

1 Computational Geometry

1.1 delaunay

```

1 template<class T>
2 class Delaunay{
3     struct PT:public point<T>{
4         int g[2];
5         PT(const point<T> &p):
6             point<T>(p){ g[0]=g[1]=-1; }
7     };
8     static bool cmp(const PT &a,const PT &b){
9         return a.x<b.x|| (a.x==b.x&&a.y<b.y);
10    }
11    struct edge{
12        int v,g[2];
13        edge(int v,int g0,int g1):
14            v(v){g[0]=g0,g[1]=g1;}
15    };
16    vector<PT> S;
17    vector<edge> E;
18    bool convex(int &from,int to,T LR){
19        for(int i=0;i<2;++i){
20            int c = E[S[from].g[i]].v;
21            auto A=S[from]-S[to], B=S[c]-S[to];
22            T v = A.cross(B)*LR;
23            if(v>0|| (v==0&&B.abs2())<A.abs2()))
24                return from = c, true;
25        }
26        return false;
27    }
28    void addEdge(int v,int g0,int g1){
29        E.emplace_back(v,g0,g1);
30        E[E.back().g[0]].g[1] = E.size()-1;
31        E[E.back().g[1]].g[0] = E.size()-1;
32    }
33    void climb(int &p, int e, int n, int nl,
34              int nr, int LR){
35        for(int i=E[e].g[LR]; (S[nr]-S[nl]).
36            cross(S[E[i].v]-S[nl])>0;){
37            if(inCircle(S[E[i].v],S[nl],S[nr],S[E[
38                E[i].g[LR]].v])>0)
39                { p = i; break; }
40            for(int j=0;j<4;++j)
41                E[E[i^j/2].g[j%2*1]].g[j%2] = E[i^j
42                    /2].g[j%2];
43            int j=i; i=E[i].g[LR];
44            E[j].g[0]=E[j].g[1]=E[j^1].g[0]=E[j
45                ^1].g[1]=-1;
46        }
47    }
48    T det3(T a11,T a12,T a13,T a21,T a22,T a23
49        ,T a31,T a32,T a33){
50        return a11*(a22*a33-a32*a23)-a12*(a21*
51            a33-a31*a23)+a13*(a21*a32-a31*a22);
52    }
53    int inCircle(const PT &a, const PT &b,
54                const PT &c, const PT &p){
55        T as = a.abs2(), bs = b.abs2(), cs = c.abs2
56            (), ps = p.abs2();
57        T res = a.x * det3(b.y,bs,1,c.y,cs,1,p.y,ps
58            ,1)
59            -a.y * det3(b.x,bs,1,c.x,cs,1,p.x,ps,1)

```

```

50 +as * det3(b.x,b.y,1,c.x,c.y,1,p.x,p.y,1)
51 -det3(b.x,b.y,bs,c.x,c.y,cs,p.x,p.y,ps);
52     return res<0 ? 1 : (res>0 ? -1 : 0);
53 }
54 void divide(int l, int r){
55     if(l>=r)return;
56     if(l+1==r){
57         int A=S[l].g[0]=S[l].g[1]=E.size();
58         E.emplace_back(r,A,A);
59         int B=S[r].g[0]=S[r].g[1]=E.size();
60         E.emplace_back(l,B,B);
61         return;
62     }
63     int mid = (l+r)/2;
64     divide(l,mid), divide(mid+1, r);
65     int nl = mid, nr = mid+1;
66     for(;;){
67         if(convex(nl,nr,1)) continue;
68         if(S[nr].g[0]!=-1&&convex(nr,nl,-1))
69             continue;
70         break;
71     }
72     addEdge(nr,S[nl].g[0],S[nl].g[1]);
73     S[nl].g[1] = E.size()-1;
74     if(S[nr].g[0]==-1){
75         addEdge(nl,E.size(),E.size());
76         S[nr].g[1] = E.size()-1;
77     }else addEdge(nl,S[nr].g[0],S[nr].g[1]);
78     S[nr].g[0] = E.size()-1;
79     int cl = nl, cr = nr;
80     for(;;){
81         int pl=-1, pr=-1, side;
82         climb(pl,E.size()-2,nl,nl,nr,1);
83         climb(pr,E.size()-1,nr,nl,nr,0);
84         if(pl==-1&&pr==1) break;
85         if(pl==1||pr==1) side = pl==1;
86         else side=inCircle(S[E[pl].v],S[nl],S[
87             nr],S[E[pr].v])<0;
88         if(side){
89             nr = E[pr].v;
90             addEdge(nr,E.size()-2,E[E.size()-2].g[1]);
91             addEdge(nl,E[pr^1].g[0],pr^1);
92         }else{
93             nl = E[pl].v;
94             addEdge(nr,pl^1,E[pl^1].g[1]);
95             addEdge(nl,E[E.size()-2].g[0],E.size()-2);
96         }
97     }
98     if(cl==nl&&cr==nr) return; //Collinearity
99     S[nl].g[0] = E.size()-2;
100    S[nr].g[1] = E.size()-1;
101 }
102 public:
103 void solve(const vector<point<T>> &P){
104     S.clear(), E.clear();
105     for(const auto &p:P) S.emplace_back(p);
106     sort(S.begin(),S.end(),cmp);
107     divide(0,int(S.size())-1);
108 }
109 vector<pair<int,int>> getEdge(){
110     vector<pair<int,int>> res;
111     for(size_t i=0;i<E.size();i+=2)
112         if(E[i].g[0]!=-1)
113             res.emplace_back(E[i].v,E[i^1].v);
114     return res;
115 }

```

1.2 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()const{ //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 }

```

```

53 T dis2(const point<T> &p,bool is_segment
54     =0)const{ //點跟直線/線段的距離平方
55     point<T> v=p2-p1,v1=p-p1;
56     if(is_segment){
57         point<T> v2=p-p2;
58         if(v.dot(v1)<=0)return v1.abs2();
59         if(v.dot(v2)>=0)return v2.abs2();
60     }
61     T tmp=v.cross(v1);
62     return tmp*tmp/v.abs2();
63 }
64 T seg_dis2(const line<T> &l)const{ //兩線段
65     距離平方
66     return min(dis2(l.p1,1),dis2(l.p2,1),l.
67         dis2(p1,1),l.dis2(p2,1));
68 }
69 point<T> projection(const point<T> &p)
70     const{ //點對直線的投影
71     point<T> n=(p2-p1).normal();
72     return p-n*(p-p1).dot(n)/n.abs2();
73 }
74 point<T> mirror(const point<T> &p)const{
75     //點對直線的鏡射 · 要先呼叫pton轉成一般式
76     point<T> R;
77     T d=a*b+b*b;
78     R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
79     R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
80     return R;
81 }
82 bool equal(const line &l)const{ //直線相等
83     return ori(l.p1)==0&&ori(l.p2)==0;
84 }
85 bool parallel(const line &l)const{
86     return (p1-p2).cross(l.p1-l.p2)==0;
87 }
88 bool cross_seg(const line &l)const{
89     return (p2-p1).cross(l.p1-p1)*(p2-p1).
90         cross(l.p2-p1)<=0; //直線是否交線段
91 }
92 int line_intersect(const line &l)const{ //
93     直線相交情況 · -1無限多點 · 1交於一點 · 0
94     不相交
95     return parallel(l)?(ori(l.p1)==0?-1:0)
96         :1;
97 }
98 int seg_intersect(const line &l)const{
99     T c1=ori(l.p1), c2=ori(l.p2);
100    T c3=1.ori(p1), c4=1.ori(p2);
101    if(c1==0&&c2==0){ //共線
102        bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
103        T a3=1.btw(p1),a4=1.btw(p2);
104        if(b1&&b2&&a3==0&&a4==0) return 2;
105        if(b1&&b2&&a3>=0&&a4==0) return 3;
106        if(b1&&b2&&a3>=0&&a4>=0) return 0;
107        return -1; //無限交點
108    }else if(c1*c2<=0&&c3*c4<=0)return 1;
109    return 0; //不相交
110 }
111 point<T> line_intersection(const line &l)
112     const{ //直線交點*/
113     point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
114     //if(a.cross(b)==0)return INF;
115     return p1+a*(s.cross(b)/a.cross(b));
116 }

```

