

# 1 Computational Geometry

## 1.1 最近點對

```

1 template<typename _IT=point<T>* >
2 T closest_pair(_IT L, _IT R){
3     if(R-L <= 1) return INF;
4     _IT mid = L+(R-L)/2;
5     T x = mid->x;
6     T d = min(closest_pair(L,mid),closest_pair(
7         mid,R));
8     inplace_merge(L, mid, R, ycmp);
9     static vector<point> b; b.clear();
10    for(auto u=L;u<R;++u){
11        if((u->x-x)*(u->x-x)>=d) continue;
12        for(auto v=b.rbegin();v!=b.rend();++v){
13            T dx=u->x-v->x, dy=u->y-v->y;
14            if(dy*dy>=d) break;
15            d=min(d,dx*dx+dy*dy);
16        }
17        b.push_back(*u);
18    }
19    return d;
20 }
21 T closest_pair(vector<point<T>> &v){
22     sort(v.begin(),v.end(),xcmp);
23     return closest_pair(v.begin(),v.end());
24 }

```

## 1.2 MinCircleCover

```

1 const double eps = 1e-10;
2 int sign(double a){
3     return fabs(a)<eps?0:a>0?-1:1;
4 }
5 template<typename T>
6 T len(point<T> p){
7     return sqrt(p.dot(p));
8 }
9 template<typename T>
10 point<T> findCircumcenter(point<T> A,point<T>
11     > B,point<T> C){
12     point<T> AB = B-A;
13     point<T> AC = C-A;
14     T AB_len_sq = AB.x*AB.x+AB.y*AB.y;
15     T AC_len_sq = AC.x*AC.x+AC.y*AC.y;
16     T D = AB.x*AC.y-AB.y*AC.x;
17     T X = A.x+(AC.y*AB_len_sq-AB.y*AC_len_sq)
18         /(2*D);
19     T Y = A.y+(AB.x*AC_len_sq-AC.x*AB_len_sq)
20         /(2*D);
21     return point<T>(X,Y);
22 }
23 template<typename T>
24 pair<T, point<T>> MinCircleCover(vector<
25     point<T>> &p){
26     // 回傳最小覆蓋圓{半徑,中心}
27     random_shuffle(p.begin(),p.end());
28     int n = p.size();
29     point<T> c = p[0]; T r = 0;

```

```

26 for(int i=1;i<n;i++){
27     if(sign(len(c-p[i])-r) > 0){ // 不在圓內
28         c = p[i], r = 0;
29         for(int j=0;j<i;j++){
30             if(sign(len(c-p[j])-r) > 0) {
31                 c = (p[i]+p[j])/2.0;
32                 r = len(c-p[j]);
33                 for(int k=0;k<j;k++){
34                     if(sign(len(c-p[k])-r) > 0){
35                         //c=triangle<T>(p[i],p[j],p[k]).
36                         circumcenter();
37                         c = findCircumcenter(p[i],p[j],
38                             p[k]);
39                         r = len(c-p[i]);
40                     }
41                 }
42             }
43         }
44     }
45     return make_pair(r,c);
46 }

```

## 1.3 Geometry

```

1 const double PI=atan2(0.0,-1.0);
2 template<typename T>
3 struct point{
4     T x,y;
5     point(){}
6     point(const T&x,const T&y):x(x),y(y){}
7     point operator+(const point &b)const{
8         return point(x+b.x,y+b.y); }
9     point operator-(const point &b)const{
10        return point(x-b.x,y-b.y); }
11     point operator*(const T &b)const{
12        return point(x*b,y*b); }
13     point operator/(const T &b)const{
14        return point(x/b,y/b); }
15     bool operator==(const point &b)const{
16        return x==b.x&&y==b.y; }
17     T dot(const point &b)const{
18        return x*b.x+y*b.y; }
19     T cross(const point &b)const{
20        return x*b.y-y*b.x; }
21     point normal()const{//求法向量
22        return point(-y,x); }
23     T abs2()const{//向量長度的平方
24        return dot(*this); }
25     T rad(const point &b)const{//兩向量的弧度
26        return fabs(atan2(fabs(cross(b)),dot(b))); }
27     T getA()const{//對x軸的弧度
28        T A=atan2(y,x);//超過180度會變負的
29        if(A<=-PI/2)A+=PI*2;
30        return A;
31    }
32 };
33 template<typename T>
34 struct line{
35     line(){}
36     point<T> p1,p2;
37     T a,b,c;//ax+by+c=0

```

```

38 line(const point<T>&x,const point<T>&y):p1
39     (x),p2(y){}
40 void pton()const{//轉成一般式
41     a=p1.y-p2.y;
42     b=p2.x-p1.x;
43     c=-a*p1.x-b*p1.y;
44 }
45 T ori(const point<T> &p)const{//點和有向直
46     線的關係 >0左邊 <0右邊
47     return (p2-p1).cross(p-p1);
48 }
49 T btw(const point<T> &p)const{//點投影落在
50     線段上<=0
51     return (p1-p).dot(p2-p);
52 }
53 bool point_on_segment(const point<T>&p)
54     const{//點是否在線段上
55     return ori(p)==0&&btw(p)<=0;
56 }
57 T dis2(const point<T> &p,bool is_segment
58     =0)const{//點跟直線/線段的距離平方
59     point<T> v=p2-p1,v1=p-p1;
60     if(is_segment){
61         point<T> v2=p-p2;
62         if(v.dot(v1)<=0)return v1.abs2();
63         if(v.dot(v2)>=0)return v2.abs2();
64     }
65     T tmp=v.cross(v1);
66     return tmp*tmp/v.abs2();
67 }
68 T seg_dis2(const line<T> &l)const{//兩線段
69     距離平方
70     return min({dis2(l.p1,1),dis2(l.p2,1),l.
71         dis2(p1,1),l.dis2(p2,1)});
72 }
73 point<T> projection(const point<T> &p)
74     const{//點對直線的投影
75     point<T> n=(p2-p1).normal();
76     return p-n*(p-p1).dot(n)/n.abs2();
77 }
78 point<T> mirror(const point<T> &p)const{
79     //點對直線的鏡射 要先呼叫ptn轉成一般式
80     point<T> R;
81     T d=a*p+b*b;
82     R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
83     R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
84     return R;
85 }
86 bool equal(const line &l)const{//直線相等
87     return ori(l.p1)==0&&ori(l.p2)==0;
88 }
89 bool parallel(const line &l)const{
90     return (p1-p2).cross(l.p1-l.p2)==0;
91 }
92 bool cross_seg(const line &l)const{
93     return (p2-p1).cross(l.p1-p1)*(p2-p1).
94         cross(l.p2-p1)<=0;//直線是否交線段
95 }
96 int line_intersect(const line &l1,const{//
97     直線相交情況 -1無限多點 1交於一點 0
98     不相交
99     return parallel(l1)?(ori(l.p1)==0?-1:0)
100         :1;
101 }

```

```

102 }
103 int seg_intersect(const line &l1)const{
104     T c1=ori(l.p1), c2=ori(l.p2);
105     T c3=l.ori(p1), c4=l.ori(p2);
106     if(c1==0&&c2==0){//共線
107         bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
108         T a3=l.btw(p1),a4=l.btw(p2);
109         if(b1&&b2&&a3==0&&a4==0) return 2;
110         if(b1&&b2&&a3>=0&&a4==0) return 3;
111         if(b1&&b2&&a3>=0&&a4>=0) return 0;
112         return -1;//無限交點
113     }else if(c1*c2<=0&&c3*c4<=0)return 1;
114     return 0;//不相交
115 }
116 point<T> line_intersection(const line &l1)
117     const{//直線交點
118     point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
119     //if(a.cross(b)==0)return INF;
120     return p1+a*(s.cross(b)/a.cross(b));
121 }
122 point<T> seg_intersection(const line &l1)
123     const{//線段交點
124     int res=seg_intersect(l1);
125     if(res==0) assert(0);
126     if(res==2) return p1;
127     if(res==3) return p2;
128     return line_intersection(l1);
129 }
130 }
131 template<typename T>
132 struct polygon{
133     polygon(){}
134     vector<point<T>> p;//逆時針順序
135     T area()const{//面積
136         T ans=0;
137         for(int i=p.size()-1,j=0;j<(int)p.size()
138             ;i=j++){
139             ans+=p[i].cross(p[j]);
140         }
141         return ans/2;
142     }
143     point<T> center_of_mass()const{//重心
144         T cx=0,cy=0,w=0;
145         for(int i=p.size()-1,j=0;j<(int)p.size()
146             ;i=j++){
147             T a=p[i].cross(p[j]);
148             cx+=(p[i].x+p[j].x)*a;
149             cy+=(p[i].y+p[j].y)*a;
150             w+=a;
151         }
152         return point<T>(cx/3/w,cy/3/w);
153     }
154     char ahas(const point<T>&t)const{//點是否
155         在簡單多邊形內 是的話回傳1、在邊上回
156         傳-1、否則回傳0
157         bool c=0;
158         for(int i=0,j=p.size()-1;i<p.size();j=i
159             ++){
160             if(line<T>(p[i],p[j]).point_on_segment
161                 (t))return -1;
162             else if((p[i].y>t.y)!=(p[j].y>t.y)&&
163                 t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j]
164                     .y-p[i].y)+p[i].x)
165                 c=!c;
166         }
167         return c;
168     }
169 }

```

```

144 }
145 char point_in_convex(const point<T>&x)
146     const{
147     int l=1,r=(int)p.size()-2;
148     while(l<=r){//點是否在凸多邊形內，是的話
149         回傳1、在邊上回傳-1、否則回傳0
150         int mid=(l+r)/2;
151         T a1=(p[mid]-p[0]).cross(x-p[0]);
152         T a2=(p[mid+1]-p[0]).cross(x-p[0]);
153         if(a1>0&&a2<=0){
154             T res=(p[mid+1]-p[mid]).cross(x-p[
155                 mid]);
156             return res>0?1:(res>=0?-1:0);
157         }else if(a1<0)r=mid-1;
158         else l=mid+1;
159     }
160     return 0;
161 }
162 vector<T> getA(const{//凸包邊對x軸的夾角
163     vector<T>res;//一定是遞增的
164     for(size_t i=0;i<p.size();++i)
165         res.push_back((p[(i+1)%p.size()]-p[i])
166             .getA());
167     return res;
168 }
169 bool line_intersect(const vector<T>&A,
170     const line<T> &l)const{//O(LogN)
171     int f1=upper_bound(A.begin(),A.end(),(l.
172         p1-l.p2).getA())-A.begin();
173     int f2=upper_bound(A.begin(),A.end(),(l.
174         p2-l.p1).getA())-A.begin();
175     return l.cross_seg(line<T>(p[f1],p[f2]))
176         ;
177 }
178 polygon cut(const line<T> &l)const{//凸包
179     對直線切割，得到直線l左側的凸包
180     polygon ans;
181     for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
182         if(l.ori(p[i])>=0){
183             ans.p.push_back(p[i]);
184             if(l.ori(p[j])<0)
185                 ans.p.push_back(l.
186                     line_intersection(line<T>(p[i]
187                         ],p[j])));
188         }else if(l.ori(p[j])>0)
189             ans.p.push_back(l.line_intersection(
190                 line<T>(p[i],p[j])));
191     }
192     return ans;
193 }
194 static bool monotone_chain_cmp(const point
195     <T>& a,const point<T>& b){//凸包排序函
196     數
197     return (a.x<b.x)||((a.x==b.x&&a.y<b.y));
198 }
199 void monotone_chain(vector<point<T> > &s){
200     //凸包
201     sort(s.begin(),s.end(),
202         monotone_chain_cmp);
203     p.resize(s.size()+1);
204     int m=0;
205     for(size_t i=0;i<s.size();++i){
206         while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]
207             -p[m-2])<=0)--m;
208         p[m++]=s[i];
209     }
210     if(s.size()>1)--m;
211     p.resize(m);
212 }
213 T diam(){//直徑
214     int n=p.size(),t=1;
215     T ans=0;p.push_back(p[0]);
216     for(int i=0;i<n;i++){
217         point<T> now=p[i+1]-p[i];
218         while(now.cross(p[t+1]-p[i])>now.cross
219             (p[t]-p[i]))t=(t+1)%n;
220         ans=max(ans,(p[i]-p[t]).abs2());
221     }
222     return p.pop_back(),ans;
223 }
224 T min_cover_rectangle(){//最小覆蓋矩形
225     int n=p.size(),t=1,r=1,l;
226     if(n<3)return 0;//也可以做最小周長矩形
227     T ans=1e99;p.push_back(p[0]);
228     for(int i=0;i<n;i++){
229         point<T> now=p[i+1]-p[i];
230         while(now.cross(p[t+1]-p[i])>now.cross
231             (p[t]-p[i]))t=(t+1)%n;
232         while(now.dot(p[r+1]-p[i])>now.dot(p[r]
233             -p[i]))r=(r+1)%n;
234         if(!i)l=r;
235         while(now.dot(p[l+1]-p[i])<=now.dot(p[
236             l]-p[i]))l=(l+1)%n;
237         T d=now.abs2();
238         T tmp=now.cross(p[t]-p[i])*(now.dot(p[
239             r]-p[i])-now.dot(p[l]-p[i]))/d;
240         ans=min(ans,tmp);
241     }
242     return p.pop_back(),ans;
243 }
244 T dis2(polygon &p1){//凸包最近距離平方
245     vector<point<T> > &P=p,&Q=p1.p;
246     int n=P.size(),m=Q.size(),l=0,r=0;
247     for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
248     for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
249     P.push_back(P[0]),Q.push_back(Q[0]);
250     T ans=1e99;
251     for(int i=0;i<n;++i){
252         while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])
253             <0)r=(r+1)%m;
254         ans=min(ans,line<T>(P[l],P[l+1]).
255             seg_dis2(line<T>(Q[r],Q[r+1])));
256         l=(l+1)%n;
257     }
258     return P.pop_back(),Q.pop_back(),ans;
259 }
260 static char sign(const point<T>&t){
261     return (t.y==0?t.x:t.y)<0;
262 }
263 static bool angle_cmp(const line<T>& A,
264     const line<T>& B){
265     point<T> a=A.p2-A.p1,b=B.p2-B.p1;
266     return sign(a)<sign(b)||((sign(a)==sign(b)
267         )&&a.cross(b)>0);
268 }
269 }
270 int halfplane_intersection(vector<line<T>
271     > &s){//半平面交
272     sort(s.begin(),s.end(),angle_cmp);//線段
273     左側為該線段半平面
274     int L,R,n=s.size();
275     vector<point<T> > px(n);
276     vector<line<T> > q(n);
277     q[L=R=0]=s[0];
278     for(int i=1;i<n;++i){
279         while(L<R&&s[i].ori(px[R-1])<=0)--R;
280         while(L<R&&s[i].ori(px[L])<=0)+L;
281         q[++R]=s[i];
282         if(q[R].parallel(q[R-1])){
283             --R;
284             if(q[R].ori(s[i].p1)>0)q[R]=s[i];
285         }
286         if(L<R)px[R-1]=q[R-1].
287             line_intersection(q[R]);
288     }
289     while(L<R&&q[L].ori(px[R-1])<=0)--R;
290     p.clear();
291     if(R-L==1)return 0;
292     px[R]=q[R].line_intersection(q[L]);
293     for(int i=L;i<=R;++i)p.push_back(px[i]);
294     return R-L+1;
295 }
296 }
297 template<typename T>
298 struct triangle{
299     point<T> a,b,c;
300     triangle(){
301         triangle(const point<T> &a,const point<T>
302             &b,const point<T> &c):a(a),b(b),c(c){}
303     }
304     T area(const{
305         T t=(b-a).cross(c-a)/2;
306         return t>0?t:-t;
307     }
308     point<T> barycenter()const{//重心
309         return (a+b+c)/3;
310     }
311     point<T> circumcenter()const{//外心
312         static line<T> u,v;
313         u.p1=(a+b)/2;
314         u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
315             b.x);
316         v.p1=(a+c)/2;
317         v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
318             c.x);
319         return u.line_intersection(v);
320     }
321     point<T> incenter()const{//內心
322         T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
323             ()),C=sqrt((a-b).abs2());
324         return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
325             B*b.y+C*c.y)/(A+B+C);
326     }
327     point<T> perpencenter()const{//垂心
328         return barycenter()*3-circumcenter()*2;
329     }
330 }
331 }
332 template<typename T>
333 struct plane{
334     point3D<T> p0,n;//平面上的點和法向量
335     plane(){
336         plane(const point3D<T> &p0,const point3D<T>
337             &n):p0(p0),n(n){
338             point3D(const T&x,const T&y,const T&z):x(x
339                 ),y(y),z(z){
340             point3D operator+(const point3D &b)const{
341                 return point3D(x+b.x,y+b.y,z+b.z);
342             }
343             point3D operator-(const point3D &b)const{
344                 return point3D(x-b.x,y-b.y,z-b.z);
345             }
346             point3D operator*(const T &b)const{
347                 return point3D(x*b,y*b,z*b);
348             }
349             point3D operator/(const T &b)const{
350                 return point3D(x/b,y/b,z/b);
351             }
352             bool operator==(const point3D &b)const{
353                 return x==b.x&&y==b.y&&z==b.z;
354             }
355             T dot(const point3D &b)const{
356                 return x*b.x+y*b.y+z*b.z;
357             }
358             point3D cross(const point3D &b)const{
359                 return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
360                     *b.y-y*b.x);
361             }
362             T abs2()const{//向量長度的平方
363                 return dot(*this);
364             }
365             T area2(const point3D &b)const{//和b、原點
366                 圍成面積的平方
367                 return cross(b).abs2()/4;
368             }
369 };
370 template<typename T>
371 struct line3D{
372     point3D<T> p1,p2;
373     line3D(){
374         line3D(const point3D<T> &p1,const point3D<
375             T> &p2):p1(p1),p2(p2){
376             T dis2(const point3D<T> &p,bool is_segment
377                 =0)const{//點跟直線/線段的距離平方
378                 point3D<T> v=p2-p1,v1=p-p1;
379                 if(is_segment){
380                     point3D<T> v2=p-p2;
381                     if(v.dot(v1)<=0)return v1.abs2();
382                     if(v.dot(v2)>=0)return v2.abs2();
383                 }
384                 point3D<T> tmp=v.cross(v1);
385                 return tmp.abs2()/(v.abs2());
386             }
387             pair<point3D<T>,point3D<T> > closest_pair(
388                 const line3D<T> &l)const{
389                 point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
390                 point3D<T> N=v1.cross(v2),ab(p1-l.p1);
391                 //if(N.abs2()==0)return NULL;平行或重合
392                 T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
393                 最近點距離
394                 point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
395                     cross(d2),G=l.p1-p1;
396                 T t1=(G.cross(d2)).dot(D)/D.abs2();
397                 T t2=(G.cross(d1)).dot(D)/D.abs2();
398                 return make_pair(p1+d1*t1,l.p1+d2*t2);
399             }
400             bool same_side(const point3D<T> &a,const
401                 point3D<T> &b)const{
402                 return (p2-p1).cross(a-p1).dot((p2-p1).
403                     cross(b-p1))>0;
404             }
405         };
406     };
407     template<typename T>
408     struct plane{
409         point3D<T> p0,n;//平面上的點和法向量
410         plane(){
411             plane(const point3D<T> &p0,const point3D<T>
412                 &n):p0(p0),n(n){
413                 point3D(const T&x,const T&y,const T&z):x(x
414                     ),y(y),z(z){
415                 point3D operator+(const point3D &b)const{
416                     return point3D(x+b.x,y+b.y,z+b.z);
417                 }
418                 point3D operator-(const point3D &b)const{
419                     return point3D(x-b.x,y-b.y,z-b.z);
420                 }
421                 point3D operator*(const T &b)const{
422                     return point3D(x*b,y*b,z*b);
423                 }
424                 point3D operator/(const T &b)const{
425                     return point3D(x/b,y/b,z/b);
426                 }
427                 bool operator==(const point3D &b)const{
428                     return x==b.x&&y==b.y&&z==b.z;
429                 }
430                 T dot(const point3D &b)const{
431                     return x*b.x+y*b.y+z*b.z;
432                 }
433                 point3D cross(const point3D &b)const{
434                     return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
435                         *b.y-y*b.x);
436                 }
437                 T abs2()const{//向量長度的平方
438                     return dot(*this);
439                 }
440                 T area2(const point3D &b)const{//和b、原點
441                     圍成面積的平方
442                     return cross(b).abs2()/4;
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 }

```

```

357 T dis2(const point3D<T> &p) const { // 點到平
    面距離的平方
358     T tmp = (p - p0).dot(n);
359     return tmp * tmp / n.abs2();
360 }
361 point3D<T> projection(const point3D<T> &p)
    const {
362     return p - n * (p - p0).dot(n) / n.abs2();
363 }
364 point3D<T> line_intersection(const line3D<
    T> &l) const {
365     T tmp = n.dot(l.p2 - l.p1); // 等於 0 表示平行或
    重合該平面
366     return l.p1 + (l.p2 - l.p1) * (n.dot(p0 - l.p1) /
    tmp);
367 }
368 line3D<T> plane_intersection(const plane &
    pl) const {
369     point3D<T> e = n.cross(pl.n), v = n.cross(e);
370     T tmp = pl.n.dot(v); // 等於 0 表示平行或重合
    該平面
371     point3D<T> q = p0 + (v * (pl.n.dot(pl.p0 - p0)) /
    tmp);
372     return line3D<T>(q, q + e);
373 }
374 };

