

# 1 Computational Geometry

## 1.1 Geometry

```

1 const double PI=atan2(0.0, -1.0);
2 template<typename T>
3 struct point{
4     T x,y;
5     point(){}
6     point(const T&x, const T&y):x(x),y(y){}
7     point operator+(const point &b) const{
8         return point(x+b.x,y+b.y); }
9     point operator-(const point &b) const{
10        return point(x-b.x,y-b.y); }
11     point operator*(const T &b) const{
12        return point(x*b,y*b); }
13     point operator/(const T &b) const{
14        return point(x/b,y/b); }
15     bool operator==(const point &b) const{
16        return x==b.x&&y==b.y; }
17     T dot(const point &b) const{
18        return x*b.x+y*b.y; }
19     T cross(const point &b) const{
20        return x*b.y-y*b.x; }
21     point normal() const{ //求法向量
22        return point(-y,x); }
23     T abs2() const{ //向量長度的平方
24        return dot(*this); }
25     T rad(const point &b) const{ //兩向量的弧度
26     return fabs(atan2(fabs(cross(b)),dot(b))); }
27     T getA() const{ //對x軸的弧度
28        T A=atan2(y,x); //超過180度會變負的
29        if(A<=-PI/2) A+=PI*2;
30        return A;
31     }
32 };
33 template<typename T>
34 struct line{
35     line(){}
36     point<T> p1,p2;
37     T a,b,c; //ax+by+c=0
38     line(const point<T>&x, const point<T>&y):p1
39         (x),p2(y){}
40     void pton() { //轉成一般式
41         a=p1.y-p2.y;
42         b=p2.x-p1.x;
43         c=-a*p1.x-b*p1.y;
44     }
45     T ori(const point<T> &p) const{ //點和有向直
46         線的關係 · >0左邊 · =0在線上 <0右邊
47         return (p2-p1).cross(p-p1);
48     }
49     T btw(const point<T> &p) const{ //點投影落在
50         線段上 <=0
51         return (p1-p).dot(p2-p);
52     }
53     bool point_on_segment(const point<T>&p)
54         const{ //點是否在線段上
55         return ori(p)==0&&btw(p)<=0;
56     }
57     T dis2(const point<T> &p, bool is_segment
58         =0) const{ //點跟直線/線段的距離平方
59
60         point<T> v=p2-p1,v1=v-p1;
61         if(is_segment){
62             point<T> v2=p-p1;
63             if(v.dot(v1)<=0) return v1.abs2();
64             if(v.dot(v2)>=0) return v2.abs2();
65         }
66         T tmp=v.cross(v1);
67         return tmp*tmp/v.abs2();
68     }
69     T seg_dis2(const line<T> &l) const{ //兩線段
70         距離平方
71         return min({dis2(l.p1,l),dis2(l.p2,l),l.
72             dis2(p1,l),l.dis2(p2,l)});
73     }
74     point<T> projection(const point<T> &p)
75         const{ //點對直線的投影
76         point<T> n=(p2-p1).normal();
77         return p-n*(p-p1).dot(n)/n.abs2();
78     }
79     point<T> mirror(const point<T> &p) const{
80         //點對直線的鏡射 · 要先呼叫pton轉成一般式
81         point<T> R;
82         T d=a*b+b*b;
83         R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
84         R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
85         return R;
86     }
87     bool equal(const line &l) const{ //直線相等
88         return ori(l.p1)==0&&ori(l.p2)==0;
89     }
90     bool parallel(const line &l) const{
91         return (p1-p2).cross(l.p1-l.p2)==0;
92     }
93     bool cross_seg(const line &l) const{
94         return (p2-p1).cross(l.p1-p1)*(p2-p1).
95             cross(l.p2-p1)<=0; //直線是否交線段
96     }
97     int line_intersect(const line &l) const{ //
98         直線相交情況 · -1無限多點 · 1交於一點 · 0
99         不相交
100        return parallel(l)?(ori(l.p1)==0?-1:0)
101            :1;
102    }
103    int seg_intersect(const line &l) const{
104        T c1=ori(l.p1), c2=ori(l.p2);
105        T c3=l.ori(p1), c4=l.ori(p2);
106        if(c1==0&&c2==0){ //共線
107            bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
108            T a3=1.btw(p1),a4=1.btw(p2);
109            if(b1&&b2&&a3==0&&a4==0) return 2;
110            if(b1&&b2&&a3>=0&&a4==0) return 3;
111            if(b1&&b2&&a3>=0&&a4>=0) return 0;
112            return -1; //無限交點
113        }else if(c1*c2<=0&&c3*c4<=0) return 1;
114        return 0; //不相交
115    }
116    point<T> line_intersection(const line &l)
117        const{ //直線交點 */
118        point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
119        //if(a.cross(b)==0) return INF;
120        return p1+a*(s.cross(b)/a.cross(b));
121    }
122    point<T> seg_intersection(const line &l)
123        const{ //線段交點
124
125        int res=seg_intersect(l);
126        if(res<=0) assert(0);
127        if(res==2) return p1;
128        if(res==3) return p2;
129        return line_intersection(l);
130    }
131 };
132 template<typename T>
133 struct polygon{
134     polygon(){}
135     vector<point<T> > p; //逆時針順序
136     T area() const{ //面積
137         T ans=0;
138         for(int i=p.size()-1,j=0;j<(int)p.size()
139             ;i=j++){
140             ans+=p[i].cross(p[j]);
141         }
142         return ans/2;
143     }
144     point<T> center_of_mass() const{ //重心
145         T cx=0,cy=0,w=0;
146         for(int i=p.size()-1,j=0;j<(int)p.size()
147             ;i=j++){
148             T a=p[i].cross(p[j]);
149             cx+=(p[i].x+p[j].x)*a;
150             cy+=(p[i].y+p[j].y)*a;
151             w+=a;
152         }
153         return point<T>(cx/3/w,cy/3/w);
154     }
155     char ahas(const point<T>& t) const{ //點是否
156         在簡單多邊形內 · 是的話回傳1 · 在邊上回
157         傳-1 · 否則回傳0
158         bool c=0;
159         for(int i=0,j=p.size()-1;i<p.size();j=i
160             ++){
161             if(line<T>(p[i],p[j]).point_on_segment
162                 (t)) return -1;
163             else if((p[i].y>t.y)!=p[j].y>t.y)&&
164                 t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j
165                     ].y-p[i].y)+p[i].x)
166                 c=!c;
167             return c;
168         }
169         char point_in_convex(const point<T>&x)
170             const{
171             int l=1,r=(int)p.size()-2;
172             while(l<r){ //點是否在凸多邊形內 · 是的話
173                 回傳1 · 在邊上回傳-1 · 否則回傳0
174                 int mid=(l+r)/2;
175                 T a1=(p[mid]-p[0]).cross(x-p[0]);
176                 T a2=(p[mid+1]-p[0]).cross(x-p[0]);
177                 if(a1>=0&&a2<=0){
178                     T res=(p[mid+1]-p[mid]).cross(x-p[
179                         mid]);
180                     return res>0?-1:(res==0?-1:0);
181                 }else if(a1<0) r=mid-1;
182                 else l=mid+1;
183             }
184             return 0;
185         }
186         vector<T> getA() const{ //凸包邊對x軸的夾角
187         vector<T> res; //一定是遞增的
188         for(size_t i=0;i<p.size();i++){
189             res.push_back((p[(i+1)%p.size()]-p[i])
190                 .getA());
191             return res;
192         }
193         bool line_intersect(const vector<T>&a,
194             const line<T> &l) const{ //0(LogN)
195             int f1=upper_bound(A.begin(),A.end(),(l.
196                 p1-l.p2).getA())-A.begin();
197             int f2=upper_bound(A.begin(),A.end(),(l.
198                 p2-l.p1).getA())-A.begin();
199             return l.cross_seg(line<T>(p[f1],p[f2]))
200                 ;
201         }
202         polygon cut(const line<T> &l) const{ //凸包
203             對直線切割 · 得到直線L左側的凸包
204             polygon ans;
205             for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
206                 if(l.ori(p[i])>=0){
207                     ans.p.push_back(p[i]);
208                     if(l.ori(p[j])<0)
209                         ans.p.push_back(l.
210                             line_intersection(line<T>(p[i
211                                 ],p[j])));
212                 }else if(l.ori(p[j])>0)
213                     ans.p.push_back(l.line_intersection(
214                         line<T>(p[i],p[j])));
215             }
216             return ans;
217         }
218         static bool monotone_chain_cmp(const point
219             <T>& a, const point<T>& b){ //凸包排序函
220             數
221             return (a.x<b.x)|| (a.x==b.x&&a.y<b.y);
222         }
223         void monotone_chain(vector<point<T> > &s){
224             //凸包
225             sort(s.begin(),s.end(),
226                 monotone_chain_cmp);
227             p.resize(s.size()+1);
228             int m=0;
229             for(size_t i=0;i<s.size();i++){
230                 while(m>=2&&(p[m-1]-p[m-2]).cross(s[i
231                     ]-p[m-2])<=0)--m;
232                 p[m++]=s[i];
233             }
234             for(int i=s.size()-2,t=m+1;i>=0;--i){
235                 while(m>=t&&(p[m-1]-p[m-2]).cross(s[i
236                     ]-p[m-2])<=0)--m;
237                 p[m++]=s[i];
238             }
239             if(s.size()>1)--m;
240             p.resize(m);
241         }
242         T diam() { //直徑
243             int n=p.size(),t=1;
244             T ans=0;p.push_back(p[0]);
245             for(int i=0;i<n;i++){
246                 point<T> now=p[i+1]-p[i];
247                 while(now.cross(p[t+1]-p[i])>now.cross
248                     (p[t]-p[i])) t=(t+1)%n;
249                 ans=max(ans,(p[i]-p[t]).abs2());
250             }
251             return p.pop_back(),ans;
252         }
253         T min_cover_rectangle() { //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]-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
248     sort(s.begin(),s.end(),angle_cmp);//線段
249     //左側為該線段半平面
250     int L,R,n=s.size();
251     vector<point<T> > px(n);
252     vector<line<T> > q(n);
253     q[L=R=0]=s[0];
254     for(int i=1;i<n;++i){
255         while(L<R&&s[i].ori(px[R-1])<=0)--R;
256         while(L<R&&s[i].ori(px[L])<=0)++L;
257         q[++R]=s[i];
258         if(q[R].parallel(q[R-1])){
259             --R;
260             if(q[R].ori(s[i].p1)>0)q[R]=s[i];
261         }
262         if(L<R)px[R-1]=q[R-1].
263             line_intersection(q[R]);
264     }
265     while(L<R&&q[L].ori(px[R-1])<=0)--R;
266     p.clear();
267
268     if(R-L<=1)return 0;
269     px[R]=q[R].line_intersection(q[L]);
270     for(int i=L;i<R;++i)p.push_back(px[i]);
271     return R-L+1;
272 }
273
274 template<typename T>
275 struct triangle{
276     point<T> a,b,c;
277     triangle(){
278         triangle(const point<T> &a,const point<T>
279               &b,const point<T> &c):a(a),b(b),c(c){}
280     }
281     T area()const{
282         T t=(b-a).cross(c-a)/2;
283         return t>0?t:-t;
284     }
285     point<T> barycenter()const{//重心
286         return (a+b+c)/3;
287     }
288     point<T> circumcenter()const{//外心
289         static line<T> u,v;
290         u.p1=(a+b)/2;
291         u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
292               b.x);
293         v.p1=(a+c)/2;
294         v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
295               c.x);
296         return u.line_intersection(v);
297     }
298     point<T> incenter()const{//內心
299         T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
300               ()),C=sqrt((a-b).abs2());
301         return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
302               B*b.y+C*c.y)/(A+B+C);
303     }
304     point<T> perpencenter()const{//垂心
305         return barycenter()*3-circumcenter()*2;
306     }
307 }
308
309 template<typename T>
310 struct point3D{
311     T x,y,z;
312     point3D(){
313         point3D(const T&x,const T&y,const T&z):x(x
314               ),y(y),z(z){}
315     }
316     point3D operator+(const point3D &b)const{
317         return point3D(x+b.x,y+b.y,z+b.z);}
318     point3D operator-(const point3D &b)const{
319         return point3D(x-b.x,y-b.y,z-b.z);}
320     point3D operator*(const T &b)const{
321         return point3D(x*b,y*b,z*b);}
322     point3D operator/(const T &b)const{
323         return point3D(x/b,y/b,z/b);}
324     bool operator==(const point3D &b)const{
325         return x==b.x&&y==b.y&&z==b.z;}
326     T dot(const point3D &b)const{
327         return x*b.x+y*b.y+z*b.z;}
328     point3D cross(const point3D &b)const{
329         return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
330               *b.y-y*b.x);}
331     T abs2()const{//向量長度的平方
332         return dot(*this);}
333     T area2(const point3D &b,const point3D &c)const{//和b、原點
334         //圍成面積的平方
335         return cross(b).abs2()/4;}
336 };
337
338 template<typename T>
339 struct line3D{
340     point3D<T> p1,p2;
341     line3D(){
342         line3D(const point3D<T> &p1,const point3D<
343               T> &p2):p1(p1),p2(p2){}
344     }
345     T dis2(const point3D<T> &p,bool is_segment
346           =0)const{//點跟直線/線段的距離平方
347         point3D<T> v=p2-p1,v1=p-p1;
348         if(is_segment){
349             point3D<T> v2=p-p2;
350             if(v.dot(v1)<=0)return v1.abs2();
351             if(v.dot(v2)>=0)return v2.abs2();
352         }
353         point3D<T> tmp=v.cross(v1);
354         return tmp.abs2()/v.abs2();
355     }
356 }
357
358 pair<point3D<T>,point3D<T> > closest_pair(
359     const line3D<T> &l1,const
360     point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
361     point3D<T> N=v1.cross(v2),ab(p1-l.p1);
362     //if(N.abs2()==0)return NULL;平行或重合
363     T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
364     最近點對距離
365     point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
366     cross(d2),G=l.p1-p1;
367     T t1=(G.cross(d2)).dot(D)/D.abs2();
368     T t2=(G.cross(d1)).dot(D)/D.abs2();
369     return make_pair(p1+d1*t1,l.p1+d2*t2);
370 }
371
372 bool same_side(const point3D<T> &a,const
373     point3D<T> &b)const{
374     return (p2-p1).cross(a-p1).dot((p2-p1).
375           cross(b-p1))>0;
376 }
377
378 template<typename T>
379 struct plane{
380     point3D<T> p0,n;//平面上的點和法向量
381     plane(){
382         plane(const point3D<T> &p0,const point3D<T>
383               &n):p0(p0),n(n){}
384     }
385     T dis2(const point3D<T> &p)const{//點到平
386           面距離的平方
387     T tmp=(p-p0).dot(n);
388     return tmp*tmp/n.abs2();
389 }
390
391 point3D<T> projection(const point3D<T> &p)
392     const{
393     return p-n*(p-p0).dot(n)/n.abs2();
394 }
395
396 point3D<T> line_intersection(const line3D<
397     T> &l1,const
398     T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
399     重合該平面
400     return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
401           tmp);
402 }
403
404 line3D<T> plane_intersection(const plane &
405     p1)const{
406     point3D<T> e=n.cross(p1.n),v=n.cross(e);
407     T tmp=p1.n.dot(v);//等於0表示平行或重合
408     該平面
409
410     point3D<T> q=p0+(v*(p1.n.dot(p1.p0-p0))/
411           tmp);
412     return line3D<T>(q,q+e);
413 }
414
415 }
416
417 1.2 MinCircleCover
418
419 1 const double eps = 1e-10;
420 2 int sign(double a){
421 3     return fabs(a)<eps?0:a>0?1:-1;
422 4 }
423
424 template<typename T>
425 T len(point<T> p){
426     return sqrt(p.dot(p));
427 }
428
429 template<typename T>
430 point<T> findCircumcenter(point<T> A,point<T>
431     B,point<T> C){
432     point<T> AB = B-A;
433     point<T> AC = C-A;
434     T AB_len_sq = AB.x*AB.x+AB.y*AB.y;
435     T AC_len_sq = AC.x*AC.x+AC.y*AC.y;
436     T D = AB.x*AC.y-AB.y*AC.x;
437     T X = A.x+(AC.y*AB_len_sq-AB.y*AC_len_sq)
438           /(2*D);
439     T Y = A.y+(AB.x*AC_len_sq-AC.x*AB_len_sq)
440           /(2*D);
441     return point<T>(X,Y);
442 }
443
444 template<typename T>
445 pair<T, point<T>> MinCircleCover(vector<
446     point<T>> &p){
447     // 回傳最小覆蓋圓{半徑, 中心}
448     random_shuffle(p.begin(),p.end());
449     int n = p.size();
450     point<T> c = p[0]; T r = 0;
451     for(int i=1;i<n;i++){
452         if(sign(len(c-p[i])-r) > 0){ // 不在圓內
453             c = p[i], r = 0;
454             for(int j=0;j<i;j++){
455                 if(sign(len(c-p[j])-r) > 0) {
456                     c = (p[i]+p[j])/2.0;
457                     r = len(c-p[i]);
458                     for(int k=0;k<j;k++){
459                         if(sign(len(c-p[k])-r) > 0){
460                             //c=triangle<T>(p[i],p[j],p[k]).
461                             circumcenter();
462                             c = findCircumcenter(p[i],p[j]
463                                   ),p[k]);
464                             r = len(c-p[i]);
465                         }
466                     }
467                 }
468             }
469         }
470     }
471     return make_pair(r,c);
472 }