```

108 point<T> seg_intersection(const line &l)
109     const{//線段交點
110     int res=seg_intersect(l);
111     if(res<=0) assert(0);
112     if(res==2) return p1;
113     if(res==3) return p2;
114     return line_intersection(l);
115 };
116 template<typename T>
117 struct polygon{
118     polygon(){}
119     vector<point<T> > p;//逆時針順序
120     T area()const{//面積
121         T ans=0;
122         for(int i=p.size()-1,j=0;j<(int)p.size();i=j++){
123             ans+=p[i].cross(p[j]);
124             return ans/2;
125         }
126     point<T> center_of_mass()const{//重心
127         T cx=0,cy=0,w=0;
128         for(int i=p.size()-1,j=0;j<(int)p.size();i=j++){
129             T a=p[i].cross(p[j]);
130             cx+=(p[i].x+p[j].x)*a;
131             cy+=(p[i].y+p[j].y)*a;
132             w+=a;
133         }
134         return point<T>(cx/3/w,cy/3/w);
135     }
136     char ahas(const point<T>& t)const{//點是否在簡單多邊形內，是的話回傳1，在邊上回傳-1，否則回傳0
137         bool c=0;
138         for(int i=0,j=p.size()-1;i<p.size();j=i++){
139             if(line<T>(p[i],p[j]).point_on_segment(t))return -1;
140             else if((p[i].y>t.y)!(p[j].y>t.y)&&t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j].y-p[i].y)+p[i].x)
141                 c=!c;
142             return c;
143         }
144     }
145     char point_in_convex(const point<T>&x)
146         const{
147         int l=1,r=(int)p.size()-2;
148         while(l<r){//點是否在凸多邊形內，是的話回傳1，在邊上回傳-1，否則回傳0
149             int mid=(l+r)/2;
150             T a1=(p[mid]-p[0]).cross(x-p[0]);
151             T a2=(p[mid+1]-p[0]).cross(x-p[0]);
152             if(a1>0&&a2<=0){
153                 T res=(p[mid+1]-p[mid]).cross(x-p[mid]);
154                 return res>0?1:(res>=0?-1:0);
155             }else if(a1<0)r=mid-1;
156             else l=mid+1;
157         }
158         return 0;
159     }
160     vector<T> getA()const{//凸包邊對x軸的夾角
161         vector<T>res;//一定是遞增的
162         for(size_t i=0;i<p.size();++i)
163             res.push_back((p[(i+1)%p.size()]-p[i]).getA());
164         return res;
165     }
166     bool line_intersect(const vector<T>&A,
167         const line<T> &l)const{//O(LogN)
168         int f1=upper_bound(A.begin(),A.end(),(l.p1-l.p2).getA())-A.begin();
169         int f2=upper_bound(A.begin(),A.end(),(l.p2-l.p1).getA())-A.begin();
170         return 1.cross_seg(line<T>(p[f1],p[f2]))
171             ;
172     }
173     polygon cut(const line<T> &l)const{//凸包對直線切割，得到直線L左側的凸包
174         polygon ans;
175         for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
176             if(1.ori(p[i])>=0){
177                 ans.p.push_back(p[i]);
178                 if(1.ori(p[j])<0)
179                     ans.p.push_back(l.line_intersection(line<T>(p[i],p[j])));
180             }else if(1.ori(p[j])>0)
181                 ans.p.push_back(l.line_intersection(line<T>(p[i],p[j])));
182             return ans;
183         }
184     }
185     static bool monotone_chain_cmp(const point<T>& a,const point<T>& b){//凸包排序函數
186         return (a.x<b.x)||((a.x==b.x&&a.y<b.y));
187     }
188     void monotone_chain(vector<point<T> > &s){
189         //凸包
190         sort(s.begin(),s.end(),monotone_chain_cmp);
191         p.resize(s.size()+1);
192         int m=0;
193         for(size_t i=0;i<s.size();++i){
194             while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
195             p[m++]=s[i];
196         }
197         for(int i=s.size()-2,t=m+1;i>=0;--i){
198             while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
199             p[m++]=s[i];
200         }
201         T diam()const{//直徑
202             int n=p.size(),t=1;
203             T ans=0;p.push_back(p[0]);
204             for(int i=0;i<n;i++){
205                 point<T> now=p[i+1]-p[i];
206                 while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t+1)%n;
207                 ans=max(ans,(p[i]-p[t]).abs2());
208             }
209             return p.pop_back(),ans;
210         }
211         T min_cover_rectangle()const{//最小覆蓋矩形
212             int n=p.size(),t=1,r=1,l;
213             if(n<3)return 0;//也可以做最小周長矩形
214             T ans=1e99;p.push_back(p[0]);
215             for(int i=0;i<n;i++){
216                 point<T> now=p[i+1]-p[i];
217                 while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t+1)%n;
218                 while(now.dot(p[r+1]-p[i])>now.dot(p[r]-p[i]))r=(r+1)%n;
219                 if(!l)l=r;
220                 while(now.dot(p[l+1]-p[i])<=now.dot(p[l]-p[i]))l=(l+1)%n;
221                 T d=now.abs2();
222                 T tmp=now.cross(p[t]-p[i])*(now.dot(p[r]-p[i])-now.dot(p[l]-p[i]))/d;
223                 ans=min(ans,tmp);
224             }
225             return p.pop_back(),ans;
226         }
227         T dis2(polygon &p1){//凸包最近距離平方
228             vector<point<T> > &P=p,&Q=p1.p;
229             int n=P.size(),m=Q.size(),l=0,r=0;
230             for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
231             for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
232             P.push_back(P[0]),Q.push_back(Q[0]);
233             T ans=1e99;
234             for(int i=0;i<n;++i){
235                 while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])<0)r=(r+1)%m;
236                 ans=min(ans,line<T>(P[l],P[l+1]).seg_dis2(line<T>(Q[r],Q[r+1])));
237                 l=(l+1)%n;
238             }
239             return P.pop_back(),Q.pop_back(),ans;
240         }
241         static char sign(const point<T>&t){
242             return (t.y==0?t.x:t.y)<0;
243         }
244         static bool angle_cmp(const line<T>& A,
245             const line<T>& B){
246             point<T> a=A.p2-A.p1,b=B.p2-B.p1;
247             return sign(a)<sign(b)||((sign(a)==sign(b)&&a.cross(b)>0));
248         }
249         int halfplane_intersection(vector<line<T> > &s){//半平面交
250             sort(s.begin(),s.end(),angle_cmp);//線段左側為該線段半平面
251             int L,R,n=s.size();
252             vector<point<T> > px(n);
253             vector<line<T> > q(n);
254             q[L=R=0]=s[0];
255             for(int i=1;i<n;++i){
256                 while(L<R&&s[i].ori(px[R-1])<=0)--R;
257                 while(L<R&&s[i].ori(px[L])<=0)+L;
258                 q[++R]=s[i];
259                 if(q[R].parallel(q[R-1])){
260                     --R;
261                     if(q[R].ori(s[i].p1)>0)q[R]=s[i];
262                 }
263                 if(L<R)px[R-1]=q[R-1].line_intersection(q[R]);
264             }
265             while(L<R&&q[L].ori(px[R-1])<=0)--R;
266             p.clear();
267             if(R-L<=1)return 0;
268             px[R]=q[R].line_intersection(q[L]);
269             for(int i=L;i<=R;++i)p.push_back(px[i]);
270             return R-L+1;
271         }
272     };
273     template<typename T>
274     struct triangle{
275         point<T> a,b,c;
276         triangle(){}
277         triangle(const point<T> &a,const point<T> &b,const point<T> &c):a(a),b(b),c(c){}
278         T area()const{
279             T t=(b-a).cross(c-a)/2;
280             return t>0?t:-t;
281         }
282         point<T> barycenter()const{//重心
283             return (a+b+c)/3;
284         }
285         point<T> circumcenter()const{//外心
286             static line<T> u,v;
287             u.p1=(a+b)/2;
288             u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-b.x);
289             v.p1=(a+c)/2;
290             v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-c.x);
291             return u.line_intersection(v);
292         }
293         point<T> incenter()const{//內心
294             T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2()),C=sqrt((a-b).abs2());
295             return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+B*b.y+C*c.y)/(A+B+C);
296         }
297         point<T> perpencenter()const{//垂心
298             return barycenter()*3-circumcenter()*2;
299         }
300     };
301     template<typename T>
302     struct point3D{
303         T x,y,z;
304         point3D(){}
305         point3D(const T&x,const T&y,const T&z):x(x),y(y),z(z){}
306         point3D operator+(const point3D &b)const{
307             return point3D(x+b.x,y+b.y,z+b.z);
308         }
309         point3D operator-(const point3D &b)const{
310             return point3D(x-b.x,y-b.y,z-b.z);
311         }
312         point3D operator*(const T &b)const{
313             return point3D(x*b,y*b,z*b);
314         }
315         point3D operator/(const T &b)const{
316             return point3D(x/b,y/b,z/b);
317         }
318         bool operator==(const point3D &b)const{
319             return x==b.x&&y==b.y&&z==b.z;
320         }
321         T dot(const point3D &b)const{
322             return x*b.x+y*b.y+z*b.z;
323         }
324         point3D cross(const point3D &b)const{
325             return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
326         }
327         T abs2()const{//向量長度的平方
328             return dot(*this);
329         }
330         T area2(const point3D &b,const point3D &c)const{//和b、原點圍成面積的平方
331             return cross(b,c).abs2()/4;
332         }
333     };

```

```

321     return cross(b).abs2()/4;
322 };
323 template<typename T>
324 struct line3D{
325     point3D<T> p1,p2;
326     line3D(const point3D<T> &p1,const point3D<
327         T> &p2):p1(p1),p2(p2){}
328     T dis2(const point3D<T> &p,bool is_segment
329         =0)const{//點跟直線/線段的距離平方
330         point3D<T> v=p2-p1,v1=p-p1;
331         if(is_segment){
332             point3D<T> v2=p-p2;
333             if(v.dot(v1)<=0)return v1.abs2();
334             if(v.dot(v2)>=0)return v2.abs2();
335         }
336         point3D<T> tmp=v.cross(v1);
337         return tmp.abs2()/v.abs2();
338     }
339     pair<point3D<T>,point3D<T> > closest_pair(
340         const line3D<T> &l)const{
341         point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
342         point3D<T> N=v1.cross(v2),ab(p1-l.p1);
343         //if(N.abs2()==0)return NULL;平行或重合
344         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
345         最近點到距離
346         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
347         cross(d2),G=l.p1-p1;
348         T t1=(G.cross(d2)).dot(D)/D.abs2();
349         T t2=(G.cross(d1)).dot(D)/D.abs2();
350         return make_pair(p1+d1*t1,l.p1+d2*t2);
351     }
352     bool same_side(const point3D<T> &a,const
353         point3D<T> &b)const{
354         return (p2-p1).cross(a-p1).dot((p2-p1).
355             cross(b-p1))>0;
356     }
357 };
358 template<typename T>
359 struct plane{
360     point3D<T> p0,n;//平面上的點和法向量
361     plane(){
362         point3D<T> p0(n),n(n){}
363     }
364     T dis2(const point3D<T> &p)const{//點到平
365         面距離的平方
366         T tmp=(p-p0).dot(n);
367         return tmp*tmp/n.abs2();
368     }
369     point3D<T> projection(const point3D<T> &p)
370         const{
371         return p-n*(p-p0).dot(n)/n.abs2();
372     }
373     point3D<T> line_intersection(const line3D<
374         T> &l)const{
375         T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
376         重合該平面
377         return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
378             tmp);
379     }
380     line3D<T> plane_intersection(const plane &
381         p1)const{
382         point3D<T> e=n.cross(p1.n),v=n.cross(e);

```

```

370     T tmp=p1.n.dot(v);//等於0表示平行或重合
371     該平面
372     point3D<T> q=p0+(v*(p1.n.dot(p1.p0-p0))/
373         tmp);
374     return line3D<T>(q,q+e);
375 }
376 template<typename T>
377 struct triangle3D{
378     point3D<T> a,b,c;
379     triangle3D(const point3D<T> &a,const
380         point3D<T> &b,const point3D<T> &c):a(a
381         ),b(b),c(c){}
382     bool point_in(const point3D<T> &p)const{//
383         點在該平面上的投影在三角形中
384         return line3D<T>(b,c).same_side(p,a)&&
385             line3D<T>(a,c).same_side(p,b)&&
386             line3D<T>(a,b).same_side(p,c);
387     }
388 };
389 template<typename T>
390 struct tetrahedron{//四面體
391     point3D<T> a,b,c,d;
392     tetrahedron(const point3D<T> &a,const
393         point3D<T> &b,const point3D<T> &c,
394         const point3D<T> &d):a(a),b(b),c(c),d(d){}
395     T volume6()const{//體積的六倍
396         return (d-a).dot((b-a).cross(c-a));
397     }
398     point3D<T> centroid()const{
399         return (a+b+c+d)/4;
400     }
401     bool point_in(const point3D<T> &p)const{
402         return triangle3D<T>(a,b,c).point_in(p)
403             &&triangle3D<T>(c,d,a).point_in(p);
404     }
405 };
406 template<typename T>
407 struct convexhull3D{
408     static const int MAXN=1005;
409     struct face{
410         int a,b,c;
411         face(int a,int b,int c):a(a),b(b),c(c){}
412     };
413     vector<point3D<T>> pt;
414     vector<face> ans;
415     int fid[MAXN][MAXN];
416     void build(){
417         int n=pt.size();
418         ans.clear();
419         memset(fid,0,sizeof(fid));
420         ans.emplace_back(0,1,2);//注意不能共線
421         ans.emplace_back(2,1,0);
422         int ftop = 0;
423         for(int i=3, ftop=1; i<n; ++i,++ftop){
424             vector<face> next;
425             for(auto &f:ans){
426                 T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[
427                     f.a]).cross(pt[f.c]-pt[f.a]));
428                 if(d<=0) next.push_back(f);
429                 int ff=0;
430                 if(d>0) ff=ftop;

```

```

423     else if(d<0) ff=-ftop;
424     fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c]
425         ][f.a]=ff;
426     }
427     for(auto &f:ans){
428         if(fid[f.a][f.b]>0 && fid[f.a][f.b]
429             !=fid[f.b][f.a])
430             next.emplace_back(f.a,f.b,i);
431         if(fid[f.b][f.c]>0 && fid[f.b][f.c]
432             !=fid[f.c][f.b])
433             next.emplace_back(f.b,f.c,i);
434         if(fid[f.c][f.a]>0 && fid[f.c][f.a]
435             !=fid[f.a][f.c])
436             next.emplace_back(f.c,f.a,i);
437     }
438     ans=next;
439     }
440     point3D<T> centroid()const{
441         point3D<T> res(0,0,0);
442         T vol=0;
443         for(auto &f:ans){
444             T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c]
445                 ));
446             res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
447             vol+=tmp;
448         }
449         return res/(vol*4);
450     }
451 };

```

```

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(
7         mid,R));
8     inplace_merge(L, mid, R, ycmp);
9     static vector<point> b; b.clear();
10    for(auto u=L;u<R;++u){
11        if((u->x-x)*(u->x-x)>=d) continue;
12        for(auto v=b.rbegin();v!=b.rend();++v){
13            T dx=u->x-v->x, dy=u->y-v->y;
14            if(dy*dy>=d) break;
15            d=min(d,dx*dx+dy*dy);
16        }
17        b.push_back(*u);
18    }
19    return d;
20 }
21 T closest_pair(vector<point<T>> &v){
22     sort(v.begin(),v.end(),xcmp);
23     return closest_pair(v.begin(),v.end());
24 }

```

2 Data Structure

2.1 CDQ DP

1.3 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*
7         c2-v.x*c1)/d);
8 }
9 void solve(PT p[],int n,PT &c,T &r2){
10     random_shuffle(p,p+n);
11     c=p[0]; r2=0; // c,r2 = 圓心,半徑平方
12     for(int i=1;i<n;i++){if((p[i]-c).abs2()>r2){
13         c=p[i]; r2=0;
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     }
24 }

