1. **杂项**
   1. C++输入挂
2. bool IOerror**=**0**;**
3. inline char nc**(){**
4. static char buf**[**100000**],\***p1**=**buf**+**100000**,\***pend**=**buf**+**100000**;**
5. **if** **(**p1**==**pend**){**
6. p1**=**buf**;** pend**=**buf**+**fread**(**buf**,**1**,**100000**,**stdin**);**
7. **if** **(**pend**==**p1**){**IOerror**=**1**;return** **-**1**;}**
8. **}**
9. **return** **\***p1**++;**
10. **}**
11. bool blank**(**char ch**){return** ch**==**' '**||**ch**==**'\n'**||**ch**==**'\r'**||**ch**==**'\t'**;}**
12. bool read**(**int **&**x**){**
13. bool sign**=**0**;** char ch**=**nc**();** x**=**0**;**
14. **for** **(;**blank**(**ch**);**ch**=**nc**());**
15. **if** **(**IOerror**)return** 0**;**
16. **if** **(**ch**==**'-'**)**sign**=**1**,**ch**=**nc**();**
17. **for** **(;**ch**>=**'0'**&&**ch**<=**'9'**;**ch**=**nc**())**x**=**x**\***10**+**ch**-**'0'**;**
18. **if** **(**sign**)**x**=-**x**;**
19. **return** 1**;**
20. **}**
21. bool read**(**char **\***s**){**
22. char ch**=**nc**();**
23. **for** **(;**blank**(**ch**);**ch**=**nc**());**
24. **if** **(**IOerror**)return** 0**;**
25. **for** **(;!**blank**(**ch**)&&!**IOerror**;**ch**=**nc**())\***s**++=**ch**;**
26. **\***s**=**0**;**
27. **return** 1**;**
28. **}**
29. std**::**ios**::**sync\_with\_stdio**(false),** cin**.**tie**(**0**);**
    1. C++输出浮点数
30. cout **<<** fixed **<<** setprecision**(**0**)** **<<** ans **<<** endl
    1. JAVA 输入输出挂
31. public class Main **{**
32. static StreamTokenizer in **=** **new** StreamTokenizer**(new** BufferedReader**(new** InputStreamReader**(**System**.**in**)));**
33. static PrintWriter out **=** **new** PrintWriter**(new** OutputStreamWriter**(**System**.**out**));**
34. public static void main**(**String**[]** args**)** throws IOException **{**
35. // TODO Auto-generated method stub
36. **while** **(**in**.**nextToken**()** **!=** StreamTokenizer**.**TT\_EOF**)** **{**
37. // 用in.nextToken()读下一个标记，StreamTokenizer.TT\_EOF这个是个参数，就是EOF
38. int a **=** **(**int**)** in**.**nval**;**
39. // 读进来的是double型的，所以要转成int
40. in**.**nextToken**();**
41. // 读入字符串b值
42. String b **=** in**.**sval**;**
43. out**.**println**(**a **+** b**);**
44. **}**
45. out**.**flush**();**
46. **}**
47. **}**
    1. Strtok
48. char buffer**[**20**]** **=** "Fred,John,Ann"**;**
49. char **\***p**[**3**];**
50. char **\***buf **=** buffer**;**
51. **while((**p**[**in**]** **=** strtok**(**buf**,** ","**))** **!=** **NULL)** **{**
52. in**++;**
53. buf **=** **NULL;**
54. **}**
    1. Spilt
55. 特殊情况有 **\*** **^** **:** **|** **.** \
56. String s**=**"a\b\c"**;**
57. String**[]** splitAddress**=**s**.**split**(**"\\"**);**
    1. 求int/ll二进制中1/0的个数
58. \_\_builtin\_popcount//1 int
59. \_\_builtin\_popcountll/1 ll
60. \_\_builtin\_ctz //0 int
61. \_\_builtin\_ctzll //0 ll
    1. Rand
62. mt19937 mrand(random\_device{}());
    1. Vector离散化
63. vector**<**int**>**v**;**
64. int getid**(**int x**){return** lower\_bound**(**v**.**begin**(),**v**.**end**(),**x**)-**v**.**begin**()+**1**;}**
65. sort**(**v**.**begin**(),**v**.**end**()),**v**.**erase**(**unique**(**v**.**begin**(),**v**.**end**()),**v**.**end**());**
    1. Vector骚排序方式
66. vector**<**int**>** id**(**n**);** iota**(**id**.**begin**(),**id**.**end**(),**1**);**//1-n填入vecotr
67. sort**(**id**.**begin**(),**id**.**end**(),** **[**k**]** **(**int i**,** int j**)** **{**//k是传参
68. **return** i**+**k**>**j**+**k**;**//定义排序方式
69. **});**
    1. 防止Unordered\_map被卡
70. //方法1：
71. mp.max\_load\_factor(0.25);
72. //方法2：
73. struct ch {
74. static uint64\_t splitmix64(uint64\_t x) {
75. x += 0x9e3779b97f4a7c15;
76. x = (x ^ (x >> 30)) \* 0xbf58476d1ce4e5b9;
77. x = (x ^ (x >> 27)) \* 0x94d049bb133111eb;
78. return x ^ (x >> 31);
79. }
80. size\_t operator()(uint64\_t x) const {
81. static const uint64\_t FIXED\_RANDOM = chrono::steady\_clock::now().time\_since\_epoch().count();
82. return splitmix64(x + FIXED\_RANDOM);
83. }
84. };
85. unordered\_map<int,int,ch>safe\_map;
