

1 Computational_Geometry

1.1 Geometry.cpp

```

1 const double PI=atan2(0.0,-1.0);
2 template<typename T>
3 struct point{
4     T x,y;
5     point(){
6     point(const T&x,const T&y):x(x),y(y){
7     point operator+(const point &b)const{
8         return point(x+b.x,y+b.y);
9     point operator-(const point &b)const{
10        return point(x-b.x,y-b.y);
11    point operator*(const T &b)const{
12        return point(x*b,y*b);
13    point operator/(const T &b)const{
14        return point(x/b,y/b);
15    bool operator==(const point &b)const{
16        return x==b.x&&y==b.y;
17    T dot(const point &b)const{
18        return x*b.x+y*b.y;
19    T cross(const point &b)const{
20        return x*b.y-y*b.x;
21    point normal()const{//求法向量
22        return point(-y,x);
23    T abs2()const{//向量長度的平方
24        return dot(*this);
25    }
26    T rad(const point &b)const{//兩向量的弧度
27        return fabs(atan2(fabs(cross(b)),dot(b)));
28    }
29    T getA()const{//對x軸的弧度
30        T A=atan2(y,x);{//超過180度會變負的
31        if(A<=-PI/2)A+=PI*2;
32        return A;
33    }
34 };
35 template<typename T>
36 struct line{
37     line(){
38     point<T> p1,p2;
39     T a,b,c;//ax+by+c=0
40     line(const point<T>&x,const point<T>&y):p1(x),p2(y){
41     void pton()const{//轉成一般式
42         a=p1.y-p2.y;
43         b=p2.x-p1.x;
44         c=-a*p1.x-b*p1.y;
45     }
46     T cross(const point<T> &p)const{//點和有向
47         //直線的關係，>0左邊、=0在線上、<0右邊
48         return (p2-p1).cross(p-p1);
49     }
50     bool point_on_segment(const point<T>&p)
51         const{//點是否線段上
52         return cross(p)==0&&(p1-p).dot(p2-p)<=0;
53     }
54     T dis2(const point<T> &p,bool is_segment
55         =0)const{//點跟直線/線段的距離平方
56     point<T> v=p2-p1,v1=p-p1;
57     if(is_segment){
58         point<T> v2=p-p2;
59         if(v.dot(v1)<=0)return v1.abs2();
60         if(v.dot(v2)>=0)return v2.abs2();
61     }
62     T tmp=v.cross(v1);
63     return tmp*tmp/v.abs2();
64 }
65 T seg_dis2(const line<T> &l)const{//兩線段
66     //距離平方
67     return min({dis2(l.p1,1),dis2(l.p2,1),l.
68         dis2(p1,1),l.dis2(p2,1)});
69 }
70 point<T> projection(const point<T> &p)
71     const{//點對直線的投影
72     point<T> n=(p2-p1).normal();
73     return p-n*(p-p1).dot(n)/n.abs2();
74 }
75 point<T> mirror(const point<T> &p)const{//
76     //點對直線的鏡射
77     //要先呼叫pton轉成一般式
78     point<T> ans;
79     T d=a*a+b*b;
80     ans.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/
81         d;
82     ans.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/
83         d;
84     return ans;
85 }
86 bool equal(const line &l)const{//直線相等
87     return cross(l.p1)==0&&cross(l.p2)==0;
88 }
89 bool parallel(const line &l)const{//直線平
90     //行
91     return (p1-p2).cross(l.p1-l.p2)==0;
92 }
93 bool cross_seg(const line &l)const{//直線
94     //是否交線段
95     return (p2-p1).cross(l.p1-p1)*(p2-p1).
96         cross(l.p2-p1)<=0;
97 }
98 char line_intersect(const line &l)const{//
99     //直線相交情況，-1無限多點、1交於一點、0
100     //不相交
101     return parallel(l)?(cross(l.p1)==0?-1:0)
102         :1;
103 }
104 char seg_intersect(const line &l)const{//
105     //線段相交情況，-1無限多點、1交於一點、0
106     //不相交
107     T c1=(p2-p1).cross(l.p1-p1);
108     T c2=(p2-p1).cross(l.p2-p1);
109     T c3=(l.p2-l.p1).cross(p1-l.p1);
110     T c4=(l.p2-l.p1).cross(p2-l.p1);
111     if(c1==0&&c2==0){
112         if(p1==l.p1&&(p2-p1).dot(l.p2)<=0)
113             return 1;
114         if(p1==l.p2&&(p2-p1).dot(l.p1)<=0)
115             return 1;
116         if(p2==l.p1&&(p1-p2).dot(l.p2)<=0)
117             return 1;
118         if(p2==l.p2&&(p1-p2).dot(l.p1)<=0)
119             return 1;
120         return 1;
121     }
122     if(c1==0&&c2==0){
123         if(p1==l.p1&&(p2-p1).dot(l.p2)<=0)
124             return 1;
125         if(p1==l.p2&&(p2-p1).dot(l.p1)<=0)
126             return 1;
127         if(p2==l.p1&&(p1-p2).dot(l.p2)<=0)
128             return 1;
129         if(p2==l.p2&&(p1-p2).dot(l.p1)<=0)
130             return 1;
131         return 1;
132     }
133     if(c1*c2<=0&&c3*c4<=0)return 1;
134     return 0;
135 }
136 point<T> line_intersection(const line &l)
137     const{//直線交點
138     point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
139     //if(a.cross(b)==0)return INF;
140     return p1+a*s.cross(b)/a.cross(b);
141 }
142 point<T> seg_intersection(const line &l)
143     const{//線段交點
144     T c1=(p2-p1).cross(l.p1-p1);
145     T c2=(p2-p1).cross(l.p2-p1);
146     T c3=(l.p2-l.p1).cross(p1-l.p1);
147     T c4=(l.p2-l.p1).cross(p2-l.p1);
148     if(c1==0&&c2==0){
149         if(p1==l.p1&&(p2-p1).dot(l.p2)<=0)
150             return p1;
151         if(p1==l.p2&&(p2-p1).dot(l.p1)<=0)
152             return p1;
153         if(p2==l.p1&&(p1-p2).dot(l.p2)<=0)
154             return p2;
155         if(p2==l.p2&&(p1-p2).dot(l.p1)<=0)
156             return p2;
157     }
158     else if(c1*c2<=0&&c3*c4<=0)return
159         line_intersection(l);
160     //return INF;
161 }
162 };
163 template<typename T>
164 struct polygon{
165     polygon(){
166     vector<point<T>> p;//逆時針順序
167     T area()const{//面積
168         T ans=0;
169         for(int i=p.size()-1,j=0;j<(int)p.size()
170             ;i=j++){
171             ans+=p[i].cross(p[j]);
172         }
173         return ans/2;
174     }
175     point<T> center_of_mass()const{//重心
176     T cx=0,cy=0,w=0;
177     for(int i=p.size()-1,j=0;j<(int)p.size()
178         ;i=j++){
179         T a=p[i].cross(p[j]);
180         cx+=(p[i].x+p[j].x)*a;
181         cy+=(p[i].y+p[j].y)*a;
182         w+=a;
183     }
184     return point<T>(cx/3/w,cy/3/w);
185 }
186 }
187 char ahas(const point<T>&t)const{//點是否
188     //在簡單多邊形內，是的話回傳1、在邊上回
189     //傳-1、否則回傳0
190     bool c=0;
191     for(int i=0,j=p.size()-1;i<p.size();i=i
192         ++){
193         if(line<T>(p[i],p[j]).point_on_segment
194             (t))return -1;
195         else if((p[i].y>t.y)!=p[j].y>t.y)&&
196             t.x<(p[j].x+p[i].x)*(t.y-p[i].y)/(p[j]
197                 .y-p[i].y)+p[i].x)
198             c=!c;
199     }
200     return c;
201 }
202 char point_in_convex(const point<T>&x)
203     const{
204     int l=1,r=(int)p.size()-2;
205     while(l<=r){//點是否在凸多邊形內，是的話
206         //回傳1、在邊上回傳-1、否則回傳0
207         int mid=(l+r)/2;
208         T a1=(p[mid]-p[0]).cross(x-p[0]);
209         T a2=(p[mid+1]-p[0]).cross(x-p[0]);
210         if(a1>=0&&a2<=0){
211             T res=(p[mid+1]-p[mid]).cross(x-p[
212                 mid]);
213             return res>0?1:(res>=0?-1:0);
214         }
215         else if(a1<0)r=mid-1;
216         else l=mid+1;
217     }
218     return 0;
219 }
220 vector<T> getA()const{//凸包邊對x軸的夾角
221     vector<T> res;//一定是遞增的
222     for(size_t i=0;i<p.size();i++){
223         res.push_back((p[(i+1)%p.size()]-p[i])
224             .getA());
225     }
226     return res;
227 }
228 bool line_intersect(const vector<T>&A,
229     const line<T> &l)const{//O(LogN)
230     int f1=upper_bound(A.begin(),A.end(),(l.
231         p1-l.p2).getA())-A.begin();
232     int f2=upper_bound(A.begin(),A.end(),(l.
233         p2-l.p1).getA())-A.begin();
234     return l.cross_seg(line<T>(p[f1],p[f2]));
235 }
236 }
237 polygon cut(const line<T> &l)const{//凸包
238     //對直線切割，得到直線L左側的凸包
239     polygon ans;
240     for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
241         if(l.cross(p[i])>=0){
242             ans.p.push_back(p[i]);
243             if(l.cross(p[j])<0)
244                 ans.p.push_back(l.
245                     line_intersection(line<T>(p[i]
246                         ,p[j])));
247         }
248         else if(l.cross(p[j])>=0)
249             ans.p.push_back(l.line_intersection(
250                 line<T>(p[i],p[j])));
251     }
252     return ans;
253 }
254 static bool graham_cmp(const point<T>&a,
255     const point<T>&b){
256     return (a.x<b.x)||a.x==b.x&&a.y<b.y;//
257     //凸包排序函數
258 }
259 void graham(vector<point<T>> &s){//凸包
260     sort(s.begin(),s.end(),graham_cmp);
261     p.resize(s.size()+1);
262     int m=0;
263     for(int i=0;i<(int)s.size();i++){
264         while(m>2&&(p[m-1]-p[m-2]).cross(s[i]
265             -p[m-2])<=0)-m;
266         p[m++]=s[i];
267     }

```

```

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

```

```

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

```

```

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

```

1.2 SmallestCircle.cpp

```

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

```

```

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

```

1.3 最近點對.cpp

```

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

```

```

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

```

1.4 浮點數誤差模板.cpp

```

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

```

2 Data_Structure

2.1 DLX.cpp

```

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

```

```

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

```

```

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

```

2.2 Dynamic_KD_tree.cpp

```

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

```

```

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

```

```

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

```


2.3 kd_tree_replace_segment

```

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

```

/*kd樹代替高維線段樹*/

```

257 struct node{
258     node *l, *r;
259     point pid, mi, ma;
260     int s;
261     int data;
262     node(const point &p, int d): l(0), r(0), pid(p
263         ), mi(p), ma(p), s(1), data(d), dmin(d),
264         dmax(d){}
265     void up(){
266         mi=ma=pid;
267         s=1;
268         if(l){
269             for(int i=0; i<kd; ++i){
270                 mi.d[i]=min(mi.d[i], l->mi.d[i]);
271                 ma.d[i]=max(ma.d[i], l->ma.d[i]);
272             }
273             s+=l->s;
274         }
275         if(r){
276             for(int i=0; i<kd; ++i){
277                 mi.d[i]=min(mi.d[i], r->mi.d[i]);
278                 ma.d[i]=max(ma.d[i], r->ma.d[i]);
279             }
280             s+=r->s;
281         }
282     }
283     void up2(){
284         //其他懶惰標記向上更新
285     }
286     void down(){
287         //其他懶惰標記下推
288     }
289 } *root;
290 /*檢查區間包含用的函數*/
291 inline bool range_include(node *o, const
292     point &L, const point &R){
293     for(int i=0; i<kd; ++i){
294         if(L.d[i]>o->ma.d[i] || R.d[i]<o->mi.d[i])
295             return 0;
296     }
297     //只要(L,R)區間有和o的區間有交集就回傳
298     true
299     return 1;
300 }
301 inline bool range_in_range(node *o, const
302     point &L, const point &R){
303     for(int i=0; i<kd; ++i){
304         if(L.d[i]>o->mi.d[i] || o->ma.d[i]>R.d[i])
305             return 0;
306     }
307     //如果(L,R)區間完全包含o的區間就回傳true
308     return 1;
309 }
310 inline bool point_in_range(node *o, const
311     point &L, const point &R){
312     for(int i=0; i<kd; ++i){
313         if(L.d[i]>o->pid.d[i] || R.d[i]<o->pid.d[i]
314             ) return 0;
315     }
316     //如果(L,R)區間完全包含o->pid這個點就回傳
317     true
318     return 1;
319 }
320 }

