

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[
227             l]-p[i]))l=(l+1)%n;
228         T d=now.abs2();
229         T tmp=now.cross(p[t]-p[i])*(now.dot(p[
230             r]-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 template<typename T>
321 struct point3D{
322     T x,y,z;
323     point3D(){
324         point3D(const T&x,const T&y,const T&z):x(x
325             ),y(y),z(z){}
326     }
327     point3D operator+(const point3D &b)const{
328         return point3D(x+b.x,y+b.y,z+b.z);
329     }
330     point3D operator-(const point3D &b)const{
331         return point3D(x-b.x,y-b.y,z-b.z);
332     }
333     point3D operator*(const T &b)const{
334         return point3D(x*b,y*b,z*b);
335     }
336     point3D operator/(const T &b)const{
337         return point3D(x/b,y/b,z/b);
338     }
339     bool operator==(const point3D &b)const{
340         return x==b.x&&y==b.y&&z==b.z;
341     }
342     T dot(const point3D &b)const{
343         return x*b.x+y*b.y+z*b.z;
344     }
345     point3D cross(const point3D &b)const{
346         return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x
347             *b.y-y*b.x);
348     }
349     T abs2()const{//向量長度的平方
350         return dot(*this);
351     }
352     T area2(const point3D &b)const{//和b、原點
353         //圍成面積的平方
354         return cross(b).abs2()/4;
355     }
356 }
357 template<typename T>
358 struct line3D{
359     point3D<T> p1,p2;
360     line3D(){
361         line3D(const point3D<T> &p1,const point3D<
362             T> &p2):p1(p1),p2(p2){}
363     }
364     T dis2(const point3D<T> &p,bool is_segment
365         =0)const{//點跟直線/線段的距離平方
366         point3D<T> v=p2-p1,v1=p-p1;
367         if(is_segment){
368             point3D<T> v2=p-p2;
369             if(v.dot(v1)<=0)return v1.abs2();
370             if(v.dot(v2)>=0)return v2.abs2();
371         }
372         point3D<T> tmp=v.cross(v1);
373         return tmp.abs2()/v.abs2();
374     }
375     pair<point3D<T>,point3D<T> > closest_pair(
376         const line3D<T> &l)const{
377         point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
378         point3D<T> N=v1.cross(v2),ab(p1-l.p1);
379         //if(N.abs2()==0)return NULL; 平行或重合
380         T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//
381         //最近點對距離
382         point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.
383             cross(d2),G=l.p1-p1;
384         T t1=(G.cross(d2)).dot(D)/D.abs2();
385         T t2=(G.cross(d1)).dot(D)/D.abs2();
386         return make_pair(p1+d1*t1,l.p1+d2*t2);
387     }
388     bool same_side(const point3D<T> &a,const
389         point3D<T> &b)const{
390         return (p2-p1).cross(a-p1).dot((p2-p1).
391             cross(b-p1))>0;
392     }
393 };
394 template<typename T>
395 struct plane{
396     point3D<T> p0,n;//平面上的點和法向量
397     plane(){
398         plane(const point3D<T> &p0,const point3D<T>
399             &n):p0(p0),n(n){}
400     }
401     T dis2(const point3D<T> &p)const{//點到平
402         //面距離的平方
403         T tmp=(p-p0).dot(n);
404         return tmp*tmp/n.abs2();
405     }
406     point3D<T> projection(const point3D<T> &p)
407         const{
408         return p-n*(p-p0).dot(n)/n.abs2();
409     }
410 }
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 template<typename T>
428 struct triangle3D{
429     point3D<T> a,b,c;
430     triangle3D(){
431         triangle3D(const point3D<T> &a,const
432             point3D<T> &b,const point3D<T> &c):a(a
433             ),b(b),c(c){}
434     }
435     bool point_in(const point3D<T> &p)const{//
436         //點在該平面上的投影在三角形中
437         return line3D<T>(b,c).same_side(p,a)&&
438             line3D<T>(a,c).same_side(p,b)&&
439             line3D<T>(a,b).same_side(p,c);
440     }
441 };
442 template<typename T>
443 struct tetrahedron{//四面體
444     point3D<T> a,b,c,d;
445     tetrahedron(){
446         tetrahedron(const point3D<T> &a,const
447             point3D<T> &b,const point3D<T> &c,
448             const point3D<T> &d):a(a),b(b),c(c),d(
449             d){}
450     }
451     T volume6()const{//體積的六倍
452         return (d-a).dot((b-a).cross(c-a));
453     }
454     point3D<T> centroid()const{
455         return (a+b+c+d)/4;
456     }
457 }

```

```

395 bool point_in(const point3D<T> &p) const {
396     return triangle3D<T>(a,b,c).point_in(p)
        && triangle3D<T>(c,d,a).point_in(p);
397 }
398 };
399 template<typename T>
400 struct convexhull3D {
401     static const int MAXN=1005;
402     struct face {
403         int a,b,c;
404         face(int a,int b,int c):a(a),b(b),c(c){}
405     };
406     vector<point3D<T>> pt;
407     vector<face> ans;
408     int fid[MAXN][MAXN];
409     void build() {
410         int n=pt.size();
411         ans.clear();
412         memset(fid,0,sizeof(fid));
413         ans.emplace_back(0,1,2); //注意不能共線
414         ans.emplace_back(2,1,0);
415         int ftop = 0;
416         for(int i=3, ftop=1; i<n; ++i, ++ftop) {
417             vector<face> next;
418             for(auto &f:ans) {
419                 T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[f.a]).cross(pt[f.c]-pt[f.a]));
420                 if(d<=0) next.push_back(f);
421                 int ff=0;
422                 if(d>0) ff=ftop;
423                 else if(d<0) ff=-ftop;
424                 fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c][f.a]=ff;
425             }
426             for(auto &f:ans) {
427                 if(fid[f.a][f.b]>0 && fid[f.a][f.b]
                    !=fid[f.b][f.a])
428                     next.emplace_back(f.a,f.b,i);
429                 if(fid[f.b][f.c]>0 && fid[f.b][f.c]
                    !=fid[f.c][f.b])
430                     next.emplace_back(f.b,f.c,i);
431                 if(fid[f.c][f.a]>0 && fid[f.c][f.a]
                    !=fid[f.a][f.c])
432                     next.emplace_back(f.c,f.a,i);
433             }
434             ans=next;
435         }
436     }
437     point3D<T> centroid() const {
438         point3D<T> res(0,0,0);
439         T vol=0;
440         for(auto &f:ans) {
441             T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c]
                ));
442             res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
443             vol+=tmp;
444         }
445         return res/(vol*4);
446     }
447 };

```

2 DP

2.1 整體二分

```

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

```

2.2 LineContainer

```

1 // Usually used for DP 斜率優化
2 template<class T>
3 T floor_div(T a, T b) {
4     return a / b - ((a ^ b) < 0 && a % b != 0)
        ;
5 }
6
7 template<class T>
8 T ceil_div(T a, T b) {
9     return a / b + ((a ^ b) > 0 && a % b != 0)
        ;
10 }
11
12 namespace line_container_internal {
13
14 struct line_t {
15     mutable long long k, m, p;
16
17     inline bool operator<(const line_t& o)
        const { return k < o.k; }
18     inline bool operator<(long long x) const {
19         return p < x; }
20 };
21 // Line_container_internal
22
23 template<bool MAX>
24 struct line_container : std::multiset<
    line_container_internal::line_t, std::
    less<>> {
25     static const long long INF = std::
        numeric_limits<long long>::max();
26
27     bool isect(iterator x, iterator y) {
28         if(y == end()) {

```

```

29         x->p = INF;
30         return 0;
31     }
32     if(x->k == y->k) {
33         x->p = (x->m > y->m ? INF : -INF);
34     } else {
35         x->p = floor_div(y->m - x->m, x->k - y
            ->k);
36     }
37     return x->p >= y->p;
38 }
39
40 void add_line(long long k, long long m) {
41     if(!MAX) {
42         k = -k;
43         m = -m;
44     }
45     auto z = insert({k, m, 0}), y = z++, x =
        y;
46     while(isect(y, z)) {
47         z = erase(z);
48     }
49     if(x != begin() && isect(--x, y)) {
50         isect(x, y = erase(y));
51     }
52     while((y = x) != begin() && (--x->p >=
        y->p) {
53         isect(x, erase(y));
54     }
55 }
56
57 long long get(long long x) {
58     assert(!empty());
59     auto l = *lower_bound(x);
60     return (l.k * x + l.m) * (MAX ? +1 : -1)
        ;
61 }
62 };

```

2.3 斜率優化

```

1 using Slope = pair<long long, long long>;
2 // 注意要避免浮點數誤差
3 bool operator<=(const Slope &a, const Slope
    &b) {
4     // a.first/a.second <= b.first/b.second
5     return 1LL * a.first * b.second <= 1LL * b
        .first * a.second;
6 }
7
8 long long solve(vector<int> C, int n, int M)
    {
9     vector<long long> DP(n + 1), S(n + 1), X(n
        + 1), Y(n + 1);
10    partial_sum(C.begin(), C.end(), S.begin())
        ; //前綴和
11    auto getSlope = [&](int a, int b) -> Slope
        {
12        return {Y[b] - Y[a], X[b] - X[a]};
13    };
14    deque<int> q(1);
15    for (int i = 1; i <= n; ++i) {

```

```

16    long long A_i = -S[i], B_i = 1, C_i = 1
        LL * S[i] * S[i] + M;
17    Slope K_i = {-A_i, B_i};
18    while (q.size() > 1 && getSlope(q[0], q
        [1]) <= K_i)
19        q.pop_front();
20    int j = q[0];
21    DP[i] = A_i * X[j] + B_i * Y[j] + C_i;
22    Y[i] = DP[i] + 1LL * S[i] * S[i], X[i] =
        2 * S[i]; //計算X_i, Y_i
23    while ((j = q.size()) > 1 &&
        getSlope(q[j - 1], i) <= getSlope
        (q[j - 2], q[j - 1]))
24        q.pop_back();
25        q.push_back(i);
26    }
27    return DP[n];
28 }
29
30 // 形式: DP[i] = C_i + max{A_j * x_i + B_j},
31 // j <= R_i

```

2.4 basic DP

```

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

```

```

36 vector<vector<pair<int,int>>> pre(a.size
    ()+1,vector<pair<int,int>> (b.size()
    +1));
37 for(int i=0;i<a.size();i++) {
38     for(int j=0;j<b.size();j++) {
39         if(a[i] == b[j]) {
40             dp[i+1][j+1] = dp[i][j] + 1;
41             pre[i+1][j+1] = {i,j};
42         }
43         else if(dp[i+1][j] >= dp[i][j
            +1]) {
44             dp[i+1][j+1] = dp[i+1][j];
45             pre[i+1][j+1] = {i+1,j};
46         }
47         else {
48             dp[i+1][j+1] = dp[i][j+1];
49             pre[i+1][j+1] = {i,j+1};
50         }
51     }
52 }
53 int index1 = a.size(), index2 = b.size()
    ;
54 string ans;
55 while(index1>0&&index2>0) {
56     if(pre[index1][index2] == make_pair(
        index1-1,index2-1)) {
57         ans+=a[index1-1];
58     }
59 }
60 pair<int,int> u = pre[index1][index2
    ];
61 index1= u.first;
62 index2= u.second;
63 }
64 for(int i=ans.size()-1;i>=0;i--)cout<<
    ans[i];
65 return 0;
66 }

```

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_indepent_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
53 const int INF = 1e9;
54 int cost(vector<tuple<int,int,int>> &point,
    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
68     //1<<n(2^n)代表1~n的所有子集·代表走過的
        點
69
70     //n代表走到的最後一個點
71     dp[0][0] = 0;
72     for(int i=1;i<(1<<n);i++) {
73         for(int j=0;j<n;j++) {
74             if(i & (1<<j)) {
75                 //j是走到的最後一個點·必須
                    要在i裡面
76                 for(int k=0;k<n;k++) {
77                     dp[i][j] = min(dp[i][j],
                        dp[i-(1<<j)][k]+cost
                            (point,k,j));
78                     //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
    走回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     vector<long long> DP(N + 1);
3     deque<int> dq(1);
4     for (int i = 1; i <= N; ++i) {
5         while (dq.front() < i - K)
6             dq.pop_front();
7         DP[i] = DP[dq.front()] + a[i];
8         while (dq.size() && DP[dq.back()] > DP[i
            ])
9             dq.pop_back();
10        dq.push_back(i);
11    }
12    long long ans = INF;
13    for (int i = N - K + 1; i <= N; ++i)
14        ans = min(ans, DP[i]);
15    return ans;
16 }

```

3 DP 優化

3.1 斜率優化

```

1 // CSES Monster Game I
2 #include <bits/stdc++.h>
3 using namespace std;
4
5 typedef long long ll;
6
7 struct Line {
8     ll a, b; // 一條 ax + b 的直線
9     Line(ll _a, ll _b): a(_a), b(_b){}
10    ll operator()(const ll x) {
11        return a * x + b;
12    }
13 };
14
15 bool check(Line l1, Line l2, Line l3) {
16     // l1 是講義中的 L_{-2}·l2 是講義中的 L_
        {-1}·l3 是想要新增的直線
17     // double v12 = (l1.b - l2.b) / (l2.a - l1
        .a)
18     // double v23 = (l2.b - l3.b) / (l3.a - l2
        .a)

```

```

19 // return v12 >= v13
20 // 但是上面的方法會有浮點數誤差·因此在這
    裡只考慮使用整數運算·方法如下：
21 return (l3.a - l2.a) * (l1.b - l2.b) >= (
    l3.b - l2.b) * (l1.a - l2.a);
22 }
23
24 const int N = 200006;
25 ll dp[N], s[N], f[N];
26
27 void solve(int n) {
28     deque<Line> dq;
29     dq.push_back(Line(f[0], dp[0]));
30     for (int i = 1; i <= n; ++i) {
31         while ((int)dq.size() >= 2 && dq[0](s[i
            ]) <= dq[1](s[i])) {
32             // 把比較差的線丟掉·注意到這邊寫 <=
                或 < 其實都 ok
33             dq.pop_front();
34         }
35         dp[i] = dq[0](s[i]);
36         Line l = Line(f[i], dp[i]);
37         while ((int)dq.size() >= 2 && check(dq[(
            int)dq.size() - 2], dq[(int)dq.size
                () - 1], l)) {
38             // 把新的線加進去·看看 L_{-2}跟 L 有
                沒有辦法把 L_{-1}殺掉
39             dq.pop_back();
40         }
41         dq.push_back(l);
42     }
43 }
44
45 int main () {
46     ios::sync_with_stdio(0); cin.tie(0);
47     int n; cin >> n >> f[0];
48     for (int i = 1; i <= n; ++i) {
49         cin >> s[i];
50     }
51     for (int i = 1; i <= n; ++i) {
52         cin >> f[i];
53     }
54     for (int i = 0; i <= n; ++i) {
55         f[i] = -f[i];
56     }
57     solve(n);
58     cout << -dp[n] << '\n';
59 }
60
61 // CSES Monster Game II
62 // 斜率優化 + CDQ
63 #include <bits/stdc++.h>
64 using namespace std;
65
66 typedef long long ll;
67 const int N = 200006;
68
69 ll s[N], f[N], dp[N];
70
71 struct Line {
72     ll a, b;
73     Line(ll _a, ll _b): a(_a), b(_b){}
74     ll operator()(const ll x) {

```



```

76     return a * x + b;
77 }
78 };
79
80 bool check(Line l1, Line l2, Line l3) {
81     return (l3.a - l2.a) * (l1.b - l2.b) >= (
82         l3.b - l2.b) * (l1.a - l2.a);
83 }
84
85 void dc(int l, int r) {
86     if (l == r) return;
87     int mid = (l + r) >> 1;
88     dc(l, mid);
89     // use [l, mid] to update [mid + 1, r]
90     vector<Line> lines;
91     for (int j = l; j <= mid; ++j) {
92         lines.push_back(Line(f[j], dp[j]));
93     }
94     sort(lines.begin(), lines.end(), [](const
95         Line &l1, const Line &l2) {
96         return make_pair(l1.a, -l1.b) >
97             make_pair(l2.a, -l2.b);
98     });
99     vector<int> qs;
100     for (int i = mid + 1; i <= r; ++i) {
101         qs.push_back(i);
102     }
103     sort(qs.begin(), qs.end(), [](const int &i
104         , const int &j) {
105         return s[i] < s[j];
106     });
107     // 把線排序好後，先把凸包建立出來
108     deque<Line> dq;
109     for (Line l_new : lines) {
110         while ((int)dq.size() >= 2 && check(dq[
111             (int)dq.size() - 2], dq[(int)dq.size
112             () - 1], l_new)) {
113             dq.pop_back();
114         }
115         dq.push_back(l_new);
116     }
117     // 再一個一個去詢問
118     for (int i : qs) {
119         while (dq.size() >= 2 && dq[0](s[i]) >
120             dq[1](s[i])) {
121             dq.pop_front();
122         }
123         dp[i] = min(dp[i], dq[0](s[i]));
124     }
125     dc(mid + 1, r);
126 }
127
128 int main () {
129     ios::sync_with_stdio(0); cin.tie(0);
130     int n; cin >> n >> f[0];
131     for (int i = 1; i <= n; ++i) {
132         cin >> s[i];
133     }
134     for (int i = 1; i <= n; ++i) {
135         cin >> f[i];
136     }
137     for (int i = 1; i <= n; ++i) {
138         dp[i] = (1ll << 60);
139     }
140     dc(0, n);

```

```

134     cout << dp[n] << '\n';
135 }
136
137 signed main(){
138     int n, x;
139     cin >> n >> x;
140     for(int i = 1; i < 4000004; i++) seg[i]
141         = {inf, inf};
142     for(int i = 1; i <= n; i++) cin >> S[i];
143     for(int i = 1; i <= n; i++) cin >> F[i];
144     F[0] = x;
145     cout << DP(n) << "\n";
146     return 0;
147 }
148 //CSES Monster Game II
149 //斜率優化 + 凸包
150 #include <bits/stdc++.h>
151 using namespace std;
152
153 typedef long long ll;
154
155 struct Line {
156     mutable ll a, b, l; // 直線為 ax + b，有
157     // 效區間的左界為 l
158     Line(ll _a, ll _b, ll _l): a(_a), b(_b), l
159         (_l){}
160     bool operator<(const Line &rhs) const {
161         return a < rhs.a;
162     }
163     bool operator<(ll rhs_l) const {
164         return l < rhs_l;
165     }
166 };
167
168 struct ConvexHullMax : std::multiset<Line,
169     std::less<>> {
170     static const ll INF = (1ll << 60);
171     static ll DivCeil(ll a, ll b) { // a / b
172         // 取上高斯
173         return a / b - ((a ^ b) < 0 && a % b);
174     }
175     bool Intersect(iterator x, iterator y) {
176         // 用斜率相鄰的兩條線 x, y 來更新 x 的
177         // 有效區間
178         if (y == end()) {
179             x->l = INF;
180             return false;
181         }
182         if (x->a == y->a) {
183             x->l = x->b > y->b ? INF : -INF;
184         }
185         else {
186             x->l = DivCeil(y->b - x->b, x->a - y->
187                 a);
188         }
189         return x->l >= y->l; // 代表斜率比較低的
190         // 線的有效區間比較好，可以把斜率比較高
191         // 的線給殺掉
192     }
193     void Insert(ll a, ll b) {
194         auto z = insert(Line(a, b, 0)), y = z++,
195             x = y;
196         while (Intersect(y, z)) z = erase(z);

```

```

187     if (x != begin() && Intersect(--x, y))
188         Intersect(x, y = erase(y));
189     while ((y = x) != begin() && (--x->l >=
190         y->l) Intersect(x, erase(y)));
191 }
192 ll query(ll x) const {
193     auto l = *lower_bound(x);
194     return l.a * x + l.b;
195 }
196 } convexhull;
197
198 const int N = 200006;
199 ll s[N], f[N], dp[N];
200
201 int main () {
202     ios::sync_with_stdio(0); cin.tie(0);
203     int n; cin >> n >> f[0];
204     for (int i = 1; i <= n; ++i) {
205         cin >> s[i]; s[i] = -s[i];
206     }
207     for (int i = 1; i <= n; ++i) {
208         cin >> f[i];
209     }
210     convexhull.Insert(f[0], dp[0]);
211     for (int i = 1; i <= n; ++i) {
212         dp[i] = convexhull.query(s[i]);
213         convexhull.Insert(f[i], dp[i]);
214     }
215     cout << -dp[n] << '\n';

```

3.2 四邊形優化 (Knuth 優化)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 typedef long long ll;
5 const int N = 5006;
6
7 int K[N][N];
8 ll dp[N][N];
9
10 ll x[N], pre[N];
11
12 ll w(int i, int j) {
13     return pre[j] - pre[i - 1];
14 }
15
16 void solve(int n) {
17     memset(K, -1, sizeof(K));
18     for (int len = 1; len <= n; ++len) {
19         for (int i = 1; i + len - 1 <= n; ++i) {
20             int j = i + len - 1;
21             if (len == 1) {
22                 dp[i][j] = 0;
23                 K[i][j] = i;
24             }
25             else {
26                 int kl = K[i][j - 1];
27                 int kr = K[i + 1][j];
28                 for (int k = kl; k <= kr; ++k) {
29                     if (K[i][j] == -1) {
30                         K[i][j] = k;

```

```

31         dp[i][j] = w(i, j) + dp[i][k] +
32             dp[k + 1][j];
33     }
34     else if (w(i, j) + dp[i][k] + dp[k
35         + 1][j] < dp[i][j]) {
36         dp[i][j] = w(i, j) + dp[i][k] +
37             dp[k + 1][j];
38         K[i][j] = k;
39     }
40 }
41 }
42
43 int main () {
44     int n; cin >> n;
45     for (int i = 1; i <= n; ++i) {
46         cin >> x[i];
47         pre[i] = pre[i - 1] + x[i];
48     }
49     solve(n);
50     cout << dp[1][n] << '\n';
51 }

```

3.3 分治優化