```

### 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);

```

```

34 } else {
35     x->p = floor_div(y->m - x->m, x->k - y->k);
36 }
37 return x->p >= y->p;
38 }
39
40 void add_line(long long k, long long m) {
41     if(!MAX) {
42         k = -k;
43         m = -m;
44     }
45     auto z = insert({k, m, 0}), y = z++, x = y;
46     while(isect(y, z)) {
47         z = erase(z);
48     }
49     if(x != begin() && isect(--x, y)) {
50         isect(x, y = erase(y));
51     }
52     while((y = x) != begin() && (--x)->p >= y->p) {
53         isect(x, erase(y));
54     }
55 }
56
57 long long get(long long x) {
58     assert(!empty());
59     auto l = *lower_bound(x);
60     return (l.k * x + l.m) * (MAX ? +1 : -1);
61 }
62 };

```

## 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;

```

## 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); k++) {
8             if (DP[mid] > f(k) + w(k, mid)) {
9                 DP[mid] = f(k) + w(k, mid);
10                opt = k;
11            }
12        }
13        compute(L, mid - 1, optL, opt);
14        compute(mid + 1, R, opt, optR);
15    }
16    // compute(1, n, 0, n);

```

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

```

```

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

```

## 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 + 1);
12         for(auto &b : bit) b = 0;
13     }
14     int query(int x) const {
15         int sum = 0;
16         for(; x; x -= lowbit(x))
17             sum += bit[x];
18     }

```

```

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
27 class BIT2D {
28     int m;
29     vector<BIT> bit1D;
30
31 public:
32     void init(int _m, int _n) {
33         m = _m;
34         bit1D.resize(m + 1);
35         for(auto &b : bit1D) b.init(_n);
36     }
37     int query(int x, int y) const {
38         int sum = 0;
39         for(; x; x -= lowbit(x))
40             sum += bit1D[x].query(y);
41         return sum;
42     }
43     void modify(int x, int y, int val) {
44         for(; x <= m; x += lowbit(x))
45             bit1D[x].modify(y, val);
46     }
47 };

```

## 3.2 BinaryTrie

```

1 template<class T>
2 struct binary_trie {
3 public:
4     binary_trie() {
5         new_node();
6     }
7
8     void clear() {
9         trie.clear();
10        new_node();
11    }
12
13    void insert(T x) {
14        for(int i = B - 1, p = 0; i >= 0; i--) {
15            int y = x >> i & 1;
16            if(trie[p].go[y] == 0) {
17                trie[p].go[y] = new_node();
18            }
19            p = trie[p].go[y];
20            trie[p].cnt += 1;
21        }
22    }
23
24    void erase(T x) {
25        for(int i = B - 1, p = 0; i >= 0; i--) {
26            p = trie[p].go[x >> i & 1];
27            trie[p].cnt -= 1;
28        }
29    }
30
31    bool contains(T x) {

```



```

32 for(int i = B - 1, p = 0; i >= 0; i--) {
33     p = trie[p].go[x >> i & 1];
34     if(trie[p].cnt == 0) {
35         return false;
36     }
37 }
38 return true;
39 }
40
41 T get_min() {
42     return get_xor_min(0);
43 }
44
45 T get_max() {
46     return get_xor_max(0);
47 }
48
49 T get_xor_min(T x) {
50     T ans = 0;
51     for(int i = B - 1, p = 0; i >= 0; i--) {
52         int y = x >> i & 1;
53         int z = trie[p].go[y];
54         if(z > 0 && trie[z].cnt > 0) {
55             p = z;
56         } else {
57             ans |= T(1) << i;
58             p = trie[p].go[y ^ 1];
59         }
60     }
61     return ans;
62 }
63
64 T get_xor_max(T x) {
65     T ans = 0;
66     for(int i = B - 1, p = 0; i >= 0; i--) {
67         int y = x >> i & 1;
68         int z = trie[p].go[y ^ 1];
69         if(z > 0 && trie[z].cnt > 0) {
70             ans |= T(1) << i;
71             p = z;
72         } else {
73             p = trie[p].go[y];
74         }
75     }
76     return ans;
77 }
78
79 private:
80 static constexpr int B = sizeof(T) * 8;
81
82 struct Node {
83     std::array<int, 2> go = {};
84     int cnt = 0;
85 };
86
87 std::vector<Node> trie;
88
89 int new_node() {
90     trie.emplace_back();
91     return (int) trie.size() - 1;
92 }
93 };

```

### 3.3 BIT