```

## 2 DP

### 2.1 整體二分

```

1 void compute(int L, int R, int optL, int
    optR) {
2     if (L > R)
3         return;
4     int mid = (L + (R - L) / 2);
5     DP[mid] = INF;
6     int opt = -1;
7     for (int k = optL; k <= min(mid - 1, optR);
        k++) {
8         if (DP[mid] > f(k) + w(k, mid)) {
9             DP[mid] = f(k) + w(k, mid);
10            opt = k;
11        }
12    }
13    compute(L, mid - 1, optL, opt);
14    compute(mid + 1, R, opt, optR);
15 }
16 // compute(1, n, 0, n);

```

### 2.2 LineContainer

```

1 // Usually used for DP 斜率優化
2 template<class T>
3 T floor_div(T a, T b) {
4     return a / b - ((a ^ b) < 0 && a % b != 0)
    ;

```

```

5 }
6
7 template<class T>
8 T ceil_div(T a, T b) {
9     return a / b + ((a ^ b) > 0 && a % b != 0)
    ;
10 }
11
12 namespace line_container_internal {
13
14 struct line_t {
15     mutable long long k, m, p;
16
17     inline bool operator<(const line_t& o)
        const { return k < o.k; }
18     inline bool operator<(long long x) const {
        return p < x; }
19 };
20
21 } // line_container_internal
22
23 template<bool MAX>
24 struct line_container : std::multiset<
    line_container_internal::line_t, std::
    less<>> {
25     static const long long INF = std::
        numeric_limits<long long>::max();
26
27     bool isect(iterator x, iterator y) {
28         if (y == end()) {
29             x->p = INF;
30             return 0;
31         }
32         if (x->k == y->k) {
33             x->p = (x->m > y->m ? INF : -INF);
34         } else {
35             x->p = floor_div(y->m - x->m, x->k - y
                ->k);
36         }
37         return x->p >= y->p;
38     }
39
40     void add_line(long long k, long long m) {
41         if (!MAX) {
42             k = -k;
43             m = -m;
44         }
45         auto z = insert({k, m, 0}), y = z++, x =
            y;
46         while (isect(y, z)) {
47             z = erase(z);
48         }
49         if (x != begin() && isect(--x, y)) {
50             isect(x, y = erase(y));
51         }
52         while ((y = x) != begin() && (--x)->p >=
            y->p) {
53             isect(x, erase(y));
54         }
55     }
56
57     long long get(long long x) {
58         assert(!empty());
59         auto l = *lower_bound(x);
60         return (l.k * x + l.m) * (MAX ? +1 : -1)
            ;

```

### 2.3 斜率優化-動態凸包

```

1 struct Line
2 {
3     mutable ll a, b, l;
4     Line(ll _a, ll _b, ll _l) : a(_a), b(_b)
        , l(_l) {}
5     bool operator<(const Line &rhs) const
6     {
7         return make_pair(-a, -b) < make_pair
            (-rhs.a, -rhs.b);
8     }
9     bool operator<(ll rhs_l) const
10    {
11        return l < rhs_l;
12    }
13 };
14
15 struct ConvexHullMin : std::multiset<Line,
    std::less<>>
16 {
17     static const ll INF = (1ll << 60);
18     static ll DivCeil(ll a, ll b)
19     {
20         return a / b - ((a ^ b) < 0 && a % b
            );
21     }
22     bool Intersect(iterator x, iterator y)
23     {
24         if (y == end())
25         {
26             x->l = INF;
27             return false;
28         }
29         if (x->a == y->a)
30         {
31             x->l = x->b < y->b ? INF : -INF;
32         }
33         else
34         {
35             x->l = DivCeil(y->b - x->b, x->a -
                y->a);
36         }
37         return x->l >= y->l;
38     }
39     void Insert(ll a, ll b)
40     {
41         auto z = insert(Line(a, b, 0)), y =
            z++, x = y;
42         while (Intersect(y, z))
43             z = erase(z);
44         if (x != begin() && Intersect(--x, y))
45             Intersect(x, y = erase(y));
46         while ((y = x) != begin() && (--x)->
            l >= y->l)
47             Intersect(x, erase(y));
48     }
49     ll query(ll x) const
50     {

```

```

51         auto l = *lower_bound(x);
52         return l.a * x + l.b;
53     }
54 } convexhull;
55
56 const ll maxn = 200005;
57 ll s[maxn];
58 ll f[maxn];
59 ll dp[maxn];
60 // CSES monster game2
61 int main()
62 {
63     Crbubble
64     ll n, m, i, k, t;
65     cin >> n >> f[0];
66     for (i = 1; i <= n; i++) cin >> s[i];
67     for (i = 1; i <= n; i++) cin >> f[i];
68     convexhull.Insert(f[0], 0);
69     for (i = 1; i <= n; i++)
70     {
71         dp[i] = convexhull.query(s[i]);
72         convexhull.Insert(f[i], dp[i]);
73     }
74     cout << dp[n] << endl;
75     return 0;

```

### 2.4 basic DP

```

1 // 0/1 背包問題
2 for (int i = 0; i < n; i++) {
3     for (int k = W; k >= w[i]; k--) {
4         dp[k] = max(dp[k], dp[k - w[i]] + v[i]);
5     }
6     // 因為不能重複拿，所以要倒回來
7 }
8 // 無限背包問題
9 dp[0] = 1;
10 for (int i = 0; i < n; i++) {
11     int a; cin >> a;
12     for (int k = a; k <= m; k++) {
13         dp[k] += dp[k - a];
14         if (dp[k] >= mod) dp[k] -= mod;
15     }
16 }
17 // LIS 問題
18 for (int i = 0; i < n; i++) {
19     cin >> x;
20     auto it = lower_bound(dp.begin(), dp.end()
        (), x);
21     if (it == dp.end()) {
22         dp.emplace_back(x);
23     }
24     else {
25         *it = x;
26     }
27 }
28 cout << dp.size();
29 // LCS 問題
30 signed main() {
31     string a, b;
32     cin >> a >> b;

```

```

33 vector<vector<int>> dp(a.size()+1,vector
    <int> (b.size()+1,0));
34 vector<vector<pair<int,int>>> pre(a.size()
    +1,vector<pair<int,int>> (b.size()
    +1));
35 for(int i=0;i<a.size();i++) {
36     for(int j=0;j<b.size();j++) {
37         if(a[i] == b[j]) {
38             dp[i+1][j+1] = dp[i][j] + 1;
39             pre[i+1][j+1] = {i,j};
40         }
41         else if(dp[i+1][j] >= dp[i][j+1]) {
42             dp[i+1][j+1] = dp[i+1][j];
43             pre[i+1][j+1] = {i+1,j};
44         }
45         else {
46             dp[i+1][j+1] = dp[i][j+1];
47             pre[i+1][j+1] = {i,j+1};
48         }
49     }
50 }
51 int index1 = a.size(), index2 = b.size();
52 string ans;
53 while(index1>0&&index2>0) {
54     if(pre[index1][index2] == make_pair(
        index1-1,index2-1)) {
55         ans+=a[index1-1];
56     }
57     pair<int,int> u = pre[index1][index2];
58     index1= u.first;
59     index2= u.second;
60 }
61 for(int i=ans.size()-1;i>=0;i--)cout<<
    ans[i];
62 return 0;
63 }
64 }

```

## 2.5 DP on Graph

```

1 //G.Longest Path
2 vector<vector<int>> G;
3 vector<int> in;
4 int n, m;
5 cin >> n >> m;
6 G.assign(n + 1, {});
7 in.assign(n + 1, 0);
8 while (m--) {
9     int u, v;
10    cin >> u >> v;
11    G[u].emplace_back(v);
12    ++in[v];
13 }
14 int solve(int n) {
15     vector<int> DP(G.size(), 0);
16     vector<int> Q;
17     for (int u = 1; u <= n; ++u)
18         if (in[u] == 0)
19             Q.emplace_back(u);
20     for (size_t i = 0; i < Q.size(); ++i) {

```

```

21     int u = Q[i];
22     for (auto v : G[u]) {
23         DP[v] = max(DP[v], DP[u] + 1);
24         if (--in[v] == 0)
25             Q.emplace_back(v);
26     }
27 }
28 return *max_element(DP.begin(), DP.end());
29 }
30 //max_independent_set on tree
31 vector<int> DP[2];
32 int dfs(int u, int pick, int parent = -1) {
33     if (u == parent) return 0;
34     if (DP[pick][u] return DP[pick][u];
35     if (Tree[u].size() == 1) return pick; //
        葉子
36     for (auto v : Tree[u]) {
37         if (pick == 0) {
38             DP[pick][u] += max(dfs(v, 0, u), dfs(v,
                1, u));
39         } else {
40             DP[pick][u] += dfs(v, 0, u);
41         }
42     }
43     return DP[pick][u] += pick;
44 }
45 int solve(int n) {
46     DP[0] = DP[1] = vector<int>(n + 1, 0);
47     return max(dfs(1, 0), dfs(1, 1));
48 }
49 //Traveling Salesman // AtCoder
50 const int INF = 1e9;
51 int cost(vector<tuple<int,int,int>> &point,
    int from, int to) {
52     auto [x,y,z] = point[from];
53     auto [X,Y,Z] = point[to];
54     return abs(X-x)+abs(Y-y)+max(0,Z-z);
55 } //從一個點走到另一個點的花費
56 signed main() {
57     int n;cin>>n;
58     vector<tuple<int,int,int>> point(n);
59     for(auto &[x,y,z]:point) {
60         cin>>x>>y>>z;
61     }
62     vector<vector<int>> dp(1<<n,vector<int>
        (n,INF));
63     //1<<n(2^n)代表1~n的所有子集，代表走過的
        點
64     //n代表走到的最後一個點
65     dp[0][0] = 0;
66     for(int i=1;i<(1<<n);i++) {
67         for(int j=0;j<n;j++) {
68             if(i & (1<<j)) {
69                 //j是走到的最後一個點，必須
                    要在i裡面
70                 for(int k=0;k<n;k++) {
71                     dp[i][j] = min(dp[i][j],
                        dp[i-(1<<j)][k]+cost
                        (point,k,j));
72                 }
73                 //i集合裡面走到j = i/{j}
                    集合裡走到k，再從k走
                    到j
74             }

```

```

75         }
76         //cout<<dp[i][j]<<' ';
77     }
78     //cout<<endl;
79 }
80 cout<<dp[(1<<n)-1][0]; //每個都要走到，要
    走回1
81 return 0;
82 }

```

## 2.6 單調隊列優化

```

1 long long solve(vector<int> a, int N, int K)
    {
2     vector<long long> DP(N + 1);
3     deque<int> dq(1);
4     for (int i = 1; i <= N; ++i) {
5         while (dq.front() < i - K)
6             dq.pop_front();
7         DP[i] = DP[dq.front()] + a[i];
8         while (dq.size() && DP[dq.back()] > DP[i])
9             dq.pop_back();
10        dq.push_back(i);
11    }
12    long long ans = INF;
13    for (int i = N - K + 1; i <= N; ++i)
14        ans = min(ans, DP[i]);
15    return ans;
16 }

```

## 3 Data Structure

### 3.1 sparse table

```

1 //CSES Static Range Minimum Queries
2 #define inf 1e9
3 vector<vector<int>> st;
4
5 void build_sparse_table(int n) {
6     st.assign(__lg(n)+1,vector<int> (n+1,inf));
7     ;
8     for(int i=1;i<=n;i++) cin>>st[0][i];
9     for(int i=1;(1<i)<=n;i++) {
10        for(int j=1;j+(1<i)-1 <= n;j++) {
11            st[i][j] = min(st[i-1][j],st[i-1][j
                +(1<i)-1]);
12        }
13    }
14 }
15 int query(int l, int r) {
16     int k = __lg(r - l + 1);
17     return min(st[k][l],st[k][r-(1<k)+1]);
18 }
19
20 signed main() {

```

## 3.2 BinaryTrie

```

1 template<class T>
2 struct binary_trie {
3     public:
4     binary_trie() {
5         new_node();
6     }
7
8     void clear() {
9         trie.clear();
10        new_node();
11    }
12
13    void insert(T x) {
14        for(int i = B - 1, p = 0; i >= 0; i--) {
15            int y = x >> i & 1;
16            if(trie[p].go[y] == 0) {
17                trie[p].go[y] = new_node();
18            }
19            p = trie[p].go[y];
20            trie[p].cnt += 1;
21        }
22    }
23
24    void erase(T x) {
25        for(int i = B - 1, p = 0; i >= 0; i--) {
26            p = trie[p].go[x >> i & 1];
27            trie[p].cnt -= 1;
28        }
29    }
30
31    bool contains(T x) {
32        for(int i = B - 1, p = 0; i >= 0; i--) {
33            p = trie[p].go[x >> i & 1];
34            if(trie[p].cnt == 0) {
35                return false;
36            }
37        }
38        return true;
39    }
40
41    T get_min() {
42        return get_xor_min(0);
43    }
44
45    T get_max() {
46        return get_xor_max(0);
47    }
48
49    T get_xor_min(T x) {
50        T ans = 0;
51        for(int i = B - 1, p = 0; i >= 0; i--) {
52            int y = x >> i & 1;
53            int z = trie[p].go[y];

```



```

54     if(z > 0 && trie[z].cnt > 0) {
55         p = z;
56     } else {
57         ans |= T(1) << i;
58         p = trie[p].go[y ^ 1];
59     }
60 }
61 return ans;
62 }
63
64 T get_xor_max(T x) {
65     T ans = 0;
66     for(int i = B - 1, p = 0; i >= 0; i--) {
67         int y = x >> i & 1;
68         int z = trie[p].go[y ^ 1];
69         if(z > 0 && trie[z].cnt > 0) {
70             ans |= T(1) << i;
71             p = z;
72         } else {
73             p = trie[p].go[y];
74         }
75     }
76     return ans;
77 }
78
79 private:
80     static constexpr int B = sizeof(T) * 8;
81
82     struct Node {
83         std::array<int, 2> go = {};
84         int cnt = 0;
85     };
86
87     std::vector<Node> trie;
88
89     int new_node() {
90         trie.emplace_back();
91         return (int) trie.size() - 1;
92     }
93 };

```

### 3.3 BIT

```

1 #define lowbit(x) x & -x
2
3 void modify(vector<int> &bit, int idx, int
4     val) {
5     for(int i = idx; i <= bit.size(); i+=
6         lowbit(i)) bit[i] += val;
7 }
8
9 int query(vector<int> &bit, int idx) {
10     int ans = 0;
11     for(int i = idx; i > 0; i-= lowbit(i)) ans
12         += bit[i];
13     return ans;
14 }
15
16 // the first i s.t. a[1]+...+a[i] >= k
17 int findK(vector<int> &bit, int k) {
18     int idx = 0, res = 0;
19     int mx = __lg(bit.size()) + 1;
20     for(int i = mx; i >= 0; i--) {

```

```

18     if((idx | (1<<i)) > bit.size()) continue
19     ;
20     if(res + bit[idx | (1<<i)] < k) {
21         idx = (idx | (1<<i));
22         res += bit[idx];
23     }
24     return idx + 1;
25 }
26 //O(n)建bit
27 for (int i = 1; i <= n; ++i) {
28     bit[i] += a[i];
29     int j = i + lowbit(i);
30     if (j <= n) bit[j] += bit[i];
31 }

```

### 3.4 Dynamic Segment Tree

```

1 using ll = long long;
2 struct node {
3     node *l, *r; ll sum;
4     void pull() {
5         sum = 0;
6         for(auto x : {l, r}) if(x) sum += x->sum
7             ;
8     }
9     node(int v = 0): sum(v) {l = r = nullptr;}
10 };
11 void upd(node& o, int x, ll v, int l, int r
12     ) {
13     if(!o) o = new node;
14     if(l == r) return o->sum += v, void();
15     int m = (l + r) / 2;
16     if(x <= m) upd(o->l, x, v, l, m);
17     else upd(o->r, x, v, m+1, r);
18     o->pull();
19 }
20 ll qry(node* o, int ql, int qr, int l, int r
21     ) {
22     if(!o) return 0;
23     if(ql <= l && r <= qr) return o->sum;
24     int m = (l + r) / 2; ll ret = 0;
25     if(ql <= m) ret += qry(o->l, ql, qr, l, m)
26         ;
27     if(qr > m) ret += qry(o->r, ql, qr, m+1, r
28         );
29     return ret;
30 }

```

### 3.5 掃描線 + 線段樹

```

1 //CSES Area of Rectangle
2 #define pb push_back
3 #define int long long
4 #define mid ((l + r) >> 1)
5 #define lc (p << 1)
6 #define rc ((p << 1) | 1)

```

```

7 struct ooo{
8     int x, l, r, v;
9 };
10 const int inf = 1e6;
11 array<int, 800004> man, tag, cnt;
12 vector<ooo> Q;
13 bool cmp(ooo a, ooo b){
14     return a.x < b.x;
15 }
16 void pull(int p){
17     man[p] = min(man[lc], man[rc]);
18     if(man[lc] < man[rc]) cnt[p] = cnt[lc];
19     else if(man[rc] < man[lc]) cnt[p] = cnt[rc];
20     else cnt[p] = cnt[lc] + cnt[rc];
21 }
22 void push(int p){
23     man[lc] += tag[p];
24     man[rc] += tag[p];
25     tag[lc] += tag[p];
26     tag[rc] += tag[p];
27     tag[p] = 0;
28 }
29 void build(int p, int l, int r){
30     if(l == r){
31         cnt[p] = 1;
32         return;
33     }
34     build(lc, l, mid);
35     build(rc, mid + 1, r);
36     pull(p);
37 }
38 void update(int p, int l, int r, int ql, int
39     qr, int x){
40     if(ql > r || qr < l) return;
41     if(ql <= l && qr >= r){
42         man[p] += x;
43         tag[p] += x;
44         return;
45     }
46     push(p);
47     update(lc, l, mid, ql, qr, x);
48     update(rc, mid + 1, r, ql, qr, x);
49     pull(p);
50 }
51 signed main(){
52     int n, x1, y1, x2, y2, p = 0, sum = 0;
53     cin >> n;
54     for(int i = 1; i <= n; i++){
55         cin >> x1 >> y1 >> x2 >> y2;
56         Q.pb({x1, y1, y2 - 1, 1});
57         Q.pb({x2, y1, y2 - 1, -1});
58     }
59     sort(Q.begin(), Q.end(), cmp);
60     build(1, -inf, inf);
61     for(int i = -inf; i < inf; i++){
62         while(p < Q.size() && Q[p].x == i){
63             auto [x, l, r, v] = Q[p++];
64             update(1, -inf, inf, l, r, v);
65         }
66         sum += 2 * inf + 1 - cnt[1];
67     }
68     cout << sum << "\n";
69 }

```

//長方形面積

```

70 long long AreaOfRectangles(vector<tuple<int,
71     int,int,int>>>v){
72     vector<tuple<int,int,int,int>>>tmp;
73     int L = INT_MAX, R = INT_MIN;
74     for(auto [x1,y1,x2,y2]:v){
75         tmp.push_back({x1,y1+1,y2,1});
76         tmp.push_back({x2,y1+1,y2,-1});
77         R = max(R,y2);
78         L = min(L,y1);
79     }
80     vector<long long>seg((R-L+1)<<2),tag((R-L
81         +1)<<2);
82     sort(tmp.begin(),tmp.end());
83     function<void(int,int,int,int,int,int)>
84         update = [&](int ql,int qr,int val,int
85             l,int r,int idx){
86         if(ql<=l and r<=qr){
87             tag[idx]+=val;
88             if(tag[idx])seg[idx] = r-l+1;
89             else if(l==r)seg[idx] = 0;
90             else seg[idx] = seg[idx<<1]+seg[idx
91                 <<1|1];
92             return;
93         }
94         int m = (l+r)>>1;
95         if(ql<=m)update(ql,qr,val,l,m,idx<<1);
96         if(qr>m)update(ql,qr,val,m+1,r,idx<<1|1);
97         if(tag[idx])seg[idx] = r-l+1;
98         else seg[idx] = seg[idx<<1]+seg[idx
99             <<1|1];
100     };
101     long long last_pos = 0,ans = 0;
102     for(auto [pos,l,r,val]:tmp){
103         ans+=(pos-last_pos)*seg[l];
104         update(l,r,val,l,r,1);
105         last_pos = pos;
106     }
107     return ans;
108 }
109 // CSES Intersection Points
110 #define int long long
111 #define pb push_back
112 struct line{
113     int p, l, r;
114 };
115 const int inf = 1e6 + 1;
116 array<int, 200004> BIT;
117 vector<line> A, Q;
118 bool cmp(line a, line b){
119     return a.p < b.p;
120 }
121 void update(int p, int x){
122     for(; p < 200004; p += p & -p) BIT[p]
123         += x;
124 }
125 int query(int p){
126     int sum = 0;
127     for(; p; p -= p & -p) sum += BIT[p];
128     return sum;
129 }
130 int run(){
131     int ans = 0, p = 0;
132     for(auto [t, l, r] : Q){
133         while(p < A.size()){

```

```

128     auto [x, y, v] = A[p];
129     if(x > t) break;
130     update(y, v);
131     p++;
132 }
133 ans += query(r) - query(l - 1);
134 }
135 return ans;
136 }
137 signed main(){
138     int n, x1, x2, y1, y2;
139     cin >> n;
140     for(int i = 0; i < n; i++){
141         cin >> x1 >> y1 >> x2 >> y2;
142         x1 += inf, x2 += inf, y1 += inf, y2
143             += inf;
144         if(x1 == x2) Q.pb({x1, y1, y2});
145         else A.pb({x1, y1, 1}), A.pb({x2 +
146             1, y2, -1});
147     }
148     sort(Q.begin(), Q.end(), cmp);
149     sort(A.begin(), A.end(), cmp);
150     cout << run() << "\n";
151 }