```

1 // CSES Subarray Squares
2 #include <bits/stdc++.h>
3 #define int long long
4 using namespace std;
5 array<int, 3004> X;
6 array<array<int, 3004>, 3004> dp;
7 int cost(int l, int r) {
8     return (X[r] - X[l]) * (X[r] - X[l]);
9 }
10 void div(int ql, int qr, int l, int r, int k
11     ){
12     int t, qm = (ql + qr) >> 1;
13     dp[k][qm] = 1e18;
14     for(int i = l; i < min(r + 1, qm); i++){
15         if(dp[k - 1][i] + cost(i, qm) < dp[k
16             ][qm]){
17             t = i;
18             dp[k][qm] = dp[k - 1][i] + cost(
19                 i, qm);
20         }
21     }
22     if(ql == qr) return;
23     div(ql, qm, l, t, k);
24     div(qm + 1, qr, t, r, k);
25 }
26 int DP(int n, int k){
27     for(int i = 1; i <= n; i++){
28         dp[1][i] = X[i] * X[i];
29     }
30     for(int i = 2; i <= k; i++){
31         div(i, n, i - 1, n, i);
32     }
33     return dp[k][n];
34 }
35 signed main(){
36     int n, k;

```

```

34  cin >> n >> k;
35  for(int i = 1; i <= n; i++){
36      cin >> X[i];
37      X[i] += X[i - 1];
38  }
39  cout << DP(n, k) << "\n";
40  return 0;
41 }
42 // CSES Houses and Schools
43 #include <bits/stdc++.h>
44 #define int long long
45 using namespace std;
46 array<int, 3004> C;
47 array<array<int, 3004>, 3004> dis, cst, turn
48     , dp;
49 void DIS(int n){
50     for(int i = 1; i <= n; i++){
51         for(int j = i - 1; j > 0; j--){
52             dis[i][j] = (i - j) * C[j] + dis
53                 [i][j + 1];
54         }
55         for(int j = i + 1; j <= n; j++){
56             dis[i][j] = (j - i) * C[j] + dis
57                 [i][j - 1];
58         }
59     }
60     for(int i = 1; i <= n; i++){
61         cst[i][i] = 0;
62         turn[i][i] = i;
63     }
64     for(int k = 1; k < n; k++){
65         for(int i = 1, j = i + k; j <= n; i
66             ++, j++){
67             cst[i][j] = 1e18;
68             for(int t = turn[i][j - 1]; t <=
69                 turn[i + 1][j]; t++){
70                 if(dis[t][i] + dis[t][j] <
71                     cst[i][j]){
72                     cst[i][j] = dis[t][i] +
73                         dis[t][j];
74                     turn[i][j] = t;
75                 }
76             }
77         }
78     }
79 void div(int ql, int qr, int l, int r, int k
80     ){
81     int t, qm = (ql + qr) >> 1;
82     dp[k][qm] = 1e18;
83     for(int i = 1; i < min(r + 1, qm); i++){
84         if(dp[k][qm] > dp[k - 1][i] + cst[i
85             + 1][qm]){
86             dp[k][qm] = dp[k - 1][i] + cst[i
87                 + 1][qm];
88             t = i;
89         }
90     }
91     if(ql == qr) return;
92     div(ql, qm, l, t, k);
93     div(qm + 1, qr, t, r, k);
94 }
95 int DP(int n, int k){
96     for(int i = 1; i <= n; i++){
97         dp[1][i] = cst[1][i];

```

```

90     }
91     for(int i = 2; i <= k; i++){
92         div(i, n, i - 1, n, i);
93     }
94     return dp[k][n];
95 }
96 signed main(){
97     int n, k;
98     cin >> n >> k;
99     for(int i = 1; i <= n; i++) cin >> C[i];
100     DIS(n);
101     cout << DP(n, k) << "\n";
102     return 0;
103 }

```

3.4 單調隊列優化

```

1 // CSES Book Shop II
2 // 單調隊列優化的有限背包問題
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 signed main() {
7     int n,x;cin>>n>>x;
8     vector<int> weight(n+1),value(n+1),copies(
9         n+1);
10    for(int i=1;i<=n;i++) cin>>weight[i];
11    for(int i=1;i<=n;i++) cin>>value[i];
12    for(int i=1;i<=n;i++) cin>>copies[i];
13    vector<vector<int>> DP(n+1,vector<int> (x
14        +1,0));
15    int ans = 0;
16    for(int i=1;i<=n;i++) {
17        for(int j=0;j<weight[i];j++) { // 對每個
18            餘數維護一個單調隊列
19            deque<int> dq;
20            for(int k=j;k<=x;k+=weight[i]) {
21                if(!dq.empty() && k - dq.front() >
22                    weight[i] * copies[i]) dq.
23                    pop_front(); // 如果轉移不過來,移
24                    動sliding window, pop_front()
25                    while(!dq.empty() && DP[i-1][k] >=
26                        DP[i-1][dq.front()] + value[i] *
27                            ((k - dq.front()) / weight[i]))
28                            dq.pop_back(); // 維護一個遞減
29                            的deque, 如果前面的比k小, 砍掉,
30                            將k丟進deque裡, 維持deque為遞
31                            減, 而deque中第一個元素, 就是轉
32                            移來源
33                dq.emplace_back(k);
34                DP[i][k] = DP[i-1][dq.front()] +
35                    value[i] * ((k - dq.front()) /
36                        weight[i]);
37                ans = max(ans, DP[i][k]);
38            }
39        }
40    }
41    cout<<ans;
42 }

```

4 Data Structure

4.1 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];
11    for(int i=1;(1<<i)<=n;i++) {
12        for(int j=1;j + (1<<i) - 1 <= n;j++) {
13            st[i][j] = min(st[i-1][j],st[i-1][j
14                +(1<<(i-1))]);
15        }
16    }
17
18 int query(int l, int r) {
19     int k = __lg(r - l + 1);
20     return min(st[k][l],st[k][r-(1<<k)+1]);
21 }
22
23 signed main() {
24     int n,q;cin>>n>>q;
25     build_sparse_table(n);
26     while(q--) {
27         int l,r;cin>>l>>r;
28         cout<<query(l,r)<<'\n';
29     }
30 }
31
32 template<class T>
33 struct binary_trie {
34 public:
35     binary_trie() {
36         new_node();
37     }
38
39     void clear() {
40         trie.clear();
41         new_node();
42     }
43
44     void insert(T x) {
45         for(int i = B - 1, p = 0; i >= 0; i--) {
46             int y = x >> i & 1;
47             if(trie[p].go[y] == 0) {
48                 trie[p].go[y] = new_node();
49             }
50             p = trie[p].go[y];
51             trie[p].cnt += 1;
52         }
53     }
54 }

```

4.2 BinaryTrie

```

23 void erase(T x) {
24     for(int i = B - 1, p = 0; i >= 0; i--) {
25         p = trie[p].go[x >> i & 1];
26         trie[p].cnt -= 1;
27     }
28 }
29
30 bool contains(T x) {
31     for(int i = B - 1, p = 0; i >= 0; i--) {
32         p = trie[p].go[x >> i & 1];
33         if(trie[p].cnt == 0) {
34             return false;
35         }
36     }
37     return true;
38 }
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

```

```

89 int new_node() {
90     trie.emplace_back();
91     return (int) trie.size() - 1;
92 }
93 };

```

4.3 BIT

```

1 //迭代
2
3 int n;
4 int bit[100000 + 9];
5 void modify(int i, int x) {
6     while (i <= n) {
7         bit[i] += x;
8         i += i & -i;
9     }
10 }
11 int query(int i) {
12     int res = 0;
13     while (i) {
14         res += bit[i];
15         i -= i & -i;
16     }
17     return res;
18 }
19
20 //bit上二分搜
21 int findk(int k) {
22     int id = 0, res = 0;
23     int mx = __lg(n) + 1;
24     for (int i = mx; i >= 0; i--) {
25         if ((id | (1<<i)) > n) continue;
26         if (res + b[id | (1<<i)] < k) {
27             id = (id | (1<<i));
28             res += b[id];
29         }
30     }
31     return id + 1;
32 }
33
34 //O(n)建bit
35 for (int i = 1; i <= n; ++i) {
36     bit[i] += a[i];
37     int j = i + lowbit(i);
38     if (j <= n) bit[j] += bit[i];
39 }

```

4.4 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     if(!o) o = new node;
13     if(l == r) return o->sum += v, void();
14     int m = (l + r) / 2;
15     if(x <= m) upd(o->l, x, v, l, m);
16     else upd(o->r, x, v, m+1, r);
17     o->pull();
18 }
19
20 ll qry(node* o, int ql, int qr, int l, int r) {
21     if(!o) return 0;
22     if(ql <= l && r <= qr) return o->sum;
23     int m = (l + r) / 2; ll ret = 0;
24     if(ql <= m) ret += qry(o->l, ql, qr, l, m);
25     if(qr > m) ret += qry(o->r, ql, qr, m+1, r);
26     return ret;
27 }

```

4.5 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, 8000004> man, tag, cnt;
14 vector<ooo> Q;
15 bool cmp(ooo a, ooo b) {
16     return a.x < b.x;
17 }
18 void pull(int p) {
19     man[p] = min(man[lc], man[rc]);
20     if(man[lc] < man[rc]) cnt[p] = cnt[lc];
21     else if(man[rc] < man[lc]) cnt[p] = cnt[rc];
22     else cnt[p] = cnt[lc] + cnt[rc];
23 }
24 void push(int p) {
25     man[lc] += tag[p];
26     man[rc] += tag[p];
27     tag[lc] += tag[p];
28     tag[rc] += tag[p];
29     tag[p] = 0;
30 }
31 void build(int p, int l, int r) {
32     if(l == r) {
33         cnt[p] = 1;
34         return;
35     }
36     build(lc, l, mid);

```

```

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

```

```

95     if(tag[idx]) seg[idx] = r-l+1;
96     else seg[idx] = seg[idx<<1] + seg[idx<<1|1];
97 };
98 long long last_pos = 0, ans = 0;
99 for(auto [pos, l, r, val]:tmp) {
100     ans += (pos - last_pos) * seg[1];
101     update(1, r, val, l, r, 1);
102     last_pos = pos;
103 }
104 return ans;
105 }
106
107 // CSES Intersection Points
108 #include <bits/stdc++.h>
109 #define int long long
110 #define pb push_back
111 using namespace std;
112 struct line {
113     int p, l, r;
114 };
115 const int inf = 1e6 + 1;
116 array<int, 2000004> BIT;
117 vector<line> A, Q;
118 bool cmp(line a, line b) {
119     return a.p < b.p;
120 }
121 void update(int p, int x) {
122     for(; p < 2000004; p += p & -p) BIT[p] += x;
123 }
124 int query(int p) {
125     int sum = 0;
126     for(; p; p -= p & -p) sum += BIT[p];
127     return sum;
128 }
129 int run() {
130     int ans = 0, p = 0;
131     for(auto [t, l, r] : Q) {
132         while(p < A.size()) {
133             auto [x, y, v] = A[p];
134             if(x > t) break;
135             update(y, v);
136             p++;
137         }
138         ans += query(r) - query(l - 1);
139     }
140     return ans;
141 }
142 signed main() {
143     int n, x1, x2, y1, y2;
144     cin >> n;
145     for(int i = 0; i < n; i++) {
146         cin >> x1 >> y1 >> x2 >> y2;
147         x1 += inf, x2 += inf, y1 += inf, y2 += inf;
148         if(x1 == x2) Q.pb({x1, y1, y2});
149         else A.pb({x1, y1, 1}), A.pb({x2 + 1, y2, -1});
150     }
151     sort(Q.begin(), Q.end(), cmp);
152     sort(A.begin(), A.end(), cmp);
153     cout << run() << "\n";
154     return 0;
155 }

```

4.6 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 };

```

4.7 DSU

```

1 vector<int> fa;
2 vector<int> sz;
3 void init(int n) {
4     fa.resize(n);
5     sz.resize(n);
6     for (int i = 0; i < n; ++ i) {
7         fa[i] = i;
8         sz[i] = 1; // 一開始樹的大小都是 1
9     }
10 }
11 int find(int x) {
12     // 注意到後面那式變成了 fa[x] = find(fa[x])
13     // 這樣一來，就可以直接將 fa[x] 變成根節點的值
14     return x == fa[x] ? x : fa[x] = find(fa[x]);
15     //Path Compression
16 }
17 void unite(int x, int y) {
18     if (find(x) == find(y)) return;
19     if (sz[find(x)] < sz[find(y)])
20         swap(x, y); // 將 x 換成比較大的那
21         // 邊、y 換成比較小的
22     fa[find(y)] = fa[find(x)];
23     sz[find(x)] += sz[find(y)]; // 記得更新
24     // 合併後的樹的大小
25     //啟發式合併

```

24 | }

4.8 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 inline unsigned sz(Treap *x) {
14     return x ? x->sz:0;
15 }
16
17 inline void Treap::push() {
18     if(!Tag) return ;
19     swap(lc,rc);
20     if(lc) lc->Tag ^= Tag;
21     if(rc) rc->Tag ^= Tag;
22     Tag = false;
23 }
24
25 inline void Treap::pull() {
26     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 *,
55     unsigned K) {
56     Treap *a = nullptr, *b = nullptr;
57     if(!x) return {a,b};
58     x->push();

```

```

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

```

4.9 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 }

```

4.10 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 }

```

4.11 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 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));

```



```

35 if (qr > m) ret += qry(ql, qr, m + 1, r,
36     rc(id));
37 return ret;
}

```

4.12 2D BIT

```

1 //2維BIT
2 #define lowbit(x) (x&-x)
3
4 class BIT {
5     int n;
6     vector<int> bit;
7
8 public:
9     void init(int _n) {
10         n = _n;
11         bit.resize(n);
12         for(auto &b : bit) b = 0;
13     }
14     int query(int x) const {
15         int sum = 0;
16         for(; x; x -= lowbit(x))
17             sum += bit[x];
18         return sum;
19     }
20     void modify(int x, int val) {
21         for(; x <= n; x += lowbit(x))
22             bit[x] += val;
23     }
24 };
25
26 class BIT2D {
27     int m;
28     vector<BIT> bit1D;
29
30 public:
31     void init(int _m, int _n) {
32         m = _m;
33         bit1D.resize(m);
34         for(auto &b : bit1D) b.init(_n);
35     }
36     int query(int x, int y) const {
37         int sum = 0;
38         for(; x; x -= lowbit(x))
39             sum += bit1D[x].query(y);
40         return sum;
41     }
42     void modify(int x, int y, int val) {
43         for(; x <= m; x += lowbit(x))
44             bit1D[x].modify(y, val);
45     }
46 };
47

```

4.13 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 }

```

4.14 Prim

```

1 int cost[MAX_V][MAX_V]; //Edge的權重 (不存在
2     時為INF)
3 int mincost[MAX_V]; //來自集合X的邊的最小權重
4 bool used[MAX_V]; //頂點i是否包含在X之中
5 int V; //頂點數
6
7 int prim() {
8     for(int i = 0; i < V; i++) {
9         mincost[i] = INF;
10        used[i] = false;
11    }
12    mincost[0] = 0;
13    int res = 0;
14    while(true) {
15        int v = -1;
16        //從不屬於X的頂點中尋找會讓來自X的邊
17        //之權重最小的頂點
18        for(int u = 0; u < V; u++) {
19            if(!used[u] && (v == -1 || mincost
20                [u] < mincost[v])) v = u;
21        }
22        if(v == -1) break;
23        used[v] = true; //將頂點v追加至X
24        res += mincost[v]; //加上邊的權重
25        for(int u = 0; u < V; u++) {
26            mincost[u] = min(mincost[u], cost
27                [v][u]);
28        }
29    }
30    return res;
31 }

```

4.15 回滾並查集

```

1 struct dsu_undo{
2     vector<int> sz, p;
3     int comps;
4     dsu_undo(int n){
5         sz.assign(n+5, 1);
6         p.resize(n+5);
7         for(int i = 1; i <= n; ++i) p[i] = i;
8         comps = n;

```

```

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

```

4.16 TimingSegmentTree

```

1 template<class T, class D> struct
2     timing_segment_tree{
3     struct node{
4         int l, r;
5         vector<T> opt;
6     };
7     vector<node> arr;
8     void build(int l, int r, int idx = 1){
9         if(idx==1) arr.resize((r-l+1)<<2);
10        if(l==r){
11            arr[idx].l = arr[idx].r = l;
12            arr[idx].opt.clear();
13            return;
14        }
15        int m = (l+r)>>1;
16        build(l, m, idx<<1);
17        build(m+1, r, idx<<1|1);
18        arr[idx].l = l, arr[idx].r = r;
19        arr[idx].opt.clear();
20    }
21    void update(int ql, int qr, T k, int idx = 1)
22    {
23        if(ql<=arr[idx].l and arr[idx].r<=qr){
24            arr[idx].opt.push_back(k);
25            return;
26        }
27        int m = (arr[idx].l+arr[idx].r)>>1;
28        if(ql<=m) update(ql, qr, k, idx<<1);
29        if(qr>m) update(ql, qr, k, idx<<1|1);
30    }
31    void dfs(D &d, vector<int> &ans, int idx = 1)
32    {
33        if(l==r) {
34            seg[id] = x; // 將a[i]改成x
35            //seg[id] += x; // 將a[i]加上x
36            return;
37        }
38        int mid = (l + r) / 2;

```

```

34 if(arr[idx].l==arr[idx].r) ans[arr[idx].l
35     ] = d.comps;
36 else{
37     dfs(d, ans, idx<<1);
38     dfs(d, ans, idx<<1|1);
39 }
40 while(cnt-->) d.undo();
41 }

```

4.17 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 void modify(int id, int l, int r, int i, int
37     x) {
38     if (l == r) {
39         seg[id] = x; // 將a[i]改成x
40         //seg[id] += x; // 將a[i]加上x
41         return;
42     }
43     int mid = (l + r) / 2;

```

```

38 // 根據修改的點在哪裡，來決定要往哪個子
   樹進行DFS
39 if (i <= mid) modify(id * 2, l, mid, i,
   x); //左
40 else modify(id * 2 + 1, mid + 1, r, i, x
   ); //右
41 seg[id] = seg[id * 2] + seg[id * 2 + 1];
42 }

```

4.18 Hash

```

1 struct custom_hash {
2     static uint64_t splitmix64(uint64_t x) {
3         x += 0x9e3779b97f4a7c15;
4         x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9
5             ;
6         x = (x ^ (x >> 27)) * 0x94d049bb133111eb
7             ;
8         return x ^ (x >> 31);
9     }
10    size_t operator()(uint64_t x) const {
11        static const uint64_t FIXED_RANDOM =
12            chrono::steady_clock::now().
13            time_since_epoch().count();
14        return splitmix64(x + FIXED_RANDOM);
15    }
16    size_t operator()(pair<uint64_t, uint64_t>
17        x) const {
18        static const uint64_t FIXED_RANDOM =
19            chrono::steady_clock::now().
20            time_since_epoch().count();
21        return splitmix64(3*x.first + x.second +
22            FIXED_RANDOM);
23    }
24 };
25 template<class T, class U> using hash_map =
26     gp_hash_table<T, U, custom_hash>;

```

4.19 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);

```

```

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

```

4.20 當作 BST 用的 Treap

```

1 //沒什麼用的BST
2 struct Treap {
3     Treap *lc = nullptr, *rc = nullptr;
4     unsigned pri, sz;
5     int Key;
6     Treap(int Key): pri(rand()), sz(1), Key(Key)
7     {}
8     void pull();
9 } *root;
10
11 inline unsigned sz(Treap *x) {
12     return x ? x->sz : 0;
13 }
14
15 inline void Treap::pull() {
16     sz = 1u + ::sz(lc) + ::sz(rc);
17 }
18
19 inline Treap *merge(Treap *a, Treap *b) {
20     if (!a || !b) return a ? a : b;
21     if (a->pri < b->pri) {
22         a->rc = merge(a->rc, b);
23         a->pull();
24         return a;
25     }
26     else {
27         b->lc = merge(a, b->lc);
28         b->pull();
29         return b;
30     }
31 }
32
33 inline pair<Treap *, Treap *> spilt(Treap *x
34     , int Key) {
35     Treap *a = nullptr, *b = nullptr;
36     if (!x) return {a, b};
37     if (x->Key < Key) {
38         a = x;
39         tie(a->rc, b) = spilt(x->rc, Key);
40     }
41     else {
42         b = x;
43         tie(a, b->lc) = spilt(x->lc, Key);

```

```

44     x->pull();
45     return {a, b};
46 }
47
48 inline pair<Treap *, Treap *> spiltK(Treap *
49     x, unsigned K) {
50     Treap *a = nullptr, *b = nullptr;
51     if (!x) return {a, b};
52     unsigned leftSize = sz(x->lc) + 1;
53     if (K >= leftSize) {
54         a = x;
55         tie(a->rc, b) = spiltK(x->rc, K -
56             leftSize);
57     }
58     else {
59         b = x;
60         tie(a, b->lc) = spiltK(x->lc, K);
61     }
62     x->pull();
63     return {a, b};
64 }
65
66 void Insert(Treap *&root, int Key) {
67     auto [a, b] = spilt(root, Key);
68     root = merge(a, merge(new Treap(Key), b));
69 }
70
71 Treap *&find(Treap *&root, int Key) {
72     if (!root || root->Key == Key) return root;
73     if (Key < root->Key) return find(root->lc,
74         Key);
75     else return find(root->rc, Key);
76 }
77
78 bool erase(Treap *&root, int Key) {
79     Treap *x = find(root, Key);
80     if (!x) return false;
81     Treap *tmp = x;
82     x = merge(x->lc, x->rc);
83     delete tmp;
84     return true;
85 }
86
87 unsigned Rank(Treap *&root, int Key) {
88     auto [a, b] = spilt(root, Key);
89     unsigned ans = sz(a);
90     root = merge(a, b);
91     return ans;
92 }
93
94 int Kth(Treap *&root, unsigned K) {
95     auto [a, b] = spiltK(root, K);
96     auto [c, d] = spiltK(a, K-1);
97     int ans = d->Key;
98     root = merge(merge(c, d), b);
99     return ans;
100 }

```

5 Flow

5.1 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

5.2 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 }

```

5.3 MinCostMaxFlow

```

1 template<class Cap_t, class Cost_t>
2 class MCMF {
3 public:
4     struct Edge {
5         int from;
6         int to;
7         Cap_t cap;
8         Cost_t cost;
9         Edge(int u, int v, Cap_t _cap, Cost_t
10             _cost): from(u), to(v), cap(_cap),
11             cost(_cost) {}
12     };
13
14     static constexpr Cap_t EPS = static_cast<
15         Cap_t>(1e-9);
16
17     int n;
18     vector<Edge> edges;
19     vector<vector<int>> g;
20     vector<Cost_t> d;
21     vector<bool> in_queue;
22     vector<int> previous_edge;
23
24     MCMF() {}

```

```

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];

```

5.4 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                }
36                que.push(v);
37            }
38        }
39        return 0;
40    }
41    T dfs(int u, int t, T f) {
42        if(u == t) {
43            return f;
44        }
45        T r = f;
46        for(int& i = cur[u]; i < (int) g[u].size
            (); ++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);
123         level.resize(n);
124         ptr.resize(n);
125     }
126
127     void add_edge(int v, int u, long long
        cap) {
128         edges.emplace_back(v, u, cap);
129         edges.emplace_back(u, v, 0);
130         adj[v].push_back(m);
131         adj[u].push_back(m + 1);
132         m += 2;
133     }
134
135     bool bfs() {
136         while(!q.empty()) {
137             int v = q.front();
138             q.pop();
139             for (int id : adj[v]) {
140                 if (edges[id].cap - edges[id]
                    .flow < 1)
141                     continue;
142                 if (level[edges[id].u] !=
                    -1)
143                     continue;
144                 level[edges[id].u] = level[v]
                    + 1;
145                 q.push(edges[id].u);
146             }
147         }
148         return level[t] != -1;
149     }
150
151     long long dfs(int v, long long pushed) {
152         if (pushed == 0)
153             return 0;
154         path.push_back(v);
155         if (v == t) {
156             for (int iiddxx = 0; iiddxx <
                pushed; ++iiddxx)
157                 paths.push_back(path);
158             path.pop_back();
159             return pushed;
160         }
161         for (int& cid = ptr[v]; cid < (int)
            adj[v].size(); cid++) {
162             int id = adj[v][cid];
163             int u = edges[id].u;
164             if (level[v] + 1 != level[u] ||
                edges[id].cap - edges[id].
                    flow < 1)
165                 continue;
166             long long tr = dfs(u, min(pushed
                , edges[id].cap - edges[id].

