

1 Computational Geometry

1.1 Geometry

```

1  const double PI=atan2(0.0, -1.0);
2  template<typename T>
3  struct point{
4      T x,y;
5      point(){}
6      point(const T&x, const T&y):x(x),y(y){}
7      point operator+(const point &b) const{
8          return point(x+b.x,y+b.y); }
9      point operator-(const point &b) const{
10         return point(x-b.x,y-b.y); }
11     point operator*(const T &b) const{
12         return point(x*b,y*b); }
13     point operator/(const T &b) const{
14         return point(x/b,y/b); }
15     bool operator==(const point &b) const{
16         return x==b.x&&y==b.y; }
17     T dot(const point &b) const{
18         return x*b.x+y*b.y; }
19     T cross(const point &b) const{
20         return x*b.y-y*b.x; }
21     point normal() const{ //求法向量
22         return point(-y,x); }
23     T abs2() const{ //向量長度的平方
24         return dot(*this); }
25     T rad(const point &b) const{ //兩向量的弧度
26     return fabs(atan2(fabs(cross(b)),dot(b))); }
27     T getA() const{ //對x軸的弧度
28         T A=atan2(y,x); //超過180度會變負的
29         if(A<=-PI/2) A+=PI*2;
30         return A;
31     }
32 };
33 template<typename T>
34 struct line{
35     line(){}
36     point<T> p1,p2;
37     T a,b,c; //ax+by+c=0
38     line(const point<T>&x, const point<T>&y):p1(x),p2(y){}
39     void pton() const{ //轉成一般式
40         a=p1.y-p2.y;
41         b=p2.x-p1.x;
42         c=-a*p1.x-b*p1.y;
43     }
44     T ori(const point<T> &p) const{ //點和有向直
45         線的關係 · >0左邊 · =0在線上 <0右邊
46         return (p2-p1).cross(p-p1);
47     }
48     T btw(const point<T> &p) const{ //點投影落在
49         線段上 <=0
50         return (p1-p).dot(p2-p);
51     }
52     bool point_on_segment(const point<T>&p)
53         const{ //點是否在線段上
54         return ori(p)==0&&btw(p)<=0;
55     }
56     T dis2(const point<T> &p, bool is_segment
57         =0) const{ //點跟直線/線段的距離平方
58         point<T> v=p2-p1,v1=v-p1;
59         if(is_segment){
60             point<T> v2=p-p2;
61             if(v.dot(v1)<=0) return v1.abs2();
62             if(v.dot(v2)>=0) return v2.abs2();
63         }
64         T tmp=v.cross(v1);
65         return tmp*tmp/v.abs2();
66     }
67     T seg_dis2(const line<T> &l) const{ //兩線段
68         距離平方
69         return min({dis2(l.p1,1),dis2(l.p2,1),l.
70             dis2(p1,1),l.dis2(p2,1)});
71     }
72     point<T> projection(const point<T> &p)
73         const{ //點對直線的投影
74         point<T> n=(p2-p1).normal();
75         return p-n*(p-p1).dot(n)/n.abs2();
76     }
77     point<T> mirror(const point<T> &p) const{
78         //點對直線的鏡射 · 要先呼叫pton轉成一般式
79         point<T> R;
80         T d=a*b+b*b;
81         R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
82         R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
83         return R;
84     }
85     bool equal(const line &l) const{ //直線相等
86         return ori(l.p1)==0&&ori(l.p2)==0;
87     }
88     bool parallel(const line &l) const{
89         return (p1-p2).cross(l.p1-l.p2)==0;
90     }
91     bool cross_seg(const line &l) const{
92         return (p2-p1).cross(l.p1-p1)*(p2-p1).
93             cross(l.p2-p1)<=0; //直線是否交線段
94     }
95     int line_intersect(const line &l) const{ //
96         直線相交情況 · -1無限多點 · 1交於一點 · 0
97         不相交
98         return parallel(l)?(ori(l.p1)==0?-1:0)
99             :1;
100     }
101     int seg_intersect(const line &l) const{
102         T c1=ori(l.p1), c2=ori(l.p2);
103         T c3=l.ori(p1), c4=l.ori(p2);
104         if(c1==0&&c2==0){ //共線
105             bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
106             T a3=1.btw(p1),a4=1.btw(p2);
107             if(b1&&b2&&a3==0&&a4==0) return 2;
108             if(b1&&b2&&a3>=0&&a4==0) return 3;
109             if(b1&&b2&&a3>=0&&a4>=0) return 0;
110             return -1; //無限交點
111         }else if(c1*c2<=0&&c3*c4<=0) return 1;
112         return 0; //不相交
113     }
114     point<T> line_intersection(const line &l)
115         const{ //直線交點*/
116         point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
117         //if(a.cross(b)==0) return INF;
118         return p1+a*(s.cross(b)/a.cross(b));
119     }
120     point<T> seg_intersection(const line &l)
121         const{ //線段交點
122         int res=seg_intersect(l);
123         if(res<=0) assert(0);
124         if(res==2) return p1;
125         if(res==3) return p2;
126         return line_intersection(l);
127     }
128 };
129 template<typename T>
130 struct polygon{
131     polygon(){}
132     vector<point<T> > p; //逆時針順序
133     T area() const{ //面積
134         T ans=0;
135         for(int i=p.size()-1,j=0;j<(int)p.size()
136             ;i=j++){
137             ans+=p[i].cross(p[j]);
138         }
139         return ans/2;
140     }
141     point<T> center_of_mass() const{ //重心
142         T cx=0,cy=0,w=0;
143         for(int i=p.size()-1,j=0;j<(int)p.size()
144             ;i=j++){
145             T a=p[i].cross(p[j]);
146             cx+=(p[i].x+p[j].x)*a;
147             cy+=(p[i].y+p[j].y)*a;
148             w+=a;
149         }
150         return point<T>(cx/3/w,cy/3/w);
151     }
152     char ahas(const point<T>& t) const{ //點是否
153         在簡單多邊形內 · 是的話回傳1 · 在邊上回
154         傳-1 · 否則回傳0
155         bool c=0;
156         for(int i=0,j=p.size()-1;i<p.size();j=i
157             ++){
158             if(line<T>(p[i],p[j]).point_on_segment
159                 (t)) return -1;
160             else if((p[i].y>t.y)!=p[j].y>t.y)&&
161                 t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j
162                     ].y-p[i].y)+p[i].x)
163                 c=!c;
164             return c;
165         }
166     }
167     char point_in_convex(const point<T>&x)
168         const{
169         int l=1,r=(int)p.size()-2;
170         while(l<r){ //點是否在凸多邊形內 · 是的話
171             回傳1 · 在邊上回傳-1 · 否則回傳0
172             int mid=(l+r)/2;
173             T a1=(p[mid]-p[0]).cross(x-p[0]);
174             T a2=(p[mid+1]-p[0]).cross(x-p[0]);
175             if(a1>=0&&a2<=0){
176                 T res=(p[mid+1]-p[mid]).cross(x-p[
177                     mid]);
178                 return res>0?-1:(res==0?-1:0);
179             }else if(a1<0) r=mid-1;
180             else l=mid+1;
181         }
182         return 0;
183     }
184     vector<T> getA() const{ //凸包邊對x軸的夾角
185     vector<T> res; //一定是遞增的
186     for(size_t i=0;i<p.size();i++){
187         res.push_back((p[(i+1)%p.size()]-p[i])
188             .getA());
189     }
190     return res;
191 }
192 bool line_intersect(const vector<T>&A,
193     const line<T> &l) const{ //0(LogN)
194     int f1=upper_bound(A.begin(),A.end(),(l.
195         p1-l.p2).getA())-A.begin();
196     int f2=upper_bound(A.begin(),A.end(),(l.
197         p2-l.p1).getA())-A.begin();
198     return l.cross_seg(line<T>(p[f1],p[f2]));
199 }
200 polygon cut(const line<T> &l) const{ //凸包
201     對直線切割 · 得到直線L左側的凸包
202     polygon ans;
203     for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
204         if(l.ori(p[i])>=0){
205             ans.p.push_back(p[i]);
206             if(l.ori(p[j])<0)
207                 ans.p.push_back(l.
208                     line_intersection(line<T>(p[i
209                         ],p[j])));
210             }else if(l.ori(p[j])>0)
211                 ans.p.push_back(l.line_intersection(
212                     line<T>(p[i],p[j])));
213         }
214     }
215     return ans;
216 }
217 static bool monotone_chain_cmp(const point
218     <T>& a, const point<T>& b){ //凸包排序函
219     數
220     return (a.x<b.x)|| (a.x==b.x&&a.y<b.y);
221 }
222 void monotone_chain(vector<point<T> > &s){
223     //凸包
224     sort(s.begin(),s.end(),
225         monotone_chain_cmp);
226     p.resize(s.size()+1);
227     int m=0;
228     for(size_t i=0;i<s.size();i++){
229         while(m>=2&&(p[m-1]-p[m-2]).cross(s[i
230             ]-p[m-2])<=0)--m;
231         p[m++]=s[i];
232     }
233     for(int i=s.size()-2,t=m+1;i>=0;--i){
234         while(m>=t&&(p[m-1]-p[m-2]).cross(s[i
235             ]-p[m-2])<=0)--m;
236         p[m++]=s[i];
237     }
238     if(s.size()>1)--m;
239     p.resize(m);
240 }
241 T diam() const{ //直徑
242     int n=p.size(),t=1;
243     T ans=0;p.push_back(p[0]);
244     for(int i=0;i<n;i++){
245         point<T> now=p[i+1]-p[i];
246         while(now.cross(p[t+1]-p[i])>now.cross
247             (p[t]-p[i])) t=(t+1)%n;
248         ans=max(ans,(p[i]-p[t]).abs2());
249     }
250     return p.pop_back(),ans;
251 }
252 T min_cover_rectangle() const{ //最小覆蓋矩形