```

1 #define lowbit(x) x & -x
2
3 void modify(vector<int> &bit, int idx, int
4     val) {
5     for(int i = idx; i <= bit.size(); i+=
6         lowbit(i)) bit[i] += val;
7 }
8
9 int query(vector<int> &bit, int idx) {
10     int ans = 0;
11     for(int i = idx; i > 0; i-= lowbit(i)) ans
12         += bit[i];
13     return ans;
14 }
15
16 // the first i s.t. a[1]+...+a[i] >= k
17 int findK(vector<int> &bit, int k) {
18     int idx = 0, res = 0;
19     int mx = __lg(bit.size()) + 1;
20     for(int i = mx; i >= 0; i--) {
21         if((idx | (1<<i)) > bit.size()) continue
22         ;
23         if(res + bit[idx | (1<<i)] < k) {
24             idx = (idx | (1<<i));
25             res += bit[idx];
26         }
27     }
28     return idx + 1;
29 }
30
31 //O(n)建bit
32 for (int i = 1; i <= n; ++i) {
33     bit[i] += a[i];
34     int j = i + lowbit(i);
35     if (j <= n) bit[j] += bit[i];
36 }

```

### 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.unit(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
6 // [1,n]
7 // tag[i] represents the modifications to be
8 // applied to the children,
9 // while seg[i] has already been modified.
10 ll seg[N << 2], tag[N << 2];
11 int n;
12
13 void pull(int id) {
14     seg[id] = seg[lc(id)] + seg[rc(id)];
15 }
16
17 void push(int id, int l, int r) {
18     if (tag[id]) {
19         int m = (l + r) >> 1;
20         tag[lc(id)] += tag[id], tag[rc(id)] +=
21             tag[id];
22         seg[lc(id)] += (m - l + 1) * tag[id],
23             seg[rc(id)] += (r - m) * tag[id];
24         tag[id] = 0;
25     }
26 }
27
28 void upd(int ql, int qr, ll v, int l = 1,
29     int r = n, int id = 1) {
30     if (ql <= l && r <= qr) return tag[id] +=
31         v, seg[id] += (r - l + 1) * v, void();
32     push(id, l, r);
33     int m = (l + r) >> 1;
34     if (ql <= m) upd(ql, qr, v, l, m, lc(id));
35     if (qr > m) upd(ql, qr, v, m + 1, r, rc(id)
36     );
37     pull(id);
38 }
39
40 ll qry(int ql, int qr, int l = 1, int r = n
41     , int id = 1) {
42     if (ql <= l && r <= qr) return seg[id];
43     push(id, l, r);
44     int m = (l + r) >> 1; ll ret = 0;
45     if (ql <= m) ret += qry(ql, qr, l, m, lc(
46         id));
47     if (qr > m) ret += qry(ql, qr, m + 1, r,
48         rc(id));
49     return ret;
50 }

```

### 3.8 LiChaoST

```

1 struct L {
2     ll m, k, id;
3     L() : id(-1) {}
4     L(ll a, ll b, ll c) : m(a), k(b), id(c) {}
5     ll at(ll x) { return m * x + k; }
6 };
7
8 class LiChao { // maintain max
9 private:
10     int n; vector<L> nodes;
11     void insert(int l, int r, int rt, L ln) {
12         int m = (l + r) >> 1;
13         if (nodes[rt].id == -1)
14             return nodes[rt] = ln, void();

```

```

14 bool atLeft = nodes[rt].at(1) < ln.at(1)
15 ;
16 if (nodes[rt].at(m) < ln.at(m))
17     atLeft ^= 1, swap(nodes[rt], ln);
18 if (r - 1 == 1) return;
19 if (atLeft) insert(1, m, rt << 1, ln);
20 else insert(m, r, rt << 1 | 1, ln);
21 }
22 ll query(int l, int r, int rt, ll x) {
23     int m = (l + r) >> 1; ll ret = -INF;
24     if (nodes[rt].id != -1) ret = nodes[rt].at(x);
25     if (r - l == 1) return ret;
26     if (x < m) return max(ret, query(l, m, rt << 1, x));
27     return max(ret, query(m, r, rt << 1 | 1, x));
28 }
29 public:
30 LiChao(int n_) : n(n_), nodes(n * 4) {}
31 void insert(L ln) { insert(0, n, 1, ln); }
32 ll query(ll x) { return query(0, n, 1, x); }
33 };

```

### 3.9 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 - k) dq.pop_front();
7         while(dq.size() && num[dq.back()] < num[i]) dq.pop_back();
8         dq.emplace_back(i);
9         if(i >= k - 1) ans.emplace_back(num[dq.front()]);
10    }
11    return ans;
12 }

```

### 3.10 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().first >= h[i]) {
8             corner = stick.top().second;
9             ans = max(ans, 1LL * (i - corner) * stick.top().first);
10            stick.pop();
11        }
12    }

```

```

13 stick.emplace(h[i], corner);
14 }
15 return ans;
16 }

```

### 3.11 pbds

```

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

```

### 3.12 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 };

```

### 3.13 Persistent Segment Tree

```

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

```

### 3.14 Prim

```

1 int cost[MAX_V][MAX_V]; // Edge 的權重 (不存在時為 INF)
2 int mincost[MAX_V]; // 來自集合 X 的邊的最小權重
3 bool used[MAX_V]; // 頂點 i 是否包含在 X 之中
4 int V; // 頂點數
5
6 int prim() {
7     for(int i = 0; i < V; i++) {
8         mincost[i] = INF;
9         used[i] = false;
10    }
11    mincost[0] = 0;
12    int res = 0;

```

```

13 while(true) {
14     int v = -1;
15     // 從不屬於 X 的頂點中尋找會讓來自 X 的邊之權重最小的頂點
16     for(int u = 0; u < V; u++) {
17         if(!used[u] && (v == -1 || mincost[u] < mincost[v])) v = u;
18     }
19     if(v == -1) break;
20     used[v] = true; // 將頂點 v 追加至 X
21     res += mincost[v]; // 加上邊的權重
22     for(int u = 0; u < V; u++) {
23         mincost[u] = min(mincost[u], cost[v][u]);
24     }
25 }
26 return res;
27 }

```

### 3.15 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 id 的節點，存的區間為 [l, r]
6     if (l == r) {
7         seg[id] = a[l]; // 葉節點的值
8         return;
9     }
10    int mid = (l + r) / 2; // 將區間切成兩半
11    build(id * 2, l, mid); // 左子節點
12    build(id * 2 + 1, mid + 1, r); // 右子節點
13    seg[id] = seg[id * 2] + seg[id * 2 + 1];
14 }
15 // 區間查詢
16
17 int query(int id, int l, int r, int ql, int qr) {
18     if (r < ql || qr < l) return 0; // 若目前的區間與詢問的區間的交集為空的話，return 0
19     if (ql <= l && r <= qr) return seg[id]; // 若目前的區間是詢問的區間的子集的話，則終止，並回傳當前節點的答案
20     int mid = (l + r) / 2;
21     return query(id * 2, l, mid, ql, qr) // 左
22         + query(id * 2 + 1, mid + 1, r, ql, qr); // 右
23 // 否則，往左、右進行遞迴
24 }
25 // 單點修改
26

```

```

31 void modify(int id, int l, int r, int i, int
    x) {
32     if (l == r) {
33         seg[id] = x; // 將a[i]改成x
34         //seg[id] += x; // 將a[i]加上x
35         return;
36     }
37     int mid = (l + r) / 2;
38     // 根據修改的點在哪裡，來決定要往哪個子
    樹進行DFS
39     if (i <= mid) modify(id * 2, l, mid, i,
        x); //左
40     else modify(id * 2 + 1, mid + 1, r, i, x
        ); //右
41     seg[id] = seg[id * 2] + seg[id * 2 + 1];
42 }

```

### 3.16 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
                +(1<<(i-1))]);
13        }
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 }

```

### 3.17 TimingSegmentTree

```

1 template<class T, class D> struct
    timing_segment_tree {
2     struct node {
3         int l, r;
4         vector<T> opt;
5     };

```

```

6     vector<node> arr;
7     void build(int l, int r, int idx = 1) {
8         if (idx == 1) arr.resize((r-l+1)<<2);
9         if (l == r) {
10            arr[idx].l = arr[idx].r = l;
11            arr[idx].opt.clear();
12            return;
13        }
14        int m = (l+r)>>1;
15        build(l, m, idx<<1);
16        build(m+1, r, idx<<1|1);
17        arr[idx].l = l, arr[idx].r = r;
18        arr[idx].opt.clear();
19    }
20    void update(int ql, int qr, T k, int idx = 1)
        {
21        if (ql<=arr[idx].l and arr[idx].r<=qr) {
22            arr[idx].opt.push_back(k);
23            return;
24        }
25        int m = (arr[idx].l+arr[idx].r)>>1;
26        if (ql<=m) update(ql, qr, k, idx<<1);
27        if (qr>m) update(ql, qr, k, idx<<1|1);
28    }
29    void dfs(D &d, vector<int>&ans, int idx = 1)
        {
30        int cnt = 0;
31        for(auto [a, b]: arr[idx].opt) {
32            if (d.Union(a, b)) cnt++;
33        }
34        if (arr[idx].l == arr[idx].r) ans[arr[idx].l
            ] = d.comps;
35        else {
36            dfs(d, ans, idx<<1);
37            dfs(d, ans, idx<<1|1);
38        }
39        while (cnt--) d.undo();
40    }
41 }

```

### 3.18 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 }

```

### 3.19 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, 8000004> 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
    qr, int x) {
41     if (ql > r || qr < l) return;
42     if (ql <= l && qr >= r) {
43         man[p] += x;
44         tag[p] += x;
45         return;
46     }
47     push(p);
48     update(lc, l, mid, ql, qr, x);

```

```

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

```

```

107 // CSES Intersection Points
108 #include <bits/stdc++.h>
109 #define int long long
110 #define pb push_back
111 using namespace std;
112 struct line{
113     int p, l, r;
114 };
115 const int inf = 1e6 + 1;
116 array<int, 2000004> BIT;
117 vector<line> A, Q;
118 bool cmp(line a, line b){
119     return a.p < b.p;
120 }
121 void update(int p, int x){
122     for(; p < 2000004; p += p & -p) BIT[p]
        += x;
123 }
124 int query(int p){
125     int sum = 0;
126     for(; p; p -= p & -p) sum += BIT[p];
127     return sum;
128 }
129 int run(){
130     int ans = 0, p = 0;
131     for(auto [t, l, r] : Q){
132         while(p < A.size()){
133             auto [x, y, v] = A[p];
134             if(x > t) break;
135             update(y, v);
136             p++;
137         }
138         ans += query(r) - query(l - 1);
139     }
140     return ans;
141 }
142 signed main(){
143     int n, x1, x2, y1, y2;
144     cin >> n;
145     for(int i = 0; i < n; i++){
146         cin >> x1 >> y1 >> x2 >> y2;
147         x1 += inf, x2 += inf, y1 += inf, y2
            += inf;
148         if(x1 == x2) Q.pb({x1, y1, y2});
149         else A.pb({x1, y1, 1}), A.pb({x2 +
            1, y2, -1});
150     }
151     sort(Q.begin(), Q.end(), cmp);
152     sort(A.begin(), A.end(), cmp);
153     cout << run() << "\n";
154     return 0;
155 }

```

### 3.20 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),
        Sum(Val),Tag(false) {}