    1. 手写hashmap
86. #define rep(ii,a,b) for(int ii=a;ii<=b;++ii)
87. template<typename key,typename val>
88. class hash\_map{public:
89. struct node{key u;val v;int next;};
90. vector<node> e;
91. int head[maxsz],nume,numk,id[maxsz];
92. bool count(key u){
93. int hs=u%maxsz;
94. for(int i=head[hs];i;i=e[i].next)
95. if(e[i].u==u) return 1;
96. return 0;
97. }
98. val& operator[](key u){
99. int hs=u%maxsz;
100. for(int i=head[hs];i;i=e[i].next)
101. if(e[i].u==u) return e[i].v;
102. if(!head[hs])id[++numk]=hs;
103. if(++nume>=e.size())e.resize(nume<<1);
104. return e[nume]=(node){u,0,head[hs]},head[hs]=nume,e[nume].v;
105. }
106. void clear(){
107. rep(i,0,numk)head[id[i]]=0;
108. numk=nume=0;
109. }
110. };
111. **数据结构**
     1. STL
        1. Nth\_element
112. int s=nth\_element(array, array+6, array+len); //排序第6个元素
113. //直接取出第几大，或者要求出前多少大的优先队列维护
     * 1. Bitset
114. bitset**<**1000**>**b**;**
115. b**.**\_Find\_first**();**//找到第一个1位置
116. b**.**\_Find\_next**(**5**);**//找到5之后第一个1位置
117. //手写count
118. bitset**<**1000**>**R**;**
119. auto data **=** **(**int**\*)&**R**;**
120. int ans **=** 0**;**
121. **if** **(**r **-** l **<=** 100**)** **{**
122. **for** **(**int i **=** l**;** i **<=** r**;** **++**i**)**
123. ans **+=** R**[**i**];**
124. **}** **else** **{**
125. **while** **(**l **&** 31**)** **{**
126. ans **+=** R**[**l**];** **++**l**;**
127. **}**
128. **++**r**;**
129. **while** **(**r **&** 31**)** **{**
130. **--**r**;** ans **+=** R**[**r**];**
131. **}**
132. l **>>=** 5**;**
133. r **>>=** 5**;**
134. **for** **(**int i **=** l**;** i **<** r**;** **++**i**)**
135. ans **+=** \_\_builtin\_popcount**(**data**[**i**]);**
136. **}**
137. //bitset实现字符串匹配
138. //给定两个字符串S和T，求T在S中出现几次。T中可能有“?”字符，这个字符可以匹配任何字符。
139. scanf**(**"%s\n%s"**,** S **+** 1**,** T **+** 1**);**
140. N **=** strlen**(**S **+** 1**);** M **=** strlen**(**T **+** 1**);**
141. **for(**int i **=** 1**;** i **<=** N**;** i**++)** B**[**S**[**i**]** **-** 'a'**].**set**(**i**);**
142. bitset**<**MAXN**>** ans**;** ans**.**set**();**
143. **for(**int i **=** 1**;** i **<=** M**;** i**++)** **if(**T**[**i**]** **!=** '?'**)** ans **&=** **(**B**[**T**[**i**]** **-** 'a'**]** **>>** **(**i **-** 1**));**
144. **for(**int i **=** 1**;** i **<=** N **-** M **+** 1**;** i**++)** **if(**ans**[**i**]** **==** 1**)** cnt**++;**
     1. 树状数组
        1. 一维树状数组
145. //若区间修改&单点查询 则 add(l,k) add(r+1,-k)
146. int c[N];
147. void add(int k,int num) {
148. for(; k<N; c[k]+=num,k+=k&(-k));
149. }
150. int Sum(int k) {
151. int sum=0;
152. for(; k>0; sum+=c[k],k-=k&(-k));
153. return sum;
154. }
     * 1. 二维树状数组 区间修改 单点查询
155. ll sum(int x,int y) {
156. ll s=0;
157. for(;x;x-=x&-x)
158. for(int yy=y;yy;yy-=yy&-yy)
159. s+=c[x][yy];
160. return s;
161. }
162. void add(int x,int y,ll v) {
163. for(;x<=n;x+=x&-x)
164. for(int yy=y;yy<=n;yy+=yy&-yy)
165. c[x][yy]+=v;
166. }
167. 从(x1,y1)到(x2,y2)
168. add(x1,y1,w);
169. add(x1,y2+1,-w);
170. add(x2+1,y1,-w);
171. add(x2+1,y2+1,w);
     * 1. 三维树状数组 区间求和 单点查询
172. ll sum(int x,int y,int z) {
173. ll s=0;
174. for(;x;x-=x&-x)
175. for(int yy=y;yy;yy-=yy&-yy)
176. for(int zz=z;zz;zz-=zz&-zz)
177. s+=c[x][yy][zz];
178. return s;
179. }
180. void add(int x,int y,int z,ll v) {
181. for(;x<=n;x+=x&-x)
182. for(int yy=y;yy<=n;yy+=yy&-yy)
183. for(int zz=z;zz<=n;zz+=zz&-zz)
184. c[x][yy][zz]+=v;
185. }
186. add(x1,y1,z1,w);
187. add(x1,y2+1,z1,-w);
188. add(x2+1,y1,z1,-w);
189. add(x2+1,y2+1,z1,w);
190. add(x1,y1,z2+1,-w);
191. add(x1,y2+1,z2+1,w);
192. add(x2+1,y1,z2+1,w);
193. add(x2+1,y2+1,z2+1,-w);
     * 1. 树状数组第K小
194. void upd(int i,int v) {
195. for(;i<N;i+=i&-i)
196. c[i]+=v;
197. }
198. int kth(int x) {//kthsmall
199. int sum=0;
200. for (int i=20;i>=0;i--) {
201. if(sum+(1<<i)<N&&c[sum+(1<<i)]<x) {
202. x-=c[sum+(1<<i)];
203. sum+=1<<i;
204. }
205. }
206. return sum+1;
207. }
     1. 并查集
        1. 普通并查集
208. int pa[M];
209. int fid(int x) {
210. return x==pa[x]?x:pa[x]=fid(pa[x]);