```

1.4 最近點對

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 const int MAXN = 100005;
4 struct node{
5     double a,b,r,k,x,y;
6     int id;
7 } p[MAXN];
8 double DP[MAXN];
9 deque<int> q;
10 bool cmpK(const node &a,const node &b){
11     return a.k>b.k;
12 }
13 bool cmpX(const node &a,const node &b){
14     return a.x<b.x||((a.x==b.x&&a.y<b.y));
15 }
16 double Slope(int a,int b){
17     if(!b) return -1e20;
18     if(p[a].x==p[b].x) return 1e20;
19     return (p[a].y-p[b].y)/(p[a].x-p[b].x);
20 }
21 void CDQ(int l, int r){
22     if(l==r){
23         DP[l] = max(DP[l],DP[l-1]);
24         p[l].y = DP[l]/(p[l].a*p[l].r+p[l].b);
25         p[l].x = p[l].y*p[l].r;
26         return;
27     }
28     int mid = (l+r)/2;
29     stable_partition(p+l,p+r+1,[&](const node
30         &d){return d.id<=mid;});
31     CDQ(l, mid); q.clear();
32     for(int i=1, j; i<=mid; ++i){

```

```

32 while((j=q.size())>1&&Slope(q[j-2],q[j
33 -1])<Slope(q[j-1],i)) q.pop_back();
34 q.push_back(i);
35 }q.push_back(0);
36 for(int i=mid+1; i<=r; ++i){
37 while(q.size()>1&&Slope(q[0],q[1])>p[i].
38 k) q.pop_front();
39 DP[p[i].id] = max(DP[p[i].id], p[i].a*p[
40 q[0]].x+p[i].b*p[q[0]].y);
41 }
42 CDQ(mid+1,r);
43 inplace_merge(p+1,p+mid+1,p+r+1,cmpX);
44 }
45 double solve(int n,double S){
46 DP[0] = S;
47 sort(p+1,p+1+n,cmpK);
48 CDQ(1,n);
49 return DP[n];
50 }
51 int main(){
52 int n; double S;
53 scanf("%d%Lf",&n,&S);
54 for(int i=1; i<=n; ++i){
55 scanf("%Lf%Lf%Lf",&p[i].a,&p[i].b,&p[i].
56 r);
57 p[i].id = i, p[i].k = -p[i].a/p[i].b;
58 }
59 printf("%.3Lf\n",solve(n,S));
60 return 0;
61 }

```

2.2 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[MAXNND]; //每個節點代表的
6 列同行
7 int L[MAXNND],R[MAXNND],U[MAXNND],D[MAXNND];
8 vector<int> ans,ans2;
9 void init(int _n,int _m){
10 n=_n,m=_m;
11 for(int i=0;i<=m;++i){
12 U[i]=D[i]=i,L[i]=i-1,R[i]=i+1;
13 S[i]=0;
14 }
15 R[m]=0,L[0]=m;
16 sz=m,ansd=INT_MAX; //ansd存最優解的個數
17 for(int i=1;i<=n;++i)H[i]=-1;
18 }
19 void add(int r,int c){
20 ++S[col[++sz]=c];
21 row[sz]=r;
22 D[sz]=D[c],U[D[c]]=sz,U[sz]=c,D[c]=sz;
23 if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
24 else R[sz]=R[H[r]],L[R[H[r]]]=sz,L[sz]=H
25 [r],R[H[r]]=sz;
26 }
27 #define DFOR(i,A,s) for(int i=A[s];i!=s;i=
28 A[i])

```

```

26 void remove(int c){ //刪除第c行和所有當前覆
27 蓋到第c行的列
28 L[R[c]]=L[c],R[L[c]]=R[c]; //這裡刪除第c
29 行，若有些行不需要處理可以在開始時呼
30 叫他
31 DFOR(i,D,c)DFOR(j,R,i){U[D[j]]=U[j],D[U[
32 j]]=D[j],--S[col[j]];}
33 }
34 void restore(int c){ //恢復第c行和所有當前
35 覆蓋到第c行的列，remove的逆操作
36 DFOR(i,U,c)DFOR(j,L,i){++S[col[j]],U[D[j
37 ]]=j,D[U[j]]=j;}
38 L[R[c]]=c,R[L[c]]=c;
39 }
40 void remove2(int nd){ //刪除nd所在的行當前
41 所有點(包括虛擬節點)，只保留nd
42 DFOR(i,D,nd)L[R[i]]=L[i],R[L[i]]=R[i];
43 }
44 void restore2(int nd){ //刪除nd所在的行當前
45 所有點，為remove2的逆操作
46 DFOR(i,U,nd)L[R[i]]=R[L[i]]=i;
47 }
48 bool vis[MAXN];
49 int h(){ //估價函數 for IDA*
50 int res=0;
51 memset(vis,0,sizeof(vis));
52 DFOR(i,R,0)if(!vis[i]){
53 vis[i]=1;
54 ++res;
55 DFOR(j,D,i)DFOR(k,R,j)vis[col[k]]=1;
56 }
57 return res;
58 }
59 bool dfs(int d){ //for精確覆蓋問題
60 if(d+h())>=ansd return 0; //找最佳解，找
61 任意解可以刪掉
62 if(!R[0]){ansd=d;return 1;}
63 int c=R[0];
64 DFOR(i,R,0)if(S[i]<S[c])c=i;
65 remove(c);
66 DFOR(i,D,c){
67 ans.push_back(row[i]);
68 DFOR(j,R,i)remove(col[j]);
69 if(dfs(d+1))return 1;
70 ans.pop_back();
71 DFOR(j,L,i)restore(col[j]);
72 }
73 restore(c);
74 return 0;
75 }
76 void dfs2(int d){ //for最小重複覆蓋問題
77 if(d+h())>=ansd return;
78 if(!R[0]){ansd=d;ans=ans2;return;}
79 int c=R[0];
80 DFOR(i,R,0)if(S[i]<S[c])c=i;
81 DFOR(i,D,c){
82 anst.push_back(row[i]);
83 remove2(i);
84 DFOR(j,R,i)remove2(j),--S[col[j]];
85 dfs2(d+1);
86 anst.pop_back();
87 DFOR(j,L,i)restore2(j),++S[col[j]];
88 restore2(i);
89 }
90 }

```

```

80 }
81 }
82 bool exact_cover(){ //解精確覆蓋問題
83 return ans.clear(), dfs(0);
84 }
85 void min_cover(){ //解最小重複覆蓋問題
86 anst.clear(); //暫存用，答案還是存在ans裡
87 dfs2(0);
88 }
89 #undef DFOR
90 };

```

2.3 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]-
8 x.d[i]);
9 return ret;
10 }
11 bool operator==(const point &p){
12 auto it=A.begin();
13 for(size_t i=0;i<kd;++i)
14 if(d[i]!=p.d[i])return 0;
15 return 1;
16 }
17 bool operator<(const point &b)const{
18 return d[0]<b.d[0];
19 }
20 };
21 private:
22 struct node{
23 node *l,*r;
24 point pid;
25 int s;
26 node(const point &p):l(0),r(0),pid(p),s
27 (1){}
28 ~node(){delete l;delete r;}
29 void up(){s=(l?l->s:0)+1+(r?r->s:0);}
30 }*root;
31 const double alpha=loga;
32 const T INF; //記得要給INF，表示極大值
33 int maxn;
34 struct __cmp{
35 int sort_id;
36 bool operator()(const node*x,const node*y)
37 const{
38 return operator()(x->pid,y->pid);
39 }
40 bool operator()(const point &x,const
41 point &y)const{
42 if(x.d[sort_id]!=y.d[sort_id])
43 return x.d[sort_id]<y.d[sort_id];
44 for(size_t i=0;i<kd;++i)
45 if(x.d[i]!=y.d[i])return x.d[i]<y.d[
46 i];
47 return 0;
48 }
49 }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 0;
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 }

```



```

104 }
105 cmp.sort_id=k;
106 if(erase(cmp(x,u->pid)?u->l:u->r,(k+1)%
107     kd,x))
108     return --u->s, 1;
109 return 0;
110 }
111 T heuristic(const T h[])const{
112     T ret=0;
113     for(size_t i=0;i<kd;++i)ret+=h[i];
114     return ret;
115 }
116 int qM;
117 priority_queue<pair<T,point>> pQ;
118 void nearest(node *u,int k,const point &x,
119     T *h,T &mndist){
120     if(u==0||heuristic(h)>=mndist)return;
121     T dist=u->pid.dist(x),old=h[k];
122     /*mndist=std::min(mndist,dist);*/
123     if(dist<mndist){
124         pQ.push(std::make_pair(dist,u->pid));
125         if((int)pQ.size()==qM+1)
126             mndist=pQ.top().first,pQ.pop();
127     }
128     if(x.d[k]<u->pid.d[k]){
129         nearest(u->l,(k+1)%kd,x,h,mndist);
130         h[k] = abs(x.d[k]-u->pid.d[k]);
131         nearest(u->r,(k+1)%kd,x,h,mndist);
132     }else{
133         nearest(u->r,(k+1)%kd,x,h,mndist);
134         h[k] = abs(x.d[k]-u->pid.d[k]);
135         nearest(u->l,(k+1)%kd,x,h,mndist);
136     }
137     h[k]=old;
138 }
139 vector<point>in_range;
140 void range(node *u,int k,const point&mi,
141     const point&ma){
142     if(!u)return;
143     bool is=1;
144     for(int i=0;i<kd;++i)
145         if(u->pid.d[i]<mi.d[i]||ma.d[i]<u->pid
146             .d[i])
147             { is=0;break; }
148     if(is) in_range.push_back(u->pid);
149     if(mi.d[k]<u->pid.d[k])range(u->l,(k+1)
150         %kd,mi,ma);
151     if(ma.d[k]>u->pid.d[k])range(u->r,(k+1)
152         %kd,mi,ma);
153 }
154 public:
155 kd_tree(const T &INF,double a=0.75):
156     root(0),alpha(a),loga(log2(1.0/a)),INF(INF
157         ),maxn(1){}
158 ~kd_tree(){delete root;}
159 void clear(){delete root,root=0,maxn=1;}
160 void build(int n,const point *p){
161     delete root,A.resize(maxn=n);
162     for(int i=0;i<n;++i)A[i]=new node(p[i]);
163     root=build(0,0,n-1);
164 }
165 void insert(const point &x){
166     insert(root,0,x,lg(size(root))/loga);
167     if(root->s>maxn)maxn=root->s;
168 }
169 bool erase(const point &p){

```

```

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

```

2.4 kd tree replace segment tree

```

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+=1->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    }
26    void up2(){/*其他懶惰標記向上更新*/}
27    void down(){/*其他懶惰標記下推*/}
28 }*root;
29 //檢查區間包含用的函數
30 bool range_include(node *o,const point &L,
31     const point &R){
32     for(int i=0;i<kd;++i){
33         if(L.d[i]>o->ma.d[i]||R.d[i]<o->mi.d[i])
34             return 0;

```