```

```

6     void pull();
7     bool Tag;
8     void push();
9 } *root;
10
11 inline unsigned sz(Treap *x) {
12     return x ? x->sz:0;
13 }
14
15 inline void Treap::push() {
16     if(!Tag) return;
17     swap(lc,rc);
18     if(lc) lc->Tag ^= Tag;
19     if(rc) rc->Tag ^= Tag;
20     Tag = false;
21 }
22
23 inline void Treap::pull() {
24     sz = 1;
25     Sum = Val;
26     if(lc) {
27         sz += lc->sz;
28         Sum += lc->Sum;
29     }
30     if(rc) {
31         sz += rc->sz;
32         Sum += rc->Sum;
33     }
34 }
35
36 Treap *merge(Treap *a, Treap *b) {
37     if(!a || !b) return a ? a : b;
38     if(a->pri < b->pri) {
39         a->push();
40         a->rc = merge(a->rc,b);
41         a->pull();
42         return a;
43     }
44     else {
45         b->push();
46         b->lc = merge(a,b->lc);
47         b->pull();
48         return b;
49     }
50 }
51
52 pair<Treap *,Treap *> splitK(Treap *x,
53     unsigned K) {
54     Treap *a = nullptr, *b = nullptr;
55     if(!x) return {a,b};
56     x->push();
57     unsigned leftSize = sz(x->lc) + 1;
58     if(K >= leftSize) {
59         a = x;
60         tie(a->rc,b) = splitK(x->rc, K -
61             leftSize);
62     }
63     else {
64         b = x;
65         tie(a, b->lc) = splitK(x->lc, K);
66     }
67     x->pull();
68     return {a,b};
69 }
70
71 Treap *init(const vector<int> &a) {

```

```

72     Treap *root = nullptr;
73     for(size_t i = 0; i < a.size(); i++) {
74         root = merge(root,new Treap(a[i]));
75     }
76     return root;
77 }
78
79 long long query(Treap *&root, unsigned ql,
80     unsigned qr) {
81     auto [a,b] = splitK(root,ql);
82     auto [c,d] = splitK(b,qr-ql+1);
83     c->push();
84     long long Sum = c->Sum;
85     root = merge(a,merge(c,d));
86     return Sum;
87 }
88
89 void Reverse(Treap *&root, unsigned ql,
90     unsigned qr) {
91     auto [a,b] = splitK(root,ql);
92     auto [c,d] = splitK(b,qr-ql+1);
93     c->Tag ^= true;
94     root = merge(a, merge(c,d));
95 }

```

## 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        h[s] = 0;
23        queue<int> que;
24        que.push(s);
25        while(!que.empty()) {
26            int u = que.front();
27            que.pop();
28            for(auto id : g[u]) {
29                const edge& e = edges[id];
30                int v = e.to;
31                if(e.cap > 0 && h[v] == -1) {
32                    h[v] = h[u] + 1;

```

```

33                if(v == t) {
34                    return 1;
35                }
36                que.push(v);
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            return f - r;
61        }
62        T flow(int s, int t, T f = numeric_limits<
63            T>::max()) {
64            T ans = 0;
65            while(f > 0 && bfs(s, t)) {
66                cur.assign(n, 0);
67                T send = dfs(s, t, f);
68                ans += send;
69                f -= send;
70            }
71            return ans;
72        }
73        vector<pair<int,int>> min_cut(int s) {
74            vector<bool> vis(n);
75            vis[s] = true;
76            queue<int> que;
77            que.push(s);
78            while(!que.empty()) {
79                int u = que.front();
80                que.pop();
81                for(auto id : g[u]) {
82                    const auto& e = edges[id];
83                    int v = e.to;
84                    if(e.cap > 0 && !vis[v]) {
85                        vis[v] = true;
86                        que.push(v);
87                    }
88                }
89            }
90            vector<pair<int,int>> cut;
91            for(int i = 0; i < (int) edges.size(); i
92                += 2) {
93                const auto& e = edges[i];
94                if(vis[e.from] && !vis[e.to]) {
95                    cut.push_back(make_pair(e.from, e.to
96                        ));

```



```

94     }
95 }
96 return cut;
97 }
98 };
99
100 //CSES Distinct Routes
101 #include <bits/stdc++.h>
102
103 using namespace std;
104
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
112 struct Dinic {
113     const long long flow_inf = 1e18;
114     vector<FlowEdge> edges;
115     vector<vector<int>> adj;
116     int n, m = 0;
117     int s, t;
118     vector<int> level, ptr, path;
119     vector<vector<int>> paths;
120     queue<int> q;
121
122     Dinic(int n, int s, int t) : n(n), s(s),
123         t(t) {
124         adj.resize(n);
125         level.resize(n);
126         ptr.resize(n);
127     }
128
129     void add_edge(int v, int u, long long
130         cap) {
131         edges.emplace_back(v, u, cap);
132         edges.emplace_back(u, v, 0);
133         adj[v].push_back(m);
134         adj[u].push_back(m + 1);
135         m += 2;
136     }
137
138     bool bfs() {
139         while (!q.empty()) {
140             int v = q.front();
141             q.pop();
142             for (int id : adj[v]) {
143                 if (edges[id].cap - edges[id]
144                     .flow < 1)
145                     continue;
146                 if (level[edges[id].u] !=
147                     -1)
148                     continue;
149                 level[edges[id].u] = level[v]
150                     + 1;
151                 q.push(edges[id].u);
152             }
153         }
154         return level[t] != -1;
155     }
156
157     long long dfs(int v, long long pushed) {
158         if (pushed == 0)
159             return 0;
160
161         path.push_back(v);
162         if (v == t) {
163             for (int iiddxx = 0; iiddxx <
164                 pushed; ++iiddxx)
165                 paths.push_back(path);
166             path.pop_back();
167             return pushed;
168         }
169         for (int& cid = ptr[v]; cid < (int)
170             adj[v].size(); cid++) {
171             int id = adj[v][cid];
172             int u = edges[id].u;
173             if (level[v] + 1 != level[u] ||
174                 edges[id].cap - edges[id].
175                 flow < 1)
176                 continue;
177             long long tr = dfs(u, min(pushed
178                 , edges[id].cap - edges[id].
179                 flow));
180             if (tr == 0)
181                 continue;
182             edges[id].flow += tr;
183             edges[id ^ 1].flow -= tr;
184             path.pop_back();
185             return tr;
186         }
187         path.pop_back();
188         return 0;
189     }
190
191     long long flow() {
192         long long f = 0;
193         while (true) {
194             fill(level.begin(), level.end(),
195                 -1);
196             level[s] = 0;
197             q.push(s);
198             if (!bfs())
199                 break;
200             fill(ptr.begin(), ptr.end(), 0);
201             while (long long pushed = dfs(s,
202                 flow_inf)) {
203                 f += pushed;
204             }
205         }
206         return f;
207     }
208 };
209
210 int main() {
211     int n, m, v, u;
212     cin >> n >> m;
213     Dinic D(n+1, 1, n);
214     for (int i = 0; i < m; ++i) {
215         cin >> v >> u;
216         D.add_edge(v, u, 1);
217     }
218     D.flow();
219     Dinic FLOW(n+1, 1, n);
220     for (auto e : D.edges) {
221         if (e.flow > 0) {
222             FLOW.add_edge(e.v, e.u, 1);
223         }
224     }
225     cout << FLOW.flow() << "\n";
226     for (auto p : FLOW.paths) {

```

```

154     path.push_back(v);
155     if (v == t) {
156         for (int iiddxx = 0; iiddxx <
157             pushed; ++iiddxx)
158             paths.push_back(path);
159         path.pop_back();
160         return pushed;
161     }
162     for (int& cid = ptr[v]; cid < (int)
163         adj[v].size(); cid++) {
164         int id = adj[v][cid];
165         int u = edges[id].u;
166         if (level[v] + 1 != level[u] ||
167             edges[id].cap - edges[id].
168             flow < 1)
169             continue;
170         long long tr = dfs(u, min(pushed
171             , edges[id].cap - edges[id].
172             flow));
173         if (tr == 0)
174             continue;
175         edges[id].flow += tr;
176         edges[id ^ 1].flow -= tr;
177         path.pop_back();
178         return tr;
179     }
180     path.pop_back();
181     return 0;
182 }
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
202 int main() {
203     int n, m, v, u;
204     cin >> n >> m;
205     Dinic D(n+1, 1, n);
206     for (int i = 0; i < m; ++i) {
207         cin >> v >> u;
208         D.add_edge(v, u, 1);
209     }
210     D.flow();
211     Dinic FLOW(n+1, 1, n);
212     for (auto e : D.edges) {
213         if (e.flow > 0) {
214             FLOW.add_edge(e.v, e.u, 1);
215         }
216     }
217     cout << FLOW.flow() << "\n";
218     for (auto p : FLOW.paths) {

```

```

212     cout << p.size() << "\n";
213     for (auto verti : p)
214         cout << verti << " ";
215     cout << "\n";
216 }
217
218 return 0;
219 }

```

## 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){

```

```

25     if(u==t)return CF;
26     T tf=CF, df;
27     for(int &i=cur[u]; ~i; i=e[i].pre){
28         if(e[i].r&&d[u]==d[e[i].v]+1){
29             df=dfs(e[i].v, s, t, min(tf, e[i].r));
30             e[i].r-=df;
31             e[i^1].r+=df;
32             if(!(tf-=df)||d[s]==n)return CF-tf;
33         }
34     }
35     int mh=n;
36     for(int i=cur[u]=g[u]; ~i; i=e[i].pre){
37         if(e[i].r&&d[e[i].v]<mh)mh=d[e[i].v];
38     }
39     if(!--gap[d[u]])d[s]=n;
40     else ++gap[d[u]]=++mh;
41     return CF-tf;
42 }
43
44 T isap(int s, int t, bool clean=true){
45     memset(d, 0, sizeof(int)*(n+1));
46     memset(gap, 0, sizeof(int)*(n+1));
47     memcpy(cur, g, sizeof(int)*(n+1));
48     if(clean) for(size_t i=0; i<e.size(); ++i)
49         e[i].r=e[i].cap;
50     T MF=0;
51     for(gap[0]=n; d[s]<n; )MF+=dfs(s, s, t);
52     return MF;
53 }
54
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
64 T min_cut(int s, int t){
65     T ans=isap(s, t);
66     memset(vis, 0, sizeof(bool)*(n+1));
67     dfs_cut(s); cut_e.clear();
68     for(int u=0; u<n; ++u)if(vis[u])
69         for(int i=g[u]; ~i; i=e[i].pre)
70             if(!vis[e[i].v])cut_e.push_back(i);
71     return ans;
72 }
73 }