211. }
212. void un(int a, int b) {
213. int a1 = fid(a);
214. int b1 = fid(b);
215. if(a1!=b1) pa[a1] = b1;
216. }
     * 1. 可拆分并查集
217. int f[N];
218. void INIT(int size)
219. { for(int i=0;i<=size;i++) f[i]=i; }
220. int findf(int x)
221. { while(f[x]!=x) x=f[x]; return x; }
222. void setroot(int x)
223. { if(f[x]!=x) setroot(f[x]); f[x]=f[f[x]]=x; }
224. void link(int a,int b)
225. { setroot(a); f[a]=b; }
226. void cut(int a,int b)
227. { setroot(a); f[b]=b; }
     * 1. 可撤回并查集
228. #include <iostream>
229. #include <algorithm>
230. #include <stdio.h>
231. using namespace std;
232. typedef long long LL;
233. #define dd c=getchar()
234. inline LL read(){LL a=0,b=1;char dd;while(!isdigit(c)&&c!='-')dd;
235. if(c=='-'){b=-b;dd;}while(isdigit(c)){a=a\*10+c-'0';dd;}return a\*b;}
236. #undef dd
237. template <int n>
238. class vervec{
239. LL tree[20\*n],ls[20\*n],rs[20\*n],hd[n],hd1[2\*n],tt,t,nh;LL\*a;
240. LL&ins(LL nod,LL l,LL r,LL x){
241. if(l==r){return tree[nod];}
242. int mid=l+r>>1;
243. if(x<=mid){tree[++tt]=tree[ls[nod]];ls[tt]=ls[ls[nod]];rs[tt]=rs[ls[nod]];
244. ls[nod]=tt;return ins(tt,l,mid,x);}
245. if(x>mid){tree[++tt]=tree[rs[nod]];ls[tt]=ls[rs[nod]];rs[tt]=rs[rs[nod]];
246. rs[nod]=tt;return ins(tt,mid+1,r,x);}
247. }
248. void chit(){LL x=hd[nh];hd[++t]=++tt;tree[tt]=tree[x];
249. ls[tt]=ls[x];rs[tt]=rs[x];nh=t;}
250. int build(LL l,LL r){
251. LL nod=++tt;if(l==r){tree[nod]=a[l];ls[nod]=rs[nod]=0;return nod;}
252. LL mid=l+r>>1;
253. ls[nod]=build(l,mid);
254. rs[nod]=build(mid+1,r);
255. return nod;
256. }
257. void init(){tt=0;hd[nh=0]=1;build(1,n);}
258. LL query(LL nod,LL l,LL r,LL x){
259. if(l==r)return tree[nod];
260. LL mid=l+r>>1;
261. if(x<=mid)return query(ls[nod],l,mid,x);
262. else if(x>mid)return query(rs[nod],mid+1,r,x);
263. }
264. public:
265. vervec(){tt=0;t=0;nh=1;}
266. void init(LL\*x){a=x;init();}
267. void mark(LL x){hd1[x]=nh;}
268. LL&operator[](LL x){chit();return ins(hd[nh],1,n,x);}
269. LL val(LL x){return query(hd[nh],1,n,x);}
270. void rever(LL x){nh=hd1[x];}
271. };
272. vervec<120003>f,c;
273. LL getf(LL v){
274. LL fa=f.val(v);
275. if(fa==v)return v;
276. return getf(fa);
277. }
278. void add(LL a,LL b){
279. a=getf(a);
280. b=getf(b);
281. if(a==b)return;
282. LL va=c.val(a),vb=c.val(b);
283. if(va==vb){
284. f[a]=b;
285. c[b]++;
286. }else if(va<vb){
287. f[a]=b;
288. }else{
289. f[b]=a;
290. }
291. }
292. LL sol[120003],n,m,x,y,z;
293. void init(LL n){
294. for(LL i=1;i<=n;i++)sol[i]=i;
295. f.init(sol);
296. for(LL i=1;i<=n;i++)sol[i]=1;
297. c.init(sol);
298. }
299. //1 a b 合并a,b所在集合
300. //2 k 回到第k次操作之后的状态(查询算作操作)
301. //3 a b 询问a,b是否属于同一集合，是则输出1否则输出0
302. int main(){
303. n=read();
304. m=read();
305. init(n);
306. f.mark(0);
307. c.mark(0);
308. for(LL i=1;i<=m;i++){
309. x=read();
310. if(x==1){
311. x=read();
312. y=read();
313. add(x,y);
314. }else if(x==2){
315. y=read();
316. f.rever(y);
317. c.rever(y);
318. }else{
319. x=read();
320. y=read();
321. puts(getf(x)==getf(y)?"1":"0");
322. }
323. f.mark(i);
324. c.mark(i);
325. }
326. return 0;
327. }
     1. 线段树
        1. 普通线段树
328. #define lson l , m , rt << 1
329. #define rson m + 1 , r , rt << 1 | 1
330. ll add[N<<2],sum[N<<2];
331. int n;
332. void up(int rt) {
333. ;
334. }
335. void down(int rt,int m) {
336. if (add[rt]) {
337. add[rt] = 0;
338. }
339. }
340. void build(int l,int r,int rt) {
341. sum[rt]=add[rt]=0;
342. if (l==r) return ;
343. int m=(l+r)>>1;
344. build(lson),build(rson);
345. up(rt);
346. }
347. void edit(int L,int R,ll c,int l=1,int r=n,int rt=1) {
348. if (L<=l&&r<=R) {
349. return ;
350. }
351. down(rt,r-l+1);
352. int m=(l+r)>>1;
353. if(L<=m) edit(L,R,c,lson);
354. if(m<R) edit(L,R,c,rson);
355. up(rt);
356. }
357. ll qry(int L,int R,int l=1,int r=n,int rt=1) {
358. if (L<=l && r<=R) return sum[rt];
359. down(rt,r-l+1);
360. int m=(l+r)>>1;
361. ll ret=0;
362. if (L<=m) ret+=qry(L,R,lson);
363. if (m<R) ret+=qry(L,R,rson);
