# HDU 4777 Rabbit Kingdom解题报告

## 题目

　　Long long ago, there was an ancient rabbit kingdom in the forest. Every rabbit in this kingdom was not cute but totally pugnacious, so the kingdom was in chaos in season and out of season.   
　　n rabbits were numbered form 1 to n. All rabbits' weight is an integer. For some unknown reason, two rabbits would fight each other if and only if their weight is NOT co-prime.   
　　Now the king had arranged the n rabbits in a line ordered by their numbers. The king planned to send some rabbits into prison. He wanted to know that, if he sent all rabbits between the i-th one and the j-th one(including the i-th one and the j-th one) into prison, how many rabbits in the prison would not fight with others.   
　　Please note that a rabbit would not fight with himself.

**Input**

　　The input consists of several test cases.   
　　The first line of each test case contains two integer n, m, indicating the number of rabbits and the queries.   
　　The following line contains n integers, and the i-th integer W i indicates the weight of the i-th rabbit.   
　　Then m lines follow. Each line represents a query. It contains two integers L and R, meaning the king wanted to ask about the situation that if he sent all rabbits from the L-th one to the R-th one into prison.   
　　(1 <= n, m, W i <= 200000, 1 <= L <= R <= n)   
　　The input ends with n = 0 and m = 0.

**Output**

　　For every query, output one line indicating the answer.

**Sample Input**

3 2

2 1 4

1 2

1 3

6 4

3 6 1 2 5 3

1 3

4 6

4 4

2 6

0 0

**Sample Output**

2

1

1

3

1

2

**Hint**

　　In the second case, the answer of the 4-th query is 2, because only 1 and 5 is co-prime with other numbers in the interval [2,6] .

## 题意梗概

（有多组数据，以0 0结束）每组数据中有n个元素及m个区间询问，求区间内与其他任何数都互质的数的个数。

## 思路

~~原谅我太菜了，这题讲道理真的好难啊= =~~

先预处理出各个数左右互质的边界（ l[i] , r[i] ），再将询问区间按左边界不下降的顺序排序。建立树状数组t,使[ i , j ]的询问转化为树状数组[ i , j ]的区间和，并对树状数组进行维护。（\*如果看不懂可以跳过这段话看后面）

### 1.预处理

#### a.线性筛求素数

略......

#### b.处理因子

设我们要求x的因子，先将x的值赋值给tmp，然后对tmp不断用不大于sqrt( tmp )的质数对tmp进行试除，一但整除则除tmp至不再含此因子，若最后tmp不为1则将tmp加至因子数组中。

###### 参考代码

int cal\_Factor(int x)

{

int tmp=x; int tot=0; // tot为统计的因子数量

for (int j=1;(p[j]\*p[j])<=tmp;j++)

{

if (tmp%p[j]) continue;

while (tmp%p[j]==0) tmp/=p[j];

fac[++tot]=p[j]; // fac[] 记录x的各个因子

}

if (tmp!=1) fac[++tot]=tmp;

return tot; //返回因子数量

}

#### c.求得互质数的左右边界

设左边界 l[x]为小于x的最大与它不互质的数的位置(无则赋值为0)，

设右边界r[x]为大于x的最小与它不互质的数的位置(无则赋值为N+1)，

（即在区间（ l[x] , r[x] ）内所有数均与x互质）

求左边界l[i] 时，i应从1到n扫一遍，用adj[j] 数组记录上一个含有j因子的位置，并初始化 l[i] = 0 , adj[j] = 0 , 将a[i] 的所有因子扫一遍，并依次取adj数组对应因子位置的较大值

右边界r[i] 同理。

###### 参考代码

void pre()

{

int adj[N];

for (int i=1;i<N;i++) adj[i]=0; //注意是N ( a[i]的最大取值) 还是n ( a[i]数量)

for (int i=1;i<=n;i++)

{

l[i]=0;

int cnt=cal\_Factor(a[i]);

for (int j=1;j<=cnt;j++)

{

l[i]=max(l[i],adj[fac[j]]); //对应不同因子对应的不同位置取最大值

adj[fac[j]]=i; //每次都要把adj[]更新为i

}

}

for (int i=1;i<N;i++) adj[i]=N+1; //求右区间重新初始化

for (int i=n;i>0;i--) //遍历方向相反

{

r[i]=n+1;

int cnt=cal\_Factor(a[i]);

for (int j=1;j<=cnt;j++)

{

r[i]=min(r[i],adj[fac[j]]);

adj[fac[j]]=i;

}

}

}

### 2.询问的排序（在线转离线）

按左边界不下降排列，并记录原询问顺序。

### 3.树状数组建立、维护

#### a.树状数组的建立

设询问区间[ L , R ] 等价于对树状数组 [ i , j ]区间求和。此时需要区间修改和单点查询，考虑使用差分。

首先先考虑 L = 1 的情况，此时根据l[i]的值分为两种情况:

&1: l[i]==0 ，即位置i上的数与在r [ i ]前面数均互质，则对询问右边界在[ i , r[i] - 1 ]的区间内均能做了一点微薄的贡献，则在i处 +1，r[i]处 -1。