```

### 3.6 Persistent DSU

```

1 int rk[200001] = {};
2 struct Persistent_DSU{
3     rope<int>*p;
4     int n;
5     Persistent_DSU(int _n = 0):n(_n){
6         if(n==0)return;
7         p = new rope<int>;
8         int tmp[n+1] = {};
9         for(int i = 1;i<=n;++i)tmp[i] = i;
10        p->append(tmp,n+1);
11    }
12    Persistent_DSU(const Persistent_DSU &tmp){
13        p = new rope<int>(*tmp.p);
14        n = tmp.n;
15    }
16    int Find(int x){
17        int px = p->at(x);
18        return px==x?x:Find(px);
19    }
20    bool Union(int a,int b){
21        int pa = Find(a),pb = Find(b);
22        if(pa==pb)return 0;
23        if(rk[pa]<rk[pb])swap(pa,pb);
24        p->replace(pb,pa);
25        if(rk[pa]==rk[pb])rk[pa]++;
26        return 1;
27    }
28 };

```

### 3.7 DSU

```

1 struct DSU {
2     vector<int> dsu, sz;

```

```

3     DSU(int n) {
4         dsu.resize(n + 1);
5         sz.resize(n + 1, 1);
6         for (int i = 0; i <= n; i++) dsu[i] = i;
7     }
8     int find(int x) {
9         return (dsu[x] == x ? x : dsu[x] = find(
10             dsu[x]));
11     }
12     int unite(int a, int b) {
13         a = find(a), b = find(b);
14         if(a == b) return 0;
15         if(sz[a] > sz[b]) swap(a, b);
16         dsu[a] = b;
17         sz[b] += sz[a];
18         return 1;
19     };

```

### 3.8 陣列上 Treap

```

1
2
3 struct Treap {
4     Treap *lc = nullptr, *rc = nullptr;
5     unsigned pri, sz;
6     long long Val, Sum;
7     Treap(int Val):pri(rand()),sz(1),Val(Val),
8         Sum(Val),Tag(false) {}
9     void pull();
10    bool Tag;
11    void push();
12 } *root;
13
14 inline unsigned sz(Treap *x) {
15     return x ? x->sz:0;
16 }
17
18 inline void Treap::push() {
19     if(!Tag) return ;
20     swap(lc,rc);
21     if(lc) lc->Tag ^= Tag;
22     if(rc) rc->Tag ^= Tag;
23     Tag = false;
24 }
25
26 inline void Treap::pull() {
27     sz = 1;
28     Sum = Val;
29     if(lc) {
30         sz += lc->sz;
31         Sum += lc->Sum;
32     }
33     if(rc) {
34         sz += rc->sz;
35         Sum += rc->Sum;
36     }
37 }
38
39 Treap *merge(Treap *a, Treap *b) {
40     if(!a || !b) return a ? a : b;
41     if(a->pri < b->pri) {
42         a->push();

```

```

43         a->rc = merge(a->rc,b);
44         a->pull();
45         return a;
46     }
47     else {
48         b->push();
49         b->lc = merge(a,b->lc);
50         b->pull();
51         return b;
52     }
53 }
54 pair<Treap *,Treap *> splitK(Treap *x,
55     unsigned K) {
56     Treap *a = nullptr, *b = nullptr;
57     if(!x) return {a,b};
58     x->push();
59     unsigned leftSize = sz(x->lc) + 1;
60     if(K >= leftSize) {
61         a = x;
62         tie(a->rc,b) = splitK(x->rc, K -
63             leftSize);
64     }
65     else {
66         b = x;
67         tie(a, b->lc) = splitK(x->lc, K);
68     }
69     x->pull();
70     return {a,b};
71 }
72
73 Treap *init(const vector<int> &a) {
74     Treap *root = nullptr;
75     for(size_t i = 0;i < a.size(); i++) {
76         root = merge(root,new Treap(a[i]));
77     }
78     return root;
79 }
80
81 long long query(Treap *root, unsigned ql,
82     unsigned qr) {
83     auto [a,b] = splitK(root,ql);
84     auto [c,d] = splitK(b,qr-ql+1);
85     c->push();
86     long long Sum = c->Sum;
87     root = merge(a,merge(c,d));
88     return Sum;
89 }
90
91 void Reverse(Treap *root, unsigned ql,
92     unsigned qr) {
93     auto [a,b] = splitK(root,ql);
94     auto [c,d] = splitK(b,qr-ql+1);
95     c->Tag ^= true;
96     root = merge(a, merge(c,d));
97 }

```

### 3.9 monotonic stack

```

1
2 long long maxRectangle(vector<int> &h) {
3     h.emplace_back(0);
4     stack<pair<int,int>> stick;

```

```

5     long long ans = 0;
6     for(int i = 0; i < h.size(); i++) {
7         int corner = i;
8         while(stick.size() && stick.top().
9             first >= h[i]) {
10            corner = stick.top().second;
11            ans = max(ans, 1LL * (i - corner
12                ) * stick.top().first);
13            stick.pop();
14        }
15        stick.emplace(h[i],corner);
16    }
17    return ans;
18 }

```

### 3.10 Kruskal

```

1 vector<tuple<int,int,int>> Edges;
2 int kruskal(int N) {
3     int cost = 0;
4     sort(Edges.begin(), Edges.end());
5
6     DisjointSet ds(N);
7
8     sort(Edges.begin(), Edges.end());
9     for(auto [w, s, t] : Edges) {
10        if (!ds.same(s, t)) {
11            cost += w;
12            ds.unit(s, t);
13        }
14    }
15    return cost;
16 }

```

### 3.11 Lazytag Segment Tree

```

1 using ll = long long;
2 const int N = 2e5 + 5;
3 #define lc(x) (x << 1)
4 #define rc(x) (x << 1 | 1)
5
6 // [1,n]
7 // tag[i] represents the modifications to be
8 // applied to the children,
9 // while seg[i] has already been modified.
10 ll seg[N << 2], tag[N << 2];
11 int n;
12
13 void pull(int id) {
14     seg[id] = seg[lc(id)] + seg[rc(id)];
15 }
16
17 void push(int id, int l, int r) {
18     if (tag[id]) {
19         int m = (l + r) >> 1;
20         tag[lc(id)] += tag[id], tag[rc(id)] +=
21             tag[id];
22         seg[lc(id)] += (m - l + 1) * tag[id],
23             seg[rc(id)] += (r - m) * tag[id];
24         tag[id] = 0;
25     }

```

```

22 }
23 }
24
25 void upd(int ql, int qr, ll v, int l = 1,
26         int r = n, int id = 1) {
27     if (ql <= l && r <= qr) return tag[id] +=
28         v, seg[id] += (r - l + 1) * v, void();
29     push(id, l, r);
30     int m = (l + r) >> 1;
31     if (ql <= m) upd(ql, qr, v, l, m, lc(id));
32     if (qr > m) upd(ql, qr, v, m + 1, r, rc(id));
33     pull(id);
34 }
35
36 ll qry(int ql, int qr, int l = 1, int r = n,
37        int id = 1) {
38     if (ql <= l && r <= qr) return seg[id];
39     push(id, l, r);
40     int m = (l + r) >> 1; ll ret = 0;
41     if (ql <= m) ret += qry(ql, qr, l, m, lc(id));
42     if (qr > m) ret += qry(ql, qr, m + 1, r, rc(id));
43     return ret;
44 }

```

## 3.12 2D BIT

```

1 //2維BIT
2 #define lowbit(x) (x&-x)
3
4 class BIT {
5     int n;
6     vector<int> bit;
7
8 public:
9     void init(int _n) {
10         n = _n;
11         bit.resize(n + 1);
12         for(auto &b : bit) b = 0;
13     }
14     int query(int x) const {
15         int sum = 0;
16         for(; x; x -= lowbit(x))
17             sum += bit[x];
18         return sum;
19     }
20     void modify(int x, int val) {
21         for(; x <= n; x += lowbit(x))
22             bit[x] += val;
23     }
24 };
25
26 class BIT2D {
27     int m;
28     vector<BIT> bit1D;
29
30 public:
31     void init(int _m, int _n) {
32         m = _m;
33         bit1D.resize(m + 1);
34     }

```

```

35         for(auto &b : bit1D) b.init(_n);
36     }
37     int query(int x, int y) const {
38         int sum = 0;
39         for(; x; x -= lowbit(x))
40             sum += bit1D[x].query(y);
41         return sum;
42     }
43     void modify(int x, int y, int val) {
44         for(; x <= m; x += lowbit(x))
45             bit1D[x].modify(y, val);
46     }
47 };

```

## 3.13 monotonic queue

```

1 vector<int> maxSlidingWindow(vector<int> &
2 num, int k) {
3     deque<int> dq;
4     vector<int> ans;
5     for(int i = 0; i < num.size(); i++) {
6         while(dq.size() && dq.front() <= i -
7             k) dq.pop_front();
8         while(dq.size() && num[dq.back()] <
9             num[i]) dq.pop_back();
10        dq.emplace_back(i);
11        if(i >= k - 1) ans.emplace_back(num[
12            dq.front()]);
13    }
14    return ans;
15 }

```

## 3.14 Prim

```

1 int cost[MAX_V][MAX_V]; //Edge的權重 (不存在
2 時為INF)
3 int mincost[MAX_V]; //來自集合X的邊的最小權重
4 bool used[MAX_V]; //頂點i是否包含在X之中
5 int V; //頂點數
6
7 int prim() {
8     for(int i = 0; i < V; i++) {
9         mincost[i] = INF;
10        used[i] = false;
11    }
12    mincost[0] = 0;
13    int res = 0;
14    while(true) {
15        int v = -1;
16        //從不屬於X的頂點中尋找會讓來自X的邊
17        //之權重最小的頂點
18        for(int u = 0; u < V; u++) {
19            if(!used[u] && (v == -1 || mincost
20                [u] < mincost[v])) v = u;
21        }
22        if(v == -1) break;
23        used[v] = true; //將頂點v追加至X
24    }

```

```

21     res += mincost[v]; //加上邊的權重
22     for(int u = 0; u < V; u++) {
23         mincost[u] = min(mincost[u], cost
24             [v][u]);
25     }
26     return res;
27 }

```

## 3.15 回滾並查集

```

1 struct dsu_undo{
2     vector<int> sz, p;
3     int comps;
4     dsu_undo(int n){
5         sz.assign(n+5, 1);
6         p.resize(n+5);
7         for(int i = 1; i <= n; ++i) p[i] = i;
8         comps = n;
9     }
10    vector<pair<int, int>> opt;
11    int Find(int x){
12        return x == p[x] ? x : Find(p[x]);
13    }
14    bool Union(int a, int b){
15        int pa = Find(a), pb = Find(b);
16        if(pa == pb) return 0;
17        if(sz[pa] < sz[pb]) swap(pa, pb);
18        sz[pa] += sz[pb];
19        p[pb] = pa;
20        opt.push_back({pa, pb});
21        comps--;
22        return 1;
23    }
24    void undo(){
25        auto [pa, pb] = opt.back();
26        opt.pop_back();
27        p[pb] = pb;
28        sz[pa] -= sz[pb];
29        comps++;
30    }
31 };

```

## 3.16 TimingSegmentTree

```

1 template<class T, class D> struct
2     timing_segment_tree{
3     struct node{
4         int l, r;
5         vector<T> opt;
6     };
7     vector<node> arr;
8     void build(int l, int r, int idx = 1){
9         if(idx == 1) arr.resize((r - l + 1) << 2);
10        if(l == r){
11            arr[idx].l = l;
12            arr[idx].r = r;
13            return;
14        }
15        int m = (l + r) >> 1;

```

```

15        build(l, m, idx << 1);
16        build(m + 1, r, idx << 1 | 1);
17        arr[idx].l = l, arr[idx].r = r;
18        arr[idx].opt.clear();
19    }
20    void update(int ql, int qr, T k, int idx = 1)
21    {
22        if(ql <= arr[idx].l && arr[idx].r <= qr){
23            arr[idx].opt.push_back(k);
24            return;
25        }
26        int m = (arr[idx].l + arr[idx].r) >> 1;
27        if(ql <= m) update(ql, qr, k, idx << 1);
28        if(qr > m) update(ql, qr, k, idx << 1 | 1);
29    }
30    void dfs(D &d, vector<int> &ans, int idx = 1)
31    {
32        int cnt = 0;
33        for(auto [a, b] : arr[idx].opt){
34            if(d.Union(a, b)) cnt++;
35        }
36        if(arr[idx].l == arr[idx].r) ans[arr[idx].l
37            ] = d.comps;
38        else{
39            dfs(d, ans, idx << 1);
40            dfs(d, ans, idx << 1 | 1);
41        }
42        while(cnt-- > 0) d.undo();
43    }
44 };

```

## 3.17 SegmentTree

```

1 //build
2 const int N = 100000 + 9;
3 int a[N]; //葉
4 int seg[4 * N];
5 void build(int id, int l, int r) { // 編號為
6     id 的節點 · 存的區間為 [l, r]
7     if (l == r) {
8         seg[id] = a[l]; // 葉節點的值
9         return;
10    }
11    int mid = (l + r) / 2; // 將區間切成兩半
12    build(id * 2, l, mid); // 左子節點
13    build(id * 2 + 1, mid + 1, r); // 右子節
14    點
15    seg[id] = seg[id * 2] + seg[id * 2 + 1]
16 }
17 //區間查詢
18
19 int query(int id, int l, int r, int ql, int
20 qr) {
21     if (r < ql || qr < l) return 0; //若目前
22     的區間與詢問的區間的交集為空的話 ·
23     return 0
24     if (ql <= l && r <= qr) return seg[id];
25     //若目前的區間是詢問的區間的字集
26     話 · 則終止 · 並回傳當前節點的答案

```

```

22 int mid = (l + r) / 2;
23 return query(id * 2, l, mid, ql, qr) //
    左
24 + query(id * 2 + 1, mid + 1, r, ql,
    qr); // 右
25 // 否則，往左、右進行遞迴
26 }
27
28 // 單點修改
29
30 void modify(int id, int l, int r, int i, int
    x) {
31     if (l == r) {
32         seg[id] = x; // 將a[i]改成x
33         // seg[id] += x; // 將a[i]加上x
34         return;
35     }
36     int mid = (l + r) / 2;
37     // 根據修改的點在哪裡，來決定要往哪個子
38     // 樹進行DFS
39     if (i <= mid) modify(id * 2, l, mid, i,
        x); // 左
40     else modify(id * 2 + 1, mid + 1, r, i, x
        ); // 右
41     seg[id] = seg[id * 2] + seg[id * 2 + 1];
42 }

```

## 3.18 Persistent Segment Tree

```

1 using ll = long long;
2 int n;
3
4 struct node {
5     node *l, *r; ll sum;
6     void pull() {
7         sum = 0;
8         for (auto x : {l, r})
9             if (x) sum += x->sum;
10    }
11    node(int v = 0): sum(v) {l = r = nullptr;}
12 } *root = nullptr;
13
14 void upd(node *prv, node* cur, int x, int v,
    int l = 1, int r = n) {
15     if (l == r) return cur->sum = v, void();
16     int m = (l + r) >> 1;
17     if (x <= m) cur->r = prv->r, upd(prv->l,
        cur->l = new node, x, v, l, m);
18     else cur->l = prv->l, upd(prv->r, cur->r =
        new node, x, v, m + 1, r);
19     cur->pull();
20 }
21
22 ll qry(node* a, node* b, int ql, int qr, int
    l = 1, int r = n) {
23     if (ql <= l && r <= qr) return b->sum - a
        ->sum;
24     int m = (l + r) >> 1; ll ret = 0;
25     if (ql <= m) ret += qry(a->l, b->l, ql, qr
        , l, m);

```

```

26 if (qr > m) ret += qry(a->r, b->r, ql, qr,
    m + 1, r);
27 return ret;
28 }

```

## 3.19 pbds

```

1 #include <ext/pb_ds/tree_policy.hpp>
2 #include <ext/pb_ds/assoc_container.hpp>
3 using namespace __gnu_pbds;
4
5 template <class T>
6 using ordered_set = tree<T, null_type, less<
    T>, rb_tree_tag,
    tree_order_statistics_node_update>;
7
8 template <class T>
9 // ordered_multiset: do not use erase method
    , use myerase() instead
10 using ordered_multiset = tree<T, null_type,
    less_equal<T>, rb_tree_tag,
    tree_order_statistics_node_update>;
11
12 template <class T>
13 void myerase(ordered_multiset<T> &ss, T v)
14 {
15     T rank = ss.order_of_key(v); //
        Number of elements that are less
        than v in ss
16     auto it = ss.find_by_order(rank); //
        Iterator that points to the element
        which index = rank
17     ss.erase(it);
18 }

```

## 3.20 LiChaoST

```

1 struct L {
2     ll m, k, id;
3     L(): id(-1) {}
4     L(ll a, ll b, ll c) : m(a), k(b), id(c) {}
5     ll at(ll x) { return m * x + k; }
6 };
7 class LiChao { // maintain max
private:
8     int n; vector<L> nodes;
9     void insert(int l, int r, int rt, L ln) {
10         int m = (l + r) >> 1;
11         if (nodes[rt].id == -1)
12             return nodes[rt] = ln, void();
13         bool atLeft = nodes[rt].at(l) < ln.at(l)
14             ;
15         if (nodes[rt].at(m) < ln.at(m))
16             atLeft ^= 1, swap(nodes[rt], ln);
17         if (r - l == 1) return;
18         if (atLeft) insert(l, m, rt << 1, ln);
19         else insert(m, r, rt << 1 | 1, ln);
20     }
21     ll query(int l, int r, int rt, ll x) {
22         int m = (l + r) >> 1; ll ret = -INF;

```

```

23         if (nodes[rt].id != -1) ret = nodes[rt].
            at(x);
24         if (r - l == 1) return ret;
25         if (x < m) return max(ret, query(l, m,
            rt << 1, x));
26         return max(ret, query(m, r, rt << 1 | 1,
            x));
27     }
28 public:
29     LiChao(int n_) : n(n_), nodes(n * 4) {}
30     void insert(L ln) { insert(0, n, 1, ln); }
31     ll query(ll x) { return query(0, n, 1, x); }
32 };

```

## 3.21 DynamicMST

```

1 int cnt[maxn], cost[maxn], st[maxn], ed[maxn]
    ;
2 pair<int, int> qr[maxn];
3 // qr[i].first = id of edge to be changed,
    qr[i].second = weight after operation
4 // cnt[i] = number of operation on edge i
5 // call solve(0, q - 1, v, 0), where v
    contains edges i such that cnt[i] == 0
6
7 void contract(int l, int r, vector<int> v,
    vector<int> &x, vector<int> &y) {
8     sort(v.begin(), v.end(), [&](int i, int j)
9         {
10             if (cost[i] == cost[j]) return i < j;
11             return cost[i] < cost[j];
12         });
13     djs.save();
14     for (int i = l; i <= r; ++i) djs.merge(st[
        qr[i].first], ed[qr[i].first]);
15     for (int i = 0; i < (int)v.size(); ++i) {
16         if (djs.find(st[v[i]]) != djs.find(ed[v[
            i]])) {
17             x.push_back(v[i]);
18             djs.merge(st[v[i]], ed[v[i]]);
19         }
20     }
21     djs.undo();
22     djs.save();
23     for (int i = 0; i < (int)x.size(); ++i)
24         djs.merge(st[x[i]], ed[x[i]]);
25     for (int i = 0; i < (int)v.size(); ++i) {
26         if (djs.find(st[v[i]]) != djs.find(ed[v[
            i]])) {
27             y.push_back(v[i]);
28             djs.merge(st[v[i]], ed[v[i]]);
29         }
30     }
31     djs.undo();
32 void solve(int l, int r, vector<int> v, long
    long c) {
33     if (l == r) {
34         cost[qr[l].first] = qr[l].second;
35         if (st[qr[l].first] == ed[qr[l].first])
36             {

```

```

36         printf("%lld\n", c);
37         return;
38     }
39     int minv = qr[l].second;
40     for (int i = 0; i < (int)v.size(); ++i)
41         minv = min(minv, cost[v[i]]);
42     printf("%lld\n", c + minv);
43     return;
44 }
45 int m = (l + r) >> 1;
46 vector<int> lv = v, rv = v;
47 vector<int> x, y;
48 for (int i = m + 1; i <= r; ++i) {
49     cnt[qr[i].first]--;
50     if (cnt[qr[i].first] == 0) lv.push_back(
        qr[i].first);
51 }
52 contract(l, m, lv, x, y);
53 long long lc = c, rc = c;
54 djs.save();
55 for (int i = 0; i < (int)x.size(); ++i) {
56     lc += cost[x[i]];
57     djs.merge(st[x[i]], ed[x[i]]);
58 }
59 solve(l, m, y, lc);
60 djs.undo();
61 x.clear(), y.clear();
62 for (int i = m + 1; i <= r; ++i) cnt[qr[i]
    ].first++;
63 for (int i = l; i <= m; ++i) {
64     cnt[qr[i].first]--;
65     if (cnt[qr[i].first] == 0) rv.push_back(
        qr[i].first);
66 }
67 contract(m + 1, r, rv, x, y);
68 djs.save();
69 for (int i = 0; i < (int)x.size(); ++i) {
70     rc += cost[x[i]];
71     djs.merge(st[x[i]], ed[x[i]]);
72 }
73 solve(m + 1, r, y, rc);
74 djs.undo();
75 for (int i = l; i <= m; ++i) cnt[qr[i].
    first]++;