364. return ret;
365. }
     * 1. 动态开点线段树
366. int ls[N\*50],rs[N\*50],sum[N\*50],tot,root;
367. void edit(int pos,int c,int l,int r,int &rt) {
368. if(!rt) rt=++tot;
369. if(l==r) {
370. sum[rt]+=c;
371. return ;
372. }
373. int mid=l+r>>1;
374. if(pos<=mid) edit(pos,c,l,mid,ls[rt]);
375. else edit(pos,c,mid+1,r,rs[rt]);
376. sum[rt]=sum[ls[rt]]+sum[rs[rt]];
377. }
378. int qry(int L,int R,int l,int r,int rt) {
379. if(!rt) return 0;
380. if(L<=l && r<=R) return sum[rt];
381. int res=0;
382. int mid=l+r>>1;
383. if(L<=mid) res+=qry(L,R,l,mid,ls[rt]);
384. if(mid<R) res+=qry(L,R,mid+1,r,rs[rt]);
385. return res;
386. }
     * 1. 线段树合并 //记得总结新建节点的线段树合并
387. const int N=100010;
388. int ls[N\*50],rs[N\*50],sum[N\*50],tot,rt[N];
389. void edit(int pos,int c,int l,int r,int &rt) {
390. if(!rt) rt=++tot;
391. if(l==r) {
392. sum[rt]+=c;
393. return ;
394. }
395. int mid=l+r>>1;
396. if(pos<=mid) edit(pos,c,l,mid,ls[rt]);

else edit(pos,c,mid+1,r,rs[rt]);

1. sum[rt]=sum[ls[rt]]+sum[rs[rt]];
2. }
3. int qry(int L,int R,int l,int r,int rt) {
4. if(!rt) return 0;
5. if(L<=l && r<=R) return sum[rt];
6. int res=0;
7. int mid=l+r>>1;
8. if(L<=mid) res+=qry(L,R,l,mid,ls[rt]);
9. if(mid<R) res+=qry(L,R,mid+1,r,rs[rt]);
10. return res;
11. }
12. int merge(int x, int y) {
13. if(!x) return y;
14. if(!y) return x;
15. sum[x] += sum[y] ;
16. ls[x] = merge(ls[x], ls[y]) ;
17. rs[x] = merge(rs[x], rs[y]) ;
18. return x;
19. }
    1. 分块
20. int block,belong[N],num,l[N],r[N],a[N],Max[N];
21. int n,q;
22. void build() {
23. for(int i=1; i<=n; i++)
24. scanf("%d",&a[i]);
25. block=sqrt(n);
26. num=n/block;
27. memset(Max,0,sizeof(int)\*(num+4));
28. if(n%block) num++;
29. for(int i=1; i<=num; i++)
30. l[i]=(i-1)\*block+1,r[i]=i\*block;
31. r[num]=n;
32. for(int i=1; i<=n; i++)
33. belong[i]=(i-1)/block+1;
34. for(int i=1; i<=num; i++)
35. for(int j=l[i]; j<=r[i]; j++)
36. Max[i]=max(Max[i],a[j]);
37. }
38. void update(int x,int y) {
39. a[x]=y,Max[belong[x]]=0;
40. for(int i=l[belong[x]]; i<=r[belong[x]]; i++)
41. Max[belong[x]]=max(Max[belong[x]],a[i]);
42. }
43. int ask(int x,int y) {
44. int ans=0;
45. if(belong[x]==belong[y]) {
46. for(int i=x; i<=y; i++)
47. ans=max(a[i],ans);
48. return ans;
49. }
50. for(int i=x; i<=r[belong[x]]; i++)
51. ans=max(ans,a[i]);
52. for(int i=belong[x]+1; i<belong[y]; i++)
53. ans=max(ans,Max[i]);
54. for(int i=l[belong[y]]; i<=y; i++)
55. ans=max(ans,a[i]);
56. return ans;
57. }
    1. 莫队
       1. 普通莫队
58. /\*
59. 小B有一个序列，包含N个1~K之间的整数。他一共有M个询问，
60. 每个询问给定一个区间[L..R]，求Sigma(c(i)^2)的值,其中i的值从1到K，
61. 其中c(i)表示数字i在[L..R]中的重复次数。小B请你帮助他回答询问。
62. \*/
63. #include <bits/stdc++.h>
64. using namespace std;
65. const int N=50007,M=50007;
66. int sz=250;
67. struct Node {
68. int l,r,id;
69. friend bool operator < ( Node a, Node b ) {
70. return (a.l/sz)== (b.l/sz) ? a.r < b.r : a.l < b.l ;
71. }
72. }q[M];
73. int pos[N],sum[N],a[N],ans[N];
74. int n,m,k,Ans=0;
75. void add ( int pos ) {
76. Ans += (sum[a[pos]]++)<<1|1 ;
77. }
78. void del ( int pos ) {
79. Ans -= (--sum[a[pos]])<<1|1 ;
80. }
81. int L=1,R=0;
82. int main() {
83. scanf("%d%d%d",&n,&m,&k);
84. for(int i=1; i<=n; i++)
85. scanf("%d",&a[i]);
86. for(int i=1; i<=m; i++) {
87. scanf("%d%d",&q[i].l,&q[i].r);
88. q[i].id=i;
89. }
90. sort(q+1,q+1+m);
91. for(int i=1; i<=m; i++) {
92. while(L<q[i].l) del(L++);
93. while(L>q[i].l) add(--L);
94. while(R<q[i].r) add(++R);
95. while(R>q[i].r) del(R--);
96. ans[q[i].id]=Ans;
97. }
98. for(int i=1;i<=m;i++)
99. printf("%d\n",ans[i]);
100. }
     * 1. 带修改莫队
101. /\*墨墨购买了一套N支彩色画笔（其中有些颜色可能相同），摆成一排，你需要回答墨墨的提问。墨墨会像你发布如下指令：
102. 1、 Q L R代表询问你从第L支画笔到第R支画笔中共有几种不同颜色的画笔。
103. 2、 R P Col 把第P支画笔替换为颜色Col。为了满足墨墨的要求，你知道你需要干什么了吗？
104. 第1行两个整数N，M，分别代表初始画笔的数量以及墨墨会做的事情的个数。
105. 第2行N个整数，分别代表初始画笔排中第i支画笔的颜色。第3行到第2+M行，每行分别代表墨墨会做的一件事情。
106. 对于每一个Query的询问，你需要在对应的行中给出一个数字，代表第L支画笔到第R支画笔中共有几种不同颜色的画笔。\*/
107. #include<bits/stdc++.h>
108. using namespace std;
109. const int N=50005,M=50005;