```

31 }//(L,R)區間有和o的區間有交集就回傳true
32 return 1;
33 }
34 bool range_in_range(node *o,const point &L,
35     const point &R){
36     for(int i=0;i<kd;++i){
37         if(L.d[i]>o->mi.d[i]||o->ma.d[i]>R.d[i])
38             return 0;
39     }//(L,R)區間完全包含o的區間就回傳true
40 return 1;
41 }
42 bool point_in_range(node *o,const point &L,
43     const point &R){
44     for(int i=0;i<kd;++i){
45         if(L.d[i]>o->pid.d[i]||R.d[i]<o->pid.d[i]
46             )return 0;
47     }//(L,R)區間完全包含o->pid這個點就回傳true
48 return 1;
49 }
50 //單點修改 · 以單點改值為例
51 void update(node *u,const point &x,int data,
52     int k=0){
53     if(!u)return;
54     u->down();
55     if(u->pid==x){
56         u->data=data;
57         u->up2();
58         return;
59     }
60     cmp.sort_id=k;
61     update(cmp(x,u->pid)?u->l:u->r,x,data,(k
62         +1)%kd);
63     u->up2();
64 }
65 //區間修改
66 void update(node *o,const point &L,const
67     point &R,int data){
68     if(!o)return;
69     o->down();
70     if(range_in_range(o,L,R)){
71         //區間懶惰標記修改
72         o->down();
73         return;
74     }
75     if(point_in_range(o,L,R)){
76         //這個點在(L,R)區間 · 但是他的左右子樹不
77         一定在區間中
78         //單點懶惰標記修改
79     }
80     if(o->l&&range_include(o->l,L,R))update(o
81         ->l,L,R,data);
82     if(o->r&&range_include(o->r,L,R))update(o
83         ->r,L,R,data);
84     o->up2();
85 }
86 //區間查詢 · 以總和為例
87 int query(node *o,const point &L,const point
88     &R){
89     if(!o)return 0;
90     o->down();
91     if(range_in_range(o,L,R))return o->sum;
92     int ans=0;
93     if(point_in_range(o,L,R))ans+=o->data;
94 }

```

```

83 if(o->l&&range_include(o->l,L,R))ans+=
84     query(o->l,L,R);
85 if(o->r&&range_include(o->r,L,R))ans+=
86     query(o->r,L,R);
87 return ans;
88 }

```

2.5 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 }

```

2.6 skew heap

```

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

```

2.7 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 }djs;

```

2.8 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 };

```

```

11 int g[MAXN];
12 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
18     directed=false){
19     e.push_back(edge(v,g[u],cap));
20     g[u]=e.size()-1;
21     e.push_back(edge(u,g[v],directed?0:cap));
22     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
20     directed=false){
21     e.push_back(edge(v,g[u],cap));
22     g[u]=e.size()-1;
23     e.push_back(edge(u,g[v],directed?0:cap));
24     g[v]=e.size()-1;
25 }
26 T dfs(int u,int s,int t,T 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));

```

```

46     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));

```

```

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

```

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;
4 //n1個點連向n2個點。其中n2個點可以匹配很多邊
5 vector<int> g[MAXN1];//圖 0-base
6 size_t c[MAXN2];
7 //每個屬於n2點最多可以接受幾條匹配邊
8 vector<int> matchs[MAXN2];
9 //每個屬於n2的點匹配了那些點
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[x],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 BronKerbosch

```

1 struct maximalCliques{
2     using Set = vector<int>;
3     size_t n; //1-base
4     vector<Set> G;
5     static Set setUnion(const Set &A, const
6         Set &B){
7         Set C(A.size() + B.size());
8         auto it = set_union(A.begin(),A.end(),B.
9             begin(),B.end(),C.begin());
10        C.erase(it, C.end());
11        return C;
12    }
13    static Set setIntersection(const Set &A,
14        const Set &B){
15        Set C(min(A.size(), B.size()));
16        auto it = set_intersection(A.begin(),A.
17            end(),B.begin(),B.end(),C.begin());
18        C.erase(it, C.end());
19        return C;
20    }
21    static Set setDifference(const Set &A,
22        const Set &B){

```

```

23        Set C(min(A.size(), B.size()));
24        auto it = set_difference(A.begin(),A.end
25            (),B.begin(),B.end(),C.begin());
26        C.erase(it, C.end());
27        return C;
28    }
29    void BronKerbosch1(Set R, Set P, Set X){
30        if(P.empty()&&X.empty()){
31            // R form an maximal clique
32            return;
33        }
34        for(auto v: P){
35            BronKerbosch1(setUnion(R,{v}),
36                setIntersection(P,G[v]),
37                setIntersection(X,G[v]));
38            P = setDifference(P,{v});
39            X = setUnion(X,{v});
40        }
41    }
42    void init(int _n){
43        G.clear();
44        G.resize((n = _n) + 1);
45    }
46    void addEdge(int u, int v){
47        G[u].emplace_back(v);
48        G[v].emplace_back(u);
49    }
50    void solve(int n){
51        Set P;
52        for(int i=1;i<n; ++i){
53            sort(G[i].begin(), G[i].end());
54            G[i].erase(unique(G[i].begin(), G[i].end()),
55                G[i].end());
56            P.emplace_back(i);
57        }
58        BronKerbosch1({}, P, {});
59    }
60 }

```

4.5 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]*A%P;
20             for(int j:g[i])f[t][i]=(f[t][i]+f[t-1][j]*B)%P;
21         }
22     }
23 }

```

```

20     for(int j:rg[i])f[t][i]=(f[t][i]+f[t
21         -1][j]*C%P)%P;
22     if(i==u)f[t][i]=D;//如果圖太大的話，
23         把這行刪掉，執行一次後f[K]就會是所
24         有點的答案
25     f[t][i]=P;
26 }
27 return f[K][u];
28 }
29 vector<long long> graph_hash(){
30     vector<long long> ans;
31     for(int i=0;i<n;++i)ans.push_back(
32         point_hash(i)); //O(N^2)
33     sort(ans.begin(),ans.end());
34     return ans;

```

4.6 is planar

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 struct FringeOpposedSubset {
4     deque<int> left, right;
5     FringeOpposedSubset() = default;
6     FringeOpposedSubset(int h) : left{h},
7         right() {}
8 };
9 template<typename T>
10 void extend(T& a, T& b, bool rev = false) {
11     rev ? a.insert(a.begin(), b.rbegin(), b.
12         rend())
13         : a.insert(a.end(), b.begin(), b.end()
14             );
15 }
16 struct Fringe {
17     deque<FringeOpposedSubset> FOPs;
18     Fringe(int h) : FOPs{{h}} {}
19     bool operator<=(const Fringe& o) const {
20         return std::tie(FOPs.back().left.back(),
21             FOPs.front().left.front()) <
22             std::tie(o.FOPs.back().left.back(),
23                 o.FOPs.front().left.front());
24     }
25     void merge(Fringe& o) {
26         o.merge_t_alike_edges();
27         merge_t_opposite_edges_into(o);
28         if (FOPs.front().right.empty())
29             o.align_duplicates(FOPs.back().left.
30                 front());
31         else
32             make_union_structure(o);
33         if (o.FOPs.front().left.size()) FOPs.
34             push_front(o.FOPs.front());
35     }
36     void merge_t_alike_edges() {
37         FringeOpposedSubset ans;
38         for (auto& FOP : FOPs) {
39             if (!FOP.right.empty()) throw
40                 runtime_error("Exception");
41             extend(ans.left, FOP.left);
42         }
43     }

```

```

35     FOPs = {ans};
36 }
37 void merge_t_opposite_edges_into(Fringe& o
38     ) {
39     while (FOPs.front().right.empty() &&
40         FOPs.front().left.front() > o.
41         FOPs.front().left.back()) {
42         extend(o.FOPs.front().right, FOPs.
43             front().left);
44         FOPs.pop_front();
45     }
46     void align_duplicates(int dfs_h) {
47         if (FOPs.front().left.back() == dfs_h) {
48             FOPs.front().left.pop_back();
49             swap_side();
50         }
51     }
52     void swap_side() {
53         if (FOPs.front().left.empty() ||
54             (!FOPs.front().right.empty() &&
55                 FOPs.front().left.back() > FOPs.
56                 front().right.back())) {
57             swap(FOPs.front().left, FOPs.front().
58                 right);
59         }
60     }
61     void make_union_structure(Fringe& o) {
62         auto low = &FOPs.front().left, high = &
63             FOPs.front().right;
64         if (FOPs.front().left.front() >= FOPs.
65             front().right.front())
66             swap(low, high);
67         if (o.FOPs.front().left.back() < low->
68             front())
69             throw runtime_error("Exception");
70         if (o.FOPs.front().left.back() < high->
71             front()) {
72             extend(*low, o.FOPs.front().left, true
73                 );
74             extend(*high, o.FOPs.front().right,
75                 true);
76             o.FOPs.front().left.clear();
77             o.FOPs.front().right.clear();
78         }
79     }
80     auto lr_condition(int deep) const {
81         bool L = !FOPs.front().left.empty() &&
82             FOPs.front().left.front() >= deep;
83         bool R = !FOPs.front().right.empty() &&
84             FOPs.front().right.front() >= deep;
85         return make_pair(L, R);
86     }
87     void prune(int deep) {
88         auto [left, right] = lr_condition(deep);
89         while (!FOPs.empty() && (left || right))
90             {
91                 if (left) FOPs.front().left.pop_front
92                     ();
93                 if (right) FOPs.front().right.
94                     pop_front();
95                 if (FOPs.front().left.empty() && FOPs.
96                     front().right.empty())
97                     FOPs.pop_front();
98                 else swap_side();
99             }
100     }

```

```

83     if (!FOPs.empty()) tie(left, right) =
84         lr_condition(deep);
85     }
86 }
87 unique_ptr<Fringe> get_merged_fringe(deque<
88     unique_ptr<Fringe>&& upper) {
89     if (upper.empty()) return nullptr;
90     sort(upper.begin(), upper.end(), [](auto&
91         a, auto& b) { return *a < *b; });
92     for (auto it = next(upper.begin()); it !=
93         upper.end(); ++it)
94         upper.front()->merge(*it);
95     return move(upper.front());
96 }
97 void merge_fringes(vector<deque<unique_ptr<
98     Fringe>>& fringes, int deep) {
99     auto mf = get_merged_fringe(fringes.back()
100         );
101     fringes.pop_back();
102     if (mf) {
103         mf->prune(deep);
104         if (mf->FOPs.size()) fringes.back().
105             push_back(move(mf));
106     }
107 }
108 struct Edge {
109     int from, to;
110     Edge(int from, int to) : from(from), to(to)
111         {}
112     bool operator==(const Edge& o) const {
113         return from == o.from && to == o.to;
114     }
115 }
116 struct Graph {
117     int n = 0;
118     vector<vector<int>> neighbor;
119     vector<Edge> edges;
120     void add_edge(int from, int to) {
121         if (from == to) return;
122         edges.emplace_back(from, to);
123         edges.emplace_back(to, from);
124     }
125     void build() {
126         sort(edges.begin(), edges.end(), [](
127             const auto& a, const auto& b) {
128             return a.from < b.from || (a.from == b
129                 .from && a.to < b.to);
130         });
131         edges.erase(unique(edges.begin(), edges.
132             end()), edges.end());
133         n = 0;
134         for (auto& e : edges) n = max(n, max(e.
135             from, e.to) + 1);
136         neighbor.resize(n);
137         for (auto& e : edges) neighbor[e.from].
138             push_back(e.to);
139     }
140 }
141 Graph g;
142 vector<int> Deeps;
143 vector<deque<unique_ptr<Fringe>>> fringes;
144 bool dfs(int x, int parent = -1) {
145     for (int y : g.neighbor[x]) {
146         if (y == parent) continue;
147         if (Deeps[y] < 0) { // tree edge

```

```

136     fringes.push_back({});
137     Deeps[y] = Deeps[x] + 1;
138     if (!dfs(y, x)) return false;
139     } else if (Deeps[x] > Deeps[y]) { //
140         back edge
141         fringes.back().push_back(make_unique<
142             Fringe>(Deeps[y]));
143     }
144     try {
145         if (fringes.size() > 1) merge_fringes(
146             fringes, Deeps[parent]);
147     } catch (const exception& e) {
148         return false;
149     }
150     return true;
151 }
152 bool is_planar() {
153     Deeps.assign(g.n, -1);
154     for (int i = 0; i < g.n; ++i) {
155         if (Deeps[i] >= 0) continue;
156         fringes.clear();
157         Deeps[i] = 0;
158         if (!dfs(i)) return false;
159     }
160     return true;
161 }
162 int main() {
163     int n, m, u, v;
164     cin >> n >> m;
165     for (int i = 0; i < m; ++i) {
166         cin >> u >> v;
167         g.add_edge(u, v);
168     }
169     g.build();
170     cout << (is_planar() ? "YES" : "NO") <<
171         endl;
172     return 0;