```

/*單點修改 · 以單點改值為例*/

```

321 void update(node *u, const point &x, int data,
322             int k=0){
323     if(!u) return;
324     u->down();
325     if(u->pid==x){
326         u->data=data;
327         u->up2();
328         return;
329     }
330     cmp.sort_id=k;
331     update(cmp(x, u->pid)?u->l:u->r, x, data, (k
332         +1)%kd);
333     u->up2();
334 }
335 /*區間修改*/
336 void update(node *o, const point &L, const
337     point &R, int data){
338     if(!o) return;
339     o->down();
340     if(range_in_range(o, L, R)){
341         //區間懶惰標記修改
342         o->down();
343         return;
344     }
345     if(point_in_range(o, L, R)){
346         //這個點在(L,R)區間 · 但是他的左右子樹不
347         一定在區間中
348         //單點懶惰標記修改
349         if(o->l && range_include(o->l, L, R)) update(o
350             ->l, L, R, data);
351         if(o->r && range_include(o->r, L, R)) update(o
352             ->r, L, R, data);
353         o->up2();
354     }
355     //區間查詢 · 以總和為例*/
356 int query(node *o, const point &L, const point
357     &R){
358     if(!o) return 0;
359     o->down();
360     if(range_in_range(o, L, R)) return o->sum;
361     int ans=0;
362     if(point_in_range(o, L, R)) ans+=o->data;
363     if(o->l && range_include(o->l, L, R)) ans+=
364         query(o->l, L, R);
365     if(o->r && range_include(o->r, L, R)) ans+=
366         query(o->r, L, R);
367     return ans;
368 }

```

2.4 persistent_segment_tree.cpp

```

369 #include <bits/stdc++.h> //POJ 2104
370 using namespace std;
371 struct node{
372     int l, r;
373     int data;

```

```

374     node(int l, int r, int d): l(l), r(r), data(d)
375     {}
376 };
377 vector<node> nds;
378 inline void up(int o, int l, int r){
379     nds[o].data=nds[l].data+nds[r].data;
380 }
381 inline int new_node(int l, int r, int d){
382     nds.push_back(node(l, r, d));
383     return nds.size()-1;
384 }
385 inline int new_node(const node &nd){
386     nds.push_back(nd);
387     return nds.size()-1;
388 }
389 int build_tree(int l, int r){
390     int nd=new_node(-1, -1, 0);
391     if(l==r) return nd;
392     int mid=(l+r)/2;
393     int L=build_tree(l, mid); //執行時vector會被
394     重構
395     int R=build_tree(mid+1, r); //一定要這樣寫
396     nds[nd].l=L;
397     nds[nd].r=R;
398     //up(nd, L, R);
399     return nd;
400 }
401 int insert(int l, int r, int rt, int x, int d){
402     if(x<l || r<x) return rt;
403     int nd=new_node(nds[rt]);
404     if(l==r && l==x) nds[nd].data+=d;
405     else{
406         int mid=(l+r)/2;
407         int L=insert(l, mid, nds[nd].l, x, d);
408         int R=insert(mid+1, r, nds[nd].r, x, d);
409         nds[nd].l=L;
410         nds[nd].r=R;
411         up(nd, L, R);
412     }
413     return nd;
414 }
415 inline int cal(int L, int R){
416     return nds[R].data-nds[L].data;
417 }
418 int find(int l, int r, int L, int R, int k){
419     if(l==r) return l;
420     int mid=(l+r)/2;
421     int add=cal(nds[L].l, nds[R].l);
422     if(k<=add) return find(l, mid, nds[L].l, nds[R
423         ].l, k);
424     return find(mid+1, r, nds[L].r, nds[R].r, k-
425         add);
426 }
427 int n, m;
428 int s[100005];
429 int root[100005];
430 int main(){
431     while(~scanf("%d%d", &n, &m)){
432         nds.clear();
433         vector<int> lsh;
434         for(int i=1; i<=n; ++i){
435             scanf("%d", &s[i]);
436             lsh.push_back(s[i]);
437         }
438         sort(lsh.begin(), lsh.end());

```

```

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

```

2.5 reference_point.cpp

```

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

```

2.6 skew_heap.cpp

```

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

```

2.7 split_merge.cpp

```

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

```

2.8 treap.cpp

```

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

```

```

26 }else{
27     o->s++;
28     bool d=o->data<data;
29     insert(o->ch[d],data);
30     if(o->ch[d]->fix>o->fix)rotate(o,!d);
31 }
32 }
33 node *merge(node *a,node *b){
34     if(!a->s||!b->s)return a->s?a:b;
35     if(a->fix>b->fix){
36         a->ch[1]=merge(a->ch[1],b);
37         a->s=a->ch[0]->s+a->ch[1]->s+1;
38         return a;
39     }else{
40         b->ch[0]=merge(a,b->ch[0]);
41         b->s=b->ch[0]->s+b->ch[1]->s+1;
42         return b;
43     }
44 }
45 bool erase(node *o,const T &data){
46     if(!o->s)return 0;
47     if(o->data==data){
48         node *t=o;
49         o=merge(o->ch[0],o->ch[1]);
50         delete t;
51         return 1;
52     }
53     if(erase(o->ch[o->data<data],data)){
54         o->s--;return 1;
55     }else return 0;
56 }
57 void clear(node *o){
58     if(o->s)clear(o->ch[0]),clear(o->ch
59         [1]),delete o;
60 }
61 public:
62     treap(unsigned s=20150119):nil(new node
63         ,root(nil),x(s){}
64     ~treap(){clear(root),delete nil;}
65     void clear(){clear(root),root=nil;}
66     void insert(const T &data){
67         insert(root,data);
68     }
69     bool erase(const T &data){
70         return erase(root,data);
71     }
72     bool find(const T&data){
73         for(node *o=root;o->s;){
74             if(o->data==data)return 1;
75             else o=o->ch[o->data<data];
76             return 0;
77         }
78     }
79     int rank(const T&data){
80         int cnt=0;
81         for(node *o=root;o->s;){
82             if(o->data<data)cnt+=o->ch[0]->s+1,o=o->ch[1];
83             else o=o->ch[0];
84             return cnt;
85         }
86     }
87     const T&kth(int k){
88         for(node *o=root;){
89             if(k<=o->ch[0]->s)o=o->ch[0];
90             else if(k==o->ch[0]->s+1)return o->
91                 data;

```

```

87     else k=o->ch[0]->s+1,o=o->ch[1];
88 }
89 const T&operator[](int k){
90     return kth(k);
91 }
92 const T&preorder(const T&data){
93     node *x=root,*y=0;
94     while(x->s)
95     if(x->data<data)y=x,x=x->ch[1];
96     else x=x->ch[0];
97     if(y)return y->data;
98     return data;
99 }
100 const T&successor(const T&data){
101     node *x=root,*y=0;
102     while(x->s)
103     if(data<x->data)y=x,x=x->ch[0];
104     else x=x->ch[1];
105     if(y)return y->data;
106     return data;
107 }
108 int size(){return root->s;}
109 };

```

2.9 操作分治.cpp

```

1 void dq(int l,int r){
2     if(l==r)return;
3     int mid=(l+r)/2;
4     dq(l,mid);
5     處理[l,mid]的操作對[mid+1,r]的影響
6     dq(mid+1,r);
7 }

```

2.10 整體二分.cpp

```

1 void BS(int l,int r,vector<Item> &vs){
2     //答案該<L會有的已經做完了
3     if(l==r)整個vs的答案=l;////////
4     int mid=(l+r)/2;
5     do_thing(l,mid);//做答案<=mid會做的事
6     vector<Item> left=vs裡滿足的;
7     vector<Item> right=vs-left;
8     undo_thing(l,mid);
9     BS(l,mid,left);
10    do_thing(l,mid);
11    BS(mid+1,r,right);////////
12 }

```

3 default

3.1 debug.cpp

```

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

```

3.2 ext.cpp

```

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

```

3.3 IncStack.cpp

```

1 //Magic
2 #pragma GCC optimize "Ofast"
3 //stack resize,change esp to rsp if 64-bit
4 system
5 asm("mov %0,%esp\n" ::"g"(mem+1000000));
6 //Linux stack resize
7 #include<sys/resource.h>
8 void increase_stack(){
9     const rlim_t ks=64*1024*1024;
10    struct rlimit rl;
11    int res=getrlimit(RLIMIT_STACK,&rl);
12    if(!res&&rl.rlim_cur<ks){
13        rl.rlim_cur=ks;
14        res=setrlimit(RLIMIT_STACK,&rl);
15    }

```

3.4 input.cpp

```

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

```

4 Flow

4.1 dinic.cpp

```

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

```

```

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

```

4.2 ISAP_with_cut.cpp

```

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

```

```

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

```

4.3 MinCostMaxFlow.cpp

```

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

```

```

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

```

```

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

```

5 Graph

5.1 Augmenting_Path.cpp

```

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

```

5.2 Augmenting_Path_multiple

```

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

```

```

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

```

5.3 blossom_matching.cpp

```

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

```



```

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

```

5.4 graphISO.cpp

```

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

```

5.5 KM.cpp

```

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

```

```

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

```

5.6 MaximumClique.cpp

```

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

```

5.7 MinimumMeanCycle.cpp

```

1  #include<cstdint> //for DBL_MAX
2  int dp[100001][100001];
3  double mnc(int n){
4      int u,v,w;
5      const int inf=0x7f7f7f7f;
6      memset(dp,0x7f,sizeof(dp));
7      memset(dp[0],0,sizeof(dp[0]));
8      for(int i=0;i<n;++i){
9          for(auto e:E){ //tuple<int,int,int>
10             of u,v,w
11             tie(u,v,w)=e;
12             if(dp[i][u]!=inf)
13                 dp[i+1][v]=min(dp[i+1][v],dp
14                     [i][u]+w);
15             }
16         }
17         double res = DBL_MAX;
18         for(int i=1;i<=n;++i){
19             double val = DBL_MIN;
20             for(int j=0;j<n;++j)
21                 val=max(val,double(dp[n][i]-
22                     dp[i][j])/(n-j));
23             res=min(res,val);
24         }
25     }
26     return res;
27 }

```

5.8 Minimum_General_Weighted

```

1  struct Graph {
2      // Minimum General Weighted Matching (
3      // Perfect Match) 0-base
4      static const int MXN = 105;
5
6      int n, edge[MXN][MXN];
7      int match[MXN],dis[MXN],onstk[MXN];
8      vector<int> stk;
9
10     void init(int _n) {
11         n = _n;
12         for (int i=0; i<n; i++)
13             for (int j=0; j<n; j++)
14                 edge[i][j] = 0;
15     }
16     void add_edge(int u, int v, int w) {
17         edge[u][v] = edge[v][u] = w;
18     }
19     bool SPFA(int u){
20         if (onstk[u]) return true;
21         stk.push_back(u);
22         onstk[u] = 1;
23         for (int v=0; v<n; v++){
24             if (u != v && match[u] != v && !onstk[
25                 v]){
26                 int m = match[v];
27                 if (dis[m] > dis[u] - edge[v][m] +
28                     edge[u][v]){
29                     dis[m] = dis[u] - edge[v][m] +
30                         edge[u][v];
31                     onstk[v] = 1;
32                     stk.push_back(v);
33                     if (SPFA(m)) return true;
34                     stk.pop_back();
35                 }
36             }
37         }
38         return false;
39     }
40 }