```

```

        flow));
    if (tr == 0)
        continue;
    edges[id].flow += tr;
    edges[id ^ 1].flow -= tr;
    path.pop_back();
    return tr;
}
path.pop_back();
return 0;
}

long long flow() {
    long long f = 0;
    while (true) {
        fill(level.begin(), level.end(), -1);
        level[s] = 0;
        q.push(s);
        if (!bfs())
            break;
        fill(ptr.begin(), ptr.end(), 0);
        while (long long pushed = dfs(s, flow_inf)) {
            f += pushed;
        }
    }
    return f;
}

int main() {
    int n, m, v, u;
    cin >> n >> m;
    Dinic D(n+1, 1, n);
    for (int i = 0; i < m; ++i) {
        cin >> v >> u;
        D.add_edge(v, u, 1);
    }
    D.flow();
    Dinic FLOW(n+1, 1, n);
    for (auto e: D.edges) {
        if (e.flow > 0) {
            FLOW.add_edge(e.v, e.u, 1);
        }
    }
    cout << FLOW.flow() << "\n";
    for (auto p: FLOW.paths) {
        cout << p.size() << "\n";
        for (auto verti: p)
            cout << verti << " ";
        cout << "\n";
    }
    return 0;
}

```

5.5 ISAP with cut

```

1 template<typename T>
2 struct ISAP{
3     static const int MAXN=105;
4     static const T INF=INT_MAX;

```

```

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

```

5.6 biGraph

```

68        if(!vis[e[i].v])cut_e.push_back(i);
69        return ans;
70    }
71    //CSES School Dance
72    //二分圖最大匹配
73    #include <bits/stdc++.h>
74    using namespace std;
75    struct FlowEdge {
76        int v, u;
77        long long cap, flow = 0;
78        FlowEdge(int v, int u, long long cap) :
79            v(v), u(u), cap(cap) {}
80    };
81    struct Dinic {
82        const long long flow_inf = 1e18;
83        vector<FlowEdge> edges;
84        vector<vector<int>> adj;
85        int n, m = 0;
86        int s, t;
87        vector<int> level, ptr, path;
88        vector<vector<int>> paths;
89        queue<int> q;
90        Dinic(int n, int s, int t) : n(n), s(s),
91            t(t) {
92            adj.resize(n);
93            level.resize(n);
94            ptr.resize(n);
95        }
96        void add_edge(int v, int u, long long
97            cap) {
98            edges.emplace_back(v, u, cap);
99            edges.emplace_back(u, v, 0);
100            adj[v].push_back(m);
101            adj[u].push_back(m + 1);
102            m += 2;
103        }
104        bool bfs() {
105            while (!q.empty()) {
106                int v = q.front();
107                q.pop();
108                for (int id : adj[v]) {
109                    if (edges[id].cap - edges[id].flow < 1)
110                        continue;
111                    if (level[edges[id].u] != -1)
112                        continue;
113                    level[edges[id].u] = level[v] + 1;
114                    q.push(edges[id].u);
115                }
116            }
117            return level[t] != -1;
118        }

```

```

119    }
120    long long dfs(int v, long long pushed) {
121        if (pushed == 0)
122            return 0;
123        path.push_back(v);
124        if (v == t) {
125            for (int iiddxx = 0; iiddxx <
126                pushed; ++iiddxx)
127                paths.push_back(path);
128            path.pop_back();
129            return pushed;
130        }
131        for (int& cid = ptr[v]; cid < (int)
132            adj[v].size(); cid++) {
133            int id = adj[v][cid];
134            int u = edges[id].u;
135            if (level[v] + 1 != level[u] ||
136                edges[id].cap - edges[id].flow < 1)
137                continue;
138            long long tr = dfs(u, min(pushed,
139                edges[id].cap - edges[id].flow));
140            if (tr == 0)
141                continue;
142            edges[id].flow += tr;
143            edges[id ^ 1].flow -= tr;
144            path.pop_back();
145            return tr;
146        }
147        path.pop_back();
148        return 0;
149    }
150    long long flow() {
151        long long f = 0;
152        while (true) {
153            fill(level.begin(), level.end(), -1);
154            level[s] = 0;
155            q.push(s);
156            if (!bfs())
157                break;
158            fill(ptr.begin(), ptr.end(), 0);
159            while (long long pushed = dfs(s,
160                flow_inf)) {
161                f += pushed;
162            }
163        }
164        return f;
165    }
166    }
167    int main() {
168        int n, m, v, u;
169        cin >> n >> m;
170        Dinic D(n+1, 1, n);
171        for (int i = 0; i < m; ++i) {
172            cin >> v >> u;
173            D.add_edge(v, u, 1);
174        }
175        D.flow();
176        Dinic FLOW(n+1, 1, n);
177        for (auto e: D.edges) {
178            if (e.flow > 0) {

```



```

109     FLOW.add_edge(e.v, e.u, 1);
110 }
111 }
112 cout << FLOW.flow() << "\n";
113 for (auto p: FLOW.paths) {
114     cout << p.size() << "\n";
115     for (auto verti: p)
116         cout << verti << " ";
117     cout << "\n";
118 }
119 return 0;
120 }
121 }

```

5.7 dinic

```

1 //CSES Downlaod Speed
2 #include <bits/stdc++.h>
3
4 using namespace std;
5
6 struct FlowEdge {
7     int v, u;
8     long long cap, flow = 0;
9     FlowEdge(int v, int u, long long cap) :
10         v(v), u(u), cap(cap) {}
11 };
12
13 struct Dinic {
14     const long long flow_inf = 1e18;
15     vector<FlowEdge> edges;
16     vector<vector<int>> adj;
17     int n, m = 0;
18     int s, t;
19     vector<int> level, ptr;
20     queue<int> q;
21
22     Dinic(int n, int s, int t) : n(n), s(s),
23         t(t) {
24         adj.resize(n);
25         level.resize(n);
26         ptr.resize(n);
27     }
28
29     void add_edge(int v, int u, long long
30         cap) {
31         edges.emplace_back(v, u, cap);
32         edges.emplace_back(u, v, 0);
33         adj[v].push_back(m);
34         adj[u].push_back(m + 1);
35         m += 2;
36     }
37
38     bool bfs() {
39         while (!q.empty()) {
40             int v = q.front();
41             q.pop();
42             for (int id : adj[v]) {
43                 if (edges[id].cap - edges[id].
44                     flow < 1)
45                     continue;
46                 if (level[edges[id].u] !=
47                     -1)
48                     continue;
49                 level[edges[id].u] = level[v] + 1;
50                 edges[id].cap - edges[id].
51                     flow < 1)
52                     continue;
53                 long long tr = dfs(u, min(push
54                     ed, edges[id].cap - edges[id].
55                     flow));
56                 if (tr == 0)
57                     continue;
58                 edges[id].flow += tr;
59                 edges[id ^ 1].flow -= tr;
60                 return tr;
61             }
62             return 0;
63         }
64     }
65
66     long long flow() {
67         long long f = 0;
68         while (true) {
69             fill(level.begin(), level.end(),
70                 -1);
71             level[s] = 0;
72             q.push(s);
73             if (!bfs())
74                 break;
75             fill(ptr.begin(), ptr.end(), 0);
76             while (long long pushed = dfs(s,
77                 flow_inf)) {
78                 f += pushed;
79             }
80             return f;
81         }
82     }
83
84     int main() {
85         int n, m, v, u, w;
86         cin >> n >> m;
87         Dinic D(n + 1, 1, n);
88         for (int i = 0; i < m; ++i) {
89             int u, v >> u >> w;
90             D.add_edge(v, u, w);
91         }
92         cout << D.flow() << "\n";
93         return 0;
94     }
95 }
96 //CSES Police Chase

```

```

43     continue;
44     level[edges[id].u] = level[v]
45         + 1;
46     q.push(edges[id].u);
47 }
48 return level[t] != -1;
49 }
50
51 long long dfs(int v, long long pushed) {
52     if (pushed == 0)
53         return 0;
54     if (v == t)
55         return pushed;
56     for (int& cid = ptr[v]; cid < (int)
57         adj[v].size(); cid++) {
58         int id = adj[v][cid];
59         int u = edges[id].u;
60         if (level[v] + 1 != level[u] ||
61             edges[id].cap - edges[id].
62             flow < 1)
63             continue;
64         long long tr = dfs(u, min(push
65             ed, edges[id].cap - edges[id].
66             flow));
67         if (tr == 0)
68             continue;
69         edges[id].flow += tr;
70         edges[id ^ 1].flow -= tr;
71         return tr;
72     }
73     return 0;
74 }
75
76 long long flow() {
77     long long f = 0;
78     while (true) {
79         fill(level.begin(), level.end(),
80             -1);
81         level[s] = 0;
82         q.push(s);
83         if (!bfs())
84             break;
85         fill(ptr.begin(), ptr.end(), 0);
86         while (long long pushed = dfs(s,
87             flow_inf)) {
88             f += pushed;
89         }
90     }
91     return f;
92 }
93
94 int main() {
95     int n, m, v, u, w;
96     cin >> n >> m;
97     Dinic D(n + 1, 1, n);
98     for (int i = 0; i < m; ++i) {
99         int u, v >> u >> w;
100         D.add_edge(v, u, w);
101     }
102     cout << D.flow() << "\n";
103     return 0;
104 }
105 //CSES Police Chase

```

```

101 //Min Cut
102 #include <bits/stdc++.h>
103 #define pb push_back
104 using namespace std;
105 struct pipe{
106     int u, v, f;
107 };
108 const int inf = 1e3;
109 int t;
110 array<int, 504> lvl, P, vis;
111 array<vector<int>, 504> G;
112 vector<pipe> E;
113 int bfs(int s){
114     int u, v;
115     queue<int> Q;
116     Q.push(s);
117     lvl[s] = 1;
118     while(!Q.empty()){
119         u = Q.front();
120         Q.pop();
121         for(int i : G[u]){
122             v = E[i].v;
123             if(lvl[v] || !E[i].f) continue;
124             lvl[v] = lvl[u] + 1;
125             Q.push(v);
126         }
127     }
128     return lvl[t];
129 }
130 int dfs(int u, int f){
131     if(u == t || !f) return f;
132     int wut, ans = 0;
133     for(int &i = P[u]; i < G[u].size(); i++){
134         pipe &e = E[G[u][i]], &b = E[G[u][i]
135             ^ 1];
136         if(lvl[e.v] == lvl[u] + 1){
137             wut = dfs(e.v, min(e.f, f));
138             e.f -= wut;
139             b.f += wut;
140             f -= wut;
141             ans += wut;
142         }
143     }
144     return ans;
145 }
146 int dinic(int s){
147     int ans = 0, tmp;
148     while(1){
149         for(int &l : lvl) l = 0;
150         if(!bfs(s)) break;
151         while(1){
152             for(int &p : P) p = 0;
153             if(tmp = dfs(s, inf)) ans += tmp;
154             else break;
155         }
156     }
157     return ans;
158 }
159 void go(int u, int c){
160     if(vis[u]) return;
161     vis[u] = c;
162     for(int i : G[u]){
163         if(!E[i].f) continue;
164         go(E[i].v, c);
165     }
166 }

```

```

164 }
165 }
166 signed main(){
167     int n, m, a, b, cnt = 0;
168     cin >> n >> m;
169     while(m--){
170         cin >> a >> b;
171         G[a].pb(cnt++);
172         G[b].pb(cnt++);
173         E.pb({a, b, 1});
174         E.pb({b, a, 1});
175     }
176     t = n;
177     cout << dinic(1) << "\n";
178     go(1, 1);
179     go(t, 2);
180     for(pipe e : E){
181         if(e.f) continue;
182         if(vis[e.u] && vis[e.v] && vis[e.u]
183             != vis[e.v]){
184             cout << e.u << " " << e.v << "\n
185                 ";
186         }
187     }
188     return 0;
189 }

```

6 Graph

6.1 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     };
21     for(int i = 0; i < n; ++i) {
22         if(id[i] == -1) dfs(i, -1);
23     }
24     return bridges;
25 }

```

6.2 SPFA

```

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

```

6.3 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 }

```

6.4 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
    &G) {
5     size_t N = G.size(), Change = 0;
6     do {
7         Change = 0;
8         for(size_t u = 0; u < N; ++u) {
9             vector<size_t> E;
10            for(size_t v = 0; v < N && E.size() <
                3; ++v)
11                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 }
25 //計算Degree
26 //O(n^2)
27 vector<size_t> getDegree(const
    AdjacencyMatrixTy &G) {
28     size_t N = G.size();
29     vector<size_t> Degree(N);
30     for(size_t u = 0; u < N; ++u)
31         for(size_t v = u + 1; v < N; ++v) {
32             if(!G[u][v]) continue;
33             ++Degree[u], ++Degree[v];
34         }
35     return Degree;
36 }
37 }
38 //判斷是否為K5 or K33
39 //O(n)

```

```

41 bool is_K5_or_K33(const vector<size_t> &
    Degree) {
42     unordered_map<size_t, size_t> Num;
43     for(auto Val : Degree) ++Num[Val];
44     size_t N = Degree.size();
45     bool isK5 = Num[4] == 5 && Num[4] + Num[0]
        == N;
46     bool isK33 = Num[3] == 6 && Num[3] + Num
        [0] == N;
47     return isK5 || isK33;
48 }

```

6.5 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.
            resize(n+5),vis.resize(n+5),is_ap.
            resize(n+5),bcc_id.resize(n+5);
9     }
10    inline void build(const vector<vector<int>
        >>&g,int u,int p = -1){
11        int child = 0;
12        dfn[u] = low[u] = ++dfn_cnt;
13        st.push(u);
14        vis[u] = 1;
15        if(g[u].empty() and p==-1){
16            bcc_id[u] = ++bcc_cnt;
17            bcc[bcc_cnt].push_back(u);
18            return;
19        }
20        for(auto v:g[u]){
21            if(v==p)continue;
22            if(!dfn[v]){
23                build(g,v,u);
24                child++;
25                if(dfn[u]<=low[v]){
26                    is_ap[u] = 1;
27                    bcc_id[u] = ++bcc_cnt;
28                    bcc[bcc_cnt].push_back(u);
29                    while(vis[v]){
30                        bcc_id[st.top()] = bcc_cnt;
31                        bcc[bcc_cnt].push_back(st.top());
32                        ;
33                        vis[st.top()] = 0;
34                        st.pop();
35                    }
36                    low[u] = min(low[u],low[v]);
37                }
38                low[u] = min(low[u],dfn[v]);
39            }
40            if(p==-1 and child<2)is_ap[u] = 0;
41            if(is_ap[u])ap.push_back(u);
42        }
43    };

```

6.6 枚舉極大團 Bron-Kerbosch

```

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

```

```

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

```

6.7 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 }

```

6.8 Dominator tree

```

1 struct dominator_tree{
2     static const int MAXN=5005;
3     int n; // 1-base
4     vector<int> G[MAXN], rG[MAXN];
5     int pa[MAXN], dfn[MAXN], id[MAXN], dfnCnt;
6     int semi[MAXN], idom[MAXN], best[MAXN];
7     vector<int> tree[MAXN]; // tree here
8     void init(int _n){
9         n = _n;
10        for(int i=1; i<=n; ++i)
11            G[i].clear(), rG[i].clear();
12    }
13    void add_edge(int u, int v){
14        G[u].push_back(v);
15        rG[v].push_back(u);
16    }
17    void dfs(int u){
18        id[dfn[u]=++dfnCnt]=u;
19        for(auto v:G[u]) if(!dfn[v])
20            dfs(v), pa[dfn[v]]=dfn[u];
21    }
22    int find(int y, int x){
23        if(y <= x) return y;
24        int tmp = find(pa[y], x);
25        if(semi[best[y]] > semi[best[pa[y]]])
26            best[y] = best[pa[y]];
27        return pa[y] = tmp;
28    }
29    void tarjan(int root){
30        dfnCnt = 0;
31        for(int i=1; i<=n; ++i){
32            dfn[i] = idom[i] = 0;
33            tree[i].clear();
34            best[i] = semi[i] = i;
35        }
36        dfs(root);
37        for(int i=dfnCnt; i>1; --i){
38            int u = id[i];
39            for(auto v:rG[u]) if(v=dfn[v]){

```

```

        find(v, i);
        semi[i]=min(semi[i], semi[best[v]]);
    }
    tree[semi[i]].push_back(i);
    for(auto v:tree[pa[i]]){
        find(v, pa[i]);
        idom[v] = semi[best[v]]==pa[i]
            ? pa[i] : best[v];
    }
    tree[pa[i]].clear();
}
for(int i=2; i<=dfnCnt; ++i){
    if(idom[i] != semi[i])
        idom[i] = idom[idom[i]];
    tree[id[idom[i]]].push_back(id[i]);
}
}
} dom;

```

6.9 判斷二分圖

```

1 vector<int> G[MAXN];
2 int color[MAXN]; // -1: not colored, 0:
3     black, 1: white
4 /* color the connected component where u is
5     */
6 /* parameter col: the color u should be
7     colored */
8 bool coloring(int u, int col) {
9     if(color[u] != -1) {
10        if(color[u] != col) return false;
11        return true;
12    }
13     color[u] = col;
14     for(int v : G[u])
15         if(!coloring(v, col ^ 1))
16             return false;
17     return true;
18 }
19 //check if a graph is a bipartite graph
20 bool checkBipartiteG(int n) {
21     for(int i = 1; i <= n; i++)
22         color[i] = -1;
23     for(int i = 1; i <= n; i++)
24         if(color[i] == -1 &&
25            !coloring(i, 0))
26             return false;
27     return true;
28 }

```

6.10 Bellman Ford

```

1 vector<tuple<int,int,int>> Edges;
2 int BellmanFord(int s, int e, int N) {
3     const int INF = INT_MAX / 2;
4     vector<int> dist(N, INF);
5

```

```

6     dist[s] = 0;
7     bool update;
8     for(int i=1; i<=N; ++i) {
9         update = false;
10        for(auto [v, u, w] : Edges)
11            if (dist[u] > dist[v] + w)
12                {
13                    dist[u] = dist[v] + w;
14                    update = true;
15                }
16    }
17    if (!update)
18        break;
19    if (i == N) // && update
20        return -1; // gg !
21 }
22 return dist[e];
23 }
24 }
25 }

```

6.11 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()) {
15        tie(std::ignore, s) = pq.top();
16        pq.pop();
17
18        if (used[s]) continue;
19        used[s] = true; // 每一個點都只看一
20            次
21
22        for (auto [e, w] : V[s]) {
23            if (dist[e] > dist[s] + w) {
24                dist[e] = dist[s] + w;
25                pq.emplace(dist[e], e);
26            }
27        }
28    }
29    return dist[e];
30 }

```

6.12 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    }
12    inline void add_edge(int u, int v){
13        g[u].push_back(v);
14    }
15    inline void build(){
16        function<void(int, int)>dfs = [&](int u,
17            int dis){
18            low[u] = dfn[u] = ++dfn_cnt, vis[u] =
19                1;
20            st.push(u);
21            for(auto v:g[u]){
22                if(!dfn[v]){
23                    dfs(v, dis+1);
24                    low[u] = min(low[u], low[v]);
25                }
26                else if(vis[v]){
27                    low[u] = min(low[u], dfn[v]);
28                }
29            }
30            if(low[u]==dfn[u]){
31                ++cnt;
32                while(vis[u]){
33                    auto v = st.top();
34                    st.pop();
35                    vis[v] = 0;
36                    scc[v] = cnt;
37                    sz[cnt]++;
38                }
39            }
40            vector<vector<int>> compress(){
41                vector<vector<int>>ans(cnt+1);
42                for(int u = 0; u<=n; ++u){
43                    for(auto v:g[u]){
44                        if(scc[u] == scc[v]){
45                            continue;
46                        }
47                        ans[scc[u]].push_back(scc[v]);
48                    }
49                }
50            }
51            for(int i = 0; i<=cnt; ++i){
52                sort(ans[i].begin(), ans[i].end());
53                ans[i].erase(unique(ans[i].begin(),
54                    ans[i].end()), ans[i].end());
55            }
56            return ans;
57        }
58    }
59 }
60 };

```

6.13 判斷環

```

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

```

6.14 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    };

```

7 Math

7.1 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};

```

7.2 FastPow

```

1 constexpr long long Pow(long long x, long
2     long n, int m) {
3     if(m == 1) return 0;
4     unsigned int _m = (unsigned int)(m);
5     unsigned long long r = 1;
6     x %= m;
7     if(x < 0) x += m;
8     unsigned long long y = x;
9     while(n) {
10        if(n & 1) r = (r * y) % _m;
11        y = (y * y) % _m;
12        n >>= 1;
13    }
14    return r;

```

7.3 中國剩餘定理

```

1 // #include "InvGCD.h"
2 // @return
3 // $\text{remainder, modulo}$
4 // or
5 // $0, 0$ if do not exist
6 pair<long long, long long> crt(const vector<
7     long long>& r, const vector<long long>&
8     m) {
9     assert(r.size()==m.size());
10    int n = r.size();
11    // Contracts: 0 <= r0 < m0
12    long long r0 = 0, m0 = 1;
13    for(int i = 0; i < n; i++) {
14        assert(1 <= m[i]);
15        long long r1 = r[i] % m[i];

```

```

14    if(r1 < 0) r1 += m[i];
15    long long m1 = m[i];
16    if(m0 < m1) {
17        swap(r0, r1);
18        swap(m0, m1);
19    }
20    if(m0 % m1 == 0) {
21        if(r0 % m1 != r1) return {0, 0};
22        continue;
23    }
24    long long g, im;
25    tie(g, im) = inv_gcd(m0, m1);
26    long long u1 = (m1 / g);
27    if((r1 - r0) % g) return {0, 0};
28    long long x = (r1 - r0) / g % u1 * im %
29        u1;
30    r0 += x * m0;
31    m0 *= u1;
32    if(r0 < 0) r0 += m0;
33    return {r0, m0};
34 }

```

7.4 ExtendGCD

```

1 // @return $x, y$ s.t. $ax + by = \gcd(a, b)$
2 $
3 #define ll long long
4 ll ext_gcd(ll a, ll b, ll& x, ll& y) {
5     if(b == 0) {
6         x = 1; y = 0;
7         return a;
8     }
9     ll x2, y2;
10    ll c = a % b;
11    if(c < 0) c += b;
12    ll g = ext_gcd(b, c, x2, y2);
13    x = y2;
14    y = x2 - (a / b) * y2;
15    return g;
16 }
17 //a^{-1} % p = x % p

```

7.5 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 signed main() {
25     cin>>n;
26     sieve((int)1e6);
27     map<int,int> divisor;
28     while(n--) {
29         divisor.clear();
30         int x; cin>>x;
31         while(x>1) {
32             divisor[LPS[x]]++;
33             x/=LPS[x];
34         }
35         int ans = 1;
36         for(auto &[x,y] : divisor) ans *= (y
37             +1);
38         cout<<ans;
39         cout<<"\n";
40     }
41 }

```

7.6 Theorem

- Modular Arithmetic

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

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

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

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

- Cramer's rule

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

- Kirchhoff's Theorem

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

- The number of undirected spanning in G is $|\det(\tilde{L}_{11})|$.