```

## 4 Flow

### 4.1 Property

- 1 最大流 = 最小割
- 2 最大獨立集 = 補圖最大團 =  $V$  - 最小頂點覆蓋
- 3 二分圖最大匹配 = 二分圖最小頂點覆蓋
- 4 二分圖最大匹配加s,t點 = 最大流



## 4.2 Gomory Hu

```

1 //最小割樹+求任兩點間最小割
2 //0-base, root=0
3 LL e[MAXN][MAXN]; //任兩點間最小割
4 int p[MAXN]; //parent
5 ISAP D; // original graph
6 void gomory_hu(){
7     fill(p, p+n, 0);
8     fill(e[0], e[n], INF);
9     for( int s = 1; s < n; ++s ) {
10         int t = p[s];
11         ISAP F = D;
12         LL tmp = F.min_cut(s, t);
13         for( int i = 1; i < s; ++i )
14             e[s][i] = e[i][s] = min(tmp, e[t][i]);
15         for( int i = s+1; i <= n; ++i )
16             if( p[i] == t && F.vis[i] ) p[i] = s;
17     }
18 }

```

## 4.3 MinCostMaxFlow

```

1 template<class Cap_t, class Cost_t>
2 class MCMF {
3 public:
4     struct Edge {
5         int from;
6         int to;
7         Cap_t cap;
8         Cost_t cost;
9         Edge(int u, int v, Cap_t _cap, Cost_t
10             _cost) : from(u), to(v), cap(_cap),
11             cost(_cost) {}
12     };
13
14     static constexpr Cap_t EPS = static_cast<
15         Cap_t>(1e-9);
16
17     int n;
18     vector<Edge> edges;
19     vector<vector<int>> g;
20     vector<Cost_t> d;
21     vector<bool> in_queue;
22     vector<int> previous_edge;
23
24     MCMF() {}
25     MCMF(int _n) : n(_n+1), g(_n+1), d(_n+1),
26         in_queue(_n+1), previous_edge(_n+1) {}
27
28     void add_edge(int u, int v, Cap_t cap,
29         Cost_t cost) {
30         assert(0 <= u && u < n);
31         assert(0 <= v && v < n);
32         g[u].push_back(edges.size());
33         edges.emplace_back(u, v, cap, cost);
34         g[v].push_back(edges.size());
35         edges.emplace_back(v, u, 0, -cost);
36     }
37
38     bool spfa(int s, int t) {
39         bool found = false;

```

```

35     fill(d.begin(), d.end(), numeric_limits<
36         Cost_t>::max());
37     d[s] = 0;
38     in_queue[s] = true;
39     queue<int> que;
40     que.push(s);
41     while(!que.empty()) {
42         int u = que.front();
43         que.pop();
44         if(u == t) {
45             found = true;
46         }
47         in_queue[u] = false;
48         for(auto& id : g[u]) {
49             const Edge& e = edges[id];
50             if(e.cap > EPS && d[u] + e.cost < d[
51                 e.to]) {
52                 d[e.to] = d[u] + e.cost;
53                 previous_edge[e.to] = id;
54                 if(!in_queue[e.to]) {
55                     que.push(e.to);
56                     in_queue[e.to] = true;
57                 }
58             }
59         }
60     }
61     return found;
62 }
63
64 pair<Cap_t, Cost_t> flow(int s, int t,
65     Cap_t f = numeric_limits<Cap_t>::max())
66 {
67     assert(0 <= s && s < n);
68     assert(0 <= t && t < n);
69     Cap_t cap = 0;
70     Cost_t cost = 0;
71     while(f > 0 && spfa(s, t)) {
72         Cap_t send = f;
73         int u = t;
74         while(u != s) {
75             const Edge& e = edges[previous_edge[
76                 u]];
77             send = min(send, e.cap);
78             u = e.from;
79         }
80         u = t;
81         while(u != s) {
82             Edge& e = edges[previous_edge[u]];
83             e.cap -= send;
84             Edge& b = edges[previous_edge[u] ^
85                 1];
86             b.cap += send;
87             f -= send;
88             cost += send * d[t];
89         }
90     }
91     return make_pair(cap, cost);
92 }

```

## 4.4 dinic

```

1 template<class T>
2 struct Dinic{
3     struct edge{
4         int from, to;
5         T cap;
6         edge(int _from, int _to, T _cap) : from(
7             _from), to(_to), cap(_cap) {}
8     };
9     int n;
10    vector<edge> edges;
11    vector<vector<int>> g;
12    vector<int> cur, h;
13    Dinic(int _n) : n(_n+1), g(_n+1) {}
14    void add_edge(int u, int v, T cap){
15        g[u].push_back(edges.size());
16        edges.push_back(edge(u, v, cap));
17        g[v].push_back(edges.size());
18        edges.push_back(edge(v, u, 0));
19    }
20    bool bfs(int s,int t){
21        h.assign(n, -1);
22        h[s] = 0;
23        queue<int> que;
24        que.push(s);
25        while(!que.empty()) {
26            int u = que.front();
27            que.pop();
28            for(auto id : g[u]) {
29                const edge& e = edges[id];
30                int v = e.to;
31                if(e.cap > 0 && h[v] == -1) {
32                    h[v] = h[u] + 1;
33                    if(v == t) {
34                        return 1;
35                    }
36                }
37                que.push(v);
38            }
39        }
40        return 0;
41    }
42    T dfs(int u, int t, T f) {
43        if(u == t) {
44            return f;
45        }
46        T r = f;
47        for(int& i = cur[u]; i < (int) g[u].size
48            (); ++i) {
49            int id = g[u][i];
50            const edge& e = edges[id];
51            int v = e.to;
52            if(e.cap > 0 && h[v] == h[u] + 1) {
53                T send = dfs(v, t, min(r, e.cap));
54                edges[id].cap -= send;
55                edges[id ^ 1].cap += send;
56                r -= send;
57                if(r == 0) {
58                    return f;
59                }
60            }
61        }
62        return f - r;
63    }

```

```

62    T flow(int s, int t, T f = numeric_limits<
63        T>::max()) {
64        T ans = 0;
65        while(f > 0 && bfs(s, t)) {
66            cur.assign(n, 0);
67            T send = dfs(s, t, f);
68            ans += send;
69            f -= send;
70        }
71        return ans;
72    }
73    vector<pair<int,int>> min_cut(int s) {
74        vector<bool> vis(n);
75        vis[s] = true;
76        queue<int> que;
77        que.push(s);
78        while(!que.empty()) {
79            int u = que.front();
80            que.pop();
81            for(auto id : g[u]) {
82                const auto& e = edges[id];
83                int v = e.to;
84                if(e.cap > 0 && !vis[v]) {
85                    vis[v] = true;
86                    que.push(v);
87                }
88            }
89        }
90        vector<pair<int,int>> cut;
91        for(int i = 0; i < (int) edges.size(); i
92            += 2) {
93            const auto& e = edges[i];
94            if(vis[e.from] && !vis[e.to]) {
95                cut.push_back(make_pair(e.from, e.to
96                    ));
97            }
98        }
99        return cut;
100    }

```

## 4.5 ISAP with cut

```

1 template<typename T>
2 struct ISAP{
3     static const int MAXN=105;
4     static const T INF=INT_MAX;
5     int n; //點數
6     int d[MAXN], gap[MAXN], cur[MAXN];
7     struct edge{
8         int v, pre;
9         T cap, r;
10        edge(int v, int pre, T cap):v(v), pre(pre),
11            cap(cap), r(cap){}
12    };
13    int g[MAXN];
14    vector<edge> e;
15    void init(int _n){
16        memset(g, -1, sizeof(int)*((n=_n)+1));
17        e.clear();
18    }
19    void add_edge(int u, int v, T cap, bool
20        directed=false){

```

```

19 e.push_back(edge(v,g[u],cap));
20 g[u]=e.size()-1;
21 e.push_back(edge(u,g[v],directed?0:cap))
22 ;
23 g[v]=e.size()-1;
24 }
25 T dfs(int u,int s,int t,T CF=INF){
26 if(u==t)return CF;
27 T tf=CF,df;
28 for(int &i=cur[u];~i;i=e[i].pre){
29 if(e[i].r&&d[u]==d[e[i].v]+1){
30 df=dfs(e[i].v,s,t,min(tf,e[i].r));
31 e[i].r-=df;
32 e[i^1].r+=df;
33 if(!(tf-=df)||d[s]==n)return CF-tf;
34 }
35 }
36 int mh=n;
37 for(int i=cur[u]=g[u];~i;i=e[i].pre){
38 if(e[i].r&&d[e[i].v]<mh)mh=d[e[i].v];
39 }
40 if(!--gap[d[u]])d[s]=n;
41 else ++gap[d[u]]+=mh;
42 return CF-tf;
43 }
44 T isap(int s,int t,bool clean=true){
45 memset(d,0,sizeof(int)*(n+1));
46 memset(gap,0,sizeof(int)*(n+1));
47 memcpy(cur,g,sizeof(int)*(n+1));
48 if(clean) for(size_t i=0;i<e.size();++i)
49 e[i].r=e[i].cap;
50 T MF=0;
51 for(gap[0]=n;d[s]<n;MF+=dfs(s,s,t);
52 return MF;
53 }
54 vector<int> cut_e;//最小割邊集
55 bool vis[MAXN];
56 void dfs_cut(int u){
57 vis[u]=1;//表示u屬於source的最小割集
58 for(int i=g[u];~i;i=e[i].pre)
59 if(e[i].r>0&&vis[e[i].v])dfs_cut(e[i].v);
60 }
61 T min_cut(int s,int t){
62 T ans=isap(s,t);
63 memset(vis,0,sizeof(bool)*(n+1));
64 dfs_cut(s); cut_e.clear();
65 for(int u=0;u<n;++u)if(vis[u])
66 for(int i=g[u];~i;i=e[i].pre)
67 if(!vis[e[i].v])cut_e.push_back(i);
68 return ans;
69 };

```

## 5 Graph

### 5.1 橋連通分量

```

1 vector<pii> findBridges(const vector<vector<
2 int>>& g) {

```

```

2 int n = (int) g.size();
3 vector<int> id(n, -1), low(n);
4 vector<pii> bridges;
5 function<void(int, int)> dfs = [&](int u,
6 int p) {
7 static int cnt = 0;
8 id[u] = low[u] = cnt++;
9 for(auto v : g[u]) {
10 if(v == p) continue;
11 if(id[v] != -1) low[u] = min(low[u],
12 id[v]);
13 else {
14 dfs(v, u);
15 low[u] = min(low[u], low[v]);
16 if(low[v] > id[u]) bridges.emplace_back(u, v);
17 }
18 }
19 };
20 for(int i = 0; i < n; ++i) {
21 if(id[i] == -1) dfs(i, -1);
22 }
23 return bridges;
24 }

```

### 5.2 SPFA

```

1 vector<long long> spfa(vector<vector<pair<
2 int, int>>> G, int S) {
3 int n = G.size(); // 假設點的編號為 0 ~ n-1
4 vector<long long> d(n, INF);
5 vector<bool> in_queue(n, false);
6 vector<int> cnt(n, 0);
7 queue<int> Q;
8 d[S] = 0;
9 auto enqueue = [&](int u) {
10 in_queue[u] = true; Q.emplace(u);
11 };
12 enqueue(S);
13 while (Q.size()) {
14 int u = Q.front();
15 Q.pop();
16 in_queue[u] = false;
17 for (auto [v, cost] : G[u])
18 if (d[v] > d[u] + cost) {
19 if (++cnt[u] >= n) return {}; // 存在負環
20 d[v] = d[u] + cost;
21 if (!in_queue[v]) enqueue(v);
22 }
23 }
24 return d;
25 }

```

### 5.3 manhattan-mst

```

1 void solve(Point *a, int n) {
2 sort(a, a + n, [](const Point &p, const
3 Point &q) {

```

```

3 return p.x + p.y < q.x + q.y;
4 });
5 set<Point> st; // greater<Point::x>
6 for (int i = 0; i < n; ++i) {
7 for (auto it = st.lower_bound(a[i]);
8 it != st.end(); it = st.erase(it)) {
9 if (it->x - it->y < a[i].x -
10 a[i].y) break;
11 es.push_back({it->u, a[i].u,
12 dist(*it, a[i])});
13 }
14 st.insert(a[i]);
15 }
16 void MST(Point *a, int n) {
17 for (int t = 0; t < 2; ++t) {
18 solve(a, n);
19 for (int i = 0; i < n; ++i) swap(a[i].x, a[i].y);
20 solve(a, n);
21 for (int i = 0; i < n; ++i) a[i].x = -a[i].x;
22 }
23 }

```

### 5.4 最大團

```

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

```

### 5.5 判斷平面圖

```

1 //做smoothing,把degree <= 2的點移除
2 //O(n^3)
3 using AdjacencyMatrixTy = vector<vector<bool>
4 >>;
5 AdjacencyMatrixTy smoothing(AdjacencyMatrix
6 &G) {
7 size_t N = G.size(), Change = 0;
8 do {
9 Change = 0;
10 for(size_t u = 0; u < N; ++u) {
11 vector<size_t> E;
12 for(size_t v = 0; v < N && E.size() < 3; ++v)
13 if(G[u][v] && u != v) E.emplace_back(v);
14 if(E.size() == 1 || E.size() == 2) {
15 ++Change;
16 for(auto v : E) G[u][v] = G[v][u] = false;
17 }
18 if(E.size() == 2) {
19 auto [a,b] = make_pair(E[0], E[1]);
20 G[a][b] = G[b][a] = true;
21 }
22 }
23 while(Change);
24 return G;
25 }
26 //計算Degree
27 //O(n^2)
28 vector<size_t> getDegree(const
29 AdjacencyMatrixTy &G) {
30 size_t N = G.size();
31 vector<size_t> Degree(N);
32 for(size_t u = 0; u < N; ++u)
33 for(size_t v = u + 1; v < N; ++v) {
34 if(!G[u][v]) continue;
35 ++Degree[u], ++Degree[v];
36 }
37 return Degree;
38 }
39 //判斷是否為K5 or K3,3
40 //O(n)
41 bool is_K5_or_K33(const vector<size_t> &
42 Degree) {

```

```

42 unordered_map<size_t, size_t> Num;
43 for(auto Val : Degree) ++Num[Val];
44 size_t N = Degree.size();
45 bool isK5 = Num[4] == 5 && Num[4] + Num[0]
    == N;
46 bool isK33 = Num[3] == 6 && Num[3] + Num
    [0] == N;
47 return isK5 || isK33;
48 }

```

## 5.6 count-bridge-online

```

1 vector<int> par, dsu_2ecc, dsu_cc,
    dsu_cc_size, last_visit;
2 int bridges, lca_iteration;
3 void init(int n) {
4     par.assign(n, -1);
5     dsu_2ecc.resize(n);
6     dsu_cc.resize(n);
7     dsu_cc_size.assign(n, 1);
8     lca_iteration = 0;
9     last_visit.assign(n, 0);
10    iota(ALL(dsu_cc), 0);
11    dsu_2ecc = dsu_cc;
12    bridges = 0;
13 }
14 int find_2ecc(int v) {
15     if(v == -1) return -1;
16     return dsu_2ecc[v] == v ? v : dsu_2ecc[v]
        = find_2ecc(dsu_2ecc[v]);
17 }
18 int find_cc(int v) {
19     v = find_2ecc(v);
20     return dsu_cc[v] == v ? v : dsu_cc[v] =
        find_cc(dsu_cc[v]);
21 }
22 void make_root(int v) {
23     v = find_2ecc(v);
24     int root = v, child = -1;
25     while(v != -1) {
26         int p = find_2ecc(par[v]);
27         par[v] = child;
28         dsu_cc[v] = root;
29         child = v;
30         v = p;
31     }
32     dsu_cc_size[root] = dsu_cc_size[child];
33 }
34 void merge_path(int a, int b) {
35     ++lca_iteration;
36     vector<int> path_a, path_b;
37     int lca = -1;
38     while(lca == -1) {
39         if(a != -1) {
40             a = find_2ecc(a);
41             path_a.push_back(a);
42             if(last_visit[a] ==
                lca_iteration){
43                 lca = a;
44                 break;
45             }
46             last_visit[a] = lca_iteration;
47             a = par[a];

```

```

48     }
49     if(b != -1) {
50         b = find_2ecc(b);
51         path_b.push_back(b);
52         if(last_visit[b] ==
            lca_iteration){
53             lca = b;
54             break;
55         }
56         last_visit[b] = lca_iteration;
57         b = par[b];
58     }
59 }
60 }
61 for(int v : path_a) {
62     dsu_2ecc[v] = lca;
63     if(v == lca) break;
64     --bridges;
65 }
66 for(int v : path_b) {
67     dsu_2ecc[v] = lca;
68     if(v == lca) break;
69     --bridges;
70 }
71 }
72 void add_edge(int a, int b) {
73     a = find_2ecc(a), b = find_2ecc(b);
74     if(a == b) return;
75     int ca = find_cc(a), cb = find_cc(b);
76     if(ca != cb) {
77         ++bridges;
78         if(dsu_cc_size[ca] > dsu_cc_size[cb]
            ) swap(a, b), swap(ca, cb);
79         make_root(a);
80         par[a] = dsu_cc[a] = b;
81         dsu_cc_size[cb] += dsu_cc_size[a];
82     } else merge_path(a, b);
83 }

```

## 5.7 雙連通分量&割點

```

1 struct BCC_AP{
2     int dfn_cnt = 0, bcc_cnt = 0, n;
3     vector<int> dfn, low, ap, bcc_id;
4     stack<int> st;
5     vector<bool> vis, is_ap;
6     vector<vector<int>> bcc;
7     BCC_AP(int _n): n(_n){
8         dfn.resize(n+5), low.resize(n+5), bcc.
            resize(n+5), vis.resize(n+5), is_ap.
            resize(n+5), bcc_id.resize(n+5);
9     }
10    inline void build(const vector<vector<int>
        >> &g, int u, int p = -1){
11        int child = 0;
12        dfn[u] = low[u] = ++dfn_cnt;
13        st.push(u);
14        vis[u] = 1;
15        if(g[u].empty() and p == -1){
16            bcc_id[u] = ++bcc_cnt;
17            bcc[bcc_cnt].push_back(u);
18            return;
19        }

```

```

20 for(auto v: g[u]){
21     if(v == p) continue;
22     if(!dfn[v]){
23         build(g, v, u);
24         child++;
25         if(dfn[u] <= low[v]){
26             is_ap[u] = 1;
27             bcc_id[u] = ++bcc_cnt;
28             bcc[bcc_cnt].push_back(u);
29             while(vis[v]){
30                 bcc_id[st.top()] = bcc_cnt;
31                 bcc[bcc_cnt].push_back(st.top());
32             }
33             vis[st.top()] = 0;
34             st.pop();
35         }
36         low[u] = min(low[u], low[v]);
37     }
38     low[u] = min(low[u], dfn[v]);
39 }
40 if(p == -1 and child < 2) is_ap[u] = 0;
41 if(is_ap[u]) ap.push_back(u);
42 }
43 }

```

## 5.8 枚舉極大團 Bron-Kerbosch

```

1 //O(3^n / 3)
2 struct maximalCliques{
3     using Set = vector<int>;
4     size_t n; //1-base
5     vector<Set> G;
6     static Set setUnion(const Set &A, const
        Set &B){
7         Set C(A.size() + B.size());
8         auto it = set_union(A.begin(), A.end(), B.
            begin(), B.end(), C.begin());
9         C.erase(it, C.end());
10        return C;
11    }
12    static Set setIntersection(const Set &A,
        const Set &B){
13        Set C(min(A.size(), B.size()));
14        auto it = set_intersection(A.begin(), A.
            end(), B.begin(), B.end(), C.begin());
15        C.erase(it, C.end());
16        return C;
17    }
18    static Set setDifference(const Set &A,
        const Set &B){
19        Set C(min(A.size(), B.size()));
20        auto it = set_difference(A.begin(), A.end
            (), B.begin(), B.end(), C.begin());
21        C.erase(it, C.end());
22        return C;
23    }
24    void BronKerbosch1(Set R, Set P, Set X){
25        if(P.empty() && X.empty()){
26            // R form an maximal clique
27            return;
28        }
29        for(auto v: P){

```

```

30        BronKerbosch1(setUnion(R, {v}),
            setIntersection(P, G[v]),
            setIntersection(X, G[v]));
31        P = setDifference(P, {v});
32        X = setUnion(X, {v});
33    }
34 }
35 void init(int _n){
36     G.clear();
37     G.resize((n = _n) + 1);
38 }
39 void addEdge(int u, int v){
40     G[u].emplace_back(v);
41     G[v].emplace_back(u);
42 }
43 void solve(int n){
44     Set P;
45     for(int i=1; i<=n; ++i){
46         sort(G[i].begin(), G[i].end());
47         G[i].erase(unique(G[i].begin(), G[i].end()),
            G[i].end());
48         P.emplace_back(i);
49     }
50     BronKerbosch1({}, P, {});
51 }
52 }
53 //判斷圖G是否能3塗色：
54 //枚舉圖G的極大獨立集I (極大獨立集 = 補圖極
    大團)
55 //若存在I使得G-I形成二分圖，則G可以三塗色
56 //反之則不能3塗色
57

```

## 5.9 Floyd Warshall

```

1 int d[100][100];
2 void FloydWarshall(int N){
3     for(int k=0; k<N; ++k)
4         for(int i=0; i<N; ++i)
5             for(int j=0; j<N; ++j)
6                 if(d[i][j] > d[i][k] + d[k][j])
7                     d[i][j] = d[i][k] + d[k][j];
8 }