```

```

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

```

5.9 Rectilinear_Steiner_tree.cpp

```

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

```

```

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

```

5.10 treeISO.cpp

```

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

```

5.11 一般圖最大權匹配.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define INF INT_MAX
4 #define MAXN 400
5 struct edge{
6     int u,v,w;
7     edge(){}
8     edge(int u,int v,int w):u(u),v(v),w(w){}
9 };
10 int n,n_x;
11 edge g[MAXN*2+1][MAXN*2+1];
12 int lab[MAXN*2+1];
13 int match[MAXN*2+1],slack[MAXN*2+1],st[MAXN*2+1],pa[MAXN*2+1];
14 int flower_from[MAXN*2+1][MAXN+1],S[MAXN*2+1],vis[MAXN*2+1];
15 vector<int> flower[MAXN*2+1];
16 queue<int> q;
17 int e_delta(const edge &e){ // does not work inside blossoms
18     return lab[e.u]+lab[e.v]-g[e.u][e.v].w*2;
19 }
20 void update_slack(int u,int x){
21     if(!slack[x]||e_delta(g[u][x])<e_delta(g[slack[x]][x]))slack[x]=u;
22 }
23 void set_slack(int x){
24     slack[x]=0;
25     for(int u=1;u<=n;u++){
26         if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)update_slack(u,x);

```

```

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

```

```

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

```

```

201 int u=q.front(); q.pop(); 202
202 if(S[st[u]]==1) continue; 203
203 for(int v=1; v<=n; ++v) 204
204 if(g[u][v].w>0 && st[u]!=st[v]){ 205
205 if(e_delta(g[u][v])==0){ 206
206 if(on_found_edge(g[u][v])) return 207
207 true; 208
208 }else update_slack(u, st[v]); 209
209 } 210
210 } 211
211 int d=INF; 212
212 for(int b=n+1; b<=n_x; ++b) 213
213 if(st[b]==b && S[b]==1) d=min(d, lab[b]/2) 214
214 ; 215
215 for(int x=1; x<=n_x; ++x) 216
216 if(st[x]==x && slack[x]){ 217
217 if(S[x]==-1) d=min(d, e_delta(g[slack[ 218
218 x]][x])); 219
219 else if(S[x]==0) d=min(d, e_delta(g[ 220
220 slack[x]][x])/2); 221
221 } 222
222 for(int u=1; u<=n; ++u){ 223
223 if(S[st[u]]==0){ 224
224 if(lab[u]<=d) return 0; 225
225 lab[u]=-d; 226
226 }else if(S[st[u]]==1) lab[u]+=d; 227
227 } 228
228 for(int b=n+1; b<=n_x; ++b) 229
229 if(st[b]==b){ 230
230 if(S[st[b]]==0) lab[b]+=d*2; 231
231 else if(S[st[b]]==1) lab[b]-=d*2; 232
232 } 233
233 q=queue<int>(); 234
234 for(int x=1; x<=n_x; ++x) 235
235 if(st[x]==x && slack[x] && st[slack[x]]!=x 236
236 && e_delta(g[slack[x]][x])==0) 237
237 if(on_found_edge(g[slack[x]][x])) 238
238 return true; 239
239 for(int b=n+1; b<=n_x; ++b) 240
240 if(st[b]==b && S[b]==1 && lab[b]==0) 241
241 expand_blossom(b); 242
242 } 243
243 return false; 244
244 } 245
245 pair<long long, int> weight_blossom(){ 246
246 memset(match+1, 0, sizeof(int)*n); 247
247 n_x=n; 248
248 int n_matches=0; 249
249 long long tot_weight=0; 250
250 for(int u=0; u<=n; ++u) st[u]=u, flower[u]. 251
251 clear(); 252
252 int w_max=0; 253
253 for(int u=1; u<=n; ++u) 254
254 for(int v=1; v<=n; ++v){ 255
255 flower_from[u][v]=(u==v?u:0); 256
256 w_max=max(w_max, g[u][v].w); 257
257 } 258
258 for(int u=1; u<=n; ++u) lab[u]=w_max; 259
259 while(matching()) ++n_matches; 260
260 for(int u=1; u<=n; ++u) 261
261 if(match[u] && match[u]<u) 262
262 tot_weight+=g[u][match[u]].w; 263
263 return make_pair(tot_weight, n_matches); 264
264 } 265
265 void init_weight_graph(){ 266
266 for(int u=1; u<=n; ++u) 267

```

```

201 for(int v=1; v<=n; ++v) 202
202 g[u][v]=edge(u, v, 0); 203
203 } 204
204 int main(){ 205
205 int m; 206
206 scanf("%d%d", &n, &m); 207
207 init_weight_graph(); 208
208 for(int i=0; i<m; ++i){ 209
209 int u, v, w; 210
210 scanf("%d%d%d", &u, &v, &w); 211
211 g[u][v].w=g[v][u].w=w; 212
212 } 213
213 printf("%d\n", weight_blossom().first); 214
214 for(int u=1; u<=n; ++u) printf("%d ", match[u 215
215 ]); puts(""); 216
216 return 0; 217
217 } /* 7 20 218
218 5 7 9 3 7 4 3 6 6 2 5 8 5 1 9 1 3 6 6 5 1 219
219 2 7 4 2 3 5 6 4 2 7 1 5 5 4 4 1 3 5 3 9 220
219 7 6 4 2 1 3 4 3 9 6 2 7 4 2 8 6 1 10 221
219 ----- 222
222 28 223
222 6 0 4 3 7 1 5 */ 224

```

5.12 全局最小割.cpp

```

1 const int INF=0x3f3f3f3f;
2 template<typename T>
3 struct stoer_wagner{// 0-base
4 static const int MAXN=150;
5 T g[MAXN][MAXN], dis[MAXN];
6 int nd[MAXN], n, s, t;
7 void init(int _n){
8 n=_n;
9 for(int i=0; i<n; ++i)
10 for(int j=0; j<n; ++j) g[i][j]=0;
11 }
12 void add_edge(int u, int v, T w){
13 g[u][v]=g[v][u]+=w;
14 }
15 T min_cut(){
16 T ans=INF;
17 for(int i=0; i<n; ++i) nd[i]=i;
18 for(int ind, tn=n; tn>1; --tn){
19 for(int i=1; i<tn; ++i) dis[nd[i]]=0;
20 for(int i=1; i<tn; ++i){
21 ind=i;
22 for(int j=i; j<tn; ++j){
23 dis[nd[j]]+=g[nd[i-1]][nd[j]];
24 if(dis[nd[ind]]<dis[nd[j]]) ind=j;
25 }
26 swap(nd[ind], nd[i]);
27 }
28 if(ans>dis[nd[ind]]) ans=dis[t=nd[ind
29 ]], s=nd[ind-1];
30 for(int i=0; i<tn; ++i)
31 g[nd[ind-1]][nd[i]]=g[nd[i]][nd[ind
32 -1]]+=g[nd[i]][nd[ind]];
33 }
34 return ans;
35 }

```

5.13 平面圖判定.cpp

```

1 static const int MAXN = 20;
2 struct Edge{
3 int u, v;
4 Edge(int s, int d) : u(s), v(d) {}
5 };
6 bool isK33(int n, int degree[]){
7 int t = 0, z = 0;
8 for(int i=0; i<n; ++i){
9 if(degree[i] == 3) ++t;
10 else if(degree[i] == 0) ++z;
11 else return false;
12 }
13 return t == 6 && t + z == n;
14 }
15 bool isK5(int n, int degree[]){
16 int f = 0, z = 0;
17 for(int i=0; i<n; ++i){
18 if(degree[i] == 4) ++f;
19 else if(degree[i] == 0) ++z;
20 else return false;
21 }
22 return f == 5 && f + z == n;
23 }
24 // it judge a given graph is Homeomorphic
25 with K3 or K5
26 bool isHomeomorphic(bool G[MAXN][MAXN],
27 const int n){
28 for(;;){
29 int cnt = 0;
30 for(int i=0; i<n; ++i){
31 vector<Edge> E;
32 for(int j=0; j<n && E.size()<3; ++j)
33 if(G[i][j] && i != j)
34 E.push_back(Edge(i, j));
35 if(E.size() == 1){
36 G[i][E[0].v] = G[E[0].v][i] = false;
37 }else if(E.size() == 2){
38 G[i][E[0].v] = G[E[0].v][i] = false;
39 G[i][E[1].v] = G[E[1].v][i] = false;
40 G[E[0].v][E[1].v] = G[E[1].v][E[0].v
41 ] = true;
42 ++cnt;
43 }
44 }
45 if(cnt == 0) break;
46 }
47 static int degree[MAXN];
48 fill(degree, degree + n, 0);
49 for(int i=0; i<n; ++i){
50 for(int j=i+1; j<n; ++j){
51 if(!G[i][j]) continue;
52 ++degree[i];
53 ++degree[j];
54 }
55 }
56 return !(isK33(n, degree) || isK5(n,
57 degree));
58 }

```

5.14 弦圖完美消除序列.cpp

```

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,
19            i));
20        for(int i=n-1; i>=0; --i) for(;;){
21            int u=pq.top().second; pq.pop();
22            if(~rank[u]) continue;
23            rank[u]=i;
24            for(auto v: G[u]) if(rank[v]==-1){
25                pq.push(make_pair(++label[v], v));
26            }
27            break;
28        }
29        vector<int> res(n);
30        for(int i=0; i<n; ++i) res[rank[i]]=i;
31        return res;
32    }
33    bool check(vector<int> ord){ // 弦圖判定
34        for(int i=0; i<n; ++i) rank[ord[i]]=i;
35        memset(mark, 0, sizeof(bool)*n);
36        for(int i=0; i<n; ++i){
37            vector<pair<int, int> > tmp;
38            for(auto u: G[ord[i]]) if(!mark[u])
39                tmp.push_back(make_pair(rank[u], u));
40            sort(tmp.begin(), tmp.end());
41            if(tmp.size()){
42                int u=tmp[0].second;
43                set<int> S;
44                for(auto v: G[u]) S.insert(v);
45                for(size_t j=1; j<tmp.size(); ++j)
46                    if(!S.count(tmp[j].second)) return
47                        0;
48            }
49            mark[ord[i]]=1;
50        }
51        return 1;
52    }
53    };

```

5.15 最小斯坦那樹 DP.cpp

```

1 // POJ 3123
2 // n個點 · m個要構生成樹的點個數
3 // 答案在 dp[1<=m][n]
4 // e表示要構生成樹的點集
5 // O( n^3 + 3^m + 2^m * n^2 )
6 #define REP(i, n) for(int i=0; i<(int)n; ++i)

```

```

7 const int MAXN=30; // 0-base
8 const int MAXM=8;
9 const int INF=0x3f3f3f3f;
10 int dp[1<=MAXM][MAXN];
11 int vis[MAXN];
12 int d[MAXN][MAXN]; // 圖
13 void init(){
14     memset(d, 0x3f, sizeof(d));
15 }
16 void add_edge(int u, int v, int w){
17     d[u][v]=d[v][u]=min(d[u][v], w);
18 }
19 void steiner(int n, int m, int *e){
20     REP(k, n) REP(i, n) REP(j, n) d[i][j]=min(d[i][j],
21         d[i][k]+d[k][j]);
22     memset(dp, 0x3f, sizeof(dp));
23     REP(k, n) d[k][k]=0;
24     REP(i, m) REP(j, n) dp[1<=i][j]=d[e[i]][j];
25     for(int i=1; i<(1<=m); ++i){
26         if(!i & (i-1)) continue;
27         REP(j, n){
28             dp[i][j]=INF;
29             for(int k=i; k; k=(k-1)&1){
30                 dp[i][j]=min(dp[i][j], dp[k][j]+dp[i-
31                     k][j]);
32             }
33             memset(vis, 0, sizeof(vis));
34             REP(j, n){
35                 int c=-1;
36                 REP(k, n) if(!vis[k] & (c==-1 || dp[i][k]<
37                     dp[i][c])) c=k;
38                 vis[c]=true;
39                 REP(k, n) dp[i][c]=min(dp[i][c], dp[i][k]
40                     +d[k][c]);
41             }
42         }
43     }
44 }

```

5.16 最小樹形圖 _ 朱劉.cpp

```

1 #define INF 0x3f3f3f3f
2 template<typename T>
3 struct zhu_liu{
4     static const int MAXN=110;
5     struct edge{
6         int u, v;
7         T w;
8         edge(int u=0, int v=0, T w=0): u(u), v(v), w(w){}
9     };
10    vector<edge> E; // 0-base
11    int pe[MAXN], id[MAXN], vis[MAXN];
12    T in[MAXN];
13    void init(){ E.clear(); }
14    void add_edge(int u, int v, T w){
15        if(u!=v) E.push_back(edge(u, v, w));
16    }
17    T build(int root, int n){
18        T ans=0; int N=n;
19        for(;;){
20            for(int u=0; u<n; ++u) in[u]=INF;
21            for(size_t i=0; i<E.size(); ++i)
22                if(E[i].u!=E[i].v & E[i].w<in[E[i].v])

```

