

# 1 Computational Geometry

## 1.1 最近點對

```

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

```

## 1.2 Geometry

```

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

```

```

29     if(A<=-PI/2)A+=PI*2;
30     return A;
31 }
32 };
33 template<typename T>
34 struct line{
35     line(){}
36     point<T> p1,p2;
37     T a,b,c;//ax+by+c=0
38     line(const point<T>&x,const point<T>&y):p1(
39         x),p2(y){}
40     void pton()const{//轉成一般式
41         a=p1.y-p2.y;
42         b=p2.x-p1.x;
43         c=-a*p1.x-b*p1.y;
44     }
45     T ori(const point<T> &p)const{//點和有向直
46         線的關係，>0左邊，=0在線上，<0右邊
47         return (p2-p1).cross(p-p1);
48     }
49     T btw(const point<T> &p)const{//點投影落在
50         線段上<=0
51         return (p1-p).dot(p2-p);
52     }
53     bool point_on_segment(const point<T>&p)
54         const{//點是否在線段上
55         return ori(p)==0&&btw(p)<=0;
56     }
57     T dis2(const point<T> &p,bool is_segment
58         =0)const{//點跟直線/線段的距離平方
59     point<T> v=p2-p1,v1=p-p1;
60     if(is_segment){
61         point<T> v2=p-p2;
62         if(v.dot(v1)<=0)return v1.abs2();
63         if(v.dot(v2)>=0)return v2.abs2();
64     }
65     T tmp=v.cross(v1);
66     return tmp*tmp/v.abs2();
67 }
68 T seg_dis2(const line<T> &l)const{//兩線段
69     距離平方
70     return min({dis2(l.p1,1),dis2(l.p2,1),l.
71         dis2(p1,1),l.dis2(p2,1)});
72 }
73 point<T> projection(const point<T> &p)
74     const{//點對直線的投影
75     point<T> n=(p2-p1).normal();
76     return p-n*(p-p1).dot(n)/n.abs2();
77 }
78 point<T> mirror(const point<T> &p)const{
79     //點對直線的鏡射，要先呼叫pton轉成一般式
80     point<T> R;
81     T d=a*a+b*b;
82     R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
83     R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
84     return R;
85 }
86 bool equal(const line &l)const{//直線相等
87     return ori(l.p1)==0&&ori(l.p2)==0;
88 }
89 bool parallel(const line &l)const{
90     return (p1-p2).cross(l.p1-l.p2)==0;
91 }
92 bool cross_seg(const line &l)const{

```

```

93     return (p2-p1).cross(l.p1-p1)*(p2-p1).
94         cross(l.p2-p1)<=0;//直線是否交線段
95 }
96 int line_intersect(const line &l)const{//
97     直線相交情況，-1無限多點，1交於一點，0
98     不相交
99     return parallel(l)?(ori(l.p1)==0?-1:0)
100        :1;
101 }
102 int seg_intersect(const line &l)const{
103     T c1=ori(l.p1), c2=ori(l.p2);
104     T c3=l.ori(p1), c4=l.ori(p2);
105     if(c1==0&&c2==0){//共線
106         bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
107         T a3=l.btw(p1),a4=l.btw(p2);
108         if(b1&&b2&&a3==0&&a4==0) return 2;
109         if(b1&&b2&&a3>0&&a4==0) return 3;
110         if(b1&&b2&&a3>0&&a4>0) return 0;
111         return -1;//無限交點
112     }else if(c1*c2<=0&&c3*c4<=0)return 1;
113     return 0;//不相交
114 }
115 point<T> line_intersection(const line &l)
116     const{//直線交點*/
117     point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
118     //if(a.cross(b)==0)return INF;
119     return p1+a*(s.cross(b)/a.cross(b));
120 }
121 point<T> seg_intersection(const line &l)
122     const{//線段交點
123     int res=seg_intersect(l);
124     if(res<=0) assert(0);
125     if(res==2) return p1;
126     if(res==3) return p2;
127     return line_intersection(l);
128 }
129 }
130 template<typename T>
131 struct polygon{
132     polygon(){}
133     vector<point<T>> p;//逆時針順序
134     T area()const{//面積
135         T ans=0;
136         for(int i=p.size()-1,j=0;j<(int)p.size()
137             ;i=j++){
138             ans+=p[i].cross(p[j]);
139         }
140         return ans/2;
141     }
142     point<T> center_of_mass()const{//重心
143         T cx=0,cy=0,w=0;
144         for(int i=p.size()-1,j=0;j<(int)p.size()
145             ;i=j++){
146             T a=p[i].cross(p[j]);
147             cx+=(p[i].x+p[j].x)*a;
148             cy+=(p[i].y+p[j].y)*a;
149             w+=a;
150         }
151         return point<T>(cx/3/w,cy/3/w);
152     }
153 }
154 char ahas(const point<T>& t)const{//點是否
155     在簡單多邊形內，是的話回傳1、在邊上回
156     傳-1、否則回傳0
157     bool c=0;

```

```

158     for(int i=0,j=p.size()-1;i<p.size();j=i
159         ++){
160         if(line<T>(p[i],p[j]).point_on_segment
161             (t))return -1;
162         else if((p[i].y>t.y)!=p[j].y>t.y)&&
163             t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j]
164                 .y-p[i].y)+p[i].x)
165             c=!c;
166         return c;
167     }
168     char point_in_convex(const point<T>&x)
169         const{
170         int l=1,r=(int)p.size()-2;
171         while(l<=r){//點是否在凸多邊形內，是的話
172             回傳1、在邊上回傳-1、否則回傳0
173             int mid=(l+r)/2;
174             T a1=(p[mid]-p[0]).cross(x-p[0]);
175             T a2=(p[mid+1]-p[0]).cross(x-p[0]);
176             if(a1>=0&&a2<=0){
177                 T res=(p[mid+1]-p[mid]).cross(x-p[
178                     mid]);
179                 return res>0?1:(res>0?-1:0);
180             }else if(a1<0)r=mid-1;
181             else l=mid+1;
182         }
183         return 0;
184     }
185     vector<T> getA()const{//凸包邊對x軸的夾角
186     vector<T>res;//一定是遞增的
187     for(size_t i=0;i<p.size();++i)
188         res.push_back((p[(i+1)%p.size()]-p[i])
189             .getA());
190     return res;
191 }
192 bool line_intersect(const vector<T>&A,
193     const line<T> &l)const{//O(LogN)
194     int f1=upper_bound(A.begin(),A.end(),(l.
195         p1-l.p2).getA())-A.begin();
196     int f2=upper_bound(A.begin(),A.end(),(l.
197         p2-l.p1).getA())-A.begin();
198     return l.cross_seg(line<T>(p[f1],p[f2]))
199         ;
200 }
201 polygon cut(const line<T> &l)const{//凸包
202     對直線切割，得到直線L左側的凸包
203     polygon ans;
204     for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
205         if(l.ori(p[i])>=0){
206             ans.p.push_back(p[i]);
207             if(l.ori(p[j])<0)
208                 ans.p.push_back(l.
209                     line_intersection(line<T>(p[i]
210                         ,p[j])));
211         }else if(l.ori(p[j])>0)
212             ans.p.push_back(l.line_intersection(
213                 line<T>(p[i],p[j])));
214     }
215     return ans;
216 }
217 static bool monotone_chain_cmp(const point
218     <T>& a,const point<T>& b){//凸包排序函
219     數
220     return (a.x<b.x)||((a.x==b.x&&a.y<b.y));
221 }