```

```

211 int n=p.size(),t=1,r=1,l;
212 if(n<3)return 0;//也可以做最小周長矩形
213 T ans=1e99;p.push_back(p[0]);
214 for(int i=0;i<n;i++){
215     point<T> now=p[i+1]-p[i];
216     while(now.cross(p[t+1]-p[i])>now.cross
217           (p[t]-p[i]))t=(t+1)%n;
218     while(now.dot(p[r+1]-p[i])>now.dot(p[r]
219           -p[i]))r=(r+1)%n;
220     if(!i)l=r;
221     while(now.dot(p[l+1]-p[i])<=now.dot(p[
222           l]-p[i]))l=(l+1)%n;
223     T d=now.abs2();
224     T tmp=now.cross(p[t]-p[i])*(now.dot(p[
225           r]-p[i])-now.dot(p[l]-p[i]))/d;
226     ans=min(ans,tmp);
227 }
228 return p.pop_back(),ans;
229
230 T dis2(polygon &p1){//凸包最近距離平方
231     vector<point<T>> > &P=p,&Q=p1.p;
232     int n=P.size(),m=Q.size(),l=0,r=0;
233     for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
234     for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
235     P.push_back(P[0]),Q.push_back(Q[0]);
236     T ans=1e99;
237     for(int i=0;i<n;++i){
238         while((P[l+1]-P[l+1]).cross(Q[r+1]-Q[r])
239               <0)r=(r+1)%m;
240         ans=min(ans,line<T>(P[l],P[l+1]).
241               seg_dis2(line<T>(Q[r],Q[r+1])));
242         l=(l+1)%n;
243     }
244     return P.pop_back(),Q.pop_back(),ans;
245 }
246
247 static char sign(const point<T>&t){
248     return (t.y==0?t.x:t.y)<0;
249 }
250
251 static bool angle_cmp(const line<T>& A,
252                       const line<T>& B){
253     point<T> a=A.p2-A.p1,b=B.p2-B.p1;
254     return sign(a)<sign(b)||((sign(a)==sign(b)
255                               )&&a.cross(b)>0);
256 }
257
258 int halfplane_intersection(vector<line<T>
259                             > &s){//半平面交
260     sort(s.begin(),s.end(),angle_cmp);//線段
261     //左側為該線段半平面
262     int L,R,n=s.size();
263     vector<point<T>> > px(n);
264     vector<line<T>> > q(n);
265     q[L=R=0]=s[0];
266     for(int i=1;i<n;++i){
267         while(L<R&&s[i].ori(px[R-1])<=0)--R;
268         while(L<R&&s[i].ori(px[L])<=0)++L;
269         q[++R]=s[i];
270         if(q[R].parallel(q[R-1])){
271             --R;
272             if(q[R].ori(s[i].p1)>0)q[R]=s[i];
273         }
274         if(L<R)px[R-1]=q[R-1].
275             line_intersection(q[R]);
276     }
277     while(L<R&&q[L].ori(px[R-1])<=0)--R;
278     p.clear();
279
280     if(R-L<=1)return 0;
281     px[R]=q[R].line_intersection(q[L]);
282     for(int i=L;i<R;++i)p.push_back(px[i]);
283     return R-L+1;
284 }
285
286 template<typename T>
287 struct triangle{
288     point<T> a,b,c;
289     triangle(){
290         triangle(const point<T> &a,const point<T>
291                 &b,const point<T> &c):a(a),b(b),c(c){
292             T area()const{
293                 T t=(b-a).cross(c-a)/2;
294                 return t>0?t:-t;
295             }
296     }
297     point<T> barycenter()const{//重心
298         return (a+b+c)/3;
299     }
300     point<T> circumcenter()const{//外心
301         static line<T> u,v;
302         u.p1=(a+b)/2;
303         u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
304                       b.x);
305         v.p1=(a+c)/2;
306         v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
307                       c.x);
308         return u.line_intersection(v);
309     }
310     point<T> incenter()const{//內心
311         T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
312               ()),C=sqrt((a-b).abs2());
313         return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
314                       B*b.y+C*c.y)/(A+B+C);
315     }
316     point<T> perpencenter()const{//垂心
317         return barycenter()*3-circumcenter()*2;
318     }
319 }
320
321 template<typename T>
322 struct point3D{
323     T x,y,z;
324     point3D(){
325         point3D(const T&x,const T&y,const T&z):x(x
326               ),y(y),z(z){
327             point3D operator+(const point3D &b)const{
328                 return point3D(x+b.x,y+b.y,z+b.z);
329             }
330             point3D operator-(const point3D &b)const{
331                 return point3D(x-b.x,y-b.y,z-b.z);
332             }
333             point3D operator*(const T &b)const{
334                 return point3D(x*b,y*b,z*b);
335             }
336             point3D operator/(const T &b)const{
337                 return point3D(x/b,y/b,z/b);
338             }
339             bool operator==(const point3D &b)const{
340                 return x==b.x&&y==b.y&&z==b.z;
341             }
342             T dot(const point3D &b)const{
343                 return x*b.x+y*b.y+z*b.z;
344             }
345             point3D cross(const point3D &b)const{
346                 return point3D(y*b.z-z*b.y,z*b.x-x*b.z,
347                               x*b.y-y*b.x);
348             }
349             T abs2()const{//向量長度的平方
350                 return dot(*this);
351             }
352             T area2(const point3D &b)const{//和b、原點
353                 //圍成面積的平方
354                 return cross(b).abs2()/4;
355             }
356 };
357
358 template<typename T>
359 struct line3D{
360     point3D<T> p1,p2;
361     line3D(){
362         line3D(const point3D<T> &p1,const point3D<
363               T> &p2):p1(p1),p2(p2){
364             T dis2(const point3D<T> &p,bool is_segment
365                   =0)const{//點跟直線/線段的距離平方
366                 point3D<T> v=p2-p1,v1=p-p1;
367                 if(is_segment){
368                     point3D<T> v2=p-p2;
369                     if(v.dot(v1)<=0)return v1.abs2();
370                     if(v.dot(v2)>=0)return v2.abs2();
371                 }
372                 point3D<T> tmp=v.cross(v1);
373                 return tmp.abs2()/v.abs2();
374             }
375             pair<point3D<T>,point3D<T>> closest_pair(
376                 const line3D<T> &l1)const{
377                 point3D<T> v1=(p1-p2),v2=(l1.p1-l.p2);
378                 point3D<T> N=v1.cross(v2),ab(p1-l.p1);
379                 //if(N.abs2()==0)return NULL;平行或重合
380                 T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
381                 //最近點對距離
382                 point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
383                     cross(d2),G=l.p1-p1;
384                 T t1=(G.cross(d2)).dot(D)/D.abs2();
385                 T t2=(G.cross(d1)).dot(D)/D.abs2();
386                 return make_pair(p1+d1*t1,l.p1+d2*t2);
387             }
388             bool same_side(const point3D<T> &a,const
389                             point3D<T> &b)const{
390                 return (p2-p1).cross(a-p1).dot((p2-p1).
391                       cross(b-p1))>0;
392             }
393 };
394
395 template<typename T>
396 struct plane{
397     point3D<T> p0,n;//平面上的點和法向量
398     plane(){
399         plane(const point3D<T> &p0,const point3D<T>
400               &n):p0(p0),n(n){
401             T dis2(const point3D<T> &p)const{//點到平
402                   面距離的平方
403                 T tmp=(p-p0).dot(n);
404                 return tmp*tmp/n.abs2();
405             }
406             point3D<T> projection(const point3D<T> &p)
407                 const{
408                 return p-n*(p-p0).dot(n)/n.abs2();
409             }
410             point3D<T> line_intersection(const line3D<
411                   T> &l1)const{
412                 T tmp=n.dot(l1.p2-l.p1);//等於0表示平行或
413                   重合該平面
414                 return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
415                       tmp);
416             }
417             line3D<T> plane_intersection(const plane &
418                   p1)const{
419                 point3D<T> e=n.cross(p1.n),v=n.cross(e);
420                 T tmp=p1.n.dot(v);//等於0表示平行或重合
421                   該平面
422             }
423 };
424
425 point3D<T> q=p0+(v*(p1.n.dot(p1.p0-p0))/
426               tmp);
427     return line3D<T>(q,q+e);
428 }
429
430 const double eps = 1e-10;
431 int sign(double a){
432     return fabs(a)<eps?0:a>0?1:-1;
433 }
434
435 template<typename T>
436 T len(point<T> p){
437     return sqrt(p.dot(p));
438 }
439
440 template<typename T>
441 point<T> findCircumcenter(point<T> A,point<T>
442                           B,point<T> C){
443     point<T> AB = B-A;
444     point<T> AC = C-A;
445     T AB_len_sq = AB.x*AB.x+AB.y*AB.y;
446     T AC_len_sq = AC.x*AC.x+AC.y*AC.y;
447     T D = AB.x*AC.y-AB.y*AC.x;
448     T X = A.x+(AC.y*AB_len_sq-AB.y*AC_len_sq)
449           /(2*D);
450     T Y = A.y+(AB.x*AC_len_sq-AC.x*AB_len_sq)
451           /(2*D);
452     return point<T>(X,Y);
453 }
454
455 template<typename T>
456 pair<T, point<T>> MinCircleCover(vector<
457     point<T>> &p){
458     // 回傳最小覆蓋圓{半徑, 中心}
459     random_shuffle(p.begin(),p.end());
460     int n = p.size();
461     point<T> c = p[0]; T r = 0;
462     for(int i=1;i<n;i++){
463         if(sign(len(c-p[i])-r) > 0){ // 不在圓內
464             c = p[i], r = 0;
465             for(int j=0;j<i;j++){
466                 if(sign(len(c-p[j])-r) > 0) {
467                     c = (p[i]+p[j])/2.0;
468                     r = len(c-p[i]);
469                     for(int k=0;k<j;k++){
470                         if(sign(len(c-p[k])-r) > 0){
471                             //c=triangle<T>(p[i],p[j],p[k]).
472                             //circumcenter();
473                             c = findCircumcenter(p[i],p[j]
474                                     ],p[k]);
475                             r = len(c-p[i]);
476                         }
477                     }
478                 }
479             }
480         }
481     }
482     return make_pair(r,c);
483 }

```

1.3 最近點對

```

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

```

2 DP

2.1 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(
21         ),x);
22     if(it == dp.end()) {
23         dp.emplace_back(x);
24     }
25     else {
26         *it = x;
27     }
28 }

```

```

28 cout<<dp.size();
29 //LCS問題
30 #include<bits/stdc++.h>
31 using namespace std;
32 signed main() {
33     string a,b;
34     cin>>a>>b;
35     vector<vector<int>> dp(a.size()+1,vector
36         <int> (b.size()+1,0));
37     vector<vector<pair<int,int>> pre(a.size(
38         )+1,vector<pair<int,int>> (b.size()
39         +1));
40     for(int i=0;i<a.size();i++){
41         for(int j=0;j<b.size();j++){
42             if(a[i] == b[j]) {
43                 dp[i+1][j+1] = dp[i][j] + 1;
44                 pre[i+1][j+1] = {i,j};
45             }
46             else if(dp[i+1][j] >= dp[i][j+1]) {
47                 dp[i+1][j+1] = dp[i+1][j];
48                 pre[i+1][j+1] = {i+1,j};
49             }
50             else {
51                 dp[i+1][j+1] = dp[i][j+1];
52                 pre[i+1][j+1] = {i,j+1};
53             }
54         }
55     }
56     int index1 = a.size(), index2 = b.size()
57     ;
58     string ans;
59     while(index1>0&&index2>0) {
60         if(pre[index1][index2] == make_pair(
61             index1-1,index2-1)) {
62             ans+=a[index1-1];
63         }
64         pair<int,int> u = pre[index1][index2]
65         ;
66         index1= u.first;
67         index2= u.second;
68     }
69     for(int i=ans.size()-1;i>=0;i--)cout<<
70     ans[i];
71     return 0;
72 }

```

2.2 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; //
36     葉子
37     for (auto v : Tree[u]) {
38         if (pick == 0) {
39             DP[pick][u] += max(dfs(v, 0, u), dfs(v
40                 , 1, u));
41         }
42         else {
43             DP[pick][u] += dfs(v, 0, u);
44         }
45     }
46     return DP[pick][u] += pick;
47 }
48 int solve(int n) {
49     DP[0] = DP[1] = vector<int>(n+1, 0);
50     return max(dfs(1, 0), dfs(1, 1));
51 }
52 //Traveling Salesman // AtCoder
53 #include<bits/stdc++.h>
54 using namespace std;
55 const int INF = 1e9;
56 int cost(vector<tuple<int,int,int>> &point,
57     int from, int to) {
58     auto [x,y,z] = point[from];
59     auto [X,Y,Z] = point[to];
60     return abs(X-x)+abs(Y-y)+max(0,Z-z);
61 }
62 //從一個點走到另一個點的花費
63 signed main() {
64     int n;cin>>n;
65     vector<tuple<int,int,int>> point(n);
66     for(auto &[x,y,z]:point) {
67         cin>>x>>y>>z;
68     }
69     vector<vector<int>> dp(1<n,vector<int>
70         (n,INF));
71     //1<n(2^n)代表1~n的所有子集，代表走過的
72     點
73     //n代表走到的最後一個點
74     dp[0][0] = 0;
75     for(int i=1;i<(1<n);i++){
76         for(int j=0;j<n;j++){
77             if(i & (1<j)) {
78                 //j是走到的最後一個點，必須
79                 要在i裡面
80                 for(int k=0;k<n;k++){
81                     dp[i][j] = min(dp[i][j],
82                         dp[i-(1<j)][k]+cost
83                         (point,k,j));
84                     //i集合裡面走到j = i/{j}
85                     集合裡走到k，再從k走
86                     到j
87                 }
88             }
89             //cout<<dp[i][j]<<' ';
90         }
91         //cout<<endl;
92     }
93     cout<<dp[(1<n)-1][0];//每個都要走到，要
94     走回1
95     return 0;
96 }

```

2.3 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)
18     const { return k < o.k; }
19 inline bool operator<(long long x) const {
20     return p < x; }
21 };
22 // Line_container_internal
23 template<bool MAX>
24 struct line_container : std::multiset<
25     line_container_internal::line_t, std:::
26     less<>> {
27     static const long long INF = std:::
28         numeric_limits<long long>::max();
29 }
30 bool isect(iterator x, iterator y) {
31     if(y == end()) {
32         x->p = INF;
33         return 0;
34     }
35     if(x->k == y->k) {
36         x->p = (x->m > y->m ? INF : -INF);
37     }
38 }

```

```

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 =
46         y;
47     while(isect(y, z)) {
48         z = erase(z);
49     }
50     if(x != begin() && isect(--x, y)) {
51         isect(x, y = erase(y));
52     }
53     while((y = x) != begin() && (--x)->p >=
54         y->p) {
55         isect(x, erase(y));
56     }
57 }
58
59 long long get(long long x) {
60     assert(!empty());
61     auto l = *lower_bound(x);
62     return (l.k * x + l.m) * (MAX ? +1 : -1);
63 }
64 }
65 }

```

2.4 單調隊列優化

```

1 long long solve(vector<int> a, int N, int K)
2 {
3     vector<long long> DP(N + 1);
4     deque<int> dq(1);
5     for (int i = 1; i <= N; ++i) {
6         while (dq.front() < i - K)
7             dq.pop_front();
8         DP[i] = DP[dq.front()] + a[i];
9         while (dq.size() && DP[dq.back()] > DP[i])
10             dq.pop_back();
11         dq.push_back(i);
12     }
13     long long ans = INF;
14     for (int i = N - K + 1; i <= N; ++i)
15         ans = min(ans, DP[i]);
16     return ans;
17 }

```

2.5 整體二分