```

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

```

4.8 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, tmp[0]=u, v=u+1; v<N; ++v)
38             if(g[u][v]) stk[1][ns++]=v;
39         dfs(ns,1), dp[u]=ans;
40     }
41     return ans;
42 }
43 };

```

4.9 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],0x3f,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 }

```

4.10 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)){
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){
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 }

```

4.11 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 }

```

4.12 一般圖最小權完美匹配

```

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

```

```

29     onstk[v] = 0;
30 }
31 }
32 }
33 onstk[u] = 0;
34 stk.pop_back();
35 return false;
36 }
37 int solve() {
38     // find a match
39     for (int i=0; i<n; i+=2){
40         match[i] = i+1, match[i+1] = i;
41     }
42     for(;;){
43         int found = 0;
44         for (int i=0; i<n; i++) dis[i] = onstk
            [i] = 0;
45         for (int i=0; i<n; i++){
46             stk.clear();
47             if (!onstk[i] && SPFA(i)){
48                 found = 1;
49                 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;

```

4.13 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=n;tn>1;--tn){
19             for(int i=1;i<tn;++i)dis[nd[i]]=0;
20             for(int i=1;i<tn;++i){

```

```

21         ind=i;
22         for(int j=i;j<tn;++j){
23             dis[nd[j]]+=g[nd[i-1]][nd[j]];
24             if(dis[nd[ind]]<dis[nd[j]])ind=j;
25         }
26         swap(nd[ind],nd[i]);
27     }
28     if(ans>dis[nd[ind]])ans=dis[t=nd[ind
        ]],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 };

```

4.14 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 };

```

4.15 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<<MAXN][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 }

```

4.16 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;

```

```

14    }
15    mem[ MAXM ]; //靜態記憶體
16    node *pq[ MAXN*2 ],*E[ MAXN*2 ];
17    int st[ MAXN*2 ],id[ MAXN*2 ],m;
18    void init(int n){
19        for(int i=1;i<=n;++i){
20            pq[i]=E[i]=0, st[i]=id[i]=i;
21        }m=0;
22    }
23    node *merge(node *a,node *b){ //skew heap
24        if(!a||!b)return a?a:b;
25        a->down(),b->down();
26        if(b->w<a->w)return merge(b,a);
27        swap(a->l,a->r);
28        a->l=merge(b,a->l);
29        return a;
30    }
31    void add_edge(int u,int v,T w){
32        if(u!=v)pq[v]=merge(pq[v],&(mem[m++]=
            node(u,v,w)));
33    }
34    int find(int x,int *st){
35        return st[x]==x?x:st[x]=find(st[x],st);
36    }
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=
                    find(E[u]->u,id)){
54                    if(pq[u])pq[u]->tag-=E[u]->w;
55                    id[find(u,id)]=N;
56                    pq[N]=merge(pq[N],pq[u]);
57                }
58                st[N]=find(i,st);
59                id[find(i,id)]=N;
60                else st[find(i,st)]=find(E[i]->u,st)
                    ,--all;
61            }
62            return all==1?ans:-INT_MAX; //圖不連通就
            無解
        }
        };

```

4.17 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    }

```

5 Language

5.1 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
10 //小寫字母為終端字符
11 vector<CNF> cnf;
12 void init(){
13     state=0;
14     rule.clear();
15     cnf.clear();
16 }
17 void add_to_cnf(char s,const string &p,int cost){
18     //加入一個s -> <p>的文法，代價為cost
19     if(rule.find(s)==rule.end())rule[s]=state++;
20     for(auto c:p)if(rule.find(c)==rule.end())rule[c]=state++;
21     if(p.size()==1){
22         cnf.push_back(CNF(rule[s],rule[p[0]],-1,cost));
23     }else{
24         int left=rule[s];
25         int sz=p.size();

```

```

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

```

6 Linear Programming

6.1 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] < a[x][0]/a[x][y]) x = i;
42        if(x == -1) return VDB(); //unbounded
43        pivot(x, y);
44    }
45    VDB ans(n + 1);
46    for(int i = 1; i <= m; ++i)
47        if(left[i] <= n) ans[left[i]] = a[i][0];
48    ans[0] = -a[0][0];
49    return ans;
50 }

```

7 Number Theory

7.1 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];

```

```

10 }
11 void all_divdown(const LL &n) { // all n/x
12     for(LL a=1;a<=n;a=n/(n/(a+1))){
13         // dosomething;
14     }
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     v.erase(unique(v.begin(), v.end()), v.end());
57     for(LL g=2;g<p;++g)
58         if(g_test(g,p,v))
59             return g;
60     puts("primitive_root NOT FOUND");
61     return -1;
62 }
63 int Legendre(const LL &a, const LL &p) {
64     return modexp(a%p,(p-1)/2,p);
65 }
66 LL inv(const LL &a, const LL &n) {
67     LL d,x,y;
68     gcd(a,n,d,x,y);
69     return d==1 ? (x+n)%n : -1;
70 }
71 int inv[maxn];
72 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
78     LL &p) {
79     // a ^ x = b ( mod p )
80     int m=sqrt(p+.5), e=1;
81     LL v=inv(modexp(a,m,p), p);
82     map<LL,int> x;
83     x[1]=0;
84     for(int i=1;i<m;++i) {
85         e = LLMul(e,a,p);
86         if(!x.count(e)) x[e] = i;
87     }
88     for(int i=0;i<m;++i) {
89         if(x.count(b)) return i*m + x[b];
90         b = LLMul(b,v,p);
91     }
92     return -1;
93 }
94
95 LL Tonelli_Shanks(const LL &n, const LL &p)
96 {
97     // x^2 = n ( mod p )
98     if(n==0) return 0;
99     if(Legendre(n,p)!=1) while(1) { puts("SQRT
100         ROOT does not exist"); }
101     int S = 0;
102     LL Q = p-1;
103     while( !(Q&1) ) { Q>>=1; ++S; }
104     if(S==1) return modexp(n%p,(p+1)/4,p);
105     LL z = 2;
106     for(; Legendre(z,p)!=-1;++z)
107         LL c = modexp(z,Q,p);
108     LL R = modexp(n%p,(Q+1)/2,p), t = modexp(n
109         %p,Q,p);
110     int M = S;
111     while(1) {
112         if(t==1) return R;
113         LL b = modexp(c,1<<(M-1),p);
114         R = LLMul(R,b,p);
115         t = LLMul( LLMul(b,b,p), t, p);
116         c = LLMul(b,b,p);
117         M = i;
118     }
119     return -1;
120 }
121
122 template<typename T>
123 T Euler(T n){
124     T ans=n;
125     for(T i=2;i*i<=n;++i){
126         if(n%i==0){
127             ans=ans/i*(i-1);
128             while(n%i==0)n/=i;
129         }
130     }
131     if(n>1)ans=ans/n*(n-1);
132     return ans;
133 }
134
135 //Chinese_remainder_theorem
136 template<typename T>
137 T pow_mod(T n,T k,T m){

```

```

138 T ans=1;
139 for(n=(n>m?n%m:n);k>>=1){
140     if(k&1)ans=ans*n%m;
141     n=n*n%m;
142 }
143 return ans;
144 }
145
146 template<typename T>
147 T crt(vector<T> &m,vector<T> &a){
148     T M=1,tM,ans=0;
149     for(int i=0;i<(int)m.size();++i)M*=m[i];
150     for(int i=0;i<(int)a.size();++i){
151         tM=M/m[i];
152         ans=(ans+(a[i]*tM*M)*pow_mod(tM,Euler(m[
153             i])-1,m[i])%M)%M;
154     }
155     /*如果m[i]是質數 · Euler(m[i])-1=m[i]-2 ·
156         就不用算Euler了*/
157     return ans;
158 }
159
160 //java code
161 //求sqrt(N)的連分數
162 public static void Pell(int n){
163     BigInteger N,p1,p2,q1,q2,a0,a1,a2,g1,g2,h1
164         ,h2,p,q;
165     g1=q2=p1=BigInteger.ZERO;
166     h1=q1=p2=BigInteger.ONE;
167     a0=a1=BigInteger.valueOf((int)Math.sqrt
168         (1.0*n));
169     BigInteger ans=a0.multiply(a0);
170     if(ans.equals(BigInteger.valueOf(n))){
171         System.out.println("No solution!");
172         return ;
173     }
174     while(true){
175         g2=a1.multiply(h1).subtract(g1);
176         h2=N.subtract(g2.pow(2)).divide(h1);
177         a2=g2.add(a0).divide(h2);
178         p=a1.multiply(p2).add(p1);
179         q=a1.multiply(q2).add(q1);
180         if(p.pow(2).subtract(N.multiply(q.pow
181             (2))).compareTo(BigInteger.ONE)==0)
182             break;
183         g1=g2;h1=h2;a1=a2;
184         p1=p2;p2=p;
185         q1=q2;q2=q;
186     }
187     System.out.println(p+" "+q);
188 }

```

7.2 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 }

```

7.3 cantor expansion

```

1 int factorial[MAXN];
2 void init(){
3     factorial[0]=1;
4     for(int i=1;i<=MAXN;++i)factorial[i]=
5         factorial[i-1]*i;
6 }
7 int encode(const vector<int> &s){
8     int n=s.size(),res=0;
9     for(int i=0;i<n;++i){
10         int t=0;
11         for(int j=i+1;j<n;++j)
12             if(s[j]<s[i])++t;
13         res+=t*factorial[n-i-1];
14     }
15     return res;
16 }
17 vector<int> decode(int a,int n){
18     vector<int> res;
19     vector<bool> vis(n,0);
20     for(int i=n-1;i>=0;--i){
21         int t=a/factorial[i],j;
22         for(j=0;j<n;++j)
23             if(!vis[j]){
24                 if(t==0)break;
25                 --t;
26             }
27         res.push_back(j);
28         vis[j]=1;
29         a%=factorial[i];
30     }
31     return res;
32 }

```

7.4 FFT

```

1 template<typename T,typename VT=vector<
2     complex<T> > >
3 struct FFT{
4     const T pi;
5     FFT(const T pi=acos((T)-1)):pi(pi){}
6     unsigned bit_reverse(unsigned a,int len){
7         a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)>>1);
8         a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)>>2);
9         a=((a&0x0F0F0F0FU)<<4)|((a&0xFF0F0F0FU)>>4);
10        a=((a&0x00FF00FFU)<<8)|((a&0xFFFF0000U)>>8);
11        a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)>>16);
12        return a>>(32-len);
13    }
14    void fft(bool is_inv,VT &in,VT &out,int N)
15    {

```

```

14 int bitlen=__lg(N),num=is_inv?-1:1;
15 for(int i=0;i<N;++i)out[bit_reverse(i,
16     bitlen)]=in[i];
17 for(int step=2;step<=N;step<=1){
18     const int mh=step>>1;
19     for(int i=0;i<mh;++i){
20         complex<T> wi=exp(complex<T>(0,i*num
21             *pi/mh));
22         for(int j=i;j<N;j+=step){
23             int k=j+mh;
24             complex<T> u=out[j],t=wi*out[k];
25             out[j]=u+t;
26             out[k]=u-t;
27         }
28     }
29     if(is_inv)for(int i=0;i<N;++i)out[i]/=N;
30 }
31 }

```

7.5 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
7     x){
8     double e = 1, s = 0;
9     for(auto i : coef) s += i*e, e *= x;
10    return s;
11 }
12 double find(const vector<double>&coef, int n
13     , double lo, double hi){
14     double sign_lo, sign_hi;
15     if( !(sign_lo = sign(get(coef,lo))) )
16         return lo;
17     if( !(sign_hi = sign(get(coef,hi))) )
18         return hi;
19     if(sign_lo * sign_hi > 0) return INF;
20     for(int stp = 0; stp < 100 && hi - lo >
21         eps; ++stp){
22         double m = (lo+hi)/2.0;
23         int sign_mid = sign(get(coef,m));
24         if(!sign_mid) return m;
25         if(sign_lo*sign_mid < 0) hi = m;
26         else lo = m;
27     }
28     return (lo+hi)/2.0;
29 }
30
31 vector<double> cal(vector<double>coef, int n
32     ){
33     vector<double>res;
34     if(n == 1){
35         if(sign(coef[1])) res.pb(-coef[0]/coef
36             [1]);
37         return res;
38     }
39     vector<double>dcoef(n);

```