```

```

185 void monotone_chain(vector<point<T> > &s){
186     //凸包
187     sort(s.begin(),s.end(),
188         monotone_chain_cmp);
189     p.resize(s.size()+1);
190     int m=0;
191     for(size_t i=0;i<s.size();++i){
192         while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]
193             -p[m-2])<=0)--m;
194         p[m++]=s[i];
195     }
196     for(int i=s.size()-2,t=m+1;i>=0;--i){
197         while(m>=t&&(p[m-1]-p[m-2]).cross(s[i]
198             -p[m-2])<=0)--m;
199         p[m++]=s[i];
200     }
201     if(s.size()>1)--m;
202     p.resize(m);
203 }
204 T diam(){//直徑
205     int n=p.size(),t=1;
206     T ans=0;p.push_back(p[0]);
207     for(int i=0;i<n;i++){
208         point<T> now=p[i+1]-p[i];
209         while(now.cross(p[t+1]-p[i])>now.cross
210             (p[t]-p[i]))t=(t+1)%n;
211         ans=max(ans,(p[i]-p[t]).abs2());
212     }
213     return p.pop_back(),ans;
214 }
215 T min_cover_rectangle(){//最小覆蓋矩形
216     int n=p.size(),t=1,r=1,l=1;
217     if(n<3)return 0;//也可以做最小周長矩形
218     T ans=1e99;p.push_back(p[0]);
219     for(int i=0;i<n;i++){
220         point<T> now=p[i+1]-p[i];
221         while(now.cross(p[t+1]-p[i])>now.cross
222             (p[t]-p[i]))t=(t+1)%n;
223         while(now.dot(p[r+1]-p[i])>now.dot(p[r]
224             -p[i]))r=(r+1)%n;
225         if(!l)l=r;
226         while(now.dot(p[l+1]-p[i])<now.dot(p[l]
227             -p[i]))l=(l+1)%n;
228         T d=now.abs2();
229         T tmp=now.cross(p[t]-p[i])*(now.dot(p[r]
230             -p[i])-now.dot(p[l]-p[i]))/d;
231         ans=min(ans,tmp);
232     }
233     return p.pop_back(),ans;
234 }
235 T dis2(polygon &p1){//凸包最近距離平方
236     vector<point<T> > &P=p,&Q=p1.p;
237     int n=P.size(),m=Q.size(),l=0,r=0;
238     for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
239     for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
240     P.push_back(P[0]),Q.push_back(Q[0]);
241     T ans=1e99;
242     for(int i=0;i<n;++i){
243         while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])
244             <0)r=(r+1)%m;
245         ans=min(ans,line<T>(P[l],P[l+1]).
246             seg_dis2(line<T>(Q[r],Q[r+1])));
247         l=(l+1)%n;
248     }
249     return P.pop_back(),Q.pop_back(),ans;
250 }
251 static char sign(const point<T>&t){
252     return (t.y==0?t.x:t.y)<0;
253 }
254 static bool angle_cmp(const line<T>& A,
255     const line<T>& B){
256     point<T> a=A.p2-A.p1,b=B.p2-B.p1;
257     return sign(a)<sign(b)||((sign(a)==sign(b)
258         )&&a.cross(b)>0);
259 }
260 int halfplane_intersection(vector<line<T>
261     > &s){//半平面交
262     sort(s.begin(),s.end(),angle_cmp);//線段
263     //左側為該線段半平面
264     int L,R,n=s.size();
265     vector<point<T> > px(n);
266     vector<line<T> > q(n);
267     q[L=R=0]=s[0];
268     for(int i=1;i<n;++i){
269         while(L<R&&s[i].ori(px[R-1])<=0)--R;
270         while(L<R&&s[i].ori(px[L])<=0)++L;
271         q[++R]=s[i];
272         if(q[R].parallel(q[R-1])){
273             --R;
274             if(q[R].ori(s[i].p1)>0)q[R]=s[i];
275         }
276         if(L<R)px[R-1]=q[R-1].
277             line_intersection(q[R]);
278     }
279     while(L<R&&q[L].ori(px[R-1])<=0)--R;
280     p.clear();
281     if(R-L<=1)return 0;
282     px[R]=q[R].line_intersection(q[L]);
283     for(int i=L;i<R;++i)p.push_back(px[i]);
284     return R-L+1;
285 }
286 template<typename T>
287 struct triangle{
288     point<T> a,b,c;
289     triangle(){
290         triangle(const point<T> &a,const point<T>
291             &b,const point<T> &c):a(a),b(b),c(c){}
292     }
293     T area(){const{
294         T t=(b-a).cross(c-a)/2;
295         return t>0?t:-t;
296     }
297     point<T> barycenter(){const{//重心
298         return (a+b+c)/3;
299     }
300     point<T> circumcenter(){const{//外心
301         static line<T> u,v;
302         u.p1=(a+b)/2;
303         u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-
304             b.x);
305         v.p1=(a+c)/2;
306         v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-
307             c.x);
308         return u.line_intersection(v);
309     }
310     point<T> incenter(){const{//內心
311         T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2
312             ()),C=sqrt((a-b).abs2());
313         return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+
314             B*b.y+C*c.y)/(A+B+C);
315     }
316 }
317 point<T> perpcenter(){const{//垂心
318     return barycenter()*3-circumcenter()*2;
319 }
320 };
321 template<typename T>
322 struct point3D{
323     T x,y,z;
324     point3D(){
325         point3D(const T&x,const T&y,const T&z):x(x
326             ),y(y),z(z){}
327     }
328     point3D operator+(const point3D &b)const{
329         return point3D(x+b.x,y+b.y,z+b.z);
330     }
331     point3D operator-(const point3D &b)const{
332         return point3D(x-b.x,y-b.y,z-b.z);
333     }
334     point3D operator*(const T &b)const{
335         return point3D(x*b,y*b,z*b);
336     }
337     point3D operator/(const T &b)const{
338         return point3D(x/b,y/b,z/b);
339     }
340     bool operator==(const point3D &b)const{
341         return x==b.x&&y==b.y&&z==b.z;
342     }
343     T dot(const point3D &b)const{
344         return x*b.x+y*b.y+z*b.z;
345     }
346     point3D cross(const point3D &b)const{
347         return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
348             *b.y-y*b.x);
349     }
350     T abs2(){const{//向量長度的平方
351         return dot(*this);
352     }
353     T area2(const point3D &b)const{//和b、原點
354         //圍成面積的平方
355         return cross(b).abs2()/4;
356     }
357 };
358 template<typename T>
359 struct line3D{
360     point3D<T> p1,p2;
361     line3D(){
362         line3D(const point3D<T> &p1,const point3D<
363             T> &p2):p1(p1),p2(p2){}
364     }
365     T dis2(const point3D<T> &p,bool is_segment
366         =0)const{//點跟直線/線段的距離平方
367         point3D<T> v=p2-p1,v1=p-p1;
368         if(is_segment){
369             point3D<T> v2=p-p2;
370             if(v.dot(v1)<=0)return v1.abs2();
371             if(v.dot(v2)>=0)return v2.abs2();
372         }
373         point3D<T> tmp=v.cross(v1);
374         return tmp.abs2()/v.abs2();
375     }
376     pair<point3D<T>,point3D<T> > closest_pair(
377         const line3D<T> &l)const{
378         point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
379         point3D<T> N=v1.cross(v2),ab(p1-l.p1);
380         //if(N.abs2()==0)return NULL;平行或重合
381         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
382         //最近點對距離
383         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
384             cross(d2),G=l.p1-p1;
385         T t1=(G.cross(d2)).dot(D)/D.abs2();
386         T t2=(G.cross(d1)).dot(D)/D.abs2();
387         return make_pair(p1+d1*t1,l.p1+d2*t2);
388     }
389     bool same_side(const point3D<T> &a,const
390         point3D<T> &b)const{
391         return (p2-p1).cross(a-p1).dot((p2-p1).
392             cross(b-p1))>0;
393     }
394 };
395 template<typename T>
396 struct plane{
397     point3D<T> p0,n;//平面上的點和法向量
398     plane(){
399         plane(const point3D<T> &p0,const point3D<T>
400             &n):p0(p0),n(n){}
401     }
402     T dis2(const point3D<T> &p)const{//點到平
403         //面距離的平方
404         T tmp=(p-p0).dot(n);
405         return tmp*tmp/n.abs2();
406     }
407     point3D<T> projection(const point3D<T> &p)
408         const{
409         return p-n*(p-p0).dot(n)/n.abs2();
410     }
411     point3D<T> line_intersection(const line3D<
412         T> &l)const{
413         T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
414         //重合該平面
415         return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
416             tmp);
417     }
418     line3D<T> plane_intersection(const plane &
419         p1)const{
420         point3D<T> e=n.cross(p1.n),v=n.cross(e);
421         T tmp=p1.n.dot(v);//等於0表示平行或重合
422         //該平面
423         point3D<T> q=p0+(v*(p1.n.dot(p1.p0-p0))/
424             tmp);
425         return line3D<T>(q,q+e);
426     }
427 };
428 return (p2-p1).cross(a-p1).dot((p2-p1).
429     cross(b-p1))>0;
430 }
431 };
432 template<typename T>
433 struct plane{
434     point3D<T> p0,n;//平面上的點和法向量
435     plane(){
436         plane(const point3D<T> &p0,const point3D<T>
437             &n):p0(p0),n(n){}
438     }
439     T dis2(const point3D<T> &p)const{//點到平
440         //面距離的平方
441         T tmp=(p-p0).dot(n);
442         return tmp*tmp/n.abs2();
443     }
444     point3D<T> projection(const point3D<T> &p)
445         const{
446         return p-n*(p-p0).dot(n)/n.abs2();
447     }
448     point3D<T> line_intersection(const line3D<
449         T> &l)const{
450         T tmp=n.dot(l.p2-l.p1);//等於0表示平行或
451         //重合該平面
452         return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/
453             tmp);
454     }
455     line3D<T> plane_intersection(const plane &
456         p1)const{
457         point3D<T> e=n.cross(p1.n),v=n.cross(e);
458         T tmp=p1.n.dot(v);//等於0表示平行或重合
459         //該平面
460         point3D<T> q=p0+(v*(p1.n.dot(p1.p0-p0))/
461             tmp);
462         return line3D<T>(q,q+e);
463     }
464 };
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

## 2 DP

### 2.1 整體二分

```

1 void compute(int L, int R, int optL, int
2     optR) {
3     if (L > R)
4         return;
5     int mid = L + (R - L) / 2;
6     DP[mid] = INF;
7     int opt = -1;
8     for (int k = optL; k <= min(mid - 1, optR)
9         ; k++) {
10         if (DP[mid] > f(k) + w(k, mid)) {
11             DP[mid] = f(k) + w(k, mid);
12             opt = k;
13         }
14     }
15     compute(L, mid - 1, optL, opt);
16     compute(mid + 1, R, opt, optR);
17 }
18 // compute(1, n, 0, n);

```

## 2.2 LineContainer

```

1 // Usually used for DP 斜率優化
2 template<class T>
3 T floor_div(T a, T b) {
4     return a / b - ((a ^ b) < 0 && a % b != 0)
5     ;
6 }
7 template<class T>
8 T ceil_div(T a, T b) {
9     return a / b + ((a ^ b) > 0 && a % b != 0)
10    ;
11 }
12 namespace line_container_internal {
13 struct line_t {
14     mutable long long k, m, p;
15
16     inline bool operator<(const line_t& o)
17         const { return k < o.k; }
18     inline bool operator<(long long x) const {
19         return p < x; }
20 };
21 // Line_container_internal
22 template<bool MAX>
23 struct line_container : std::multiset<
24     line_container_internal::line_t, std::
25     less<>> {
26     static const long long INF = std::
27         numeric_limits<long long>::max();
28
29     bool isect(iterator x, iterator y) {
30         if(y == end()) {
31             x->p = INF;
32             return 0;
33         }
34         if(x->k == y->k) {
35             x->p = (x->m > y->m ? INF : -INF);
36         }
37         else {
38             x->p = floor_div(y->m - x->m, x->k - y->k);
39         }
40         return x->p >= y->p;
41     }
42
43     void add_line(long long k, long long m) {
44         if(!MAX) {
45             k = -k;
46             m = -m;
47         }
48         auto z = insert({k, m, 0}), y = z++, x =
49             y;
50         while(isect(y, z)) {
51             z = erase(z);
52         }
53         if(x != begin() && isect(--x, y)) {
54             isect(x, y = erase(y));
55         }
56         while((y = x) != begin() && (--x)->p >=
57             y->p) {
58             isect(x, erase(y));
59         }
60     }
61 }
62 };

```

## 2.3 斜率優化-動態凸包

```

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

```

```

44     if (x != begin() && Intersect(--x, y
45         ))
46         Intersect(x, y = erase(y));
47     while ((y = x) != begin() && (--x)->
48         l >= y->l)
49         Intersect(x, erase(y));
50 }
51 ll query(ll x) const
52 {
53     auto l = *lower_bound(x);
54     return l.a * x + l.b;
55 }
56 } convexhull;
57 const ll maxn = 200005;
58 ll s[maxn];
59 ll f[maxn];
60 ll dp[maxn];
61 // CSES monster game2
62 int main()
63 {
64     Crbubble
65     ll n,m,i,k,t;
66     cin >> n >> f[0];
67     for(i=1;i<=n;i++) cin >> s[i];
68     for(i=1;i<=n;i++) cin >> f[i];
69     convexhull.Insert(f[0],0);
70     for(i=1;i<=n;i++)
71     {
72         dp[i] = convexhull.query(s[i]);
73         convexhull.Insert(f[i],dp[i]);
74     }
75     cout << dp[n] << endl;
76     return 0;
77 }

```

## 2.4 basic DP

```

1 // 0/1背包問題
2 for(int i=0;i<n;i++) {
3     for(int k = W; k >= w[i]; k--) {
4         dp[k] = max(dp[k],dp[k-w[i]]+v[i]);
5     }
6 }
7 //因為不能重複拿，所以要倒回來
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

```

```

25         *it = x;
26     }
27 }
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
47                 +1]) {
48                 dp[i+1][j+1] = dp[i+1][j];
49                 pre[i+1][j+1] = {i+1,j};
50             }
51             else {
52                 dp[i+1][j+1] = dp[i][j+1];
53                 pre[i+1][j+1] = {i,j+1};
54             }
55         }
56     }
57     int index1 = a.size(), index2 = b.size()
58     ;
59     string ans;
60     while(index1>0&&index2>0) {
61         if(pre[index1][index2] == make_pair(
62             index1-1,index2-1)) {
63             ans+=a[index1-1];
64         }
65         pair<int,int> u = pre[index1][index2
66             ];
67         index1= u.first;
68         index2= u.second;
69     }
70     for(int i=ans.size()-1;i>=0;i--)cout<<
71         ans[i];
72     return 0;
73 }

```

## 2.5 DP on Graph

```

1 //G.Longest Path
2 vector<vector<int>>> G;
3 vector<int> in;
4 int n, m;
5 cin >> n >> m;
6 G.assign(n + 1, {});
7 in.assign(n + 1, 0);
8 while (m--) {
9     int u, v;
10    cin >> u >> v;

```

```

11 G[u].emplace_back(v);
12 ++in[v];
13 }
14 int solve(int n) {
15     vector<int> DP(G.size(), 0);
16     vector<int> Q;
17     for (int u = 1; u <= n; ++u)
18         if (in[u] == 0)
19             Q.emplace_back(u);
20     for (size_t i = 0; i < Q.size(); ++i) {
21         int u = Q[i];
22         for (auto v : G[u]) {
23             DP[v] = max(DP[v], DP[u] + 1);
24             if (--in[v] == 0)
25                 Q.emplace_back(v);
26         }
27     }
28     return *max_element(DP.begin(), DP.end());
29 }
30 //max_independent_set on tree
31 vector<int> DP[2];
32 int dfs(int u, int pick, int parent = -1) {
33     if (u == parent) return 0;
34     if (DP[pick][u]) return DP[pick][u];
35     if (Tree[u].size() == 1) return pick; // 葉子
36     for (auto v : Tree[u]) {
37         if (pick == 0) {
38             DP[pick][u] += max(dfs(v, 0, u), dfs(v, 1, u));
39         } else {
40             DP[pick][u] += dfs(v, 0, u);
41         }
42     }
43     return DP[pick][u] += pick;
44 }
45 int solve(int n) {
46     DP[0] = DP[1] = vector<int>(n + 1, 0);
47     return max(dfs(1, 0), dfs(1, 1));
48 }
49 //Traveling Salesman // AtCoder
50 #include <bits/stdc++.h>
51 using namespace std;
52 const int INF = 1e9;
53 int cost(vector<tuple<int, int, int>> &point,
54         int from, int to) {
55     auto [x, y, z] = point[from];
56     auto [X, Y, Z] = point[to];
57     return abs(X - x) + abs(Y - y) + max(0, Z - z);
58 } //從一個點走到另一個點的花費
59
60 signed main() {
61     int n; cin >> n;
62     vector<tuple<int, int, int>> point(n);
63     for (auto &[x, y, z]: point) {
64         cin >> x >> y >> z;
65     }
66     vector<vector<int>> dp(1 << n, vector<int>(n, INF));
67     //1 << n(2^n) 代表 1~n 的所有子集 · 代表走過的點
68     //n 代表走到的最後一個點
69     dp[0][0] = 0;
70     for (int i = 1; i < (1 << n); i++) {

```