```

23                pe[E[i].v]=i, in[E[i].v]=E[i].w;
24            for(int u=0; u<n; ++u) // 無解
25                if(u!=root & in[u]==INF) return -INF;
26            int cntnode=0;
27            memset(id, -1, sizeof(int)*N);
28            memset(vis, -1, sizeof(int)*N);
29            for(int u=0; u<n; ++u){
30                if(u!=root) ans+=in[u];
31                int v=u;
32                for(;; vis[v]!=u & id[v]==-1 & v!=root; v=
33                    E[pe[v]].u)
34                    vis[v]=u;
35                if(v!=root & id[v]==-1){
36                    for(int x=E[pe[v]].u; x!=v; x=E[pe[x]
37                        ].u)
38                        id[x]=cntnode++;
39                    id[v]=cntnode++;
40                }
41            }
42            if(!cntnode) break; // 無環
43            for(int u=0; u<n; ++u) if(id[u]==-1) id[u]
44                =cntnode++;
45            for(size_t i=0; i<E.size(); ++i){
46                int v=E[i].v;
47                E[i].u=id[E[i].u];
48                E[i].v=id[E[i].v];
49                if(E[i].u!=E[i].v) E[i].w-=in[v];
50            }
51            n=cntnode;
52            root=id[root];
53            return ans;
54        }
55    };

```

5.17 穩定婚姻模板.cpp

```

1 queue<int> Q;
2 for ( i : 所有考生 ) {
3     設定在第0志願;
4     Q.push( 考生i );
5 }
6 while(Q.size()){
7     當前考生=Q.front(); Q.pop();
8     while ( 此考生未分發 ) {
9         指標移到下一志願;
10        if ( 已經沒有志願 or 超出志願總數 )
11            break;
12        計算該考生在該科系加權後的總分;
13        if ( 不符合科系需求 ) continue;
14        if ( 目前科系有餘額 ) {
15            依加權後分數高低順序將考生id加入科系錄取名單中;
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 );

```

6 language

6.1 CNF.cpp

```

1 #define MAXN 55
2 struct CNF{
3     int s, x, y; // s->xy | s->x, if y==1
4     int cost;
5     CNF(){}
6     CNF(int s, int x, int y, int c): s(s), x(x), y(y), cost(c){}
7 };
8 int state; // 規則數量
9 map<char, int> rule; // 每個字元對應到的規則 · 小寫字母為終端字元
10 vector<CNF> cnf;
11 inline void init(){
12     state=0;
13     rule.clear();
14     cnf.clear();
15 }
16 inline void add_to_cnf(char s, const string & p, int cost){
17     // 加入一個 s -> <p> 的文法 · 代價為 cost
18     if(rule.find(s)==rule.end()) rule[s]=state++;
19     for(auto c: p) if(rule.find(c)==rule.end()) rule[c]=state++;
20     if(p.size()==1){
21         cnf.push_back(CNF(rule[s], rule[p[0]], -1, cost));
22     }
23     else{
24         int left=rule[s];
25         int sz=p.size();
26         for(int i=0; i<sz-2; ++i){
27             cnf.push_back(CNF(left, rule[p[i]], state, 0));
28             left=state++;
29             cnf.push_back(CNF(left, rule[p[sz-2]], rule[p[sz-1]], cost));
30         }
31     }
32     vector<long long> dp[MAXN][MAXN];
33     vector<bool> neg_INF[MAXN][MAXN]; // 如果花費是負的可能會有無限小的情形
34     inline void relax(int l, int r, const CNF &c, long long cost, bool neg_c=0){
35         if(!neg_INF[l][r][c.s] & (neg_INF[l][r][c.x] || cost<dp[l][r][c.s])){
36             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             }
40             else dp[l][r][c.s]=cost;
41         }
42     }

```



```

41 }
42 inline void bellman(int l,int r,int n){
43     for(int k=1;k<=state;++k)
44         for(auto c:cnf)
45             if(c.y==1)relax(l,r,c,dp[l][r][c.x]+c
46                 .cost,k==n);
47 }
48 inline void cyk(const vector<int> &tok){
49     for(int i=0;i<(int)tok.size();++i){
50         for(int j=0;j<(int)tok.size();++j){
51             dp[i][j]=vector<long long>(state+1,
52                 INT_MAX);
53             neg_INF[i][j]=vector<bool>(state+1,
54                 false);
55         }
56         dp[i][i][tok[i]]=0;
57         bellman(i,i,tok.size());
58     }
59     for(int r=1;r<(int)tok.size();++r){
60         for(int l=r-1;l>=0;--l){
61             for(int k=1;k<=r;++k)
62                 for(auto c:cnf)
63                     if(~c.y)relax(l,r,c,dp[l][k][c.x]+
64                         dp[k+1][r][c.y]+c.cost);
65             bellman(l,r,tok.size());
66         }
67     }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

6.2 earley.cpp

```

1 struct Rule{
2     vector<vector<Rule*> > p;
3     void add(const vector<Rule*> &l){
4         p.push_back(l);
5     }
6 };
7 map<string,Rule*> NameRule;
8 map<Rule*,string> RuleName;
9 inline void init_Rule(){
10     for(auto r:RuleName)delete r.first;
11     RuleName.clear();
12     NameRule.clear();
13 }
14 inline Rule *add_rule(const string &s){
15     if(NameRule.find(s)!=NameRule.end())return
16         NameRule[s];
17     Rule *r=new Rule();
18     RuleName[r]=s;
19     NameRule[s]=r;
20     return r;
21 }
22 typedef vector<Rule*> production;
23 struct State{
24     Rule *r;
25     int rid,dot_id,start,end;
26     State(Rule *r,int rid,int dot,int start):r
27         (r),rid(rid),dot_id(dot),start(start),
28         end(-1){}
29     State(Rule *r=0,int col=0):r(r),rid(-1),
30         dot_id(-1),start(-1),end(col){}
31     bool completed()const{
32         return rid==-1||dot_id>=r->p[rid].
33             size();
34     }
35     Rule *next_term()const{
36         if(completed())return 0;
37         return r->p[rid][dot_id];
38     }
39 }
40 bool operator<(const State& b)const{
41     if(start!=b.start)return start<b.start;
42     if(dot_id!=b.dot_id)return dot_id<b.
43         dot_id;
44     if(r!=b.r)return r<b.r;
45     return rid<b.rid;
46 }
47 void print()const{
48     cout<<RuleName[r]<<"-";
49     if(rid!=-1)for(size_t i=0;++i){
50         if((int)i==dot_id)cout<<" "<<"$";
51         if(i>=r->p[rid].size())break;
52         cout<<" "<<RuleName[r->p[rid][i]];
53     }
54     cout<<" "<<"["<<start<<" "<<end<<"]"<<
55         endl;
56 }
57 struct Column{
58     Rule *term;
59     string value;
60     vector<State> s;
61     map<State,set<pair<State,State> > > div;
62     //div比較像一棵 左兄右子的樹
63     Column(Rule *,const string &s):term(r),
64         value(s){}
65     Column(){}
66     bool add(const State &st,int col){
67         if(div.find(st)==div.end()){
68             div[st];
69             s.push_back(st);
70             s.back().end=col;
71             return true;
72         }else return false;
73 }
74 inline vector<Column> lexer(string text){
75     //tokenize · 要自己寫 · 以下為範例
76     //他會把 input stream 變成 token stream ·
77     就是 (terminal,value)pair
78     vector<Column> token;
79     replace(text.begin(),text.end(),',',' ');
80     stringstream ss(text);
81     while(ss>>text){
82         if(text=="a"||text=="of")continue;
83         if(text=="list"){
84             token.push_back(Column(NameRule["(",
85                 ")"]);
86         }else if(text=="and"){
87             token.push_back(Column(NameRule["&"],
88                 ")"]);
89         }else token.push_back(Column(NameRule["T
90             "],text));
91     }
92     return token;
93 }
94 vector<Column> table;
95 inline void predict(int col,Rule *rul){
96     for(size_t i=0;i<rul->p.size();++i){
97         table[col].add(State(rul,i,0,col),col);
98     }
99 }
100 inline void scan(int col,const State &s,Rule
101     *){
102     if(r!=table[col].term)return;
103     State ns(s.r,s.rid,s.dot_id+1,s.start);
104     table[col].add(ns,col);
105     table[col].div[ns].insert(make_pair(s,
106         State(r,col)));
107 }
108 inline void complete(int col,const State &s)
109     {
110         for(size_t i=0;i<table[s.start].s.size()
111             ++i){
112             State &st=table[s.start].s[i];
113             Rule *term=st.next_term();
114             if(!term||term->p.size()==0)continue;
115             if(term==s.r){
116                 State nst(st.r,st.rid,st.dot_id+1,st.
117                     start);
118                 table[col].add(nst,col);
119                 table[col].div[nst].insert(make_pair(
120                     st,s));
121             }
122         }
123 }
124 inline pair<bool,State> parse(Rule *GAMMA,
125     const vector<Column> &token){
126     table.resize(token.size()+1);
127     for(size_t i=0;i<token.size();++i)table[i
128         +1]=Column(token[i]);
129     table[0]=Column();
130     table[0].add(State(GAMMA,0,0,0),0);
131     for(size_t i=0;i<table.size();++i){
132         for(size_t j=0;j<table[i].s.size();++j){
133             State state=table[i].s[j];
134             if(state.completed())complete(i,state)
135                 ;
136             else{
137                 Rule *term=state.next_term();
138                 if(term->p.size())predict(i+1,term);
139                 else if(i+1<table.size())scan(i+1,
140                     state,term);
141             }
142         }
143     }
144     for(size_t i=0;i<table.back().s.size();++i
145         ){
146         if(table.back().s[i].r==GAMMA&&table.
147             back().s[i].completed()){
148             return make_pair(true,table.back().s[i
149                 ]);
150         }
151     }
152     return make_pair(false,State(0,-1));
153 }
154 struct node{//語法樹的節點
155     State s;
156     vector<vector<node*> > child;//vector<node
157         *>.size()>1表示ambiguous
158     node(const State &s):s(s){}
159     node(){}
160 }
161 struct State_end_cmp{
162     bool operator()(const State &a,const State
163         &b)const{
164         return a.end<b.end||(a.end==b.end&&a<b);
165     }
166 };
167 map<State,node*,State_end_cmp> cache;
168 vector<node*> node_set;
169 inline void init_cache(){
170     for(auto d:node_set)delete d;
171     cache.clear();
172     node_set.clear();
173 }
174 void _build_tree(const State &s,node *pa,
175     bool amb=0){
176     if(cache.find(s)!=cache.end()){
177         pa->child.push_back(vector<node*>(1,
178             cache[s]));
179         return;
180     }
181     node *o;
182     if(s.completed()){
183         o=new node(s);
184         if(amb)pa->child.back().push_back(o);
185         else pa->child.push_back(vector<node
186             *>(1,o));
187     }else o=pa->child.back().back();
188     amb=0;
189     for(auto div:table[s.end].div[s]){
190         if(!amb)_build_tree(div.first,pa);
191         _build_tree(div.second,o,amb);
192         amb=1;
193     }
194     if(s.completed())cache[s]=o;
195 }
196 inline node *build_tree(const State &s){
197     init_cache();
198     node o;
199     _build_tree(s,&o);
200     assert(o.child.size()==1);
201     assert(o.child.back().size()==1);
202     return o.child.back().back();
203 }
204 void print_tree(node *o,int dep=0){
205     cout<<string(dep,' '),o->s.print();
206     for(auto div:o->child){
207         for(auto nd:div){
208             print_tree(nd,dep+2);
209         }
210     }
211 }
212 //開始寫code:以下為加入語法的範例
213 inline Rule *get_my_Rule(){
214     Rule *S=add_rule("S"),*E=add_rule("E"),*L=
215         add_rule("L");
216     Rule *list=add_rule("("),*AND=add_rule("&")
217         ,*T=add_rule("T");
218     S->add({list,E});
219     S->add({list,L});
220     L->add({E,L});
221     L->add({E,AND,E});
222     E->add({T});
223     E->add({S});
224     Rule *GAMMA=add_rule("GAMMA");//一定要有
225         gamma rule當作是最上層的語法