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

• Tutte's Matrix

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

• Cayley's Formula

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

• Erdős–Gallai theorem

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

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

• Gale–Ryser theorem

A pair of sequences of nonnegative integers $a_1 \geq \dots \geq a_n$ and b_1, \dots, b_n is bigraphic if and only if $\sum_{i=1}^n a_i = \sum_{i=1}^n b_i$ and $\sum_{i=1}^k a_i \leq \sum_{i=1}^k \min(b_i, k)$ holds for every $1 \leq k \leq n$.

• Fulkerson–Chen–Anstee theorem

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

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

• Möbius inversion formula

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

• Spherical cap

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

7.7 FloorSum

```
1 //f(a, b, c, n) = \sum_{i=0}^{n-1} \lfloor \frac{ai+b}{c} \rfloor
2 long long floor_sum(long long a, long long b
3   , long long c, long long n) {
4   long long ans = 0;
5   if(a >= c) {
6     ans += (n - 1) * n * (a / c) / 2;
7     a %= c;
8   }
9   if(b >= c) {
10    ans += n * (b / c);
11    b %= c;
12  }
13  long long y_max = (a * n + b) / c;
14  long long x_max = y_max * c - b;
15  if(y_max == 0) {
16    return ans;
17  }
18  ans += (n - (x_max + a - 1) / a) * y_max;
19  return ans + floor_sum(c, (a - x_max % a)
20    % a, a, y_max);
21 }
```

7.8 Numbers

• Bernoulli numbers

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

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

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

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

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

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

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

• Pentagonal number theorem

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

• Catalan numbers

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

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

• Eulerian numbers

Number of permutations $\pi \in S_n$ in which exactly k elements are greater than the previous element. k : j 's s.t. $\pi(j) > \pi(j+1)$, $k+1$: j 's s.t. $\pi(j) \geq j$, k : j 's s.t. $\pi(j) > j$.

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

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

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

7.9 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 }
```

7.10 ModInt

```
1 template<int id>
2 struct modint {
3 public:
4   static constexpr int mod() { return id; }
5
6   constexpr modint() : value(0) {}
7   modint(long long x) : value(x % mod()) {
8     if(value < 0) value += mod();
9   }
```

```
10   constexpr int val() const { return value; }
11   }
12   constexpr modint inv() const {
13     return Pow(value, mod()-2, mod());
14   }
15   }
16   constexpr modint& operator+=(const modint&
17     rhs) & {
18     value += rhs.value;
19     if(value >= mod()) {
20       value -= mod();
21     }
22     return *this;
23   }
24   }
25   constexpr modint& operator-=(const modint&
26     rhs) & {
27     value -= rhs.value;
28     if(value < 0) {
29       value += mod();
30     }
31     return *this;
32   }
33   }
34   constexpr modint& operator*=(const modint&
35     rhs) & {
36     value = 1LL * value * rhs.value % mod();
37     return *this;
38   }
39   }
40   constexpr modint& operator/=(const modint&
41     rhs) & {
42     return *this *= rhs.inv();
43   }
44   }
45   friend constexpr modint operator+(modint
46     lhs, modint rhs) { return lhs += rhs; }
47   friend constexpr modint operator-(modint
48     lhs, modint rhs) { return lhs -= rhs; }
49   friend constexpr modint operator*(modint
50     lhs, modint rhs) { return lhs *= rhs; }
51   friend constexpr modint operator/(modint
52     lhs, modint rhs) { return lhs /= rhs; }
53   }
54   constexpr modint operator+() const {
55     return *this; }
56   constexpr modint operator-() const {
57     return modint() - *this; }
58   constexpr bool operator==(const modint&
59     rhs) const { return value == rhs.value; }
60   }
61   constexpr bool operator!=(const modint&
62     rhs) const { return value != rhs.value; }
63   }
64   int value;
```

7.11 Generating Functions

- Ordinary Generating Function $A(x) = \sum_{i \geq 0} a_i x^i$

$$\begin{aligned} - A(rx) &\Rightarrow r^n a_n \\ - A(x) + B(x) &\Rightarrow a_n + b_n \\ - A(x)B(x) &\Rightarrow \sum_{i=0}^n a_i b_{n-i} \\ - A(x)^k &\Rightarrow \sum_{i_1+i_2+\dots+i_k=n} a_{i_1} a_{i_2} \dots a_{i_k} \\ - xA(x)' &\Rightarrow na_n \\ - \frac{A(x)}{1-x} &\Rightarrow \sum_{i=0}^n a_i \end{aligned}$$

- Exponential Generating Function $A(x) = \sum_{i \geq 0} \frac{a_i}{i!} x^i$

$$\begin{aligned} - A(x) + B(x) &\Rightarrow a_n + b_n \\ - A^{(k)}(x) &\Rightarrow a_n + b_n \\ - A(x)B(x) &\Rightarrow \sum_{i=0}^n \binom{n}{i} a_i b_{n-i} \\ - A(x)^k &\Rightarrow \sum_{i_1+i_2+\dots+i_k=n} \binom{n}{i_1, i_2, \dots, i_k} a_{i_1} a_{i_2} \dots a_{i_k} \\ - xA(x) &\Rightarrow na_n \end{aligned}$$

- Special Generating Function

$$\begin{aligned} - (1+x)^n &= \sum_{i \geq 0} \binom{n}{i} x^i \\ - \frac{1}{(1-x)^n} &= \sum_{i \geq 0} \binom{n-1}{i} x^i \end{aligned}$$

```

35 }
36
37 vector<int> multiply(const vector<int>& a,
38     const vector<int>& b) {
39     vector<cd> fa(a.begin(), a.end());
40     vector<cd> fb(b.begin(), b.end());
41     int n = 1;
42     while(n < (int) a.size() + (int) b.size()
43         - 1) {
44         n <= 1;
45     }
46     fa.resize(n);
47     fb.resize(n);
48     FFT(fa, false);
49     FFT(fb, false);
50     for(int i = 0; i < n; ++i) {
51         fa[i] *= fb[i];
52     }
53     FFT(fa, true);
54     vector<int> c(a.size() + b.size() - 1);
55     for(int i = 0; i < (int) c.size(); ++i) {
56         c[i] = round(fa[i].real());
57     }
58     return c;
59 }

```

```

92 for(int i = 1; i <= q; i++){
93     cin >> l >> r;
94     Q.pb({l, r, i});
95     ans[i] = 1;
96 }
97 find(n);
98 for(int i = 1; i <= q; i++) cout << ans[
99     i] << "\n";
100 return 0;

```

7.12 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 }

```

8 RMQ

8.1 Doubling RMQ

```

1 //Movie Festival Queries
2 #include <bits/stdc++.h>
3 using namespace std;
4 array<array<int, 24>, 100004> W;
5 void build(int n){
6     for(int i = 1; i <= n; i++){
7         W[i][0] = max(W[i][0], W[i - 1][0]);
8     }
9     for(int j = 1; j < 20; j++){
10        for(int i = 1; i <= n; i++){
11            W[i][j] = max(W[i][j - 1], W[i - 1][j]);
12        }
13    }
14 }
15 int query(int l, int r){
16     int ans = 0;
17     for(int i = 19; i >= 0; i--){
18         if(W[r][i] >= l){
19             ans += 1 << i;
20             r = W[r][i];
21         }
22     }
23     return ans;
24 }
25 signed main(){
26     int n, q, l, r;
27     cin >> n >> q;
28     while(n--){
29         cin >> l >> r;
30         W[r][0] = max(l, W[r][0]);
31     }

```

```

32 build(1e6);
33 while(q--){
34     cin >> l >> r;
35     cout << query(l, r) << "\n";
36 }
37 return 0;
38 }
39
40 //Missing Coin Sum Queries
41 #include <bits/stdc++.h>
42 #define pb push_back
43 #define int long long
44 using namespace std;
45 struct ooo{
46     int l, r, t;
47 };
48 array<int, 200004> X, ans, P;
49 array<int, 400004> S;
50 array<array<int, 200004>, 20> T;
51 vector<ooo> Q;
52 int highbit(int x){
53     int p = 0;
54     for(int i = 4; i >= 0; i--){
55         if(1 << (p + (1 << i)) <= x) p += (1
56             << i);
57     }
58     return p;
59 }
60 void build(int n){
61     for(int i = 1; 1 << i <= n; i++){
62         for(int j = 1; j <= n; j++){
63             T[i][j] = min(T[i - 1][j], T[i -
64                 1][j + (1 << (i - 1))]);
65         }
66     }
67 }
68 void find(int n){
69     int h;
70     for(int k = 1, p = 1; k < 1 << 30; k <=
71         1, p = 1){
72         S[1] = 0, T[0][1] = 111 << 60;
73         for(int i = 1; i <= n; i++){
74             if(X[i] >= k && X[i] < k << 1){
75                 P[i] = ++p;
76                 S[p] = X[i] + S[p - 1];
77                 T[0][p] = X[i];
78                 p++;
79             } else P[i] = p;
80             S[p] = S[p - 1], T[0][p] = 111
81                 << 60;
82         }
83     }
84     build(p);
85     for(auto [l, r, t] : Q){
86         l = P[l], r = P[r], h = highbit(
87             r - l + 1);
88         if(ans[t] >= min(T[h][l], T[h][r
89             + 1 - (1 << h)])){
90             ans[t] += S[r] - S[l - 1];
91         }
92     }
93 }
94 signed main(){
95     int n, q, l, r;
96     cin >> n >> q;
97     for(int i = 1; i <= n; i++) cin >> X[i];

```

8.2 SegTree

```

1 //Hotel Queries (線段樹上二分搜)
2 #include <bits/stdc++.h>
3 #define mid ((l + r) >> 1)
4 #define lc (p << 1)
5 #define rc ((p << 1) | 1)
6 using namespace std;
7 array<int, 800004> S;
8 void pull(int p){
9     S[p] = max(S[lc], S[rc]);
10 }
11 void update(int p, int c, int x, int l, int
12     r){
13     if(c < l || c > r) return;
14     if(l == r){
15         S[p] += x;
16         return;
17     }
18     update(lc, c, x, l, mid);
19     update(rc, c, x, mid + 1, r);
20     pull(p);
21 }
22 int query(int p, int x, int l, int r){
23     if(S[p] < x) return 0;
24     if(l == r) return l;
25     if(S[lc] >= x) return query(lc, x, l,
26         mid);
27     else return query(rc, x, mid + 1, r);
28 }
29 signed main(){
30     int n, m, h, r, p;
31     cin >> n >> m;
32     for(int i = 1; i <= n; i++){
33         cin >> h;
34         update(1, i, h, 1, n);
35     }
36     while(m--){
37         cin >> r;
38         p = query(1, r, 1, n);
39         cout << p << " ";
40         if(p) update(1, p, -r, 1, n);
41     }
42     return 0;

```

8.3 Treap

```

1 //CSES Reversals and Sums
2 #include <bits/stdc++.h>
3 using namespace std;
4 #define ll long long
5
6 struct Treap {
7     Treap *lc = nullptr, *rc = nullptr;
8     int pri, sz;
9     ll Val, Sum;
10    bool Tag;
11    Treap(ll Val): pri(rand()),sz(1),Val(Val),
12        Sum(Val),Tag(false) {}
13    void pull();
14    void push();
15 } *root;
16
17 inline int sz(Treap *x) {
18     return x ? x->sz:0;
19 }
20
21 inline void Treap::push() {
22     if(!Tag) return;
23     swap(lc,rc);
24     if(lc) lc->Tag ^= Tag;
25     if(rc) rc->Tag ^= Tag;
26     Tag = false;
27 }
28
29 inline void Treap::pull() {
30     sz = 1;
31     Sum = Val;
32     if(lc) {
33         sz += lc->sz;
34         Sum += lc->Sum;
35     }
36     if(rc) {
37         sz += rc->sz;
38         Sum += rc->Sum;
39     }
40 }
41
42 Treap *merge(Treap *a, Treap *b) {
43     if(!a || !b) return a ? a : b;
44     if(a->pri < b->pri) {
45         a->push();
46         a->rc = merge(a->rc,b);
47         a->pull();
48         return a;
49     }
50     else {
51         b->push();
52         b->lc = merge(a,b->lc);
53         b->pull();
54         return b;
55     }
56 }
57
58 pair<Treap *, Treap *> splitK(Treap *x, int
59 K) {
60     Treap *a = nullptr, *b = nullptr;
61     x->push();
62     int leftSize = sz(x->lc) + 1;
63     if(K >= leftSize) {
64         a = x;
65         tie(a->rc,b) = splitK(x->rc, K -
66             leftSize);
67     }
68     else {
69         b = x;
70         tie(a, b->lc) = splitK(x->lc,K);
71     }
72     x->pull();
73     return {a,b};
74 }
75
76 ll query(Treap *&root, int ql, int qr) {
77     auto [a,b] = splitK(root, ql-1);
78     auto [c,d] = splitK(b,qr-ql+1);
79     c->push();
80     ll Sum = c->Sum;
81     root = merge(a,merge(c,d));
82     return Sum;
83 }
84
85 void Reverse(Treap *&root, int ql, int qr) {
86     auto [a,b] = splitK(root,ql-1);
87     auto [c,d] = splitK(b,qr-ql+1);
88     c->Tag ^= true;
89     root = merge(a,merge(c,d));
90 }
91
92 signed main() {
93     int n,q;cin>>n>>q;
94     for(int i=0,x;i<n;i++) {
95         cin>>x;
96         root = merge(root,new Treap(x));
97     }
98     while(q--) {
99         int cmd,l,r;cin>>cmd>>l>>r;
100         if(cmd == 1) Reverse(root,l,r);
101         else cout<<query(root,l,r)<<'\n';
102     }
103 }
104
105 //Reversal Sorting
106 #include <bits/stdc++.h>
107 #define pb push_back
108 using namespace std;
109 const int inf = 1e9;
110 struct treap{
111     int val, man, siz, pri;
112     bool rev;
113     treap *lc, *rc;
114     treap(int v){
115         val = man = v;
116         siz = 1;
117         pri = rand();
118         rev = 0;
119         lc = rc = nullptr;
120     }
121     void pull(){
122         man = min(min(lc? lc->man : inf, rc?
123             rc->man : inf), val);
124         siz = (lc? lc->siz : 0) + (rc? rc->
125             siz : 0) + 1;
126     }
127     void push(){
128         if(!rev) return;
129         swap(lc, rc);
130         if(lc) lc->rev ^= 1;
131         if(rc) rc->rev ^= 1;
132         rev = 0;
133     }
134     int find(int k){
135         push();
136         int ls = (lc? lc->siz : 0) + 1;
137         if(val == k) return ls;
138         if(lc && lc->man == k) return lc->
139             find(k);
140         else return rc->find(k) + ls;
141     }
142 }
143
144 vector<pair<int, int>> ans;
145 int size(treap *t){
146     return t? t->siz : 0;
147 }
148
149 treap* merge(treap *a, treap *b){
150     if(!a || !b) return a ? a : b;
151     if(a->pri > b->pri){
152         a->push();
153         a->rc = merge(a->rc, b);
154         a->pull();
155         return a;
156     }
157     else{
158         b->push();
159         b->lc = merge(a, b->lc);
160         b->pull();
161         return b;
162     }
163 }
164
165 void split(treap *t, treap *&a, treap *&b,
166 int k){
167     if(!t){
168         a = b = nullptr;
169         return;
170     }
171     t->push();
172     if(k <= size(t->lc)){
173         b = t;
174         split(t->lc, a, b->lc, k);
175         b->pull();
176     }
177     else{
178         a = t;
179         split(t->rc, a->rc, b, k - size(t->
180             lc) - 1);
181         a->pull();
182     }
183 }
184
185 signed main(){
186     srand(time(NULL));
187     int n, x, p;
188     treap *t = nullptr, *a = nullptr, *b =
189         nullptr, *c = nullptr;
190     cin >> n;
191     for(int i = 1; i <= n; i++){
192         cin >> x;
193         t = merge(t, new treap(x));
194     }
195     for(int i = 1; i <= n; i++){
196         split(t, a, b, i - 1);
197         p = b->find(i);
198         if(p == 1){
199             t = merge(a, b);
200             continue;
201         }
202         ans.pb({i, i + p - 1});
203         split(b, b, c, p);
204         b->rev ^= 1;
205         t = merge(a, merge(b, c));
206     }
207     cout << ans.size() << "\n";
208     for(auto [l, r] : ans) cout << l << " "
209         << r << "\n";
210     return 0;
211 }

```

9 Square root decomposition

9.1 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            while(l<ql)sub(l++);
40            while(r>qr)sub(r--);
41            ans[id] = {mx,cntcnt[mx]};
42        }
43    }
44 };

```

10 Tree

10.1 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 }

```

10.2 HLD

```

1 struct heavy_light_decomposition{
2     int n;
3     vector<int> dep, father, sz, mxson, topf, id;
4     vector<vector<int>>> g;
5     heavy_light_decomposition(int _n = 0) : n(_n) {
6         g.resize(n+5);
7         dep.resize(n+5);
8         father.resize(n+5);
9         sz.resize(n+5);
10        mxson.resize(n+5);
11        topf.resize(n+5);
12        id.resize(n+5);
13    }
14    void add_edge(int u, int v){
15        g[u].push_back(v);
16        g[v].push_back(u);
17    }
18    void dfs(int u, int p){
19        dep[u] = dep[p]+1;
20        father[u] = p;
21        sz[u] = 1;
22        mxson[u] = 0;
23        for(auto v:g[u]){
24            if(v==p) continue;
25            dfs(v, u);
26            sz[u] += sz[v];
27            if(sz[v] > sz[mxson[u]]) mxson[u] = v;
28        }
29    }
30    void dfs2(int u, int top){
31        static int idn = 0;
32        topf[u] = top;
33        id[u] = ++idn;
34        if(mxson[u]) dfs2(mxson[u], top);

```

```

35        for(auto v:g[u]){
36            if(v!=father[u] and v!=mxson[u]){
37                dfs2(v, v);
38            }
39        }
40    }
41    void build(int root){
42        dfs(root, 0);
43        dfs2(root, root);
44    }
45    vector<pair<int, int>> path(int u, int v){
46        vector<pair<int, int>> ans;
47        while(topf[u] != topf[v]){
48            if(dep[topf[u]] < dep[topf[v]]) swap(u, v);
49            ans.push_back({id[topf[u]], id[u]});
50            u = father[topf[u]];
51        }
52        if(id[u] > id[v]) swap(u, v);
53        ans.push_back({id[u], id[v]});
54        return ans;
55    }
56 }

```

10.3 POJ Tree

```

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

```

```

35     pair<int, int> res(INT_MAX, -1);
36     int ma=0;
37     for(size_t i=0; i<g[u].size(); ++i){
38         int v=g[u][i].first;
39         if(v==pa || vis[v]) continue;
40         res=min(res, tree_centroid(v, u, sz));
41         size[u] += size[v];
42         ma=max(ma, size[v]);
43     }
44     ma=max(ma, sz-size[u]);
45     return min(res, make_pair(ma, u));
46 }
47 int tree_DC(int u, int sz){
48     int center=tree_centroid(u, -1, sz).second;
49     int ans=cal(center, 0);
50     vis[center]=1;
51     for(size_t i=0; i<g[center].size(); ++i){
52         int v=g[center][i].first, w=g[center][i].second;
53         if(vis[v]) continue;
54         ans+=cal(v, w);
55         ans+=tree_DC(v, size[v]);
56     }
57     return ans;
58 }
59 int main(){
60     while(scanf("%d%d", &n, &k), n || k){
61         init();
62         for(int i=1; i<n; ++i){
63             int u, v, w;
64             scanf("%d%d%d", &u, &v, &w);
65             g[u].push_back(make_pair(v, w));
66             g[v].push_back(make_pair(u, w));
67         }
68         printf("%d\n", tree_DC(1, n));
69     }
70     return 0;
71 }