```

71         for (int j = 0; j < n; j++) {
72             if (i & (1 << j)) {
73                 //j 是走到的最後一個點 · 必須要在 i 裡面
74                 for (int k = 0; k < n; k++) {
75                     dp[i][j] = min(dp[i][j], dp[i - (1 << j)][k] + cost(point, k, j));
76                     //i 集合裡面走到 j = i / j, 集合裡走到 k · 再從 k 走到 j
77                 }
78             }
79             //cout << dp[i][j] << ' ';
80         }
81         //cout << endl;
82     }
83     cout << dp[(1 << n) - 1][0]; //每個都要走到 · 要走到 1
84     return 0;
85 }

```

## 2.6 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;

```

## 3 Data Structure

### 3.1 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     ;
10     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 }

```

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

```

## 3.3 BIT

```

1 #define lowbit(x) x & -x
2
3 void modify(vector<int> &bit, int idx, int val) {
4     for (int i = idx; i <= bit.size(); i += lowbit(i)) bit[i] += val;

```



```

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

```

### 3.4 Dynamic Segment Tree

```

1 using ll = long long;
2 struct node {
3     node *l, *r; ll sum;
4     void pull() {
5         sum = 0;
6         for(auto x : {l, r}) if(x) sum += x->sum;
7     }
8     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     if(qr > m) ret += qry(o->r, ql, qr, m+1, r);
28     return ret;
29 }

```

### 3.5 掃描線 + 線段樹

```

1 //CSES Area of Rectangle
2 #include <bits/stdc++.h>
3 #define pb push_back
4 #define int long long
5 #define mid ((l + r) >> 1)
6 #define lc (p << 1)
7 #define rc ((p << 1) | 1)
8 using namespace std;
9 struct ooo{
10     int x, l, r, v;
11 };
12 const int inf = 1e6;
13 array<int, 800004> man, tag, cnt;
14 vector<ooo> Q;
15 bool cmp(ooo a, ooo b){
16     return a.x < b.x;
17 }
18 void pull(int p){
19     man[p] = min(man[lc], man[rc]);
20     if(man[lc] < man[rc]) cnt[p] = cnt[lc];
21     else if(man[rc] < man[lc]) cnt[p] = cnt[rc];
22     else cnt[p] = cnt[lc] + cnt[rc];
23 }
24 void push(int p){
25     man[lc] += tag[p];
26     man[rc] += tag[p];
27     tag[lc] += tag[p];
28     tag[rc] += tag[p];
29     tag[p] = 0;
30 }
31 void build(int p, int l, int r){
32     if(l == r){
33         cnt[p] = 1;
34         return;
35     }
36     build(lc, l, mid);
37     build(rc, mid + 1, r);
38     pull(p);
39 }
40 void update(int p, int l, int r, int ql, int
41     qr, int x){
42     if(ql > r || qr < l) return;
43     if(ql <= l && qr >= r){
44         man[p] += x;
45         tag[p] += x;
46         return;
47     }
48     push(p);
49     update(lc, l, mid, ql, qr, x);
50     update(rc, mid + 1, r, ql, qr, x);
51     pull(p);
52 }
53 signed main(){
54     int n, x1, y1, x2, y2, p = 0, sum = 0;
55     cin >> n;
56     for(int i = 1; i <= n; i++){
57         cin >> x1 >> y1 >> x2 >> y2;
58         Q.pb({x1, y1, y2 - 1, 1});
59         Q.pb({x2, y1, y2 - 1, -1});
60     }
61     sort(Q.begin(), Q.end(), cmp);
62     build(1, -inf, inf);

```

```

63     for(int i = -inf; i < inf; i++){
64         while(p < Q.size() && Q[p].x == i){
65             auto [x, l, r, v] = Q[p++];
66             update(1, -inf, inf, l, r, v);
67         }
68         sum += 2 * inf + 1 - cnt[1];
69     }
70     cout << sum << "\n";
71     return 0;
72 }
73 //長方形面積
74 long long AreaOfRectangles(vector<tuple<int,
75     int,int,int>>>v){
76     vector<tuple<int,int,int,int>>tmp;
77     int L = INT_MAX, R = INT_MIN;
78     for(auto [x1,y1,x2,y2]:v){
79         tmp.push_back({x1,y1+1,y2,1});
80         tmp.push_back({x2,y1+1,y2,-1});
81         R = max(R,y2);
82         L = min(L,y1);
83     }
84     vector<long long>seg((R-L+1)<<2),tag((R-L
85         +1)<<2);
86     sort(tmp.begin(),tmp.end());
87     function<void(int,int,int,int,int,int)>
88     update = [&](int ql,int qr,int val,int
89         l,int r,int idx){
90         if(ql<=l and r<=qr){
91             tag[idx]+=val;
92             if(tag[idx])seg[idx] = r-l+1;
93             else if(l==r)seg[idx] = 0;
94             else seg[idx] = seg[idx<<1]+seg[idx
95                 <<1|1];
96             return;
97         }
98         int m = (l+r)>>1;
99         if(ql<=m)update(ql,qr,val,l,m,idx<<1);
100         if(qr>m)update(ql,qr,val,m+1,r,idx<<1|1);
101         ;
102         if(tag[idx])seg[idx] = r-l+1;
103         else seg[idx] = seg[idx<<1]+seg[idx
104             <<1|1];
105     };
106     long long last_pos = 0,ans = 0;
107     for(auto [pos,l,r,val]:tmp){
108         ans+=(pos-last_pos)*seg[1];
109         update(1,r,val,L,R,1);
110         last_pos = pos;
111     }
112     return ans;
113 }
114 // CSES Intersection Points
115 #include <bits/stdc++.h>
116 #define int long long
117 #define pb push_back
118 using namespace std;
119 struct line{
120     int p, l, r;
121 };
122 const int inf = 1e6 + 1;
123 array<int, 200004> BIT;
124 vector<line> A, Q;
125 bool cmp(line a, line b){
126     return a.p < b.p;
127 }

```

```

128 }
129 void update(int p, int x){
130     for(; p < 200004; p += p & -p) BIT[p]
131         += x;
132 }
133 int query(int p){
134     int sum = 0;
135     for(; p; p -= p & -p) sum += BIT[p];
136     return sum;
137 }
138 int run(){
139     int ans = 0, p = 0;
140     for(auto [t, l, r] : Q){
141         while(p < A.size()){
142             auto [x, y, v] = A[p];
143             if(x > t) break;
144             update(y, v);
145             p++;
146         }
147         ans += query(r) - query(l - 1);
148     }
149     return ans;
150 }
151 signed main(){
152     int n, x1, x2, y1, y2;
153     cin >> n;
154     for(int i = 0; i < n; i++){
155         cin >> x1 >> y1 >> x2 >> y2;
156         x1 += inf, x2 += inf, y1 += inf, y2
157             += inf;
158         if(x1 == x2) Q.pb({x1, y1, y2});
159         else A.pb({x1, y1, 1}), A.pb({x2 +
160             1, y2, -1});
161     }
162     sort(Q.begin(), Q.end(), cmp);
163     sort(A.begin(), A.end(), cmp);
164     cout << run() << "\n";
165     return 0;
166 }

```

### 3.6 Persistent DSU

```

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

```

```

22 if(pa==pb)return 0;
23 if(rk[pa]<rk[pb])swap(pa,pb);
24 p->replace(pb,pa);
25 if(rk[pa]==rk[pb])rk[pa]++;
26 return 1;
27 }
28 };

```

### 3.7 DSU

```

1 struct DSU {
2     vector<int> dsu, sz;
3     DSU(int n) {
4         dsu.resize(n + 1);
5         sz.resize(n + 1, 1);
6         for (int i = 0; i <= n; i++) dsu[i] = i;
7     }
8     int find(int x) {
9         return (dsu[x] == x ? x : dsu[x] = find(
10             dsu[x]));
11     }
12     int unite(int a, int b) {
13         a = find(a), b = find(b);
14         if(a == b) return 0;
15         if(sz[a] > sz[b]) swap(a, b);
16         dsu[a] = b;
17         sz[b] += sz[a];
18         return 1;
19     }
20 };

```

### 3.8 陣列上 Treap

```

1
2
3 struct Treap {
4     Treap *lc = nullptr, *rc = nullptr;
5     unsigned pri, sz;
6     long long Val, Sum;
7     Treap(int Val):pri(rand()),sz(1),Val(Val),
8         Sum(Val),Tag(false) {}
9     void pull();
10    bool Tag;
11    void push();
12 } *root;
13
14 inline unsigned sz(Treap *x) {
15     return x ? x->sz:0;
16 }
17
18 inline void Treap::push() {
19     if(!Tag) return;
20     swap(lc,rc);
21     if(lc) lc->Tag ^= Tag;
22     if(rc) rc->Tag ^= Tag;
23     Tag = false;
24 }
25
26 inline void Treap::pull() {
27     sz = 1;

```

```

27 Sum = Val;
28 if(lc) {
29     sz += lc->sz;
30     Sum += lc->Sum;
31 }
32 if(rc) {
33     sz += rc->sz;
34     Sum += rc->Sum;
35 }
36 }
37
38 Treap *merge(Treap *a, Treap *b) {
39     if(!a || !b) return a ? a : b;
40     if(a->pri < b->pri) {
41         a->push();
42         a->rc = merge(a->rc,b);
43         a->pull();
44         return a;
45     }
46     else {
47         b->push();
48         b->lc = merge(a,b->lc);
49         b->pull();
50         return b;
51     }
52 }
53
54 pair<Treap *,Treap *> splitK(Treap *x,
55     unsigned K) {
56     Treap *a = nullptr, *b = nullptr;
57     x->push();
58     unsigned leftSize = sz(x->lc) + 1;
59     if(K >= leftSize) {
60         a = x;
61         tie(a->rc,b) = splitK(x->rc, K -
62             leftSize);
63     }
64     else {
65         b = x;
66         tie(a, b->lc) = splitK(x->lc, K);
67     }
68     x->pull();
69     return {a,b};
70 }
71
72 Treap *init(const vector<int> &a) {
73     Treap *root = nullptr;
74     for(size_t i = 0; i < a.size(); i++) {
75         root = merge(root,new Treap(a[i]));
76     }
77     return root;
78 }
79
80 long long query(Treap *&root, unsigned ql,
81     unsigned qr) {
82     auto [a,b] = splitK(root,ql);
83     auto [c,d] = splitK(b,qr-ql+1);
84     c->push();
85     long long Sum = c->Sum;
86     root = merge(a,merge(c,d));
87     return Sum;
88 }
89
90 void Reverse(Treap *&root, unsigned ql,
91     unsigned qr) {

```

```

89 auto [a,b] = splitK(root,ql);
90 auto [c,d] = splitK(b,qr-ql+1);
91 c->Tag ^= true;
92 root = merge(a, merge(c,d));
93 }

```

### 3.9 monotonic stack

```

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

```

### 3.10 Kruskal

```

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

```

### 3.11 Lazytag Segment Tree

```

1 using ll = long long;
2 const int N = 2e5 + 5;
3 #define lc(x) (x << 1)
4 #define rc(x) (x << 1 | 1)
5 ll seg[N << 2], tag[N << 2];
6 int n;
7
8 void pull(int id) {

```

```

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

```

### 3.12 2D BIT

```

1
2 //2維BIT
3 #define lowbit(x) (x&-x)
4
5 class BIT {
6     int n;
7     vector<int> bit;
8
9 public:
10     void init(int _n) {
11         n = _n;
12         bit.resize(n);
13         for(auto &b : bit) b = 0;
14     }
15     int query(int x) const {
16         int sum = 0;
17         for(; x; x -= lowbit(x))
18             sum += bit[x];
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);
35         for(auto &b : bit1D) b.init(_n);
36     }
37     int query(int x, int y) const {
38         int sum = 0;
39         for(; x; x -= lowbit(x))
40             sum += bit1D[x].query(y);
41         return sum;
42     }
43     void modify(int x, int y, int val) {
44         for(; x <= m; x += lowbit(x))
45             bit1D[x].modify(y, val);
46     }
47 };