```

1 void compute(int L, int R, int optL, int
2     optR) {
3     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)
8         ; k++) {
9         if (DP[mid] > f(k) + w(k, mid)) {
10             DP[mid] = f(k) + w(k, mid);
11             opt = k;
12         }
13     }
14     compute(L, mid - 1, optL, opt);
15     compute(mid + 1, R, opt, optR);
16 }

```

2.6 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)
5         , l(_l) {}
6     bool operator<(const Line &rhs) const
7     {
8         return make_pair(-a, -b) < make_pair
9             (-rhs.a, -rhs.b);
10     }
11     bool operator<(ll rhs_l) const
12     {
13         return l < rhs_l;
14     }
15 };
16
17 struct ConvexHullMin : std::multiset<Line,
18     std::less<>>
19 {
20     static const ll INF = (1ll << 60);
21     static ll DivCeil(ll a, ll b)
22     {
23         return a / b - ((a ^ b) < 0 && a % b
24             );
25     }
26     bool Intersect(iterator x, iterator y)
27     {
28         if (y == end())
29             return false;
30         {
31             x->l = INF;
32             return false;
33         }
34         if (x->a == y->a)
35         {
36             x->l = x->b < y->b ? INF : -INF;
37         }
38         else
39         {
40             x->l = DivCeil(y->b - x->b, x->a
41                 - y->a);
42         }
43         return x->l >= y->l;
44     }
45     void Insert(ll a, ll b)
46     {

```

```

47         auto z = insert(Line(a, b, 0)), y =
48             z++, x = y;
49         while (Intersect(y, z))
50             z = erase(z);
51         if (x != begin() && Intersect(--x, y
52             ))
53             Intersect(x, y = erase(y));
54         while ((y = x) != begin() && (--x)->
55             l >= y->l)
56             Intersect(x, erase(y));
57     }
58     ll query(ll x) const
59     {
60         auto l = *lower_bound(x);
61         return l.a * x + l.b;
62     }
63 }
64
65 convexhull;
66
67 const ll maxn = 200005;
68 ll s[maxn];
69 ll f[maxn];
70 ll dp[maxn];
71 // CSES monster game2
72 int main()
73 {
74     Crbubble
75     ll n, m, i, k, t;
76     cin >> n >> f[0];
77     for(i=1; i<=n; i++) cin >> s[i];
78     for(i=1; i<=n; i++) cin >> f[i];
79     convexhull.Insert(f[0], 0);
80     for(i=1; i<=n; i++)
81     {
82         dp[i] = convexhull.query(s[i]);
83         convexhull.Insert(f[i], dp[i]);
84     }
85     cout << dp[n] << endl;
86     return 0;
87 }

```

3 Data Structure

3.1 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);
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     }

```

```

19     return sum;
20 }
21 void modify(int x, int val) {
22     for(; x <= n; x += lowbit(x))
23         bit[x] += val;
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);
34         for(auto &b : bit1D) b.init(_n);
35     }
36     int query(int x, int y) const {
37         int sum = 0;
38         for(; x <= n; x += lowbit(x))
39             sum += bit1D[x].query(y);
40         return sum;
41     }
42     void modify(int x, int y, int val) {
43         for(; x <= m; x += lowbit(x))
44             bit1D[x].modify(y, val);
45     }
46 }
47 }

```

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 int findK(vector<int> &bit, int k) {
17     int idx = 0, res = 0;
18     int mx = __lg(bit.size()) + 1;
19     for(int i = mx; i >= 0; i--) {
20         if((idx | (1<<i)) > bit.size()) continue
21         ;
22         if(res + bit[idx | (1<<i)] < k) {
23             idx = (idx | (1<<i));
24             res += bit[idx];
25         }
26     }
27     return idx + 1;
28 }
29
30 //O(n)建bit
31 for (int i = 1; i <= n; ++i) {
32     bit[i] += a[i];
33     int j = i + lowbit(i);
34     if (j <= n) bit[j] += bit[i];
35 }

```

3.4 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     }
20 };

```

3.5 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     node(int v = 0): sum(v) {l = r = nullptr;}
9 };
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
21 ll qry(node* o, int ql, int qr, int l, int r
22     ) {
23     if(!o) return 0;
24     if(ql <= l && r <= qr) return o->sum;
25     int m = (l + r) / 2; ll ret = 0;
26     if(ql <= m) ret += qry(o->l, ql, qr, l, m)
27     ;
28     if(qr > m) ret += qry(o->r, ql, qr, m+1, r
29     );
30     return ret;
31 }

```

3.6 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.unite(s, t);
13         }
14     }
15     return cost;
16 }

```

3.7 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 ll seg[N << 2], tag[N << 2];
6 int n;
7
8 void pull(int id) {
9     seg[id] = seg[lc(id)] + seg[rc(id)];
10 }
11
12 void push(int id, int l, int r) {
13     if (tag[id]) {
14         int m = (l + r) >> 1;
15         tag[lc(id)] += tag[id], tag[rc(id)] +=
16             tag[id];
17         seg[lc(id)] += (m - l + 1) * tag[id],
18             seg[rc(id)] += (r - m) * tag[id];
19         tag[id] = 0;
20     }
21 }
22
23 void upd(int ql, int qr, ll v, int l = 1,
24     int r = n, int id = 1) {
25     if (ql <= l && r <= qr) return tag[id] +=
26         v, seg[id] += (r - l + 1) * v, void();
27     push(id, l, r);
28     int m = (l + r) >> 1;
29     if (ql <= m) upd(ql, qr, v, l, m, lc(id));
30     if (qr > m) upd(ql, qr, v, m + 1, r, rc(id));
31     pull(id);
32 }
33
34 ll qry(int ql, int qr, int l = 1, int r = n,
35     int id = 1) {
36     if (ql <= l && r <= qr) return seg[id];
37     push(id, l, r);
38     int m = (l + r) >> 1; ll ret = 0;
39     if (ql <= m) ret += qry(ql, qr, l, m, lc(
40         id));
41     if (qr > m) ret += qry(ql, qr, m + 1, r,
42         rc(id));
43     return ret;
44 }

```

3.8 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 }

```

3.9 monotonic stack

```

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

```

3.10 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<
7     T>, rb_tree_tag,
8     tree_order_statistics_node_update>;
9
10 template <class T>
11 // ordered_multiset: do not use erase method
12 // use myerase() instead
13 using ordered_multiset = tree<T, null_type,
14     less_equal<T>, rb_tree_tag,
15     tree_order_statistics_node_update>;
16
17 template<class T>
18 void myerase(ordered_multiset<T> &ss, T v)
19 {
20     T rank = ss.order_of_key(v); //
21     // Number of elements that are less
22     // than v in ss
23     auto it = ss.find_by_order(rank); //
24     // Iterator that points to the element
25     // which index = rank
26     ss.erase(it);
27 }

```

3.11 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 };

```

3.12 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,
15        int l = 1, int r = n) {
16        if (l == r) return cur->sum = v, void();
17        int m = (l + r) >> 1;
18        if (x <= m) cur->r = prv->r, upd(prv->l,
19            cur->l = new node(x, v, l, m);
20        else cur->l = prv->l, upd(prv->r, cur->r =
21            new node(x, v, m + 1, r);
22        cur->pull();
23    }
24
25    ll qry(node* a, node* b, int ql, int qr, int
26        l = 1, int r = n) {
27        if (ql <= l && r <= qr) return b->sum - a
28            ->sum;
29        int m = (l + r) >> 1; ll ret = 0;
30        if (ql <= m) ret += qry(a->l, b->l, ql, qr,
31            l, m);
32        if (qr > m) ret += qry(a->r, b->r, ql, qr,
33            m + 1, r);
34        return ret;
35    }

```

3.13 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        res += mincost[v]; //加上邊的權重
25        for(int u = 0; u < V; u++) {
26            mincost[u] = min(mincost[u], cost
27                [v][u]);
28        }
29    }
30    return res;
31 }

```

3.14 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 int query(int id, int l, int r, int ql, int
19     qr) {

```

```

20     if (r < ql || qr < l) return 0; //若目前
21     // 的區間與詢問的區間的交集為空的話 ·
22     // return 0
23     if (ql <= l && qr <= r) return seg[id];
24     //若目前的區間是詢問的區間的子集的
25     // 話 · 則終止 · 並回傳當前節點的答案
26     int mid = (l + r) / 2;
27     return query(id * 2, l, mid, ql, qr) //
28         // 左
29         + query(id * 2 + 1, mid + 1, r, ql,
30             qr); // 右
31     //否則 · 往左 · 右進行遞迴
32 }
33
34 //單點修改
35 void modify(int id, int l, int r, int i, int
36     x) {
37     if (l == r) {
38         seg[id] = x; // 將a[i]改成x
39         //seg[id] += x; // 將a[i]加上x
40         return;
41     }
42     int mid = (l + r) / 2;
43     // 根據修改的點在哪裡 · 來決定要往哪個子
44     // 樹進行DFS
45     if (i <= mid) modify(id * 2, l, mid, i,
46         x); // 左
47     else modify(id * 2 + 1, mid + 1, r, i, x);
48     // 右
49     seg[id] = seg[id * 2] + seg[id * 2 + 1];
50 }

```

3.15 sparse table

```

1 //CSES Static Range Minimum Queries
2 #include<bits/stdc++.h>
3 using namespace std;
4 #define inf 1e9
5 vector<vector<int>> st;
6
7 void build_sparse_table(int n) {
8     st.assign(__lg(n)+1, vector<int>(n+1, inf));
9     for(int i=1; i<=n; i++) cin>>st[0][i];
10    for(int i=1; (1<<i)<=n; i++) {
11        for(int j=1; j+(1<<i)-1 <= n; j++) {
12            st[i][j] = min(st[i-1][j], st[i-1][j
13                +(1<<(i-1))]);
14        }
15    }
16 }
17 int query(int l, int r) {
18     int k = __lg(r - l + 1);
19     return min(st[k][l], st[k][r-(1<<k)+1]);
20 }
21
22 signed main() {
23     int n, q; cin>>n>>q;

```

```

24 build_sparse_table(n);
25 while(q--){
26     int l,r;cin>>l>>r;
27     cout<<query(l,r)<<'\n';
28 }
29 }

```

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 = arr[idx].r = l;
12            arr[idx].opt.clear();
13            return;
14        }
15        int m = (l+r)>>1;
16        build(l,m,idx<<1);
17        build(m+1,r,idx<<1|1);
18        arr[idx].l = l,arr[idx].r = r;
19        arr[idx].opt.clear();
20    }
21    void update(int ql,int qr,T k,int idx = 1)
22    {
23        if(ql<=arr[idx].l and arr[idx].r<=qr){
24            arr[idx].opt.push_back(k);
25            return;
26        }
27        int m = (arr[idx].l+arr[idx].r)>>1;
28        if(ql<=m)update(ql,qr,k,idx<<1);
29        if(qr>m)update(ql,qr,k,idx<<1|1);
30    }
31    void dfs(D &d,vector<int>&ans,int idx = 1)
32    {
33        int cnt = 0;
34        for(auto [a,b]:arr[idx].opt){
35            if(d.Union(a,b))cnt++;
36        }
37        if(arr[idx].l==arr[idx].r)ans[arr[idx].l
38            ] = d.comps;
39        else{
40            dfs(d,ans,idx<<1);
41            dfs(d,ans,idx<<1|1);
42        }
43        while(cnt-->0)d.undo();
44    }
45 };

```

3.17 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 };

```

3.18 掃描線 + 線段樹

```

1 //CSES Area of Rectangle
2 #include <bits/stdc++.h>
3 #define pb push_back
4 #define int long long
5 #define mid ((l + r) >> 1)
6 #define lc (p << 1)
7 #define rc ((p << 1) | 1)
8 using namespace std;
9 struct ooo{
10     int x, l, r, v;
11 };
12 const int inf = 1e6;
13 array<int, 800004> man, tag, cnt;
14 vector<ooo> Q;
15 bool cmp(ooo a, ooo b){
16     return a.x < b.x;
17 }
18 void pull(int p){
19     man[p] = min(man[lc], man[rc]);
20     if(man[lc] < man[rc]) cnt[p] = cnt[lc];
21     else if(man[rc] < man[lc]) cnt[p] = cnt[rc];
22     else cnt[p] = cnt[lc] + cnt[rc];
23 }
24 void push(int p){
25     man[lc] += tag[p];
26     man[rc] += tag[p];
27     tag[lc] += tag[p];
28     tag[rc] += tag[p];
29     tag[p] = 0;
30 }
31 void build(int p, int l, int r){