110. int col[N],n,m,sum[1000005],be[N],last[N];
111. int ANS,qnum,cnum,ans[N];
112. struct qs{
113. int l,r,t,id;
114. bool friend operator <(qs x,qs y) {
115. if(be[x.l]!=be[y.l])return x.l<y.l;
116. if(be[x.r]!=be[y.r])return x.r<y.r;
117. return x.t<y.t;
118. }
119. }q[M];
120. struct cs{
121. int pos,pre,val;
122. }c[N];
123. void add(int x) {
124. if(!sum[col[x]])
125. ++ANS;
126. ++sum[col[x]];
127. }
128. void del(int x) {
129. --sum[col[x]];
130. if(!sum[col[x]])
131. --ANS;
132. }
133. int l,r,now;
134. void change(int pos,int val){
135. if(pos<=r && pos>=l) {
136. sum[col[pos]]--;
137. if(!sum[col[pos]])ANS--;
138. if(!sum[val])ANS++;
139. sum[val]++;
140. }col[pos]=val;
141. }
142. int main(){
143. scanf("%d%d",&n,&m)
144. int xx=pow(n,2.0/3)+1;
145. char op[10];
146. for(int i=1;i<=n;i++){
147. scanf("%d",&col[i]);
148. last[i]=col[i];
149. be[i]=i/xx+1;
150. }
151. qnum=cnum=0;
152. for(int i=1;i<=m;i++){
153. scanf("%s",op);
154. if(op[0]=='Q'){
155. ++qnum;
156. scanf("%d%d",&q[qnum].l,&q[qnum].r);
157. q[qnum].t=cnum;
158. q[qnum].id=qnum;
159. }else{
160. ++cnum;
161. scanf("%d%d",&c[cnum].pos,&c[cnum].val);
162. c[cnum].pre=last[c[cnum].pos];
163. last[c[cnum].pos]=c[cnum].val;
164. }
165. }
166. sort(q+1,q+qnum+1);
167. r=ANS=now=0;l=1;
168. for(int i=1;i<=qnum;i++){
169. while(l<q[i].l)del(l++);
170. while(l>q[i].l)add(--l);
171. while(r>q[i].r)del(r--);
172. while(r<q[i].r)add(++r);
173. while(now<q[i].t) {
174. now++;
175. change(c[now].pos,c[now].val);
176. }
177. while(now>q[i].t) {
178. change(c[now].pos,c[now].pre);
179. now--;
180. }
181. ans[q[i].id]=ANS;
182. }
183. for(int i=1;i<=qnum;i++)
184. printf("%d\n",ans[i]);
185. return 0;
186. }
     1. 中缀表达式转后缀表达式并计算
187. #include<bits/stdc++.h>
188. using namespace std;
189. struct node {
190. int op,v;
191. char c;
192. };
193. queue<node>q;
194. stack<char>stk;
195. stack<int>sa;
196. vector<node>s;
197. int cg[200];
198. int gao(){
199. int n=s.size();
200. for(int i=0;i<n;i++) {
201. if(s[i].op==0) {
202. q.push(s[i]);
203. }else{
204. if(s[i].c=='(') {
205. stk.push(s[i].c);
206. }else if(s[i].c==')') {
207. while(stk.size()&&stk.top()!='(') {
208. q.push({1,0,stk.top()});
209. stk.pop();
210. }
211. if(stk.size()) stk.pop();
212. }else if(cg[s[i].c]==2) {
213. while(stk.size()&&cg[stk.top()]==2) {
214. q.push({1,0,stk.top()});
215. stk.pop();
216. }
217. stk.push(s[i].c);
218. }else if(cg[s[i].c]==1) {
219. while(stk.size()&&cg[stk.top()]) {
220. q.push({1,0,stk.top()});
221. stk.pop();
222. }
223. stk.push(s[i].c);
224. }
225. }
226. }
227. while (!stk.empty()){
228. q.push({1,0,stk.top()});
229. stk.pop();
230. }
231. while(!q.empty()) {
232. node p=q.front();
233. q.pop();
234. if(p.op==0) sa.push(p.v);
235. else {
236. int a=sa.top();
237. sa.pop();
238. int b=sa.top();
239. sa.pop();
240. if(p.c=='+') sa.push(b+a);
241. if(p.c=='-') sa.push(b-a);
242. if(p.c=='\*') sa.push(b\*a);
243. if(p.c=='/') sa.push(b/a);
244. }
245. }
246. return sa.top();
247. }
248. int main(){
249. //"2\*(2+3)-4/(1+1)";
250. s.resize(15);
251. s[0].op=0,s[0].v=2;
252. s[1].op=1,s[1].c='\*';
253. s[2].op=1,s[2].c='(';
254. s[3].op=0,s[3].v=2;
255. s[4].op=1,s[4].c='+';
256. s[5].op=0,s[5].v=3;
257. s[6].op=1,s[6].c=')';
258. s[7].op=1,s[7].c='-';
259. s[8].op=0,s[8].v=4;
260. s[9].op=1,s[9].c='/';
261. s[10].op=1,s[10].c='(';
262. s[11].op=0,s[11].v=1;
263. s[12].op=1,s[12].c='+';
264. s[13].op=0,s[13].v=1;
265. s[14].op=1,s[14].c=')';
266. cg['\*']=cg['/']=2;
267. cg['+']=cg['-']=1;
268. printf("%d\n",gao());
269. return 0;
270. }
     1. ST表
        1. 一维ST表
271. int mn[22][N];
272. int lg[N];
273. void init(int n){
274. for(int i=1;i<=n;i++) mn[0][i]=height[i];
275. for(int j=1;j<=21;j++)
276. for(int i=1;i+(1<<j)-1<=n;i++)
277. mn[j][i]=min(mn[j-1][i],mn[j-1][i+(1<<(j-1))]);
278. }
279. int query(int l,int r){
280. int k=lg[r-l+1];
281. return min(mn[k][l],mn[k][r-(1<<k)+1]);
282. }
283. int main() {
284. lg[0]=-1;