```

### 3.13 monotonic queue

```

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

```

### 3.14 Prim

```

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

```

```

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
16     //之權重最小的頂點
17     for(int u = 0; u < V; u++) {
18         if(!used[u] && (v == -1 || mincost
19             [u] < mincost[v])) v = u;
20     }
21     if(v == -1) break;
22     used[v] = true; //將頂點v追加至X
23     res += mincost[v]; //加上邊的權重
24     for(int u = 0; u < V; u++) {
25         mincost[u] = min(mincost[u], cost
26             [v][u]);
27     }
28 }
29 return res;

```

### 3.15 回滾並查集

```

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

```

### 3.16 TimingSegmentTree

```

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

```

```

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

```

### 3.17 SegmentTree

```

1 //build
2 const int N = 100000 + 9;
3 int a[N]; //葉
4 int seg[4 * N];
5 void build(int id, int l, int r) { // 編號為
6     id 的節點 · 存的區間為 [l, r]
7     if (l == r) {
8         seg[id] = a[l]; // 葉節點的值
9         return;
10    }
11    int mid = (l + r) / 2; // 將區間切成兩半
12    build(id * 2, l, mid); // 左子節點
13    build(id * 2 + 1, mid + 1, r); // 右子節
14    點
15    seg[id] = seg[id * 2] + seg[id * 2 + 1]
16 }

```

```

17 //區間查詢
18
19 int query(int id, int l, int r, int ql, int
20     qr) {
21     if (r < ql || qr < l) return 0; //若目前
22     的區間與詢問的區間的交集為空的話 ·
23     return 0
24     if (ql <= l && r <= qr) return seg[id];
25     //若目前的區間是詢問的區間的子集的
26     話 · 則終止 · 並回傳當前節點的答案
27     int mid = (l + r) / 2;
28     return query(id * 2, l, mid, ql, qr) //
29     左
30     + query(id * 2 + 1, mid + 1, r, ql,
31     qr); //右
32     //否則 · 往左 · 右進行遞迴
33 }
34
35 //單點修改
36
37 void modify(int id, int l, int r, int i, int
38     x) {
39     if (l == r) {
40         seg[id] = x; // 將a[i]改成x
41         //seg[id] += x; // 將a[i]加上x
42         return;
43     }
44     int mid = (l + r) / 2;
45     // 根據修改的點在哪裡 · 來決定要往哪個子
46     樹進行DFS
47     if (i <= mid) modify(id * 2, l, mid, i,
48         x); //左
49     else modify(id * 2 + 1, mid + 1, r, i, x
50         ); //右
51     seg[id] = seg[id * 2] + seg[id * 2 + 1];
52 }

```

### 3.18 Persistent Segment Tree

```

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

```

```

19 cur->pull();
20 }
21
22 ll qry(node* a, node* b, int ql, int qr, int
    l = 1, int r = n) {
23     if (ql <= l && r <= qr) return b->sum - a
        ->sum;
24     int m = (l + r) >> 1; ll ret = 0;
25     if (ql <= m) ret += qry(a->l, b->l, ql, qr,
        l, m);
26     if (qr > m) ret += qry(a->r, b->r, ql, qr,
        m + 1, r);
27     return ret;
28 }

```

### 3.19 pbds

```

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

```

## 4 Flow

### 4.1 Property

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

### 4.2 Gomory Hu

```

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

```

### 4.3 MinCostMaxFlow

```

1 template<class Cap_t, class Cost_t>
2 class MCMF {
3 public:
4     struct Edge {
5         int from;
6         int to;
7         Cap_t cap;
8         Cost_t cost;
9         Edge(int u, int v, Cap_t _cap, Cost_t
            _cost) : from(u), to(v), cap(_cap),
            cost(_cost) {}
10    };
11
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    {
64        assert(0 <= s && s < n);
65        assert(0 <= t && t < n);
66        Cap_t cap = 0;
67        Cost_t cost = 0;
68        while(f > 0 && spfa(s, t)) {
69            Cap_t send = f;
70            int u = t;
71            while(u != s) {
72                const Edge& e = edges[previous_edge[
                    u]];
73                send = min(send, e.cap);
74                u = e.from;
75            }
76            u = t;
77            while(u != s) {
78                Edge& e = edges[previous_edge[u]];
79                e.cap -= send;
80                Edge& b = edges[previous_edge[u] ^
                    1];
81                b.cap += send;
82                u = e.from;
83            }
84            cap += send;
85            f -= send;
86            cost += send * d[t];
87        }
88        return make_pair(cap, cost);
89    }

```

### 4.4 dinic

```

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

```



```

62 T flow(int s, int t, T f = numeric_limits<
    T>::max()) {
63     T ans = 0;
64     while(f > 0 && bfs(s, t)) {
65         cur.assign(n, 0);
66         T send = dfs(s, t, f);
67         ans += send;
68         f -= send;
69     }
70     return ans;
71 }
72 vector<pair<int,int>> min_cut(int s) {
73     vector<bool> vis(n);
74     vis[s] = true;
75     queue<int> que;
76     que.push(s);
77     while(!que.empty()) {
78         int u = que.front();
79         que.pop();
80         for(auto id : g[u]) {
81             const auto& e = edges[id];
82             int v = e.to;
83             if(e.cap > 0 && !vis[v]) {
84                 vis[v] = true;
85                 que.push(v);
86             }
87         }
88     }
89     vector<pair<int,int>> cut;
90     for(int i = 0; i < (int) edges.size(); i
        += 2) {
91         const auto& e = edges[i];
92         if(vis[e.from] && !vis[e.to]) {
93             cut.push_back(make_pair(e.from, e.to
                ));
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) :
        v(v), u(u), cap(cap) {}
109 };
110
111 struct Dinic {
112     const long long flow_inf = 1e18;
113     vector<FlowEdge> edges;
114     vector<vector<int>> adj;
115     int n, m = 0;
116     int s, t;
117     vector<int> level, ptr, path;
118     vector<vector<int>> paths;
119     queue<int> q;
120
121     Dinic(int n, int s, int t) : n(n), s(s),
        t(t) {
122         adj.resize(n);
        level.resize(n);
        ptr.resize(n);
    }
123
124     void add_edge(int v, int u, long long
        cap) {
125         edges.emplace_back(v, u, cap);
126         edges.emplace_back(u, v, 0);
127         adj[v].push_back(m);
128         adj[u].push_back(m + 1);
129         m += 2;
130     }
131
132     bool bfs() {
133         while (!q.empty()) {
134             int v = q.front();
135             q.pop();
136             for (int id : adj[v]) {
137                 if (edges[id].cap - edges[id]
                    .flow < 1)
138                     continue;
139                 if (level[edges[id].u] !=
                    -1)
140                     continue;
141                 level[edges[id].u] = level[v]
                    + 1;
142                 q.push(edges[id].u);
143             }
144         }
145         return level[t] != -1;
146     }
147
148     long long dfs(int v, long long pushed) {
149         if (pushed == 0)
150             return 0;
151         path.push_back(v);
152         if (v == t) {
153             for (int iiddxx = 0; iiddxx <
                pushed; ++iiddxx)
154                 paths.push_back(path);
155             path.pop_back();
156             return pushed;
157         }
158         for (int& cid = ptr[v]; cid < (int)
            adj[v].size(); cid++) {
159             int id = adj[v][cid];
160             int u = edges[id].u;
161             if (level[v] + 1 != level[u] ||
                edges[id].cap - edges[id].
                    flow < 1)
162                 continue;
163             long long tr = dfs(u, min(pushed
                , edges[id].cap - edges[id].
                    flow));
164             if (tr == 0)
165                 continue;
166             edges[id].flow += tr;
167             edges[id ^ 1].flow -= tr;
168             path.pop_back();
169             return tr;
170         }
171         path.pop_back();
172         return 0;
173     }
174
175     long long flow() {
176         long long f = 0;
177         while (true) {
178             fill(level.begin(), level.end(),
                -1);
179             level[s] = 0;
180             q.push(s);
181             if (!bfs())
182                 break;
183             fill(ptr.begin(), ptr.end(), 0);
184             while (long long pushed = dfs(s,
                flow_inf)) {
185                 f += pushed;
186             }
187         }
188         return f;
189     }
190
191 int main() {
192     int n, m, v, u;
193     cin >> n >> m;
194     Dinic D(n+1, 1, n);
195     for (int i = 0; i < m; ++i) {
196         cin >> v >> u;
197         D.add_edge(v, u, 1);
198     }
199     D.flow();
200     Dinic FLOW(n+1, 1, n);
201     for (auto e : D.edges) {
202         if (e.flow > 0) {
203             FLOW.add_edge(e.v, e.u, 1);
204         }
205     }
206     cout << FLOW.flow() << "\n";
207     for (auto p : FLOW.paths) {
208         cout << p.size() << "\n";
209         for (auto verti : p)
210             cout << verti << " ";
211         cout << "\n";
212     }
213 }
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

## 4.5 ISAP with cut

```

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

```

## 5 Graph

### 5.1 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.EB(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    }

```

## 5.2 SPFA

```

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

```

## 5.3 最大團

```

1 struct MaxClique{

```

```

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

```

## 5.4 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 AdjacencyMatrixTy smoothing(AdjacencyMatrix
5     &G) {
6     size_t N = G.size(), Change = 0;
7     do {
8         Change = 0;
9         for(size_t u = 0; u < N; ++u) {
10            vector<size_t> E;
11            for(size_t v = 0; v < N && E.size() < 3; ++v)
12                if(G[u][v] && u != v) E.emplace_back(v);

```

```

12         if(E.size() == 1 || E.size() == 2) {
13             ++Change;
14             for(auto v : E) G[u][v] = G[v][u] = false;
15         }
16         if(E.size() == 2) {
17             auto [a,b] = make_pair(E[0], E[1]);
18             G[a][b] = G[b][a] = true;
19         }
20     }
21     while(Change);
22     return G;
23 }
24 //計算Degree
25 //O(n^2)
26 vector<size_t> getDegree(const
27     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> &
40     Degree) {
41     unordered_map<size_t, size_t> Num;
42     for(auto Val : Degree) ++Num[Val];
43     size_t N = Degree.size();
44     bool isK5 = Num[4] == 5 && Num[4] + Num[0] == N;
45     bool isK33 = Num[3] == 6 && Num[3] + Num[0] == N;
46     return isK5 || isK33;
47 }

```

## 5.5 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>>>&g,int u,int p = -1){
13         int child = 0;
14         dfn[u] = low[u] = ++dfn_cnt;
15         st.push(u);
16         vis[u] = 1;

```

```

15         if(g[u].empty() and p==-1){
16             bcc_id[u] = ++bcc_cnt;
17             bcc[bcc_cnt].push_back(u);
18             return;
19         }
20         for(auto v:g[u]){
21             if(v==p)continue;
22             if(!dfn[v]){
23                 build(g,v,u);
24                 child++;
25                 if(dfn[u]<=low[v]){
26                     is_ap[u] = 1;
27                     bcc_id[u] = ++bcc_cnt;
28                     bcc[bcc_cnt].push_back(u);
29                     while(vis[v]){
30                         bcc_id[st.top()] = bcc_cnt;
31                         bcc[bcc_cnt].push_back(st.top());
32                         vis[st.top()] = 0;
33                         st.pop();
34                     }
35                     low[u] = min(low[u],low[v]);
36                 }
37                 low[u] = min(low[u],dfn[v]);
38             }
39             if(p==-1 and child<2)is_ap[u] = 0;
40             if(is_ap[u])ap.push_back(u);
41         }
42     }
43 };

```

## 5.6 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 }

```

## 5.7 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;
19     for(auto v:G[u]) if(!dfn[v])
20         dfs(v, pa[dfn[v]] = dfn[u]);
21 }
22 int find(int y, int x){
23     if(y <= x) return y;
24     int tmp = find(pa[y], x);
25     if(semi[best[y]] > semi[best[pa[y]]])
26         best[y] = best[pa[y]];
27     return pa[y] = tmp;
28 }
29 void tarjan(int root){
30     dfnCnt = 0;
31     for(int i=1; i<=n; ++i){
32         dfn[i] = idom[i] = 0;
33         tree[i].clear();
34         best[i] = semi[i] = i;
35     }
36     dfs(root);
37     for(int i=dfnCnt; i>1; --i){
38         int u = id[i];
39         for(auto v:rG[u]) if(v=dfn[v]){
40             find(v, i);
41             semi[i] = min(semi[i], semi[best[v]]);
42         }
43         tree[semi[i]].push_back(i);
44         for(auto v:tree[pa[i]]){
45             find(v, pa[i]);
46             idom[v] = semi[best[v]] == pa[i]
47                 ? pa[i] : best[v];
48         }
49         tree[pa[i]].clear();
50     }
51     for(int i=2; i<=dfnCnt; ++i){
52         if(idom[i] != semi[i])
53             idom[i] = idom[idom[i]];
54         tree[id[idom[i]]].push_back(i);
55     }
56 }
57 }dom;