```

10.4 HeavyLight

```

1 #include<vector>
2 #define MAXN 100005
3 int siz[MAXN], max_son[MAXN], pa[MAXN], dep[MAXN];
4 int link_top[MAXN], link[MAXN], cnt;
5 vector<int> G[MAXN];
6 void find_max_son(int u){
7     siz[u]=1;
8     max_son[u]=-1;
9     for(auto v:G[u]){
10        if(v==pa[u]) continue;
11        pa[v]=u;
12        dep[v]=dep[u]+1;
13        find_max_son(v);
14        if(max_son[u]==-1 || siz[v]>siz[max_son[u]]) max_son[u]=v;
15        siz[u] += siz[v];
16    }
17 }
18 void build_link(int u, int top){
19     link[u] = ++cnt;
20     link_top[u] = top;

```

```

21     if(max_son[u]==-1) return;
22     build_link(max_son[u], top);
23     for(auto v:G[u]){
24         if(v==max_son[u] || v==pa[u]) continue;
25         build_link(v, v);
26     }
27 }
28 int find_lca(int a, int b){
29     //求LCA · 可以在過程中對區間進行處理
30     int ta=link_top[a], tb=link_top[b];
31     while(ta!=tb){
32         if(dep[ta]<dep[tb]){
33             swap(ta, tb);
34             swap(a, b);
35         }
36         //這裡可以對a所在的鏈做區間處理
37         //區間為(Link[ta], Link[a])
38         ta=link_top[a=pa[ta]];
39     }
40     //最後a,b會在同一條鏈 · 若a!=b還要在進行一次區間處理
41     return dep[a]<dep[b]?a:b;
42 }

```

10.5 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, int p){
6         sz[u] = 1;
7         for(auto v:g[u]){
8             if(v==p) continue;
9             if(vis[v]) continue;
10            dfs1(v, u);
11            sz[u] += sz[v];
12        }
13    };
14    function<int(int, int)> dfs2 = [&](int u, int p){
15        for(auto v:g[u]){
16            if(v==p) continue;
17            if(vis[v]) continue;
18            if(sz[v]*2<n) continue;
19            return dfs2(v, u);
20        }
21        return u;
22    };
23    dfs1(u, -1);
24    return dfs2(u, -1);
25 }
26 int cal(int u, int p = -1, int deep = 1){
27     int ans = 0;
28     tmp.pb(deep);
29     sz[u] = 1;
30     for(auto v:g[u]){
31         if(v==p) continue;
32         if(vis[v]) continue;
33         ans += cal(v, u, deep+1);
34         sz[u] += sz[v];

```



```

35 }
36 //calcuat the answer
37 return ans;
38 }
39 int centroid_decomposition(int u,int
    tree_size){
40     int center = tree_centroid(u,tree_size);
41     vis[center] = 1;
42     int ans = 0;
43     for(auto v:g[center]){
44         if(vis[v])continue;
45         ans+=cal(v);
46         for(int i = sz(tmp)-sz[v];i<sz(tmp);++i)
47             {
48                 //update
49             }
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;

```

10.6 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]
        .ch[1]!=x;
11 }
12 void down(int x){ //懶惰標記下推
13     if(nd[x].rev){
14         if(nd[x].ch[0])nd[nd[x].ch[0]].rev^=1;
15         if(nd[x].ch[1])nd[nd[x].ch[1]].rev^=1;
16         swap(nd[x].ch[0],nd[x].ch[1]);
17         nd[x].rev=0;
18     }
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]==
        x);
27     nd[x].pa=z;

```

```

28     if(!isroot(y))nd[z].ch[nd[z].ch[1]==y]=x;
29     nd[y].ch[d]=nd[x].ch[d^1];
30     nd[nd[y].ch[d]].pa=y;
31     nd[y].pa=x,nd[x].ch[d^1]=y;
32     up(y),up(x);
33 }
34 void splay(int x){ //將x伸展到splay tree的根
35     push_down(x);
36     while(!isroot(x)){
37         int y=nd[x].pa;
38         if(!isroot(y)){
39             int z=nd[y].pa;
40             if((nd[z].ch[0]==y)^(nd[y].ch[0]==x))
41                 rotate(y);
42             else rotate(x);
43         }
44         rotate(x);
45     }
46 int access(int x){
47     int last=0;
48     while(x){
49         splay(x);
50         nd[x].ch[1]=last;
51         up(x);
52         last=x;
53         x=nd[x].pa;
54     }
55     return last; //access後splay tree的根
56 }
57 void access(int x,bool is=0){ //is=0就是一般
    的access
58     int last=0;
59     while(x){
60         splay(x);
61         if(is&&!nd[x].pa){
62             //printf("%d\n",max(nd[last].ma,nd[nd[
                x].ch[1]].ma));
63         }
64         nd[x].ch[1]=last;
65         up(x);
66         last=x;
67         x=nd[x].pa;
68     }
69 }
70 void query_edge(int u,int v){
71     access(u);
72     access(v,1);
73 }
74 void make_root(int x){
75     access(x),splay(x);
76     nd[x].rev^=1;
77 }
78 void make_root(int x){
79     nd[access(x)].rev^=1;
80     splay(x);
81 }
82 void cut(int x,int y){
83     make_root(x);
84     access(y);
85     splay(y);
86     nd[y].ch[0]=0;
87     nd[x].pa=0;
88 }
89 void cut_parents(int x){

```

```

90     access(x);
91     splay(x);
92     nd[nd[x].ch[0]].pa=0;
93     nd[x].ch[0]=0;
94 }
95 void link(int x,int y){
96     make_root(x);
97     nd[x].pa=y;
98 }
99 int find_root(int x){
100     x=access(x);
101     while(nd[x].ch[0])x=nd[x].ch[0];
102     splay(x);
103     return x;
104 }
105 int query(int u,int v){
106     //傳回uv路徑splay tree的根結點
107     //這種寫法無法求LCA
108     make_root(u);
109     return access(v);
110 }
111 int query_lca(int u,int v){
112     //假設路徑上點權的總和，sum是子樹的權重，
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]].
            sum
119     }else{
120         //return nd[lca].data+nd[nd[lca].ch[1]].
            sum+nd[u].sum
121     }
122 }
123 struct EDGE{
124     int a,b,w;
125     }e[10005];
126 int n;
127 vector<pair<int,int>> G[10005];
128 //first表示子節點，second表示邊的編號
129 int pa[10005],edge_node[10005];
130 //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
131 void bfs(int root){
132     //在建構的時候把每個點都設成一個splay tree
133     queue<int> q;
134     for(int i=1;i<=n;++i)pa[i]=0;
135     q.push(root);
136     while(q.size()){
137         int u=q.front();
138         q.pop();
139         for(auto P:G[u]){
140             int v=P.first;
141             if(v!=pa[u]){
142                 pa[v]=u;
143                 nd[v].pa=u;
144                 nd[v].data=e[P.second].w;
145                 edge_node[P.second]=v;
146                 up(v);
147                 q.push(v);
148             }
149         }
150     }

```

```

150 }
151 void change(int x,int b){
152     splay(x);
153     //nd[x].data=b;
154     up(x);
155 }

```

10.7 LCA

```

1 const int MAXN=200000; // 1-base
2 const int MLG=__lg(MAXN) + 1; //Log2(MAXN)
    +1;
3 int pa[MLG+2][MAXN+5];
4 int dep[MAXN+5];
5 vector<int> G[MAXN+5];
6 void dfs(int x,int p=0){ //dfs(root);
7     pa[0][x]=p;
8     for(int i=0;i<=MLG;++i)
9         pa[i+1][x]=pa[i][pa[i][x]];
10     for(auto &i:G[x]){
11         if(i==p)continue;
12         dep[i]=dep[x]+1;
13         dfs(i,x);
14     }
15 }
16 inline int jump(int x,int d){
17     for(int i=0;i<=MLG;++i)
18         if((d>>i)&1) x=pa[i][x];
19     return x;
20 }
21 inline int find_lca(int a,int b){
22     if(dep[a]>dep[b])swap(a,b);
23     b=jump(b,dep[b]-dep[a]);
24     if(a==b)return a;
25     for(int i=MLG;i>=0;--i){
26         if(pa[i][a]!=pa[i][b]){
27             a=pa[i][a];
28             b=pa[i][b];
29         }
30     }
31     return pa[0][a];
32 }
33 }
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 | }

```

10.8 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
15 |    ()) - level.begin();
16 | dfs(a); // a 必然是直徑的其中一個端點
17 | int b = max_element(level.begin(), level.end
18 |    ()) - level.begin();
19 | cout << level[b] << endl;
20 |
21 | //紀錄每個點的最長距離跟次長距離
22 | vector<int> D1, D2; // 最遠、次遠距離
23 | int ans = 0; // 直徑長度
24 |
25 | void dfs(int u, int parent = -1) {
26 |     D1[u] = D2[u] = 0;
27 |     for (int v : Tree[u]) {
28 |         if (v == parent) continue;
29 |         dfs(v, u);
30 |         int dis = D1[v] + 1;
31 |         if (dis > D1[u]) {
32 |             D2[u] = D1[u];
33 |             D1[u] = dis;
34 |         } else
35 |             D2[u] = max(D2[u], dis);
36 |     }
37 |     ans = max(ans, D1[u] + D2[u]);
38 | }

```

10.9 樹壓平

```

1 | //紀錄in & out
2 | vector<int> Arr;
3 | vector<int> In, Out;
4 | void dfs(int u) {
5 |     Arr.push_back(u);
6 |     In[u] = Arr.size() - 1;
7 |     for (auto v : Tree[u]) {
8 |         if (v == parent[u])
9 |             continue;
10 |        parent[v] = u;
11 |        dfs(v);
12 |    }

```

```

13 | Out[u] = Arr.size() - 1;
14 | }
15 |
16 | //進去出來都紀錄
17 | vector<int> Arr;
18 | void dfs(int u) {
19 |     Arr.push_back(u);
20 |     for (auto v : Tree[u]) {
21 |         if (v == parent[u])
22 |             continue;
23 |         parent[v] = u;
24 |         dfs(v);
25 |     }
26 |     Arr.push_back(u);
27 | }
28 |
29 | //用Treap紀錄
30 | Treap *root = nullptr;
31 | vector<Treap*> In, Out;
32 | void dfs(int u) {
33 |     In[u] = new Treap(cost[u]);
34 |     root = merge(root, In[u]);
35 |     for (auto v : Tree[u]) {
36 |         if (v == parent[u])
37 |             continue;
38 |         parent[v] = u;
39 |         dfs(v);
40 |     }
41 |     Out[u] = new Treap(0);
42 |     root = merge(root, Out[u]);
43 | }
44 | //Treap紀錄Parent
45 | struct Treap {
46 |     Treap *lc = nullptr, *rc = nullptr;
47 |     Treap *pa = nullptr;
48 |     unsigned pri, size;
49 |     long long Val, Sum;
50 |     Treap(int Val):
51 |         pri(rand()), size(1),
52 |         Val(Val), Sum(Val) {}
53 |     void pull();
54 | };
55 |
56 | void Treap::pull() {
57 |     size = 1;
58 |     Sum = Val;
59 |     pa = nullptr;
60 |     if (lc) {
61 |         size += lc->size;
62 |         Sum += lc->Sum;
63 |         lc->pa = this;
64 |     }
65 |     if (rc) {
66 |         size += rc->size;
67 |         Sum += rc->Sum;
68 |         rc->pa = this;
69 |     }
70 | }
71 | //找出節點在中序的編號
72 | size_t getIdx(Treap *x) {
73 |     assert(x);
74 |     size_t Idx = 0;
75 |     for (Treap *child = x->rc; x;) {
76 |         if (child == x->rc)
77 |             Idx += 1 + size(x->lc);

```

```

78 |     child = x;
79 |     x = x->pa;
80 | }
81 | return Idx;
82 | }
83 | //切出想要的東西
84 | void move(Treap *&root, int a, int b) {
85 |     size_t a_in = getIdx(In[a]), a_out =
86 |         getIdx(Out[a]);
87 |     auto [L, tmp] = splitK(root, a_in - 1);
88 |     auto [tree_a, R] = splitK(tmp, a_out -
89 |         a_in + 1);
90 |     root = merge(L, R);
91 |     tie(L, R) = splitK(root, getIdx(In[b]));
92 |     root = merge(L, merge(tree_a, R));

```

11 string

11.1 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
14 |    fail)*/
15 | int kmp_match(char *A,int lenA,char *B,int
16 |    lenB,int *fail){
17 |     int id=-1,ans=0;
18 |     for(int i=0;i<lenA;++i){
19 |         while(~id&&B[id+1]!=A[i])id=fail[id];
20 |         if(B[id+1]==A[i])++id;
21 |         if(id==lenB-1){/*匹配成功*/
22 |             ++ans, id=fail[id];
23 |             match_index.emplace_back(i + 1 -lenB);
24 |         }
25 |     }
26 |     return ans;

```

11.2 reverseBWT

```

1 | const int MAXN = 305, MAXC = 'Z';
2 | int ranks[MAXN], tots[MAXC], first[MAXC];
3 | void rankBWT(const string &bw){
4 |     memset(ranks,0,sizeof(int)*bw.size());
5 |     memset(tots,0,sizeof(tots));
6 |     for(size_t i=0;i<bw.size();++i)

```

```

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

```

11.3 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
6 |             +1;
7 |         while(i+z[i]<len&&s[i+z[i]]==s[z[i]])++z
8 |             [i];
9 |         if(i+z[i]-1>r)r=i+z[i]-1,l=i;
10 |    }
11 | }

```

11.4 Trie

```

1 | template<int ALPHABET = 26, char MIN_CHAR =
2 |     'a'>
3 | class trie {
4 | public:
5 |     struct Node {
6 |         int go[ALPHABET];
7 |         Node() {
8 |             memset(go, -1, sizeof(go));
9 |         }
10 |    };
11 |    trie() {
12 |        newNode();
13 |    }
14 |
15 |     inline int next(int p, int v) {
16 |         return nodes[p].go[v] != -1 ? nodes[p].
17 |             go[v] : nodes[p].go[v] = newNode();
18 |     }

```

```

19 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 };

```

```

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             void add_char(char c) {
69                 for(int i = 0; i < HASH_COUNT; ++i) {
70                     const auto& p = HASH_PAIRS[i];
71                     pref[i].push_back((1LL * (n == 0 ? 0 :
72                         pref[i].back()) * p.first + c) %
73                         p.second);
74                 }
75                 n += 1;
76                 if(PRECOMPUTE_POWERS) {
77                     build_powers(n);
78                 }
79             }
80
81             // Return hash values for [l, r)
82             array<int, HASH_COUNT> substr(int l, int r)
83             {
84                 array<int, HASH_COUNT> res{};
85                 for(int i = 0; i < HASH_COUNT; ++i) {
86                     res[i] = substr(i, l, r);
87                 }
88                 return res;
89             }
90
91             array<int, HASH_COUNT> merge(const vector<
92                 pair<int, int>>& seg) {
93                 array<int, HASH_COUNT> res{};
94                 for(int i = 0; i < HASH_COUNT; ++i) {
95                     const auto& p = HASH_PAIRS[i];
96                     for(auto [l, r] : seg) {
97                         res[i] = (1LL * res[i] * get_power(i,
98                             r - l) + substr(i, l, r)) % p.
99                             second;
100                     }
101                 }
102                 return res;
103             }
104
105             // build powers up to x^k
106             void build_powers(int k) {
107                 for(int i = 0; i < HASH_COUNT; ++i) {
108                     const auto& p = HASH_PAIRS[i];
109                     int sz = (int) POW[i].size();
110                     if(sz > k) {
111                         continue;
112                     }
113                     if(sz == 0) {
114                         POW[i].push_back(1);
115                         sz = 1;
116                     }
117                     while(sz <= k) {
118                         POW[i].push_back(1LL * POW[i].back()
119                             * p.first % p.second);
120                         sz += 1;
121                     }
122                 }
123             }
124
125             inline int size() const {
126                 return n;
127             }
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                 }
147
148                 int get_power(int a, int b) {
149                     if(PRECOMPUTE_POWERS) {
150                         build_powers(b);
151                         return POW[a][b];
152                     }
153                     const auto& p = HASH_PAIRS[a];
154                     return power(p.first, b, p.second);
155                 }
156
157                 template<int A, bool B> vector<int> Hash<A,
158                     B>::POW[Hash::MAX_HASH_PAIRS];
159
160                 void solve() {
161                     int n;
162                     string s;
163                     cin >> n >> s;
164                     Hash<10, true> h(s);
165                     set<array<int, 10>> used;
166                     for(int i = 0; i + 1 < n; ++i) {
167                         used.insert(h.merge({{0, i}, {i + 2, n}}));
168                     }
169                     cout << used.size() << "\n";
170                 }
171
172                 int main() {
173                     ios::sync_with_stdio(false);
174                     cin.tie(0);
175                     int tt;
176                     cin >> tt;
177                     while(tt--) {
178                         solve();
179                     }
180                     return 0;
181                 }
182             }
183
184             #define radix_sort(x,y){\
185                 for(i=0;i<A;++i)c[i]=0;\
186                 for(i=0;i<n;++i)c[x[y[i]]]++;\
187                 int n;
188                 static vector<int> POW[MAX_HASH_PAIRS];
189                 array<vector<int>, HASH_COUNT> pref;
190
191                 int substr(int k, int l, int r) {
192                     assert(0 <= k && k < HASH_COUNT);
193                     assert(0 <= l && l <= r && r <= n);
194                     const auto& p = HASH_PAIRS[k];
195                     if(l == r) {
196                         return 0;
197                     }
198                     int res = pref[k][r - 1];
199                     if(l > 0) {
200                         res -= 1LL * pref[k][l - 1] *
201                             get_power(k, r - l) % p.second;
202                     }
203                 }
204
205                 int get_power(int a, int b) {
206                     if(PRECOMPUTE_POWERS) {
207                         build_powers(b);
208                         return POW[a][b];
209                     }
210                     const auto& p = HASH_PAIRS[a];
211                     return power(p.first, b, p.second);
212                 }
213
214                 template<int A, bool B> vector<int> Hash<A,
215                     B>::POW[Hash::MAX_HASH_PAIRS];
216
217                 void solve() {
218                     int n;
219                     string s;
220                     cin >> n >> s;
221                     Hash<10, true> h(s);
222                     set<array<int, 10>> used;
223                     for(int i = 0; i + 1 < n; ++i) {
224                         used.insert(h.merge({{0, i}, {i + 2, n}}));
225                     }
226                     cout << used.size() << "\n";
227                 }
228
229                 int main() {
230                     ios::sync_with_stdio(false);
231                     cin.tie(0);
232                     int tt;
233                     cin >> tt;
234                     while(tt--) {
235                         solve();
236                     }
237                     return 0;
238                 }
239             }
240
241             #define AC(r,a,b)\
242                 r[a]!=(r[b]|a+k)>=n||r[a+k]!=r[b+k]
243             void suffix_array(const char *s,int n,int *
244                 sa,int *rank,int *tmp,int *c){
245                 int A='z'+1,i,k,id=0;
246                 for(i=0;i<n;++i)rank[tmp[i]=i]=s[i];
247                 radix_sort(rank,tmp);
248                 for(k=1;id<n-1;k<=1){
249                     for(id=0,i=n-k;i<n;++i)tmp[id++]=i;
250                     for(i=0;i<n;++i)
251                         if(sa[i]>=k)tmp[id++]=sa[i]-k;
252                     radix_sort(rank,tmp);
253                 }
254             }

```

11.5 Rolling Hash

```

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

```

11.6 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     int n;
5     static vector<int> POW[MAX_HASH_PAIRS];
6     array<vector<int>, HASH_COUNT> pref;
7
8     int substr(int k, int l, int r) {
9         assert(0 <= k && k < HASH_COUNT);
10        assert(0 <= l && l <= r && r <= n);
11        const auto& p = HASH_PAIRS[k];
12        if(l == r) {
13            return 0;
14        }
15        int res = pref[k][r - 1];
16        if(l > 0) {
17            res -= 1LL * pref[k][l - 1] *
18                get_power(k, r - l) % p.second;
19        }
20    }
21
22    int get_power(int a, int b) {
23        if(PRECOMPUTE_POWERS) {
24            build_powers(b);
25            return POW[a][b];
26        }
27        const auto& p = HASH_PAIRS[a];
28        return power(p.first, b, p.second);
29    }
30
31    template<int A, bool B> vector<int> Hash<A,
32        B>::POW[Hash::MAX_HASH_PAIRS];
33
34    void solve() {
35        int n;
36        string s;
37        cin >> n >> s;
38        Hash<10, true> h(s);
39        set<array<int, 10>> used;
40        for(int i = 0; i + 1 < n; ++i) {
41            used.insert(h.merge({{0, i}, {i + 2, n}}));
42        }
43        cout << used.size() << "\n";
44    }
45
46    int main() {
47        ios::sync_with_stdio(false);
48        cin.tie(0);
49        int tt;
50        cin >> tt;
51        while(tt--) {
52            solve();
53        }
54        return 0;
55    }
56
57    #define AC(r,a,b)\
58        r[a]!=(r[b]|a+k)>=n||r[a+k]!=r[b+k]
59    void suffix_array(const char *s,int n,int *
60        sa,int *rank,int *tmp,int *c){
61        int A='z'+1,i,k,id=0;
62        for(i=0;i<n;++i)rank[tmp[i]=i]=s[i];
63        radix_sort(rank,tmp);
64        for(k=1;id<n-1;k<=1){
65            for(id=0,i=n-k;i<n;++i)tmp[id++]=i;
66            for(i=0;i<n;++i)
67                if(sa[i]>=k)tmp[id++]=sa[i]-k;
68            radix_sort(rank,tmp);
69        }
70    }