```

## 5.10 Dominator tree

```

1 struct dominator_tree{
2     static const int MAXN=5005;
3     int n; // 1-base
4     vector<int> G[MAXN], rG[MAXN];
5     int pa[MAXN], dfn[MAXN], id[MAXN], dfnCnt;
6     int semi[MAXN], idom[MAXN], best[MAXN];
7     vector<int> tree[MAXN]; // tree here
8     void init(int _n){
9         n = _n;
10        for(int i=1; i<=n; ++i)
11            G[i].clear(), rG[i].clear();

```

```

12 }
13 void add_edge(int u, int v){
14     G[u].push_back(v);
15     rG[v].push_back(u);
16 }
17 void dfs(int u){
18     id[dfn[u]=++dfnCnt]=u;
19     for(auto v:G[u] if(!dfn[v]))
20         dfs(v,pa[dfn[v]]=dfn[u]);
21 }
22 int find(int y,int x){
23     if(y <= x) return y;
24     int tmp = find(pa[y],x);
25     if(semi[best[y]] > semi[best[pa[y]]])
26         best[y] = best[pa[y]];
27     return pa[y] = tmp;
28 }
29 void tarjan(int root){
30     dfnCnt = 0;
31     for(int i=1; i<=n; ++i){
32         dfn[i] = idom[i] = 0;
33         tree[i].clear();
34         best[i] = semi[i] = i;
35     }
36     dfs(root);
37     for(int i=dfnCnt; i>1; --i){
38         int u = id[i];
39         for(auto v:rG[u] if(v=dfn[v])){
40             find(v,i);
41             semi[i]=min(semi[i],semi[best[v]]);
42         }
43         tree[semi[i]].push_back(i);
44         for(auto v:tree[pa[i]]){
45             find(v, pa[i]);
46             idom[v] = semi[best[v]]==pa[i]
47                 ? pa[i] : best[v];
48         }
49         tree[pa[i]].clear();
50     }
51     for(int i=2; i<=dfnCnt; ++i){
52         if(idom[i] != semi[i])
53             idom[i] = idom[idom[i]];
54         tree[id[idom[i]]].push_back(id[i]);
55     }
56 }
57 }dom;

```

## 5.11 判斷二分圖

```

1 vector<int> G[MAXN];
2 int color[MAXN]; // -1: not colored, 0:
3     black, 1: white
4 /* color the connected component where u is
5     */
6 /* parameter col: the color u should be
7     colored */
8 bool coloring(int u, int col) {
9     if(color[u] != -1) {
10         if(color[u] != col) return false;
11         return true;
12     }
13     color[u] = col;

```

```

12     for(int v : G[u])
13         if(!coloring(v, col ^ 1))
14             return false;
15     return true;
16 }
17 //check if a graph is a bipartite graph
18 bool checkBipartiteG(int n) {
19     for(int i = 1; i <= n; i++)
20         color[i] = -1;
21     for(int i = 1; i <= n; i++)
22         if(color[i] == -1 &&
23             !coloring(i, 0))
24             return false;
25     return true;
26 }

```

## 5.12 Bellman Ford

```

1 vector<tuple<int,int,int>> Edges;
2 int BellmanFord(int s, int e, int N) {
3     const int INF = INT_MAX / 2;
4     vector<int> dist(N, INF);
5
6     dist[s] = 0;
7     bool update;
8     for(int i=1; i<=N; ++i) {
9         update = false;
10        for(auto [v, u, w] : Edges)
11            if (dist[u] > dist[v] + w)
12                {
13                    dist[u] = dist[v] + w;
14                    update = true;
15                }
16    }
17
18    if (!update)
19        break;
20    if (i == N) // && update
21        return -1; // gg !
22
23    return dist[e];
24 }

```

## 5.13 Dijkstra

```

1 int Dijkstra(int s, int e, int N) {
2     const int INF = INT_MAX / 2;
3     vector<int> dist(N, INF);
4     vector<bool> used(N, false);
5
6     using T = tuple<int,int>;
7     priority_queue<T, vector<T>, greater<T>>
8         pq;
9
10    dist[s] = 0;
11    pq.emplace(0, s); // (w, e) 讓 pq 優先用
12        w 來比較

```

```

11 while (!pq.empty()) {
12     tie(std::ignore, s) = pq.top();
13     pq.pop();
14
15     if (used[s]) continue;
16     used[s] = true; // 每一個點都只看一
17         次
18
19     for (auto [e, w] : V[s]) {
20         if (dist[e] > dist[s] + w) {
21             dist[e] = dist[s] + w;
22             pq.emplace(dist[e], e);
23         }
24     }
25 }
26
27 return dist[e];
28 }

```

## 5.14 SCC

```

1 struct SCC{
2     int n,cnt = 0,dfn_cnt = 0;
3     vector<vector<int>>g;
4     vector<int>sz,scc,low,dfn;
5     stack<int>st;
6     vector<bool>vis;
7     SCC(int _n = 0) : n(_n){
8         sz.resize(n+5),scc.resize(n+5),low.
9             resize(n+5),dfn.resize(n+5),vis.
10             resize(n+5);
11         g.resize(n+5);
12     }
13     inline void add_edge(int u, int v){
14         g[u].push_back(v);
15     }
16     inline void build(){
17         function<void(int, int)>dfs = [&](int u,
18             int dis){
19             low[u] = dfn[u] = ++dfn_cnt,vis[u] =
20                 1;
21             st.push(u);
22             for(auto v:g[u]){
23                 if(!dfn[v]){
24                     dfs(v, dis+1);
25                     low[u] = min(low[u],low[v]);
26                 }
27                 else if(vis[v]){
28                     low[u] = min(low[u],dfn[v]);
29                 }
30             }
31             if(low[u]==dfn[u]){
32                 ++cnt;
33                 while(vis[u]){
34                     auto v = st.top();
35                     st.pop();
36                     vis[v] = 0;
37                     scc[v] = cnt;
38                     sz[cnt]++;
39                 }
40             }

```

```

37     };
38     for(int i = 0; i<=n; ++i){
39         if(!scc[i]){
40             dfs(i, 1);
41         }
42     }
43     vector<vector<int>> compress(){
44         vector<vector<int>>ans(cnt+1);
45         for(int u = 0; u<=n; ++u){
46             for(auto v:g[u]){
47                 if(scc[u] == scc[v]){
48                     continue;
49                 }
50                 ans[scc[u]].push_back(scc[v]);
51             }
52         }
53         for(int i = 0; i<=cnt; ++i){
54             sort(ans[i].begin(), ans[i].end());
55             ans[i].erase(unique(ans[i].begin(),
56                 ans[i].end()), ans[i].end());
57         }
58         return ans;
59     }
60 }

```

## 5.15 Minimum Clique Cover

```

1 struct Clique_Cover { // 0-base, O(n2^n)
2     int co[1 << N], n, E[N];
3     int dp[1 << N];
4     void init(int _n) {
5         n = _n, fill_n(dp, 1 << n, 0);
6         fill_n(E, n, 0), fill_n(co, 1 << n, 0);
7     }
8     void add_edge(int u, int v) {
9         E[u] |= 1 << v, E[v] |= 1 << u;
10    }
11    int solve() {
12        for (int i = 0; i < n; ++i)
13            co[1 << i] = E[i] | (1 << i);
14        co[0] = (1 << n) - 1;
15        dp[0] = (n & 1) * 2 - 1;
16        for (int i = 1; i < (1 << n); ++i) {
17            int t = i & -i;
18            dp[i] = -dp[i ^ t];
19            co[i] = co[i ^ t] & co[t];
20        }
21        for (int i = 0; i < (1 << n); ++i)
22            co[i] = (co[i] & i) == i;
23        fwt(co, 1 << n, 1);
24        for (int ans = 1; ans < n; ++ans) {
25            int sum = 0; // probabilistic
26            for (int i = 0; i < (1 << n); ++i)
27                sum += (dp[i] * co[i]);
28            if (sum) return ans;
29        }
30        return n;
31    }
32 }

```

## 5.16 判斷環

```

1 vector<int> G[MAXN];
2 bool visit[MAXN];
3 /* return if the connected component where u
4  is
5  contains a cycle*/
6 bool dfs(int u, int pre) {
7     if(visit[u]) return true;
8     visit[u] = true;
9     for(int v : G[u])
10         if(v != pre && dfs(v, u))
11             return true;
12     return false;
13 }
14
15 //check if a graph contains a cycle
16 bool checkCycle(int n) {
17     for(int i = 1; i <= n; i++)
18         if(!visit[i] && dfs(i, -1))
19             return true;
20     return false;
21 }
22 }

```

## 5.17 2-SAT

```

1 struct two_sat{
2     SCC s;
3     vector<bool>ans;
4     int have_ans = 0;
5     int n;
6     two_sat(int _n) : n(_n) {
7         ans.resize(n+1);
8         s = SCC(2*n);
9     }
10    int inv(int x){
11        if(x>n)return x-n;
12        return x+n;
13    }
14    void add_or_clause(int u, bool x, int v,
15        bool y){
16        if(!x)u = inv(u);
17        if(!y)v = inv(v);
18        s.add_edge(inv(u), v);
19        s.add_edge(inv(v), u);
20    }
21    void check(){
22        if(have_ans!=0)return;
23        s.build();
24        for(int i = 0;i<n;++i){
25            if(s.scc[i]==s.scc[inv(i)]){
26                have_ans = -1;
27                return;
28            }
29            ans[i] = (s.scc[i]<s.scc[inv(i)]);
30        }
31        have_ans = 1;
32    };

```

## 6 Math

### 6.1 Primes

```

1 /* 12721 13331 14341 75577 123457 222557
2    556679 999983 1097774749 1076767633
3    100102021 999997771 1001010013
4    1000512343 987654361 999991231 999888733
5    98789101 987777733 999991921 1010101333
6    1010102101 1000000000039
7    100000000000037 2305843009213693951
8    4611686018427387847 9223372036854775783
9    18446744073709551557 */

```

### 6.2 InvGCD

```

1 pair<long long, long long> inv_gcd(long long
2     a, long long b) {
3     a %= b;
4     if(a < 0) a += b;
5     if(a == 0) return {b, 0};
6     long long s = b, t = a;
7     long long m0 = 0, m1 = 1;
8     while(t) {
9         long long u = s / t;
10        s -= t * u;
11        m0 -= m1 * u;
12        swap(s, t);
13        swap(m0, m1);
14    }
15    if(m0 < 0) m0 += b / s;
16    return {s, m0};

```

### 6.3 LinearCongruence

```

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

```

```

17 return make_pair(lastb<0?lastb+lastm:lastb
18     ,lastm);

```

### 6.4 Bit Set

```

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 }

```

### 6.5 Lucas

```

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

```

### 6.6 ExtendGCD

```

1 // ax + by = gcd(a, b)
2 ll ext_gcd(ll a, ll b, ll& x, ll& y) {
3     if(b == 0) {
4         x = 1, y = 0;
5         return a;
6     }
7     ll x1, y1;
8     ll g = ext_gcd(b, a % b, x1, y1);
9     x = y1, y = x1 - (a / b) * y1;
10    return g;
11 }

```

## 6.7 Basic

```

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

```



```

59 puts("primitive_root NOT FOUND");
60 return -1;
61 }
62 int Legendre(const LL &a, const LL &p) {
63     return modexp(a%p, (p-1)/2, p); }
64
65 LL inv(const LL &a, const LL &n) {
66     LL d, x, y;
67     gcd(a, n, d, x, y);
68     return d==1 ? (x+n)%n : -1;
69 }
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
78 LL log_mod(const LL &a, const LL &b, const
79             LL &p) {
80     // a ^ x = b ( mod p )
81     int m=sqrt(p+.5), e=1;
82     LL v=inv(modexp(a,m,p), p);
83     map<LL, int> x;
84     x[1]=0;
85     for(int i=1; i<m; ++i) {
86         e = LLMul(e, a, p);
87         if(!x.count(e)) x[e] = i;
88     }
89     for(int i=0; i<m; ++i) {
90         if(x.count(b)) return i*m + x[b];
91         b = LLMul(b, v, p);
92     }
93     return -1;
94 }
95
96 LL Tonelli_Shanks(const LL &n, const LL &p)
97 {
98     // x^2 = n ( mod p )
99     if(n==0) return 0;
100     if(Legendre(n,p)!=1) while(1) { puts("SQRT
101         ROOT does not exist"); }
102     int S = 0;
103     LL Q = p-1;
104     while( !(Q&1) ) { Q>>=1; ++S; }
105     if(S==1) return modexp(n%p, (p+1)/4, p);
106     LL z = 2;
107     for(; Legendre(z,p)!=-1; ++z)
108         LL c = modexp(z, Q, p);
109     LL R = modexp(n%p, (Q+1)/2, p), t = modexp(n
110         %p, Q, p);
111     int M = S;
112     while(1) {
113         if(t==1) return R;
114         LL b = modexp(c, 1<<(M-i-1), p);
115         R = LLMul(R, b, p);
116         t = LLMul( LLMul(b, b, p), t, p);
117         c = LLMul(b, b, p);
118         M = i;
119     }
120     return -1;
121 }
122
123 template<typename T>
124 T Euler(T n){

```

```

125     T ans=n;
126     for(T i=2; i*i<=n; ++i){
127         if(n%i==0){
128             ans=ans/i*(i-1);
129             while(n%i==0)n/=i;
130         }
131     }
132     if(n>1)ans=ans/n*(n-1);
133     return ans;
134 }
135
136 //Chinese_remainder_theorem
137 template<typename T>
138 T pow_mod(T n, T k, T m){
139     T ans=1;
140     for(n=(n>m%n%m:n); k;k>>=1){
141         if(k&1)ans=ans*n%m;
142         n=n*n%m;
143     }
144     return ans;
145 }
146
147 template<typename T>
148 T crt(vector<T> &m, vector<T> &a){
149     T M=1, tM, ans=0;
150     for(int i=0; i<(int)m.size(); ++i) M*=m[i];
151     for(int i=0; i<(int)a.size(); ++i){
152         tM=M/m[i];
153         ans=(ans+(a[i]*tM%M)*pow_mod(tM, Euler(m[i])-1, m[i])%M)%M;
154     }
155     /*如果m[i]是質數 · Euler(m[i])-1=m[i]-2 ·
156         就不用算Euler了*/
157     return ans;
158 }

```

## 6.8 Xor-Basis

```

1 template<int B>
2 struct xor_basis {
3     using T = long long;
4     bool zero = false, change = false;
5     int cnt = 0;
6     array<T, B> p = {};
7     vector<T> d;
8     void insert(T x) {
9         IREP(i, B) {
10             if(x >> i & 1) {
11                 if(!p[i]) {
12                     p[i] = x, cnt++;
13                     change = true;
14                     return;
15                 } else x ^= p[i];
16             }
17         }
18         if(!zero) zero = change = true;
19     }
20     T get_min() {
21         if(zero) return 0;
22         REP(i, B) if(p[i]) return p[i];
23     }
24     T get_max() {
25         T ans = 0;

```

```

26         IREP(i, B) ans = max(ans, ans ^ p[i]);
27         return ans;
28     }
29     T get_kth(long long k) {
30         k++;
31         if(k == 1 && zero) return 0;
32         k -= zero;
33         if(k >= (1LL << cnt)) return -1;
34         update();
35         T ans = 0;
36         REP(i, SZ(d)) if(k >> i & 1) ans ^= d[i]
37             ];
38         return ans;
39     }
40     bool contains(T x) {
41         if(x == 0) return zero;
42         IREP(i, B) if(x >> i & 1) x ^= p[i];
43         return x == 0;
44     }
45     void merge(const xor_basis& other) { REP(i
46         , B) if(other.p[i]) insert(other.p[i])
47         ; }
48     void update() {
49         if(!change) return;
50         change = false;
51         d.clear();
52         REP(j, B) IREP(i, j) if(p[j] >> i & 1) p
53             [j] ^= p[i];
54         REP(i, B) if(p[i]) d.pb(p[i]);
55     }
56 };

```

## 6.9 Theorem

- Modular Arithmetic

$$(a + b) \bmod m = (a \bmod m + b \bmod m) \bmod m$$

$$(a - b) \bmod m = (a \bmod m - b \bmod m) \bmod m$$

$$(a \cdot b) \bmod m = ((a \bmod m) \cdot (b \bmod m)) \bmod m$$

$$a^b \bmod m = (a \bmod m)^{b \bmod m-1} \bmod m$$

- Cramer's rule

$$\begin{aligned} ax + by &= e \\ cx + dy &= f \end{aligned} \Rightarrow \begin{aligned} x &= \frac{ed - bf}{ad - bc} \\ y &= \frac{af - ec}{ad - bc} \end{aligned}$$

- Kirchhoff's Theorem

Denote  $L$  be a  $n \times n$  matrix as the Laplacian matrix of graph  $G$ , where  $L_{ii} = d(i)$ ,  $L_{ij} = -c$  where  $c$  is the number of edge  $(i, j)$  in  $G$ .

- The number of undirected spanning in  $G$  is  $|\det(\tilde{L}_{11})|$ .
- The number of directed spanning tree rooted at  $r$  in  $G$  is  $|\det(\tilde{L}_{rr})|$ .

- Tutte's Matrix

Let  $D$  be a  $n \times n$  matrix, where  $d_{ij} = x_{ij}$  ( $x_{ij}$  is chosen uniformly at random) if  $i < j$  and  $(i, j) \in E$ , otherwise  $d_{ij} = -d_{ji}$ .  $\frac{\text{rank}(D)}{2}$  is the maximum matching on  $G$ .

- Cayley's Formula

- Given a degree sequence  $d_1, d_2, \dots, d_n$  for each labeled vertices, there are  $\frac{(d_1-1)!(d_2-1)! \cdots (d_n-1)!}{(n-2)!}$  spanning trees.
- Let  $T_{n,k}$  be the number of labeled forests on  $n$  vertices with  $k$  components, such that vertex  $1, 2, \dots, k$  belong to different components. Then  $T_{n,k} = kn^{n-k-1}$ .

- Erdős–Gallai theorem

A sequence of nonnegative integers  $d_1 \geq \dots \geq d_n$  can be represented as the degree sequence of a finite simple graph on  $n$  vertices if and only if  $d_1 + \dots + d_n$  is even

$$\text{and } \sum_{i=1}^k d_i \leq k(k-1) + \sum_{i=k+1}^n \min(d_i, k) \text{ holds for every } 1 \leq k \leq n.$$

- Gale–Ryser theorem

A pair of sequences of nonnegative integers  $a_1 \geq \dots \geq a_n$  and  $b_1, \dots, b_n$  is bigraphic if and only if

$$\sum_{i=1}^n a_i = \sum_{i=1}^n b_i \text{ and } \sum_{i=1}^k a_i \leq \sum_{i=1}^n \min(b_i, k) \text{ holds for every } 1 \leq k \leq n.$$

- Fulkerson–Chen–Anstee theorem

A sequence  $(a_1, b_1), \dots, (a_n, b_n)$  of nonnegative integer pairs with  $a_1 \geq \dots \geq a_n$  is digraphic if and only if

$$\text{if } \sum_{i=1}^n a_i = \sum_{i=1}^n b_i \text{ and } \sum_{i=1}^k a_i \leq \sum_{i=1}^k \min(b_i, k-1) + \sum_{i=k+1}^n \min(b_i, k) \text{ holds for every } 1 \leq k \leq n.$$

- $M\Box$ bius inversion formula

$$\begin{aligned} -f(n) &= \sum_{d|n} g(d) \Leftrightarrow g(n) = \sum_{d|n} \mu(d) f\left(\frac{n}{d}\right) \\ -f(n) &= \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d} \mu\left(\frac{d}{n}\right) f(d) \end{aligned}$$

- Spherical cap

- A portion of a sphere cut off by a plane.
- $r$ : sphere radius,  $a$ : radius of the base of the cap,  $h$ : height of the cap,  $\theta$ :  $\arcsin(a/r)$ .
- Volume  $= \pi h^2(3r - h)/3 = \pi h(3a^2 + h^2)/6 = \pi r^3(2 + \cos\theta)(1 - \cos\theta)^2/3$ .
- Area  $= 2\pi r h = \pi(a^2 + h^2) = 2\pi r^2(1 - \cos\theta)$ .

## 6.10 Pisano number

```

1 // pisano number: 費氏數列 mod m
2 // 情況下多長會循環
3 // Can be proved under O(6m)
4 ll find_pisano(ll m) {
5     ll a = 0, b = 1, c;
6     for(i=0;;i++) {
7         c = (a+b) % m;
8         a = b; b = c;
9         if(!a && b == 1)
10            return i+1;
11     }
12 }

```

## 6.11 Matrix

```

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

```

```

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

```

## 6.12 Numbers

- Bernoulli numbers

$$B_0 = 1, B_1^\pm = \pm \frac{1}{2}, B_2 = \frac{1}{6}, B_3 = 0$$

$$\sum_{j=0}^m \binom{m+1}{j} B_j = 0, \text{EGF is } B(x) = \frac{x}{e^x - 1}$$

$$\sum_{n=0}^{\infty} B_n \frac{x^n}{n!}$$

$$S_m(n) = \sum_{k=1}^n k^m$$

$$\frac{1}{m+1} \sum_{k=0}^m \binom{m+1}{k} B_k^+ n^{m+1-k}$$

- Stirling numbers of the second kind Partitions of  $n$  distinct elements into exactly  $k$  groups.