```

## 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     };

```

```

12 static constexpr Cap_t EPS = static_cast<
    Cap_t>(1e-9);
13
14 int n;
15 vector<Edge> edges;
16 vector<vector<int>> g;
17 vector<Cost_t> d;
18 vector<bool> in_queue;
19 vector<int> previous_edge;
20
21 MCMF() {}
22 MCMF(int _n) : n(_n+1), g(_n+1), d(_n+1),
    in_queue(_n+1), previous_edge(_n+1) {}
23
24 void add_edge(int u, int v, Cap_t cap,
    Cost_t cost) {
25     assert(0 <= u && u < n);
26     assert(0 <= v && v < n);
27     g[u].push_back(edges.size());
28     edges.emplace_back(u, v, cap, cost);
29     g[v].push_back(edges.size());
30     edges.emplace_back(v, u, 0, -cost);
31 }
32
33 bool spfa(int s, int t) {
34     bool found = false;
35     fill(d.begin(), d.end(), numeric_limits<
        Cost_t>::max());
36     d[s] = 0;
37     in_queue[s] = true;
38     queue<int> que;
39     que.push(s);
40     while(!que.empty()) {
41         int u = que.front();
42         que.pop();
43         if(u == t) {
44             found = true;
45         }
46         in_queue[u] = false;
47         for(auto& id : g[u]) {
48             const Edge& e = edges[id];
49             if(e.cap > EPS && d[u] + e.cost < d[
                e.to]) {
50                 d[e.to] = d[u] + e.cost;
51                 previous_edge[e.to] = id;
52                 if(!in_queue[e.to]) {
53                     que.push(e.to);
54                     in_queue[e.to] = true;
55                 }
56             }
57         }
58     }
59     return found;
60 }
61
62 pair<Cap_t, Cost_t> flow(int s, int t,
    Cap_t f = numeric_limits<Cap_t>::max()
    ) {
63     assert(0 <= s && s < n);
64     assert(0 <= t && t < n);
65     Cap_t cap = 0;
66     Cost_t cost = 0;
67     while(f > 0 && spfa(s, t)) {
68         Cap_t send = f;
69         int u = t;
70         while(u != s) {

```

```

71         const Edge& e = edges[previous_edge[
            u]];
72         send = min(send, e.cap);
73         u = e.from;
74     }
75     u = t;
76     while(u != s) {
77         Edge& e = edges[previous_edge[u]];
78         e.cap -= send;
79         Edge& b = edges[previous_edge[u] ^
            1];
80         b.cap += send;
81         u = e.from;
82     }
83     cap += send;
84     f -= send;
85     cost += send * d[t];
86 }
87 return make_pair(cap, cost);
88 }
89 }

```

## 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,
        bool y){
15        if(!x)u = inv(u);
16        if(!y)v = inv(v);
17        s.add_edge(inv(u), v);
18        s.add_edge(inv(v), u);
19    }
20    void check(){
21        if(have_ans!=0)return;
22        s.build();
23        for(int i = 0;i<n;++i){

```

```

24         if(s.scc[i]==s.scc[inv(i)]){
25             have_ans = -1;
26             return;
27         }
28         ans[i] = (s.scc[i]<s.scc[inv(i)]);
29     }
30     have_ans = 1;
31 }
32 }

```

## 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        if (!update)
19            break;
20        if (i == N) // && update
21            return -1; // gg !
22    }
23    return dist[e];
24 }
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
6     using T = tuple<int,int>;
7     priority_queue<T, vector<T>, greater<T>>
        pq;
8
9     dist[s] = 0;
10    pq.emplace(0, s); // (w, e) 讓 pq 優先用
        w 來比較
11
12    while (!pq.empty()) {
13        tie(std::ignore, s) = pq.top();
14        pq.pop();
15
16        if (used[s]) continue;

```

```

17        used[s] = true; // 每一個點都只看一
        次
18
19        for (auto [e, w] : V[s]) {
20            if (dist[e] > dist[s] + w) {
21                dist[e] = dist[s] + w;
22                pq.emplace(dist[e], e);
23            }
24        }
25    }
26
27    return dist[e];
28 }

```

## 5.4 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][j])
7                     d[i][j] = d[i][k] + d[k][j];
8 }

```

## 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){
19             low[u] = dfn[u] = ++dfn_cnt, vis[u] =
20                 1;
21             st.push(u);
22             for(auto v: g[u]){
23                 if(!dfn[v]){
24                     dfs(v, dis+1);
25                     low[u] = min(low[u], low[v]);
26                 }
27                 else if(vis[v]){
28                     low[u] = min(low[u], dfn[v]);
29                 }
30             }
31             if(low[u] == dfn[u]){
32                 ++cnt;
33                 while(vis[u]){
34                     auto v = st.top();

```

```

31         st.pop();
32         vis[v] = 0;
33         scc[v] = cnt;
34         sz[cnt]++;
35     }
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);
13     while (Q.size()) {
14         int u = Q.front();
15         Q.pop();
16         in_queue[u] = false;
17         for (auto [v, cost] : G[u])
18             if (d[v] > d[u] + cost) {
19                 if (++cnt[u] >= n) return false; // 存在負環
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: black, 1: white
3 /* color the connected component where u is */
4 /* parameter col: the color u should be colored */
5 bool coloring(int u, int col) {
6     if(color[u] != -1) {
7         if(color[u] != col) return false;
8         return true;
9     }
10    color[u] = col;
11    for(int v : G[u])
12        if(!coloring(v, col ^ 1))
13            return false;
14    return true;
15 }
16 //check if a graph is a bipartite graph
17 bool checkBipartiteG(int n) {
18     for(int i = 1; i <= n; i++)
19         color[i] = -1;
20     for(int i = 1; i <= n; i++)
21         if(color[i] == -1 && !coloring(i, 0))
22             return false;
23     return true;
24 }

```

## 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 {
9         Change = 0;
10        for(size_t u = 0; u < N; ++u) {
11            vector<size_t> E;
12            for(size_t v = 0; v < N && E.size() < 3; ++v)
13                if(G[u][v] && u != v) E.emplace_back(v);
14            if(E.size() == 1 || E.size() == 2) {
15                ++Change;
16                for(auto v : E) G[u][v] = G[v][u] = false;
17            }
18        }
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    } while(Change);
24 }

```

```

22 while(Change);
23 return G;
24 }
25 //計算Degree
26 //O(n^2)
27 vector<size_t> getDegree(const AdjacencyMatrixTy &G) {
28     size_t N = G.size();
29     vector<size_t> Degree(N);
30     for(size_t u = 0; u < N; ++u)
31         for(size_t v = u + 1; v < N; ++v) {
32             if(!G[u][v]) continue;
33             ++Degree[u], ++Degree[v];
34         }
35     return Degree;
36 }
37 //判斷是否為K5 or K33
38 //O(n)
39 bool is_K5_or_K33(const vector<size_t> &Degree) {
40     unordered_map<size_t, size_t> Num;
41     for(auto Val : Degree) ++Num[Val];
42     size_t N = Degree.size();
43     bool isK5 = Num[4] == 5 && Num[4] + Num[0] == N;
44     bool isK33 = Num[3] == 6 && Num[3] + Num[0] == N;
45     return isK5 || isK33;
46 }

```

## 5.10 判斷環

```

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

```

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

```

## 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 }

```

```

15 }
16 static Set setIntersection(const Set &A,
17     const Set &B){
18     Set C(min(A.size(), B.size()));
19     auto it = set_intersection(A.begin(),A.
20        end(),B.begin(),B.end(),C.begin());
21     C.erase(it, C.end());
22     return C;
23 }
24 static Set setDifference(const Set &A,
25     const Set &B){
26     Set C(min(A.size(), B.size()));
27     auto it = set_difference(A.begin(),A.end
28        (),B.begin(),B.end(),C.begin());
29     C.erase(it, C.end());
30     return C;
31 }
32 void BronKerbosch1(Set R, Set P, Set X){
33     if(P.empty()&&X.empty()){
34         // R form an maximal clique
35         return;
36     }
37     for(auto v: P){
38         BronKerbosch1(setUnion(R,{v}),
39            setIntersection(P,G[v]),
40            setIntersection(X,G[v]));
41         P = setDifference(P,{v});
42         X = setUnion(X,{v});
43     }
44 }
45 void init(int _n){
46     G.clear();
47     G.resize((n = _n) + 1);
48 }
49 void addEdge(int u, int v){
50     G[u].emplace_back(v);
51     G[v].emplace_back(u);
52 }
53 void solve(int n){
54     Set P;
55     for(int i=1; i<=n; ++i){
56         sort(G[i].begin(), G[i].end());
57         G[i].erase(unique(G[i].begin(), G[i].end()),
58            G[i].end());
59         P.emplace_back(i);
60     }
61     BronKerbosch1({}, P, {});
62 }
63 }
64 }

```

*//判斷圖G是否能3塗色：*  
*//枚舉圖G的極大獨立集I (極大獨立集 = 補圖極大團)*  
*//若存在I使得G-I形成二分圖，則G可以三塗色*  
*//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        for(int i = 0; i < n; ++i) {
21            if(id[i] == -1) dfs(i, -1);
22        }
23        return bridges;
24    };
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            if(!dfn[v]){
26                build(g,v,u);
27                child++;
28                if(dfn[u]<=low[v]){
29                    is_ap[u] = 1;
30                    bcc_id[u] = ++bcc_cnt;
31                    bcc[bcc_cnt].push_back(u);
32                    while(vis[v]){
33                        bcc_id[st.top()] = bcc_cnt;
34                        bcc[bcc_cnt].push_back(st.top());
35                        bcc_id[st.top()] = 0;
36                        st.pop();
37                    }
38                }
39            }
40        }
41    }
42 }

```

```

36         low[u] = min(low[u],low[v]);
37     }
38     low[u] = min(low[u],dfn[v]);
39 }
40 if(p==-1 and child<2)is_ap[u] = 0;
41 if(is_ap[u])ap.push_back(u);
42 }
43 }

```

## 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             if(k>=MAXPRIME) break;
35             iscom[k] = prime[j];
36             if(i%prime[j]==0) {
37                 mu[k] = 0;
38                 phi[k] = phi[i] * prime[j];
39                 break;
40             } else {
41                 mu[k] = -mu[i];
42                 phi[k] = phi[i] * (prime[j]-1);
43             }
44         }
45     }
46 }

```

```

45 bool g_test(const LL &g, const LL &p, const
    vector<LL> &v) {
46     for(int i=0; i<v.size(); ++i)
47         if(modexp(g, (p-1)/v[i], p) == 1)
48             return false;
49     return true;
50 }
51 LL primitive_root(const LL &p) {
52     if(p==2) return 1;
53     vector<LL> v;
54     Factor(p-1, v);
55     v.erase(unique(v.begin(), v.end(), v.end
        ());
56     for(LL g=2; g<p; ++g)
57         if(g_test(g, p, v))
58             return g;
59     puts("primitive_root NOT FOUND");
60     return -1;
61 }
62 int Legendre(const LL &a, const LL &p) {
63     return modexp(a%p, (p-1)/2, p);
64 }
65 LL inv(const LL &a, const LL &n) {
66     LL d, x, y;
67     gcd(a, n, d, x, y);
68     return d==1 ? (x+n)%n : -1;
69 }
70 int inv[maxN];
71 LL invtable(int n, LL P){
72     inv[1]=1;
73     for(int i=2; i<n; ++i)
74         inv[i]=(P-(P/i))*inv[P%i]%P;
75 }
76
77 LL log_mod(const LL &a, const LL &b, const
    LL &p) {
78     // a ^ x = b ( mod p )
79     int m=sqrt(p+.5), e=1;
80     LL v=inv(modexp(a, m, p), p);
81     map<LL, int> x;
82     x[1]=0;
83     for(int i=1; i<m; ++i) {
84         e = LLMul(e, a, p);
85         if(!x.count(e)) x[e] = i;
86     }
87     for(int i=0; i<m; ++i) {
88         if(x.count(b)) return i*m + x[b];
89         b = LLMul(b, v, p);
90     }
91     return -1;
92 }
93
94 LL Tonelli_Shanks(const LL &n, const LL &p)
    {
95     // x^2 = n ( mod p )
96     if(n==0) return 0;
97     if(Legendre(n, p)!=1) while(1) { puts("SQRT
        ROOT does not exist");
98     int S = 0;
99     LL Q = p-1;
100     while( !(Q&1) ) { Q>>=1; ++S; }
101     if(S==1) return modexp(n%p, (p+1)/4, p);
102     LL z = 2;
103     for(; Legendre(z, p)!=-1; ++z)
104         LL c = modexp(z, Q, p);