```

## 5.8 判斷二分圖

```

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

```

```

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

```

## 5.9 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 }

```

## 5.10 Dijkstra

```

1 int Dijkstra(int s, int e, int N) {
2     const int INF = INT_MAX / 2;
3     vector<int> dist(N, INF);
4     vector<bool> used(N, false);
5
6     using T = tuple<int,int>;
7     priority_queue<T, vector<T>, greater<T>>
8         pq;
9
10    dist[s] = 0;
11    pq.emplace(0, s); // (w, e) 讓 pq 優先用
12        w 來比較
13
14    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
18             次
19
20         for (auto [e, w] : V[s]) {
21             if (dist[e] > dist[s] + w) {
22                 dist[e] = dist[s] + w;
23                 pq.emplace(dist[e], e);
24             }
25         }
26
27         return dist[e];
28 }

```

## 5.11 SCC

```

1 struct SCC{
2     int n, cnt = 0, dfn_cnt = 0;
3     vector<vector<int>> g;
4     vector<int> sz, scc, low, dfn;
5     stack<int> st;
6     vector<bool> vis;
7     SCC(int _n = 0) : n(_n){
8         sz.resize(n+5), scc.resize(n+5), low.
9             resize(n+5), dfn.resize(n+5), vis.
10                 resize(n+5);
11         g.resize(n+5);
12     }
13     inline void add_edge(int u, int v){
14         g[u].push_back(v);
15     }
16     inline void build(){
17         function<void(int, int)> dfs = [&](int u,
18             int dis){
19             low[u] = dfn[u] = ++dfn_cnt, vis[u] =
20                 1;
21             st.push(u);
22             for(auto v:g[u]){
23                 if(!dfn[v]){
24                     dfs(v, dis+1);
25                     low[u] = min(low[u], low[v]);
26                 }
27                 else if(vis[v]){
28                     low[u] = min(low[u], dfn[v]);
29                 }
30             }
31             if(low[u] == dfn[u]){
32                 ++cnt;
33                 while(vis[u]){
34                     auto v = st.top();
35                     st.pop();
36                     vis[v] = 0;
37                     scc[v] = cnt;
38                     sz[cnt]++;
39                 }
40             }
41         };
42         for(int i = 0; i < n; ++i)
43             if(!vis[i]) dfs(i, 0);
44     }

```

```

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

```

## 5.12 判斷環

```

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

```

## 5.13 2-SAT

```

1 struct two_sat{
2     SCC s;
3     vector<bool> ans;
4     int have_ans = 0;
5     int n;
6     two_sat(int _n) : n(_n) {
7         ans.resize(n+1);
8         s = SCC(2*n);

```

```

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

```

## 6 Math

### 6.1 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};

```

### 6.2 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 }

```

### 6.3 LinearCongruence

```

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

```

### 6.4 Miller-Rabin

```

1 bool is_prime(ll n, vector<ll> x) {
2     ll d = n - 1;
3     d >>= __builtin_ctzll(d);
4     for(auto a : x) {
5         if(n <= a) break;
6         ll t = d, y = 1, b = t;
7         while(b) {
8             if(b & 1) y = i128(y) * a % n;
9             a = i128(a) * a % n;
10            b >>= 1;
11        }
12        while(t != n - 1 && y != 1 && y != n -
13            1) {
14            y = i128(y) * y % n;
15            t <<= 1;
16        }
17        if(y != n - 1 && t % 2 == 0) return 0;
18    }
19    return 1;
20 }
21 bool is_prime(ll n) {
22     if(n <= 1) return 0;
23     if(n % 2 == 0) return n == 2;
24     if(n < (1LL << 30)) return is_prime(n, {2,
25         7, 61});
26     return is_prime(n, {2, 325, 9375, 28178,
27         450775, 9780504, 1795265022});
28 }

```

### 6.5 Bit Set

```

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

```

### 6.6 Lucas

```

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

```

### 6.7 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 }

```

### 6.8 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];

```

```

7 void phiTable(){
8     for(int i=1;i<=N;i++)phi[i]=i;
9     for(int i=1;i<=N;i++)for(x=i*2;x<=N;x+=i)
10        phi[x]-=phi[i];
11 }
12 void all_divdown(const LL &n) { // all n/x
13     for(LL a=1;a<=n;a=n/(n/(a+1))) {
14         // dosomething;
15     }
16 }
17 const int MAXPRIME = 1000000;
18 int iscom[MAXPRIME], prime[MAXPRIME],
19     primecnt;
20 int phi[MAXPRIME], mu[MAXPRIME];
21 void sieve(void){
22     memset(iscom,0,sizeof(iscom));
23     primecnt = 0;
24     phi[1] = mu[1] = 1;
25     for(int i=2;i<MAXPRIME;++i) {
26         if(!iscom[i]) {
27             prime[primecnt++] = i;
28             mu[i] = -1;
29             phi[i] = i-1;
30         }
31         for(int j=0;j<primecnt;++j) {
32             int k = i * prime[j];
33             if(k>=MAXPRIME) break;
34             iscom[k] = prime[j];
35             if(i%prime[j]==0) {
36                 mu[k] = 0;
37                 phi[k] = phi[i] * prime[j];
38                 break;
39             } else {
40                 mu[k] = -mu[i];
41                 phi[k] = phi[i] * (prime[j]-1);
42             }
43         }
44     }
45 }
46 bool g_test(const LL &g, const LL &p, const
47     vector<LL> &v) {
48     for(int i=0;i<v.size();++i)
49         if(modexp(g,(p-1)/v[i],p)!=1)
50             return false;
51     return true;
52 }
53 LL primitive_root(const LL &p) {
54     if(p==2) return 1;
55     vector<LL> v;
56     Factor(p-1,v);
57     v.erase(unique(v.begin(), v.end()), v.end()
58         ());
59     for(LL g=2;g<p;++g)
60         if(g_test(g,p,v))
61             return g;
62     puts("primitive_root NOT FOUND");
63     return -1;
64 }
65 int Legendre(const LL &a, const LL &p) {
66     return modexp(a%p,(p-1)/2,p);
67 }
68 LL inv(const LL &a, const LL &n) {
69     LL d,x,y;
70     gcd(a,n,d,x,y);
71     return d==1 ? (x+n)%n : -1;
72 }

```

```

68 }
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
78             LL &p) {
79     // a ^ x = b ( mod p )
80     int m=sqrt(p+.5), e=1;
81     LL v=inv(modexp(a,m,p), p);
82     map<LL, int> x;
83     x[1]=0;
84     for(int i=1; i<m; ++i) {
85         e = LLMul(e, a, p);
86         if(!x.count(e)) x[e] = i;
87     }
88     for(int i=0; i<m; ++i) {
89         if(x.count(b)) return i*m + x[b];
90         b = LLMul(b, v, p);
91     }
92     return -1;
93 }
94
95 LL Tonelli_Shanks(const LL &n, const LL &p)
96 {
97     // x^2 = n ( mod p )
98     if(n==0) return 0;
99     if(Legendre(n,p)!=1) while(1) { puts("SQRT
100         ROOT does not exist"); }
101     int S = 0;
102     LL Q = p-1;
103     while( !(Q&1) ) { Q>>=1; ++S; }
104     if(S==1) return modexp(n%p, (p+1)/4, p);
105     LL z = 2;
106     for(; Legendre(z,p)!=-1; ++z)
107         LL c = modexp(z, Q, p);
108     LL R = modexp(n%p, (Q+1)/2, p), t = modexp(n
109         %p, Q, p);
110     int M = S;
111     while(1) {
112         if(t==1) return R;
113         LL b = modexp(c, 1L<<(M-i-1), p);
114         R = LLMul(R, b, p);
115         t = LLMul(LLmul(b, b, p), t, p);
116         c = LLMul(b, b, p);
117         M = i;
118     }
119     return -1;
120 }
121
122 template<typename T>
123 T Euler(T n){
124     T ans=n;
125     for(T i=2; i*i<=n; ++i){
126         if(n%i==0){
127             ans=ans/i*(i-1);
128             while(n%i==0) n/=i;
129         }
130     }
131     if(n>1) ans=ans/n*(n-1);
132     return ans;
133 }

```

```

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[
149             i])-1, m[i])%M)%M;
150     }
151     /*如果m[i]是質數 · Euler(m[i])-1=m[i]-2 ·
152         就不用算Euler了*/
153     return ans;
154 }

```

## 6.9 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-->0) {
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 }

```

## 6.10 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  $\frac{(d_1-1)!(d_2-1)!\dots(d_n-1)!}{n^{n-2}}$  spanning trees.
- Let  $T_{n,k}$  be the number of labeled forests on  $n$  vertices with  $k$  components, such that vertex  $1, 2, \dots, k$  belong to different components. Then  $T_{n,k} = kn^{n-k-1}$ .

- Erdős–Gallai theorem

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

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

- Gale–Ryser theorem

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

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

- Fulkerson–Chen–Anstee theorem

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

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

- Möbius inversion formula

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

- Spherical cap

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

## 6.11 Matrix

```

1 template<typename T>
2 struct Matrix{
3     using rt = std::vector<T>;
4     using mt = std::vector<rt>;
5     using matrix = Matrix<T>;
6     int r, c;
7     mt m;
8     Matrix(int r, int c):r(r),c(c),m(r,rt(c)){}
9     rt& operator[](int i){return m[i];}
10    matrix operator+(const matrix &a){
11        matrix rev(r,c);
12        for(int i=0; i<r; ++i)
13            for(int j=0; j<c; ++j)
14                rev[i][j]=m[i][j]+a.m[i][j];
15        return rev;
16    }

```



```

17 matrix operator-(const matrix &a){
18     matrix rev(r,c);
19     for(int i=0;i<r;++i)
20         for(int j=0;j<c;++j)
21             rev[i][j]=m[i][j]-a.m[i][j];
22     return rev;
23 }
24 matrix operator*(const matrix &a){
25     matrix rev(r,a.c);
26     matrix tmp(a.c,a.r);
27     for(int i=0;i<a.r;++i)
28         for(int j=0;j<a.c;++j)
29             tmp[j][i]=a.m[i][j];
30     for(int i=0;i<r;++i)
31         for(int j=0;j<a.c;++j)
32             for(int k=0;k<c;++k)
33                 rev.m[i][j]+=m[i][k]*tmp[j][k];
34     return rev;
35 }
36 bool inverse(){
37     Matrix t(r,r+c);
38     for(int y=0;y<r;y++){
39         t.m[y][c+y] = 1;
40         for(int x=0;x<c;++x)
41             t.m[y][x]=m[y][x];
42     }
43     if( !t.gas() )
44         return false;
45     for(int y=0;y<r;y++){
46         for(int x=0;x<c;++x)
47             m[y][x]=t.m[y][c+x]/t.m[y][y];
48     return true;
49 }
50 T gas(){
51     vector<T> lazy(r,1);
52     bool sign=false;
53     for(int i=0;i<r;++i){
54         if( m[i][i]==0 ){
55             int j=i+1;
56             while(j<r&&!m[j][i])j++;
57             if(j==r)continue;
58             m[i].swap(m[j]);
59             sign=!sign;
60         }
61         for(int j=0;j<r;++j){
62             if(i==j)continue;
63             lazy[j]=lazy[j]*m[i][i];
64             T mx=m[j][i];
65             for(int k=0;k<c;++k)
66                 m[j][k]=m[j][k]*m[i][i]-m[i][k]*mx;
67         }
68     }
69     T det=sign?-1:1;
70     for(int i=0;i<r;++i){
71         det = det*m[i][i];
72         det = det/lazy[i];
73         for(auto &j:m[i])j/=lazy[i];
74     }
75     return det;
76 }
77 };

```

## 6.12 Numbers

- Bernoulli numbers

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

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

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

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

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

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

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

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

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

- Pentagonal number theorem

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

- Catalan numbers

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

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

- Eulerian numbers

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

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

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

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

## 6.13 FWT

```

1 vector<int> F_OR_T(vector<int> f, bool
2     inverse){
3     for(int i=0; (2<<i)<=f.size(); ++i)
4         for(int j=0; j<f.size(); j+=2<<i)
5             f[j+k+(1<<i)] += f[j+k]*(inverse
6                 ?-1:1);
7     return f;