```

```

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

```

11.7 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[S[t].fail].efl;
46         q.push_back(t);
47         ++qe;
48     }
49 }
50 /*DP 出每個前綴在字串s出現的次數並傳回所有
51 字串被s匹配成功的次數O(N*M)*/
52 int match_0(const char *s){
53     int ans = 0, id, p = 0, i;
54     for(i = 0; s[i]; ++i){
55         id = s[i] - L;
56         while(!S[p].next[id] && p) p = S[p].fail;
57         if(!S[p].next[id]) continue;
58         p = S[p].next[id];
59         ++S[p].cnt_dp; /*匹配成功則它所有後綴都
60 可以被匹配(DP計算)*/
61     }
62     for(i = qe - 1; i >= 0; --i){
63         ans += S[q[i]].cnt_dp * S[q[i]].ed;
64         if(~S[q[i]].fail) S[S[q[i]].fail].cnt_dp += S[q[i]].cnt_dp;
65     }
66     return ans;
67 }
68 /*多串匹配走efl邊並傳回所有字串被s匹配成功的
69 次數O(N*M^1.5)*/
70 int match_1(const char *s) const{
71     int ans = 0, id, p = 0, t;
72     for(int i = 0; s[i]; ++i){
73         id = s[i] - L;
74         while(!S[p].next[id] && p) p = S[p].fail;
75         if(!S[p].next[id]) continue;
76         p = S[p].next[id];
77         if(S[p].ed) ans += S[p].ed;
78         for(t = S[p].efl; ~t; t = S[t].efl){
79             ans += S[t].ed; /*因為都走efl邊所以保證
80 匹配成功*/
81         }
82     }
83     return ans;
84 }
85 /*枚舉(s的子字串a)的所有相異字串各恰一次
86 並傳回次數O(N*M*(1/3))*/
87 int match_2(const char *s){
88     int ans = 0, id, p = 0, t;
89     ++vt;
90     /*把數記vt+=1，只要vt沒溢位，所有S[p].
91     vis==vt就會變成false
92     這種利用vt的方法可以O(1)歸零vis陣列*/
93     for(int i = 0; s[i]; ++i){
94         id = s[i] - L;
95         while(!S[p].next[id] && p) p = S[p].fail;
96         if(!S[p].next[id]) continue;
97         p = S[p].next[id];
98         if(S[p].ed && S[p].vis != vt){
99             S[p].vis = vt;
100             ans += S[p].ed;
101         }
102     }

```

```

103     for(t = S[p].efl; ~t && S[t].vis != vt; t = S[t].efl){
104         S[t].vis = vt;
105         ans += S[t].ed; /*因為都走efl邊所以保證
106 匹配成功*/
107     }
108     return ans;
109 }
110 /*把AC自動機變成真的自動機*/
111 void evolution(){
112     for(qs = 1; qs != qe;){
113         int p = q[qs++];
114         for(int i = 0; i < R-L; ++i)
115             if(S[p].next[i] == 0) S[p].next[i] = S[S[p].fail].next[i];
116     }
117 }

```

11.8 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 }

```

11.9 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 }

```

12 tools

12.1 Template

```

1 #include <bits/stdc++.h>
2 #include <bits/stdc++.h>
3 #pragma GCC optimize("O3,unroll-loops")
4 #pragma GCC target("avx2,bmi,bmi2,lzcnt,
5     popcnt")
6 #define IOS ios::sync_with_stdio(0), cin.tie(0), cout.tie(0)
7 #define int long long
8 #define double long double
9 #define pb push_back
10 #define sz(x) (int)(x).size()
11 #define all(v) begin(v), end(v)
12 #define debug(x) cerr<<#x<<" = "<<x<<'\n'
13 #define LINE cout<<"\n-----\n"
14 #define endl '\n'
15 #define VI vector<int>
16 #define F first
17 #define S second
18 #define MP(a,b) make_pair(a,b)
19 #define rep(i,m,n) for(int i = m; i <= n; ++i)
20 #define res(i,m,n) for(int i = m; i >= n; --i)
21 #define gcd(a,b) __gcd(a,b)
22 #define lcm(a,b) a*b/gcd(a,b)
23 #define Case() int _; cin>>_; for(int Case = 1; Case <= _; ++Case)
24 #define pii pair<int,int>
25 using namespace __gnu_cxx;
26 using namespace __gnu_pbds;
27 template <typename K, typename cmp = less<K>,
28         typename T = thin_heap_tag> using
29     _heap = __gnu_pbds::priority_queue<K, cmp, T>;
30 template <typename K, typename M = null_type,
31         > using _hash = gp_hash_table<K, M>;
32 const int N = 1e6+5, L = 20, mod = 1e9+7;
33 const long long inf = 2e18+5;
34 const double eps = 1e-7, pi = acos(-1);
35 void solve(){
36 }
37 signed main(){
38     IOS;
39     solve();
40 }
41 //使用內建紅黑樹
42 template<class T, typename cmp=less<>>struct
43     _tree{//#include<bits/stdc++.h>
44         tree<pair<T,int>, null_type, cmp, rb_tree_tag,
45             tree_order_statistics_node_update>st;
46         int id = 0;
47         void insert(T x){st.insert({x, id++});}
48         void erase(T x){st.erase(st.lower_bound({x, 0}));}
49         int order_of_key(T x){return st.
50             order_of_key(*st.lower_bound({x, 0}));}
51         T find_by_order(int x){return st.
52             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 };

```

12.2 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(),
6     Bucket.begin());
7   vector<unsigned> Ans(Arr.size());
8   for(auto Iter = Arr.rbegin(); Iter != Arr.
9     rend(); ++Iter) Ans[--Bucket[*Iter]] =
10     *Iter;
11   return Ans;
12 }

```

12.3 HashMap

```

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

```

12.4 Bsearch

```

1 //Lower bound
2 int lower_bound(int arr[], int n, int val) {
3   int l = 0, r = n-1, mid, ret = -1; //沒搜
4   到return -1
5   while (l <= r) {

```

```

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

```

12.5 relabel

```

1 template<class T>
2 vector<int> Discrete(const vector<T>&v){
3   vector<int> ans;
4   vector<T> tmp(v);
5   sort(begin(tmp), end(tmp));
6   tmp.erase(unique(begin(tmp), end(tmp)), end(
7     tmp));
8   for(auto i:v) ans.push_back(lower_bound(
9     begin(tmp), end(tmp), i)-tmp.begin()+1);
10  return ans;
11 }

```

12.6 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 // return the maximum of f(x) in [l, r]
13 int ternary_search(int l, int r) {
14   while(r - l > 1) {
15     int mid = (l + r) / 2;
16     if(f(m) > f(m + 1)) r = m;
17     else l = m;
18   }
19   return r;
20 }
21 }

```

12.7 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 }

```

12.8 bitset

```

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

```

13 圖論

13.1 SCC

```

1 //CSES Reachability Queries
2 #include<bits/stdc++.h>
3 using namespace std;
4 const int N = 5e4+5;
5
6 int n,m,q,Time = 0,SCCID = 0;
7 vector<vector<int>> G(N), invDAG(N);
8 vector<int> low(N,0),depth(N,0),inStack(N),
9   SCC(N,-1), indegree_SCC(N,0);
10 stack<int> stk;
11 bitset<N> canReach[N];
12 void DFS(int u, int fa) {
13   depth[u] = low[u] = ++Time;

```

```

13   stk.emplace(u);
14   inStack[u] = 1;
15   for(int v : G[u]) {
16     if(!depth[v]) {
17       DFS(v,u);
18       low[u] = min(low[u], low[v]);
19     }
20     else if (inStack[v]) {
21       low[u] = min(low[u], depth[v]);
22     }
23   }
24   if(depth[u] == low[u]) {
25     int x;
26     do {
27       x = stk.top();
28       stk.pop();
29       SCC[x] = SCCID;
30       inStack[x] = 0;
31     } while(x != u);
32     ++SCCID;
33   }
34   return ;
35 }
36 void Compress() {
37   for(int u = 1; u <= n; u++) {
38     for(int v : G[u]) {
39       if(SCC[v] == SCC[u]) continue;
40       invDAG[SCC[v]].emplace_back(SCC[u]);
41       indegree_SCC[SCC[u]]++;
42     }
43   }
44 }
45 void topological_sort() {
46   queue<int> Q;
47   for(int i = 0; i < SCCID; i++) {
48     canReach[i][i] = 1;
49     if(indegree_SCC[i] == 0) Q.emplace(i);
50   }
51   while(!Q.empty()) {
52     int u = Q.front(); Q.pop();
53     for(int v : invDAG[u]) {
54       canReach[v] |= canReach[u];
55       indegree_SCC[v]--;
56       if(indegree_SCC[v] == 0) Q.emplace(v);
57     }
58   }
59 }
60 signed main() {
61   cin>>n>m>q;
62   while(m--) {
63     int u,v; cin>>u>>v;
64     G[u].emplace_back(v);
65   }
66   for(int i = 1; i <= n; i++) if(!depth[i])
67     DFS(i,i);
68   Compress();
69   topological_sort();
70   while(q--) {
71     int a,b; cin>>a>>b;
72     cout<<(canReach[SCC[a]][SCC[b]] ? "YES\n
   " : "NO\n");
73   }
74 }

```

```

77 //CSES Giant Pizza
78 //2-SAT
79 #include <bits/stdc++.h>
80 #define pb push_back
81 using namespace std;
82 int k = 0;
83 array<vector<int>, 200004> G, B, C, S;
84 array<bool, 200004> vis;
85 array<int, 200004> scc, ans, in;
86 stack<int> out, ord;
87 void bfs(int u){
88     if(vis[u]) return;
89     vis[u] = 1;
90     for(int v : B[u]) bfs(v);
91     out.push(u);
92 }
93 void dfs(int u){
94     if(scc[u]) return;
95     scc[u] = k;
96     for(int v : G[u]) dfs(v);
97 }
98 void topu(int n){
99     int u;
100     bool ok;
101     queue<int> Q;
102     for(int i = 1; i <= 2 * n + 1; i++){
103         if(!in[i]) Q.push(i);
104         ans[i] = -1;
105     }
106     while(!Q.empty()){
107         u = Q.front();
108         Q.pop();
109         ord.push(u);
110         for(int v : S[u]){
111             in[v]--;
112             if(!in[v]) Q.push(v);
113         }
114     }
115     while(!ord.empty()){
116         u = ord.top();
117         ord.pop();
118         ok = 1;
119         for(int v : C[u]){
120             if(ans[v / 2] >= 0) ok = 0;
121         }
122         if(!ok) continue;
123         for(int v : C[u]){
124             ans[v / 2] = v & 1;
125         }
126     }
127 }
128 signed main(){
129     int n, m, a, b, o;
130     char wa, wb;
131     bool ok = 1;
132     cin >> m >> n;
133     while(m--){
134         cin >> wa >> a >> wb >> b;
135         a = 2 * a;
136         b = 2 * b;
137         if(wa == '+') a++;
138         if(wb == '+') b++;
139         G[a ^ 1].pb(b);
140         B[b].pb(a ^ 1);
141         G[b ^ 1].pb(a);
142         B[a].pb(b ^ 1);

```

```

143     }
144     for(int i = 2; i <= 2 * n + 1; i++){
145         bfs(i);
146     }
147     while(!out.empty()){
148         o = out.top();
149         out.pop();
150         if(!scc[o]) k++, dfs(o);
151     }
152     for(int i = 1; i <= n; i++){
153         //cout << scc[2 * i] << " " << scc[2
154             * i + 1] << "\n";
155         if(scc[2 * i] == scc[2 * i + 1]) ok
156             = 0;
157     }
158     if(!ok){
159         cout << "IMPOSSIBLE";
160         return 0;
161     }
162     for(int i = 2; i <= 2 * n + 1; i++){
163         C[scc[i]].pb(i);
164         for(int v : G[i]){
165             if(scc[v] == scc[i]) continue;
166             in[scc[v]]++;
167             S[scc[i]].pb(scc[v]);
168         }
169     }
170     topu(n);
171     for(int i = 1; i <= n; i++){
172         cout << (ans[i]? "+" : "- ");
173     }
174     //CSES Planets and Kingdoms
175     #include <bits/stdc++.h>
176     #define pb push_back
177     using namespace std;
178     int k = 0;
179     array<vector<int>, 100004> G, B;
180     array<bool, 100004> vis;
181     array<int, 100004> scc;
182     stack<int> out;
183     void bfs(int u){
184         if(vis[u]) return;
185         vis[u] = 1;
186         for(int v : B[u]){
187             bfs(v);
188         }
189         out.push(u);
190     }
191     void dfs(int u){
192         if(scc[u]) return;
193         scc[u] = k;
194         for(int v : G[u]){
195             dfs(v);
196         }
197     }
198     signed main(){
199         int n, m, a, b, o;
200         cin >> n >> m;
201         while(m--){
202             cin >> a >> b;
203             G[a].pb(b);
204             B[b].pb(a);
205         }
206         for(int i = 1; i <= n; i++){

```

```

207         bfs(i);
208     }
209     while(!out.empty()){
210         o = out.top();
211         out.pop();
212         if(!scc[o]) ++k, dfs(o);
213     }
214     cout << k << "\n";
215     for(int i = 1; i <= n; i++){
216         cout << scc[i] << " ";
217     }
218     return 0;
219 }
220 //CSES Coin Collector
221 //SCC + Topological Sort + DP
222 #include <bits/stdc++.h>
223 #define pb push_back
224 #define int long long
225 using namespace std;
226 int k = 0;
227 array<int, 100004> K, scc, dp, val, in;
228 array<bool, 100004> vis;
229 array<vector<int>, 100004> G, B, S;
230 stack<int> out, ord;
231 void bfs(int u){
232     if(vis[u]) return;
233     vis[u] = 1;
234     for(int v : B[u]) bfs(v);
235     out.push(u);
236 }
237 void dfs(int u){
238     if(scc[u]) return;
239     scc[u] = k;
240     for(int v : G[u]) dfs(v);
241 }
242 int topu(int n){
243     int u, ans = 0;
244     queue<int> Q;
245     for(int i = 1; i <= n; i++){
246         if(!in[i]) Q.push(i);
247     }
248     while(!Q.empty()){
249         u = Q.front();
250         Q.pop();
251         ord.push(u);
252         for(int v : S[u]){
253             in[v]--;
254             if(!in[v]) Q.push(v);
255         }
256     }
257     while(!ord.empty()){
258         u = ord.top();
259         ord.pop();
260         dp[u] = val[u];
261         for(int v : S[u]){
262             dp[u] = max(dp[u], dp[v] + val[u
263                 ]);
264         }
265         ans = max(ans, dp[u]);
266     }
267     return ans;
268 }
269 signed main(){
270     int n, m, a, b, o;
271     cin >> n >> m;
272     for(int i = 1; i <= n; i++){

```

```

272         cin >> K[i];
273     }
274     while(m--){
275         cin >> a >> b;
276         G[a].pb(b);
277         B[b].pb(a);
278     }
279     for(int i = 1; i <= n; i++){
280         bfs(i);
281     }
282     while(!out.empty()){
283         o = out.top();
284         out.pop();
285         if(!scc[o]) k++, dfs(o);
286     }
287     for(int i = 1; i <= n; i++){
288         val[scc[i]] += K[i];
289         for(int v : G[i]){
290             if(scc[v] == scc[i]) continue;
291             in[scc[v]]++;
292             S[scc[i]].pb(scc[v]);
293         }
294     }
295     cout << topu(k);
296     return 0;
297 }

```

14 字符串

14.1 suffix array

```

1 //CSES Distinct Substrings
2 //suffix array
3 #include <bits/stdc++.h>
4 #define pb push_back
5 #define int long long
6 using namespace std;
7 array<int, 100004> SA, RNK, F, L, LCP;
8 array<vector<int>, 100004> buk;
9 void sort(array<int, 100004> &A, int n){
10     int cnt = 0;
11     for(int i = 0; i < n; i++){
12         buk[A[SA[i]]].pb(SA[i]);
13     }
14     for(int i = 0; i < max(n, 2611); i++){
15         for(int x : buk[i]){
16             SA[cnt++] = x;
17         }
18         buk[i].clear();
19     }
20 }
21 void suf(string &S){
22     int n = S.size(), cnt = -1, ff = -1, ll
23         = -1;
24     for(int i = 0; i < n; i++){
25         SA[i] = n - i - 1;
26         F[i] = S[i] - 'a';
27     }
28     sort(F, n);
29     for(int i = 0; i < n; i++){

```

```

29     if(F[SA[i]] == ff && L[SA[i]] == ll)
30         RNK[SA[i]] = cnt;
31     else RNK[SA[i]] = ++cnt;
32     ff = F[SA[i]], ll = L[SA[i]];
33 }
34 for(int j = 1; j < n; j <= 1){
35     cnt = ff = ll = -1;
36     for(int i = 0; i < n; i++){
37         F[i] = RNK[i];
38         L[i] = i + j < n? RNK[i + j] :
39             0;
40     }
41     sort(L, n);
42     sort(F, n);
43     for(int i = 0; i < n; i++){
44         if(F[SA[i]] == ff && L[SA[i]] ==
45             ll) RNK[SA[i]] = cnt;
46         else RNK[SA[i]] = ++cnt;
47         ff = F[SA[i]], ll = L[SA[i]];
48     }
49 }
50 int cp(string &S){
51     int n = S.size(), lcp = 0, k, sum = 0;
52     for(int i = 0; i < n; i++){
53         RNK[SA[i]] = i;
54     }
55     for(int i = 0; i < n; i++){
56         if(!RNK[i]) continue;
57         k = SA[RNK[i] - 1];
58         if(lcp) lcp--;
59         while(S[i + lcp] == S[k + lcp]) lcp
60             ++;
61         LCP[RNK[i]] = lcp;
62         sum += lcp;
63     }
64     return sum;
65 }
66 signed main(){
67     int n;
68     string S;
69     cin >> S;
70     n = S.size();
71     suf(S);
72     cout << n * (n + 1) / 2 - cp(S);
73     return 0;
74 }
75 //CSES Repeating Substring
76 #include <bits/stdc++.h>
77 #define pb push_back
78 using namespace std;
79 array<int, 100004> SA, RNK, F, L;
80 array<vector<int>, 100004> buk;
81 void sort(array<int, 100004> &A, int n){
82     int cnt = 0;
83     for(int i = 0; i < n; i++){
84         buk[A[SA[i]]].pb(SA[i]);
85     }
86     for(int i = 0; i < max(n, 26); i++){
87         for(int x : buk[i]){
88             SA[cnt++] = x;
89         }
90     }
91     sort(L, n);
92     sort(F, n);
93     for(int i = 0; i < n; i++){
94         if(F[SA[i]] == ff && L[SA[i]] ==
95             ll) RNK[SA[i]] = cnt;
96         else RNK[SA[i]] = ++cnt;
97         ff = F[SA[i]], ll = L[SA[i]];
98     }
99 }
100 string lcp(string &S){
101     int n = S.size(), cp = 0, k, lng = 0,
102         ans;
103     for(int i = 0; i < n; i++){
104         RNK[SA[i]] = i;
105     }
106     for(int i = 0; i < n; i++){
107         if(!RNK[i]) continue;
108         k = SA[RNK[i] - 1];
109         if(cp) cp--;
110         while(S[i + cp] == S[k + cp]) cp++;
111         if(cp > lng){
112             lng = cp;
113             ans = i;
114         }
115     }
116     if(!lng) return "-1";
117     return S.substr(ans, lng);
118 }
119 signed main(){
120     string S;
121     cin >> S;
122     suf(S);
123     cout << lcp(S);
124     return 0;
125 }
126 //CSES Substring Distribution
127 #include <bits/stdc++.h>
128 #define int long long
129 #define pb push_back
130 using namespace std;
131 array<int, 100004> SA, RNK, F, L, sum;
132 array<vector<int>, 100004> buk;
133 void sort(array<int, 100004> &A, int n){
134     int cnt = 0;
135     for(int i = 0; i < n; i++){
136         buk[A[SA[i]]].pb(SA[i]);
137     }
138     for(int i = 0; i < max(n, 26); i++){
139         for(int x : buk[i]){
140             SA[cnt++] = x;
141         }
142     }
143     sort(L, n);
144     sort(F, n);
145     for(int i = 0; i < n; i++){
146         if(F[SA[i]] == ff && L[SA[i]] ==
147             ll) RNK[SA[i]] = cnt;
148         else RNK[SA[i]] = ++cnt;
149         ff = F[SA[i]], ll = L[SA[i]];
150     }
151     for(int j = 1; j < n; j <= 1){
152         cnt = ff = ll = -1;
153         for(int i = 0; i < n; i++){
154             F[i] = RNK[i];
155             L[i] = i + j < n? RNK[i + j] :
156                 0;
157         }
158         sort(L, n);
159         sort(F, n);
160         for(int i = 0; i < n; i++){
161             if(F[SA[i]] == ff && L[SA[i]] ==
162                 ll) RNK[SA[i]] = cnt;
163             else RNK[SA[i]] = ++cnt;
164             ff = F[SA[i]], ll = L[SA[i]];
165         }
166     }
167     return sum;
168 }
169 signed main(){
170     int n = S.size(), cp = 0, k;
171     for(int i = 0; i < n; i++) RNK[SA[i]] =
172         i;
173     for(int i = 0; i < n; i++){
174         if(!RNK[i]){
175             sum[0]++;
176             sum[n - i]--;
177             continue;
178         }
179         k = SA[RNK[i] - 1];
180         if(cp) cp--;
181         while(S[i + cp] == S[k + cp]) cp++;
182         sum[cp]++;
183         sum[n - i]--;
184     }
185 }
186 signed main(){
187     int ans = 0;
188     string S;
189     cin >> S;
190     suf(S);
191     lcp(S);
192     for(int i = 0; i < S.size(); i++){
193         ans += sum[i];
194         cout << ans << " ";
195     }
196     return 0;
197 }
198 //CSES Finding Patterns
199 int cnt = 0;
200 for(int i = 0; i < n; i++){
201     buk[A[SA[i]]].pb(SA[i]);
202 }
203 for(int i = 0; i < max(n, 26); i++){
204     for(int x : buk[i]){
205         SA[cnt++] = x;
206     }
207 }
208 buk[i].clear();
209 }
210 void suf(string &S){
211     int n = S.size(), cnt, ff, ll;
212     for(int i = 0; i < n; i++){
213         SA[i] = n - i - 1;
214         RNK[i] = F[i] = S[i] - 'a';
215     }
216     sort(F, n);
217     for(int j = 1; j < n; j <= 1){
218         cnt = ff = ll = -1;
219         for(int i = 0; i < n; i++){
220             F[i] = RNK[i];
221             L[i] = i + j < n? RNK[i + j] :
222                 0;
223         }
224         sort(L, n);
225         sort(F, n);
226         for(int i = 0; i < n; i++){
227             if(F[SA[i]] == ff && L[SA[i]] ==
228                 ll) RNK[SA[i]] = cnt;
229             else RNK[SA[i]] = ++cnt;
230             ff = F[SA[i]], ll = L[SA[i]];
231         }
232     }
233 }
234 void lcp(string &S){
235     int n = S.size(), cp = 0, k;
236     for(int i = 0; i < n; i++) RNK[SA[i]] =
237         i;
238     for(int i = 0; i < n; i++){
239         if(!RNK[i]){
240             sum[0]++;
241             sum[n - i]--;
242             continue;
243         }
244         k = SA[RNK[i] - 1];
245         if(cp) cp--;
246         while(S[i + cp] == S[k + cp]) cp++;
247         sum[cp]++;
248         sum[n - i]--;
249     }
250 }
251 signed main(){
252     int ans = 0;
253     string S;
254     cin >> S;
255     suf(S);
256     lcp(S);
257     for(int i = 0; i < S.size(); i++){
258         ans += sum[i];
259         cout << ans << " ";
260     }
261     return 0;
262 }
263 //AC 自動機(suffix array)
264 #include <bits/stdc++.h>
265 #define pb push_back
266 #define mid (l + r) / 2
267 using namespace std;
268 array<int, 100004> SA, RNK, F, L;
269 array<vector<int>, 100004> buk;
270 void sort(array<int, 100004> &A, int n){
271     int cnt = 0;
272     for(int i = 0; i < n; i++){
273         buk[A[SA[i]]].pb(SA[i]);
274     }
275     for(int i = 0; i < max(n, 26); i++){
276         for(int x : buk[i]){
277             SA[cnt++] = x;
278         }
279     }
280     buk[i].clear();
281 }
282 void suf(string &S){
283     int n = S.size(), cnt, ff, ll;
284     for(int i = 0; i < n; i++){
285         SA[i] = n - i - 1;
286         RNK[i] = F[i] = S[i] - 'a';
287     }
288     sort(F, n);
289     for(int j = 1; j < n; j <= 1){
290         cnt = ff = ll = -1;
291         for(int i = 0; i < n; i++){
292             F[i] = RNK[i];
293             L[i] = i + j < n? RNK[i + j] :
294                 0;
295         }
296         sort(L, n);
297         sort(F, n);
298         for(int i = 0; i < n; i++){
299             if(F[SA[i]] == ff && L[SA[i]] ==
300                 ll) RNK[SA[i]] = cnt;
301             else RNK[SA[i]] = ++cnt;
302             ff = F[SA[i]], ll = L[SA[i]];
303         }
304     }
305 }
306 bool cmp(string &T, string &S, int k){
307     for(int i = 0; i < T.size() && i + k < S
308         .size(); i++){
309         if(T[i] < S[k + i]) return 1;
310         else if(T[i] > S[k + i]) return 0;
311     }
312     if(T.size() > S.size() - k) return 0;
313     return 1;
314 }
315 bool BS(string &S, string &T){
316     int l = 0, r = S.size() - 1;
317     while(l != r){
318         if(cmp(T, S, SA[mid])) r = mid;
319         else l = mid + 1;
320     }
321 }