```

```

32     if(l == r){
33         cnt[p] = 1;
34         return;
35     }
36     build(lc, l, mid);
37     build(rc, mid + 1, r);
38     pull(p);
39 }
40 void update(int p, int l, int r, int ql, int
41     qr, int x){
42     if(ql > r || qr < l) return;
43     if(ql <= l && qr >= r){
44         man[p] += x;
45         tag[p] += x;
46         return;
47     }
48     push(p);
49     update(lc, l, mid, ql, qr, x);
50     update(rc, mid + 1, r, ql, qr, x);
51     pull(p);
52 }
53 signed main(){
54     int n, x1, y1, x2, y2, p = 0, sum = 0;
55     cin >> n;
56     for(int i = 1; i <= n; i++){
57         cin >> x1 >> y1 >> x2 >> y2;
58         Q.pb({x1, y1, y2 - 1, 1});
59         Q.pb({x2, y1, y2 - 1, -1});
60     }
61     sort(Q.begin(), Q.end(), cmp);
62     build(1, -inf, inf);
63     for(int i = -inf; i < inf; i++){
64         while(p < Q.size() && Q[p].x == i){
65             auto [x, l, r, v] = Q[p++];
66             update(l, -inf, inf, l, r, v);
67         }
68         sum += 2 * inf + 1 - cnt[1];
69     }
70     cout << sum << "\n";
71     return 0;
72 }
73 //長方形面積
74 long long AreaOfRectangles(vector<tuple<int,
75     int,int,int>>>v){
76     vector<tuple<int,int,int,int>>tmp;
77     int L = INT_MAX,R = INT_MIN;
78     for(auto [x1,y1,x2,y2]:v){
79         tmp.push_back({x1,y1+1,y2,1});
80         tmp.push_back({x2,y1+1,y2,-1});
81         R = max(R,y2);
82         L = min(L,y1);
83     }
84     vector<long long>seg((R-L+1)<<2),tag((R-L
85         +1)<<2);
86     sort(tmp.begin(),tmp.end());
87     function<void(int,int,int,int,int)>
88         update = [&](int ql,int qr,int val,int
89             l,int r,int idx){
90         if(ql<=l and r<=qr){
91             tag[idx]+=val;
92             if(tag[idx])seg[idx] = r-l+1;
93             else if(l==r)seg[idx] = 0;
94             else seg[idx] = seg[idx<<1]+seg[idx
95                 <<1|1];
96             return;
97         }
98         push(p);
99         update(lc, l, mid, ql, qr, val);
100        update(rc, mid + 1, r, ql, qr, val);
101        pull(p);
102    }
103    signed main(){
104        int n, x1, y1, x2, y2, p = 0, sum = 0;
105        cin >> n;
106        for(int i = 0; i < n; i++){
107            cin >> x1 >> y1 >> x2 >> y2;
108            x1 += inf, x2 += inf, y1 += inf, y2
109                += inf;
110            if(x1 == x2) Q.pb({x1, y1, y2});
111            else A.pb({x1, y1, 1}), A.pb({x2 +
112                1, y2, -1});
113        }
114        sort(Q.begin(), Q.end(), cmp);

```

```

91     }
92     int m = (l+r)>>1;
93     if(ql<=m)update(ql,qr,val,l,m,idx<<1);
94     if(qr>m)update(ql,qr,val,m+1,r,idx<<1|1)
95         ;
96     if(tag[idx])seg[idx] = r-l+1;
97     else seg[idx] = seg[idx<<1]+seg[idx
98         <<1|1];
99 }
100 long long last_pos = 0,ans = 0;
101 for(auto [pos,l,r,val]:tmp){
102     ans+=(pos-last_pos)*seg[l];
103     update(l,r,val,L,R,1);
104     last_pos = pos;
105 }
106 return ans;
107 }
108 // CSES Intersection Points
109 #include <bits/stdc++.h>
110 #define int long long
111 #define pb push_back
112 using namespace std;
113 struct line{
114     int p, l, r;
115 };
116 const int inf = 1e6 + 1;
117 array<int, 200004> BIT;
118 vector<line> A, Q;
119 bool cmp(line a, line b){
120     return a.p < b.p;
121 }
122 void update(int p, int x){
123     for(; p < 200004; p += p & -p) BIT[p]
124         += x;
125 }
126 int query(int p){
127     int sum = 0;
128     for(; p; p -= p & -p) sum += BIT[p];
129     return sum;
130 }
131 int run(){
132     int ans = 0, p = 0;
133     for(auto [t, l, r] : Q){
134         while(p < A.size()){
135             auto [x, y, v] = A[p];
136             if(x > t) break;
137             update(y, v);
138             p++;
139         }
140         ans += query(r) - query(l - 1);
141     }
142     return ans;
143 }
144 signed main(){
145     int n, x1, x2, y1, y2;
146     cin >> n;
147     for(int i = 0; i < n; i++){
148         cin >> x1 >> y1 >> x2 >> y2;
149         x1 += inf, x2 += inf, y1 += inf, y2
150             += inf;
151         if(x1 == x2) Q.pb({x1, y1, y2});
152         else A.pb({x1, y1, 1}), A.pb({x2 +
153             1, y2, -1});
154     }
155     sort(Q.begin(), Q.end(), cmp);

```

```

152 sort(A.begin(), A.end(), cmp);
153 cout << run() << "\n";
154 return 0;
155 }

```

3.19 陣列上 Treap

```

1 struct Treap {
2     Treap *lc = nullptr, *rc = nullptr;
3     unsigned pri, sz;
4     long long Val, Sum;
5     Treap(int Val):pri(rand()),sz(1),Val(Val),
6         Sum(Val),Tag(false) {}
7     void pull();
8     bool Tag;
9     void push();
10 } *root;
11
12 inline unsigned sz(Treap *x) {
13     return x ? x->sz:0;
14 }
15
16 inline void Treap::push() {
17     if(!Tag) return;
18     swap(lc,rc);
19     if(lc) lc->Tag ^= Tag;
20     if(rc) rc->Tag ^= Tag;
21     Tag = false;
22 }
23
24 inline void Treap::pull() {
25     sz = 1;
26     Sum = Val;
27     if(lc) {
28         sz += lc->sz;
29         Sum += lc->Sum;
30     }
31     if(rc) {
32         sz += rc->sz;
33         Sum += rc->Sum;
34     }
35 }
36
37 Treap *merge(Treap *a, Treap *b) {
38     if(!a || !b) return a ? a : b;
39     if(a->pri < b->pri) {
40         a->push();
41         a->rc = merge(a->rc,b);
42         a->pull();
43         return a;
44     }
45     else {
46         b->push();
47         b->lc = merge(a,b->lc);
48         b->pull();
49         return b;
50     }
51 }
52
53 pair<Treap *,Treap *> splitK(Treap *x,
54     unsigned K) {

```

```

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

```

4 Flow

4.1 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     queue<int> que;
23     que.push(s);
24     while(!que.empty()) {
25         int u = que.front();
26         que.pop();
27         for(auto id : g[u]) {
28             const edge& e = edges[id];
29             int v = e.to;
30             if(e.cap > 0 && h[v] == -1) {
31                 h[v] = h[u] + 1;
32                 if(v == t) {
33                     return 1;
34                 }
35                 que.push(v);
36             }
37         }
38     }
39     return 0;
40 }
41 T dfs(int u, int t, T f) {
42     if(u == t) {
43         return f;
44     }
45     T r = f;
46     for(int& i = cur[u]; i < (int) g[u].size
47         (); ++i) {
48         int id = g[u][i];
49         const edge& e = edges[id];
50         int v = e.to;
51         if(e.cap > 0 && h[v] == h[u] + 1) {
52             T send = dfs(v, t, min(r, e.cap));
53             edges[id].cap -= send;
54             edges[id ^ 1].cap += send;
55             r -= send;
56             if(r == 0) {
57                 return f;
58             }
59         }
60     }
61     return f - r;
62 }
63 T flow(int s, int t, T f = numeric_limits<
64     T>::max()) {
65     T ans = 0;
66     while(f > 0 && bfs(s, t)) {
67         cur.assign(n, 0);
68         T send = dfs(s, t, f);
69         ans += send;
70         f -= send;
71     }
72     return ans;
73 }
74 vector<pair<int,int>> min_cut(int s) {
75     vector<bool> vis(n);
76     vis[s] = true;
77     queue<int> que;
78     que.push(s);
79     while(!que.empty()) {

```

```

80     int u = que.front();
81     que.pop();
82     for(auto id : g[u]) {
83         const auto& e = edges[id];
84         int v = e.to;
85         if(e.cap > 0 && !vis[v]) {
86             vis[v] = true;
87             que.push(v);
88         }
89     }
90 }
91 vector<pair<int,int>> cut;
92 for(int i = 0; i < (int) edges.size(); i
93     += 2) {
94     const auto& e = edges[i];
95     if(vis[e.from] && !vis[e.to]) {
96         cut.push_back(make_pair(e.from, e.to
97             ));
98     }
99 }
100 return cut;
101 };
102 //CSES Distinct Routes
103 #include <bits/stdc++.h>
104 using namespace std;
105 struct FlowEdge {
106     int v, u;
107     long long cap, flow = 0;
108     FlowEdge(int v, int u, long long cap) :
109         v(v), u(u), cap(cap) {}
110 };
111 struct Dinic {
112     const long long flow_inf = 1e18;
113     vector<FlowEdge> edges;
114     vector<vector<int>> adj;
115     int n, m = 0;
116     int s, t;
117     vector<int> level, ptr, path;
118     vector< vector<int> > paths;
119     queue<int> q;
120
121     Dinic(int n, int s, int t) : n(n), s(s),
122         t(t) {
123         adj.resize(n);
124         level.resize(n);
125         ptr.resize(n);
126     }
127     void add_edge(int v, int u, long long
128         cap) {
129         edges.emplace_back(v, u, cap);
130         edges.emplace_back(u, v, 0);
131         adj[v].push_back(m);
132         adj[u].push_back(m + 1);
133         m += 2;
134     }
135     bool bfs() {
136         while(!q.empty()) {
137             int v = q.front();
138             q.pop();

```



```

139     for (int id : adj[v]) {
140         if (edges[id].cap - edges[id]
141             .flow < 1)
142             continue;
143         if (level[edges[id].u] !=
144             -1)
145             continue;
146         level[edges[id].u] = level[v]
147             + 1;
148         q.push(edges[id].u);
149     }
150     return level[t] != -1;
151 }
152 long long dfs(int v, long long pushed) {
153     if (pushed == 0)
154         return 0;
155     path.push_back(v);
156     if (v == t) {
157         for (int iiddxx = 0; iiddxx <
158             pushed; ++iiddxx)
159             paths.push_back(path);
160         path.pop_back();
161         return pushed;
162     }
163     for (int& cid = ptr[v]; cid < (int)
164         adj[v].size(); cid++) {
165         int id = adj[v][cid];
166         int u = edges[id].u;
167         if (level[v] + 1 != level[u] ||
168             edges[id].cap - edges[id].
169             flow < 1)
170             continue;
171         long long tr = dfs(u, min(pushed
172             , edges[id].cap - edges[id].
173             flow));
174         if (tr == 0)
175             continue;
176         edges[id].flow += tr;
177         edges[id ^ 1].flow -= tr;
178         path.pop_back();
179         return tr;
180     }
181     path.pop_back();
182     return 0;
183 }
184 long long flow() {
185     long long f = 0;
186     while (true) {
187         fill(level.begin(), level.end(),
188             -1);
189         level[s] = 0;
190         q.push(s);
191         if (!bfs())
192             break;
193         fill(ptr.begin(), ptr.end(), 0);
194         while (long long pushed = dfs(s,
195             flow_inf)) {
196             f += pushed;
197         }
198     }
199     return f;
200 }
201 };

```

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 ISAP with cut

```

1 template<typename T>
2 struct ISAP {
3     static const int MAXN=105;
4     static const T INF=INT_MAX;
5     int n; //點數
6     int d[MAXN], gap[MAXN], cur[MAXN];
7     struct edge {
8         int v, pre;
9         T cap, r;

```

```

10     edge(int v, int pre, T cap):v(v), pre(pre),
11         cap(cap), r(cap){}
12 };
13 int g[MAXN];
14 vector<edge> e;
15 void init(int _n) {
16     memset(g, -1, sizeof(int)*((n=_n)+1));
17     e.clear();
18 }
19 void add_edge(int u, int v, T cap, bool
20     directed=false) {
21     e.push_back(edge(v, g[u], cap));
22     g[u] = e.size()-1;
23     e.push_back(edge(u, g[v], directed?0:cap));
24     g[v] = e.size()-1;
25 }
26 T dfs(int u, int s, int t, T CF=INF) {
27     if (u==t) return CF;
28     T tf=CF, df;
29     for (int &i=cur[u]; ~i; i=e[i].pre) {
30         if (e[i].r && d[u]==d[e[i].v]+1) {
31             df=dfs(e[i].v, s, t, min(tf, e[i].r));
32             e[i].r -= df;
33             e[e[i]^1].r += df;
34             if (! (tf==df) || d[s]==n) return CF-tf;
35         }
36     }
37     int mh=n;
38     for (int i=cur[u]=g[u]; ~i; i=e[i].pre) {
39         if (e[i].r && d[e[i].v] < mh) mh=d[e[i].v];
40     }
41     if (!--gap[d[u]]) d[s]=n;
42     else ++gap[d[u]]=++mh;
43     return CF-tf;
44 }
45 T isap(int s, int t, bool clean=true) {
46     memset(d, 0, sizeof(int)*(n+1));
47     memset(gap, 0, sizeof(int)*(n+1));
48     memcpy(cur, g, sizeof(int)*(n+1));
49     if (clean) for (size_t i=0; i<e.size(); ++i)
50         e[i].r=e[i].cap;
51     T MF=0;
52     for (gap[0]=n; d[s]<n; ) MF+=dfs(s, s, t);
53     return MF;
54 }
55 vector<int> cut_e; //最小割邊集
56 bool vis[MAXN];
57 void dfs_cut(int u) {
58     vis[u]=1; //表示u屬於source的最小割集
59     for (int i=g[u]; ~i; i=e[i].pre)
60         if (e[i].r > 0 && !vis[e[i].v]) dfs_cut(e[i]
61             .v);
62 }
63 T min_cut(int s, int t) {
64     T ans=isap(s, t);
65     memset(vis, 0, sizeof(bool)*(n+1));
66     dfs_cut(s); cut_e.clear();
67     for (int u=0; u<n; ++u) if (vis[u])
68         for (int i=g[u]; ~i; i=e[i].pre)
69             if (!vis[e[i].v]) cut_e.push_back(i);
70     return ans;
71 }
72 };

```

4.4 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     static constexpr Cap_t EPS = static_cast<
14         Cap_t>(1e-9);
15     int n;
16     vector<Edge> edges;
17     vector<vector<int>>> g;
18     vector<Cost_t> d;
19     vector<bool> in_queue;
20     vector<int> previous_edge;
21     MCMF() {}
22     MCMF(int _n) : n(_n+1), g(_n+1), d(_n+1),
23         in_queue(_n+1), previous_edge(_n+1) {}
24     void add_edge(int u, int v, Cap_t cap,
25         Cost_t cost) {
26         assert(0 <= u && u < n);
27         assert(0 <= v && v < n);
28         g[u].push_back(edges.size());
29         edges.emplace_back(u, v, cap, cost);
30         g[v].push_back(edges.size());
31         edges.emplace_back(v, u, 0, -cost);
32     }
33     bool spfa(int s, int t) {
34         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     }