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

$$S(n, k) = \frac{1}{k!} \sum_{i=0}^k (-1)^{k-i} \binom{k}{i} i^n$$

$$x^n = \sum_{i=0}^n S(n, i) (x)_i$$

- Pentagonal number theorem

$$\prod_{n=1}^{\infty} (1 - x^n) = 1 + \sum_{k=1}^{\infty} (-1)^k \left( x^{k(3k+1)/2} + x^{k(3k-1)/2} \right)$$

- Catalan numbers

$$C_n^{(k)} = \frac{1}{(k-1)n+1} \binom{kn}{n}$$

$$C^{(k)}(x) = 1 + x[C^{(k)}(x)]^k$$

- Eulerian numbers

Number of permutations  $\pi \in S_n$  in which exactly  $k$  elements are greater than the previous element.  $k$   $j$ :s s.t.  $\pi(j) > \pi(j+1)$ ,  $k+1$   $j$ :s s.t.  $\pi(j) \geq j$ ,  $k$   $j$ :s s.t.  $\pi(j) > j$ .

$$E(n, k) = (n-k)E(n-1, k-1) + (k+1)E(n-1, k)$$

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

$$E(n, k) = \sum_{j=0}^k (-1)^j \binom{n+1}{j} (k+1-j)^n$$

## 6.13 Triangle

```

1 // Counts x, y >= 0 such that Ax + By <= C.
2 // Requires A, B > 0. Runs in Log time.
3 // Also representable as sum_{0 <= x <= C / A} floor((C - Ax) / B + 1).
4 ll count_triangle(ll A, ll B, ll C) {
5     if(C < 0) return 0;
6     if(A < B) swap(A, B);
7     ll m = C / A, k = A / B;
8     ll h = (C - m * A) / B + 1;
9     return m * (m + 1) / 2 * k + (m + 1) * h
10        + count_triangle(B, A - k * B, C - B * (k * m + h));
11 }
12 // Counts 0 <= x < RA, 0 <= y < RB such that
13 // Ax + By <= C. Requires A, B > 0.
14 ll count_triangle_rectangle_intersection(ll A, ll B, ll C, ll RA, ll RB) {
15     if(C < 0 || RA <= 0 || RB <= 0) return 0;
16     if(C >= A * (RA - 1) + B * (RB - 1))
17         return RA * RB;
18     return count_triangle(A, B, C) -
19        count_triangle(A, B, C - A * RA) -
20        count_triangle(A, B, C - B * RB);

```

## 6.14 GCD-Convolution

```

1 // 2, 3, 5, 7, ...
2 vector<int> prime_enumerate(int N) {
3     vector<bool> sieve(N / 3 + 1, 1);
4     for(int p = 5, d = 4, i = 1, sqn = sqrt(N); p <= sqn; p += d = 6 - d, i++) {

```

```

5         if(!sieve[i]) continue;
6         for(int q = p * p / 3, r = d * p / 3 + (d * p % 3 == 2), s = 2 * p; q < SZ(sieve); q += r = s - r) sieve[q] = 0;
7     }
8     vector<int> ret{2, 3};
9     for(int p = 5, d = 4, i = 1; p <= N; p += d = 6 - d, i++) {
10         if(sieve[i]) {
11             ret.pb(p);
12         }
13     }
14     while(SZ(ret) && ret.back() > N) ret.pop_back();
15     return ret;
16 }
17 struct divisor_transform {
18     template<class T>
19     static void zeta_transform(vector<T>& a) {
20         int n = a.size() - 1;
21         for(auto p : prime_enumerate(n)) {
22             for(int i = 1; i * p <= n; i++) {
23                 a[i * p] += a[i];
24             }
25         }
26     }
27     template<class T>
28     static void mobius_transform(vector<T>& a) {
29         int n = a.size() - 1;
30         for(auto p : prime_enumerate(n)) {
31             for(int i = n / p; i > 0; i--) {
32                 a[i * p] -= a[i];
33             }
34         }
35     }
36 };
37 struct multiple_transform {
38     template<class T>
39     static void zeta_transform(vector<T>& a) {
40         int n = a.size() - 1;
41         for(auto p : prime_enumerate(n)) {
42             for(int i = n / p; i > 0; i--) {
43                 a[i] += a[i * p];
44             }
45         }
46     }
47     template<class T>
48     static void mobius_transform(vector<T>& a) {
49         int n = a.size() - 1;
50         for(auto p : prime_enumerate(n)) {
51             for(int i = 1; i * p <= n; i++) {
52                 a[i] -= a[i * p];
53             }
54         }
55     }
56 };
57 // lcm: multiple -> divisor
58 template<class T>
59 vector<T> gcd_convolution(const vector<T>& a, const vector<T>& b) {
60     assert(a.size() == b.size());
61     auto f = a, g = b;
62     multiple_transform::zeta_transform(f);

```

```

63 multiple_transform::zeta_transform(g);
64 REP(i, SZ(f)) f[i] *= g[i];
65 multiple_transform::mobius_transform(f);
66 return f;
67 }

```

## 6.15 Combination

```

1 mint binom(int n, int k) {
2     if(k < 0 || k > n) return 0;
3     return fact[n] * inv_fact[k] * inv_fact[n
4         - k];
5 }
6 // a_1 + a_2 + ... + a_n = k, a_i >= 0
7 mint stars_and_bars(int n, int k) { return
8     binom(k + n - 1, n - 1); }
9 // number of ways from (0, 0) to (n, m)
10 mint paths(int n, int m) { return binom(n +
11     m, n); }
12 mint catalan(int n) { return binom(2 * n, n)
13     - binom(2 * n, n + 1); }

```

## 6.16 FWT

```

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

```

## 6.17 找實根

```

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 }

```

```

1 vector<double> cal(vector<double>coef, int n
2     ){
3     vector<double>res;
4     if(n == 1){
5         if(sign(coef[1])) res.pb(-coef[0]/coef
6             [1]);
7         return res;
8     }
9     vector<double>dcoef(n);
10    for(int i = 0; i < n; ++i) dcoef[i] = coef
11        [i+1]*(i+1);
12    vector<double>droot = cal(dcoef, n-1);
13    droot.insert(droot.begin(), -INF);
14    droot.pb(INF);
15    for(int i = 0; i+1 < droot.size(); ++i){
16        double tmp = find(coef, n, droot[i],
17            droot[i+1]);
18        if(tmp < INF) res.pb(tmp);
19    }
20    return res;
21 }

```

```

1 int main () {
2     vector<double>ve;
3     vector<double>ans = cal(ve, n);
4     // 視情況把答案 +eps · 避免 -0
5 }

```

## 6.18 NTT

```

1 const ll mod = (119 << 23) + 1, root = 62;
2 // = 998244353

```

```

1 // For p < 2^30 there is also e.g. 5 << 25,
2     7 << 26, 479 << 21
3 // and 483 << 21 (same root). The last two
4     are > 10^9.
5 typedef vector<ll> vl;
6 void ntt(vl &a) {
7     int n = SZ(a), L = 31 - __builtin_clz(n);
8     static vl rt(2, 1);
9     for(static int k = 2, s = 2; k < n; k *=
10         2, s++) {
11         rt.resize(n);
12         ll z[] = {1, mod_pow(root, mod >> s, mod
13             )};
14         FOR(i, k, 2 * k) rt[i] = rt[i / 2] * z[i
15             & 1] % mod;
16     }
17     vi rev(n);
18     REP(i, n) rev[i] = (rev[i / 2] | (i & 1)
19         << L) / 2;
20     REP(i, n) if (i < rev[i]) swap(a[i], a[rev
21         [i]]);
22     for(int k = 1; k < n; k *= 2)
23         for(int i = 0; i < n; i += 2 * k) REP(j,
24             k) {
25             ll z = rt[j + k] * a[i + j + k] % mod,
26                 &ai = a[i + j];
27             a[i + j + k] = ai - z + (z > ai ? mod
28                 : 0);
29             ai += (ai + z >= mod ? z - mod : z);
30         }
31     }
32 }
33 vl conv(const vl &a, const vl &b) {
34     if(a.empty() || b.empty()) return {};
35     int s = SZ(a) + SZ(b) - 1, B = 32 -
36         __builtin_clz(s), n = 1 << B;
37     ll inv = mod_pow(n, mod - 2, mod);
38     vl L(a), R(b), out(n);
39     L.resize(n), R.resize(n);
40     ntt(L), ntt(R);
41     REP(i, n) out[-i & (n - 1)] = inv * L[i] %
42         mod * R[i] % mod;
43     ntt(out);
44     return {out.begin(), out.begin() + s};
45 }

```

## 6.19 FFT

```

1 // Fast-Fourier-Transform
2 using cd = complex<double>;
3 const double PI = acos(-1);
4
5 void FFT(vector<cd>&a, bool inv) {
6     int n = (int) a.size();
7     for(int i = 1, j = 0; i < n; ++i) {
8         int bit = n >> 1;
9         for(; j < bit; bit >= 1) {
10             j ^= bit;
11         }
12         j ^= bit;
13         if(i < j) {
14             swap(a[i], a[j]);
15         }
16     }

```

```

17 for(int len = 2; len <= n; len <= 1) {
18     const double ang = 2 * PI / len * (inv ?
19         -1 : +1);
20     cd rot(cos(ang), sin(ang));
21     for(int i = 0; i < n; i += len) {
22         cd w(1);
23         for(int j = 0; j < len / 2; ++j) {
24             cd u = a[i + j], v = a[i + j + len /
25                 2] * w;
26             a[i + j] = u + v;
27             a[i + j + len / 2] = u - v;
28             w *= rot;
29         }
30     }
31     if(inv) {
32         for(auto& x : a) {
33             x /= n;
34         }
35     }
36 }
37 vector<int> multiply(const vector<int>&a,
38     const vector<int>&b) {
39     vector<cd> fa(a.begin(), a.end());
40     vector<cd> fb(b.begin(), b.end());
41     int n = 1;
42     while(n < (int) a.size() + (int) b.size()
43         - 1) {
44         n <<= 1;
45     }
46     fa.resize(n);
47     fb.resize(n);
48     FFT(fa, false);
49     FFT(fb, false);
50     for(int i = 0; i < n; ++i) {
51         fa[i] *= fb[i];
52     }
53     FFT(fa, true);
54     vector<int> c(a.size() + b.size() - 1);
55     for(int i = 0; i < (int) c.size(); ++i) {
56         c[i] = round(fa[i].real());
57     }
58     return c;
59 }

```

## 6.20 Gauss-Jordan

```

1 int GaussJordan(vector<vector<ld>>&a) {
2     // -1 no sol, 0 inf sol
3     int n = SZ(a);
4     REP(i, n) assert(SZ(a[i]) == n + 1);
5     REP(i, n) {
6         int p = i;
7         REP(j, n) {
8             if(j < i && abs(a[j][j]) > EPS)
9                 continue;
10             if(abs(a[j][i]) > abs(a[p][i])) p = j;
11         }
12         REP(j, n + 1) swap(a[i][j], a[p][j]);
13         if(abs(a[i][i]) <= EPS) continue;
14         REP(j, n) {
15             if(i == j) continue;

```

```

15     ld delta = a[j][i] / a[i][i];
16     FOR(k, i, n + 1) a[j][k] -= delta * a[i][k];
17 }
18 }
19 bool ok = true;
20 REP(i, n) {
21     if(abs(a[i][i]) <= EPS) {
22         if(abs(a[i][n]) > EPS) return -1;
23         ok = false;
24     }
25 }
26 return ok;
27 }

```

## 6.21 Pollard-Rho

```

1 #define ull unsigned long long
2 #define ldb long double
3
4 vector<ll> factor;
5 vector<pair<ll,ll>> fac;
6
7 ll fpow(ll x, ll y, ll p) {
8     ll res = 1;
9     while (y) {
10         if (y & 1) res = (__int128)res * x % p;
11         x = (__int128)x * x % p;
12         y >>= 1;
13     }
14     return res;
15 }
16
17 bool mr(ll x, ll p) {
18     if (fpow(x, p - 1, p) != 1) return 0;
19     ll y = p - 1, z;
20     while (!(y & 1)) {
21         y >>= 1;
22         z = fpow(x, y, p);
23         if (z != 1 && z != p - 1) return 0;
24         if (z == p - 1) return 1;
25     }
26     return 1;
27 }
28
29 // Miller Rabin ~O(log p)
30 bool is_prime(ll p) {
31     if (p < 2) return 1;
32     if (p==2 || p==3 || p==5 || p==7 || p==43) return 1;
33     return mr(2,p) && mr(3,p) && mr(5,p) && mr(7,p) && mr(43,p);
34 }
35
36 // O(1) 快速乘(防LL overflow)
37 ll ksc(ull x, ull y, ll p) {
38     return (x*y-(ull)((ldb)x/p*y)*p+p)%p;
39 }
40
41 //求n任一真因数(需保证n非质数) O(n^1/4)
42 ll pollar_rho(ll n) {
43     ll x,y,z,c,g,i,j;

```

```

44 while(1) {
45     x = y = rand()%n;
46     z = 1;
47     c = rand()%n;
48     i = 0, j = 1;
49     while(++i) {
50         x = (ksc(x,x,n) + c)%n;
51         z = ksc(z,abs(y-x),n);
52         if(x == y || !z) break;
53         if(!(i%127) || i == j) {
54             g = __gcd(z,n);
55             if(g > 1) return g;
56             if(i == j) y = x, j <= 1;
57         }
58     }
59 }
60 }
61
62 void factorization(ll n) {
63     while(!is_prime(n)) {
64         ll f = pollar_rho(n);
65         while(!is_prime(f)) {
66             f = pollar_rho(f);
67         }
68         ll cou = 0;
69         while(n%f == 0) n /= f, cou++;
70         fac.push_back({f,cou});
71     }
72     if(n != 1) fac.push_back({n,1});
73 }
74
75 void get_factors(ll now, ll cou) {
76     if(now >= fac.size()) {
77         factor.push_back(cou);
78         return;
79     }
80     get_factors(now+1,cou);
81     for(ll i=1;i<=fac[now].second;i++) {
82         cou *= fac[now].first;
83         get_factors(now+1,cou);
84     }
85 }

```

## 7 Square root decomposition

### 7.1 MoAlgo

```

1 struct qry{
2     int ql,qr,id;
3 };
4 template<class T>struct Mo{
5     int n,m;
6     vector<pii>ans;
7     Mo(int _n,int _m): n(_n),m(_m){
8         ans.resize(m);
9     }
10    void solve(vector<T>&v,vector<qry>&q){
11        int l = 0, r = -1;
12        vector<int>cnt,cntcnt;
13        cnt.resize(n+5);
14        cntcnt.resize(n+5);

```

```

15    int mx = 0;
16    function<void(int)>add = [&](int pos){
17        cntcnt[cnt[v[pos]]]--;
18        cnt[v[pos]]++;
19        cntcnt[cnt[v[pos]]]++;
20        mx = max(mx,cnt[v[pos]]);
21    };
22    function<void(int)>sub = [&](int pos){
23        if(--cntcnt[cnt[v[pos]]] and cnt[v[pos]]==mx)mx--;
24        cnt[v[pos]]--;
25        cntcnt[cnt[v[pos]]]++;
26        mx = max(mx,cnt[v[pos]]);
27    };
28    sort(all(q),[&](qry a,qry b){
29        static int B = max((int)1,n/max((int)sqrt(m),(int)1));
30        if(a.ql/B!=b.ql/B)return a.ql<b.ql;
31        if((a.ql/B)&1)return a.qr>b.qr;
32        return a.qr<b.qr;
33    });
34    for(auto [ql,qr,id]:q){
35        while(l>ql)add(--l);
36        while(r<qr)add(++r);
37        while(l<ql)sub(l--);
38        while(r>qr)sub(r--);
39        ans[id] = {mx,cntcnt[mx]};
40    }
41 }
42 };

```

### 7.2 莫隊

```

1 void remove(idx); // TODO: remove value at
   idx from data structure
2 void add(idx); // TODO: add value at idx
   from data structure
3 int get_answer(); // TODO: extract the
   current answer of the data structure
4
5 int block_size;
6
7 struct Query {
8     int l, r, idx;
9     bool operator<(Query other) const {
10         return make_pair(l / block_size, r)
11             < make_pair(other.l / block_size, other.r);
12     }
13 }
14 };
15
16 vector<int> mo_s_algorithm(vector<Query>
   queries) {
17     vector<int> answers(queries.size());
18     sort(queries.begin(), queries.end());
19
20     // TODO: initialize data structure
21
22     int cur_l = 0;
23     int cur_r = -1;

```

```

24 // invariant: data structure will always
   reflect the range [cur_l, cur_r]
25 for (Query q : queries) {
26     while (cur_l > q.l) {
27         cur_l--;
28         add(cur_l);
29     }
30     while (cur_r < q.r) {
31         cur_r++;
32         add(cur_r);
33     }
34     while (cur_l < q.l) {
35         remove(cur_l);
36         cur_l++;
37     }
38     while (cur_r > q.r) {
39         remove(cur_r);
40         cur_r--;
41     }
42     answers[q.idx] = get_answer();
43 }
44 return answers;
45 }

```

### 7.3 分塊 cf455D

```

1 const ll block_siz = 320;
2 const ll maxn = 100005;
3 ll a[maxn];
4 ll cnt[block_siz+1][maxn]; // i-th block, k'
   s cou
5 deque<ll> q[block_siz+1];
6
7 void print_all(ll n)
8 {
9     for(int i=0;i<n;i++)
10     {
11         cout << q[i/block_siz][i-i/block_siz
12             *block_siz] << ' ';
13     }
14     cout << endl << endl;
15 }
16
17 int main()
18 {
19     Crbubble
20     ll n,m,i,k,t;
21     ll l,r,ord,pre,id,id2, ans = 0;
22     cin >> n;
23     for(i=0;i<n;i++)
24     {
25         cin >> a[i];
26         id = i/block_siz;
27         q[id].push_back(a[i]);
28         cnt[id][a[i]]++;
29     }
30     cin >> t;
31     while(t--)
32     {
33         cin >> ord >> l >> r;
34         l = (l+ans-1)%n+1 -1;
35         r = (r+ans-1)%n+1 -1;
36         if(l > r) swap(l,r);
37         id = l/block_siz; l %= block_siz;

```

## 7.4 Mos Algorithm On Tree

```

36 id2 = r/block_siz; r %= block_siz;
37 if(ord == 1)
38 {
39     if(id == id2)
40     {
41         pre = q[id][r];
42         for(i=r;i>l;i--)
43         {
44             q[id][i] = q[id][i-1];
45         }
46         q[id][l] = pre;
47     }
48     else
49     {
50         pre = q[id].back();
51         cnt[id][pre]--;
52         q[id].pop_back();
53
54         for(i=id+1;i<id2;i++)
55         {
56             q[i].push_front(pre);
57             cnt[i][pre]++;
58             pre = q[i].back();
59             cnt[i][pre]--;
60             q[i].pop_back();
61         }
62         q[id2].push_front(pre);
63         cnt[id2][pre]++;
64         pre = q[id2][r+1];
65         cnt[id2][pre]--;
66         q[id2].erase(q[id2].begin()+
67             r+1);
68
69         q[id].insert(q[id].begin()+1
70             , pre);
71         cnt[id][pre]++;
72         //print_all(n);
73     }
74     else
75     {
76         cin >> m;
77         m = (m+ans-1)%n+1;
78         ans = 0;
79         if(id == id2)
80         {
81             for(i=l;i<=r;i++) ans += (q[
82                 id][i] == m);
83         }
84         else
85         {
86             for(i=l;i<block_siz;i++) ans
87                 += (q[id][i] == m);
88             for(i=0;i<=r;i++) ans += (q[
89                 id2][i] == m);
90             for(i=id+1;i<id2;i++) ans +=
91                 cnt[i][m];
92         }
93         cout << ans << endl;
94     }
95 }
96 return 0;
97 }

```

```

1  /*
2  Mo's Algorithm On Tree
3  Preprocess:
4  1) LCA
5  2) dfs with in[u] = dft++, out[u] = dft++
6  3) ord[in[u]] = ord[out[u]] = u
7  4) bitset<MAXN> inset
8  */
9  struct Query {
10     int L, R, LBid, lca;
11     Query(int u, int v) {
12         int c = LCA(u, v);
13         if (c == u || c == v)
14             q.lca = -1, q.L = out[c ^ u ^ v], q.R
15                 = out[c];
16         else if (out[u] < in[v])
17             q.lca = c, q.L = out[u], q.R = in[v];
18         else
19             q.lca = c, q.L = out[v], q.R = in[u];
20         q.Lid = q.L / blk;
21     }
22     bool operator<(const Query &q) const {
23         if (LBid != q.LBid) return LBid < q.LBid;
24         ;
25         return R < q.R;
26     }
27     void flip(int x) {
28         if (inset[x]) sub(arr[x]); // TODO
29         else add(arr[x]); // TODO
30         inset[x] = ~inset[x];
31     }
32     void solve(vector<Query> query) {
33         sort(ALL(query));
34         int L = 0, R = 0;
35         for (auto q : query) {
36             while (R < q.R) flip(ord[++R]);
37             while (L > q.L) flip(ord[--L]);
38             while (R > q.R) flip(ord[R--]);
39             while (L < q.L) flip(ord[L++]);
40             if (~q.lca) add(arr[q.lca]);
41             // answer query
42             if (~q.lca) sub(arr[q.lca]);
43         }
44     }

```

## 8 Tree

### 8.1 Tree centroid

```

1  //找出其中一個樹重心
2  vector<int> size;
3
4  int ans = -1;
5  void dfs(int u, int parent = -1) {
6      size[u] = 1;
7      int max_son_size = 0;
8      for (auto v : Tree[u]) {

```

```

9      if (v == parent) continue;
10     dfs(v, u);
11     size[u] += size[v];
12     max_son_size = max(max_son_size, size[v]);
13 }
14 max_son_size = max(max_son_size, n - size[u]);
15 if (max_son_size <= n / 2) ans = u;
16 }