```

```

105     LL R = modexp(n%p, (Q+1)/2, p), t = modexp(n
        %p, Q, p);
106     int M = S;
107     while(1) {
108         if(t==1) return R;
109         LL b = modexp(c, 1LL<<(M-i-1), p);
110         R = LLMul(R, b, p);
111         t = LLMul( LLMul(b, b, p), t, p);
112         c = LLMul(b, b, p);
113         M = i;
114     }
115     return -1;
116 }
117
118 template<typename T>
119 T Euler(T n){
120     T ans=n;
121     for(T i=2; i*i<=n; ++i){
122         if(n%i==0){
123             ans=ans/i*(i-1);
124             while(n%i==0)n/=i;
125         }
126     }
127     if(n>1)ans=ans/n*(n-1);
128     return ans;
129 }
130
131 //Chinese_remainder_theorem
132 template<typename T>
133 T pow_mod(T n, T k, T m){
134     T ans=1;
135     for(n=(n>m?n%m:n); k>=>1){
136         if(k&1)ans=ans*n%m;
137         n=n*n%m;
138     }
139     return ans;
140 }
141
142 template<typename T>
143 T crt(vector<T> &m, vector<T> &a){
144     T M=1, tM, ans=0;
145     for(int i=0; i<(int)m.size(); ++i) M*=m[i];
146     for(int i=0; i<(int)a.size(); ++i){
147         tM=M/m[i];
148         ans=(ans+(a[i]*tM%M)*pow_mod(tM, Euler(m
            [i])-1, m[i])%M)%M;
149     }
150     //如果m[i]是質數 · Euler(m[i])-1=m[i]-2 ·
        就不用算Euler了*/
151     return ans;

```

## 6.2 Bit Set

```

1 void sub_set(int S){
2     int sub=S;
3     do{
4         //對某集合的子集合的處理
5         sub=(sub-1)&S;
6     }while(sub!=S);
7 }
8 void k_sub_set(int k, int n){
9     int comb=(1<<k)-1, S=1<<n;

```

```

10 while(comb<S){
11     //對大小為k的子集合的處理
12     int x=comb&~comb, y=comb+x;
13     comb=((comb&~y)/x>>1)|y;
14 }
15 }

```

## 6.3 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
5 void FFT(vector<cd> &a, bool inv) {
6     int n = (int)a.size();
7     for(int i = 1, j = 0; i < n; ++i) {
8         int bit = n >> 1;
9         for(; j & bit; bit >>= 1) {
10             j ^= bit;
11         }
12         j ^= bit;
13         if(i < j) {
14             swap(a[i], a[j]);
15         }
16     }
17     for(int len = 2; len <= n; len <= 1) {
18         const double ang = 2 * PI / len * (inv ?
            -1 : +1);
19         cd rot(cos(ang), sin(ang));
20         for(int i = 0; i < n; i += len) {
21             cd w(1);

```

```

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

```

## 6.6 FWT

```

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

```



```

18   for(int j=0; j<f.size(); j+=2<<i)
19       for(int k=0; k<(1<<i); ++k){
20           int u=f[j+k], v=f[j+k+(1<<i)];
21           f[j+k+(1<<i)] = u-v, f[j+k] = u+v;
22       }
23   if(inverse) for(auto &a:f) a/=f.size();
24   return f;
25 }

```

## 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[i][k];
18         }
19         bool ok = true;
20         REP(i, n) {
21             if(abs(a[i][i]) <= EPS) {
22                 if(abs(a[i][n]) > EPS) return -1;
23                 ok = false;
24             }
25         }
26         return ok;
27     }

```

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

```

## 6.9 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
14             (-1LL,0LL);
15         lastb = LLmul((lastb-b[i])/d,x,(lastm/d)
16             )*m[i];
17         lastm = (lastm/d)*m[i];
18         lastb = (lastb+b[i])%lastm;
19     }
20     return make_pair(lastb<0?lastb+lastm:lastb
21         ,lastm);

```

## 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),
8          true);
9      phi.resize(n); mobius.resize(n);
10     phi[1] = mobius[1] = 1;
11     least[0] = 0, least[1] = 1;
12     for(int i = 2; i < n; ++i) {
13         if(is_prime[i]) {
14             primes.push_back(i);
15             phi[i] = i - 1;
16             mobius[i] = -1;
17             least[i] = i;
18         }
19         for(auto j : primes) {
20             if(i * j >= n) break;
21             is_prime[i * j] = false;
22             least[i * j] = j;
23             if(i % j == 0) {
24                 mobius[i * j] = 0;
25                 phi[i * j] = phi[i] * j;
26                 break;
27             } else {
28                 mobius[i * j] = mobius[i] * mobius[j];
29                 phi[i * j] = phi[i] * phi[j];
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)*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         det = det/lazy[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

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

- Catalan numbers

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

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

- Eulerian numbers

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

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

## 6.15 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(7,p) && mr(43,p);
35 }
36
37 // O(1) 快速乘(防ll overflow)
38 ll ksc(ull x, ull y, ll p) {
39     return (x*y-(ull)((ldb)x/p*y)*p+p)%p;
40 }
41
42 //求n任一真因数(需保證n非質數) O(n^1/4)
43 ll pollar_rho(ll n) {
44     ll x,y,z,c,g,i,j;
45     while(1) {
46         x = y = rand()%n;
47         z = 1;
48         c = rand()%n;
49         i = 0, j = 1;
50         while(++i) {
51             x = (ksc(x,x,n) + c)%n;
52             z = ksc(z,abs(y-x),n);
53             if(x == y || !z) break;
54             if(!(i%127) || i == j) {
55                 g = __gcd(z,n);
56                 if(g > 1) return g;
57                 if(i == j) y = x, j <= 1;
58             }
59         }
60     }
61 }
62
63 void factorization(ll n) {
64     while(!is_prime(n)) {
65         ll f = pollar_rho(n);
66         while(!is_prime(f)) {
67             f = pollar_rho(f);
68         }
69         ll cou = 0;
70         while(n%f == 0) n /= f, cou++;
71         fac.push_back({f,cou});
72     }
73     if(n != 1) fac.push_back({n,1});
74 }
75
76 void get_factors(ll now, ll cou) {
77     if(now >= fac.size()) {
78         factor.push_back(cou);
79         return;
80     }
81     get_factors(now+1,cou);
```

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

## 6.16 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  $T_{n,k} = kn^{\overline{n-k-1}}$ .

- 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  $\sum_{i=1}^k a_i \leq \sum_{i=1}^n \min(b_i, k)$  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  $\sum_{i=1}^k a_i \leq \sum_{i=1}^n \min(b_i, k-1) + \sum_{i=k+1}^n \min(b_i, k)$  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,  $\theta$ : arcsin( $a/r$ ).
- Volume  $= \pi h^2(3r - h)/3 = \pi h(3a^2 + h^2)/6 = \pi r^3(2 + \cos \theta)(1 - \cos \theta)^2/3$ .
- Area  $= 2\pi r h = \pi(a^2 + h^2) = 2\pi r^2(1 - \cos \theta)$ .

## 6.17 找實根

```

1 //  $an^x + \dots + a_1x + a_0 = 0$ ;
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;
16     if( !(sign_lo = sign(get(coef,lo))) )
17         return lo;
18     if( !(sign_hi = sign(get(coef,hi))) )
19         return hi;
20     if(sign_lo * sign_hi > 0) return INF;
21     for(int stp = 0; stp < 100 && hi - lo >
22         eps; ++stp){
23         double m = (lo+hi)/2.0;
24         int sign_mid = sign(get(coef,m));
25         if(!sign_mid) return m;
26         if(sign_lo*sign_mid < 0) hi = m;
27         else lo = m;
28     }
29     return (lo+hi)/2.0;
30 }
31
32 vector<double> cal(vector<double>coef, int n
33 ){
34     vector<double>res;
35     if(n == 1){
36         if(sign(coef[1])) res.pb(-coef[0]/coef
37             [1]);
38         return res;
39     }
40     vector<double>dcoef(n);
41     for(int i = 0; i < n; ++i) dcoef[i] = coef
42         [i+1]*(i+1);
43     vector<double>droot = cal(dcoef, n-1);
44     droot.insert(droot.begin(), -INF);
45     droot.pb(INF);
46     for(int i = 0; i+1 < droot.size(); ++i){
47         double tmp = find(coef, n, droot[i],
48             droot[i+1]);
49         if(tmp < INF) res.pb(tmp);
50     }
51     return res;
52 }
53
54 int main () {
55     vector<double>ve;
56     vector<double>ans = cal(ve, n);
57     // 視情況把答案 +eps · 避免 -0
58 }

```

## 6.18 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         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     }
42 }

```

## 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    }

```

```

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));
32     if(a.ql/B!=b.ql/B)return a.ql<b.ql;
33     if((a.ql/B)&1)return a.qr>b.qr;
34     return a.qr<b.qr;
35 });
36 for(auto [ql,qr,id]:q){
37     while(l>ql)add(--l);
38     while(r<qr)add(++r);
39     while(l<ql)sub(l--);
40     while(r>qr)sub(r--);
41     ans[id] = {mx,cntcnt[mx]};
42 }
43 }
44 }

```

### 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;

```

```

29 while(t--){
30     {
31         cin >> ord >> l >> r;
32         l = (l+ans-1)%n+1 -1;
33         r = (r+ans-1)%n+1 -1;
34         if(l > r) swap(l,r);
35         id = l/block_siz; l %= block_siz;
36         id2 = r/block_siz; r %= block_siz;
37         if(ord == 1)
38         {
39             if(id == id2)
40             {
41                 pre = q[id][r];
42                 for(i=r;i>l;i--)
43                 {
44                     q[id][i] = q[id][i-1];
45                 }
46                 q[id][l] = pre;
47             }
48             else
49             {
50                 pre = q[id].back();
51                 cnt[id][pre]--;
52                 q[id].pop_back();
53                 for(i=id+1;i<id2;i++)
54                 {
55                     q[i].push_front(pre);
56                     cnt[i][pre]++;
57                     pre = q[i].back();
58                     cnt[i][pre]--;
59                     q[i].pop_back();
60                 }
61                 q[id2].push_front(pre);
62                 cnt[id2][pre]++;
63                 pre = q[id2][r+1];
64                 cnt[id2][pre]--;
65                 q[id2].erase(q[id2].begin()+
66                     r+1);
67                 q[id].insert(q[id].begin()+l
68                     , pre);
69                 cnt[id][pre]++;
70             }
71             //print_all(n);
72         }
73         else
74         {
75             // query m cnt
76             cin >> m;
77             m = (m+ans-1)%n+1;
78             ans = 0;
79             if(id == id2)
80             {
81                 for(i=l;i<=r;i++) ans += (q[
82                     id][i] == m);
83             }
84             else
85             {
86                 for(i=l;i<block_siz;i++) ans
87                     += (q[id][i] == m);
88                 for(i=id+1;i<id2;i++) ans +=
89                     (q[id2][i] == m);
90                 for(i=id+1;i<id2;i++) ans +=
91                     cnt[i][m];
92             }
93             cout << ans << endl;
94         }
95     }
96 }