```

```

57     }
58   }
59   return found;
60 }
61
62 pair<Cap_t, Cost_t> flow(int s, int t,
63   Cap_t f = numeric_limits<Cap_t>::max())
64 {
65   assert(0 <= s && s < n);
66   assert(0 <= t && t < n);
67   Cap_t cap = 0;
68   Cost_t cost = 0;
69   while(f > 0 && spfa(s, t)) {
70     Cap_t send = f;
71     int u = t;
72     while(u != s) {
73       const Edge& e = edges[previous_edge[
74         u]];
75       send = min(send, e.cap);
76       u = e.from;
77     }
78     u = t;
79     while(u != s) {
80       Edge& e = edges[previous_edge[u]];
81       e.cap -= send;
82       Edge& b = edges[previous_edge[u] ^
83         1];
84       b.cap += send;
85       u = e.from;
86     }
87     cap += send;
88     f -= send;
89     cost += send * d[t];
90   }
91   return make_pair(cap, cost);
92 }

```

4.5 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

```

5 Graph

5.1 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  }
33 }

```

5.2 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      {
12        if (dist[u] > dist[v] + w)
13          {
14            dist[u] = dist[v] + w;
15            update = true;
16          }
17      }
18
19    if (!update)
20      break;
21    if (i == N) // && update
22      return -1; // gg !
23  }
24  return dist[e];
25 }

```

5.3 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   using T = tuple<int,int>;
6   priority_queue<T, vector<T>, greater<T>>
7     pq;
8
9   dist[s] = 0;
10  pq.emplace(0, s); // (w, e) 讓 pq 優先用
11    w 來比較
12
13  while (!pq.empty()) {
14    tie(std::ignore, s) = pq.top();
15    pq.pop();
16
17    if (used[s]) continue;
18    used[s] = true; // 每一個點都只看一
19      次
20
21    for (auto [e, w] : V[s]) {
22      if (dist[e] > dist[s] + w) {
23        dist[e] = dist[s] + w;
24        pq.emplace(dist[e], e);
25      }
26    }
27  }
28  return dist[e];

```

5.4 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;

```

5.5 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][
7           j])
8           d[i][j] = d[i][k] + d[k]

```

5.6 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){

```

```

16 low[u] = dfn[u] = ++dfn_cnt, vis[u] =
17 1;
18 st.push(u);
19 for(auto v: g[u]){
20     if(!dfn[v]){
21         dfs(v, dis+1);
22         low[u] = min(low[u], low[v]);
23     }
24     else if(vis[v]){
25         low[u] = min(low[u], dfn[v]);
26     }
27 }
28 if(low[u]==dfn[u]){
29     ++cnt;
30     while(vis[u]){
31         auto v = st.top();
32         st.pop();
33         vis[v] = 0;
34         scc[v] = cnt;
35         sz[cnt]++;
36     }
37 }
38 for(int i = 0; i <= n; ++i){
39     if(!scc[i]){
40         dfs(i, 1);
41     }
42 }
43 }
44 vector<vector<int>> compress(){
45     vector<vector<int>> ans(cnt+1);
46     for(int u = 0; u <= n; ++u){
47         for(auto v: g[u]){
48             if(scc[u] == scc[v]){
49                 continue;
50             }
51             ans[scc[u]].push_back(scc[v]);
52         }
53     }
54     for(int i = 0; i <= cnt; ++i){
55         sort(ans[i].begin(), ans[i].end());
56         ans[i].erase(unique(ans[i].begin(),
57                             ans[i].end()), ans[i].end());
58     }
59     return ans;
60 }

```

5.7 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);

```

```

12 while (Q.size()) {
13     int u = Q.front();
14     Q.pop();
15     in_queue[u] = false;
16     for (auto [v, cost] : G[u])
17         if (d[v] > d[u] + cost) {
18             if (++cnt[u] >= n) return {}; // 存在
19             負環
20             d[v] = d[u] + cost;
21             if (!in_queue[v]) enqueue(v);
22         }
23     }
24     return d;
25 }

```

5.8 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;
14     for(int v : G[u])
15         if(!coloring(v, col ^ 1))
16             return false;
17     return true;
18 }
19 //check if a graph is a bipartite graph
20 bool checkBipartiteG(int n) {
21     for(int i = 1; i <= n; i++)
22         color[i] = -1;
23     for(int i = 1; i <= n; i++)
24         if(color[i] == -1 &&
25             !coloring(i, 0))
26             return false;
27     return true;
28 }

```

5.9 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 {

```

```

7 Change = 0;
8 for(size_t u = 0; u < N; ++u) {
9     vector<size_t> E;
10     for(size_t v = 0; v < N && E.size() <
11         3; ++v)
12         if(G[u][v] && u != v) E.emplace_back
13             (v);
14     if(E.size() == 1 || E.size() == 2) {
15         ++Change;
16         for(auto v : E) G[u][v] = G[v][u] =
17             false;
18     }
19     if(E.size() == 2) {
20         auto [a,b] = make_pair(E[0], E[1]);
21         G[a][b] = G[b][a] = true;
22     }
23 }
24 while(Change);
25 return G;
26 }
27 //計算Degree
28 //O(n^2)
29 vector<size_t> getDegree(const
30     AdjacencyMatrixTy &G) {
31     size_t N = G.size();
32     vector<size_t> Degree(N);
33     for(size_t u = 0; u < N; ++u)
34         for(size_t v = u + 1; v < N; ++v) {
35             if(!G[u][v]) continue;
36             ++Degree[u], ++Degree[v];
37         }
38     return Degree;
39 }
40 //判斷是否為K5 or K33
41 //O(n)
42 bool is_K5_or_K33(const vector<size_t> &
43     Degree) {
44     unordered_map<size_t, size_t> Num;
45     for(auto Val : Degree) ++Num[Val];
46     size_t N = Degree.size();
47     bool isK5 = Num[4] == 5 && Num[4] + Num[0]
48         == N;
49     bool isK33 = Num[3] == 6 && Num[3] + Num
50         [0] == N;
51     return isK5 || isK33;
52 }

```

5.10 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 //check if a graph contains a cycle
15 bool checkCycle(int n) {
16     for(int i = 1; i <= n; i++)
17         if(!visit[i] && dfs(i, -1))
18             return true;
19     return false;
20 }

```

5.11 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     ];
6     int sol[MAXN],tmp[MAXN]; //sol[0~ans-1]為答
7     案
8     void init(int n){
9         N=n; //0-base
10         memset(g,0,sizeof(g));
11     }
12     void add_edge(int u,int v){
13         g[u][v]=g[v][u]=1;
14     }
15     int dfs(int ns,int dep){
16         if(!ns){
17             if(dep>ans){
18                 ans=dep;
19                 memcpy(sol,tmp,sizeof tmp);
20                 return 1;
21             }else return 0;
22         }
23         for(int i=0;i<ns;++i){
24             if(dep+ns-i<=ans)return 0;
25             int u=stk[dep][i],cnt=0;
26             if(dep+dp[u]<=ans)return 0;
27             for(int j=i+1;j<ns;++j){
28                 int v=stk[dep][j];
29                 if(g[u][v])stk[dep+1][cnt++]=v;
30             }
31             tmp[dep]=u;
32             if(dfs(cnt,dep+1))return 1;
33         }
34         return 0;
35     }
36     int clique(){
37         int u,v,ns;
38         for(ans=0,u=N-1;u>=0;--u){
39             for(ns=0,tmp[0]=u,v=u+1;v<N;++v)
40                 if(g[u][v])stk[1][ns++]=v;
41             dfs(ns,1),dp[u]=ans;
42         }
43     };

```

5.12 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
7         Set &B){
8         Set C(A.size() + B.size());
9         auto it = set_union(A.begin(),A.end(),B.
10             begin(),B.end(),C.begin());
11         C.erase(it, C.end());
12         return C;
13     }
14     static Set setIntersection(const Set &A,
15         const Set &B){
16         Set C(min(A.size(), B.size()));
17         auto it = set_intersection(A.begin(),A.
18             end(),B.begin(),B.end(),C.begin());
19         C.erase(it, C.end());
20         return C;
21     }
22     static Set setDifference(const Set &A,
23         const Set &B){
24         Set C(min(A.size(), B.size()));
25         auto it = set_difference(A.begin(),A.end
26             (),B.begin(),B.end(),C.begin());
27         C.erase(it, C.end());
28         return C;
29     }
30     void BronKerbosch1(Set R, Set P, Set X){
31         if(P.empty() && X.empty()){
32             // R form an maximal clique
33             return;
34         }
35         for(auto v: P){
36             BronKerbosch1(setUnion(R,{v}),
37                 setIntersection(P,G[v]),
38                 setIntersection(X,G[v]));
39             P = setDifference(P,{v});
40             X = setUnion(X,{v});
41         }
42     }
43     void init(int _n){
44         G.clear();
45         G.resize((n = _n) + 1);
46     }
47     void addEdge(int u, int v){
48         G[u].emplace_back(v);
49         G[v].emplace_back(u);
50     }
51     void solve(int n){
52         Set P;
53         for(int i=1; i<=n; ++i){
54             sort(G[i].begin(), G[i].end());
55             G[i].erase(unique(G[i].begin(), G[i].end()),
56                 G[i].end());
57             P.emplace_back(i);
58         }
59         BronKerbosch1({}, P, {});
60     }
61 }
62 //判斷圖G是否能3塗色：

```

```

55 //枚舉圖G的極大獨立集I (極大獨立集 = 補圖極
56 //大團)
57 //若存在I使得G-I形成二分圖，則G可以三塗色
58 //反之則不能3塗色

```

5.13 橋連通分量

```

1 vector<pii> findBridges(const vector<vector<
2     int>>& g) {
3     int n = (int) g.size();
4     vector<int> id(n, -1), low(n);
5     vector<pii> bridges;
6     function<void(int, int)> dfs = [&](int u,
7         int p) {
8         static int cnt = 0;
9         id[u] = low[u] = cnt++;
10        for(auto v : g[u]) {
11            if(v == p) continue;
12            if(id[v] != -1) low[u] = min(low[u],
13                id[v]);
14            else {
15                dfs(v, u);
16                low[u] = min(low[u], low[v]);
17                if(low[v] > id[u]) bridges.emplace_back(u, v);
18            }
19        }
20    };
21    for(int i = 0; i < n; ++i) {
22        if(id[i] == -1) dfs(i, -1);
23    }
24    return bridges;
25 }

```

5.14 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.
9             resize(n+5), vis.resize(n+5), is_ap.
10                resize(n+5), bcc_id.resize(n+5);
11     }
12     inline void build(const vector<vector<int>
13         >>&g, int u, int p = -1){
14         int child = 0;
15         dfn[u] = low[u] = ++dfn_cnt;
16         st.push(u);
17         vis[u] = 1;
18         if(g[u].empty() and p == -1){
19             bcc_id[u] = ++bcc_cnt;
20             bcc[bcc_cnt].push_back(u);
21             return;
22         }
23         for(auto v: g[u]){
24             if(v == p) continue;
25         }
26     }

```

```

27     if(!dfn[v]){
28         build(g, v, u);
29         child++;
30         if(dfn[u] <= low[v]){
31             is_ap[u] = 1;
32             bcc_id[u] = ++bcc_cnt;
33             bcc[bcc_cnt].push_back(u);
34             while(vis[v]){
35                 bcc_id[st.top()] = bcc_cnt;
36                 bcc[bcc_cnt].push_back(st.top());
37                 st.pop();
38             }
39             vis[st.top()] = 0;
40             st.pop();
41         }
42         low[u] = min(low[u], low[v]);
43     }
44     low[u] = min(low[u], dfn[v]);
45     if(p == -1 and child < 2) is_ap[u] = 0;
46     if(is_ap[u]) ap.push_back(u);
47 }
48 }