```

```

194 GAMMA->add({S});
195 return GAMMA;
196 }

```

7 Linear_Programming

7.1 最大密度子圖.cpp

```

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

```

```

51 T l=0,r=U,k=1.0/(n*n);
52 while(r-l>k){ //二分搜最大值
53     T mid=(l+r)/2;
54     build(mid);
55     T res=(U*n-isap.isap(s,t))/2;
56     if(res>0)l=mid;
57     else r=mid;
58 }
59 build(l);
60 isap.min_cut(s,t);
61 vector<int> ans;
62 for(int i=1;i<=n;++i){
63     if(isap.vis[i])ans.push_back(i);
64 }
65 printf("%d\n",ans.size());
66 for(size_t i=0;i<ans.size();++i){
67     printf("%d\n",ans[i]);
68 }
69 }
70 return 0;
71 }

```

8 Number_Theory

8.1 basic.cpp

```

1 template<typename T>
2 void gcd(const T &a,const T &b,T &d,T &x,T &y){
3     if(!b) d=a,x=1,y=0;
4     else gcd(b,a%b,d,y,x), y-=x*(a/b);
5 }
6 long long int phi[N+1];
7 void phiTable(){
8     for(int i=1;i<=N;i++) phi[i]=i;
9     for(int i=1;i<=N;i++) for(x=i*2;x<=N;x+=i)
10         phi[x]-=phi[i];
11 }
12 void all_divdown(const LL &n){ // all n/x
13     for(LL a=1;a<=n;a=n/(n/(a+1))) {
14         // dosomething;
15     }
16 }
17 const int MAXPRIME = 1000000;
18 int iscom[MAXPRIME], prime[MAXPRIME],
19     primecnt;
20 phi[MAXPRIME], mu[MAXPRIME];
21 void sieve(void){
22     memset(iscom,0,sizeof(iscom));
23     primecnt = 0;
24     phi[1] = mu[1] = 1;
25     for(int i=2;i<MAXPRIME;++i) {
26         if(!iscom[i]) {
27             prime[primecnt++] = i;
28             mu[i] = -1;
29             phi[i] = i-1;
30         }
31         for(int j=0;j<primecnt;++j) {
32             int k = i * prime[j];
33             if(k>=MAXPRIME) break;
34             iscom[k] = prime[j];

```

```

33     if(i%prime[j]==0) {
34         mu[k] = 0;
35         phi[k] = phi[i] * prime[j];
36         break;
37     } else {
38         mu[k] = -mu[i];
39         phi[k] = phi[i] * (prime[j]-1);
40     }
41 }
42 }
43 }
44 }
45 bool g_test(const LL &g, const LL &p, const
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){
73     inv[1]=1;
74     for(int i=2;i<n;++i)
75         inv[i]=(P-(P/i))*inv[P%i]%P;
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 LL Tonelli_Shanks(const LL &n, const LL &p)
95 {
96     // x^2 = n ( mod p )
97     if(n==0) return 0;
98     if(Legendre(n,p)!=1) while(1) { puts("SQRT
99         ROOT does not exist"); }
100     int S = 0;
101     LL Q = p-1;
102     while( !(Q&1) ) { Q>>=1; ++S; }
103     if(S==1) return modexp(n%p,(p+1)/4,p);
104     LL z = 2;
105     for(;; Legendre(z,p)!=-1; ++z)
106         LL c = modexp(z,Q,p);
107         LL R = modexp(n%p,(Q+1)/2,p), t = modexp(n
108             %p,Q,p);
109         int M = S;
110         while(1) {
111             if(t==1) return R;
112             LL b = modexp(c,1L<<(M-i-1),p);
113             R = LLmul(R,b,p);
114             t = LLmul(LLmul(b,b,p), t, p);
115             c = LLmul(b,b,p);
116             M = i;
117         }
118         return -1;
119 }
120 template<typename T>
121 T Euler(T n){
122     T ans=n;
123     for(T i=2;i<=n;++i){
124         if(n%i==0){
125             ans=ans/i*(i-1);
126             while(n%i==0)n/=i;
127         }
128     }
129     if(n>1)ans=ans/n*(n-1);
130     return ans;
131 }
132 //Chinese_remainder_theorem
133 template<typename T>
134 T pow_mod(T n,T k,T m){
135     T ans=1;
136     for(n=(n==m?n%m:n);k;k>>=1){
137         if(k&1)ans=ans*n%m;
138         n=n*n%m;
139     }
140     return ans;
141 }
142 template<typename T>
143 T crt(vector<T> &m,vector<T> &a){
144     T M=1,tM,ans=0;
145     for(int i=0;i<(int)m.size();++i)M*=m[i];
146     for(int i=0;i<(int)a.size();++i){
147         tM=M/m[i];
148         ans=(ans+(a[i]*tM%M)*pow_mod(tM,Euler(m
149             [i])-1,m[i])%M)%M;
150     }
151     return ans;
152 }
153 //java code

```

```

154 //求sqrt(N)的連分數
155 public static void Pell(int n){
156     BigInteger N,p1,p2,q1,q2,a0,a1,a2,g1,g2,h1
157         ,h2,p,q;
158     g1=q2=p1=BigInteger.ZERO;
159     h1=q1=p2=BigInteger.ONE;
160     a0=a1=BigInteger.valueOf((int)Math.sqrt
161         (1.0*n));
162     BigInteger ans=a0.multiply(a0);
163     if(ans.equals(BigInteger.valueOf(n))){
164         System.out.println("No solution!");
165         return ;
166     }
167     while(true){
168         g2=a1.multiply(h1).subtract(g1);
169         h2=N.subtract(g2.pow(2)).divide(h1);
170         a2=g2.add(a0).divide(h2);
171         p=a1.multiply(p2).add(p1);
172         q=a1.multiply(q2).add(q1);
173         if(p.pow(2).subtract(N.multiply(q.pow
174             (2))).compareTo(BigInteger.ONE)==0)
175             break;
176         g1=g2;h1=h2;a1=a2;
177         p1=p2;p2=p;
178         q1=q2;q2=q;
179     }
180     System.out.println(p+" "+q);
181 }

```

8.2 bit_set.cpp

```

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

```

8.3 cantor_expansion.cpp

```

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

```

```

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

```

8.4 FFT.cpp

```

1 template<typename T,typename VT=std::vector<
2     std::complex<T> > >
3 struct FFT{
4     const T pi;
5     FFT(const T pi=acos((T)-1)):pi(pi){}
6     unsigned int bit_reverse(unsigned int a,
7         int len){
8         a=((a&0x55555555U)<<1)|((a&0xAAAAAAAU)
9             >>1);
10        a=((a&0x33333333U)<<2)|((a&0xCCCCCCU)
11            >>2);
12        a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0FU)
13            >>4);
14        a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)
15            >>8);
16        a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)
17            >>16);
18        return a>>(32-len);
19 }
20 void fft(bool is_inv,VT &in,VT &out,int N)
21 {
22     int bitlen=std::__lg(N),num=is_inv?-1:1;
23     for(int i=0;i<N;++i)out[bit_reverse(i,
24         bitlen)]=in[i];
25     for(int step=2;step<=N;step<=<=1){
26         const int mh=step>>1;
27         for(int i=0;i<mh;++i){
28             std::complex<T> wi=exp(std::complex<
29                 T>(0,i*num*pi/mh));
30             for(int j=i;j<N;j+=step){
31                 int k=j+mh;
32                 std::complex<T> u=out[j],t=wi*out[
33                     k];
34                 out[j]=u+t;
35                 out[k]=u-t;
36             }
37         }
38     }
39 }

```

```

28     if(is_inv)for(int i=0;i<N;++i)out[i]/=N; 49| }
29 }
30 };

```

8.5 find_real_root.cpp

```

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

```

8.6 LinearCongruence.cpp

```

1 pair<LL,LL> LinearCongruence(LL a[],LL b[],
2     LL m[],int n) {
3     // a[i]*x = b[i] ( mod m[i] )
4     for(int i=0;i<n;++i) {
5         LL x, y, d = extgcd(a[i],m[i],x,y);
6         if(b[i]%d!=0) return make_pair(-1LL,0LL)
7             ;
8         m[i] /= d;
9         b[i] = LLmul(b[i]/d,x,m[i]);
10    }
11    LL lastb = b[0], lastm = m[0];
12    for(int i=1;i<n;++i) {
13        LL x, y, d = extgcd(m[i],lastm,x,y);
14        if((lastb-b[i])%d!=0) return make_pair
15            (-1LL,0LL);
16        lastb = LLmul((lastb-b[i])/d,x,(lastm/d)
17            )*m[i];
18        lastm = (lastm/d)*m[i];
19        lastb = (lastb+b[i])%lastm;
20    }
21    return make_pair(lastb<0?lastb+lastm:lastb
22        ,lastm);
23 }

```

8.7 Lucas.cpp

```

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

```

8.8 Matrix.cpp

```

1 template<typename T>
2 struct Matrix{
3     using rt = std::vector<T>;
4     using mt = std::vector<rt>;
5     using matrix = Matrix<T>;
6     int r,c;
7     mt m;

```

```

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

```

8.9 MillerRobin.cpp

```

1 LL LLmul(LL a, LL b, const LL &mod) {
2   LL ans=0;
3   while(b) {
4     if(b&1) {
5       ans+=a;
6       if(ans>=mod) ans-=mod;
7     }
8     a<<=1, b>>=1;
9     if(a==mod) a-=mod;
10  }
11  return ans;
12 }
13 long long mod_mul(long long a,long long b,
14   long long m){
15   a%=m,b%=m;
16   long long y=(long long)((double)a*b/m+0.5)
17   ;/* fast for m < 2^58 */
18   long long r=(a*b-y*m)%m;
19   return r<0?r+m:r;
20 }
21 template<typename T>
22 T pow(T a,T b,T mod){/*a^b%mod
23   T ans=1;
24   for(;b;a=mod_mul(a,a,mod),b>>=1)
25     if(b&1)ans=mod_mul(ans,a,mod);
26   return ans;
27 }
28 int sprp[3]={2,7,61};/*int範圍可解
29 int llsprp
30 [7]={2,325,9375,28178,450775,9780504,179526
31   //至少unsigned long long範圍
32 }
33 template<typename T>
34 bool isprime(T n,int *sprp,int num){
35   if(n==2)return 1;
36   if(n<2||n%2==0)return 0;
37   int t=0;
38   T u=n-1;
39   for(;u%2==0;++t)u>>=1;
40   for(int i=0;i<num;++i){
41     T a=sprp[i]%n;
42     if(a==0||a==1||a==n-1)continue;
43     T x=pow(a,u,n);
44     if(x==1||x==n-1)continue;
45     for(int j=0;j<t;++j){
46       x=mod_mul(x,x,n);
47       if(x==1)return 0;
48       if(x==n-1)break;
49     }
50     if(x==n-1)continue;
51     return 0;
52   }
53   return 1;
54 }

```

8.10 NTT.cpp

```

1 2615053605667*(2^18)+1,3
2 15*(2^27)+1,31
3 479*(2^21)+1,3
4 7*17*(2^23)+1,3
5 3*3*211*(2^19)+1,5
6 25*(2^22)+1,3
7 template<typename T,typename VT=std::vector<
8   T>>
9 struct NTT{
10   const T P,G;
11   NTT(T p=(1<<23)*7*17+1,T g=3):P(p),G(g){}
12   unsigned int bit_reverse(unsigned int a,
13     int len){
14     a=((a&0x55555555U)<<1)|((a&0xAAAAAAAU)
15       >>1);
16     a=((a&0x33333333U)<<2)|((a&0xCCCCCCU)
17       >>2);
18     a=((a&0x0F0F0F0FU)<<4)|((a&0xFF0F0F0FU)
19       >>4);
20     a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)
21       >>8);
22     a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)
23       >>16);
24     return a>>(32-len);
25 }
26 T pow_mod(T n,T k,T m){
27   T ans=1;
28   for(n=(n==m%n?n:n);k>>=1){
29     if(k&1)ans=ans*n%m;
30     n=n*n%m;
31   }
32   return ans;
33 }
34 void ntt(bool is_inv,VT &in,VT &out,int N)
35 {
36   int bitlen=std::__lg(N);
37   for(int i=0;i<N;++i)out[bit_reverse(i,
38     bitlen)]=in[i];
39   for(int step=2,id=1;step<=N;step<=1,++
40     id){
41     T wn=pow_mod(G,(P-1)>>id,P),wi=1,u,t;
42     const int mh=step>>1;
43     for(int i=0;i<mh;++i){
44       for(int j=i;j<N;j+=step){
45         u=out[j],t=wi*out[j+mh]%P;
46         out[j]=u+t;
47         out[j+mh]=u-t;
48         if(out[j]>=P)out[j]-=P;
49         if(out[j+mh]<0)out[j+mh]+=P;
50       }
51       wi=wi*wn%P;
52     }
53   }
54   if(is_inv){
55     for(int i=1;i<N/2;++i)std::swap(out[i],
56       out[N-i]);
57     T invn=pow_mod(N,P-2,P);
58     for(int i=0;i<N;++i)out[i]=out[i]*invn
59       %P;
60   }
61 }