```

```

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

```

## 6.14 找實根

```

1 // an*x^n + ... + a1x + a0 = 0;
2 int sign(double x){
3     return x < -eps ? -1 : x > eps;
4 }
5
6 double get(const vector<double>&coef, double
7     x){
8     double e = 1, s = 0;
9     for(auto i : coef) s += i*e, e *= x;
10    return s;
11 }
12 double find(const vector<double>&coef, int n
13     , double lo, double hi){
14     double sign_lo, sign_hi;
15     if( !(sign_lo = sign(get(coef, lo))) )
16         return lo;
17     if( !(sign_hi = sign(get(coef, hi))) )
18         return hi;
19     if(sign_lo * sign_hi > 0) return INF;
20     for(int stp = 0; stp < 100 && hi - lo >
21         eps; ++stp){
22         double m = (lo+hi)/2.0;
23         int sign_mid = sign(get(coef, m));
24         if(!sign_mid) return m;
25         if(sign_lo*sign_mid < 0) hi = m;
26         else lo = m;
27     }
28     return (lo+hi)/2.0;
29 }
30
31 vector<double> cal(vector<double>coef, int n
32     ){
33     vector<double>res;
34     if(n == 1){
35         if(sign(coef[1])) res.pb(-coef[0]/coef
36             [1]);
37         return res;
38     }
39     vector<double>dcoef(n);

```

```

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

```

## 6.15 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    }

```

## 6.16 FFT

```

1 // Fast-Fourier-Transform
2 using cd = complex<double>;
3 const double PI = acos(-1);
4
5 void FFT(vector<cd>& a, bool inv) {
6     int n = (int) a.size();
7     for(int i = 1, j = 0; i < n; ++i) {
8         int bit = n >> 1;
9         for(; j & bit; bit >>= 1) {
10             j ^= bit;
11         }
12         j ^= bit;
13         if(i < j) {
14             swap(a[i], a[j]);
15         }
16     }
17     for(int len = 2; len <= n; len <= 1) {
18         const double ang = 2 * PI / len * (inv ?
19             -1 : +1);
20         cd rot(cos(ang), sin(ang));
21         for(int i = 0; i < n; i += len) {
22             cd w(1);
23             for(int j = 0; j < len / 2; ++j) {
24                 cd u = a[i + j], v = a[i + j + len /
25                     2] * w;
26                 a[i + j] = u + v;
27                 a[i + j + len / 2] = u - v;
28                 w *= rot;
29             }
30         }
31     }
32     if(inv) {
33         for(auto& x : a) {
34             x /= n;
35         }
36     }
37 }
38
39 vector<int> multiply(const vector<int>& a,
40     const vector<int>& b) {
41     vector<cd> fa(a.begin(), a.end());
42     vector<cd> fb(b.begin(), b.end());
43     int n = 1;
44     while(n < (int) a.size() + (int) b.size()
45         - 1) {
46         n <<= 1;
47     }
48     fa.resize(n);
49     fb.resize(n);
50     FFT(fa, false);
51     FFT(fb, false);
52     for(int i = 0; i < n; ++i) {
53         fa[i] *= fb[i];
54     }
55     FFT(fa, true);
56     vector<int> c(a.size() + b.size() - 1);
57     for(int i = 0; i < (int) c.size(); ++i) {
58         c[i] = round(fa[i].real());
59     }
60     return c;
61 }

```

## 6.17 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    }
20    bool ok = true;
21    REP(i, n) {
22        if(abs(a[i][i]) <= EPS) {
23            if(abs(a[i][n]) > EPS) return -1;
24            ok = false;
25        }
26    }
27    return ok;
28 }

```

## 6.18 Pollard-Rho

```

1 void PollardRho(map<ll, int>& mp, ll n) {
2     if(n == 1) return;
3     if(is_prime(n)) return mp[n]++, void();
4     if(n % 2 == 0) {
5         mp[2] += 1;
6         PollardRho(mp, n / 2);
7         return;
8     }
9     ll x = 2, y = 2, d = 1, p = 1;
10    #define f(x, n, p) ((i128(x) * x % n + p)
11        % n)
12    while(1) {
13        if(d != 1 && d != n) {
14            PollardRho(mp, d);
15            PollardRho(mp, n / d);
16            return;
17        }
18        p += (d == n);
19        x = f(x, n, p), y = f(f(y, n, p), n, p);
20        d = __gcd(abs(x - y), n);
21    }
22    #undef f
23 }
24
25 vector<ll> get_divisors(ll n) {
26     if(n == 0) return {};
27     map<ll, int> mp;
28     PollardRho(mp, n);
29 }

```

```

28 vector<pair<ll, int>> v(ALL(mp));
29 vector<ll> res;
30 auto f = [&](auto f, int i, ll x) -> void
31 {
32     if(i == SZ(v)) {
33         res.pb(x);
34         return;
35     }
36     for(int j = v[i].second; ; j--) {
37         f(f, i + 1, x);
38         if(j == 0) break;
39         x *= v[i].first;
40     }
41 }
42 f(f, 0, 1);
43 sort(ALL(res));
44 return res;

```

## 7 Square root decomposition

### 7.1 MoAlgo

```

1 struct qry{
2     int ql,qr,id;
3 };
4 template<class T>struct Mo{
5     int n,m;
6     vector<pii>ans;
7     Mo(int _n,int _m): n(_n),m(_m){
8         ans.resize(m);
9     }
10    void solve(vector<T>&v,vector<qry>&q){
11        int l = 0,r = -1;
12        vector<int>cnt,cntcnt;
13        cnt.resize(n+5);
14        cntcnt.resize(n+5);
15        int mx = 0;
16        function<void(int)>add = [&](int pos){
17            cntcnt[cnt[v[pos]]]--;
18            cnt[v[pos]]++;
19            cntcnt[cnt[v[pos]]]++;
20            mx = max(mx,cnt[v[pos]]);
21        };
22        function<void(int)>sub = [&](int pos){
23            if(--cntcnt[cnt[v[pos]]] and cnt[v[
24                pos]]==mx)mx--;
25            cnt[v[pos]]--;
26            cntcnt[cnt[v[pos]]]++;
27            mx = max(mx,cnt[v[pos]]);
28        };
29        sort(all(q),[&](qry a,qry b){
30            static int B = max((int)1,n/max((int)
31                sqrt(m),(int)1));
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        }
40    }
41 }

```

```

37 while(l<ql)sub(l++);
38 while(r>qr)sub(r--);
39 ans[id] = {mx,cntcnt[mx]};
40 }
41 }
42 };

```

## 7.2 莫隊

```

1 void remove(idx); // TODO: remove value at
2     idx from data structure
3 void add(idx); // TODO: add value at idx
4     from data structure
5 int get_answer(); // TODO: extract the
6     current answer of the data structure
7
8 int block_size;
9
10 struct Query {
11     int l, r, idx;
12     bool operator<(Query other) const
13     {
14         return make_pair(l / block_size, r)
15             < make_pair(other.l /
16                 block_size, other.r);
17     }
18 };
19
20 vector<int> mo_s_algorithm(vector<Query>
21     queries) {
22     vector<int> answers(queries.size());
23     sort(queries.begin(), queries.end());
24
25     // TODO: initialize data structure
26
27     int cur_l = 0;
28     int cur_r = -1;
29     // invariant: data structure will always
30     // reflect the range [cur_l, cur_r]
31     for (Query q : queries) {
32         while (cur_l > q.l) {
33             cur_l--;
34             add(cur_l);
35         }
36         while (cur_r < q.r) {
37             cur_r++;
38             add(cur_r);
39         }
40         while (cur_l < q.l) {
41             remove(cur_l);
42             cur_l++;
43         }
44         while (cur_r > q.r) {
45             remove(cur_r);
46             cur_r--;
47         }
48         answers[q.idx] = get_answer();
49     }
50     return answers;
51 }

```

## 7.3 分塊 cf455D

```

1 const ll block_siz = 320;
2 const ll maxn = 100005;
3 ll a[maxn];
4 ll cnt[block_siz+1][maxn]; // i-th block, k'
5     s cou
6 deque<ll> q[block_siz+1];
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 int main()
17 {
18     Crbubble
19     ll n,m,i,k,t;
20     ll l,r,ord,pre,id,id2, ans = 0;
21     cin >> n;
22     for(i=0;i<n;i++)
23     {
24         cin >> a[i];
25         id = i/block_siz;
26         q[id].push_back(a[i]);
27         cnt[id][a[i]]++;
28     }
29     cin >> t;
30     while(t--)
31     {
32         cin >> ord >> l >> r;
33         l = (l+ans-1)%n+1 -1;
34         r = (r+ans-1)%n+1 -1;
35         if(l > r) swap(l,r);
36         id = l/block_siz; l %= block_siz;
37         id2 = r/block_siz; r %= block_siz;
38         if(ord == 1)
39         {
40             if(id == id2)
41             {
42                 pre = q[id][r];
43                 for(i=r;i>l;i--)
44                 {
45                     q[id][i] = q[id][i-1];
46                 }
47                 q[id][l] = pre;
48             }
49             else
50             {
51                 pre = q[id].back();
52                 cnt[id][pre]--;
53                 q[id].pop_back();
54                 for(i=id+1;i<id2;i++)
55                 {
56                     q[i].push_front(pre);
57                     cnt[i][pre]++;
58                     pre = q[i].back();
59                     cnt[i][pre]--;
60                     q[i].pop_back();
61                 }

```

```

62         q[id2].push_front(pre);
63         cnt[id2][pre]++;
64         pre = q[id2][r+1];
65         cnt[id2][pre]--;
66         q[id2].erase(q[id2].begin()+
67             r+1);
68         q[id].insert(q[id].begin()+1
69             , pre);
70         cnt[id][pre]++;
71     }
72     //print_all(n);
73     else
74     { // query m cnt
75         cin >> m;
76         m = (m+ans-1)%n+1;
77         ans = 0;
78         if(id == id2)
79         {
80             for(i=l;i<=r;i++) ans += (q[
81                 id][i] == m);
82         }
83         else
84         {
85             for(i=l;i<block_siz;i++) ans
86                 += (q[id][i] == m);
87             for(i=0;i<=r;i++) ans += (q[
88                 id2][i] == m);
89             for(i=id+1;i<id2;i++) ans +=
90                 cnt[i][m];
91         }
92         cout << ans << endl;
93     }
94     return 0;
95 }

```

## 8 Tree

### 8.1 Tree centroid

```

1 //找出其中一個樹重心
2 vector<int> size;
3
4 int ans = -1;
5 void dfs(int u, int parent = -1) {
6     size[u] = 1;
7     int max_son_size = 0;
8     for (auto v : Tree[u]) {
9         if (v == parent) continue;
10        dfs(v, u);
11        size[u] += size[v];
12        max_son_size = max(max_son_size, size[v]);
13    }
14    max_son_size = max(max_son_size, n - size[u]);
15    if (max_son_size <= n / 2) ans = u;
16 }

```

## 8.2 HLD

```

1 struct heavy_light_decomposition{
2     int n;
3     vector<int> dep, father, sz, mxson, topf, id;
4     vector<vector<int>>> g;
5     heavy_light_decomposition(int _n = 0) : n(
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 }

```

## 8.3 HeavyLight

```

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

```

## 8.4 centroidDecomposition

```

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    };

```

```

14 function<int(int,int)>dfs2 = [&](int u,int
15     p){
16     for(auto v:g[u]){
17         if(v==p)continue;
18         if(vis[v])continue;
19         if(sz[v]*2<n)continue;
20         return dfs2(v,u);
21     }
22     return u;
23 };
24 dfs1(u,-1);
25 return dfs2(u,-1);
26 }
27 int cal(int u,int p = -1,int deep = 1){
28     int ans = 0;
29     tmp.pb(deep);
30     sz[u] = 1;
31     for(auto v:g[u]){
32         if(v==p)continue;
33         if(vis[v])continue;
34         ans+=cal(v,u,deep+1);
35         sz[u]+=sz[v];
36     }
37     //calcuat 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         ans+=centroid_decomposition(v,sz[v]);
58     }
59     return ans;