```

```

275     if(T.size() > S.size() - SA[l]) return
276         0;
277     for(int i = 0; i < T.size(); i++){
278         if(T[i] != S[SA[l] + i]) return 0;
279     }
280     return 1;
281 signed main(){
282     int k;
283     string S, T;
284     cin >> S >> k;
285     suf(S);
286     while(k--){
287         cin >> T;
288         cout << (BS(S, T)? "YES\n" : "NO\n")
289             ;
290     }
291     return 0;
292 }
293 //CSES Counting Patterns
294 //AC 自動機(suffix array)
295 #include <bits/stdc++.h>
296 #define pb push_back
297 #define mid (l + r) / 2
298 using namespace std;
299 array<int, 100004> SA, RNK, F, L;
300 array<vector<int>, 100004> buk;
301 void sort(array<int, 100004> &A, int n){
302     int cnt = 0;
303     for(int i = 0; i < n; i++){
304         buk[A[SA[i]].pb(SA[i]);
305     }
306     for(int i = 0; i < max(n, 26); i++){
307         for(int x : buk[i]){
308             SA[cnt++] = x;
309         }
310         buk[i].clear();
311     }
312 }
313 void suf(string &S){
314     int n = S.size(), cnt = -1, ff = -1, ll
315         = -1;
316     for(int i = 0; i < n; i++){
317         SA[i] = i;
318         F[i] = S[i] - 'a';
319     }
320     sort(F, n);
321     for(int i = 0; i < n; i++){
322         if(F[SA[i]] == ff && L[SA[i]] == ll)
323             RNK[SA[i]] = cnt;
324         else RNK[SA[i]] = ++cnt;
325         ff = F[SA[i]], ll = L[SA[i]];
326     }
327     for(int j = 1; j < n; j <= 1){
328         cnt = ff = ll = -1;
329         for(int i = 0; i < n; i++){
330             F[i] = RNK[i];
331             L[i] = i + j < n? RNK[i + j] :
332                 0;
333         }
334         sort(L, n);
335         sort(F, n);
336         for(int i = 0; i < n; i++){
337             if(F[SA[i]] == ff && L[SA[i]] ==
338                 ll) RNK[SA[i]] = cnt;
339             else RNK[SA[i]] = ++cnt;
340             ff = F[SA[i]], ll = L[SA[i]];
341         }
342     }
343 }
344 bool cmp(string &S, string &T, int k, bool t
345 ){
346     for(int i = 0; i < T.size() && i + k < S
347         .size(); i++){
348         if(T[i] < S[i + k]) return 1;
349         else if(T[i] > S[i + k]) return 0;
350     }
351     if(T.size() > S.size() - k) return 0;
352     if(t) return 0;
353     else return 1;
354 }
355 int BS(string &S, string &T, bool t){
356     int l = 0, r = S.size() - 1;
357     while(l != r){
358         if(cmp(S, T, SA[mid], t)) r = mid;
359         else l = mid + 1;
360     }
361     return l;
362 }
363 signed main(){
364     int k;
365     string S, T;
366     cin >> S >> k;
367     S += '~';
368     suf(S);
369     while(k--){
370         cin >> T;
371         cout << BS(S, T, 1) - BS(S, T, 0) <<
372             "\n";
373     }
374     return 0;
375 }
376 //CSES Pattern Positions
377 //AC 自動機(suffix array)
378 #include <bits/stdc++.h>
379 #define pb push_back
380 #define mid ((l + r) >> 1)
381 #define lc (p << 1)
382 #define rc ((p << 1) | 1)
383 using namespace std;
384 array<int, 400004> seg;
385 array<int, 100004> SA, RNK, F, L;
386 array<vector<int>, 100004> buk;
387 void update(int p, int c, int x, int l, int
388     r){
389     if(c < l || c > r) return;
390     if(l == r){
391         seg[p] = x;
392         return;
393     }
394     update(lc, c, x, l, mid);
395     update(rc, c, x, mid + 1, r);
396     seg[p] = min(seg[lc], seg[rc]);
397 }
398 int query(int p, int ql, int qr, int l, int
399     r){
400     if(ql > r || qr < l) return 1e9;
401     if(ql <= l && qr >= r) return seg[p];
402     return min(query(lc, ql, qr, l, mid),
403         query(rc, ql, qr, mid + 1, r));
404 }
405 void sort(array<int, 100004> &A, int n){
406     int cnt = 0;
407     for(int i = 0; i < n; i++){
408         buk[A[SA[i]].pb(SA[i]);
409     }
410     for(int i = 0; i < max(n, 26); i++){
411         for(int x : buk[i]){
412             SA[cnt++] = x;
413         }
414         buk[i].clear();
415     }
416 }
417 void suf(string &S){
418     int n = S.size(), cnt = -1, ff = -1, ll
419         = -1;
420     for(int i = 0; i < n; i++){
421         SA[i] = i;
422         F[i] = S[i] - 'a';
423     }
424     sort(F, n);
425     for(int i = 0; i < n; i++){
426         if(F[SA[i]] == ff && L[SA[i]] == ll)
427             RNK[SA[i]] = cnt;
428         else RNK[SA[i]] = ++cnt;
429         ff = F[SA[i]], ll = L[SA[i]];
430     }
431     for(int j = 1; j < n; j <= 1){
432         cnt = ll = ff = -1;
433         for(int i = 0; i < n; i++){
434             F[i] = RNK[i];
435             L[i] = i + j < n? RNK[i + j] :
436                 0;
437         }
438         sort(L, n);
439         sort(F, n);
440         for(int i = 0; i < n; i++){
441             if(F[SA[i]] == ff && L[SA[i]] ==
442                 ll) RNK[SA[i]] = cnt;
443             else RNK[SA[i]] = ++cnt;
444             ff = F[SA[i]], ll = L[SA[i]];
445         }
446     }
447 }
448 bool cmp(string &S, string &T, int k, bool t
449 ){
450     for(int i = 0; i < T.size() && i + k < S
451         .size(); i++){
452         if(T[i] < S[i + k]) return 1;
453         else if(T[i] > S[i + k]) return 0;
454     }
455     if(T.size() > S.size() - k) return 0;
456     return t ^ 1;
457 }
458 int BS(string &S, string &T, bool t){
459     int l = 0, r = S.size() - 1;
460     while(l != r){
461         if(cmp(S, T, SA[mid], t)) r = mid;
462         else l = mid + 1;
463     }
464     return l;
465 }
466 }
467 return 1;
468 }
469 int pos(string &S, string &T){
470     if(BS(S, T, 1) == BS(S, T, 0)) return
471         -1;
472     return query(1, BS(S, T, 0), BS(S, T, 1)
473         - 1, 0, S.size() - 1) + 1;
474 }
475 signed main(){
476     int k;
477     string S, T;
478     cin >> S >> k;
479     S += '~';
480     for(int &s : seg) s = 1e9;
481     suf(S);
482     while(k--){
483         cin >> T;
484         cout << pos(S, T) << "\n";
485     }
486     return 0;
487 }

```

14.2 Trie

```

1 //CSES Maximum XOR Subarray
2 #include <bits/stdc++.h>
3 using namespace std;
4 int cnt = 0;
5 array<int, 200004> X;
6 array<array<int, 2>, 600004> trie;
7 void update(int p, int x, int d){
8     if(d < 0) return;
9     int c = (x >> d) & 1;
10     if(!trie[p][c]) trie[p][c] = ++cnt;
11     update(trie[p][c], x, d - 1);
12 }
13 int query(int p, int x, int d){
14     if(d < 0) return x;
15     int c = ((x >> d) & 1) ^ 1;
16     if(!trie[p][c]) c ^= 1;
17     return query(trie[p][c], x, d - 1) ^ (c
18         << d);
19 }
20 int run(int n){
21     int x = 0, ans = 0;
22     update(0, 0, 30);
23     for(int i = 1; i <= n; i++){
24         x ^= X[i];
25         ans = max(ans, query(0, x, 30));
26         update(0, x, 30);
27     }
28     return ans;
29 }
30 signed main(){
31     int n;
32     cin >> n;
33     for(int i = 1; i <= n; i++) cin >> X[i];
34     cout << run(n) << "\n";
35     return 0;
36 }
37 //CSES Word Combinations

```

```

38 //You are given a string of length n and a
   //dictionary containing k words. In how
   //many ways can you create the string
   //using the words?
39 //Trie + DP
40 #include <bits/stdc++.h>
41 #define int long long
42 using namespace std;
43 const int mod = 1e9 + 7;
44 int p = 0;
45 string S;
46 array<array<int, 26>, 1000004> trie;
47 array<int, 1000004> cnt;
48 array<int, 5004> dp;
49 void update(string s){
50     int u = 0;
51     for(int i = 0; i < s.size(); i++){
52         if(!trie[u][s[i] - 'a']) trie[u][s[i]
53             ] - 'a'] = ++p;
54         u = trie[u][s[i] - 'a'];
55     }
56     cnt[u]++;
57 }
58 int query(int i){
59     int u = 0, ans = 0;
60     for(; i < S.size(); i++){
61         if(!trie[u][S[i] - 'a']) return ans;
62         u = trie[u][S[i] - 'a'];
63         ans += (cnt[u] * dp[i + 1]) % mod;
64         ans %= mod;
65     }
66     return ans;
67 }
68 signed main(){
69     int k;
70     string K;
71     cin >> S >> k;
72     while(k--){
73         cin >> K;
74         update(K);
75     }
76     dp[S.size()] = 1;
77     for(int i = S.size() - 1; i >= 0; i--){
78         dp[i] += query(i);
79     }
80     cout << dp[0];
81     return 0;

```

14.3 KMP&Z

```

1 //CSes String Matching
2 //KMP
3 #include <bits/stdc++.h>
4 using namespace std;
5 array<int, 1000004> F;
6 void build(string T){
7     int p;
8     F[0] = -1;
9     for(int i = 1; i < T.size(); i++){
10         p = F[i - 1];
11         while(~p && T[i] != T[p + 1]) p = F[p];

```

```

12         if(T[i] == T[p + 1]) p++;
13         F[i] = p;
14     }
15 }
16 int match(string T, string S){
17     int p = -1, cnt = 0;
18     for(int i = 0; i < S.size(); i++){
19         while(~p && S[i] != T[p + 1]) p = F[p];
20         if(S[i] == T[p + 1]) p++;
21         if(p + 1 == T.size()) cnt++, p = F[p];
22     }
23     return cnt;
24 }
25 signed main(){
26     string S, T;
27     cin >> S >> T;
28     build(T);
29     cout << match(T, S);
30     return 0;
31 }
32 //CSes Finding Borders
33 //Z
34 #include <bits/stdc++.h>
35 using namespace std;
36 array<int, 1000004> Z;
37 signed main(){
38     string S;
39     int l = 0, r = 0;
40     cin >> S;
41     Z[0] = S.size();
42     for(int i = 1; i < S.size(); i++){
43         if(i + Z[i - 1] <= r) Z[i] = Z[i - 1];
44         else{
45             l = i;
46             if(i > r) r = i;
47             while(r < S.size() && S[r] == S[
48                 r - 1]) r++;
49             r--;
50             Z[i] = r - l + 1;
51         }
52     }
53     for(int i = S.size() - 1; i > 0; i--){
54         if(Z[i] == .size() - i) cout << Z[i]
55             << " ";
56     }
57     return 0;
58 }
59 //CSes Finding Periods
60 #include <bits/stdc++.h>
61 using namespace std;
62 array<int, 1000004> Z;
63 signed main(){
64     int l = 0, r = 0;
65     string S;
66     cin >> S;
67     Z[0] = S.size();
68     for(int i = 1; i < S.size(); i++){
69         if(i + Z[i - 1] <= r) Z[i] = Z[i - 1];
70         else{
71             l = i;
72             if(i > r) r = i;

```

```

73         while(r < S.size() && S[r] == S[
74             r - 1]) r++;
75         Z[i] = r - l + 1;
76         if(Z[i] == S.size() - i) cout <<
77             i << " ";
78     }
79     cout << S.size();
80     return 0;
81 }
82 //CSes Required Substring
83 //KMP + DP
84 #include <bits/stdc++.h>
85 #define int long long
86 using namespace std;
87 const int mod = 1e9 + 7;
88 array<int, 104> F;
89 array<array<int, 104>, 1004> dp;
90 void KMP(string &S){
91     int p = 0;
92     for(int i = 2; i < S.size(); i++){
93         while(p && S[i - 1] != S[p]) p = F[p];
94         if(S[i - 1] == S[p]) p++;
95         F[i] = p;
96     }
97 }
98 int DP(string &S, int n, int m){
99     int p;
100     dp[0][0] = 1;
101     for(int i = 1; i <= m; i++){
102         for(int j = 0; j < n; j++){
103             for(char k = 'A'; k <= 'Z'; k++){
104                 p = j;
105                 while(p && S[p] != k) p = F[p];
106                 if(S[p] == k) p++;
107                 dp[i][p] += dp[i - 1][j];
108                 dp[i][p] %= mod;
109             }
110         }
111         dp[i][n] += dp[i - 1][n] * 26;
112         dp[i][n] %= mod;
113     }
114     return dp[m][n];
115 }
116 signed main(){
117     int n, m;
118     string S;
119     cin >> m >> S;
120     n = S.size();
121     KMP(S);
122     cout << DP(S, n, m);
123     return 0;
124 }
125 //CSes String Functions
126 //Z + KMP
127 #include <bits/stdc++.h>
128 using namespace std;
129 array<int, 1000004> Z, F;
130 void ZZZ(string &S){
131     int l = 0, r = 0;

```

```

132     Z[0] = 0;
133     for(int i = 1; i < S.size(); i++){
134         if(i + Z[i - 1] < r) Z[i] = Z[i - 1];
135         else{
136             l = i;
137             if(i > r) r = i;
138             while(r < S.size() && S[r] == S[
139                 r - 1]) r++;
140             Z[i] = r - l;
141         }
142     }
143 }
144 void KMP(string &S){
145     int p;
146     F[0] = -1;
147     for(int i = 1; i < S.size(); i++){
148         p = F[i - 1];
149         while(~p && S[p + 1] != S[i]) p = F[p];
150         if(S[i] == S[p + 1]) p++;
151         F[i] = p;
152     }
153 }
154 signed main(){
155     string S;
156     cin >> S;
157     ZZZ(S);
158     for(int i = 0; i < S.size(); i++){
159         cout << Z[i] << " ";
160     }
161     cout << "\n";
162     KMP(S);
163     for(int i = 0; i < S.size(); i++){
164         cout << F[i] + 1 << " ";
165     }
166     return 0;
167 }

```

14.4 Hash

```

1 //Tree Isomorphism I
2 #include <bits/stdc++.h>
3 #define int long long
4 #define pb push_back
5 using namespace std;
6 const int mod = 1000696969, c = 41;
7 array<int, 100004> C, S1, S2;
8 array<vector<int>, 100004> T1, T2;
9 void build(int n){
10     C[0] = 1;
11     for(int i = 1; i <= n; i++){
12         C[i] = c * C[i - 1] % mod;
13     }
14 }
15 int DFS(array<vector<int>, 100004> &T, array
16     <int, 100004> &S, int u, int pre){
17     int sum = 0;
18     vector<pair<int, int>> H;
19     for(int v : T[u]){
20         if(v == pre) continue;
21         H.pb({DFS(T, S, v, u), S[v]});

```



```

22 sort(H.begin(), H.end());
23 for(auto [h, s] : H){
24     sum = (sum + h * C[S[u]]) % mod;
25     S[u] += s;
26 }
27 S[u]++;
28 sum = (sum + S[u] * C[S[u]]) % mod;
29 return sum;
30 }
31 signed main(){
32     int t, n, a, b;
33     build(100000);
34     cin >> t;
35     while(t--){
36         cin >> n;
37         for(int i = 1; i <= n; i++){
38             S1[i] = S2[i] = 0;
39             T1[i].clear();
40             T2[i].clear();
41         }
42         for(int i = 1; i < n; i++){
43             cin >> a >> b;
44             T1[a].pb(b);
45             T1[b].pb(a);
46         }
47         for(int i = 1; i < n; i++){
48             cin >> a >> b;
49             T2[a].pb(b);
50             T2[b].pb(a);
51         }
52         if(DFS(T1, S1, 1, 0) == DFS(T2, S2, 1, 0)) cout << "YES\n";
53         else cout << "NO\n";
54     }
55     return 0;
56 }
57 //Palindrome Queries
58 //Hash + RMQ
59 #include <bits/stdc++.h>
60 #define int long long
61 #define mid ((l + r) >> 1)
62 #define lc (p << 1)
63 #define rc ((p << 1) | 1)
64 using namespace std;
65 const int mod = 1e9 + 7;
66 array<int, 200004> H;
67 array<int, 800004> F, B;
68 void ha(int n){
69     H[0] = 1;
70     for(int i = 1; i <= n; i++){
71         H[i] = (H[i - 1] * 29) % mod;
72     }
73 }
74 void pull(int p, int l, int r){
75     F[p] = (F[lc] + H[mid - 1 + 1] * F[rc]) % mod;
76     B[p] = (B[rc] + H[r - mid] * B[lc]) % mod;
77 }
78 void update(int p, int c, int x, int l, int r){
79     if(c < l || c > r) return;
80     if(l == r){
81         F[p] = B[p] = x;
82         return;
83     }

```

```

84     }
85     update(lc, c, x, l, mid);
86     update(rc, c, x, mid + 1, r);
87     pull(p, l, r);
88 }
89 int query(int p, int ql, int qr, int l, int r, bool t){
90     if(ql > r || qr < l) return 0;
91     if(t){
92         if(ql <= l && qr >= r) return F[p];
93         return (query(lc, ql, qr, l, mid, t) + H[max(0ll, mid - max(l, ql) + 1)] * query(rc, ql, qr, mid + 1, r, t)) % mod;
94     }else{
95         if(ql <= l && qr >= r) return B[p];
96         return (query(rc, ql, qr, mid + 1, r, t) + H[max(0ll, min(r, qr) - mid]) * query(lc, ql, qr, l, mid, t)) % mod;
97     }
98 }
99 signed main(){
100     int n, q, t, l, r, k, f, b;
101     char x;
102     cin >> n >> q;
103     ha(n);
104     for(int i = 1; i <= n; i++){
105         cin >> x;
106         update(1, i, x - 'a', 1, n);
107     }
108     while(q--){
109         cin >> t;
110         if(t == 1){
111             cin >> k >> x;
112             update(1, k, x - 'a', 1, n);
113         }else{
114             cin >> l >> r;
115             cout << (query(1, l, r, 1, n, 1) == query(1, l, r, 1, n, 0)) ? "YES\n" : "NO\n";
116         }
117     }
118     return 0;
119 }

```

15 計算幾何

15.1 Convex Hull

```

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 struct PT{
5     long long x,y;
6     PT(long long x=0,long long y=0):x(x),y(y){}
7     PT operator+(const PT &b)const{
8         return PT(x+b.x,y+b.y);
9     }
10    PT operator-(const PT &b)const{