```

6 Math

6.1 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 void all_divdown(const LL &n) { // all n/x
14     for(LL a=1; a<=n; a=n/(n/(a+1))) {
15         // dosomething;
16     }
17 }
18 const int MAXPRIME = 1000000;
19 int iscom[MAXPRIME], prime[MAXPRIME],
20     primecnt;
21 int phi[MAXPRIME], mu[MAXPRIME];
22 void sieve(void){
23     memset(iscom, 0, sizeof(iscom));
24     primecnt = 0;
25     phi[1] = mu[1] = 1;
26     for(int i=2; i<MAXPRIME; ++i) {
27         if(!iscom[i]) {
28             prime[primecnt++] = i;
29             mu[i] = -1;
30             phi[i] = i-1;
31         }
32         for(int j=0; j<primecnt; ++j) {
33             int k = i * prime[j];
34         }
35     }

```

```

36     if(k <= MAXPRIME) break;
37     iscom[k] = prime[j];
38     if(i%prime[j] == 0) {
39         mu[k] = 0;
40         phi[k] = phi[i] * prime[j];
41         break;
42     } else {
43         mu[k] = -mu[i];
44         phi[k] = phi[i] * (prime[j]-1);
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         ());
61     for(LL g=2; g<p; ++g)
62         if(g_test(g, p, v))
63             return g;
64     puts("primitive_root NOT FOUND");
65     return -1;
66 }
67 int Legendre(const LL &a, const LL &p) {
68     return modexp(a, (p-1)/2, p);
69 }
70 LL inv(const LL &a, const LL &n) {
71     LL d, x, y;
72     gcd(a, n, d, x, y);
73     return d==1 ? (x+n)%n : -1;
74 }
75 int inv[maxN];
76 LL invtable(int n, LL P){
77     inv[1]=1;
78     for(int i=2; i<=n; ++i)
79         inv[i]=(P-(P/i))*inv[P%i]%P;
80 }
81 LL log_mod(const LL &a, const LL &b, const
82     LL &p) {
83     // a ^ x = b (mod p)
84     int m=sqrt(p+.5), e=1;
85     LL v=inv(modexp(a, m, p), p);
86     map<LL, int> x;
87     x[1]=0;
88     for(int i=1; i<=m; ++i) {
89         e = LLmul(e, a, p);
90         if(!x.count(e)) x[e] = i;
91     }
92     for(int i=0; i<=m; ++i) {
93         if(x.count(b)) return i*m + x[b];
94         b = LLmul(b, v, p);
95     }
96     return -1;
97 }

```

6.2 Bit Set

```

93 LL Tonelli_Shanks(const LL &n, const LL &p)
94 {
95     //  $x^2 = n \pmod p$ 
96     if(n==0) return 0;
97     if(Legendre(n,p)!=1) while(1) { puts("SQRT
98         ROOT does not exist"); }
99     int S = 0;
100     LL Q = p-1;
101     while( !(Q&1) ) { Q>>=1; ++S; }
102     if(S==1) return modexp(n%p,(p+1)/4,p);
103     LL z = 2;
104     for(; Legendre(z,p)!=-1; ++z)
105     LL c = modexp(z,Q,p);
106     LL R = modexp(n%p,(Q+1)/2,p), t = modexp(n
107         %p,Q,p);
108     int M = S;
109     while(1) {
110         if(t==1) return R;
111         LL b = modexp(c,1<<(M-i-1),p);
112         R = LLMul(R,b,p);
113         t = LLMul( LLMul(b,b,p), t, p);
114         c = LLMul(b,b,p);
115         M = i;
116     }
117     return -1;
118 }
119 template<typename T>
120 T Euler(T n){
121     T ans=n;
122     for(T i=2;i<=n;++i){
123         if(n%i==0){
124             ans=ans/i*(i-1);
125             while(n%i==0)n/=i;
126         }
127     }
128     if(n>1)ans=ans/n*(n-1);
129     return ans;
130 }
131 //Chinese_remainder_theorem
132 template<typename T>
133 T pow_mod(T n,T k,T m){
134     T ans=1;
135     for(n=(n>=m?n%m:n);k>>=1){
136         if(k&1)ans=ans*n%m;
137         n=n*n%m;
138     }
139     return ans;
140 }
141 template<typename T>
142 T crt(vector<T> &m,vector<T> &a){
143     T M=1,tM,ans=0;
144     for(int i=0;i<(int)m.size();++i)M*=m[i];
145     for(int i=0;i<(int)a.size();++i){
146         tM=M/m[i];
147         ans=(ans+(a[i]*tM%M)*pow_mod(tM,Euler(m[
148             i])-1,m[i])%M)%M;
149     }
150     /*如果m[i]是質數·Euler(m[i])-1=m[i]-2·
151     就不用算Euler了*/
152     return ans;
153 }

```

```

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

```

6.3 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 }

```

6.4 FastPow

```

1 ll modexp(ll x, ll k, ll p) {
2     ll ans = 1;
3     for(int i = 1; i <= k; i <= 1) {
4         if(i & k) ans *= x, ans %= p;
5         x *= x, x %= p;
6     }
7     return ans;
8 }

```

6.5 FFT

```

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

```

```

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 }

```

6.6 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             f[j+k+(1<<i)] += f[j+k]*(inverse
6                 ?-1:1);
7     return f;
8 }

```

```

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

```

6.7 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;
16            ld delta = a[j][i] / a[i][i];
17            FOR(k, i, n + 1) a[j][k] -= delta * a[
18                i][k];
19        }
20    }
21    bool ok = true;
22    REP(i, n) {
23        if(abs(a[i][i]) <= EPS) {
24            if(abs(a[i][n]) > EPS) return -1;
25            ok = false;
26        }
27    }
28    return ok;
29 }

```

6.8 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 }

```



```

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

```

6.9 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(-1LL,0LL);
14 |        lastb = Llmul((lastb-b[i])/d,x,(lastm/d)*m[i]);
15 |        lastm = (lastm/d)*m[i];
16 |        lastb = (lastb+b[i])%lastm;
17 |    }
18 |    return make_pair(lastb<0?lastb+lastm:lastb,lastm);

```

6.10 LinearSieve

```

1 | vector<bool> is_prime;
2 | vector<int> primes, phi, mobius, least;
3 | void linear_sieve(int n) {
4 |     n += 1;
5 |     is_prime.resize(n);
6 |     least.resize(n);
7 |     fill(2 + begin(is_prime),end(is_prime),true);
8 |     phi.resize(n); mobius.resize(n);
9 |     phi[1] = mobius[1] = 1;
10 |    least[0] = 0,least[1] = 1;
11 |    for(int i = 2; i < n; ++i) {
12 |        if(is_prime[i]) {
13 |            primes.push_back(i);
14 |            phi[i] = i - 1;
15 |            mobius[i] = -1;
16 |            least[i] = i;
17 |        }
18 |        for(auto j : primes) {

```

```

19 |         if(i * j >= n) break;
20 |         is_prime[i * j] = false;
21 |         least[i * j] = j;
22 |         if(i % j == 0) {
23 |             mobius[i * j] = 0;
24 |             phi[i * j] = phi[i] * j;
25 |             break;
26 |         } else {
27 |             mobius[i * j] = mobius[i] * mobius[j];
28 |             phi[i * j] = phi[i] * phi[j];
29 |         }
30 |     }
31 | }
32 | }

```

6.11 Lucas

```

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

```

6.12 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 |            lazy[i]=det/m[i][i];
73 |            for(auto &j:m[i])j/=lazy[i];
74 |        }
75 |        return det;
76 |    }
77 | }

```

6.13 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

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

- 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. $\pi(j) > j$.

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

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

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

6.14 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)
33         return 1;
34     return mr(2,p) && mr(3,p) && mr(5,p) && mr
35         (7,p) && mr(43,p);
36 }
37
38 // O(1) 快速乘(防LL overflow)
39 ll ksc(ull x, ull y, ll p) {
40     return (x*y-(ull)((lbd)x/p*y)*p+p)%p;
41 }
42
43 //求n任一真因数(需保证n非质数) O(n^1/4)
44 ll pollar_rho(ll n) {
45     ll x,y,z,c,g,i,j;
46     while(1) {
47         x = y = rand()%n;
48         z = 1;
49         c = rand()%n;
50         i = 0, j = 1;
51         while(++i) {
52             x = (ksc(x,x,n) + c)%n;
53             z = ksc(z,abs(y-x),n);
54             if(x == y || !z) break;
55             if(!(i%127) || i == j) {
56                 g = __gcd(z,n);
57                 if(g > 1) return g;
58                 if(i == j) y = x, j <= 1;
59             }
60         }
61     }
62 }
63
64 void factorization(ll n) {
65     while(!is_prime(n)) {
66         ll f = pollar_rho(n);
67         while(!is_prime(f)) {
68             f = pollar_rho(f);
69         }
70         ll cou = 0;
71         while(n%f == 0) n /= f, cou++;
72         fac.push_back({f,cou});
73     }
74     if(n != 1) fac.push_back({n,1});
75 }
76
77 void get_factors(ll now, ll cou) {
78     if(now >= fac.size()) {
79         factor.push_back(cou);
80         return;
81     }
82     get_factors(now+1, cou);
83 }

```

```

81 for(ll i=1;i<=fac[now].second;i++) {
82     cou *= fac[now].first;
83     get_factors(now+1,cou);
84 }
85 }

```

6.15 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

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

- 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 $\frac{(n-2)!}{(d_1-1)!(d_2-1)!\dots(d_n-1)!}$ 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 and $\sum_{i=1}^k d_i \leq k(k-1) + \sum_{i=k+1}^n \min(d_i, k)$ 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 $\sum_{i=1}^n a_i = \sum_{i=1}^n b_i$ and $\sum_{i=1}^k a_i \leq \sum_{i=1}^k \min(b_i, k)$ 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 $\sum_{i=1}^n a_i = \sum_{i=1}^n b_i$ and $\sum_{i=1}^k a_i \leq \sum_{i=1}^k \min(b_i, k-1) + \sum_{i=k+1}^n \min(b_i, k)$ holds for every $1 \leq k \leq n$.

- Möbius 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, θ : $\arcsin(a/r)$.
- 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 = $2\pi rh = \pi(a^2 + h^2) = 2\pi r^2(1 - \cos\theta)$.

6.16 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
13 double find(const vector<double>&coef, int n
14     , double lo, double hi){
15     double sign_lo, sign_hi;

```

```

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

```

6.17 質因數分解

```

1 //CSES Counting Divisors
2 #include<bits/stdc++.h>
3 using namespace std;
4
5 int n;
6
7 vector<int> primes;
8 vector<int> LPS;
9
10 void sieve(int n) {
11     LPS.assign(n+1,1);
12     for(int i=2;i<n;i++){
13         if(LPS[i]==1) {
14             primes.emplace_back(i);
15             LPS[i] = i;
16         }
17     }
18     for(auto p:primes) {

```

```

18         if(1LL*i*p > n) break;
19         LPS[i*p] = p;
20         if(i%p==0) break;
21     }
22 }
23 }
24 }
25 signed main() {
26     cin>>n;
27     sieve((int)1e6);
28     map<int,int> divisor;
29     while(n--) {
30         divisor.clear();
31         int x;cin>>x;
32         while(x>1) {
33             divisor[LPS[x]]++;
34             x/=LPS[x];
35         }
36         int ans = 1;
37         for(auto &[x,y] : divisor) ans *= (y
38             +1);
39         cout<<ans;
40         cout<<'\\n';
41     }

```

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[
24                pos]]==mx)mx--;
25            cnt[v[pos]]--;
26            cntcnt[cnt[v[pos]]]++;
27            mx = max(mx,cnt[v[pos]]);
28        };
29        sort(all(q),[&](qry a,qry b){
30            static int B = max((int)1,n/max((int)
31                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 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'
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;
38         id2 = r/block_siz; r %= block_siz;
39         if(ord == 1)
40         {
41             if(id == id2)
42             {
43                 pre = q[id][r];
44                 for(i=r;i>l;i--)
45                     q[id][i] = q[id][i-1];

```

```

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     }
73     //print_all(n);
74 }
75 else
76 {
77     // query m cnt
78     cin >> m;
79     m = (m+ans-1)%n+1;
80     ans = 0;
81     if(id == id2)
82     {
83         for(i=l;i<=r;i++) ans += (q[
84             id][i] == m);
85     }
86     else
87     {
88         for(i=l;i<block_siz;i++) ans
89             += (q[id][i] == m);
90         for(i=0;i<=r;i++) ans += (q[
91             id2][i] == m);
92         for(i=id+1;i<id2;i++) ans +=
93             cnt[i][m];
94     }
95     cout << ans << endl;
96 }
97 }
98 return 0;
99 }

```

7.3 莫隊

```

1 void remove(idx); // TODO: remove value at
2 void add(idx); // TODO: add value at idx
3 int get_answer(); // TODO: extract the
4