```

34 for(int i = 0; i < n; ++i) dcoef[i] = coef
    [i+1]*(i+1);
35 vector<double>droot = cal(dcoef, n-1);
36 droot.insert(droot.begin(), -INF);
37 droot.pb(INF);
38 for(int i = 0; i+1 < droot.size(); ++i){
39     double tmp = find(coef, n, droot[i],
        droot[i+1]);
40     if(tmp < INF) res.pb(tmp);
41 }
42 return res;
43 }
44
45 int main () {
46     vector<double>ve;
47     vector<double>ans = cal(ve, n);
48     // 視情況把答案 +eps · 避免 -0
49 }

```

7.6 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
                        ?-1:1);
6     return f;
7 }
8 vector<int> rev(vector<int> A) {
9     for(int i=0; i<A.size(); i+=2)
10         swap(A[i],A[i^(A.size()-1)]);
11     return A;
12 }
13 vector<int> F_AND_T(vector<int> f, bool
    inverse){
14     return rev(F_OR_T(rev(f), inverse));
15 }
16 vector<int> F_XOR_T(vector<int> f, bool
    inverse){
17     for(int i=0; (2<<i)<=f.size(); ++i)
18         for(int j=0; j<f.size(); j+=2<<i)
19             for(int k=0; k<(1<<i); ++k){
20                 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 }

```

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

```

7.8 Lucas

```

1 ll C(ll n, ll m, ll p){// n!/m!/(n-m)!
2     if(n<m) return 0;
3     return f[n]*inv(f[m],p)%p*inv(f[n-m],p)%p;
4 }
5 ll L(ll n, ll m, ll p){
6     if(!m) return 1;
7     return C(n%p,m%p,p)*L(n/p,m/p,p)%p;
8 }
9 ll Wilson(ll n, ll p){ // n!%p
10    if(!n)return 1;
11    ll res=Wilson(n/p, p);
12    if((n/p)%2) return res*(p-f[n%p])%p;
13    return res*f[n%p]%p; // (p-1)!%p=-1
14 }

```

7.9 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<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     T det=sign?-1:1;
72     for(int i=0;i<r;++i){
73         det = det*m[i][i];
74         for(auto &j:m[i])j/=lazy[i];
75     }
76     return det;
77 }

```

7.10 MillerRobin

```

1 ULL LLmul(ULL a, ULL b, const ULL &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 ULL mod_mul(ULL a,ULL b,ULL m){
14     a%=m,b%=m; /* fast for m < 2^58 */
15     ULL y=(ULL)((double)a*b/m+0.5);
16     ULL 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 llsprp
    [7]={2,325,9375,28178,450775,9780504,
28     1795265022}; //至少 unsigned long long範圍
29 template<typename T>
30 bool isprime(T n,int *sprp,int num){
31     if(n==2)return 1;
32     if(n<2||n%2==0)return 0;
33     int t=0;
34     T u=n-1;
35     for(;u%2==0;++t)u>>=1;
36     for(int i=0;i<num;++i){
37         T a=sprp[i]%n;
38         if(a==0||a==1||a==n-1)continue;
39         T x=pow(a,u,n);
40         if(x==1||x==n-1)continue;
41         for(int j=0;j<t;++j){
42             x=mod_mul(x,x,n);
43             if(x==1)return 0;
44             if(x==n-1)break;
45         }
46         if(x==n-1)continue;
47         return 0;
48     }
49     return 1;
50 }

```

7.11 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;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 }

```

7.12 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;
8     double L=simpson(a,c),R=simpson(c,b);
9     if( abs(L+R-A)<15*eps )
10         return L+R+(L+R-A)/15.0;
11     return asr(a,c,eps/2,L)+asr(c,b,eps/2,R);
12 }
13 double asr(double a,double b,double eps){
14     return asr(a,b,eps,simpson(a,b));
15 }

```

7.13 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("%Ld",&a[i]);
50         scanf("%d",&mod);
51         printf("%Ld\n",high_pow(a,n,mod));
52     }
53     return 0;
54 }

```

7.14 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 }

```

7.15 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 pollrho(const LL n, const int c) { // 循
6     環長度
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) {

```

```

44     if(n<1e9) {
45         smallfactor(n,v);
46         return;
47     }
48     if(Isprime(n)) {
49         v.push_back(n);
50         return;
51     }
52     LL d;
53     for(int c=3; ; ++c) {
54         d = pollrho(n,c);
55         if(d!=n) break;
56     }
57     comfactor(d,v);
58     comfactor(n/d,v);
59 }
60 void Factor(const LL &x, vector<LL> &v) {
61     LL n = x;
62     if(n==1) { puts("Factor 1"); return; }
63     prefactor(n,v);
64     if(n==1) return;
65     comfactor(n,v);
66     sort(v.begin(),v.end());
67 }
68 void AllFactor(const LL &n,vector<LL> &v) {
69     vector<LL> tmp;
70     Factor(n,tmp);
71     v.clear();
72     v.push_back(1);
73     int len;
74     LL now=1;
75     for(int i=0;i<tmp.size();++i) {
76         if(i==0 || tmp[i]!=tmp[i-1]) {
77             len = v.size();
78             now = 1;
79         }
80         now*=tmp[i];
81         for(int j=0;j<len;++j)
82             v.push_back(v[j]*now);
83     }
84 }
85 }
86 }

```

8 String

8.1 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,e1,ed,cnt_dp,vis;
5         joe():ed(0),cnt_dp(0),vis(0){
6             for(int i=0;i<=R-L;++i)next[i]=0;
7         }
8     };
9     public:
10     std::vector<joe> S;
11     std::vector<int> q;
12     int qs,qe,vt;
13     ac_automaton():S(1),qs(0),qe(0),vt(0){}

```

```

14 void clear(){
15     q.clear();
16     S.resize(1);
17     for(int i=0;i<=R-L;++i)S[0].next[i]=0;
18     S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
19 }
20 void insert(const char *s){
21     int o=0;
22     for(int i=0,id;s[i];++i){
23         id=s[i]-L;
24         if(!S[o].next[id]){
25             S.push_back(joe());
26             S[o].next[id]=S.size()-1;
27         }
28         o=S[o].next[id];
29     }
30     ++S[o].ed;
31 }
32 void build_fail(){
33     S[0].fail=S[0].efl=-1;
34     q.clear();
35     q.push_back(0);
36     ++qe;
37     while(qs!=qe){
38         int pa=q[qs++],id,t;
39         for(int i=0;i<=R-L;++i){
40             t=S[pa].next[i];
41             if(!t)continue;
42             id=S[pa].fail;
43             while(~id&&!S[id].next[i])id=S[id].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;
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字串被s匹配成功的次數O(N*M)*/
52 int match_0(const char *s){
53     int ans=0,id,p=0,i;
54     for(i=0;s[i];++i){
55         id=s[i]-L;
56         while(!S[p].next[id]&&p)p=S[p].fail;
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可以被匹配(DP計算)*/
60     }
61     for(i=qe-1;i>=0;--i){
62         ans+=S[q[i]].cnt_dp*S[q[i]].ed;
63         if(~S[q[i]].fail)S[q[i]].fail].cnt_dp+=S[q[i]].cnt_dp;
64     }
65     return ans;
66 }
67 /*多串匹配走efl邊並傳回所有字串被s匹配成功的次數O(N*M^1.5)*/
68 int match_1(const char *s) const{
69     int ans=0,id,p=0,t;
70     for(int i=0;s[i];++i){
71         id=s[i]-L;
72         while(!S[p].next[id]&&p)p=S[p].fail;

```

```

73         if(!S[p].next[id])continue;
74         p=S[p].next[id];
75         if(S[p].ed)ans+=S[p].ed;
76         for(t=S[p].efl;~t;t=S[t].efl){
77             ans+=S[t].ed; /*因為都走efl邊所以保證匹配成功*/
78         }
79     }
80     return ans;
81 }
82 /*枚舉(s的子字串a)的所有相異字串各恰一次並傳回次數O(N*M^(1/3))*/
83 int match_2(const char *s){
84     int ans=0,id,p=0,t;
85     ++vt;
86     /*把戳記vt+=1，只要vt沒溢位，所有S[p].vis==vt就會變成false
87     這種利用vt的方法可以O(1)歸零vis陣列*/
88     for(int i=0;s[i];++i){
89         id=s[i]-L;
90         while(!S[p].next[id]&&p)p=S[p].fail;
91         if(!S[p].next[id])continue;
92         p=S[p].next[id];
93         if(S[p].ed&&S[p].vis!=vt){
94             S[p].vis=vt;
95             ans+=S[p].ed;
96         }
97         for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t].efl){
98             S[t].vis=vt;
99             ans+=S[t].ed; /*因為都走efl邊所以保證匹配成功*/
100         }
101     }
102     return ans;
103 }
104 /*把AC自動機變成真的自動機*/
105 void evolution(){
106     for(qs=1;qs!=qe;){
107         int p=q[qs++];
108         for(int i=0;i<=R-L;++i)
109             if(S[p].next[i]==0)S[p].next[i]=S[S[p].fail].next[i];
110     }
111 }
112 };

```

8.2 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 }
15 T get_hash(int l,int r){ /*閉區間寫法，設編號為0 ~ Len-1*/
16     return (h[r+1]-(h[l]*h_base[r-l+1])%mod+mod)%mod;
17 }

```

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

```

8.4 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 }

```

8.5 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 }

```

8.6 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[ bw[i] ]++;
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[ bw[i] ] + ranks[i];
26     }while( i != begin );
27     return res;
28 }

```

8.7 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 *sa,int *rank,int *tmp,int *c){
10     int A='z'+1,i,k,id=0;
11     for(i=0;i<n;++i)rank[tmp[i]=i]=s[i];
12     radix_sort(rank,tmp);
13     for(k=1;id<n-1;k<=1){
14         for(id=0,i=n-k;i<n;++i)tmp[id++] = i;
15         for(i=0;i<n;++i)
16             if(sa[i]>=k)tmp[id++] = sa[i]-k;

```

```

17 radix_sort(rank,tmp);
18 swap(rank,tmp);
19 for(rank[sa[0]]=id=0,i=1;i<n;++i)
20   rank[sa[i]]=id+=AC(tmp,sa[i-1],sa[i]);
21   A=id+1;
22 }
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]);

```

8.8 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-1]<z[l]?z[i-1]:r-i+1);
6     while(i+z[i]<len&&s[i+z[i]]==s[z[i]])++z[i];
7     if(i+z[i]-1>r)r=i+z[i]-1,l=i;
8   }
9 }

```

9 Tarjan

9.1 dominator tree

```

1 struct dominator_tree{
2   static const int MAXN=5005;
3   int n;// 1-base
4   vector<int> G[MAXN], rG[MAXN];
5   int pa[MAXN], dfn[MAXN], id[MAXN], dfnCnt;
6   int semi[MAXN], idom[MAXN], best[MAXN];
7   vector<int> tree[MAXN]; // tree here
8   void init(int _n){
9     n=_n;
10    for(int i=1;i<=n;++i)
11      G[i].clear(), rG[i].clear();
12  }
13  void add_edge(int u, int v){
14    G[u].push_back(v);
15    rG[v].push_back(u);
16  }
17  void dfs(int u){
18    id[dfn[u]]=++dfnCnt=u;
19    for(auto v:G[u]) if(!dfn[v])
20      dfs(v),pa[dfn[v]]=dfn[u];

```