```

```

11    return PT(x-b.x,y-b.y);
12 }
13 PT operator*(long long b)const{
14     return PT(x*b,y*b);
15 }
16 PT operator/(long long b)const{
17     return PT(x/b,y/b);
18 }
19 long long dot(const PT &b)const{
20     return x*b.x+y*b.y;
21 }
22 long long cross(const PT &b)const{
23     return x*b.y-y*b.x;
24 }
25 long long abs2() const {
26     return dot(*this);
27 }
28 PT normal() const {
29     return PT(-y,x);
30 }
31 };//基礎運算
32
33 bool btw(const PT &p1,const PT &p2,const PT &p3){
34     return (p1-p3).dot(p2-p3)<=0;
35 }//p3在不在p1跟p2之中
36
37 bool collinear(const PT &p1,const PT &p2,const PT &p3){
38     return (p1-p3).cross(p2-p3)==0;
39 }//共線
40
41 bool pointOnSegment(const PT &p1,const PT &p2,const PT &p3){
42     return collinear(p1,p2,p3)&&btw(p1,p2,p3);
43 }//判斷p3是不是在線段(p1,p2)上
44
45 int ori(const PT &p1,const PT &p2,const PT &p3){
46     long long a=(p2-p1).cross(p3-p1);
47     if(a==0)return 0;
48     return a>0 ?1:-1;
49 }//有向面積正負
50
51 bool seg_intersect(const PT &p1,const PT &p2,const PT &p3,const PT &p4){
52     int a123=ori(p1,p2,p3);
53     int a124=ori(p1,p2,p4);
54     int a341=ori(p3,p4,p1);
55     int a342=ori(p3,p4,p2);
56     if(a123==0&&a124==0)
57         return btw(p1,p2,p3)||btw(p1,p2,p4)||btw(p3,p4,p1)||btw(p3,p4,p2);
58     else if(a123*a124<=0&&a341*a342<=0)
59         return true;
60     return false;
61 }//線段相交
62
63 PT intersect(const PT &p1,const PT &p2,const PT &p3,const PT &p4){
64     long long a123 = (p2-p1).cross(p3-p1);

```

```

65     long long a124 = (p2-p1).cross(p4-p1);
66     return (p4*a123-p3*a124) / (a123-a124);
67 }//找交點
68
69 long long area(const vector<PT> &Polygon) {
70     if(Polygon.size() <= 1) return 0;
71     long long ans = 0;
72     for(auto a = --Polygon.end(), b = Polygon.begin(); b != Polygon.end(); a = b++) ans += a->cross(*b);
73     return ans / 2;
74 }//多邊形面積
75
76 int PointInPolygon(const vector<PT> &Polygon, const PT &p) {
77     int ans = 0;
78     for(auto a = --Polygon.end(), b = Polygon.begin(); b != Polygon.end(); a = b++){
79         if(pointOnSegment(*a, *b, p)) return -1;
80         if((a->y > p.y) != (b->y > p.y) && (p.x - b->x) < (a->x - b->x) * (p.y - b->y) / (a->y - b->y)) ans = !ans;
81     }
82     return ans;
83 }//點是否在簡單多邊形內，是的話回傳1、在邊上回傳-1、否則回傳0
84
85 bool x_cmp(const PT &a, const PT &b) {
86     return (a.x < b.x) || (a.x == b.x && a.y < b.y);
87 }
88
89 //Andrew 's Monotone Chain求凸包
90 vector<PT> getConvexHull(vector<PT> Points) {
91     sort(Points.begin(), Points.end(), x_cmp);
92     vector<PT> ans;
93     int m = 0, t = 1;
94     auto addPoint = [&](const PT &p) {
95         while (m > t && (ans[m - 1] - ans[m - 2]).cross(p - ans[m - 2]) < 0)
96             ans.pop_back(), --m;
97         ans.emplace_back(p);
98         ++m;
99     };
100     for (size_t i = 0; i < Points.size(); ++i)
101         addPoint(Points[i]);
102     t = m;
103     for (int i = int(Points.size()) - 2; i >= 0; --i) addPoint(Points[i]);
104     if (Points.size() > 1) ans.pop_back();
105     return ans;
106 }
107
108 //旋轉卡尺
109 long long rotatingClaiper(vector<PT> p){ //計算最遠點對距離的平方
110     int n=p.size(),t=1;
111     long long ans=0;
112     p.push_back(p[0]);
113     for(int i=0;i<n;i++){

```

```

118     PT now=p[i+1]-p[i]; //當前這條線的方向向量
119     //找出距離邊 (p[i],p[i+1]) 最遠的點P[t]
120     while(now.cross(p[t+1]-p[t]) > now.cross(p[t]-p[i]))
121         t=(t+1)%n;
122     ans = max(ans, (p[i]-p[t]).abs2());
123 }
124 return ans;
125 }
126
127 signed main() {
128     int n;cin>>n;
129     vector<PT> points(n);
130     for(auto &p:points) cin>>p.x>>p.y;
131     vector<PT> ConvexHull = getConvexHull(points);
132     cout<<ConvexHull.size()<<'\\n';
133     for(auto p:ConvexHull) cout<<p.x<<' ' <<p.y<<'\\n';
134 }

```

16 進階樹

16.1 樹重心分治

```

1 //CSES Fixed-Length Paths I
2 #include <bits/stdc++.h>
3 #define int long long
4 #define pb push_back
5 using namespace std;
6 int n, k;
7 array<bool, 200004> vis;
8 array<int, 200004> S, M, cnt;
9 array<vector<int>, 200004> T, C;
10 vector<int> leaf;
11 int dfsiz(int u){
12     if(vis[u]) return 0;
13     int tmp;
14     leaf.pb(u);
15     vis[u] = 1;
16     S[u] = 1;
17     M[u] = 0;
18     for(int v : T[u]){
19         tmp = dfsiz(v);
20         M[u] = max(M[u], tmp);
21         S[u] += tmp;
22     }
23     return S[u];
24 }
25 int cut(int root){
26     leaf.clear();
27     int cen, s;
28     dfsiz(root);
29     s = leaf.size();
30     for(int u : leaf){
31         if(max(M[u], s - S[u]) <= s / 2) cen = u;
32         vis[u] = 0;

```

```

33     }
34     vis[cen] = 1;
35     for(int v : T[cen]){
36         if(!vis[v]) C[cen].pb(cut(v));
37     }
38     return cen;
39 }
40 int dfs(int u, int pre, int s){
41     if(vis[u] || s > k) return 0;
42     int sum = 0;
43     sum += cnt[k - s];
44     cnt[s]++;
45     for(int v : T[u]){
46         if(v == pre) continue;
47         sum += dfs(v, u, s + 1);
48     }
49     return sum;
50 }
51 int path(int u){
52     int ans = 0, tmp;
53     ans += dfs(u, u, 0);
54     for(int i = 0; i <= k && cnt[i]; i++){
55         cnt[i] = 0;
56         vis[u] = 1;
57         for(int v : T[u]){
58             tmp = dfs(v, u, 1);
59             ans -= tmp;
60             for(int i = 1; i <= k && cnt[i]; i++){
61                 cnt[i] = 0;
62             }
63             for(int v : C[u]){
64                 ans += path(v);
65             }
66             return ans;
67         }
68     }
69     signed main(){
70         int a, b, c;
71         cin >> n >> k;
72         for(int i = 1; i < n; i++){
73             cin >> a >> b;
74             T[a].pb(b);
75             T[b].pb(a);
76         }
77         c = cut(1);
78         for(bool &v : vis) v = 0;
79         cout << path(c);
80         return 0;
81     }
82 //CSES Fixed-Length Paths II
83 //樹重心分治+BIT
84 #include <bits/stdc++.h>
85 #define int long long
86 #define pb push_back
87 using namespace std;
88 int n, k1, k2, d;
89 array<bool, 200004> vis;
90 array<int, 200004> S, M, BNT;
91 array<vector<int>, 200004> T, C;
92 vector<int> leaf, see;
93 void update(int p){
94     p++;
95     for(; p <= n; p += p & -p){
96         if(!BNT[p]) see.pb(p);
97         BNT[p]++;

```

```

98     }
99 }
100 int query(int p){
101     p++;
102     if(p <= 0) return 0;
103     int sum = 0;
104     for(; p > 0; p -= p & -p) sum += BNT[p];
105     return sum;
106 }
107 int dfsiz(int u){
108     if(vis[u]) return 0;
109     int tmp;
110     leaf.pb(u);
111     vis[u] = 1;
112     S[u] = 1;
113     M[u] = 0;
114     for(int v : T[u]){
115         tmp = dfsiz(v);
116         S[u] += tmp;
117         M[u] = max(M[u], tmp);
118     }
119     return S[u];
120 }
121 int dfs(int u, int pre, int s){
122     if(vis[u] || s > k2) return 0;
123     int sum = 0;
124     sum += query(k2 - s) - query(k1 - s - 1);
125     update(s);
126     for(int v : T[u]){
127         if(v == pre) continue;
128         sum += dfs(v, u, s + 1);
129     }
130     return sum;
131 }
132 int cut(int root){
133     int cen, s, ans = 0;
134     leaf.clear();
135     dfsiz(root);
136     s = leaf.size();
137     for(int u : leaf){
138         if(max(M[u], s - S[u]) <= s / 2) cen = u;
139         vis[u] = 0;
140     }
141     ans += dfs(cen, cen, 0);
142     for(int s : see) BNT[s] = 0;
143     see.clear();
144     vis[cen] = 1;
145     for(int v : T[cen]){
146         ans -= dfs(v, cen, 1);
147         for(int s : see) BNT[s] = 0;
148         see.clear();
149     }
150     for(int v : T[cen]){
151         if(!vis[v]) ans += cut(v);
152     }
153     return ans;
154 }
155 signed main(){
156     cin.tie(0), cout.tie(0), ios::sync_with_stdio(0);
157     int a, b;
158     cin >> n >> k1 >> k2;
159     for(int i = 1; i < n; i++){
160         cin >> a >> b;

```

```

159     T[a].pb(b);
160     T[b].pb(a);
161 }
162 cout << cut(1);
163 return 0;
164 }

```

16.2 樹鍊剖分

```

1 //CSES Path Queries
2 //樹鍊剖分+BIT
3 #include <bits/stdc++.h>
4 #define int long long
5 #define pb push_back
6 using namespace std;
7 struct BIT{
8     vector<int> bit;
9     void update(int p, int v){
10         for(; p < bit.size(); p += p & -p)
11             bit[p] += v;
12     }
13     int query(int p){
14         int sum = 0;
15         for(; p > 0; p -= p & -p) sum += bit[p];
16         return sum;
17     }
18 };
19 int cnt = 1;
20 array<int, 200004> V, H, P, S, D, pre, C;
21 array<vector<int>, 200004> T;
22 array<BIT, 200004> B;
23 int dfsiz(int u, int p, int dep){
24     int tmp, siz = 1, mx = 0;
25     D[u] = dep;
26     pre[u] = p;
27     for(int v : T[u]){
28         if(v == p) continue;
29         tmp = dfsiz(v, u, dep + 1);
30         siz += tmp;
31         if(tmp > mx){
32             mx = tmp;
33             S[u] = v;
34         }
35     }
36     return siz;
37 }
38 void cut(int u, int h, int c, int p){
39     H[u] = h;
40     C[u] = c;
41     P[u] = p;
42     if(p == 1) B[c].bit.pb(0);
43     B[c].bit.pb(0);
44     for(int v : T[u]){
45         if(v == pre[u]) continue;
46         if(v == S[u]) cut(v, h, c, p + 1);
47         else cut(v, v, ++cnt, 1);
48     }
49 }
50 int path(int s){
51     int ans = 0;
52     while(s){
53         ans += B[C[s]].query(P[s]);

```

```

53     s = pre[H[s]];
54 }
55 return ans;
56 }
57 signed main(){
58     int n, q, a, b, t, s, x;
59     cin >> n >> q;
60     for(int i = 1; i <= n; i++){
61         cin >> V[i];
62     }
63     for(int i = 1; i < n; i++){
64         cin >> a >> b;
65         T[a].pb(b);
66         T[b].pb(a);
67     }
68     dfsiz(1, 0, 1);
69     cut(1, 1, 1, 1);
70     for(int i = 1; i <= n; i++){
71         B[C[i]].update(P[i], V[i]);
72     }
73     while(q--){
74         cin >> t >> s;
75         if(t == 1){
76             cin >> x;
77             B[C[s]].update(P[s], x - V[s]);
78             V[s] = x;
79         }else cout << path(s) << "\n";
80     }
81     return 0;
82 }
83 //CSES Path Queries II
84 #include <bits/stdc++.h>
85 #define mid ((l + r) >> 1)
86 #define pb push_back
87 using namespace std;
88 struct seg{
89     int val;
90     seg *lc, *rc;
91     seg(){val = 0; lc = rc = nullptr;}
92     void pull(){
93         val = max(lc? lc->val : 0, rc? rc->val : 0);
94     }
95 }
96 void update(int x, int v, int l, int r){
97     if(l == r){
98         val = v;
99         return;
100     }
101     if(x <= mid){
102         if(!lc) lc = new seg;
103         lc->update(x, v, l, mid);
104     }else{
105         if(!rc) rc = new seg;
106         rc->update(x, v, mid + 1, r);
107     }
108     pull();
109 }
110 int query(int ql, int qr, int l, int r){
111     if(ql > r || qr < l) return 0;
112     if(ql <= l && qr >= r) return val;
113     return max(lc? lc->query(ql, qr, l, mid) : 0, rc? rc->query(ql, qr, mid + 1, r) : 0);
114 }
115 };

```

```

116 int cnt = 1, n;
117 array<int, 200004> P, C, D, pre, H, M, V, Z;
118 array<vector<int>, 200004> T;
119 array<seg*, 200004> S;
120 int dfs(int u, int p, int dep){
121     int s = 1, tmp, mx = 0;
122     D[u] = dep;
123     pre[u] = p;
124     for(int v : T[u]){
125         if(v == p) continue;
126         tmp = dfs(v, u, dep + 1);
127         s += tmp;
128         if(tmp > mx){
129             mx = tmp;
130             M[u] = v;
131         }
132     }
133     return s;
134 }
135 void cut(int u, int h, int c, int p){
136     H[u] = h;
137     C[u] = c;
138     P[u] = p;
139     Z[c] = p;
140     for(int v : T[u]){
141         if(v == pre[u]) continue;
142         if(v == M[u]) cut(v, h, c, p + 1);
143         else cut(v, v, ++cnt, 1);
144     }
145 }
146 int path(int a, int b){
147     int ans = 0;
148     while(H[a] != H[b]){
149         if(D[H[a]] < D[H[b]]) swap(a, b);
150         ans = max(ans, S[C[a]]->query(1, P[a], 1, Z[C[a]]));
151         a = pre[H[a]];
152     }
153     if(D[a] < D[b]) swap(a, b);
154     return max(ans, S[C[a]]->query(P[b], P[a], 1, Z[C[a]]));
155 }
156 signed main(){
157     cin.tie(0), cout.tie(0), ios::sync_with_stdio(0);
158     int n, q, a, b, t, s, x;
159     cin >> n >> q;
160     for(int i = 1; i <= n; i++){
161         cin >> V[i];
162     }
163     for(int i = 1; i < n; i++){
164         cin >> a >> b;
165         T[a].pb(b);
166         T[b].pb(a);
167     }
168     dfs(1, 0, 1);
169     cut(1, 1, 1, 1);
170     for(int i = 1; i <= n; i++){
171         if(!S[C[i]]) S[C[i]] = new seg;
172         S[C[i]]->update(P[i], V[i], 1, Z[C[i]]);
173     }
174     while(q--){
175         cin >> t;
176         if(t == 1){
177             cin >> s >> x;

```

```

178         S[C[s]]->update(P[s], x, 1, Z[C[s]]);
179     }else{
180         cin >> a >> b;
181         cout << path(a, b) << "\n";
182     }
183 }
184 return 0;
185 }

```

16.3 LCA

```

1 //Distance Queries
2 #include <bits/stdc++.h>
3 #define pb push_back
4 using namespace std;
5 int cnt = 0;
6 array<int, 200004> in, out;
7 array<array<int, 20>, 200004> A;
8 array<vector<int>, 200004> T;
9 void dfs(int u){
10     in[u] = ++cnt;
11     for(int v : T[u]){
12         if(in[v]) continue;
13         dfs(v);
14         A[v][0] = u;
15     }
16     out[u] = ++cnt;
17 }
18 void dabo(int n){
19     in[0] = 0, out[0] = 1e9;
20     for(int j = 1; j < 20; j++){
21         for(int i = 1; i <= n; i++){
22             A[i][j] = A[A[i][j - 1]][j - 1];
23         }
24     }
25 }
26 int LCA(int a, int b){
27     int dis = 0;
28     for(int i = 19; i >= 0; i--){
29         if(in[A[a][i]] >= in[b] || out[A[a][i]] <= out[b]){
30             dis += 1 << i;
31             a = A[a][i];
32         }
33     }
34     if(in[a] > in[b] || out[a] < out[b]){
35         dis++;
36         a = A[a][0];
37     }
38     for(int i = 19; i >= 0; i--){
39         if(in[A[b][i]] >= in[a] || out[A[b][i]] <= out[a]){
40             dis += 1 << i;
41             b = A[b][i];
42         }
43     }
44     return dis;
45 }
46 signed main(){
47     int n, q, a, b;
48     cin >> n >> q;
49     for(int i = 1; i <= n; i++){

```

```

50         cin >> a >> b;
51         T[a].pb(b);
52         T[b].pb(a);
53     }
54     dfs(1);
55     dabo(n);
56     while(q--){
57         cin >> a >> b;
58         cout << LCA(a, b) << "\n";
59     }
60     return 0;
61 }

```

16.4 樹壓平

```

1 //CSES Subtree Queries
2 //樹壓平+BIT
3 #include <bits/stdc++.h>
4 #define int long long
5 #define pb push_back
6 using namespace std;
7 int cnt = 0;
8 array<int, 200004> BIT, L, P, V;
9 array<vector<int>, 200004> T;
10 void update(int p, int v){
11     for(; p < 200004; p += p & -p) BIT[p] += v;
12 }
13 int query(int p){
14     int sum = 0;
15     for(; p > 0; p -= p & -p) sum += BIT[p];
16     return sum;
17 }
18 int press(int u, int pre){
19     L[u] = 1e9;
20     for(int v : T[u]){
21         if(v == pre) continue;
22         L[u] = min(L[u], press(v, u));
23     }
24     P[u] = ++cnt;
25     return L[u] = min(L[u], P[u]);
26 }
27 signed main(){
28     int n, q, a, b, t, s, x;
29     cin >> n >> q;
30     for(int i = 1; i <= n; i++){
31         cin >> V[i];
32     }
33     for(int i = 1; i < n; i++){
34         cin >> a >> b;
35         T[a].pb(b);
36         T[b].pb(a);
37     }
38     press(1, 0);
39     for(int i = 1; i <= n; i++){
40         update(P[i], V[i]);
41     }
42     while(q--){
43         cin >> t >> s;
44         if(t == 1){
45             cin >> x;
46             update(P[s], x - V[s]);
47             V[s] = x;

```

```
48 |         }else cout << query(P[s]) - query(L[
49 |             s] - 1) << "\n";
50 |     }
51 |     return 0;
    }
```

ACM ICPC Team Reference - Angry Crow Takes Flight!

Contents

1	Computational Geometry	1
1.1	最近點對	1
1.2	Geometry	1
2	DP	3
2.1	整體二分	3
2.2	LineContainer	3
2.3	斜率優化	3
2.4	basic DP	3
2.5	DP on Graph	4
2.6	單調隊列優化	4
3	DP 優化	4
3.1	斜率優化	4
3.2	四邊形優化 (Knuth 優化)	5
3.3	分治優化	5
3.4	單調隊列優化	6
4	Data Structure	6
4.1	sparse table	6
4.2	BinaryTrie	6
4.3	BIT	7
4.4	Dynamic Segment Tree	7

4.5	掃描線 + 線段樹	7
4.6	Persistent DSU	8
4.7	DSU	8
4.8	陣列上 Treap	8
4.9	monotonic stack	8
4.10	Kruskal	8
4.11	Lazytag Segment Tree	8
4.12	2D BIT	9
4.13	monotonic queue	9
4.14	Prim	9
4.15	回滾並查集	9
4.16	TimingSegmentTree	9
4.17	SegmentTree	9
4.18	Hash	10
4.19	Persistent Segment Tree	10
4.20	當作 BST 用的 Treap	10
5	Flow	10
5.1	Property	10
5.2	Gomory Hu	10
5.3	MinCostMaxFlow	10
5.4	dinic	11
5.5	ISAP with cut	12
5.6	biGraph	12
5.7	dinic	13
6	Graph	13
6.1	橋連通分量	13
6.2	SPFA	13
6.3	最大團	14
6.4	判斷平面圖	14
6.5	雙連通分量 & 割點	14
6.6	枚舉極大團 Bron-Kerbosch	14
6.7	Floyd Warshall	15
6.8	Dominator tree	15
6.9	判斷二分圖	15
6.10	Bellman Ford	15

6.11	Dijkstra	15
6.12	SCC	15
6.13	判斷環	16
6.14	2-SAT	16
7	Math	16
7.1	InvGCD	16
7.2	FastPow	16
7.3	中國剩餘定理	16
7.4	ExtendGCD	16
7.5	質因數分解	16
7.6	Theorem	16
7.7	FloorSum	17
7.8	Numbers	17
7.9	LinearSieve	17
7.10	ModInt	17
7.11	GeneratingFunctions	18
7.12	FFT	18
8	RMQ	18
8.1	Doubling RMQ	18
8.2	SegTree	18
8.3	Treap	19
9	Square root decomposition	19
9.1	MoAlgo	19
10	Tree	20
10.1	Tree centroid	20
10.2	HLD	20
10.3	POJ Tree	20
10.4	HeavyLight	20
10.5	centroidDecomposition	20
10.6	link cut tree	21
10.7	LCA	21
10.8	Tree diameter	22
10.9	樹壓平	22

11	string	22
11.1	KMP	22
11.2	reverseBWT	22
11.3	Z	22
11.4	Trie	22
11.5	Rolling Hash	23
11.6	suffix array lcp	23
11.7	AC 自動機	24
11.8	minimal string rotation	24
11.9	manacher	24
12	tools	24
12.1	Template	24
12.2	Counting Sort	25
12.3	HashMap	25
12.4	Bsearch	25
12.5	relabel	25
12.6	TenarySearch	25
12.7	DuiPai	25
12.8	bitset	25
13	圖論	25
13.1	SCC	25
14	字串	26
14.1	suffix array	26
14.2	Trie	28
14.3	KMP&Z	29
14.4	Hash	29
15	計算幾何	30
15.1	Convex Hull	30
16	進階樹	31
16.1	樹重心分治	31
16.2	樹鍊剖分	31
16.3	LCA	32
16.4	樹壓平	32

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 }
```