### 0.0.1 莫队算法

处理的东西主要是在于区间，这里的区间不是单纯区间，而是你可以向上向下走的问题，并且有O(1)的递推式就可以 走，这里的关键在于玄学把复杂度降到$N^{\frac{3}{2}}$然后需要处理的就是你走所带来的一个权值变化，这个是关键的问题

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1<<20;
//莫对经典算法例题
//莫队的意义首先你要可以做到可以离线，其次是你要知道它向上向下走价值的变化
//其他的就是板子的问题了，因为这里的关键就变成了复杂度变成了求曼哈顿距离，如何把这些点变
struct node
{
  int l,r,id;
}Q[N];
ll ans[N];  //这里的ans存的是最后的答案
int a[N],pos[N];//这里的a存的是原始数组，pos存的是分块
int n,m,k;
int L = 1,R = 0;
ll num = 0;
bool cmp(node a,node b)  //这个是块之间与块内排序
{
  if(pos[a.l] == pos[b.l])
  {
    return a.r<b.r;
  }
  return pos[a.l]<pos[b.l];
}
void add(int x) //这里是关键
{

}
void del(int x)
{

}
int main()
{
  cin>>n>>m>>k;
  int sz = sqrt(n);
  for(int i = 1;i<=n;i++)
  {
    cin>>a[i];
```

```
    a[i]^=a[i-1];
    pos[i] = i/sz;
  }
  for(int i = 1;i<=m;i++)
  {
    cin>>Q[i].l>>Q[i].r;
    Q[i].id = i;
  }
  sort(Q+1,Q+m+1,cmp);
  flag[0] = 1;
  for(int i = 1;i<=m;i++)
  {
    while(L<Q[i].l)
    {
      del(L-1);
      L++;
    }
    while(L>Q[i].l)
    {
      L--;
      add(L-1);
    }
    while(R<Q[i].r)
    {
      R++;
      add(R);
    }
    while(R>Q[i].r)
    {
      del(R);
      R--;
    }
    ans[Q[i].id]  = num;
  }
  return 0;
}
```