```

### 8.2 HLD

```

1 struct heavy_light_decomposition{
2     int n;
3     vector<int> dep, father, sz, mxson, topf, id;
4     vector<vector<int>>> g;
5     heavy_light_decomposition(int _n = 0) : n(_n) {
6         g.resize(n+5);
7         dep.resize(n+5);
8         father.resize(n+5);
9         sz.resize(n+5);
10        mxson.resize(n+5);
11        topf.resize(n+5);
12        id.resize(n+5);
13    }
14    void add_edge(int u, int v){
15        g[u].push_back(v);
16        g[v].push_back(u);
17    }
18    void dfs(int u, int p){
19        dep[u] = dep[p]+1;
20        father[u] = p;
21        sz[u] = 1;
22        mxson[u] = 0;
23        for(auto v:g[u]){
24            if(v==p) continue;
25            dfs(v, u);
26            sz[u] += sz[v];
27            if(sz[v] > sz[mxson[u]]) mxson[u] = v;
28        }
29    }
30    void dfs2(int u, int top){
31        static int idn = 0;
32        topf[u] = top;
33        id[u] = ++idn;
34        if(mxson[u]) dfs2(mxson[u], top);
35        for(auto v:g[u]){
36            if(v!=father[u] and v!=mxson[u]){
37                dfs2(v, v);
38            }
39        }
40    }
41    void build(int root){
42        dfs(root, 0);
43        dfs2(root, root);
44    }
45    vector<pair<int, int>> path(int u, int v){
46        vector<pair<int, int>> ans;
47        while(topf[u] != topf[v]){
48            if(dep[topf[u]] < dep[topf[v]]) swap(u, v);

```

```

49        ans.push_back({id[topf[u]], id[u]});
50        u = father[topf[u]];
51    }
52    if(id[u] > id[v]) swap(u, v);
53    ans.push_back({id[u], id[v]});
54    return ans;
55 }
56 }

```

### 8.3 HeavyLight

```

1 #include<vector>
2 #define MAXN 100005
3 int siz[MAXN], max_son[MAXN], pa[MAXN], dep[
4     MAXN];
5 int link_top[MAXN], link[MAXN], cnt;
6 vector<int> G[MAXN];
7 void find_max_son(int u){
8     siz[u]=1;
9     max_son[u]=-1;
10    for(auto v:G[u]){
11        if(v==pa[u]) continue;
12        pa[v]=u;
13        dep[v]=dep[u]+1;
14        find_max_son(v);
15        if(max_son[u]==-1 || siz[v]>siz[max_son[u]]
16            ) max_son[u]=v;
17        siz[u] += siz[v];
18    }
19 }
20 void build_link(int u, int top){
21     link[u] = ++cnt;
22     link_top[u] = top;
23     if(max_son[u] == -1) return;
24     build_link(max_son[u], top);
25     for(auto v:G[u]){
26         if(v==max_son[u] || v==pa[u]) continue;
27         build_link(v, v);
28     }
29 }
30 int find_lca(int a, int b){
31     //求LCA · 可以在過程中對區間進行處理
32     int ta=link_top[a], tb=link_top[b];
33     while(ta!=tb){
34         if(dep[ta]<dep[tb]){
35             swap(ta, tb);
36             swap(a, b);
37         }
38         //這裡可以對a所在的鏈做區間處理
39         //區間為(Link[ta], Link[a])
40         ta=link_top[a=pa[ta]];
41     }
42     //最後a,b會在同一條鏈 · 若a!=b還要在進行一
43     //次區間處理
44     return dep[a]<dep[b]?a:b;
45 }

```

### 8.4 centroidDecomposition



## 8.5 link cut tree

```

1 vector<vector<int>>>g;
2 vector<int>sz,tmp;
3 vector<bool>vis; //visit_centroid
4 int tree_centroid(int u,int n){
5     function<void(int,int)>dfs1 = [&](int u,
6         int p){
7         sz[u] = 1;
8         for(auto v:g[u]){
9             if(v==p)continue;
10            if(vis[v])continue;
11            dfs1(v,u);
12            sz[u]+=sz[v];
13        }
14    };
15    function<int(int,int)>dfs2 = [&](int u,int
16        p){
17        for(auto v:g[u]){
18            if(v==p)continue;
19            if(vis[v])continue;
20            if(sz[v]*2<n)continue;
21            return dfs2(v,u);
22        }
23        return u;
24    };
25    dfs1(u,-1);
26    return dfs2(u,-1);
27 }
28 int cal(int u,int p = -1,int deep = 1){
29     int ans = 0;
30     tmp.pb(deep);
31     sz[u] = 1;
32     for(auto v:g[u]){
33         if(v==p)continue;
34         if(vis[v])continue;
35         ans+=cal(v,u,deep+1);
36         sz[u]+=sz[v];
37     }
38     //calcuat the answer
39     return ans;
40 }
41 int centroid_decomposition(int u,int
42     tree_size){
43     int center = tree_centroid(u,tree_size);
44     vis[center] = 1;
45     int ans = 0;
46     for(auto v:g[center]){
47         if(vis[v])continue;
48         ans+=cal(v);
49         for(int i = sz(tmp)-sz[v];i<sz(tmp);++i)
50             //update
51     }
52     while(!tmp.empty()){
53         //roll_back(tmp.back())
54         tmp.pop_back();
55     }
56     for(auto v:g[center]){
57         if(vis[v])continue;
58         ans+=centroid_decomposition(v,sz[v]);
59     }
60     return ans;
61 }

```

```

1 struct splay_tree{
2     int ch[2],pa; //子節點跟父母
3     bool rev; //反轉的懶惰標記
4     splay_tree():pa(0),rev(0){ch[0]=ch[1]=0;}
5 };
6 vector<splay_tree> nd;
7 //有的時候用vector會TLE，要注意
8 //這邊以node[0]作為null節點
9 bool isroot(int x){ //判斷是否為這棵splay
10     tree的根
11     return nd[nd[x].pa].ch[0]!=x&&nd[nd[x].pa]
12         .ch[1]!=x;
13 }
14 void down(int x){ //懶惰標記下推
15     if(nd[x].rev){
16         if(nd[x].ch[0]nd[nd[x].ch[0]].rev^=1;
17         if(nd[x].ch[1]nd[nd[x].ch[1]].rev^=1;
18         swap(nd[x].ch[0],nd[x].ch[1]);
19         nd[x].rev=0;
20     }
21 }
22 void push_down(int x){ //所有祖先懶惰標記下推
23     if(!isroot(x))push_down(nd[x].pa);
24     down(x);
25 }
26 void up(int x){ //將子節點的資訊向上更新
27 }
28 void rotate(int x){ //旋轉，會自行判斷轉的方
29     向
30     int y=nd[x].pa,z=nd[y].pa,d=(nd[y].ch[1]==
31         x);
32     nd[x].pa=z;
33     if(!isroot(y))nd[z].ch[nd[z].ch[1]==y]=x;
34     nd[y].ch[d]=nd[x].ch[d^1];
35     nd[nd[y].ch[d]].pa=y;
36     nd[y].pa=x,nd[x].ch[d^1]=y;
37     up(y),up(x);
38 }
39 void splay(int x){ //將x伸展到splay tree的根
40     push_down(x);
41     while(!isroot(x)){
42         int y=nd[x].pa;
43         if(!isroot(y)){
44             int z=nd[y].pa;
45             if((nd[z].ch[0]==y)^((nd[y].ch[0]==x))
46                 rotate(y);
47             else rotate(x);
48         }
49         rotate(x);
50     }
51 }
52 int access(int x){
53     int last=0;
54     while(x){
55         splay(x);
56         nd[x].ch[1]=last;
57         up(x);
58         last=x;
59         x=nd[x].pa;
60     }
61     return last; //access後splay tree的根
62 }

```

```

57 void access(int x,bool is=0){ //is=0就是一般
58     的access
59     int last=0;
60     while(x){
61         splay(x);
62         if(is&&!nd[x].pa){
63             //printf("%d\n",max(nd[Last].ma,nd[nd[
64                 x].ch[1]].ma));
65         }
66         nd[x].ch[1]=last;
67         up(x);
68         last=x;
69         x=nd[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     nd[x].rev^=1;
79 }
80 void make_root(int x){
81     nd[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     nd[y].ch[0]=0;
89     nd[x].pa=0;
90 }
91 void cut_parents(int x){
92     access(x);
93     splay(x);
94     nd[nd[x].ch[0]].pa=0;
95     nd[x].ch[0]=0;
96 }
97 void link(int x,int y){
98     make_root(x);
99     nd[x].pa=y;
100 }
101 int find_root(int x){
102     x=access(x);
103     while(nd[x].ch[0])x=nd[x].ch[0];
104     return x;
105 }
106 int query(int u,int v){
107     //傳回uv路徑splay tree的根結點
108     //這種寫法無法求LCA
109     make_root(u);
110     return access(v);
111 }
112 int query_lca(int u,int v){
113     //假設求鏈上點權的總和，sum是子樹的權重和，
114     data是節點的權重
115     access(u);
116     int lca=access(v);
117     splay(u);
118     if(u==lca){
119         //return nd[lca].data+nd[nd[lca].ch[1]].
120         sum
121     }
122 }

```

```

123 }else{
124     //return nd[lca].data+nd[nd[lca].ch[1]].
125     sum+nd[u].sum
126 }
127 }
128 struct EDGE{
129     int a,b,w;
130 }e[10005];
131 int n;
132 vector<pair<int,int>> G[10005];
133 //first表示子節點，second表示邊的編號
134 int pa[10005],edge_node[10005];
135 //pa是父母節點，暫存用的，edge_node是每個編
136 被存在哪個點裡面的陣列
137 void bfs(int root){
138     //在建構的時候把每個點都設成一個splay tree
139     queue<int> q;
140     for(int i=1;i<=n;++i)pa[i]=0;
141     q.push(root);
142     while(q.size()){
143         int u=q.front();
144         q.pop();
145         for(auto P:G[u]){
146             int v=P.first;
147             if(v!=pa[u]){
148                 pa[v]=u;
149                 nd[v].pa=u;
150                 nd[v].data=e[P.second].w;
151                 edge_node[P.second]=v;
152                 up(v);
153                 q.push(v);
154             }
155         }
156     }
157 }
158 void change(int x,int b){
159     splay(x);
160     //nd[x].data=b;
161     up(x);
162 }

```

## 8.6 LCA

```

1 const int MAXN=200000; // 1-base
2 const int MLG=__lg(MAXN) + 1; //Log2(MAXN)
3 +1;
4 int pa[MLG+2][MAXN+5];
5 int dep[MAXN+5];
6 vector<int> G[MAXN+5];
7 void dfs(int x,int p=0){ //dfs(root);
8     pa[0][x]=p;
9     for(int i=0;i<=MLG;++i)
10         pa[i+1][x]=pa[i][pa[i][x]];
11     for(auto &i:G[x]){
12         if(i==p)continue;
13         dep[i]=dep[x]+1;
14         dfs(i,x);
15     }
16 }
17 inline int jump(int x,int d){
18     for(int i=0;i<=MLG;++i)
19         if((d>>i)&1) x=pa[i][x];
20 }

```

```

19 return x;
20 }
21 inline int find_lca(int a, int b){
22     if(dep[a]>dep[b])swap(a,b);
23     b=jump(b,dep[b]-dep[a]);
24     if(a==b)return a;
25     for(int i=MLG;i>0;--i){
26         if(pa[i][a]!=pa[i][b]){
27             a=pa[i][a];
28             b=pa[i][b];
29         }
30     }
31     return pa[0][a];
32 }
33
34 //用樹壓平做
35 #define MAXN 100000
36 typedef vector<int> ::iterator VIT;
37 int dep[MAXN+5], in[MAXN+5];
38 int vs[2*MAXN+5];
39 int cnt; /*時間戳*/
40 vector<int> >G[MAXN+5];
41 void dfs(int x, int pa){
42     in[x]=++cnt;
43     vs[cnt]=x;
44     for(VIT i=G[x].begin(); i!=G[x].end(); ++i){
45         if(*i==pa)continue;
46         dep[*i]=dep[x]+1;
47         dfs(*i, x);
48         vs[++cnt]=x;
49     }
50 }
51
52 inline int find_lca(int a, int b){
53     if(in[a]>in[b])swap(a,b);
54     return RMQ(in[a], in[b]);
55 }

```

## 8.7 樹壓平

```

1 //紀錄 in & out
2 vector<int> Arr;
3 vector<int> In, Out;
4 void dfs(int u) {
5     Arr.push_back(u);
6     In[u] = Arr.size() - 1;
7     for (auto v : Tree[u]) {
8         if (v == parent[u])
9             continue;
10        parent[v] = u;
11        dfs(v);
12    }
13    Out[u] = Arr.size() - 1;
14 }
15
16 //進去出來都紀錄
17 vector<int> Arr;
18 void dfs(int u) {
19     Arr.push_back(u);
20     for (auto v : Tree[u]) {
21         if (v == parent[u])
22             continue;

```

```

23     parent[v] = u;
24     dfs(v);
25 }
26 Arr.push_back(u);
27 }
28
29 //用Treap紀錄
30 Treap *root = nullptr;
31 vector<Treap*> In, Out;
32 void dfs(int u) {
33     In[u] = new Treap(cost[u]);
34     root = merge(root, In[u]);
35     for (auto v : Tree[u]) {
36         if (v == parent[u])
37             continue;
38         parent[v] = u;
39         dfs(v);
40     }
41     Out[u] = new Treap(0);
42     root = merge(root, Out[u]);
43 }
44
45 //Treap紀錄Parent
46 struct Treap {
47     Treap *lc = nullptr, *rc = nullptr;
48     Treap *pa = nullptr;
49     unsigned pri, size;
50     long long Val, Sum;
51     Treap(int Val):
52         pri(rand()), size(1),
53         Val(Val), Sum(Val) {}
54     void pull();
55 };
56
57 void Treap::pull() {
58     size = 1;
59     Sum = Val;
60     pa = nullptr;
61     if (lc) {
62         size += lc->size;
63         Sum += lc->Sum;
64         lc->pa = this;
65     }
66     if (rc) {
67         size += rc->size;
68         Sum += rc->Sum;
69         rc->pa = this;
70     }
71 }
72
73 //找出節點在中序的編號
74 size_t getIdx(Treap *x) {
75     assert(x);
76     size_t Idx = 0;
77     for (Treap *child = x->rc; x;) {
78         if (child == x->rc)
79             Idx += 1 + size(x->lc);
80         child = x;
81         x = x->pa;
82     }
83     return Idx;
84 }
85
86 //切出想要的東西
87 void move(Treap *&root, int a, int b) {
88     size_t a_in = getIdx(In[a]), a_out =
89         getIdx(Out[a]);
90     auto [L, tmp] = splitK(root, a_in - 1);

```

```

87 auto [tree_a, R] = splitK(tmp, a_out -
88     a_in + 1);
89 root = merge(L, R);
90 tie(L, R) = splitK(root, getIdx(In[b]));
91 root = merge(L, merge(tree_a, R));
92 }

```

## 9 string

### 9.1 KMP

```

1 const int N = 1e6+5;
2 /*產生fail function*/
3 void kmp_fail(char *s, int len, int *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 }
12 vector<int> match_index;
13 /*以字串B匹配字串A，傳回匹配成功的數量(用B的fail)*/
14 int kmp_match(char *A, int lenA, char *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, id=fail[id];
21             match_index.emplace_back(i + 1 - lenB);
22         }
23     }
24     return ans;
25 }

```

### 9.2 reverseBWT

```

1 const int MAXN = 305, MAXC = 'Z';
2 int ranks[MAXN], tots[MAXC], first[MAXC];
3 void rankBWT(const string &bw){
4     memset(ranks, 0, sizeof(int)*bw.size());
5     memset(tots, 0, sizeof(tots));
6     for(size_t i=0;i<bw.size();++i)
7         ranks[i] = tots[int(bw[i])]+1;
8 }
9
10 void firstCol(){
11     memset(first, 0, sizeof(first));
12     int totc = 0;
13     for(int c='A'; c<='Z'; ++c){
14         if(!tots[c]) continue;
15         first[c] = totc;
16         totc += tots[c];
17     }

```

```

17 }
18 string reverseBwt(string bw, int begin){
19     rankBWT(bw, firstCol());
20     int i = begin; //原字串最後一個元素的位置
21     string res;
22     do{
23         char c = bw[i];
24         res = c + res;
25         i = first[int(c)] + ranks[i];
26     }while( i != begin );
27     return res;
28 }

```

### 9.3 Z

```

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

```

### 9.4 Trie

```

1 template<int ALPHABET = 26, char MIN_CHAR = 'a'>
2 class trie {
3 public:
4     struct Node {
5         int go[ALPHABET];
6         Node() {
7             memset(go, -1, sizeof(go));
8         }
9     };
10
11     trie() {
12         newNode();
13     }
14
15     inline int next(int p, int v) {
16         return nodes[p].go[v] != -1 ? nodes[p].go[v] : nodes[p].go[v] = newNode();
17     }
18
19     inline void insert(const vector<int>& a, int p = 0) {
20         for(int v : a) {
21             p = next(p, v);
22         }
23     }
24
25     inline void clear() {
26         nodes.clear();
27         newNode();
28     }

```

```

28 }
29
30 inline int longest_common_prefix(const
    vector<int>& a, int p = 0) const {
31     int ans = 0;
32     for(int v : a) {
33         if(nodes[p].go[v] != -1) {
34             ans += 1;
35             p = nodes[p].go[v];
36         } else {
37             break;
38         }
39     }
40     return ans;
41 }
42
43 private:
44     vector<Node> nodes;
45
46     inline int newNode() {
47         nodes.emplace_back();
48         return (int) nodes.size() - 1;
49     }
50 };

```

## 9.5 Rolling Hash

```

1 const ll M = 911382323, mod = 972663749;
2 ll Get(vector<ll>& h, int l, int r) {
3     if(!l) return h[r]; // p[i] = M^i % mod
4     ll ans = (h[r] - h[l - 1] * p[r - l + 1])
5         % mod;
6     return (ans + mod) % mod;
7 }
8 vector<ll> Hash(string s) {
9     vector<ll> ans(SZ(s));
10    ans[0] = s[0];
11    for(int i = 1; i < SZ(s); i++) ans[i] = (
12        ans[i - 1] * M + s[i]) % mod;
13 }

```

## 9.6 SAM

```

1 const int MAXM = 1000010;
2 struct SAM {
3     int tot, root, lst, mom[MAXM], mx[MAXM];
4     int nxt[MAXM][33], cnt[MAXM], in[MAXM];
5     inline int newNode() {
6         int res = ++tot;
7         fill(nxt[res], nxt[res] + 33, 0);
8         mom[res] = mx[res] = cnt[res] = in[res]
9             = 0;
10        return res;
11    }
12    void init() {
13        tot = 0;
14        root = newNode();
15        mom[root] = 0, mx[root] = 0;
16        lst = root;
17    }

```

```

16 }
17 void push(int c) {
18     int p = lst;
19     int np = newNode();
20     mx[np] = mx[p] + 1;
21     for (; p && nxt[p][c] == 0; p = mom[p])
22         nxt[p][c] = np;
23     if (p == 0) mom[np] = root;
24     else {
25         int q = nxt[p][c];
26         if (mx[p] + 1 == mx[q]) mom[np] = q;
27         else {
28             int nq = newNode();
29             mx[nq] = mx[p] + 1;
30             for (int i = 0; i < 33; i++)
31                 nxt[nq][i] = nxt[p][i];
32             mom[nq] = mom[q];
33             mom[p] = nq;
34             mom[np] = nq;
35             for (; p && nxt[p][c] == q; p = mom[
36                 p])
37                 nxt[p][c] = nq;
38         }
39         lst = np, cnt[np] = 1;
40     }
41     void push(char *str) {
42         for (int i = 0; str[i]; i++)
43             push(str[i] - 'a' + 1);
44     }
45     void count() {
46         for (int i = 1; i <= tot; ++i)
47             ++in[mom[i]];
48         queue<int> q;
49         for (int i = 1; i <= tot; ++i)
50             if (!in[i]) q.push(i);
51         while (!q.empty()) {
52             int u = q.front();
53             q.pop();
54             cnt[mom[u]] += cnt[u];
55             if (--in[mom[u]] == 0)
56                 q.push(mom[u]);
57         }
58     }
59 } sam;

```

## 9.7 suffix array lcp

```

1 #define radix_sort(x,y){\
2     for(i=0;i<A;++i)c[i]=0;\
3     for(i=0;i<n;++i)c[x[y[i]]]++;\
4     for(i=1;i<A;++i)c[i]+=c[i-1];\
5     for(i=n-1;~i;--i)sa[--c[x[y[i]]]]=y[i];\
6 }
7 #define AC(r,a,b)\
8     r[a]!=(r[b]||a+k>n||r[a+k]!=r[b+k])
9 void suffix_array(const char *s,int n,int *
    sa,int *rank,int *tmp,int *c){
10     int A='z'+1,i,k,id=0;
11     for(i=0;i<n;++i)rank[tmp[i]=i]=s[i];
12     radix_sort(rank,tmp);
13     for(k=1;id<n-1;k<=1){
14         for(id=0,i=n-k;i<n;++i)tmp[id++]=i;

