008240243104218$
- 格雷碼 = $n \oplus (n >> 1)$
- $SG(A+B) = SG(A) \oplus SG(B)$
- 逆轉矩陣 $M(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

13.1.6 基本數論

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

13.1.7 排組公式

1. k 卡特蘭 $\frac{C_n^{kn}}{n(k-1)+1} \cdot C_m^n = \frac{n!}{m!(n-m)!}$
2. $H(n, m) \cong x_1 + x_2 + \dots + x_n = k, num = C_k^{n+k-1}$
3. Stirling number of 2^{nd} , n 人分 k 組方法數目
 - (a) $S(0, 0) = S(n, n) = 1$
 - (b) $S(n, 0) = 0$
 - (c) $S(n, k) = kS(n-1, k) + S(n-1, k-1)$
4. Bell number, n 人分任意多組方法數目

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

5. Derangement, 錯排, 沒有人在自己位置上

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

6. Binomial Equality

- (a) $\sum_k \binom{r}{m+k} \binom{s}{n-k} = \binom{r+s}{m+n}$
- (b) $\sum_k \binom{m}{m+k} \binom{n}{n-k} = \binom{l+s}{l+m+n}$
- (c) $\sum_k \binom{m}{m+k} \binom{s+k}{n} (-1)^k = (-1)^{l+m} \binom{s-m}{n-l}$
- (d) $\sum_{k \leq l} \binom{l-k}{m} \binom{s}{k-n} (-1)^k = (-1)^{l+m} \binom{s-m-1}{l-n-m}$
- (e) $\sum_{0 \leq k \leq l} \binom{l-k}{m} \binom{q+k}{n} = \binom{l+q+1}{m+n+1}$
- (f) $\binom{r}{k} = (-1)^k \binom{k-r-1}{k}$
- (g) $\binom{r}{m} \binom{m}{k} = \binom{r}{m-k} \binom{r-k}{k}$
- (h) $\sum_{k \leq n} \binom{r+k}{k} = \binom{r+n+1}{n}$
- (i) $\sum_{0 \leq k \leq n} \binom{m}{m} = \binom{m+1}{m+1}$
- (j) $\sum_{k \leq m} \binom{m+r}{k} x^k y^{m-k} = \sum_{k \leq m} \binom{-r}{k} (-x)^k (x+y)^{m-k}$

13.1.8 幕次, 幕次和

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

13.1.9 Burnside's lemma

1. $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$
2. $X^g = t^{c(g)}$
3. 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。
4. 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 $\frac{1}{57} (3^6 + 6 \times 3^3 + 3 \times 3^4 + 8 \times 3^2 + 6 \times 3^3) = \frac{24}{57}$

13.1.10 Count on a tree

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

13.2 java

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
```

```
21 }
22 }
```

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());
```

14

14.1 ganadoQuote

```
1 ¡Allí está!
2 ¡Un forastero!
3 ¡Agarrenlo!
4 ¡Os voy a romper a pedazos!
5 ¡Cógelo!
6 ¡Te voy a hacer picadillo!
7 ¡Te voy a matar!
8 ¡Míralo, está herido!
9 ¡Sos cerdo!
10 ¿Dónde estás?
11 ¡Detrás de ti, imbécil!
12 ¡No dejes que se escape!
13 ¡Basta, hijo de puta!
14 Lord Saddler...
15
16 ¡Mátalo!
17 ¡Allí está!
18 Morir es vivir.
19 ¡Síííí, ¡Quiero matar!
20 Muere, muere, muere...
21 Cerebros, cerebros, cerebros...
22 Cógedlo, cógedlo, cógedlo...
23 Lord Saddler...
24 Dieciséis.
25
26 ¡Va por él!
27 ¡Muérete!
28 ¡Cógelo!
29 ¡Te voy a matar!
30 ¡Bloqueale el paso!
31 ¡Te cogí!
32 ¡No dejes que se escape!
33
34 ¿Qué carajo estás haciendo aquí? ¡Lárgate,
35 cabrón!
36 Hay un rumor de que hay un extranjero entre
37 nosotros.
38 Nuestro jefe se encargará de la rata.
39 Su "Las Plagas" es mucho mejor que la
40 nuestra.
41 Tienes razón, es un hombre.
42 Usa los músculos.
43 Se vuelve loco!
44 ¡Hey, acá!
45 ¡Por aquí!
46 ¡El Gigante!
47 ¡Del Lago!
48 ¡Cógelo!
49 ¡Cógenlo!
50 ¡Allí!
51 ¡Rápido!
52 ¡Empieza a rezar!
53 ¡Mátalos!
54 ¡Te voy a romper en pedazos!
55 ¡La campana!
56 Ya es hora de rezar.
57 Tenemos que irnos.
58 ¡Maldita sea, mierda!
59 ¡Ya es hora de aplastar!
60 ¡Mierda!
61 ¡Puedes correr, pero no te puedes esconder!
62 ¡Sos cerdo!
63 ¡Está en la trampa!
64 ¡Ah, que madre!
65 ¡Vámonos!
```

```

63 ¡Ándale!
64 ¡Cabron!
65 ¡Coño!
66 ¡Agárrenlo!
67 Cógerlo, Cógerlo...
68 ¡Allí está, mátalo!
69 ¡No dejas que se escape de la isla vivo!
70 ¡Hasta luego!
71 ¡Rápido, es un intruso!