```

```

89     }
90 }
91     return 0;
92 }

```

## 7.3 莫隊

```

1 void remove(idx); // TODO: remove value at
  idx from data structure
2 void add(idx); // TODO: add value at idx
  from data structure
3 int get_answer(); // TODO: extract the
  current answer of the data structure
4
5 int block_size;
6
7 struct Query {
8     int l, r, idx;
9     bool operator<(Query other) const
10    {
11        return make_pair(l / block_size, r)
12            <
13            make_pair(other.l /
14                block_size, other.r);
15    }
16 };
17
18 vector<int> mo_s_algorithm(vector<Query>
19     queries) {
20     vector<int> answers(queries.size());
21     sort(queries.begin(), queries.end());
22
23     // TODO: initialize data structure
24
25     int cur_l = 0;
26     int cur_r = -1;
27     // invariant: data structure will always
28     // reflect the range [cur_l, cur_r]
29     for (Query q : queries) {
30         while (cur_l > q.l) {
31             cur_l--;
32             add(cur_l);
33         }
34         while (cur_r < q.r) {
35             cur_r++;
36             add(cur_r);
37         }
38         while (cur_l < q.l) {
39             remove(cur_l);
40             cur_l++;
41         }
42         while (cur_r > q.r) {
43             remove(cur_r);
44             cur_r--;
45         }
46         answers[q.idx] = get_answer();
47     }
48     return answers;
49 }

```

## 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);
12            sz[u]+=sz[v];
13        }
14    };
15    function<int(int,int)>dfs2 = [&](int u,int
16        p){
17        for(auto v:g[u]){
18            if(v==p)continue;
19            if(vis[v])continue;
20            if(sz[v]*2<n)continue;
21            return dfs2(v,u);
22        }
23    };
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     //calcuete 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     while(!tmp.empty()){
52         //roll_back(tmp.back())
53         tmp.pop_back();
54     }
55     for(auto v:g[center]){
56         if(vis[v])continue;
57     }
58 }

```

```

56     ans+=centroid_decomposition(v,sz[v]);
57 }
58     return ans;
59 }

```

### 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;
9     max_son[u]=-1;
10    for(auto v:G[u]){
11        if(v==pa[u])continue;
12        pa[v]=u;
13        dep[v]=dep[u]+1;
14        find_max_son(v);
15        if(max_son[u]==-1||siz[v]>siz[max_son[u]
16            ]){max_son[u]=v;
17            siz[u]+=siz[v];
18        }
19    }
20 }
21 void build_link(int u,int top){
22     link[u]=++cnt;
23     link_top[u]=top;
24     if(max_son[u]==-1)return;
25     build_link(max_son[u],top);
26     for(auto v:G[u]){
27         if(v==max_son[u]||v==pa[u])continue;
28         build_link(v,v);
29     }
30 }
31 int find_lca(int a,int b){
32     //求LCA·可以在過程中對區間進行處理
33     int ta=link_top[a],tb=link_top[b];
34     while(ta!=tb){
35         if(dep[ta]<dep[tb]){
36             swap(ta,tb);
37             swap(a,b);
38         }
39         //這裡可以對a所在的鏈做區間處理
40         //區間為(Link[ta],Link[a])
41         ta=link_top[a=pa[ta]];
42     }
43     //最後a,b會在同一條鏈·若a!=b還要在進行一
44     //次區間處理
45     return dep[a]<dep[b]?a:b;
46 }

```

### 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;
23        mxson[u] = 0;
24        for(auto v:g[u]){
25            if(v==p)continue;
26            dfs(v,u);
27            sz[u]+=sz[v];
28            if(sz[v]>sz[mxson[u]])mxson[u] = v;
29        }
30    }
31    void dfs2(int u,int top){
32        static int idn = 0;
33        topf[u] = top;
34        id[u] = ++idn;
35        if(mxson[u])dfs2(mxson[u],top);
36        for(auto v:g[u]){
37            if(v!=father[u] and v!=mxson[u]){
38                dfs2(v,v);
39            }
40        }
41    }
42    void build(int root){
43        dfs(root,0);
44        dfs2(root,root);
45    }
46    vector<pair<int, int>> path(int u,int v){
47        vector<pair<int, int>>ans;
48        while(topf[u]!=topf[v]){
49            if(dep[topf[u]]<dep[topf[v]])swap(u,v);
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    }
57 };

```

### 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];

```

```

6 void dfs(int x,int p=0){//dfs(root);
7   pa[0][x]=p;
8   for(int i=0;i<=MLG;++i)
9     pa[i+1][x]=pa[i][pa[i][x]];
10  for(auto &i:G[x]){
11    if(i==p)continue;
12    dep[i]=dep[x]+1;
13    dfs(i,x);
14  }
15 }
16 inline int jump(int x,int d){
17   for(int i=0;i<=MLG;++i)
18     if((d>>i)&1) x=pa[i][x];
19   return x;
20 }
21 inline int find_lca(int a,int b){
22   if(dep[a]>dep[b])swap(a,b);
23   b=jump(b,dep[b]-dep[a]);
24   if(a==b)return a;
25   for(int i=MLG;i>0;--i){
26     if(pa[i][a]!=pa[i][b]){
27       a=pa[i][a];
28       b=pa[i][b];
29     }
30   }
31   return pa[0][a];
32 }
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 }
51 inline int find_lca(int a,int b){
52   if(in[a]>in[b])swap(a,b);
53   return RMQ(in[a],in[b]);
54 }
55 }

```

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

```

```

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   //假設鏈上點權的總和，sum是子樹的權重和，
113   data是節點的權重
114   access(u);
115   int lca=access(v);
116   splay(u);
117   if(u==lca){
118     //return nd[lca].data+nd[nd[lca].ch[1]].
119     sum
120   }else{
121     //return nd[lca].data+nd[nd[lca].ch[1]].
122     sum+nd[u].sum
123   }
124 }
125 struct EDGE{
126   int a,b,w;
127   int e[10005];
128   int n;
129   vector<pair<int,int>> G[10005];
130   //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
130   被存在哪個點裡面的陣列
131 void bfs(int root){
132   //在建構的時候把每個點都設成一個splay tree
133   queue<int > q;
134   for(int i=1;i<=n;++i)pa[i]=0;
135   q.push(root);
136   while(q.size()){
137     int u=q.front();
138     q.pop();
139     for(auto P:G[u]){
140       int v=P.first;
141       if(v!=pa[u]){
142         pa[v]=u;
143         nd[v].pa=u;
144         nd[v].data=e[P.second].w;
145         edge_node[P.second]=v;
146         up(v);
147         q.push(v);
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  dfs(1); // 隨便選一個點
13  int a = max_element(level.begin(), level.end
14  ()) - 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
17  ()) - level.begin();
18  cout << level[b] << endl;
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  void dfs(int u, int parent = -1) {
23      D1[u] = D2[u] = 0;
24      for (int v : Tree[u]) {
25          if (v == parent) continue;
26          dfs(v, u);
27          int dis = D1[v] + 1;
28          if (dis > D1[u]) {
29              D2[u] = D1[u];
30              D1[u] = dis;
31          } else
32              D2[u] = max(D2[u], dis);
33      }
34      ans = max(ans, D1[u] + D2[u]);
35  }
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 //進去出來都紀錄
16 vector<int> Arr;
17 void dfs(int u) {
18     Arr.push_back(u);
19     for (auto v : Tree[u]) {
20         if (v == parent[u])
21             continue;
22         parent[v] = u;
23         dfs(v);
24     }
25     Arr.push_back(u);
26 }

```

```

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

```

```

91 }

```

## 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].efl;
46                     q.push_back(t);
47                     ++qe;
48                 }
49             }
50         }
51         /*DP出每個前綴在字串s出現的次數並傳回所有
52             字串被s匹配成功的次數O(N+M)*/

```

```

52 int match_0(const char *s){
53     int ans=0, id, p=0, i;
54     for(i=0; s[i]; ++i){
55         id=s[i]-L;
56         while(!S[p].next[id]&&p) p=S[p].fail;
57         if(!S[p].next[id]) continue;
58         p=S[p].next[id];
59         ++S[p].cnt_dp; /*匹配成功則它所有後綴都
60             可以被匹配(DP計算)*/
61     }
62     for(i=qe-1; i>=0; --i){
63         ans+=S[q[i]].cnt_dp*S[q[i]].ed;
64         if(!S[q[i]].fail) S[q[i]].fail=-1;
65         cnt_dp+=S[q[i]].cnt_dp;
66     }
67     return ans;
68 }
69 /*多串匹配走efl邊並傳回所有字串被s匹配成功
70     的次數O(N*M^1.5)*/
71 int match_1(const char *s) const {
72     int ans=0, id, p=0, t;
73     for(int i=0; s[i]; ++i){
74         id=s[i]-L;
75         while(!S[p].next[id]&&p) p=S[p].fail;
76         if(!S[p].next[id]) continue;
77         p=S[p].next[id];
78         if(S[p].ed) ans+=S[p].ed;
79         for(t=S[p].efl; ~t; t=S[t].efl){
80             ans+=S[t].ed; /*因為都走efl邊所以保證
81                 匹配成功*/
82         }
83     }
84     return ans;
85 }
86 /*枚舉(s的子字串nA)的所有相異字串各恰一次
87     並傳回次數O(N*M^(1/3))*/
88 int match_2(const char *s){
89     int ans=0, id, p=0, t;
90     ++vt;
91     /*把戳記vt+=1，只要vt沒溢位，所有S[p].
92         vis==vt就會變成false
93     這種利用vt的方法可以O(1)歸零vis陣列*/
94     for(int i=0; s[i]; ++i){
95         id=s[i]-L;
96         while(!S[p].next[id]&&p) p=S[p].fail;
97         if(!S[p].next[id]) continue;
98         p=S[p].next[id];
99         if(S[p].ed&&S[p].vis!=vt){
100             S[p].vis=vt;
101             ans+=S[p].ed;
102         }
103         for(t=S[p].efl; ~t&&S[t].vis!=vt; t=S[t].efl){
104             S[t].vis=vt;
105             ans+=S[t].ed; /*因為都走efl邊所以保證
106                 匹配成功*/
107         }
108     }
109     return ans;
110 }
111 /*把AC自動機變成真的自動機*/
112 void evolution(){
113     for(qs=1; qs!=qe;){

```

```

107     int p=q[qs++];
108     for(int i=0;i<R-L;++i)
109         if(S[p].next[i]==0)S[p].next[i]=S[S[
110             p].fail].next[i];
111     }
112 };

```

## 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 }
12 vector<int> match_index;
13 /*以字串B匹配字串A · 傳回匹配成功的數量(用B的
14    fail)*/
15 int kmp_match(char *A,int lenA,char *B,int
16    lenB,int *fail){
17     int id=-1,ans=0;
18     for(int i=0;i<lenA;++i){
19         while(~id&&B[id+1]!=A[i])id=fail[id];
20         if(B[id+1]==A[i])++id;
21         if(id==lenB-1){/*匹配成功*/
22             ++ans, id=fail[id];
23             match_index.emplace_back(i + 1 -lenB);
24         }
25     }
26     return ans;
27 }

```

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