```

21 }
22 int find(int y,int x){
23   if(y <= x) return y;
24   int tmp = find(pa[y],x);
25   if(semi[best[y]] > semi[best[pa[y]]])
26     best[y] = best[pa[y]];
27   return pa[y] = tmp;
28 }
29 void tarjan(int root){
30   dfnCnt = 0;
31   for(int i=1;i<=n;++i){
32     dfn[i] = idom[i] = 0;
33     tree[i].clear();
34     best[i] = semi[i] = i;
35   }
36   dfs(root);
37   for(int i=dfnCnt;i>1;--i){
38     int u = id[i];
39     for(auto v:rG[u]) if(v=dfn[v]){
40       find(v,i);
41       semi[i]=min(semi[i],semi[best[v]]);
42     }
43     tree[semi[i]].push_back(i);
44     for(auto v:tree[pa[i]]){
45       find(v, pa[i]);
46       idom[v] = semi[best[v]]==pa[i]
47         ? pa[i] : best[v];
48     }
49     tree[pa[i]].clear();
50   }
51   for(int i=2;i<=dfnCnt;++i){
52     if(idom[i] != semi[i])
53       idom[i] = idom[idom[i]];
54     tree[id[idom[i]]].push_back(i);
55   }
56 }
57 }dom;

```

9.2 tnfshb017 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 }

```

9.3 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());

```

```

16   E.emplace_back(v,u);
17 }
18 void dfs(int u,int re=-1){ // u當前點 · re為u連
19   接前一個點的邊
20   int v;
21   low[u]=vis[u]++Time;
22   st[top++]=u;
23   for(int e:G[u]){
24     v=E[e].v;
25     if(!vis[v]){
26       dfs(v,e^1); // e^1反向邊
27       low[u]=min(low[u],low[v]);
28       if(vis[u]<low[v]){
29         E[e].is_bridge=E[e^1].is_bridge=1;
30         ++bridge_cnt;
31       }else if(vis[v]<vis[u]&&e!=re)
32         low[u]=min(low[u],vis[v]);
33     }
34   }
35   if(vis[u]==low[u]){ // 處理BCC
36     ++bcc_cnt; // 1-base
37     do bcc_id[v=st[--top]]=bcc_cnt; // 每個點
38       所在的BCC
39     while(v!=u);
40   }
41 }
42 void bcc_init(int n){
43   Time=bcc_cnt=bridge_cnt=top=0;
44   E.clear();
45   for(int i=1;i<=n;++i){
46     G[i].clear();
47     vis[i]=bcc_id[i]=0;
48   }

```

9.4 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       }
24       bcc_id[u]=bcc_cnt;
25       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 }

```

10 Tree Problem

10.1 HeavyLight

```

1 #include<vector>
2 #define MAXN 100005
3 int siz[MAXN],max_son[MAXN],pa[MAXN],dep[
    MAXN];
4 int link_top[MAXN],link[MAXN],cnt;
5 vector<int> G[MAXN];
6 void find_max_son(int u){
7     siz[u]=1;
8     max_son[u]=-1;
9     for(auto v:G[u]){
10         if(v==pa[u])continue;
11         pa[v]=u;
12         dep[v]=dep[u]+1;
13         find_max_son(v);
14         if(max_son[u]==-1||siz[v]>siz[max_son[u]
            ]))max_son[u]=v;
15         siz[u]+=siz[v];
16     }
17 }
18 void build_link(int u,int top){
19     link[u]++cnt;
20     link_top[u]=top;
21     if(max_son[u]==-1)return;
22     build_link(max_son[u],top);
23     for(auto v:G[u]){
24         if(v==max_son[u]||v==pa[u])continue;
25         build_link(v,v);
26     }
27 }
28 int find_lca(int a,int b){
29     //求LCA，可以在過程中對區間進行處理
30     int ta=link_top[a],tb=link_top[b];
31     while(ta!=tb){
32         if(dep[ta]<dep[tb]){
33             swap(ta,tb);
34             swap(a,b);
35         }
36         //這裡可以對a所在的鏈做區間處理
37         //區間為(Link[ta],Link[a])
38         ta=link_top[a=pa[ta]];
39     }

```

```

40     //最後a,b會在同一條鏈，若a!=b還要在進行一
        次區間處理
41     return dep[a]<dep[b]?a:b;
42 }

```

10.2 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 }

```

10.3 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
    tree的根
10     return nd[nd[x].pa].ch[0]!=x&&nd[nd[x].pa]
        .ch[1]!=x;
11 }
12 void down(int x){//懶惰標記下推
13     if(nd[x].rev){
14         if(nd[x].ch[0]nd[nd[x].ch[0]].rev^=1;

```

```

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

```

```

74 void make_root(int x){
75     access(x),splay(x);
76     nd[x].rev^=1;
77 }
78 void make_root(int x){
79     nd[access(x)].rev^=1;
80     splay(x);
81 }
82 void cut(int x,int y){
83     make_root(x);
84     access(y);
85     splay(y);
86     nd[y].ch[0]=0;
87     nd[x].pa=0;
88 }
89 void cut_parents(int x){
90     access(x);
91     splay(x);
92     nd[nd[x].ch[0]].pa=0;
93     nd[x].ch[0]=0;
94 }
95 void link(int x,int y){
96     make_root(x);
97     nd[x].pa=y;
98 }
99 int find_root(int x){
100     x=access(x);
101     while(nd[x].ch[0])x=nd[x].ch[0];
102     splay(x);
103     return x;
104 }
105 int query(int u,int v){
106     //傳回uv路徑splay tree的根結點
107     //這種寫法無法求LCA
108     make_root(u);
109     return access(v);
110 }
111 int query_lca(int u,int v){
112     //假設求鏈上點權的總和，sum是子樹的權重和，
        data是節點的權重
113     access(u);
114     int lca=access(v);
115     splay(u);
116     if(u==lca){
117         //return nd[lca].data+nd[nd[lca].ch[1]].
            sum
118     }else{
119         //return nd[lca].data+nd[nd[lca].ch[1]].
            sum+nd[u].sum
120     }
121 }
122 struct EDGE{
123     int a,b,w;
124 }e[10005];
125 int n;
126 vector<pair<int,int>> G[10005];
127 //first表示子節點，second表示邊的編號
128 int pa[10005],edge_node[10005];
129 //pa是父母節點，暫存用的，edge_node是每個編
        被存在哪個點裡面的陣列
130 void bfs(int root){
131     //在建構的時候把每個點都設成一個splay tree
132     queue<int> q;
133     for(int i=1;i<=n;++i)pa[i]=0;

```

```

134 q.push(root);
135 while(q.size()){
136     int u=q.front();
137     q.pop();
138     for(auto P:G[u]){
139         int v=P.first;
140         if(v!=pa[u]){
141             pa[v]=u;
142             nd[v].pa=u;
143             nd[v].data=e[P.second].w;
144             edge_node[P.second]=v;
145             up(v);
146             q.push(v);
147         }
148     }
149 }
150 }
151 void change(int x,int b){
152     splay(x);
153     //nd[x].data=b;
154     up(x);
155 }

```

10.4 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,
15     int d){
16     dis.push_back(d);
17     for(size_t i=0;i<g[u].size();++i){
18         int v=g[u][i].first,w=g[u][i].second;
19         if(v!=pa&&!vis[v])get_dis(dis,v,u,d+w);
20     }
21 }
22 vector<int> dis; //這東西如果放在函數裡會TLE
23 int cal(int u,int d){
24     dis.clear();
25     get_dis(dis,u,-1,d);
26     sort(dis.begin(),dis.end());
27     int l=0,r=dis.size()-1,res=0;
28     while(l<r){
29         while(l<r&&dis[l]+dis[r]>k)--r;
30         res+=r-(l++);
31     }
32     return res;
33 }
34 pair<int,int> tree_centroid(int u,int pa,
35     const int sz){
36     size[u]=1; //找樹重心
37     pair<int,int> res(INT_MAX,-1);

```

```

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

```

11 default

11.1 debug

```

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

```

11.2 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>,
8     rb_tree_tag,
9     tree_order_statistics_node_update>;
10 template<typename T,typename U>
11 using pbds_map = tree<T,U,less<T>,
12     rb_tree_tag,
13     tree_order_statistics_node_update>;
14 using heap=__gnu_pbds::priority_queue<int>;
15 //s.find_by_order(1); //0 base
16 //s.order_of_key(1);

```

11.3 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:102400000,102400000")
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         rl.rlim_cur=ks;
15         res=setrlimit(RLIMIT_STACK,&rl);
16     }
17 }

```

11.4 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

```

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<3)
4         return (d+2*m+3*(m+1)/5+y+y/4+5)%7;
5     return (d+2*m+3*(m+1)/5+y+y/4-y/100+y/400)%7;
6 }

```

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-1+l>d;++l){
14             if(da.front()==1)da.pop_front();
15             if(db.front()==1)db.pop_front();
16             if(da.size()&&db.size()){
17                 d=a[da.front()]+b[db.front()];
18             }
19         }
20         ans=max(ans,r-l+1);
21     }
22     printf("%d\n",ans);
23 }

```

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.first);
13         }
14         if(h>st.top().first){

```

```

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

```

13 other language

13.1 java

13.1.1 文件操作

```

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

```

13.1.2 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    });

```

13.1.3 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 }

```

13.1.4 sort

```

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

```

13.2 python heap

```

1 import heapq
2
3 heap = [7,1,2,2]
4 heapq.heapify(heap)
5 print(heap) # [1, 2, 2, 7]
6 heapq.heappush(heap, 5)
7 print(heap) # [1, 2, 2, 7, 5]
8 print(heapq.heappop(heap)) # 1
9 print(heap) # [2, 2, 5, 7]

```

13.3 python input

```

1 ans = sum(map(float, input().split()))
2 # input: 1.1 2.2 3.3 4.4 5.5
3 print(ans) # 16.5
4
5 (n, m) = map(int, input().split()) # 300 200
6 print(n * m) # 60000
7
8 Arr = list(map(int, input().split()))
9 # input: 1 2 3 4 5
10 print(Arr) # [1, 2, 3, 4, 5]

```

13.4 python output

```

1 hello = 'Hello'
2 world = 7122
3 print(f'{hello} {world}') # Hello 7122
4
5 import math
6 print(f'PI is approximately {math.pi:.3f}.')
7 # PI is approximately 3.142.
8
9 print('AAA {} BBB "{}!"'.format('Jin', 'Kela'))
10 # AAA Jin BBB "Kela!"
11
12 hello = 'hello, world\n'
13 hellos = repr(hello)
14 print(hellos) # 'hello, world\n'
15
16 x = 32.5
17 y = 40000
18 print(repr((x, y, ('spam', 'eggs'))))
19 # "(32.5, 40000, ('spam', 'eggs'))"
20
21 x = 7
22 print(eval('3 * x')) # 21

```

14 zformula

14.1 formula

14.1.1 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

14.1.2 圖論

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

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

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

- 對於連通二分圖：

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

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

- 最大權閉子圖：

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

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

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

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

- 最大密度子圖：

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

- $U = \sum_{v \in V} 2W_v + \sum_{e \in E} W_e$
- $C(u, v) = W_{(u, v)}, (u, v) \in E$ ，雙向邊
- $C(S, v) = U, v \in V$
- $D_u = \sum_{(u, v) \in E} W_{(u, v)}$
- $C(v, T) = U + 2g - D_v - 2W_v, v \in V$
- 二分搜 g ：
 $l = 0, r = U, \text{eps} = 1/n^2$
if $(U \times |V| - \text{flow}(S, T))/2 > 0$ $l = \text{mid}$
else $r = \text{mid}$
- $\text{ans} = \min_cut(S, T)$
- $|E| = 0$ 要特殊判斷

7. 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，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

14.1.3 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： $O \left(V^{1/2} E \right)$

14.1.4 0-1 分數規劃

$x_i \in \{0, 1\}$ ， x_i 可能會有其他限制，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， $f(g) < 0$ 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 | }

```

14.1.5 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 $H_n = \ln(n) + \gamma + 1/(2n) - 1/(12n^2) + 1/(120n^4)$
- $\gamma = 0.57721566490153286060651209008240243104215$
- 格雷碼 $= n \oplus (n >> 1)$
- $SG(A+B) = SG(A) \oplus SG(B)$
- 選轉矩陣 $M(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