285. for(int i=1;i<N;i++)
286. lg[i]=lg[i>>1]+1;
287. init(n);
288. }
     * 1. 二维ST表
289. int mx[9][9][310][310],lg[310];
290. void init(int n,int m) {
291. for(int i=1; i<=n; i++)
292. for(int j=1; j<=m; j++)
293. mx[0][0][i][j]=a[i][j];
294. for(int ii=0; ii<=lg[n]; ii++)
295. for(int jj=0; jj<=lg[m]; jj++) if(ii+jj)
296. for(int i=1; i+(1<<ii)-1<=n; i++)
297. for(int j=1; j+(1<<jj)-1<=m; j++)
298. if(ii) mx[ii][jj][i][j] = max(mx[ii-1][jj][i][j],mx[ii-1][jj][i+(1<<(ii-1))][j]);
299. else mx[ii][jj][i][j] = max(mx[ii][jj-1][i][j],mx[ii][jj-1][i][j+(1<<(jj-1))]);
300. }
301. int rmq(int x1,int y1,int x2,int y2) {
302. int k1 = lg[x2-x1+1];
303. int k2 = lg[y2-y1+1];
304. x2=x2-(1<<k1)+1;
305. y2=y2-(1<<k2)+1;
306. return max(max(mx[k1][k2][x1][y1],mx[k1][k2][x1][y2]),max(mx[k1][k2][x2][y1],mx[k1][k2][x2][y2]));
307. }
308. lg[0]=-1;
309. for(int i=1;i<305;i++)
310. lg[i]=lg[i>>1]+1;
     * 1. 区间最小值的下标ST表
311. void init(int n) {
312. for(int i=1;i<=n;i++) st[0][i]=i;
313. for(int j=1;(1<<j)<=n;j++){
314. for(int i=1;i<=n;i++){
315. if((i+(1<<(j-1)))>n||a[st[j-1][i]]<a[st[j-1][i+(1<<(j-1))]]) st[j][i]=st[j-1][i];
316. else st[j][i]=st[j-1][i+(1<<(j-1))];
317. }
318. }
319. }
320. int qry(int l,int r){
321. int f=lg[r-l+1];
322. if(a[st[f][l]]<a[st[f][r-(1<<f)+1]]) return st[f][l];
323. return st[f][r-(1<<f)+1];
324. }
     1. 单调栈
325. int l[N],r[N],n,a[N],t1[N],t2[N],tt;
326. void getlr() {
327. tt=0;
328. for(int i=1;i<=n;i++) if(i==1) {
329. t1[++tt]=i;
330. l[i]=i;
331. }else {
332. for(; tt&&a[t1[tt]]>=a[i]; tt--) ;
333. if(tt) l[i]=t1[tt]+1;
334. else l[i]=1;
335. t1[++tt]=i;
336. }
337. tt=0;
338. for(int i=n;i>=1;i--) if(i==n) {
339. t2[++tt]=i;
340. r[i]=i;
341. }else {
342. for(; tt&&a[t2[tt]]>=a[i]; tt--) ;
343. if(tt) r[i]=t2[tt]-1;
344. else r[i]=n;
345. t2[++tt]=i;
346. }
347. }
     1. 单调队列
348. struct node {
349. int x,y;
350. } v[N]; //x值，y下标
351. int a[N],n,m,mx[N],mn[N];
352. void getmin() {
353. int i,head=1,tail=0;
354. for(i=1;i<m;i++) {
355. while(head<=tail && v[tail].x>=a[i]) tail--;
356. v[++tail].x=a[i],v[tail].y=i;
357. }
358. for(;i<=n;i++) {
359. while(head<=tail && v[tail].x>=a[i]) tail--;
360. v[++tail].x=a[i],v[tail].y=i;
361. while(v[head].y<i-m+1) head++;
362. mn[i-m+1]=v[head].x;
363. }
364. }
365. **字符串**
     1. Hash
366. ull sed = 1331;
367. ull ha[N],h[N];
368. ull get(int l,int r) {
369. return h[r]-h[l-1]\*ha[r-l+1];
370. }
371. ha[0]=1;
372. for(int i=1;i<N;i++) ha[i]=ha[i-1]\*sed;
373. for(int i=1;i<=n;i++) h[i]=h[i-1]\*sed+s[i];
     1. Tire树
374. int tr[N][26],tot;
375. bool k[N];
376. void ins(char \*s) {
377. int len=strlen(s);
378. int rt=0;
379. for(int i=0;i<len;i++) {
380. int id=s[i]-'a';
381. if(!tr[rt][id]) tr[rt][id]=++tot;
382. rt=tr[rt][id];
383. }
384. k[rt]=1;
385. }
386. bool fid(char \*s) {
387. int len=strlen(s);
388. int rt=0;
389. for(int i=0;i<len;i++) {
390. int id=s[i]-'a';
391. if(!tr[rt][id]) return 0;
392. rt=tr[rt][id];
393. }
394. return 1;
395. }
396. void init() {
397. for(int i=0;i<=tot;i++) {
398. k[i]=0;
399. for(int j=0;j<26;j++) tr[i][j]=0;
400. }
401. tot=0;
402. }
     1. KMP
403. char S[N],T[N];
404. void gao(){
405. int j=0;
406. for (int i=2; i<=m; i++) {
407. while(j&&T[i]!=T[j+1]) j=nex[j];
408. if(T[j+1]==T[i]) j++;
409. nex[i]=j;
410. }
411. }
412. int match() {
413. int j=0;
414. for(int i=1; i<=n; i++) {
415. while(j&&T[j+1]!=S[i]) j=nex[j];
416. if (T[j+1]==S[i]) j++;
417. if (j==m) {
418. //cout<<<<endl;
419. return i-m+1;
420. //不匹配其他的可以直接return了
421. j=nex[j];
422. }
423. }
424. return -1;
425. }
     1. EX-KMP
426. inline void Z(char \*s, int n) {
427. for (int i = 1; i <= n; i++) z[i] = 0;
428. z[1] = n;
429. for (int i = 2, l = 0, r = 0; i <= n; i++) {
430. if (i <= r) z[i] = min(z[i-l+1], r - i + 1);
431. while (i + z[i] <= n && s[i+z[i]] == s[z[i]+1]) ++z[i];
432. if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
433. }
434. }
435. inline void exkmp(char \*s, int n, char \*t, int m, int \*p) {