```

8.11 Simpson.cpp

```

1 double simpson(double a,double b){
2   double c=a+(b-a)/2;
3   return (F(a)+4*F(c)+F(b))*(b-a)/6;
4 }
5 double asr(double a,double b,double eps,
6   double A){
7   double c=a+(b-a)/2;
8   double L=simpson(a,c),R=simpson(c,b);
9   if( abs(L+R-A)<15*eps )
10     return L+R+(L+R-A)/15.0;
11   return asr(a,c,eps/2,L)+asr(c,b,eps/2,R)
12     ;
13 }
14 double asr(double a,double b,double eps){
15   return asr(a,b,eps,simpson(a,b));
16 }

```

8.12 外星模運算.cpp

```

1 //a[0]^a[1]^a[2]^...
2 #include<bits/stdc++.h>
3 using namespace std;
4 #define maxn 1000000
5 int euler[maxn+5];
6 bool is_prime[maxn+5];
7 inline void init_euler(){
8   is_prime[1]=1;/*不是質數
9   for(int i=1;i<=maxn;i++)euler[i]=i;
10   for(int i=2;i<=maxn;i++){
11     if(!is_prime[i]){/*是質數
12       euler[i]--;
13       for(int j=i<1;j<=maxn;j+=i){
14         is_prime[j]=1;
15         euler[j]=euler[j]/i*(i-1);
16       }
17     }
18   }
19 }
20 inline long long pow(long long a,long long b
21   ,long long mod){/*a^b%mod
22   long long ans=1;
23   for(;b;a=a*a%mod,b>>=1)
24     if(b&1)ans=ans*a%mod;
25   return ans;
26 }
27 bool isless(long long *a,int n,int k){
28   if(*a==1)return k>1;
29   if(--n==0)return *a<k;
30   int next=0;
31   for(long long b=1;b<k;++next)
32     b*=*a;
33   return isless(a+1,n,next);
34 }
35 long long high_pow(long long *a,int n,long
36   long mod){
37   if(*a==1||--n==0)return *a%mod;
38   int k=0,r=euler[mod];
39   for(long long tma=1;tma!=pow(*a,k+r,mod)
40     ;++k)
41     tma=tma*(a)%mod;

```



```

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

```

8.13 模運算模板.cpp

```

1 template<typename T,long long mod>
2 struct mod_t{//mod只能是質數
3     T data;
4     mod_t(){
5         mod_t(const T &d):data((d%mod+mod)%mod){
6             mod_t pow(T b)const{
7                 mod_t ans(1);
8                 for(mod_t now=*this;b;now=now*b,b/=2)
9                     if(b%2)ans=ans*now;
10                return ans;
11            }
12            mod_t operator-(int)const{
13                return mod_t(mod-data);
14            }
15            mod_t operator+(const mod_t &b)const{
16                return mod_t((data+b.data)%mod);
17            }
18            mod_t operator-(const mod_t &b)const{
19                return mod_t((data-b.data+mod)%mod);
20            }
21            mod_t operator*(const mod_t &b)const{
22                return mod_t((data*b.data)%mod);
23            }
24            mod_t operator/(const mod_t &b)const{
25                return *this*b.pow(mod-2);/*this *
                Inverse(b)
26            }
27            operator T()const{return data;}
28            friend istream &operator>>(istream &i,
                mod_t &b){
29                T d;
30                i>>d;
31                b=mod_t(d);
32                return i;
33            }
34 };

```

8.14 質因數分解.cpp

```

1 LL func(const LL n,const LL mod,const int c)
2 {
3     return (LLmul(n,n,mod)+c+mod)%mod;
4 }
5 LL pollrho(const LL n, const int c) { //循環
6     環節長度
7     LL a=1, b=1;
8     a=func(a,n,c)%n;
9     b=func(b,n,c)%n; b=func(b,n,c)%n;
10    while(gcd(abs(a-b),n)!=1) {
11        a=func(a,n,c)%n;
12        b=func(b,n,c)%n; b=func(b,n,c)%n;
13    }
14    return gcd(abs(a-b),n);
15 }
16 void prefactor(LL &n, vector<LL> &v) {
17     for(int i=0;i<12;++i) {
18         while(n%prime[i]==0) {
19             v.push_back(prime[i]);
20             n/=prime[i];
21         }
22     }
23 }
24 void smallfactor(LL n, vector<LL> &v) {
25     if(n<MAXPRIME) {
26         while(isp[(int)n]) {
27             v.push_back(isp[(int)n]);
28             n/=isp[(int)n];
29         }
30         v.push_back(n);
31     } else {
32         for(int i=0;i<primecnt&&prime[i]*prime[i]
33             ]<=n;++i) {
34             while(n%prime[i]==0) {
35                 v.push_back(prime[i]);
36                 n/=prime[i];
37             }
38         }
39         if(n!=1) v.push_back(n);
40     }
41 }
42 void comfactor(const LL &n, vector<LL> &v) {
43     if(n<1e9) {
44         smallfactor(n,v);
45         return;
46     }
47     if(Isprime(n)) {
48         v.push_back(n);
49         return;
50     }
51     LL d;
52     for(int c=3;++c) {
53         d = pollrho(n,c);
54         if(d!=n) break;
55     }
56     comfactor(d,v);
57     comfactor(n/d,v);
58 }
59 void Factor(const LL &x, vector<LL> &v) {
60     LL n = x;
61 }

```

```

63 if(n==1) { puts("Factor 1"); return; }
64 prefactor(n,v);
65 if(n==1) return;
66 comfactor(n,v);
67 sort(v.begin(),v.end());
68 }
69 void AllFactor(const LL &n,vector<LL> &v) {
70     vector<LL> tmp;
71     Factor(n,tmp);
72     v.clear();
73     v.push_back(1);
74     int len;
75     LL now=1;
76     for(int i=0;i<tmp.size();++i) {
77         if(i==0 || tmp[i]!=tmp[i-1]) {
78             len = v.size();
79             now = 1;
80         }
81         now*=tmp[i];
82         for(int j=0;j<len;++j)
83             v.push_back(v[j]*now);
84     }
85 }
86 }

```

9 other

9.1 WhatDay.cpp

```

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

```

9.2 上下最大正方形.cpp

```

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

```

```

18 }
19 }
20 ans=max(ans,r-l+1);
21 }
22 printf("%d\n",ans);
23 }

```

9.3 最大矩形.cpp

```

1 long long max_rectangle(vector<int> s){
2     stack<pair<int,int> > st;
3     st.push(make_pair(-1,0));
4     s.push_back(0);
5     long long 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,(long long)(i-now.second)*
13                now.first);
14        }
15        if(h>st.top().first){
16            st.push(make_pair(h,now.second));
17        }
18    }
19    return ans;
20 }

```

10 String

10.1 AC 自動機.cpp

```

1 template<char L='a',char R='z'>
2 class ac_automaton{
3 private:
4     struct joe{
5         int next[R-L+1],fail,efl,ed,cnt_dp,vis;
6     };
7     joe():ed(0),cnt_dp(0),vis(0){
8         for(int i=0;i<R-L;++i)next[i]=0;
9     };
10 public:
11     std::vector<joe> S;
12     std::vector<int> q;
13     int qs,qe,vt;
14     ac_automaton():S(1),qs(0),qe(0),vt(0){}
15     void clear(){
16         q.clear();
17         S.resize(1);
18         for(int i=0;i<R-L;++i)S[0].next[i]=0;
19         S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
20     }
21     void insert(const char *s){
22         int o=0;
23         for(int i=0,id;s[i];++i){
24             int c=s[i]-L;
25             if(!S[o].next[c]){
26                 S.push_back(joe());
27                 S.back().fail=S[o].efl;
28                 S.back().ed=0;
29                 S.back().cnt_dp=0;
30                 S.back().vis=0;
31                 S[o].next[c]=S.size()-1;
32             }
33             o=S[o].next[c];
34             if(i==s.length()-1)S[o].ed=1;
35         }
36     }
37     void get_fail(){
38         queue<int> q;
39         q.push(0);
40         while(!q.empty()){
41             int u=q.front();q.pop();
42             for(int i=0;i<R-L;++i){
43                 int v=S[u].next[i];
44                 if(v){
45                     S[v].fail=S[S[u].fail].next[i];
46                     q.push(v);
47                 }
48             }
49         }
50     }
51     int query(const char *s){
52         int o=0;
53         for(int i=0;s[i];++i){
54             int c=s[i]-L;
55             if(!S[o].next[c])return 0;
56             o=S[o].next[c];
57         }
58         return S[o].ed;
59     }
60 }

```

```

24     id=s[i]-L;
25     if(!S[o].next[id]){
26         S.push_back(joe());
27         S[o].next[id]=S.size()-1;
28     }
29     o=S[o].next[id];
30 }
31 ++S[o].ed;
32 }
33 void build_fail(){
34     S[0].fail=S[0].efl=-1;
35     q.clear();
36     q.push_back(0);
37     ++qe;
38     while(qs!=qe){
39         int pa=q[qs++],id,t;
40         for(int i=0;i<=R-L;++i){
41             t=S[pa].next[i];
42             if(!t)continue;
43             id=S[pa].fail;
44             while(~id&&S[id].next[i])id=S[id].fail;
45             S[t].fail=~id?S[id].next[i]:0;
46             S[t].efl=S[t].fail].ed?S[t].fail:
47             :S[t].fail].efl;
48             q.push_back(t);
49             ++qe;
50         }
51     }
52     /*DP出每個前綴在字串s出現的次數並傳回所
53     有字串被s匹配成功的次數O(N*M)*/
54     int match_0(const char *s){
55         int ans=0,id,p=0,t;
56         for(i=0;s[i];++i){
57             id=s[i]-L;
58             while(!S[p].next[id]&&p=S[p].fail;
59             if(!S[p].next[id])continue;
60             p=S[p].next[id];
61             ++S[p].cnt_dp; /*匹配成功則它所有後綴
62             都可以被匹配(DP計算)*/
63         }
64         for(i=qe-1;i>=0;--i){
65             ans+=S[q[i]].cnt_dp*S[q[i]].ed;
66             if(~S[q[i]].fail)S[q[i]].fail].
67             cnt_dp+=S[q[i]].cnt_dp;
68         }
69         return ans;
70     }
71     /*多串匹配走efl邊並傳回所有字串被s匹配成
72     功的次數O(N*M^1.5)*/
73     int match_1(const char *s) const{
74         int ans=0,id,p=0,t;
75         for(int i=0;s[i];++i){
76             id=s[i]-L;
77             while(!S[p].next[id]&&p=S[p].fail;
78             if(!S[p].next[id])continue;
79             p=S[p].next[id];
80             if(S[p].ed)ans+=S[p].ed;
81             for(t=S[p].efl;~t;t=S[t].efl){
82                 ans+=S[t].ed; /*因為都走efl邊所以保
83                 證匹配成功*/
84             }
85         }
86     }