&2:l[i]>0，则将在l[i]处的vector数组末端加入i（后面有用）

#### b.树状数组的维护

设上一步的左边界为L，则当前维护左边界为L+1的树状数组（ 类似数学归纳法 ），主要有两步变化：

&1:对于L位置上的a[L]，此时已经在询问区间左侧，无法对区间的答案产生影响了，故应该在L处 -1 ，在 r[i]处 +1 （反向改值）

&2:对于原本l[i]=L的数，此时进入询问区间而对应被算上，类似于建立，若l[i]=L，则在i处 +1 ，r[i]处 -1

\*解释一下，数字i能被计入[L,R]的询问的充要条件是i∈[L,R]， 且 [ l[i] , r[i] ] ⊇ [L,R]

#### c.树状数组的求和

略...

###### (树状数组部分)参考代码

//树状数组的建立

for (int i=1;i<=n;i++)

if (!l[i])

{

Edit(i,1); Edit(r[i],-1);

}

else vec[l[i]].push\_back(i); //在l[i]处加入i

//树状数组的维护

int x=0,y=1; //x代表询问左区间L , y代表回答到第几条询问

while (x<n)

{

x++;

if (x>1)Edit(x-1,-1),Edit(r[x-1],1); //1为左边界时无需复原

while (q[y].l==x) //询问的左边界还是x时

{

ans[q[y].id]=Query(q[y].r)-Query(q[y].l-1);

y++;

}

vector <int>::iterator it;

for (it=vec[x].begin();it!=vec[x].end();it++) //提取l[i]处的i

{

Edit(\*it,1);

Edit(r[\*it],-1);

}

}

## 完整代码（比较冗长不建议看）

#include<bits/stdc++.h>

#define N 200077

using namespace std;

int p[N],ans[N],l[N],r[N],t[N],n,m,a[N],fac[N];

vector <int>vec[N];

struct node{

int l,r,id;

}q[N];

inline bool cmp(const node &x,const node &y) //询问排序

{

if (x.l==y.l) return x.id<y.id;

return x.l<y.l;

}

void init() //初始化

{

memset(t,0,sizeof(t)); memset(ans,0,sizeof(ans));

for (int i=1;i<=n;i++) vec[i].clear();

}

void cal\_Prime() //统计质数

{

bool is\_p[N]; int cnt=0;

memset(is\_p,1,sizeof(is\_p));

for (int i=2;i<N;i++)

{

if (is\_p[i]) p[++cnt]=i;

for (int j=1;j<=cnt&&(i\*p[j]<N);j++)

{

is\_p[i\*p[j]]=0;

if (i%p[j]==0) break;

}

}

}

int cal\_Factor(int x)

{

int tmp=x; int tot=0;

for (int j=1;(p[j]\*p[j])<=tmp;j++)

{

if (tmp%p[j]) continue;

while (tmp%p[j]==0) tmp/=p[j];

fac[++tot]=p[j];

}

if (tmp!=1) fac[++tot]=tmp;

return tot;

}

void pre()

{

int adj[N];

for (int i=1;i<N;i++) adj[i]=0;

for (int i=1;i<=n;i++)

{

l[i]=0;

int cnt=cal\_Factor(a[i]);

for (int j=1;j<=cnt;j++)

{

l[i]=max(l[i],adj[fac[j]]);

adj[fac[j]]=i;

}

}

for (int i=1;i<N;i++) adj[i]=N+1;

for (int i=n;i>0;i--)

{

r[i]=n+1;

int cnt=cal\_Factor(a[i]);

for (int j=1;j<=cnt;j++)

{

r[i]=min(r[i],adj[fac[j]]);

adj[fac[j]]=i;

}

}

}

inline int Lowbit(int x)

{

return x&(-x);

}

void Edit(int x,int val)

{

for (int i=x;i<=n;i+=Lowbit(i))

t[i]+=val;

}

int Query(int x)

{

int ans=0;

for (int i=x;i>0;i-=Lowbit(i))

ans+=t[i];

return ans;

}

int main()

{

cal\_Prime();

while (1)

{

scanf("%d%d",&n,&m);

if (n==0&&m==0) return 0; //输入为0 0时结束程序

for (int i=1;i<=n;i++)

scanf("%d",&a[i]);

for (int i=1;i<=m;i++)

scanf("%d%d",&q[i].l,&q[i].r),q[i].id=i;

init();

pre();

sort(q+1,q+1+m,cmp);

for (int i=1;i<=n;i++)

if (!l[i])

{

Edit(i,1); Edit(r[i],-1);

}

else vec[l[i]].push\_back(i);

int x=0,y=1;

while (x<n)

{

x++;

if (x>1)Edit(x-1,-1),Edit(r[x-1],1);

while (q[y].l==x)

{

ans[q[y].id]=Query(q[y].r)-Query(q[y].l-1);

y++;

}

vector <int>::iterator it;

for (it=vec[x].begin();it!=vec[x].end();it++)

{

Edit(\*it,1);

Edit(r[\*it],-1);

}

}

for (int i=1;i<=m;i++)

{

printf("%d\n",ans[i]);

}

}

}