```

## 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
    tree的根

```

```

10     return nd[nd[x].pa].ch[0]!=x&&nd[nd[x].pa
11         ].ch[1]!=x;
12 }
13 void down(int x){ //懶惰標記下推
14     if(nd[x].rev){
15         if(nd[x].ch[0])nd[nd[x].ch[0]].rev^=1;
16         if(nd[x].ch[1])nd[nd[x].ch[1]].rev^=1;
17         swap(nd[x].ch[0],nd[x].ch[1]);
18         nd[x].rev=0;
19     }
20 void push_down(int x){ //所有祖先懶惰標記下推
21     if(!isroot(x))push_down(nd[x].pa);
22     down(x);
23 }
24 void up(int x){ //將子節點的資訊向上更新
25 void rotate(int x){ //旋轉，會自行判斷轉的方
    向
26     int y=nd[x].pa,z=nd[y].pa,d=(nd[y].ch[1]==
27         x);
28     nd[x].pa=z;
29     if(!isroot(y))nd[z].ch[nd[z].ch[1]==y]=x;
30     nd[y].ch[d]=nd[x].ch[d^1];
31     nd[nd[y].ch[d]].pa=y;
32     nd[y].pa=x,nd[x].ch[d^1]=y;
33     up(y),up(x);
34 }
35 void splay(int x){ //將x伸展到splay tree的根
36     push_down(x);
37     while(!isroot(x)){
38         int y=nd[x].pa;
39         if(!isroot(y)){
40             int z=nd[y].pa;
41             if((nd[z].ch[0]==y)^(nd[y].ch[0]==x))
42                 rotate(y);
43             else rotate(x);
44         }
45         rotate(x);
46     }
47 int access(int x){
48     int last=0;
49     while(x){
50         splay(x);
51         nd[x].ch[1]=last;
52         up(x);
53         last=x;
54         x=nd[x].pa;
55     }
56     return last; //access後splay tree的根
57 }
58 void access(int x,bool is=0){ //is=0就是一般
    的access
59     int last=0;
60     while(x){
61         splay(x);
62         if(is&&!nd[x].pa){
63             //printf("%d\n",max(nd[last].ma,nd[nd[
64                 x].ch[1]].ma));
65         }
66         nd[x].ch[1]=last;
67         up(x);
68         last=x;
69         x=nd[x].pa;

```

```

68     }
69 }
70 void query_edge(int u,int v){
71     access(u);
72     access(v,1);
73 }
74 void make_root(int x){
75     access(x),splay(x);
76     nd[x].rev^=1;
77 }
78 void make_root(int x){
79     nd[access(x)].rev^=1;
80     splay(x);
81 }
82 void cut(int x,int y){
83     make_root(x);
84     access(y);
85     splay(y);
86     nd[y].ch[0]=0;
87     nd[x].pa=0;
88 }
89 void cut_parents(int x){
90     access(x);
91     splay(x);
92     nd[nd[x].ch[0]].pa=0;
93     nd[x].ch[0]=0;
94 }
95 void link(int x,int y){
96     make_root(x);
97     nd[x].pa=y;
98 }
99 int find_root(int x){
100     x=access(x);
101     while(nd[x].ch[0])x=nd[x].ch[0];
102     splay(x);
103     return x;
104 }
105 int query(int u,int v){
106     //傳回uv路徑splay tree的根結點
107     //這種寫法無法求LCA
108     make_root(u);
109     return access(v);
110 }
111 int query_lca(int u,int v){
112     //假設求鏈上點權的總和，sum是子樹的權重和，
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     }
121     else{
122         //return nd[lca].data+nd[nd[lca].ch[1]].
123             sum+nd[u].sum
124     }
125 }
126 struct EDGE{
127     int a,b,w;
128     e[10005];
129 int n;
130 vector<pair<int,int>> G[10005];
131 //first表示子節點，second表示邊的編號
132 int pa[10005],edge_node[10005];

```

```

129 //pa是父母節點，暫存用的，edge_node是每個編
    被存在哪個點裡面的陣列
130 void bfs(int root){
131     //在建構的時候把每個點都設成一個splay tree
132     queue<int> q;
133     for(int i=1;i<=n;++i)pa[i]=0;
134     q.push(root);
135     while(q.size()){
136         int u=q.front();
137         q.pop();
138         for(auto P:G[u]){
139             int v=P.first;
140             if(v!=pa[u]){
141                 pa[v]=u;
142                 nd[v].pa=u;
143                 nd[v].data=e[P.second].w;
144                 edge_node[P.second]=v;
145                 up(v);
146                 q.push(v);
147             }
148         }
149     }
150 }
151 void change(int x,int b){
152     splay(x);
153     //nd[x].data=b;
154     up(x);
155 }

```

## 8.6 LCA

```

1 const int MAXN=200000; // 1-base
2 const int MLG=__lg(MAXN) + 1; //Log2(MAXN)
3 +1;
4 int pa[MLG+2][MAXN+5];
5 int dep[MAXN+5];
6 vector<int> G[MAXN+5];
7 void dfs(int x,int p=0){ //dfs(root);
8     pa[0][x]=p;
9     for(int i=0;i<=MLG;++i)
10         pa[i+1][x]=pa[i][pa[i][x]];
11     for(auto &i:G[x]){
12         if(i==p)continue;
13         dep[i]=dep[x]+1;
14         dfs(i,x);
15     }
16 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
34 //用樹壓平做
35 #define MAXN 100000
36 typedef vector<int> ::iterator VIT;
37 int dep[MAXN+5], in[MAXN+5];
38 int vs[2*MAXN+5];
39 int cnt; /*時間戳*/
40 vector<int> >G[MAXN+5];
41 void dfs(int x, int pa){
42     in[x]++; cnt;
43     vs[cnt]=x;
44     for(VIT i=G[x].begin(); i!=G[x].end(); ++i){
45         if(*i==pa) continue;
46         dep[*i]=dep[x]+1;
47         dfs(*i, x);
48         vs[++cnt]=x;
49     }
50 }
51
52 inline int find_lca(int a, int b){
53     if(in[a]>in[b]) swap(a, b);
54     return RMQ(in[a], in[b]);
55 }

```

## 8.7 Tree diameter

```

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

```

```

33         D2[u] = max(D2[u], dis);
34     }
35     ans = max(ans, D1[u] + D2[u]);
36 }

```

## 8.8 樹壓平

```

1 //紀錄 in & out
2 vector<int> Arr;
3 vector<int> In, Out;
4 void dfs(int u) {
5     Arr.push_back(u);
6     In[u] = Arr.size() - 1;
7     for(auto v : Tree[u]) {
8         if(v == parent[u])
9             continue;
10        parent[v] = u;
11        dfs(v);
12    }
13    Out[u] = Arr.size() - 1;
14 }
15
16 //進去出來都紀錄
17 vector<int> Arr;
18 void dfs(int u) {
19     Arr.push_back(u);
20     for(auto v : Tree[u]) {
21         if(v == parent[u])
22             continue;
23         parent[v] = u;
24         dfs(v);
25     }
26     Arr.push_back(u);
27 }
28
29 //用Treap紀錄
30 Treap *root = nullptr;
31 vector<Treap*> In, Out;
32 void dfs(int u) {
33     In[u] = new Treap(cost[u]);
34     root = merge(root, In[u]);
35     for(auto v : Tree[u]) {
36         if(v == parent[u])
37             continue;
38         parent[v] = u;
39         dfs(v);
40     }
41     Out[u] = new Treap(0);
42     root = merge(root, Out[u]);
43 }
44
45 //Treap紀錄Parent
46 struct Treap {
47     Treap *lc = nullptr, *rc = nullptr;
48     Treap *pa = nullptr;
49     unsigned pri, size;
50     long long Val, Sum;
51     Treap(int Val):
52         pri(rand()), size(1),
53         Val(Val), Sum(Val) {}
54     void pull();
55 };

```

```

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

```

## 9 string

### 9.1 KMP

```

1 const int N = 1e6+5;
2 /*產生fail function*/
3 void kmp_fail(char *s, int len, int *fail){
4     int id=-1;
5     fail[0]=-1;
6     for(int i=1; i<len; ++i){
7         while(~id&&s[id+1]!=s[i]) id=fail[id];
8         if(s[id+1]==s[i]) ++id;
9         fail[i]=id;
10    }
11 }
12 vector<int> match_index;
13 //以字串B匹配字串A，傳回匹配成功的數量(用B的fail)*/
14 int kmp_match(char *A, int lenA, char *B, int lenB, int *fail){
15     int id=-1, ans=0;

```

```

16 for(int i=0; i<lenA; ++i){
17     while(~id&&B[id+1]!=A[i]) id=fail[id];
18     if(B[id+1]==A[i]) ++id;
19     if(id==lenB-1){ /*匹配成功*/
20         ++ans, id=fail[id];
21         match_index.emplace_back(i + 1 - lenB);
22     }
23 }
24 return ans;
25 }

```

## 9.2 reverseBWT

```

1 const int MAXN = 305, MAXC = 'Z';
2 int ranks[MAXN], tots[MAXC], first[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])]+1;
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 }

```

## 9.3 Z

```

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

```



## 9.4 Trie

```

1 template<int ALPHABET = 26, char MIN_CHAR =
  'a'>
2 class trie {
3 public:
4     struct Node {
5         int go[ALPHABET];
6         Node() {
7             memset(go, -1, sizeof(go));
8         }
9     };
10    trie() {
11        newNode();
12    }
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
19    inline void insert(const vector<int>& a,
20        int p = 0) {
21        for(int v : a) {
22            p = next(p, v);
23        }
24
25        inline void clear() {
26            nodes.clear();
27            newNode();
28        }
29
30        inline int longest_common_prefix(const
31            vector<int>& a, int p = 0) const {
32            int ans = 0;
33            for(int v : a) {
34                if(nodes[p].go[v] != -1) {
35                    ans += 1;
36                    p = nodes[p].go[v];
37                } else {
38                    break;
39                }
40            }
41            return ans;
42        }
43    private:
44        vector<Node> nodes;
45
46        inline int newNode() {
47            nodes.emplace_back();
48            return (int) nodes.size() - 1;
49        }
50    };

```

## 9.5 Rolling Hash