```

```

81     return ans;
82 }
83 /*枚舉(s的字子串nA)的所有相異字串各恰一
84 次並傳回次數O(N*M^(1/3))*/
85 int match_2(const char *s){
86     int ans=0,id,p=0,t;
87     ++vt;
88     /*把戳記vt+=1，只要vt沒溢位，所有S[p].
89     vis==vt就會變成false
90     這種利用vt的方法可以O(1)歸零vis陣列*/
91     for(int i=0;s[i];++i){
92         id=s[i]-L;
93         while(!S[p].next[id]&&p=S[p].fail;
94         if(!S[p].next[id])continue;
95         p=S[p].next[id];
96         if(S[p].ed&&S[p].vis!=vt){
97             S[p].vis=vt;
98             ans+=S[p].ed;
99         }
100         for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t].efl){
101             S[t].vis=vt;
102             ans+=S[t].ed; /*因為都走efl邊所以保
103             證匹配成功*/
104         }
105     }
106     return ans;
107 }
108 /*把AC自動機變成真的自動機*/
109 void evolution(){
110     for(qs=1;qs!=qe;){
111         int p=q[qs++];
112         for(int i=0;i<=R-L;++i)
113             if(S[p].next[i]==0)S[p].next[i]=S[
114             S[p].fail].next[i];
115     }
116 }

```

10.2 hash.cpp

```

1 #define MAXN 1000000
2 #define prime_mod 1073676287
3 /*prime_mod 必須要質數*/
4 typedef long long T;
5 char s[MAXN+5];
6 T h[MAXN+5]; /*hash陣列*/
7 T h_base[MAXN+5]; /*h_base[n]=(prime^n)%
8 prime_mod*/
9 inline void hash_init(int len,T prime=0
10 xdefaced){
11     h_base[0]=1;
12     for(int i=1;i<=len;++i){
13         h[i]=(h[i-1]*prime+s[i-1])%prime_mod;
14         h_base[i]=(h_base[i-1]*prime)%prime_mod;
15     }
16 inline T get_hash(int l,int r){ /*閉區間寫
17 法，設編號為0 ~ Len-1*/
18     return (h[r+1]-(h[l]*h_base[r-l+1])%
19     prime_mod+prime_mod)%prime_mod;

```

```

17 }

```

10.3 KMP.cpp

```

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

```

10.4 manacher.cpp

```

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

```

10.5 minimal_string_rotation.cpp

```

1 int min_string_rotation(const string &s){
2     int n=s.size(),i=0,j=1,k=0;
3     while(i<n&&j<n&&k<n){
4         int t=s[(i+k)%n]-s[(j+k)%n];
5         ++k;
6         if(t){
7             if(t>0)i+=k;
8             else j+=k;
9             if(i==j)++j;

```

```

10     k=0;
11     }
12 }
13 return min(i,j); //傳回最小循環表示法起始位
14 置

```

10.6 suffix_array_lcp.cpp

```

1 #define radix_sort(x,y){\
2     for(i=0;i<A;++i)c[i]=0;\
3     for(i=0;i<len;++i)c[x[y[i]]]++;\
4     for(i=1;i<A;++i)c[i]+=c[i-1];\
5     for(i=len-1;i>=0;--i)sa[--c[x[y[i]]]]=y[i
6     ];\
7 }
8 void suffix_array(const char *s,int len,int
9 *sa,int *rank,int *tmp,int *c){
10     int A='z'+1,i,k,id,*t;
11     for(i=0;i<len;++i){
12         tmp[i]=i;
13         rank[i]=s[i];
14     }
15     radix_sort(rank,tmp);
16     for(k=1;id<len-1;k<=1){
17         id=0;
18         for(i=len-k;i<len;++i)tmp[id++]=i;
19         for(i=0;i<len;++i){
20             if(sa[i]>=k)tmp[id++]=sa[i]-k;
21         }
22         radix_sort(rank,tmp);
23         t=rank;rank=tmp;tmp=t;
24         id=0;
25         rank[sa[0]]=0;
26         for(i=1;i<len;++i){
27             if(tmp[sa[i-1]]!=tmp[sa[i]]||sa[i-1]+k
28             >=len||tmp[sa[i-1]+k]!=tmp[sa[i]+k]
29             )++id;
30             rank[sa[i]]=id;
31         }
32     }
33 #undef radix_sort
34 //h:高度數組 sa:後綴數組 rank:排名
35 inline void suffix_array_lcp(const char *s,
36 int len,int *h,int *sa,int *rank){
37     for(int i=0;i<len;++i)rank[sa[i]]=i;
38     for(int i=0,k=0;i<len;++i){
39         if(rank[i]==0)continue;
40         if(k--<0);
41         while(s[i+k]==s[sa[rank[i]-1]+k])++k;
42         h[rank[i]]=k;
43     }
44     h[0]=0;

```

10.7 Z.cpp

```

1 inline void z_alg(char *s,int len,int *z){
2     int l=0,r=0;
3     z[0]=len;
4     for(int i=1;i<len;++i){
5         z[i]=i>r?0:(i-1+z[i-1]<z[l]?z[i-1]:r-i
6             +1);
7         while(i+z[i]<len&&s[i+z[i]]==s[z[i]])++z
8             [i];
9         if(i+z[i]-1>r)r=i+z[i]-1,l=i;
10    }
11 }

```

11 Tarjan

11.1 dominator_tree.cpp

```

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

```

```

43         semi[y]=min(semi[y],semi[best[z]]);
44     }
45     dom[semi[y]].push_back(y);
46     anc[y]=x;
47     for(auto z:dom[x]){
48         find(z);
49         idom[z]=semi[best[z]<x?best[z]:x];
50     }
51     dom[x].clear();
52 }
53 for(int u=2;u<=Time;++u){
54     if(idom[u]!=semi[u])idom[u]=idom[idom[
55         u]];
56     dom[id[idom[u]]].push_back(id[u]);
57 }
58 }dom;

```

11.2 tnfsb017_2_sat.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define MAXN 8001
4 #define MAXN2 MAXN*4
5 #define n(X) ((X)+2*N)
6 vector<int> v[MAXN2];
7 vector<int> rv[MAXN2];
8 vector<int> vis_t;
9 int N,M;
10 void addedge(int s,int e){
11     v[s].push_back(e);
12     rv[e].push_back(s);
13 }
14 int scc[MAXN2];
15 bool vis[MAXN2]={false};
16 void dfs(vector<int> *uv,int n,int k=-1){
17     vis[n]=true;
18     for(int i=0;i<uv[n].size();++i)
19         if(!vis[uv[n][i]])
20             dfs(uv,uv[n][i],k);
21     if(uv==v)vis_t.push_back(n);
22     scc[n]=k;
23 }
24 void solve(){
25     for(int i=1;i<=N;++i){
26         if(!vis[i])dfs(v,i);
27         if(!vis[n(i)])dfs(v,n(i));
28     }
29     memset(vis,0,sizeof(vis));
30     int c=0;
31     for(int i=vis_t.size()-1;i>=0;--i)
32         if(!vis[vis_t[i]])
33             dfs(rv,vis_t[i],c++);
34 }
35 int main(){
36     int a,b;
37     scanf("%d%d",&N,&M);
38     for(int i=1;i<=N;++i){
39         // (A or B)&(A & !B) A^B
40         a=i*2-1;
41         b=i*2;
42         addedge(n(a),b);
43         addedge(n(b),a);

```

```

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

```

11.3 橋連通分量.cpp

```

1 #define N 1005
2 struct edge{
3     int u,v;
4     bool is_bridge;
5     edge(int u=0,int v=0):u(u),v(v),is_bridge
6         (0){}
7 };
8 vector<edge> E;
9 vector<int> G[N];// 1-base
10 int low[N],vis[N],Time;
11 int bcc_id[N],bridge_cnt,bcc_cnt;// 1-base
12 int st[N],top;//BCC用
13 inline void add_edge(int u,int v){
14     G[u].push_back(E.size());
15     E.push_back(edge(u,v));
16     G[v].push_back(E.size());
17     E.push_back(edge(v,u));
18 }
19 void dfs(int u,int re=-1){//u當前點·re為u連
20     接前一個點的邊
21     int v;
22     low[u]=vis[u]=++Time;
23     st[top++]=u;
24     for(size_t i=0;i<G[u].size();++i){
25         int e=G[u][i];v=E[e].v;
26         if(!vis[v]){
27             dfs(v,e^1);//e^1反向邊
28             low[u]=min(low[u],low[v]);
29             if(vis[u]<low[v]){
30                 E[e].is_bridge=E[e^1].is_bridge=1;
31                 ++bridge_cnt;

```

```

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

```

11.4 雙連通分量 & 割點.cpp

```

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

```

12 Tree_problem

12.1 HeavyLight.cpp

```

1 #include<vector>
2 #define MAXN 100005
3 typedef std::vector<int >::iterator VIT;
4 int siz[MAXN], max_son[MAXN], pa[MAXN], dep[
    MAXN];
5 int link_top[MAXN], link[MAXN], cnt;
6 std::vector<int >G[MAXN];
7 void find_max_son(int x){
8     siz[x]=1;
9     max_son[x]=-1;
10    for(VIT i=G[x].begin(); i!=G[x].end(); ++i){
11        if(*i==pa[x]) continue;
12        pa[*i]=x;
13        dep[*i]=dep[x]+1;
14        find_max_son(*i);
15        if(max_son[x]==-1 || siz[*i]>siz[max_son[x]
            ]) max_son[x]=*i;
16        siz[x]+=siz[*i];
17    }
18 }
19 void build_link(int x, int top){
20     link[x]=++cnt;
21     link_top[x]=top;
22     if(max_son[x]==-1) return;
23     build_link(max_son[x], top);
24     for(VIT i=G[x].begin(); i!=G[x].end(); ++i){
25         if(*i==max_son[x] || *i==pa[x]) continue;
26         build_link(*i, *i);
27     }
28 }
29 inline int find_lca(int a, int b){
30     //求LCA · 可以在過程中對區間進行處理
31     int ta=link_top[a], tb=link_top[b];
32     while(ta!=tb){
33         if(dep[ta]<dep[tb]){
34             std::swap(ta, tb);
35             std::swap(a, b);
36         }
37         //這裡可以對a所在的鏈做區間處理
38         //區間為(Link[ta], Link[a])
39         ta=link_top[a=pa[ta]];
40     }
41     //最後a,b會在同一條鏈 · 若a!=b還要在進行一
    次區間處理
42     return dep[a]<dep[b]?a:b;
43 }

```

12.2 LCA.cpp

```

1 #define MAXN 100000
2 #define MAX_LOG 17
3 int pa[MAX_LOG+1][MAXN+5];
4 int dep[MAXN+5];
5 vector<int>G[MAXN+5];
6 void dfs(int x, int p){ //dfs(1, -1);

```

```

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

```

12.3 link_cut_tree.cpp

```

1 #include<vector>
2 struct splay_tree{
3     int ch[2], pa; //子節點跟父母
4     bool rev; //反轉的懶惰標記
5     splay_tree(): pa(0), rev(0){ ch[0]=ch[1]=0; }
6 };
7 vector<splay_tree> node;
8 //有的時候用vector會TLE · 要注意
9 //這邊以node[0]作為null節點
10 bool isroot(int x){ //判斷是否為這棵splay
    tree的根
11     return node[node[x].pa].ch[0]!=x && node[
        node[x].pa].ch[1]!=x;
12 }
13 void down(int x){ //懶惰標記下推
14     if(node[x].rev){
15         if(node[x].ch[0]) node[node[x].ch[0]].rev
            ^=1;
16         if(node[x].ch[1]) node[node[x].ch[1]].rev
            ^=1;
17         std::swap(node[x].ch[0], node[x].ch[1]);
18         node[x].rev ^=1;
19     }
20 }
21 void push_down(int x){ //將所有祖先的懶惰標記
    下推
22     if(!isroot(x)) push_down(node[x].pa);
23     down(x);
24 }
25 void up(int x){ //將子節點的資訊向上更新
26 void rotate(int x){ //旋轉 · 會自行判斷轉的方
    向

```