14.1.6 基本數論

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

14.1.7 排組公式

- k 卡特蘭 $\frac{C_n^{kn}}{n(k-1)+1} \cdot C_m^n = \frac{n!}{m!(n-m)!}$
- $H(n, m) \cong x_1 + x_2 \dots + x_n = k, num = C_k^{n+k-1}$
- Stirling number of 2^{n^d} , n 入分 k 組方法數目
 - $S(0, 0) = S(n, n) = 1$
 - $S(n, 0) = 0$
 - $S(n, k) = kS(n-1, k) + S(n-1, k-1)$
- Bell number, n 入分任意多組方法數目
 - $B_0 = 1$
 - $B_n = \sum_{i=0}^n S(n, i)$
 - $B_{n+1} = \sum_{k=0}^n C_k^n B_k$
 - $B_{p+n} = B_n + B_{n+1} \bmod p$, p is prime
 - $B_{p^m+n} \equiv mB_n + B_{n+1} \bmod p$, p is prime
 - From $B_0 : 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975$
- Derangement, 錯排, 沒有人在自己位置上
 - $D_n = n!(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} \dots + (-1)^n \frac{1}{n!})$
 - $D_n = (n-1)(D_{n-1} + D_{n-2}), D_0 = 1, D_1 = 0$
 - From $D_0 : 1, 0, 1, 2, 9, 44, 265, 1854, 14833, 133496$
- Binomial Equality
 - $\sum_k \binom{r}{m+k} \binom{s}{n-k} = \binom{r+s}{m+n}$

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

14.1.8 冪次, 冪次和

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

14.1.9 Burnside's lemma

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

14.1.10 Count on a tree

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

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

15 Интернационал

15.1 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 tí, 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!

```

```

50 | ¡Mátenlos!
51 | ¡Te voy a romper en pedazos!
52 | ¡la campana!
53 | Ya es hora de rezar.
54 | Tenemos que irnos.
55 | ¡Maldita sea, mierda!
56 | ¡Ya es hora de aplastar!
57 | ¡Mierda!
58 | ¡Puedes correr, pero no te puedes esconder!
59 | ¡Sos cerdo!
60 | ¡Está en la trampa!
61 | ¡Ah, que madre!
62 | ¡Vámonos!
63 | ¡Ándale!
64 | ¡Cabrón!
65 | ¡Coño!
66 | ¡Agárrenlo!
67 | Cógerlo, Cógerlo...
68 | ¡Allí está, mávalo!
69 | ¡No dejas que se escape de la isla vivo!
70 | ¡Hasta luego!
71 | ¡Rápido, es un intruso!

```

15.2 Интернационал

```

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 насилья мы разрушим
22 | До основанья, а затем
23 | Мы наш, мы новый мир построим, –
24 | Кто был ничем, тот станет всем.
25 |
26 | Chorus
27 | Это есть наш последний
28 | И решительный бой;
29 | С Интернационалом
30 | Воспримет род людской!
31 |
32 | Никто не даст нам избавленья:
33 | Ни бог, ни царь и не герой!
34 | Добьёмся мы освобожденья
35 | Своею собственной рукой.
36 | Чтоб свергнуть гнёт рукой умелой,
37 | Отвоевать своё добро, –
38 | Вдувайте горн и куйте смело,

```


39 Пока железо горячо!
40
41 Chorus
42
43 Довольно кровь сосать, вампиры,
44 Тирьмой, налогом, нищетой!
45 У вас — вся власть, все блага мира,
46 А наше право — звук пустой!
47 Мы жизнь построим по-иному —
48 И вот наш лозунг боевой:
49 Вся власть народу трудовому!
50 А дармоедов всех долой!
51
52 Chorus
53
54 Презренны вы в своём богатстве,
55 Угля и стали короли!
56 Вы ваши троны, тунейдцы,
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 героически 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

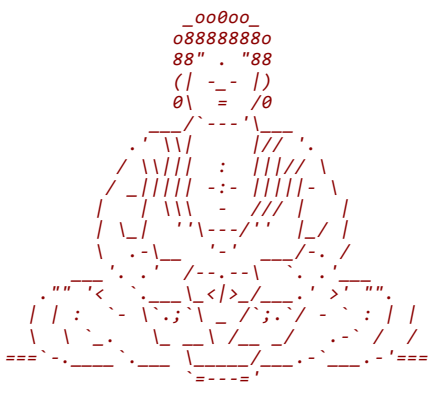
15.3 保佑

1
2
3
4
5
6
7
8
9
10
11
12
13

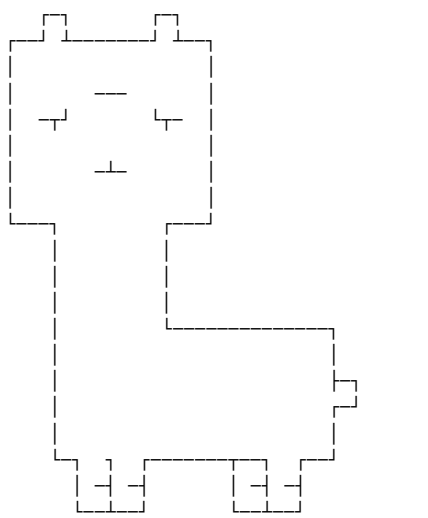
```

14 //
15 //
16 //
17 //
18 //
19 //
20 //
21 //
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 #

```



佛祖保佑 永無BUG



神獸保佑 永無BUG!

72			
73	//	##	#####
74	//	##	##
75	//	##	##
76	//	##	##
77	//	##	##
78	//	##	##
79	//	##	##
80	//	##	##
81	//	##	##
82	//	##	##
83	//	##	##
84	//	##	##
85	//	#####	#####
86	//	##	##
87	//	##	##
88	//	##	##
89	//	##	##
90	//	##	##
91	//	##	##
92	//	##	##
93	//	#####	##
94	//		
95	//	元首保佑 永無BUG	
96			
97	//		
98	//		
99	//	< @ \	
100	//	_/_/_/_/_/ */_/_/_/_/_/	
101	//	_/_/_/_/_/ */_/_/_/_/_/	
102	//	u/u/u/ */_/_/_/_/_/	
103	//	u/u/u/ */_/_/_/_/_/	
104	//	* * *	
105	//		
106	//	/+--+ \	
107	//	卐	
108	//	\\==//	
109	//		
110	//	神獸保佑 永無BUG	

ACM ICPC Team Reference - Angry Crow Takes Flight!

Contents

1 Computational Geometry	1	3.4 MinCostMaxFlow	6	7.8 Lucas	13	11.4 input	18
1.1 delaunay	1	4 Graph	7	7.9 Matrix	13	12 other	18
1.2 Geometry	1	4.1 Augmenting Path	7	7.10 MillerRobin	13	12.1 WhatDay	18
1.3 SmallestCircle	3	4.2 Augmenting Path multiple	7	7.11 NTT	13	12.2 上下最大正方形	18
1.4 最近點對	3	4.3 blossom matching	7	7.12 Simpson	14	12.3 最大矩形	18
2 Data Structure	3	4.4 BronKerbosch	7	7.13 外星模運算	14	13 other language	19
2.1 CDQ DP	3	4.5 graphISO	7	7.14 數位統計	14	13.1 java	19
2.2 DLX	4	4.6 is planar	8	7.15 質因數分解	14	13.1.1 文件操作	19
2.3 Dynamic KD tree	4	4.7 KM	8	8 String	14	13.1.2 優先隊列	19
2.4 kd tree replace segment tree	5	4.8 MaximumClique	9	8.1 AC 自動機	14	13.1.3 Map	19
2.5 reference point	5	4.9 MinimumMeanCycle	9	8.2 hash	15	13.1.4 sort	19
2.6 skew heap	5	4.10 Rectilinear MST	9	8.3 KMP	15	13.2 python heap	19
2.7 undo disjoint set	5	4.11 treeISO	9	8.4 manacher	15	13.3 python input	19
2.8 整體二分	6	4.12 一般圖最小權完美匹配	9	8.5 minimal string rotation	15	13.4 python output	19
3 Flow	6	4.13 全局最小割	10	8.6 reverseBWT	15	14 zformula	19
3.1 dinic	6	4.14 弦圖完美消除序列	10	8.7 suffix array lcp	15	14.1 formula	19
3.2 Gomory Hu	6	4.15 最小斯坦納樹 DP	10	8.8 Z	16	14.1.1 Pick 公式	19
3.3 ISAP with cut	6	4.16 最小樹形圖朱劉	10	9 Tarjan	16	14.1.2 圖論	19
		4.17 穩定婚姻模板	10	9.1 dominator tree	16	14.1.3 dinic 特殊圖複雜度	19
		5 Language	11	9.2 tnfsb017 2 sat	16	14.1.4 0-1 分數規劃	19
		5.1 CNF	11	9.3 橋連通分量	16	14.1.5 學長公式	20
		6 Linear Programming	11	9.4 雙連通分量 & 割點	16	14.1.6 基本數論	20
		6.1 simplex	11	10 Tree Problem	17	14.1.7 排組公式	20
		7 Number Theory	11	10.1 HeavyLight	17	14.1.8 冪次, 冪次和	20
		7.1 basic	11	10.2 LCA	17	14.1.9 Burnside's lemma	20
		7.2 bit set	12	10.3 link cut tree	17	14.1.10 Count on a tree	20
		7.3 cantor expansion	12	10.4 POJ tree	18	15 Интернационал	20
		7.4 FFT	12	11 default	18	15.1 ganadoQuote	20
		7.5 find real root	12	11.1 debug	18	15.2 Интернационал	20
		7.6 FWT	13	11.2 ext	18	15.3 保佑	21
		7.7 LinearCongruence	13	11.3 IncStack	18		

ACM ICPC Judge Test - Angry Crow Takes Flight!

C++ Resource Test

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 namespace system_test {
5
6 const size_t KB = 1024;
7 const size_t MB = KB * 1024;
8 const size_t GB = MB * 1024;
```

```
9 size_t block_size, bound;
10 void stack_size_dfs(size_t depth = 1) {
11     if (depth >= bound)
12         return;
13     int8_t ptr[block_size]; // 若無法編譯將
14                             // block_size 改成常數
15     memset(ptr, 'a', block_size);
16     cout << depth << endl;
17     stack_size_dfs(depth + 1);
18 }
19
20 void stack_size_and_runtime_error(size_t
21     block_size, size_t bound = 1024) {
22     system_test::block_size = block_size;
23     system_test::bound = bound;
24     stack_size_dfs();
25 }
26
27 double speed(int iter_num) {
28     const int block_size = 1024;
29     volatile int A[block_size];
30     auto begin = chrono::high_resolution_clock
31         ::now();
32     while (iter_num--)
33         for (int j = 0; j < block_size; ++j)
34             A[j] += j;
35     auto end = chrono::high_resolution_clock::
36         now();
```

```
37 chrono::duration<double> diff = end -
38     begin;
39     return diff.count();
40 }
41
42 void runtime_error_1() {
43     // Segmentation fault
44     int *ptr = nullptr;
45     *(ptr + 7122) = 7122;
46 }
47
48 void runtime_error_2() {
49     // Segmentation fault
50     int *ptr = (int *)memset;
51     *ptr = 7122;
52 }
53
54 void runtime_error_3() {
55     // munmap_chunk(): invalid pointer
56     int *ptr = (int *)memset;
57     delete ptr;
58 }
59
60 void runtime_error_4() {
61     // free(): invalid pointer
62     int *ptr = new int[7122];
63     ptr += 1;
64     delete[] ptr;
65 }
```

```
62
63 void runtime_error_5() {
64     // maybe illegal instruction
65     int a = 7122, b = 0;
66     cout << (a / b) << endl;
67 }
68
69 void runtime_error_6() {
70     // floating point exception
71     volatile int a = 7122, b = 0;
72     cout << (a / b) << endl;
73 }
74
75 void runtime_error_7() {
76     // call to abort.
77     assert(false);
78 }
79
80 } // namespace system_test
81
82 #include <sys/resource.h>
83 void print_stack_limit() { // only work in
84     Linux
85     struct rlimit l;
86     getrlimit(RLIMIT_STACK, &l);
87     cout << "stack_size = " << l.rlim_cur << "
88         byte" << endl;
89 }
```