```

```

10    for(int i = 1; i < SZ(s); i++) ans[i] = (
11        ans[i - 1] * M + s[i]) % mod;
12    return ans;

```

## 9.7 suffix array lcp

```

1 #define radix_sort(x,y){\
2     for(i=0;i<A;++i)c[i]=0;\
3     for(i=0;i<n;++i)c[x[y[i]]]++;\
4     for(i=1;i<A;++i)c[i]+=c[i-1];\
5     for(i=n-1;~i;--i)sa[--c[x[y[i]]]]=y[i];\
6 }
7 #define AC(r,a,b)\
8     r[a]!=r[b]||a+k>=n||r[a+k]!=r[b+k]
9 void suffix_array(const char *s,int n,int *
10    sa,int *rank,int *tmp,int *c){
11     int A='z'+1,i,k,id=0;
12     for(i=0;i<n;++i)rank[tmp[i]]=s[i];
13     radix_sort(rank,tmp);
14     for(k=1;id<n-1;k<=1){
15         for(id=0,i=n-k;i<n;++i)tmp[id++] = i;
16         for(i=0;i<n;++i)
17             if(sa[i]>=k)tmp[id++] = sa[i]-k;
18         radix_sort(rank,tmp);
19         swap(rank,tmp);
20         for(rank[sa[0]]=id=0,i=1;i<n;++i)
21             rank[sa[i]]=id+=AC(tmp,sa[i-1],sa[i]);
22         A=id+1;
23     }
24 }
25 //h:高度數組 sa:後綴數組 rank:排名
26 void suffix_array_lcp(const char *s,int len,
27    int *h,int *sa,int *rank){
28     for(int i=0;i<len;++i)rank[sa[i]]=i;
29     for(int i=0,k=0;i<len;++i){
30         if(rank[i]==0)continue;
31         if(k--<0)
32             while(s[i+k]==s[sa[rank[i]-1]+k])++k;
33         h[rank[i]]=k;
34     }
35 }

```

## 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    inline int next(int p, int v) {
15        return nodes[p].go[v] != -1 ? nodes[p].
16            go[v] : nodes[p].go[v] = newNode();
17    }
18    inline void insert(const vector<int>& a,
19        int p = 0) {
20        for(int v : a) {
21            p = next(p, v);
22        }
23    }
24    inline void clear() {
25        nodes.clear();
26        newNode();
27    }
28    inline int longest_common_prefix(const
29        vector<int>& a, int p = 0) const {
30        int ans = 0;
31        for(int v : a) {
32            if(nodes[p].go[v] != -1) {
33                ans += 1;
34                p = nodes[p].go[v];
35            } else {
36                break;
37            }
38        }
39        return ans;
40    }
41 private:
42    vector<Node> nodes;
43    inline int newNode() {
44        nodes.emplace_back();
45        return (int) nodes.size() - 1;
46    }
47 }
48 }
49 }
50 }

```

## 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\
28                n",t,et-st);
29            return 0;
30        }
31        else {
32            printf("\033[0;32mAccepted\033[0
33                m on test #%d, Time %.0lfms\
34                n", t, et - st);
35        }
36    }
```

### 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,sse3,sse4,
3     popcnt,abm,mmx,avx,tune=native")
```

```
3 #pragma GCC optimize("inline")
4 #pragma GCC optimize("-fgcse")
5 #pragma GCC optimize("-fgcse-lm")
6 #pragma GCC optimize("-fipa-sra")
7 #pragma GCC optimize("-ftree-pre")
8 #pragma GCC optimize("-ftree-vrp")
9 #pragma GCC optimize("-fpeephole2")
10 #pragma GCC optimize("-ffast-math")
11 #pragma GCC optimize("-fsched-spec")
12 #pragma GCC optimize("-falign-jumps")
13 #pragma GCC optimize("-falign-loops")
14 #pragma GCC optimize("-falign-labels")
15 #pragma GCC optimize("-fdevirtualize")
16 #pragma GCC optimize("-fcaller-saves")
17 #pragma GCC optimize("-fcrossjumping")
18 #pragma GCC optimize("-fthread-jumps")
19 #pragma GCC optimize("-funroll-loops")
20 #pragma GCC optimize("-fwhole-program")
21 #pragma GCC optimize("-freorder-blocks")
22 #pragma GCC optimize("-fschedule-insns")
23 #pragma GCC optimize("inline-functions")
24 #pragma GCC optimize("-ftree-tail-merge")
25 #pragma GCC optimize("-fschedule-insns2")
26 #pragma GCC optimize("-fststrict-aliasing")
27 #pragma GCC optimize("-fststrict-overflow")
28 #pragma GCC optimize("-falign-functions")
29 #pragma GCC optimize("-fcse-skip-blocks")
30 #pragma GCC optimize("-fcse-follow-jumps")
31 #pragma GCC optimize("-fsched-interblock")
32 #pragma GCC optimize("-fpartial-inlining")
33 #pragma GCC optimize("no-stack-protector")
34 #pragma GCC optimize("-freorder-functions")
35 #pragma GCC optimize("-findirect-inlining")
36 #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-
40     functions")
41 #pragma GCC optimize("-ftree-switch-
42     conversion")
43 #pragma GCC optimize("-foptimize-sibling-
44     calls")
45 #pragma GCC optimize("-fexpensive-
46     optimizations")
47 #pragma GCC optimize("-funsafe-loop-
48     optimizations")
49 #pragma GCC optimize("inline-functions-
50     called-once")
51 #pragma GCC optimize("-fdelete-null-pointer-
52     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));
```

```
7     for(auto i:v)ans.push_back(lower_bound(
8         begin(tmp),end(tmp),i)-tmp.begin()+1);
9     return ans;
```

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

```

46 T find_by_order(int x){return st.
    find_by_order(x)->first;}
47 T lower_bound(T x){return st.lower_bound({
    x,0})->first;}
48 T upper_bound(T x){return st.upper_bound({
    x,(int)1e9+7})->first;}
49 T smaller_bound(T x){return (--st.
    lower_bound({x,0}))->first;}
50 };

```

## 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::
    sync_with_stdio(false);
6 #define aqua clock_t qua = clock();
7 #define aquaa cout << "Aqua says: " << (
    double)(clock()-qua)/CLOCKS_PER_SEC << "
    sec!\n";
8 #define random_set(m,n) random_device rd; \
9     mt19937 gen=mt19937(
    rd()); \
10     uniform_ll_distribution
    <ll> dis(m,n); \
11     auto rnd=bind(dis,
    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

<b>1 Computational Geometry</b>	<b>1</b>	3.7 Lazytag Segment Tree . . . .	5	5.10 判斷環 . . . . .	11	<b>8 Tree</b>	<b>17</b>
1.1 Geometry . . . . .	1	3.8 LiChaoST . . . . .	5	5.11 最大團 . . . . .	12	8.1 centroidDecomposition . . . .	17
1.2 MinCircleCover . . . . .	2	3.9 monotonic queue . . . . .	6	5.12 枚舉極大團 Bron-Kerbosch .	12	8.2 HeavyLight . . . . .	17
1.3 最近點對 . . . . .	3	3.10 monotonic stack . . . . .	6	5.13 橋連通分量 . . . . .	12	8.3 HLD . . . . .	17
<b>2 DP</b>	<b>3</b>	3.11 pbds . . . . .	6	5.14 雙連通分量 & 割點 . . . . .	12	8.4 LCA . . . . .	17
2.1 basic DP . . . . .	3	3.12 Persistent DSU . . . . .	6	<b>6 Math</b>	<b>12</b>	8.5 link cut tree . . . . .	18
2.2 DP on Graph . . . . .	3	3.13 Persistent Segment Tree . . .	6	6.1 Basic . . . . .	12	8.6 Tree centroid . . . . .	18
2.3 LineContainer . . . . .	3	3.14 Prim . . . . .	6	6.2 Bit Set . . . . .	13	8.7 Tree diameter . . . . .	18
2.4 單調隊列優化 . . . . .	4	3.15 SegmentTree . . . . .	6	6.3 ExtendGCD . . . . .	13	8.8 樹壓平 . . . . .	19
2.5 整體二分 . . . . .	4	3.16 sparse table . . . . .	7	6.4 FastPow . . . . .	13	<b>9 string</b>	<b>19</b>
2.6 斜率優化-動態凸包 . . . . .	4	3.17 TimingSegmentTree . . . . .	7	6.5 FFT . . . . .	13	9.1 AC 自動機 . . . . .	19
<b>3 Data Structure</b>	<b>4</b>	3.18 回滾並查集 . . . . .	7	6.6 FWT . . . . .	13	9.2 KMP . . . . .	20
3.1 2D BIT . . . . .	4	3.19 掃描線 + 線段樹 . . . . .	7	6.7 Gauss-Jordan . . . . .	14	9.3 manacher . . . . .	20
3.2 BinaryTrie . . . . .	4	3.20 陣列上 Treap . . . . .	8	6.8 InvGCD . . . . .	14	9.4 minimal string rotation . . . .	20
3.3 BIT . . . . .	5	<b>4 Flow</b>	<b>8</b>	6.9 LinearCongruence . . . . .	14	9.5 reverseBWT . . . . .	20
3.4 DSU . . . . .	5	4.1 dinic . . . . .	8	6.10 LinearSieve . . . . .	14	9.6 Rolling Hash . . . . .	20
3.5 Dynamic Segment Tree . . . .	5	4.2 Gomory Hu . . . . .	9	6.11 Lucas . . . . .	14	9.7 suffix array lcp . . . . .	20
3.6 Kruskal . . . . .	5	4.3 ISAP with cut . . . . .	9	6.12 Matrix . . . . .	14	9.8 Trie . . . . .	20
		4.4 MinCostMaxFlow . . . . .	9	6.13 Numbers . . . . .	14	9.9 Z . . . . .	20
		4.5 Property . . . . .	10	6.14 Pisano number . . . . .	15	<b>10 tools</b>	<b>21</b>
		<b>5 Graph</b>	<b>10</b>	6.15 Pollard-Rho . . . . .	15	10.1 bitset . . . . .	21
		5.1 2-SAT . . . . .	10	6.16 Theorem . . . . .	15	10.2 Bsearch . . . . .	21
		5.2 Bellman Ford . . . . .	10	6.17 找實根 . . . . .	16	10.3 Counting Sort . . . . .	21
		5.3 Dijkstra . . . . .	10	6.18 質因數分解 . . . . .	16	10.4 DuiPai . . . . .	21
		5.4 Dominator tree . . . . .	10	<b>7 Square root decomposition</b>	<b>16</b>	10.5 HashMap . . . . .	21
		5.5 Floyd Warshall . . . . .	11	7.1 MoAlgo . . . . .	16	10.6 pragma . . . . .	21
		5.6 SCC . . . . .	11	7.2 分塊 cf455D . . . . .	16	10.7 relabel . . . . .	21
		5.7 SPFA . . . . .	11	7.3 莫隊 . . . . .	17	10.8 Template . . . . .	21
		5.8 判斷二分圖 . . . . .	11			10.9 template bubble . . . . .	22
		5.9 判斷平面圖 . . . . .	11			10.10TenarySearch . . . . .	22



# 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
```