```

27     int y=node[x].pa, z=node[y].pa, d=(node[y].
        ch[1]==x);
28     node[x].pa=z;
29     if(!isroot(y)) node[z].ch[node[z].ch[1]==y
        ]=x;
30     node[y].ch[d]=node[x].ch[d^1];
31     node[node[y].ch[d]].pa=y;
32     node[y].pa=x, node[x].ch[d^1]=y;
33     up(y);
34     up(x);
35 }
36 void splay(int x){ //將節點x伸展到所在splay
    tree的根
37     push_down(x);
38     while(!isroot(x)){
39         int y=node[x].pa;
40         if(!isroot(y)){
41             int z=node[y].pa;
42             if((node[z].ch[0]==y)^(node[y].ch[0]==
                x)) rotate(y);
43             else rotate(x);
44         }
45         rotate(x);
46     }
47 }
48 int access(int x){
49     int last=0;
50     while(x){
51         splay(x);
52         node[x].ch[1]=last;
53         up(x);
54         last=x;
55         x=node[x].pa;
56     }
57     return last; //回傳access後splay tree的根
58 }
59 void access(int x, bool is=0){ //is=0就是一般
    的access
60     int last=0;
61     while(x){
62         splay(x);
63         if(is && !node[x].pa){
64             //printf("%d\n", max(node[last].ma, node
                [node[x].ch[1]].ma));
65         }
66         node[x].ch[1]=last;
67         up(x);
68         last=x;
69         x=node[x].pa;
70     }
71 }
72 void query_edge(int u, int v){
73     access(u);
74     access(v, 1);
75 }
76 void make_root(int x){
77     access(x), splay(x);
78     node[x].rev ^=1;
79 }
80 void make_root(int x){
81     node[access(x)].rev ^=1;
82     splay(x);
83 }
84 void cut(int x, int y){
85     make_root(x);

```

```

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

```



```

145     node[v].data=e[G[u][i].second].w;
146     edge_node[G[u][i].second]=v;
147     up(v);
148     q.push(v);
149 }
150 }
151 }
152 }
153 void change(int x,int b){
154     splay(x);
155     //node[x].data=b;
156     up(x);
157 }

```

```

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

```

12.4 POJ_tree.cpp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define MAXN 10005
4 int n,k;
5 vector<pair<int,int> >g[MAXN];
6 int size[MAXN];
7 bool vis[MAXN];
8 inline void init(){
9     for(int i=0;i<=n;++i){
10         g[i].clear();
11         vis[i]=0;
12     }
13 }
14 void get_dis(vector<int> &dis,int u,int pa,
15     int d){
16     dis.push_back(d);
17     for(size_t i=0;i<g[u].size();++i){
18         int v=g[u][i].first,w=g[u][i].second;
19         if(v!=pa&&vis[v])get_dis(dis,v,u,d+w);
20     }
21 vector<int> dis;//這東西如果放在函數裡會TLE
22 int cal(int u,int d){
23     dis.clear();
24     get_dis(dis,u,-1,d);
25     sort(dis.begin(),dis.end());
26     int l=0,r=dis.size()-1,res=0;
27     while(l<r){
28         while(l<r&&dis[l]+dis[r]>k)--r;
29         res+=r-(l++);
30     }
31     return res;
32 }
33 pair<int,int> tree_centroid(int u,int pa,
34     const int sz){
35     size[u]=1;//找樹重心，second是重心
36     pair<int,int> res(INT_MAX,-1);
37     int ma=0;
38     for(size_t i=0;i<g[u].size();++i){
39         int v=g[u][i].first;
40         if(v==pa||vis[v])continue;
41         res=min(res,tree_centroid(v,u,sz));
42         size[u]+=size[v];
43         ma=max(ma,size[v]);
44     }
45     ma=max(ma,sz-size[u]);

```

```

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

```

13 zformula

13.1 formula.tex

13.1.1 Pick 公式

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

13.1.2 圖論

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

- $I(G) + C_v(G) = |V|$
- $M(G) + C_e(G) = |V|$

- 對於連通二分圖：

- $I(G) = C_v(G)$
- $M(G) = C_e(G)$

- 最大權閉合圖：

- $C(u, V) = \infty, (u, v) \in E$
- $C(S, v) = W_v, W_v > 0$
- $C(v, T) = -W_v, W_v < 0$

- 最大密度子圖：

- $C(u, v) = 1, (u, v) \in E$
- $C(S, v) = U_v, v \in V$
- $C(v, T) = U + 2g - d_v, v \in V$

- 弦圖：

- 完美消除序列從後往前依次給每個點染色，給每個點染上可以染的最小顏色
- 最大團大小 = 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数据區間圖染色：用線段樹做

```

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

```

13.1.3 學長公式

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

13.1.4 基本數論

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

13.1.5 排組公式

- 卡卡特蘭 $\frac{C_n^{kn}}{n(k-1)+1} \cdot C_m^n = \frac{n!}{m!(n-m)!}$
- $H(n, m) \cong x_1 + x_2 + \dots + x_n = k, num = C_n^{n+k-1}$
- Stirling number of 2^{nd} , n 人分 k 組方法數目

- $S(0, 0) = S(n, n) = 1$
- $S(n, 0) = 0$

- $S(n, k) = kS(n-1, k) + S(n-1, k-1)$

- Bell number, n 人分任意多組方法數目

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

- Derangement, 錯排，沒有人在自己位置上

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

13.1.6 冪次，冪次和

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

13.1.7 Burnside's lemma

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

13.1.8 Count on a tree

- Rooted tree: $s_{n+1} = \frac{1}{n} \sum_{i=1}^n (i \times a_i \times \sum_{j=1}^{\lfloor n/i \rfloor} a_{n+1-i \times j})$
- Unrooted tree:

- Odd: $a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$
- Even: $Odd + \frac{1}{2} a_{n/2} (a_{n/2} + 1)$

3. Spanning Tree

(a) 完全圖 $n^n - 2$ (b) 一般圖 (Kirchhoff's theorem) $M[i][i] = \text{degree}(V_i), M[i][j] = -1, \text{if have } E(i, j), 0$
if no edge. delete any one row and col in A , $ans = \det(A)$

```

22 | pw.close();//关闭流并释放 · 这个很重要 ·
    | 否则是没有输出的
23 | sc.close();//关闭流并释放
    | }
24 | }
25 | }

```

13.2.2 优先队列

13.1.9 積分表

1. $\int \frac{1}{x} dx = \ln|x|$
2. $\int u dv = uv - \int v du$
3. $\int a^x dx = \frac{1}{\ln a} a^x$
4. $\int \ln x dx = x \ln x - x$
5. $\int \tan x dx = \ln|\sec x|$
6. $\int \sec x dx = \ln|\sec x + \tan x|$
7. $\int \sec^2 x dx = \tan x$
8. $\int \sec \tan x dx = \sec x$
9. $\int \frac{a}{a^2+x^2} dx = \tan^{-1} \frac{x}{a}$
10. $\int \frac{a}{a^2-x^2} dx = \frac{1}{2} \ln \left| \frac{x+a}{x-a} \right|$
11. $\int \frac{1}{\sqrt{a^2-x^2}} dx = \sin^{-1} \frac{x}{a}$
12. $\int \frac{a}{x\sqrt{x^2-a^2}} dx = \sec^{-1} \frac{x}{a}$
13. $\int \frac{1}{\sqrt{x^2-a^2}} dx = \cosh^{-1} \frac{x}{a} = \ln(x + \sqrt{x^2-a^2})$
14. $\int \frac{1}{\sqrt{x^2+a^2}} dx = \sinh^{-1} \frac{x}{a} = \ln(x + \sqrt{x^2+a^2})$

```

1 | PriorityQueue queue = new PriorityQueue( 1,
    | new Comparator()
2 | {
3 |     public int compare( Point a, Point b )
4 |     {
5 |         if( a.x < b.x || a.x == b.x && a.y < b.y )
6 |             return -1;
7 |         else if( a.x == b.x && a.y == b.y )
8 |             return 0;
9 |         else
10 |             return 1;
11 |     }
12 | });

```

13.2.3 Map

```

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

```

13.2 java.tex

13.2.1 文件操作

```

1 | import java.io.*;
2 | import java.util.*;
3 | import java.math.*;
4 | import java.text.*;
5 |
6 | public class Main
7 | {
8 |
9 |     public static void main(String args[])
10 |     {
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 |         {
21 |             pw.println("Case #"+ci+": easy for

```

13.2.4 sort

```

1 | static class cmp implements Comparator
2 | {
3 |     public int compare(Object o1,Object o2)
4 |     {
5 |         BigInteger b1=(BigInteger)o1;
6 |         BigInteger b2=(BigInteger)o2;
7 |         return b1.compareTo(b2);
8 |     }
9 | }
10 | public static void main(String[] args)
11 | {
12 |     Scanner cin = new Scanner(System.in);
13 |     int n;
14 |     n=cin.nextInt();
15 |     BigInteger[] seg = new BigInteger[n];
16 |     for (int i=0;i<n;i++)
17 |         seg[i]=cin.nextBigInteger();
18 |     Arrays.sort(seg,new cmp());
19 | }

```

ACM ICPC TEAM REFERENCE - NTHU JINKELA

Contents

1 Computational_Geometry	1	2.9 操作分治.cpp	7	6 language	12	10.5 minimal_string_rotation.cpp	18
1.1 Geometry.cpp	1	2.10 整體二分.cpp	7	6.1 CNF.cpp	12	10.6 suffix_array_lcp.cpp	18
1.2 SmallestCircle.cpp	3	3 default	7	6.2 earley.cpp	13	10.7 Z.cpp	18
1.3 最近點對.cpp	3	3.1 debug.cpp	7	7 Linear_Programming	14	11 Tarjan	19
1.4 浮點數誤差模板.cpp	3	3.2 ext.cpp	7	7.1 最大密度子圖.cpp	14	11.1 dominator_tree.cpp	19
2 Data_Structure	3	3.3 IncStack.cpp	7	8 Number_Theory	14	11.2 tnfsb017_2_sat.cpp	19
2.1 DLX.cpp	3	3.4 input.cpp	7	8.1 basic.cpp	14	11.3 橋連通分量.cpp	19
2.2 Dynamic_KD_tree.cpp	4	4 Flow	7	8.2 bit_set.cpp	15	11.4 雙連通分量 & 割點.cpp	19
2.3 kd_tree_replace_segment_tree.cpp	5	4.1 dinic.cpp	7	8.3 cantor_expansion.cpp	15	12 Tree_problem	20
2.4 persistent_segment_tree.cpp	5	4.2 ISAP_with_cut.cpp	7	8.4 FFT.cpp	15	12.1 HeavyLight.cpp	20
2.5 reference_point.cpp	6	4.3 MinCostMaxFlow.cpp	8	8.5 find_real_root.cpp	15	12.2 LCA.cpp	20
2.6 skew_heap.cpp	6	5 Graph	8	8.6 LinearCongruence.cpp	15	12.3 link_cut_tree.cpp	20
2.7 split_merge.cpp	6	5.1 Augmenting_Path.cpp	8	8.7 Lucas.cpp	15	12.4 POJ_tree.cpp	21
2.8 treap.cpp	6	5.2 Augmenting_Path_multiple.cpp	8	8.8 Matrix.cpp	15	13 zformula	21
		5.3 blossom_matching.cpp	8	8.9 MillerRobin.cpp	16	13.1 formula.tex	21
		5.4 graphISO.cpp	9	8.10 NTT.cpp	16	13.1.1 Pick 公式	21
		5.5 KM.cpp	9	8.11 Simpson.cpp	16	13.1.2 圖論	21
		5.6 MaximumClique.cpp	9	8.12 外星模運算.cpp	16	13.1.3 學長公式	21
		5.7 MinimumMeanCycle.cpp	9	8.13 模運算模板.cpp	17	13.1.4 基本數論	21
		5.8 Minimum_General_Weighted_Matching.cpp	9	8.14 質因數分解.cpp	17	13.1.5 排組公式	21
		5.9 Rectilinear_Steiner_tree.cpp	10	9 other	17	13.1.6 冪次, 冪次和	21
		5.10 treeISO.cpp	10	9.1 WhatDay.cpp	17	13.1.7 Burnside's lemma	21
		5.11 一般圖最大權匹配.cpp	10	9.2 上下最大正方形.cpp	17	13.1.8 Count on a tree	21
		5.12 全局最小割.cpp	11	9.3 最大矩形.cpp	17	13.1.9 積分表	22
		5.13 平面圖判定.cpp	11	10 String	17	13.2 java.tex	22
		5.14 弦圖完美消除序列.cpp	11	10.1 AC 自動機.cpp	17	13.2.1 文件操作	22
		5.15 最小斯坦那樹 DP.cpp	12	10.2 hash.cpp	18	13.2.2 優先队列	22
		5.16 最小樹形圖 _ 朱劉.cpp	12	10.3 KMP.cpp	18	13.2.3 Map	22
		5.17 穩定婚姻模板.cpp	12	10.4 manacher.cpp	18	13.2.4 sort	22