```

```

5 int block_size;
6
7 struct Query {
8     int l, r, idx;
9     bool operator<(Query other) const
10     {
11         return make_pair(l / block_size, r)
12             < make_pair(other.l /
13                 block_size, other.r);
14     }
15 };
16 vector<int> mo_s_algorithm(vector<Query>
17     queries) {
18     vector<int> answers(queries.size());
19     sort(queries.begin(), queries.end());
20     // TODO: initialize data structure
21
22     int cur_l = 0;
23     int cur_r = -1;
24     // invariant: data structure will always
25     // reflect the range [cur_l, cur_r]
26     for (Query q : queries) {
27         while (cur_l > q.l) {
28             cur_l--;
29             add(cur_l);
30         }
31         while (cur_r < q.r) {
32             cur_r++;
33             add(cur_r);
34         }
35         while (cur_l < q.l) {
36             remove(cur_l);
37             cur_l++;
38         }
39         while (cur_r > q.r) {
40             remove(cur_r);
41             cur_r--;
42         }
43         answers[q.idx] = get_answer();
44     }
45     return answers;
46 }

```

8 Tree

8.1 centroidDecomposition

```

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);

```

```

11     sz[u]+=sz[v];
12 }
13 };
14 function<int(int,int)>dfs2 = [&](int u,int
15     p){
16     for(auto v:g[u]){
17         if(v==p)continue;
18         if(vis[v])continue;
19         if(sz[v]*2<n)continue;
20         return dfs2(v,u);
21     }
22     return u;
23 };
24 dfs1(u,-1);
25 return dfs2(u,-1);
26 }
27 int cal(int u,int p = -1,int deep = 1){
28     int ans = 0;
29     tmp.pb(deep);
30     sz[u] = 1;
31     for(auto v:g[u]){
32         if(v==p)continue;
33         if(vis[v])continue;
34         ans+=cal(v,u,deep+1);
35         sz[u]+=sz[v];
36     }
37     //calculate the answer
38     return ans;
39 }
40 int centroid_decomposition(int u,int
41     tree_size){
42     int center = tree_centroid(u,tree_size);
43     vis[center] = 1;
44     int ans = 0;
45     for(auto v:g[center]){
46         if(vis[v])continue;
47         ans+=cal(v);
48         for(int i = sz(tmp)-sz[v];i<sz(tmp);++i)
49             //update
50     }
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;

```

8.2 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;

```

```

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

```

8.3 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(
6         _n) {
7         g.resize(n+5);
8         dep.resize(n+5);
9         father.resize(n+5);
10        sz.resize(n+5);
11        mxson.resize(n+5);
12        topf.resize(n+5);
13        id.resize(n+5);
14    }
15    void add_edge(int u, int v){
16        g[u].push_back(v);
17        g[v].push_back(u);
18    }
19    void dfs(int u,int p){
20        dep[u] = dep[p]+1;
21        father[u] = p;
22        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            ;
50            ans.push_back({id[topf[u]], id[u]});
51            u = father[topf[u]];
52        }
53        if(id[u]>id[v])swap(u,v);
54        ans.push_back({id[u], id[v]});
55        return ans;
56    };

```

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

```

8.5 link cut tree

```

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

```

```

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

```

```

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

```

```

145         up(v);
146         q.push(v);
147     }
148 }
149 }
150 }
151 void change(int x,int b){
152     splay(x);
153     //nd[x].data=b;
154     up(x);
155 }

```

8.6 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 }

```

8.7 Tree diameter

```

1 //dfs兩次
2 vector<int> level;
3
4 void dfs(int u, int parent = -1) {
5     if(parent == -1) level[u] = 0;
6     else level[u] = level[parent] + 1;
7     for (int v : Tree[u]) {
8         if (v == parent) continue;
9         dfs(v, u);
10    }
11 }
12
13 dfs(1); // 隨便選一個點
14 int a = max_element(level.begin(), level.end
    ()) - level.begin();
15 dfs(a); // a 必然是直徑的其中一個端點
16 int b = max_element(level.begin(), level.end
    ()) - level.begin();
17 cout << level[b] << endl;
18
19 //紀錄每個點的最長距離跟次長距離
20 vector<int> D1, D2; // 最遠、次遠距離
21 int ans = 0; // 直徑長度
22

```

```

23 void dfs(int u, int parent = -1) {
24     D1[u] = D2[u] = 0;
25     for (int v : Tree[u]) {
26         if (v == parent) continue;
27         dfs(v, u);
28         int dis = D1[v] + 1;
29         if (dis > D1[u]) {
30             D2[u] = D1[u];
31             D1[u] = dis;
32         } else
33             D2[u] = max(D2[u], dis);
34     }
35     ans = max(ans, D1[u] + D2[u]);
36 }

```

8.8 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 //Treap紀錄Parent
45 struct Treap {

```



```

46 Treap *lc = nullptr, *rc = nullptr;
47 Treap *pa = nullptr;
48 unsigned pri, size;
49 long long Val, Sum;
50 Treap(int Val):
51     pri(rand()), size(1),
52     Val(Val), Sum(Val) {}
53 void pull();
54 };
55
56 void Treap::pull() {
57     size = 1;
58     Sum = Val;
59     pa = nullptr;
60     if (lc) {
61         size += lc->size;
62         Sum += lc->Sum;
63         lc->pa = this;
64     }
65     if (rc) {
66         size += rc->size;
67         Sum += rc->Sum;
68         rc->pa = this;
69     }
70 }
71 //找出節點在中序的編號
72 size_t getIdx(Treap *x) {
73     assert(x);
74     size_t Idx = 0;
75     for (Treap *child = x->rc; x;) {
76         if (child == x->rc)
77             Idx += 1 + size(x->lc);
78         child = x;
79         x = x->pa;
80     }
81     return Idx;
82 }
83 //切出想要的東西
84 void move(Treap *&root, int a, int b) {
85     size_t a_in = getIdx(In[a]), a_out =
86         getIdx(Out[a]);
87     auto [L, tmp] = splitK(root, a_in - 1);
88     auto [tree_a, R] = splitK(tmp, a_out -
89         a_in + 1);
90     root = merge(L, R);
91     tie(L, R) = splitK(root, getIdx(In[b]));
92     root = merge(L, merge(tree_a, R));
93 }

```

9 string

9.1 AC 自動機

```

1 template<char L='a',char R='z'>
2 class ac_automaton{
3     struct joe{
4         int next[R-L+1], fail, efl, ed, cnt_dp, vis;
5         joe():ed(0), cnt_dp(0), vis(0){
6             for(int i=0; i<=R-L; ++i) next[i]=0;
7         }

```

```

8     };
9     public:
10     std::vector<joe> S;
11     std::vector<int> q;
12     int qs, qe, vt;
13     ac_automaton():S(1), qs(0), qe(0), vt(0){
14         void clear(){
15             q.clear();
16             S.resize(1);
17             for(int i=0; i<=R-L; ++i) S[0].next[i]=0;
18             S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
19         }
20         void insert(const char *s){
21             int o=0;
22             for(int i=0, id=s[i]; ++i){
23                 id=s[i]-L;
24                 if(!S[o].next[id]){
25                     S.push_back(joe());
26                     S[o].next[id]=S.size()-1;
27                 }
28                 o=S[o].next[id];
29             }
30             ++S[o].ed;
31         }
32         void build_fail(){
33             S[0].fail=S[0].efl=-1;
34             q.clear();
35             q.push_back(0);
36             ++qe;
37             while(qs!=qe){
38                 int pa=q[qs++], id, t;
39                 for(int i=0; i<=R-L; ++i){
40                     t=S[pa].next[i];
41                     if(!t) continue;
42                     id=S[pa].fail;
43                     while(~id&&!S[id].next[i]) id=S[id].fail;
44                     S[t].fail=~id?S[id].next[i]:0;
45                     S[t].efl=S[S[t].fail].ed?S[t].fail:S[t].fail;
46                     q.push_back(t);
47                     ++qe;
48                 }
49             }
50         }
51         //DP出每個前綴在字串s出現的次數並傳回所有
52         //字串被s匹配成功的次數O(N*M)*/
53         int match_0(const char *s){
54             int ans=0, id, p=0, t;
55             for(i=0; s[i]; ++i){
56                 id=s[i]-L;
57                 while(!S[p].next[id]&&p) p=S[p].fail;
58                 if(!S[p].next[id]) continue;
59                 p=S[p].next[id];
60                 ++S[p].cnt_dp; /*匹配成功則它所有後綴都
61                     可以被匹配(DP計算)*/
62             }
63             for(i=qe-1; i>=0; --i){
64                 ans+=S[q[i]].cnt_dp*S[q[i]].ed;
65                 if(~S[q[i]].fail) S[q[i]].fail.
66                     cnt_dp+=S[q[i]].cnt_dp;
67             }
68             return ans;
69         }

```

```

67         /*多串匹配走efl邊並傳回所有字串被s匹配成功
68         的次數O(N*M^1.5)*/
69         int match_1(const char *s) const{
70             int ans=0, id, p=0, t;
71             for(int i=0; s[i]; ++i){
72                 id=s[i]-L;
73                 while(!S[p].next[id]&&p) p=S[p].fail;
74                 if(!S[p].next[id]) continue;
75                 p=S[p].next[id];
76                 if(S[p].ed) ans+=S[p].ed;
77                 for(t=S[p].efl; ~t; t=S[t].efl){
78                     ans+=S[t].ed; /*因為都走efl邊所以保證
79                     匹配成功*/
80                 }
81             }
82             return ans;
83         }
84         /*枚舉(s的子字串nA)的所有相異字串各恰一次
85         並傳回次數O(N*M^(1/3))*/
86         int match_2(const char *s){
87             int ans=0, id, p=0, t;
88             ++vt;
89             /*把截記vt+=1. 只要vt沒溢位. 所有S[p].
90             vis==vt就會變成false
91             這種利用vt的方法可以O(1)歸零vis陣列*/
92             for(int i=0; s[i]; ++i){
93                 id=s[i]-L;
94                 while(!S[p].next[id]&&p) p=S[p].fail;
95                 if(!S[p].next[id]) continue;
96                 p=S[p].next[id];
97                 if(S[p].ed&&S[p].vis!=vt){
98                     S[p].vis=vt;
99                     ans+=S[p].ed;
100                 }
101                 for(t=S[p].efl; ~t&&S[t].vis!=vt; t=S[t].efl){
102                     S[t].vis=vt;
103                     ans+=S[t].ed; /*因為都走efl邊所以保證
104                     匹配成功*/
105                 }
106             }
107             return ans;
108         }
109         /*把AC自動機變成真的自動機*/
110         void evolution(){
111             for(qs=1; qs!=qe; ++q){
112                 int p=q[qs-1];
113                 for(int i=0; i<=R-L; ++i){
114                     if(S[p].next[i]==0) S[p].next[i]=S[S[p].fail].next[i];
115                 }
116             }
117         }

```

9.2 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    vector<int> match_index;
12    /*以字串B匹配字串A. 傳回匹配成功的數量(用B的
13    fail)*/
14    int kmp_match(char *A, int lenA, char *B, int
15    lenB, int *fail){
16        int id=-1, ans=0;
17        for(int i=0; i<lenA; ++i){
18            while(~id&&B[id+1]!=A[i]) id=fail[id];
19            if(B[id+1]==A[i]) ++id;
20            if(id==lenB-1){ /*匹配成功*/
21                ++ans, id=fail[id];
22                match_index.emplace_back(i + 1 - lenB);
23            }
24        }
25        return ans;
26    }

```

9.3 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], 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 }

```

9.4 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 }

```

9.5 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])]++;
8 }
9 void firstCol(){
10     memset(first,0,sizeof(first));
11     int totc = 0;
12     for(int c='A';c<='Z';++c){
13         if(!tots[c]) continue;
14         first[c] = totc;
15         totc += tots[c];
16     }
17 }
18 string reverseBwt(string bw,int begin){
19     rankBWT(bw), firstCol();
20     int i = begin; //原字串最後一個元素的位置
21     string res;
22     do{
23         char c = bw[i];
24         res = c + res;
25         i = first[int(c)] + ranks[i];
26     }while( i != begin );
27     return res;
28 }

```

9.6 Rolling Hash

```

1 //Rolling Hash(10 Hash) CF 1800 D. Remove
2 //Two Letters
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 constexpr long long power(long long x, long
7     long n, int m) {
8     if(m == 1) return 0;
9     unsigned int _m = (unsigned int)(m);
10    unsigned long long r = 1;
11    x %= m;
12    if(x < 0) {
13        x += m;
14    }
15    unsigned long long y = x;
16    while(n) {
17        if(n & 1) r = (r * y) % _m;
18        y = (y * y) % _m;
19        n >>= 1;
20    }
21    return r;
22 }
23 template<int HASH_COUNT, bool
24     PRECOMPUTE_POWERS = false>
25 class Hash {
26 public:
27     static constexpr int MAX_HASH_PAIRS = 10;
28     // {mul, mod}

```