```

14.2

```

1  /*****
2  L'Internationale,
3      Sera Le genre humain.
4
5
6
7
8
9
10
11
12
13
14
15
16 *****/
17 Вставай, проклятьем заклеимённый,
18 Весь мир голодных и рабов!
19 Кипит наш разум возмущённый
20 И в смертный бой вести готов.
21 Весь мир насилиья мы разрушим
22 До основания, а затем
23 Мы наш, мы новый мир построим, —
24 Кто был ничем, тот станет всем.

```

```

25 Chorus
26 Это есть наш последний
27 И решительный бой;
28 С Интернационалом
29 Воспрянет род людской!
30
31 Никто не даст нам избавленья:
32 Ни бог, ни царь и не герой!
33 Добьёмся мы освобожденья
34 Своею собственной рукой.
35 Чтоб свергнуть гнёт рукой умелой,
36 Отвоевать своё добро, —
37 Вдуйте горн и куйте смело,
38 Пока железо горячо!
39
40 Chorus
41 Довольно кровь сосать, вампиры,
42 Тюрьмой, налогом, нищетой!
43 У вас — вся власть, все блага мира,
44 А наше право — звук пустой !
45 Мы жизнь построим по-иному —
46 И вот наш лозунг боевой:
47 Вся власть народу трудовому!
48 А дармоедов всех долой!

```

```

52 Chorus
53
54 Презренны вы в своём богатстве,
55 Угля и стали короли!
56 Вы ваши троны, туеядцы,
57 На наших спинах возвели.
58 Заводы, фабрики, палаты —
59 Всё нашим создано трудом.
60 Пора! Мы требуем возврата
61 Того, что взято грабежом.
62
63 Chorus
64
65 Довольно королям в угоду
66 Дурманить нас в чаду войны!
67 Война тиранам! Мир Народу!
68 Бастуйте, армии сыны!
69 Когда ж тираны нас заставят
70 В бою геройски пасть за них —
71 Убийцы, в вас тогда направим
72 Мы жерла пушек боевых!
73
74 Chorus
75
76 Лишь мы, работники всемирной
77 Великой армии труда,
78 Владеть землёй имеем право,
79 Но паразиты — никогда!
80 И если гром великий грянет
81 Над сворой псов и палачей, —
82 Для нас всё так же солнце станет
83 Сиять огнём своих лучей.
84
85 Chorus

```

14.3 保佑

```

1  //      _ooθoo_
2  //      o8888888o
3  //      88" . "88
4  //      (/ - - \)
5  //      θ \ = / θ
6  //
7  //
8  //
9  //
10 //
11 //
12 //
13 //
14 //
15 //
16 //
17 //
18 //
19 //
20 // ~~~~~
21 //      佛祖保佑      永無BUG
22
23
24
25
26