```

```

15     for(i=0;i<n;++i)
16         if(sa[i]>=k)tmp[id++]=sa[i]-k;
17     radix_sort(rank,tmp);
18     swap(rank,tmp);
19     for(rank[sa[0]]=id=0,i=1;i<n;++i)
20         rank[sa[i]]=id+=AC(tmp,sa[i-1],sa[i]);
21     A=id+1;
22 }
23 //h:高度數組 sa:後綴數組 rank:排名
24 void suffix_array_lcp(const char *s,int len,
25     int *h,int *sa,int *rank){
26     for(int i=0;i<len;++i)rank[sa[i]]=i;
27     for(int i=0,k=0;i<len;++i){
28         if(rank[i]==0)continue;
29         if(k)--k;
30         while(s[i+k]==s[sa[rank[i]-1]+k])++k;
31         h[rank[i]]=k;
32     }
33     h[0]=0; // h[k]=lcp(sa[k],sa[k-1]);
34 }

```

## 9.8 AC 自動機

```

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

```

```

40            t=S[pa].next[i];
41            if(!t)continue;
42            id=S[pa].fail;
43            while(~id&&!S[id].next[i])id=S[id].
44                fail;
45            S[t].fail=~id?S[id].next[i]:0;
46            S[t].efl=S[S[t].fail].ed?S[t].fail:S
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有
53        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,i;
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都
62                    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[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匹配成功
72        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邊所以保證
83                        匹配成功*/
84                }
85            }
86            return ans;
87        }
88    /*枚舉(s的子字串nA)的所有相異字串各恰一次
89        並傳回次數O(N*M^(1/3))*/
90    int match_2(const char *s){
91        int ans=0,id,p=0,t;
92        ++vt;
93        /*把記號vt+=1,只要vt沒溢位,所有S[p].
94            vis=vt就會變成false
95            這種利用vt的方法可以O(1)歸零vis陣列*/
96        for(int i=0;s[i];++i){
97            id=s[i]-L;
98            while(!S[p].next[id]&&p=S[p].fail;
99                if(!S[p].next[id])continue;
100                p=S[p].next[id];
101                if(S[p].ed&&S[p].vis!=vt){
102                    S[p].vis=vt;

```

```

95     ans+=S[p].ed;
96 }
97 for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t]
98     ].efl){
99     S[t].vis=vt;
100     ans+=S[t].ed; /*因為都走efl邊所以保證
101                  匹配成功*/
102 }
103 }
104 return ans;
105 }
106 /*把AC自動機變成真的自動機*/
107 void evolution(){
108     for(qs=1;qs!=qe;){
109         int p=q[qs++];
110         for(int i=0;i<=R-L;++i)
111             if(S[p].next[i]==0)S[p].next[i]=S[S[
112                 p].fail].next[i];

```

## 9.9 minimal string rotation

```

1 //找最小循環表示法起始位置
2 int min_string_rotation(const string &s){
3     int n=s.size(),i=0,j=1,k=0;
4     while(i<n&&j<n&&k<n){
5         int t=s[(i+k)%n]-s[(j+k)%n];
6         ++k;
7         if(t){
8             if(t>0)i+=k;
9             else j+=k;
10            if(i==j)++j;
11            k=0;
12        }
13    }
14    return min(i,j); //最小循環表示法起始位置
15 }

```

## 9.10 De Bruijn sequence

```

1 constexpr int MAXC = 10, MAXN = 1e5 + 10;
2 struct DBSeq {
3     int C, N, K, L, buf[MAXC * MAXN]; // K <=
4     C^N
5     void dfs(int *out, int t, int p, int &ptr)
6     {
7         if (ptr >= L) return;
8         if (t > N) {
9             if (N % p) return;
10            for (int i = 1; i <= p && ptr < L; ++i)
11                out[ptr++] = buf[i];
12        } else {
13            buf[t] = buf[t - p], dfs(out, t + 1, p
14                , ptr);
15            for (int j = buf[t - p] + 1; j < C; ++
16                j)

```

```

13         buf[t] = j, dfs(out, t + 1, t, ptr);
14     }
15 }
16 void solve(int _c, int _n, int _k, int *
17     out) {
18     int p = 0;
19     C = _c, N = _n, K = _k, L = N + K - 1;
20     dfs(out, 1, 1, p);
21     if (p < L) fill(out + p, out + L, 0);
22 }
23 }
24 }

```

## 9.11 manacher

```

1 //找最長迴文子字串
2 //原字串: asdsasdsa
3 //先把字串變成這樣: @#a#s#d#s#a#s#d#s#a#
4 void manacher(char *s, int len, int *z){
5     int l=0, r=0;
6     for(int i=1; i<len; ++i){
7         z[i]=r>i?min(z[2*i-l-i], r-i):1;
8         while(s[i+z[i]]==s[i-z[i]])++z[i];
9         if(z[i]+i>r)r=z[i]+i, l=i;
10    } //ans = max(z)-1
11 }

```

## 10 tools

### 10.1 Template

```

1 #include <bits/extc++.h>
2 #include <bits/stdc++.h>
3 #pragma GCC optimize("O3,unroll-loops")
4 #pragma GCC target("avx2,bmi,bmi2,lzcnt,
5     popcnt")
6 #define IOS ios::sync_with_stdio(0), cin.tie
7     (0), cout.tie(0)
8 #define int long long
9 #define double long double
10 #define pb push_back
11 #define sz(x) (int)(x).size()
12 #define all(v) begin(v), end(v)
13 #define debug(x) cerr<<"x<<" = "<<x<<"\n"
14 #define LINE cout<<"\n-----\n"
15 #define endl '\n'
16 #define VI vector<int>
17 #define F first
18 #define S second
19 #define MP(a,b) make_pair(a,b)
20 #define rep(i,m,n) for(int i = m; i<=n; ++i)
21 #define res(i,m,n) for(int i = m; i>=n; --i)
22 #define gcd(a,b) __gcd(a,b)
23 #define lcm(a,b) a*b/gcd(a,b)
24 #define Case() int _; cin>>_; for(int Case =
25     1; Case<=;; ++Case)
26 #define pii pair<int,int>
27 using namespace __gnu_cxx;

```

```

25 using namespace __gnu_pbds;
26 using namespace std;
27 template <typename K, typename cmp = less<K
28     >, typename T = thin_heap_tag> using
29     _heap = __gnu_pbds::priority_queue<K,
30     cmp, T>;
31 template <typename K, typename M = null_type
32     > using _hash = gp_hash_table<K, M>;
33 const int N = 1e6+5, L = 20, mod = 1e9+7;
34 const long long inf = 2e18+5;
35 const double eps = 1e-7, pi = acos(-1);
36 void solve(){
37 }
38 signed main(){
39     IOS;
40     solve();
41 }
42 //使用內建紅黑樹
43 template<class T, typename cmp=less<>>struct
44     _tree{ // #include<bits/extc++.h>
45         tree<pair<T, int>, null_type, cmp, rb_tree_tag
46             , tree_order_statistics_node_update> st;
47         int id = 0;
48         void insert(T x){st.insert({x, id++});}
49         void erase(T x){st.erase(st.lower_bound({x
50             , 0}));}
51         int order_of_key(T x){return st.
52             order_of_key(*st.lower_bound({x, 0}));}
53         T find_by_order(int x){return st.
54             find_by_order(x)->first;}
55         T lower_bound(T x){return st.lower_bound({
56             x, 0})->first;}
57         T upper_bound(T x){return st.upper_bound({
58             x, (int)1e9+7})->first;}
59         T smaller_bound(T x){return (--st.
60             lower_bound({x, 0}))->first;}
61     };

```

### 10.2 Counting Sort

```

1 vector<unsigned> counting_sort(const vector<
2     unsigned> &Arr, unsigned K) {
3     vector<unsigned> Bucket(K, 0);
4     for(auto x: Arr)
5         ++Bucket[x];
6     partial_sum(Bucket.begin(), Bucket.end(),
7         Bucket.begin());
8     vector<unsigned> Ans(Arr.size());
9     for(auto Iter = Arr.rbegin(); Iter != Arr.
10         rend(); ++Iter) Ans[--Bucket[*Iter]] =
11         *Iter;
12     return Ans;
13 }

```

### 10.3 relabel

```

1 template<class T>
2 vector<int> Discrete(const vector<T>&v){

```

```

3     vector<int> ans;
4     vector<T> tmp(v);
5     sort(begin(tmp), end(tmp));
6     tmp.erase(unique(begin(tmp), end(tmp)), end(
7         tmp));
8     for(auto i:v) ans.push_back(lower_bound(
9         begin(tmp), end(tmp), i)-tmp.begin()+1);
10    return ans;

```

## 10.4 TouristIO

```

1 static struct FastInput {
2     static constexpr int BUF_SIZE = 1 << 20;
3     char buf[BUF_SIZE];
4     size_t chars_read = 0;
5     size_t buf_pos = 0;
6     FILE *in = stdin;
7     char cur = 0;
8     inline char get_char() {
9         if(buf_pos >= chars_read) {
10             chars_read = fread(buf, 1, BUF_SIZE,
11                 in);
12             buf_pos = 0;
13             buf[0] = (chars_read == 0 ? -1 : buf
14                 [0]);
15         }
16         return cur = buf[buf_pos++];
17         // return cur = getchar_unlocked();
18     }
19     inline void tie(int) {}
20     inline explicit operator bool() {
21         return cur != -1;
22     }
23 }
24
25 inline static bool is_blank(char c) {
26     return c <= ' ';
27 }
28
29 inline bool skip_blanks() {
30     while(is_blank(cur) && cur != -1) {
31         get_char();
32     }
33     return cur != -1;
34 }
35
36 inline FastInput& operator>>(char& c) {
37     skip_blanks();
38     c = cur;
39     return *this;
40 }
41
42 inline FastInput& operator>>(string& s) {
43     if(skip_blanks()) {
44         s.clear();
45         do {
46             s += cur;
47         } while(!is_blank(get_char()));
48     }
49     return *this;

```

```

50 }
51
52 template<class T>
53 inline FastInput& read_integer(T& n) {
54     // unsafe, doesn't check that characters
55     // are actually digits
56     n = 0;
57     if(skip_blanks()) {
58         int sign = +1;
59         if(cur == '-') {
60             sign = -1;
61             get_char();
62         }
63         do {
64             n += n * (n << 3) + cur - '0';
65             while(!is_blank(get_char()));
66             n *= sign;
67         }
68         return *this;
69     }
70
71     template<class T>
72     inline typename enable_if<is_integral<T>::
73         value, FastInput&>::type operator>>(T&
74         n) {
75         return read_integer(n);
76     }
77
78     #if!defined(_WIN32) || defined(_WIN64)
79     inline FastInput& operator>>(__int128& n)
80     {
81         return read_integer(n);
82     }
83     #endif
84
85     template<class T>
86     inline typename enable_if<
87         is_floating_point<T>::value, FastInput
88         &>::type operator>>(T& n) {
89         // not sure if really fast, for
90         // compatibility only
91         n = 0;
92         if(skip_blanks()) {
93             string s;
94             (*this) >> s;
95             sscanf(s.c_str(), "%Lf", &n);
96         }
97         return *this;
98     }
99 } fast_input;
100
101 #define cin fast_input
102
103 static struct FastOutput {
104     static constexpr int BUF_SIZE = 1 << 20;
105     char buf[BUF_SIZE];
106     size_t buf_pos = 0;
107     static constexpr int TMP_SIZE = 1 << 20;
108     char tmp[TMP_SIZE];
109     FILE *out = stdout;
110
111     inline void put_char(char c) {
112         buf[buf_pos++] = c;
113         if(buf_pos == BUF_SIZE) {
114             fwrite(buf, 1, buf_pos, out);
115             buf_pos = 0;
116         }
117     }
118 }

```

```

109 }
110 // putchar_unlocked(c);
111 }
112
113 ~FastOutput() {
114     fwrite(buf, 1, buf_pos, out);
115 }
116
117 inline FastOutput& operator<<(char c) {
118     put_char(c);
119     return *this;
120 }
121
122 inline FastOutput& operator<<(const char*
123     s) {
124     while(*s) {
125         put_char(*s++);
126     }
127     return *this;
128 }
129
130 inline FastOutput& operator<<(const string
131     & s) {
132     for(int i = 0; i < (int) s.size(); i++)
133     {
134         put_char(s[i]);
135     }
136     return *this;
137 }
138
139 template<class T>
140 inline char* integer_to_string(T n) {
141     // beware of TMP_SIZE
142     char* p = tmp + TMP_SIZE - 1;
143     if(n == 0) {
144         *--p = '0';
145     }
146     else {
147         bool is_negative = false;
148         if(n < 0) {
149             is_negative = true;
150             n = -n;
151         }
152         while(n > 0) {
153             *--p = (char) ('0' + n % 10);
154             n /= 10;
155         }
156         if(is_negative) {
157             *--p = '-';
158         }
159     }
160     return p;
161 }
162
163 template<class T>
164 inline typename enable_if<is_integral<T>::
165     value, char*>::type stringify(T n) {
166     return integer_to_string(n);
167 }
168
169 #if!defined(_WIN32) || defined(_WIN64)
170 inline char* stringify(__int128 n) {
171     return integer_to_string(n);
172 }
173 #endif
174
175 template<class T>

```

```

176 inline typename enable_if<
177     is_floating_point<T>::value, char*>::
178     type stringify(T n) {
179     sprintf(tmp, "%.17f", n);
180     return tmp;
181 }
182
183 template<class T>
184 inline FastOutput& operator<<(const T& n)
185 {
186     auto p = stringify(n);
187     for(; *p != 0; p++) {
188         put_char(*p);
189     }
190     return *this;
191 }
192
193 #define cout fast_output

```

## 10.5 TernarySearch

```

1 // return the maximum of f(x) in [L, r]
2 double ternary_search(double l, double r) {
3     while(r - l > EPS) {
4         double m1 = l + (r - l) / 3;
5         double m2 = r - (r - l) / 3;
6         double f1 = f(m1), f2 = f(m2);
7         if(f1 < f2) l = m1;
8         else r = m2;
9     }
10    return f(l);
11 }
12
13 // return the maximum of f(x) in [L, r]
14 int ternary_search(int l, int r) {
15     while(r - l > 1) {
16         int mid = (l + r) / 2;
17         if(f(mid) > f(mid + 1)) r = mid;
18         else l = mid;
19     }
20    return r;
21 }

```

## 10.6 template bubble

```

1 #define lim 1000000007
2 #define ll long long
3 #define endl "\n"
4 #define Crbubble cin.tie(0); ios_base::
5     sync_with_stdio(false);
6 #define aqua clock_t qua = clock();
7 #define aquaa cout << "Aqua says: " << (
8     double)(clock()-qua)/CLOCKS_PER_SEC << "
9     sec!\n";
10 #define random_set(m,n) random_device rd; \
11     mt19937 gen=mt19937(
12     rd()); \
13     uniform_int_distribution
14     <ll> dis(m,n); \

```

```

10 auto rnd=bind(dis,
11     gen);

```

## 10.7 DuiPai

```

1 int main(){
2     string sol,bf,make;
3     cout<<"Your solution file name :";
4     cin>>sol;
5     cout<<"Brute force file name :";
6     cin>>bf;
7     cout<<"Make data file name :";
8     cin>>make;
9     system(("g++ "+sol+" -o sol").c_str());
10    system(("g++ "+bf+" -o bf").c_str());
11    system(("g++ "+make+" -o make").c_str());
12    for(int t = 0;t<10000;++t){
13        system("./make > ./1.in");
14        double st = clock();
15        system("./sol < ./1.in > ./1.ans");
16        double et = clock();
17        system("./bf < ./1.in > ./1.out");
18        if(system("diff ./1.out ./1.ans")) {
19            printf("\033[0;31mWrong Answer\033[0m
20                on test #%d",t);
21            return 0;
22        }
23        else if(et-st>=2000){
24            printf("\033[0;32mTime Limit exceeded
25                \033[0m on test #%d, Time %.0lfms\
26                n",t,et-st);
27            return 0;
28        }
29        else {
30            printf("\033[0;32mAccepted\033[0m
31                m on test #%d, Time %.0lfms\
32                n", t, et - st);
33        }
34    }
35 }

```

## 10.8 bitset

```

1 |bitset<size> b(a):長度為size · 初始化為a
2 |b[i]:第i位元的值(0 or 1)
3 |b.size():有幾個位元
4 |b.count():有幾個1
5 |b.set():所有位元設為1
6 |b.reset():所有位元設為0
7 |b.flip():所有位元反轉

```



# ACM ICPC Team Reference - DreaminBubble

## Contents

<b>1 Computational Geometry</b>	<b>1</b>	3.9 monotonic stack . . . . .	6	5.13 Dijkstra . . . . .	12	<b>8 Tree</b>	<b>18</b>
1.1 最近點對 . . . . .	1	3.10 Kruskal . . . . .	6	5.14 SCC . . . . .	12	8.1 Tree centroid . . . . .	18
1.2 MinCircleCover . . . . .	1	3.11 Lazytag Segment Tree . . . .	6	5.15 Minimum Clique Cover . . .	12	8.2 HLD . . . . .	18
1.3 Geometry . . . . .	1	3.12 2D BIT . . . . .	7	5.16 判斷環 . . . . .	13	8.3 HeavyLight . . . . .	18
<b>2 DP</b>	<b>3</b>	3.13 monotonic queue . . . . .	7	5.17 2-SAT . . . . .	13	8.4 centroidDecomposition . . . .	18
2.1 整體二分 . . . . .	3	3.14 Prim . . . . .	7	<b>6 Math</b>	<b>13</b>	8.5 link cut tree . . . . .	19
2.2 LineContainer . . . . .	3	3.15 回滾並查集 . . . . .	7	6.1 Primes . . . . .	13	8.6 LCA . . . . .	19
2.3 斜率優化-動態凸包 . . . . .	3	3.16 TimingSegmentTree . . . . .	7	6.2 InvGCD . . . . .	13	8.7 樹壓平 . . . . .	20
2.4 basic DP . . . . .	3	3.17 SegmentTree . . . . .	7	6.3 LinearCongruence . . . . .	13	<b>9 string</b>	<b>20</b>
2.5 DP on Graph . . . . .	4	3.18 Persistent Segment Tree . . .	8	6.4 Bit Set . . . . .	13	9.1 KMP . . . . .	20
2.6 單調隊列優化 . . . . .	4	3.19 pbds . . . . .	8	6.5 Lucas . . . . .	13	9.2 reverseBWT . . . . .	20
<b>3 Data Structure</b>	<b>4</b>	3.20 LiChaoST . . . . .	8	6.6 ExtendGCD . . . . .	13	9.3 Z . . . . .	20
3.1 sparse table . . . . .	4	3.21 DynamicMST . . . . .	8	6.7 Basic . . . . .	13	9.4 Trie . . . . .	20
3.2 BinaryTrie . . . . .	4	<b>4 Flow</b>	<b>8</b>	6.8 XOR-Basis . . . . .	14	9.5 Rolling Hash . . . . .	21
3.3 BIT . . . . .	5	4.1 Property . . . . .	8	6.9 Theorem . . . . .	14	9.6 SAM . . . . .	21
3.4 Dynamic Segment Tree . . . .	5	4.2 Gomory Hu . . . . .	9	6.10 Pisano number . . . . .	15	9.7 suffix array lcp . . . . .	21
3.5 掃描線 + 線段樹 . . . . .	5	4.3 MinCostMaxFlow . . . . .	9	6.11 Matrix . . . . .	15	9.8 AC 自動機 . . . . .	21
3.6 Persistent DSU . . . . .	6	4.4 dinic . . . . .	9	6.12 Numbers . . . . .	15	9.9 minimal string rotation . . . .	22
3.7 DSU . . . . .	6	4.5 ISAP with cut . . . . .	9	6.13 Triangle . . . . .	15	9.10 De Bruijn sequence . . . . .	22
3.8 陣列上 Treap . . . . .	6	<b>5 Graph</b>	<b>10</b>	6.14 GCD-Convolution . . . . .	15	9.11 manacher . . . . .	22
		5.1 橋連通分量 . . . . .	10	6.15 Combination . . . . .	16	<b>10 tools</b>	<b>22</b>
		5.2 SPFA . . . . .	10	6.16 FWT . . . . .	16	10.1 Template . . . . .	22
		5.3 manhattan-mst . . . . .	10	6.17 找實根 . . . . .	16	10.2 Counting Sort . . . . .	22
		5.4 最大團 . . . . .	10	6.18 NTT . . . . .	16	10.3 relabel . . . . .	22
		5.5 判斷平面圖 . . . . .	10	6.19 FFT . . . . .	16	10.4 TouristIO . . . . .	22
		5.6 count-bridge-online . . . . .	11	6.20 Gauss-Jordan . . . . .	16	10.5 TenarySearch . . . . .	23
		5.7 雙連通分量 & 割點 . . . . .	11	6.21 Pollard-Rho . . . . .	17	10.6 template bubble . . . . .	23
		5.8 枚舉極大團 Bron-Kerbosch .	11	<b>7 Square root decomposition</b>	<b>17</b>	10.7 DuiPai . . . . .	23
		5.9 Floyd Warshall . . . . .	11	7.1 MoAlgo . . . . .	17	10.8 bitset . . . . .	23
		5.10 Dominator tree . . . . .	11	7.2 莫隊 . . . . .	17		
		5.11 判斷二分圖 . . . . .	12	7.3 分塊 cf455D . . . . .	17		
		5.12 Bellman Ford . . . . .	12	7.4 Mos Algorithm On Tree . . .	18		

# ACM ICPC Judge Test - DreaminBubble

## C++ Resource Test

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 namespace system_test {
5
6     const size_t KB = 1024;
7     const size_t MB = KB * 1024;
8     const size_t GB = MB * 1024;
9
10    size_t block_size, bound;
11    void stack_size_dfs(size_t depth = 1) {
```

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

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

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