```

29 static constexpr const pair<int, int>
30     HASH_PAIRS[] = {{827167801,
31         999999937},
32         {998244353,
33             999999929},
34         {146672737,
35             922722049},
36         {204924373,
37             952311013},
38         {585761567,
39             955873937},
40         {484547929,
41             901981687},
42         {856009481,
43             987877511},
44         {852853249,
45             996724213},
46         {937381759,
47             994523539},
48         {116508269,
49             993179543}};
50
51 Hash() : Hash("") {}
52
53 Hash(const string& s) : n(s.size()) {
54     static_assert(HASH_COUNT > 0 &&
55         HASH_COUNT <= MAX_HASH_PAIRS);
56     for(int i = 0; i < HASH_COUNT; ++i) {
57         const auto& p = HASH_PAIRS[i];
58         pref[i].resize(n);
59         pref[i][0] = s[0];
60         for(int j = 1; j < n; ++j) {
61             pref[i][j] = (1LL * pref[i][j - 1] *
62                 p.first + s[j]) % p.second;
63         }
64     }
65     if(PRECOMPUTE_POWERS) {
66         build_powers(n);
67     }
68
69 void add_char(char c) {
70     for(int i = 0; i < HASH_COUNT; ++i) {
71         const auto& p = HASH_PAIRS[i];
72         pref[i].push_back((1LL * (n == 0 ? 0 :
73             pref[i].back()) * p.first + c) %
74             p.second);
75     }
76     n += 1;
77     if(PRECOMPUTE_POWERS) {
78         build_powers(n);
79     }
80 }
81
82 // Return hash values for [l, r)
83 array<int, HASH_COUNT> substr(int l, int r
84 ) {

```

```

85     array<int, HASH_COUNT> res{};
86     for(int i = 0; i < HASH_COUNT; ++i) {
87         res[i] = substr(i, l, r);
88     }
89     return res;
90 }
91
92 array<int, HASH_COUNT> merge(const vector<
93     pair<int, int>>& seg) {
94     array<int, HASH_COUNT> res{};
95     for(int i = 0; i < HASH_COUNT; ++i) {
96         const auto& p = HASH_PAIRS[i];
97         for(auto [l, r] : seg) {
98             res[i] = (1LL * res[i] * get_power(i
99                 , r - l) + substr(i, l, r)) % p.
100                 second;
101         }
102     }
103     return res;
104 }
105
106 // build powers up to x^k
107 void build_powers(int k) {
108     for(int i = 0; i < HASH_COUNT; ++i) {
109         const auto& p = HASH_PAIRS[i];
110         int sz = (int) POW[i].size();
111         if(sz > k) {
112             continue;
113         }
114         if(sz == 0) {
115             POW[i].push_back(1);
116             sz = 1;
117         }
118         while(sz <= k) {
119             POW[i].push_back(1LL * POW[i].back()
120                 * p.first % p.second);
121             sz += 1;
122         }
123     }
124 }
125
126 inline int size() const {
127     return n;
128 }
129
130 private:
131 int n;
132 static vector<int> POW[MAX_HASH_PAIRS];
133 array<vector<int>, HASH_COUNT> pref;
134
135 int substr(int k, int l, int r) {
136     assert(0 <= k && k < HASH_COUNT);
137     assert(0 <= l && l <= r && r <= n);
138     const auto& p = HASH_PAIRS[k];
139     if(l == r) {
140         return 0;
141     }
142     int res = pref[k][r - 1];
143     if(l > 0) {
144         res -= 1LL * pref[k][l - 1] *
145             get_power(k, r - l) % p.second;
146     }
147     if(res < 0) {
148         res += p.second;
149     }
150     return res;
151 }

```

```

152 }
153
154 int get_power(int a, int b) {
155     if(PRECOMPUTE_POWERS) {
156         build_powers(b);
157         return POW[a][b];
158     }
159     const auto& p = HASH_PAIRS[a];
160     return power(p.first, b, p.second);
161 }
162
163 template<int A, bool B> vector<int> Hash<A,
164     B>::POW[Hash::MAX_HASH_PAIRS];
165
166 void solve() {
167     int n;
168     string s;
169     cin >> n >> s;
170     Hash<10, true> h(s);
171     set<array<int, 10>> used;
172     for(int i = 0; i + 1 < n; ++i) {
173         used.insert(h.merge({{0, i}, {i + 2, n
174             }}));
175     }
176     cout << used.size() << "\n";
177 }
178
179 int main() {
180     ios::sync_with_stdio(false);
181     cin.tie(0);
182     int tt;
183     cin >> tt;
184     while(tt--) {
185         solve();
186     }
187     return 0;
188 }

```

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>=0;i--)sa[--c[x[y[i]]]]=y[i]; \
6 }
7
8 #define AC(r,a,b){\
9     r[a]!r[b]||a+k>n||r[a+k]!r[b+k]\
10 void suffix_array(const char *s,int n,int *
11     sa,int *rank,int *tmp,int *c){
12     int A='z'+1,i,k,id=0;
13     for(i=0;i<n;++i)rank[tmp[i]=i]=s[i];
14     radix_sort(rank,tmp);
15     for(k=1;id<n-1;k<=1){
16         for(id=0,i=n-k;i<n;++i)tmp[id++]=i;
17         for(i=0;i<n;++i)
18             if(sa[i]>=k)tmp[id++]=sa[i]-k;
19         radix_sort(rank,tmp);
20         swap(rank,tmp);
21         for(rank[sa[0]]=id=0,i=1;i<n;++i)
22             rank[sa[i]]=id+=AC(tmp,sa[i-1],sa[i]);
23         A=id+1;
24     }
25 }

```

```

23 }
24 //h:高度數組 sa:後綴數組 rank:排名
25 void suffix_array_lcp(const char *s,int len,
26     int *h,int *sa,int *rank){
27     for(int i=0;i<len;++i)rank[sa[i]]=i;
28     for(int i=0,k=0;i<len;++i){
29         if(rank[i]==0)continue;
30         if(k)--k;
31         while(s[i+k]==s[sa[rank[i]-1]+k])++k;
32         h[rank[i]]=k;
33     }
34     h[0]=0;// h[k]=Lcp(sa[k],sa[k-1]);

```

9.8 Trie

```

1 template<int ALPHABET = 26, char MIN_CHAR =
2     'a'>
3 class trie {
4 public:
5     struct Node {
6         int go[ALPHABET];
7         Node() {
8             memset(go, -1, sizeof(go));
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].
17            go[v] : nodes[p].go[v] = newNode();
18    }
19
20    inline void insert(const vector<int>& a,
21        int p = 0) {
22        for(int v : a) {
23            p = next(p, v);
24        }
25    }
26
27    inline void clear() {
28        nodes.clear();
29        newNode();
30    }
31
32    inline int longest_common_prefix(const
33        vector<int>& a, int p = 0) const {
34        int ans = 0;
35        for(int v : a) {
36            if(nodes[p].go[v] != -1) {
37                ans += 1;
38                p = nodes[p].go[v];
39            } else {
40                break;
41            }
42        }
43        return ans;
44    }
45 private:

```

9.9 Z

```

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

```

10 tools

10.1 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

```

10.2 Bsearch

```

1 //Lower bound
2 int lower_bound(int arr[], int n, int val) {
3     int l = 0, r = n-1, mid, ret = -1;//沒搜
4     到return -1
5     while (l <= r) {
6         mid = (l+r)/2;
7         if (arr[mid] >= val) ret = mid, r =
8             mid-1;
9         else l = mid+1;
10    }
11    return ret;

```

10.3 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;

```

10.4 DuiPai

```

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

```

10.5 HashMap

```

1 struct splitmix64_hash {
2     static ull splitmix64(ull x) {
3         x += 0x9e3779b97f4a7c15;
4         x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9
5             ;
6         x = (x ^ (x >> 27)) * 0x94d049bb133111eb
7             ;
8         return x ^ (x >> 31);
9     }
10    ull operator()(ull x) const {
11        static const ull FIXED_RANDOM = RAND;
12        return splitmix64(x + FIXED_RANDOM);
13    };
14 };
15 template<class T, class U, class H =
16     splitmix64_hash> using hash_map =
17     gp_hash_table<T, U, H>;
18 template<class T, class H = splitmix64_hash>
19     using hash_set = hash_map<T, null_type,
20     H>;

```

10.6 pragma

```

1 #pragma GCC optimize("Ofast,unroll-loops")
2 #pragma GCC target("sse,sse2,ssse3,sse4,
3     popcnt,abm,mmx,avx,tune=native")
4 #pragma GCC optimize("inline")
5 #pragma GCC optimize("-fgcse")
6 #pragma GCC optimize("-fgcse-lm")
7 #pragma GCC optimize("-fipa-sra")
8 #pragma GCC optimize("-ftree-pre")
9 #pragma GCC optimize("-ftree-vec")
10 #pragma GCC optimize("-fpeephole2")
11 #pragma GCC optimize("-ffast-math")
12 #pragma GCC optimize("-fsched-spec")
13 #pragma GCC optimize("-falign-jumps")
14 #pragma GCC optimize("-falign-loops")
15 #pragma GCC optimize("-falign-labels")
16 #pragma GCC optimize("-fdevirtualize")
17 #pragma GCC optimize("-fcaller-saves")
18 #pragma GCC optimize("-fcrossjumping")
19 #pragma GCC optimize("-fthread-jumps")
20 #pragma GCC optimize("-funroll-loops")
21 #pragma GCC optimize("-fwhole-program")
22 #pragma GCC optimize("-freorder-blocks")
23 #pragma GCC optimize("-fschedule-insns")
24 #pragma GCC optimize("inline-functions")
25 #pragma GCC optimize("-ftree-tail-merge")
26 #pragma GCC optimize("-fschedule-insns2")
27 #pragma GCC optimize("-fstree-aliasing")
28 #pragma GCC optimize("-fstree-overflow")
29 #pragma GCC optimize("-falign-functions")
30 #pragma GCC optimize("-fcse-skip-blocks")
31 #pragma GCC optimize("-fcse-follow-jumps")
32 #pragma GCC optimize("-fsched-interblock")
33 #pragma GCC optimize("-fpartial-inlining")
34 #pragma GCC optimize("no-stack-protector")
35 #pragma GCC optimize("-freorder-functions")
36 #pragma GCC optimize("-findirect-inlining")
37 #pragma GCC optimize("-fhoist-adjacent-loads")

```

```

37 #pragma GCC optimize("-frerun-cse-after-loop")
38 #pragma GCC optimize("inline-small-functions")
39 #pragma GCC optimize("-finline-small-functions")
40 #pragma GCC optimize("-ftree-switch-conversion")
41 #pragma GCC optimize("-foptimize-sibling-calls")
42 #pragma GCC optimize("-fexpensive-optimizations")
43 #pragma GCC optimize("-funsafe-loop-optimizations")
44 #pragma GCC optimize("inline-functions-called-once")
45 #pragma GCC optimize("-fdelete-null-pointer-checks")

```

10.7 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;
11 }

```

10.8 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)

```

```

23 #define pii pair<int,int>
24 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 };

```

10.9 template bubble

```

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

```

10.10 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(m) > f(m + 1)) r = m;
18         else l = m;
19     }
20    return r;
21 }

```

ACM ICPC

Team Reference -

DreaminBubble

Contents

1	Computational Geometry	1	3.19	陣列上 Treap	8	6.5	FFT	15	9.2	KMP	19				
	1.1	Geometry	1	4	Flow	6.6	FWT	13	9.3	manacher	19				
	1.2	MinCircleCover	2			6.7	Gauss-Jordan	13	9.4	minimal string rotation	19				
	1.3	最近點對	3			6.8	InvGCD	13	9.5	reverseBWT	19				
2	DP	3	4.1	dinic	8	6.9	LinearCongruence	14	9.6	Rolling Hash	20				
	2.1	basic DP	3	4.2	Gomory Hu	9	6.10	LinearSieve	14	9.7	suffix array lcp	20			
	2.2	DP on Graph	3	4.3	ISAP with cut	9	6.11	Lucas	14	9.8	Trie	21			
	2.3	LineContainer	3	4.4	MinCostMaxFlow	9	6.12	Matrix	14	9.9	Z	21			
	2.4	單調隊列優化	4	4.5	Property	10	6.13	Numbers	14	10	tools	21			
	2.5	整體二分	4	5	Graph	10	6.14	Pollard-Rho	14				10.1	bitset	21
	2.6	斜率優化-動態凸包	4				5.1	2-SAT	10				6.15	Theorem	15
3	Data Structure	4	5.2	Bellman Ford	10	6.16	找實根	15	10.3	Counting Sort	21				
	3.1	2D BIT	4	5.3	Dijkstra	10	6.17	質因數分解	15	10.4	DuiPai	21			
	3.2	BinaryTrie	4	5.4	Dominator tree	10	7	Square root decomposition	16	10.5	HashMap	21			
	3.3	BIT	5	5.5	Floyd Warshall	10				10.6	pragma	21			
	3.4	DSU	5	5.6	SCC	10				10.7	relabel	22			
	3.5	Dynamic Segment Tree	5	5.7	SPFA	11				10.8	Template	22			
	3.6	Kruskal	5	5.8	判斷二分圖	11				10.9	template bubble	22			
				5.9	判斷平面圖	11				10.10	TenarySearch	22			
			5.10	判斷環	11	7.1	MoAlgo	16							
						7.2	分塊 cf455D	16							
						7.3	莫隊	16							

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
```