436. Z(t, m);
437. for (int i = 1; i <= n; i++) p[i] = 0;
438. for (int i = 1, l = 0, r = 0; i <= n; i++) {
439. if (i <= r) p[i] = min(z[i-l+1], r - i + 1);
440. while (i + p[i] <= n && s[i+p[i]] == t[p[i]+1]) ++p[i];
441. if (i + p[i] - 1 > r) l = i, r = i + p[i] - 1;
442. }
443. }
     1. AC自动机
        1. 初始版本
444. #include<bits/stdc++.h>
445. using namespace std;
446. typedef long long ll;
447. const int N=1e6+2,M=5e5+2;
448. struct Aho {
449. int ch[M][26],fail[M],cnt[M],q[M];
450. int tot;
451. void init() {tot=0;memset(ch[0],0,sizeof ch[0]);}
452. int newnode(){
453. memset(ch[++tot],0,sizeof ch[tot]);
454. fail[tot]=cnt[tot]=0;
455. return tot;
456. }
457. void insert(char \*s){
458. int n = strlen(s+1);
459. int now = 0;
460. for (int i=1;i<=n; i++) {
461. if (!ch[now][s[i]-'a'])
462. ch[now][s[i]-'a']=newnode();
463. now=ch[now][s[i]-'a'];
464. }
465. cnt[now]++;
466. }
467. void build() {
468. int hd=0,tl=0;
469. q[++tl]=0;
470. while(hd<tl) {
471. int u=q[++hd];
472. for(int i=0;i<26;i++) if(ch[u][i]) {
473. q[++tl]=ch[u][i];
474. if(!u) continue;
475. int v=fail[u];
476. while(v) if(ch[v][i]) break;
477. else v=fail[v];
478. if(ch[v][i]) fail[ch[u][i]]=ch[v][i];
479. }
480. }
481. }
482. int Get(int u) {
483. int res = 0;
484. while(u&&cnt[u]!=-1){
485. res += cnt[u];
486. cnt[u] = -1;
487. u = fail[u];
488. }
489. return res;
490. }
491. int match(char \*s) {
492. int n = strlen(s+1);
493. int res = 0,now = 0;
494. for(int i=1; i<=n; i++) {
495. char c=s[i];
496. if (ch[now][c-'a']) now=ch[now][c-'a'];
497. else {
498. while(now&&!ch[now][c-'a']) now=fail[now];
499. if(ch[now][c-'a']) now=ch[now][c-'a'];
500. }
501. res=res+Get(now);
502. }
503. return res;
504. }
505. }aho;
506. char s[N];
507. int main(){
508. int t,n;
509. scanf("%d", &t);
510. while(t--) {
511. aho.init();
512. scanf("%d", &n);
513. for(int i = 0; i < n; i++) {
514. scanf("%s",s+1);
515. aho.insert(s);
516. }
517. aho.build();
518. scanf("%s",s+1);
519. printf("%d\n",aho.match(s));
520. }
521. return 0;
522. }
     * 1. 优化版（没有保留原有tire图）
523. #include<bits/stdc++.h>
524. using namespace std;
525. typedef long long ll;
526. const int N=1e6+2,M=5e5+2;
527. struct Aho {
528. int ch[M][26],fail[M],cnt[M],q[M];
529. int tot;
530. void init() {tot=0;memset(ch[0],0,sizeof ch[0]);}
531. int newnode(){
532. memset(ch[++tot],0,sizeof ch[tot]);
533. fail[tot]=cnt[tot]=0;
534. return tot;
535. }
536. void insert(char \*s){
537. int n = strlen(s+1);
538. int now = 0;
539. for (int i=1;i<=n; i++) {
540. if (!ch[now][s[i]-'a'])
541. ch[now][s[i]-'a']=newnode();
542. now=ch[now][s[i]-'a'];
543. }
544. cnt[now]++;
545. }
546. void build() {
547. int hd=0,tl=0;
548. for(int i=0;i<26;i++) if(ch[0][i]) q[++tl]=ch[0][i];
549. while(hd<tl) {
550. int u=q[++hd];
551. for(int i=0;i<26;i++) if(ch[u][i]) {
552. q[++tl]=ch[u][i];
553. fail[ch[u][i]]=ch[fail[u]][i];
554. }else ch[u][i]=ch[fail[u]][i];
555. }
556. }
557. int Get(int u) {
558. int res = 0;
559. while(u&&cnt[u]!=-1){
560. res += cnt[u];
561. cnt[u] = -1;
562. u = fail[u];
563. }
564. return res;
565. }
566. int match(char \*s) {
567. int n = strlen(s+1);
568. int res = 0,now = 0;
569. for(int i=1; i<=n; i++) {
570. char c=s[i];
571. if (ch[now][c-'a']) now=ch[now][c-'a'];
572. else {
573. if(ch[now][c-'a']) now=ch[now][c-'a'];
574. else now=0;
575. }
576. res=res+Get(now);
577. }
578. return res;
579. }
580. }aho;
581. char s[N];
582. int main(){
583. int t,n;
584. scanf("%d", &t);
585. while(t--) {
586. aho.init();
587. scanf("%d", &n);
588. for(int i = 0; i < n; i++) {
589. scanf("%s",s+1);
590. aho.insert(s);
591. }
592. aho.build();
593. scanf("%s",s+1);
594. printf("%d\n",aho.match(s));
595. }
596. return 0;
597. }
     1. 后缀自动机 //总结fail树上跳的倍增算法
598. struct SuffixAutoMaton{
599. int ch[N<<1][26],fa[N<<1],l[N<<1],size[N<<1],k[N<<1],c[N<<1];
600. int last,tot;
601. void init() {last=tot=1;memset(ch[1],0,sizeof ch[1]);}
602. void ins(int c) {