```

```

27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45 #
46 #
47 #
48 #
49 #
50 #
51 #
52 #
53 #
54 #
55 #
56 #
57 #
58 #
59 #
60 #
61 #
62 #
63 #
64 #
65 #
66
67
68 // ##      #####
69 // ##      ##
70 // ##      ##
71 // ##      ##
72 // ##      ##
73 // ##      ##
74 // ##      ##
75 // ##      ##
76 // #####
77 //      ##      ##
78 //      ##      ##
79 //      ##      ##
80 //      ##      ##
81 //      ##      ##
82 //      ##      ##
83 //      ##      ##
84 // #####      ##
85 //
86 //      元首保佑 永無BUG
87

```

```

88 //
89 //
90 //
91 //
92 //
93 //
94 //
95 //
96 //
97 //
98 //
99 //
100 //
101 //

```

神獸保佑 永無BUG

ACM ICPC TEAM REFERENCE - MADE IN ABYSS

Contents

1 Computational_Geometry	1		
1.1 Geometry	1		
1.2 SmallestCircle	3		
1.3 最近點對	3		
2 Data_Structure	3		
2.1 DLX	3		
2.2 Dynamic_KD_tree	3		
2.3 kd_tree_replace_segment_tree	4		
2.4 reference_point	5		
2.5 skew_heap	5		
2.6 undo_disjoint_set	5		
2.7 整體二分	5		
3 Flow	5		
3.1 dinic	5		
3.2 Gomory_Hu	5		
3.3 ISAP_with_cut	5		
		4 Graph	6
		4.1 Augmenting_Path	6
		4.2 Augmenting_Path_multiple	6
		4.3 blossom_matching	6
		4.4 graphISO	6
		4.5 KM	7
		4.6 MaximumClique	7
		4.7 MinimumMeanCycle	7
		4.8 Rectilinear_MST	7
		4.9 treeISO	7
		4.10 一般圖最小權完美匹配	8
		4.11 全局最小割	8
		4.12 平面圖判定	8
		4.13 弦圖完美消除序列	8
		4.14 最小斯坦納樹 DP	8
		4.15 最小樹形圖_朱劉	9
		4.16 穩定婚姻模板	9
		5 Linear_Programming	9
		5.1 simplex	9
		6 Number_Theory	9
		6.1 basic	9
		6.2 bit_set	10
		6.3 cantor_expansion	10
		6.4 FFT	10
		6.5 find_real_root	10
		6.6 FWT	11
		6.7 LinearCongruence	11
		6.8 Lucas	11
		6.9 Matrix	11
		6.10 MillerRobin	11
		6.11 NTT	12
		6.12 Simpson	12
		6.13 外星模運算	12
		6.14 數位統計	12
		6.15 質因數分解	12
		7 String	13
		7.1 AC 自動機	13
		7.2 hash	13
		7.3 KMP	13
		7.4 manacher	13
		7.5 minimal_string_rotation	13
		7.6 reverseBWT	13
		7.7 suffix_array_lcp	14
		7.8 Z	14
		8 Tarjan	14
		8.1 dominator_tree	14
		8.2 tnfsb017_2_sat	14
		8.3 橋連通分量	14
		8.4 雙連通分量 & 割點	15
		9 Tree_problem	15
		9.1 HeavyLight	15
		9.2 LCA	15
		9.3 link_cut_tree	15
		9.4 POJ_tree	16
		10 default	16
		10.1 debug	16
		10.2 ext	16
		10.3 IncStack	16
		10.4 input	16
		11 language	16
		11.1 CNF	16
		12 other	17
		12.1 WhatDay	17
		12.2 上下最大正方形	17
		12.3 最大矩形	17
		13 zformula	17
		13.1 formula	17
		13.1.1 Pick 公式	17
		13.1.2 圖論	17
		13.1.3 dinic 特殊圖複雜度	17
		13.1.4 0-1 分數規劃	17
		13.1.5 學長公式	17
		13.1.6 基本數論	17
		13.1.7 排組公式	18
		13.1.8 冪次, 冪次和	18
		13.1.9 Burnside's lemma	18
		13.1.10 Count on a tree	18
		13.2 java	18
		13.2.1 文件操作	18
		13.2.2 优先队列	18
		13.2.3 Map	18
		13.2.4 sort	18
		14	18
		14.1 ganadoQuote	18
		14.2	19
		14.3 保佑	19