题意：

求区间内所有子串的异或值等于k的有多少个。

分析：

**询问区间中子区间异或值为k的个数。我们如果提前处理出异或前缀那么区间[L,R]的异或的值就为sum[R]^sum[L-1] = k,转化一下，我们就可以知道sum[L-1] = sum[R]^k,sum[R] = sum[L-1]^k,那么区间想右移动一位增加的区间为k的数量就是在[L,R+1]中num[sum[R]^k].那么我们就可以通过莫队算法解决这个问题。**

Code：

#include<iostream>

#include<cstdio>

#include<cstring>

#include<string>

#include<algorithm>

#include<cmath>

#include<queue>

#include<map>

#include<vector>

#include<set>

#include<bitset>

#include<stack>

#include<cctype>

#define sc(x) scanf("%d",&x);

#define PR(x) cout<<#x<<": "<<x<<endl;

using namespace std;

typedef long long ll;

typedef pair<int,int> p;

const double eps = 1e-4;

const int mod = 1e9+7;

const ll INF = 0x3f3f3f3f3f3f3f3f;

const int inf = 0x3f3f3f3f;

const int maxn = 2e6+7;

const int maxm = 2e7+7;

const double pi = acos(-1.0);

int n,m,k,unit;

struct Query{

int l,r,id;

}node[maxn];

bool cmp(Query a,Query b){

if(a.l/unit!=b.l/unit)

return a.l/unit<b.l/unit;

else

return a.r<b.r;

}

ll ans[maxn];

int a[maxn],sum[maxn];

int num[maxm];

void solve(){

ll temp=0;

memset(num,0,sizeof num);

int l=1,r=0;

for(int i=0; i<m; i++){

while(r<node[i].r){

r++;

temp+=num[sum[r]^k];

num[sum[r]]++;

}

while(r>node[i].r){

num[sum[r]]--;

temp-=num[sum[r]^k];

r--;

}

while(l<node[i].l){

num[sum[l]]--;

temp-=num[sum[l]^k];

l++;

}

while(l>node[i].l){

l--;

temp+=num[sum[l]^k];

num[sum[l]]++;

}

// printf("Debug: %d %d %d\n",l,r,temp);

ans[node[i].id]=temp;

}

}

int main(){

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

sum[0]=0;

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

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

sum[i]=sum[i-1]^a[i];

}

for(int i=0; i<m; i++){

scanf("%d%d",&node[i].l,&node[i].r);

node[i].l--;

node[i].id=i;

}

unit=(int)sqrt(n);

sort(node,node+m,cmp);

solve();

for(int i=0; i<m; i++)

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

return 0;

}