603. int p=last,np=++tot;last=np;l[np]=l[p]+1;
604. memset(ch[tot],0,sizeof ch[tot]);
605. for(;p&&!ch[p][c];p=fa[p])ch[p][c]=np;
606. if(!p)fa[np]=1;
607. else {
608. int q=ch[p][c];
609. if(l[p]+1==l[q])fa[np]=q;
610. else {
611. int nq=++tot;l[nq]=l[p]+1;
612. memcpy(ch[nq],ch[q],sizeof(ch[q]));
613. fa[nq]=fa[q];fa[q]=fa[np]=nq;
614. for(;ch[p][c]==q;p=fa[p])ch[p][c]=nq;
615. }
616. }
617. size[np]=1;
618. }
619. void build() {
620. init();
621. n=strlen(s+1);
622. for(int i=1;i<=n;i++) ins(s[i]-'a');
623. for(int i=1;i<=tot;i++) c[l[i]]++;
624. for(int i=1;i<=tot;i++) c[i]+=c[i-1];
625. for(int i=1;i<=tot;i++) k[c[l[i]]--]=i;
626. for(int i=1;i<=tot;i++) c[i]=size[i]=0;
627. }
628. }sam;
     1. Manacher
629. char s[N],s2[N\*2+5];
630. int p[N\*2+5],len;
631. int Manacher() {
632. int lens=strlen(s+1);
633. len=2;s2[0]='$';s2[1]='#';
634. for(int i=1;i<=lens;i++) {
635. s2[len++] = s[i];
636. s2[len++] = '#';
637. }s2[len--]='\0';
638. int id,mx=0;
639. for (int i=1; i<=len; i++) {
640. if (i<mx) p[i] = min(p[2\*id-i],mx-i);
641. else p[i] = 1;
642. while (s2[i-p[i]] == s2[i+p[i]]) p[i]++;
643. if (mx<i+p[i]) {
644. id=i;
645. mx=i+p[i];
646. }
647. }
648. int mxlen=0;
649. for (int i=1; i<=len; i++) mxlen=max(mxlen,--p[i]);
650. return mxlen;
651. }
652. /\*
653. ababa -> # a # b # a # b # a #
654. 0 1 0 3 0 5 0 3 0 1 0
655. aabaa -> # a # a # b # a # a #
656. 0 1 2 1 0 5 0 1 2 1 0
657. l=i/2-p[i]/2+(i&1);
658. r=i/2+p[i]/2;
659. if(l>r) continue;
660. \*/
661. 数学
     1. 大数开根
662. public static BigInteger isqrtNewton(BigInteger n) {
663. BigInteger a = BigInteger.ONE.shiftLeft(n.bitLength() / 2);
664. boolean p\_dec = false;
665. for (;;) {
666. BigInteger b = n.divide(a).add(a).shiftRight(1);
667. if (a.compareTo(b) == 0 || a.compareTo(b) < 0 && p\_dec)
668. break;
669. p\_dec = a.compareTo(b) > 0;
670. a = b;
671. }
672. return a;
673. }
674. 几何
     1. 线段树维护凸包并判断凸包与直线是否存在交点
675. #include <bits/stdc++.h>
676. #define lson l,m,rt<<1
677. #define rson m+1,r,rt<<1|1
678. using namespace std;
679. typedef long long int ll ;
680. const int N = 100000+7;
681. struct Point {
682. ll x,y;
683. Point(ll x=0, ll y=0):x(x),y(y) { }
684. inline void read() {cin>>x>>y;}
685. }a[N];
686. typedef vector<Point> Polygon;
687. typedef Point Vector;
688. inline Vector operator+ (Vector A, Vector B) { return Vector(A.x + B.x, A.y + B.y); }
689. inline Vector operator- (Point A, Point B) { return Vector(A.x - B.x, A.y - B.y); }
690. inline bool operator < (Point a, Point b) { return a.x < b.x || (a.x == b.x && a.y < b.y); }
691. inline bool operator == (Point a, Point b) { return (a.x - b.x) == 0 && (a.y - b.y) == 0; }
692. inline ll Cross(Vector A, Vector B) { return A.x \* B.y - A.y \* B.x; }
693. vector<Point>v[N<<2];
694. void ins(vector<Point> &v,Point p) {
695. int m = v.size();
696. while(m>1 && Cross(p-v[m-2],v[m-1]-v[m-2])<=0) m--;
697. v.resize(m);v.push\_back(p);
698. }
699. void build(int l,int r,int rt) {
700. v[rt].clear();
701. for(int i=l;i<=r+1;i++) ins(v[rt],a[i]);
702. if (l==r) return ;
703. int m=(l+r)>>1;
704. build(lson),build(rson);
705. }
706. bool check(Point a,Point b,vector<Point>&v){//判断凸包与直线是否存在交点
707. int l=0,r=v.size()-2;
708. while(l<r){
709. int mid=(l+r)>>1;
710. if(Cross(v[mid]-a,b-a)<Cross(v[mid+1]-a,b-a)) r=mid;// cross(x,y)里面的x与y的顺序和方向有关
711. else l=mid+1;
712. }
713. return Cross(v[l]-a,b-a)<0||Cross(v[l+1]-a,b-a)<0;
714. }
715. int qry(int L,int R,int l,int r,int rt,Point a,Point b) {
716. if (L<=l && r<=R) {
717. if(!check(a,b,v[rt])) return 0;
718. if(l==r) return l;
719. }
720. int m=(l+r)>>1,ret=0;
721. if (L<=m) ret=qry(L,R,lson,a,b);
722. if (!ret&&m<R) ret=qry(L,R,rson,a,b);
723. return ret;
724. }
725. int main(){
726. int t,n;
727. cin>>t;
728. while(t--) {
729. cin>>n;
730. for(int i=1;i<=n;i++) a[i].read();
731. build(1,n-1,1);
732. for(int i=1;i<=n-1;i++) {
733. printf("%d ",qry(i+1,n-1,1,n-1,1,a[i],a[i+1]));
734. }puts("");
735. }
736. return 0;
737. }