```

1 //Rolling Hash(10 Hash) CF 1800 D. Remove
  Two Letters
2

```

```

3 #include <bits/stdc++.h>
4 using namespace std;
5
6 constexpr long long power(long long x, long
  long n, int m) {
7     if(m == 1) return 0;
8     unsigned int _m = (unsigned int)(m);
9     unsigned long long r = 1;
10    x %= m;
11    if(x < 0) {
12        x += m;
13    }
14    unsigned long long y = x;
15    while(n) {
16        if(n & 1) r = (r * y) % _m;
17        y = (y * y) % _m;
18        n >>= 1;
19    }
20    return r;
21 }
22
23 template<int HASH_COUNT, bool
  PRECOMPUTE_POWERS = false>
24 class Hash {
25 public:
26     static constexpr int MAX_HASH_PAIRS = 10;
27
28     // {mul, mod}
29     static constexpr const pair<int, int>
30         HASH_PAIRS[] = {{827167801,
31             999999937},
32             {998244353,
33             999999929},
34             {146672737,
35             922722049},
36             {204924373,
37             952311013},
38             {585761567,
39             955873937},
40             {484547929,
41             901981687},
42             {856009481,
43             987877511},
44             {852853249,
45             996724213},
46             {937381759,
47             994523539},
48             {116508269,
49             993179543}};
50
51    Hash() : Hash("") {}
52
53    Hash(const string& s) : n(s.size()) {
54        static_assert(HASH_COUNT > 0 &&
55            HASH_COUNT <= MAX_HASH_PAIRS);
56        for(int i = 0; i < HASH_COUNT; ++i) {
57            const auto& p = HASH_PAIRS[i];
58            pref[i].resize(n);
59            pref[i][0] = s[0];
60            for(int j = 1; j < n; ++j) {
61                pref[i][j] = (1LL * pref[i][j - 1] *
62                    p.first + s[j]) % p.second;
63            }
64            if(PRECOMPUTE_POWERS) {
65                build_powers(n);
66            }
67        }
68
69        void add_char(char c) {
70            for(int i = 0; i < HASH_COUNT; ++i) {
71                const auto& p = HASH_PAIRS[i];
72                pref[i].push_back((1LL * (n == 0 ? 0 :
73                    pref[i].back()) * p.first + c) %
74                    p.second);
75            }
76            n += 1;
77            if(PRECOMPUTE_POWERS) {
78                build_powers(n);
79            }
80        }
81
82        // Return hash values for [l, r)
83        array<int, HASH_COUNT> substr(int l, int r
84            ) {
85            array<int, HASH_COUNT> res{};
86            for(int i = 0; i < HASH_COUNT; ++i) {
87                res[i] = substr(i, l, r);
88            }
89            return res;
90        }
91
92        array<int, HASH_COUNT> merge(const vector<
93            pair<int, int>>& seg) {
94            array<int, HASH_COUNT> res{};
95            for(int i = 0; i < HASH_COUNT; ++i) {
96                const auto& p = HASH_PAIRS[i];
97                for(auto [l, r] : seg) {
98                    res[i] = (1LL * res[i] * get_power(i
99                        , r - l) + substr(i, l, r)) % p.
100                        second;
101                }
102            }
103            return res;
104        }
105
106        // build powers up to x^k
107        void build_powers(int k) {
108            for(int i = 0; i < HASH_COUNT; ++i) {
109                const auto& p = HASH_PAIRS[i];
110                int sz = (int) POW[i].size();
111                if(sz > k) {
112                    continue;
113                }
114                if(sz == 0) {
115                    POW[i].push_back(1);
116                    sz = 1;
117                }
118                while(sz <= k) {
119                    POW[i].push_back(1LL * POW[i].back()
120                        * p.first % p.second);
121                    sz += 1;
122                }
123            }
124        }
125
126        inline int size() const {
127            return n;
128        }
129    private:
130        int n;
131        static vector<int> POW[MAX_HASH_PAIRS];
132        array<vector<int>, HASH_COUNT> pref;
133
134        int substr(int k, int l, int r) {
135            assert(0 <= k && k < HASH_COUNT);
136            assert(0 <= l && l <= r && r <= n);
137            const auto& p = HASH_PAIRS[k];
138            if(l == r) {
139                return 0;
140            }
141            int res = pref[k][r - 1];
142            if(l > 0) {
143                res -= 1LL * pref[k][l - 1] *
144                    get_power(k, r - l) % p.second;
145            }
146            if(res < 0) {
147                res += p.second;
148            }
149            return res;
150        }
151
152        int get_power(int a, int b) {
153            if(PRECOMPUTE_POWERS) {
154                build_powers(b);
155                return POW[a][b];
156            }
157            const auto& p = HASH_PAIRS[a];
158            return power(p.first, b, p.second);
159        }
160    };
161
162 template<int A, bool B> vector<int> Hash<A,
  B>::POW[Hash::MAX_HASH_PAIRS];
163
164 void solve() {
165     int n;
166     string s;
167     cin >> n >> s;
168     Hash<10, true> h(s);
169     set<array<int, 10>> used;
170     for(int i = 0; i + 1 < n; ++i) {
171         used.insert(h.merge({{0, i}, {i + 2, n
172             }}));
173     }
174     cout << used.size() << "\n";
175 }
176
177 int main() {
178     ios::sync_with_stdio(false);
179     cin.tie(0);
180     int tt;
181     cin >> tt;
182     while(tt--) {
183         solve();
184     }
185     return 0;
186 }

```

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

```

## 9.7 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)*/
53 int match_0(const char *s){
54   int ans=0,id,p=0,i;
55   for(i=0;s[i];++i){
56     id=s[i]-L;
57     while(!S[p].next[id]&&p=S[p].fail;
58       if(!S[p].next[id])continue;
59     p=S[p].next[id];
60     ++S[p].cnt_dp; /*匹配成功則它所有後綴都
61     可以被匹配(DP計算)*/
62   }
63   for(i=qe-1;i>=0;--i){
64     ans+=S[q[i]].cnt_dp*S[q[i]].ed;
65     if(~S[q[i]].fail)S[S[q[i]].fail].cnt_dp+=S[q[i]].cnt_dp;
66   }
67   return ans;
68 }
69 int match_1(const char *s)const{
70   int ans=0,id,p=0,t;
71   for(int i=0;s[i];++i){
72     id=s[i]-L;
73     while(!S[p].next[id]&&p=S[p].fail;
74       if(!S[p].next[id])continue;
75     p=S[p].next[id];
76     if(S[p].ed)ans+=S[p].ed;
77     for(t=S[p].efl;~t;t=S[t].efl){
78       ans+=S[t].ed; /*因為都走efl邊所以保證
79       匹配成功*/
80     }
81   }
82   return ans;
83 }
84 int match_2(const char *s){
85   int ans=0,id,p=0,t;
86   ++vt;
87   /*把戳記vt+=1 只要vt沒溢位 所有S[p].vis==vt就會變成false
88   這種利用vt的方法可以O(1)歸零vis陣列*/
89   for(int i=0;s[i];++i){
90     id=s[i]-L;
91     while(!S[p].next[id]&&p=S[p].fail;
92       if(!S[p].next[id])continue;
93     p=S[p].next[id];
94     if(S[p].ed&&S[p].vis!=vt){
95       S[p].vis=vt;
96       ans+=S[p].ed;
97     }
98     for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t].efl){
99       S[t].vis=vt;
100       ans+=S[t].ed; /*因為都走efl邊所以保證
101       匹配成功*/
102     }
103   }
104   return ans;
105 }
106 void evolution(){
107   for(qs=1;qs!=qe;){
108     int p=q[qs++];
109     for(int i=0;i<R-L;++i)
110       if(S[p].next[i]==0)S[p].next[i]=S[S[p].fail].next[i];
111   }
112 }

```

## 9.8 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 }

```

## 9.9 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-1],r-i):1;
8     while(s[i+z[i]]==s[i-z[i]])++z[i];
9     if(z[i]+i>r)r=z[i]+i,l=i;
10  } //ans = max(z)-1
11 }

```

## 10 tools

### 10.1 pragma

```

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

```

```

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

```

## 10.2 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,
    popcnt")
5 #define IOS ios::sync_with_stdio(0),cin.tie
    (0),cout.tie(0)
6 #define int long long
7 #define double long double
8 #define pb push_back
9 #define sz(x) (int)(x).size()
10 #define all(v) begin(v),end(v)
11 #define debug(x) cerr<<#x<<" = "<<#x<<"\n"
12 #define LINE cout<<"\n-----\n"
13 #define endl '\n'
14 #define VI vector<int>
15 #define F first
16 #define S second
17 #define MP(a,b) make_pair(a,b)
18 #define rep(i,m,n) for(int i = m;i<=n;++i)
19 #define res(i,m,n) for(int i = m;i>=n;--i)
20 #define gcd(a,b) __gcd(a,b)
21 #define lcm(a,b) a*b/gcd(a,b)
22 #define Case() int _;cin>>_;for(int Case =
    1;Case<=++;Case)
23 #define pii pair<int,int>
24 using namespace __gnu_cxx;
25 using namespace __gnu_pbds;
26 using namespace std;
27 template <typename K, typename cmp = less<K
    >, typename T = thin_heap_tag> using
    _heap = __gnu_pbds::priority_queue<K,
    cmp, T>;
28 template <typename K, typename M = null_type
    > using _hash = gp_hash_table<K, M>;
29 const int N = 1e6+5,L = 20,mod = 1e9+7;
30 const long long inf = 2e18+5;
31 const double eps = 1e-7,pi = acos(-1);
32 void solve(){
33 }
34 signed main(){
35     IOS;
36     solve();
37 }
38
39 //使用內建紅黑樹

```

```

40 template<class T, typename cmp=less<>>struct
    _tree{//#include<bits/extc++.h>
41     tree<pair<T,int>,null_type,cmp,rb_tree_tag
        ,tree_order_statistics_node_update>st;
42     int id = 0;
43     void insert(T x){st.insert({x,id++});}
44     void erase(T x){st.erase(st.lower_bound({x
        ,0}));}
45     int order_of_key(T x){return st.
        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 };

```

## 10.3 Counting Sort

```

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

```

## 10.4 HashMap

```

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

```

## 10.5 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
        到return -1
4     while (l <= r) {
5         mid = (l+r)/2;
6         if (arr[mid] >= val) ret = mid, r =
            mid-1;
7         else l = mid+1;
8     }
9     return ret;
10 }

```

## 10.6 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(
        tmp));
7     for(auto i:v)ans.push_back(lower_bound(
        begin(tmp),end(tmp),i)-tmp.begin()+1);
8     return ans;
9 }

```

## 10.7 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 }

```

## 10.8 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);

```

## 10.9 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
                on test #%d",t);
22            return 0;
23        }
24        else if(et-st>=2000){
25            printf("\033[0;32mTime Limit exceeded
                \033[0m on test #%d, Time %.0lfms\
                n",t,et-st);
26            return 0;
27        }
28        else {
29            printf("\033[0;32mAccepted\033[0
                m on test #%d, Time %.0lfms\
                n", t, et - st);
30        }
31    }
32 }

```

## 10.10 bitset

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

# ACM ICPC Team Reference - Angry Crow Takes Flight!

## Contents

<b>1</b>	<b>Computational Geometry</b>	<b>1</b>
1.1	最近點對	1
1.2	Geometry	1
<b>2</b>	<b>DP</b>	<b>2</b>
2.1	整體二分	2
2.2	LineContainer	3
2.3	斜率優化-動態凸包	3
2.4	basic DP	3
2.5	DP on Graph	3
2.6	單調隊列優化	4
<b>3</b>	<b>Data Structure</b>	<b>4</b>
3.1	sparse table	4
3.2	BinaryTrie	4
3.3	BIT	4
3.4	Dynamic Segment Tree	5

3.5	掃描線 + 線段樹	5
3.6	Persistent DSU	5
3.7	DSU	6
3.8	陣列上 Treap	6
3.9	monotonic stack	6
3.10	Kruskal	6
3.11	Lazytag Segment Tree	6
3.12	2D BIT	6
3.13	monotonic queue	7
3.14	Prim	7
3.15	回滾並查集	7
3.16	TimingSegmentTree	7
3.17	SegmentTree	7
3.18	Persistent Segment Tree	7
3.19	pbds	8
<b>4</b>	<b>Flow</b>	<b>8</b>
4.1	Property	8
4.2	Gomory Hu	8
4.3	MinCostMaxFlow	8
4.4	dinic	8
4.5	ISAP with cut	9
<b>5</b>	<b>Graph</b>	<b>9</b>
5.1	橋連通分量	9
5.2	SPFA	10
5.3	最大團	10
5.4	判斷平面圖	10
5.5	雙連通分量 & 割點	10
5.6	Floyd Warshall	10
5.7	Dominator tree	10
5.8	判斷二分圖	11

5.9	Bellman Ford	11
5.10	Dijkstra	11
5.11	SCC	11
5.12	判斷環	11
5.13	2-SAT	11
<b>6</b>	<b>Math</b>	<b>12</b>
6.1	InvGCD	12
6.2	FastPow	12
6.3	LinearCongruence	12
6.4	Miller-Rabin	12
6.5	Bit Set	12
6.6	Lucas	12
6.7	ExtendGCD	12
6.8	Basic	12
6.9	質因數分解	13
6.10	Theorem	13
6.11	Matrix	13
6.12	Numbers	14
6.13	FWT	14
6.14	找實根	14
6.15	LinearSieve	14
6.16	FFT	15
6.17	Gauss-Jordan	15
6.18	Pollard-Rho	15
<b>7</b>	<b>Square root decomposition</b>	<b>15</b>
7.1	MoAlgo	15
7.2	莫隊	15
7.3	分塊 cf455D	16

<b>8</b>	<b>Tree</b>	<b>16</b>
8.1	Tree centroid	16
8.2	HLD	16
8.3	HeavyLight	16
8.4	centroidDecomposition	16
8.5	link cut tree	17
8.6	LCA	17
8.7	Tree diameter	18
8.8	樹壓平	18
<b>9</b>	<b>string</b>	<b>18</b>
9.1	KMP	18
9.2	reverseBWT	18
9.3	Z	18
9.4	Trie	19
9.5	Rolling Hash	19
9.6	suffix array lcp	20
9.7	AC 自動機	20
9.8	minimal string rotation	20
9.9	manacher	20
<b>10</b>	<b>tools</b>	<b>20</b>
10.1	pragma	20
10.2	Template	21
10.3	Counting Sort	21
10.4	HashMap	21
10.5	Bsearch	21
10.6	relabel	21
10.7	TenarySearch	21
10.8	template bubble	21
10.9	DuiPai	21
10.10	bitset	22



# ACM ICPC Judge Test - Angry Crow Takes Flight!

## C++ Resource Test

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

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

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

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