莫队

莫队

简介

严格来说，莫队只是一种对暴力做法的优化，主要用于维护区间答案或者维护区间上的数据结构。可以被拓展为带修改莫队，树上莫队，树上带修改莫队等。

那它能干啥

莫队主要用于给定一段连续的序列，让你求出序列中每个数出现的次数或者处理和数出现次数有关的问题，事实上，一些题目可以用权值树状数组或主席树做~~（那要它有个啥用）~~。**但是莫队对于序列中数出现次数的维护较为灵活，维护个数的方便程度明显要高于权值数据结构**。

复杂度

较为玄学，一般来说，看分块的大小，一般来说为O(n√n)

具体操作

操作挺简单的，~~方便ctrl+c~~

现在考虑一个问题，给一个n个数的序列。m次询问，每次询问你区间 [l,r]有多少种不同的数。数据范围。n≤ 100000,m≤ 100000

一般来说，咱喜欢暴力，对于每个区间，我们选择建一个数组cnt[i]。从头到尾扫一边序列，遇到一个数j。就让cnt[j]++,最后在在数据范围内跑一边数组cnt。看有多少个cnt不是0。~~这样没问题啊~~，可是这样做复杂度可怕的一批，分分钟爆炸。考虑几个优化。

优化1：对于每个区间从区间头开始扫，与到没遇见的ans++。标记一下，然后继续往下扫到查询区间结束。这样我们成功把复杂度降低到了O(nm)

优化2：还是开一个cnt数组，用两个指针i。j在数列上移动，来缩减目标范围，遇到一个数j。如果cnt[j]为0，则ans++。当要缩减区间范围时若cnt[j]正好为1，则ans-。由于每次接着上次操作的结果，只要对cnt数组进行操作。复杂度的到一定的优化，然而，对于查询区间为[1,n][1,n-1][1,n-2].....这样和重新扫没区别，复杂度依旧还是O(nm)。一样爆炸……

简要放个代码描述一下处理过程

```
#define add(x) ans+=!cnt[x]++
#define del(x) ans-=!--cnt[x]
for(int i=1;i<=m;i++)
{
  while(r<tt[i].r) add(a[++r]);
```

```
  while(r>tt[i].r) del(a[r--]);
  while(l<tt[i].l) del(a[l++]);
  while(l>tt[i].l) add(a[--l]);
  end[tt[i].num]=ans;
     }
```

所以就有了莫队的分块的做法。我们考虑将序列分成√n取整块，然后将查询区间按照起始点 l 所属块的大小排序，然后对于起点所属块相同的点，就按照终点从小到大排序，然后存储询问的区间离线做。这样对于每个块最多移动√n次，在每一个块中重点最多移动n次，这样最终复杂度变成成功达到了根号级别。

这里大致放个代码

1 分块过程

```
for(int i=1;i<=x;i++)
    {
        for(int j=(i-1)*num+1;j<=min(n,i*num);j++)
        {
            bel[j]=i;
        }
    }
```

2 排序

```
bool cmp(node a,node b)
{
    return
bel[a.l]==bel[b.l]?(bel[a.l]&1?a.r<b.r:a.r>b.r):bel[a.l]<bel[b.l];
 }
```

这里我没有按照我在上面所说的，先按询问区间左端点排序，再按右端点从大到小排序，而是选择判断询问区间左端点所属块的奇偶性，来判断是从大到小，还是从小到大。这样其实是一个优化，由于我们分块就是要保证跳区间的连序性，而奇偶分能更好的保存其跳区间的连续性，就比如有一系列询问[1,2],[1,3],[1,4].....[1,n],[2,1],[2,2].....[2,n-1],我们假设块的大小为为1，~~当然这几乎不可能~~，若按照传统的莫队的排序，我们再从块1跳到块2时需要从n跳到1，增加了许多此必要的运算，如果按奇偶排序，则可以直接从n跳到n-1这样可以省许多次计算。（~~但其实也不一定有用啦啦啦~~

初级莫队就这些内容 最后放一个模板小b的询问，这是一道板子题啦。

```
#include <bits/stdc++.h>
using namespace std;
const int maxn=5e4+13;
struct node{
    int l,r;
    int num;
}tt[maxn];
int n,m,k,ans=0;
int bel[maxn];
int a[maxn];
int cnt[maxn];//村每个数出现的次数
int end[maxn];//存答案
```

```
bool cmp(node a,node b)
{
    return
bel[a.l]==bel[b.l]?(bel[a.l]&1?a.r<b.r:a.r>b.r):bel[a.l]<bel[b.l];
 }
#define add(x) ans+=!cnt[x]++
#define del(x) ans-=!--cnt[x] //提速法
int main()
{
    int l,r;
    scanf("%d%d%d",&n,&m,&k);
    int num=(int)sqrt((double)(n));
    int x=ceil((double)(n)/num);
    for(int i=1;i<=x;i++)
    {
        for(int j=(i-1)*num+1;j<=min(n,i*num);j++)
        {
            bel[j]=i;
        }
    }
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    for(int i=1;i<=m;i++)
    {
        scanf("%d%d",&l,&r);
        tt[i]=(node){l,r,i};
     }
     sort(tt+1,tt+1+m,cmp);//分块
    l=1,r=0;
    for(int i=1;i<=m;i++)//莫队四件套
    {
        while(r<tt[i].r) add(a[++r]);
        while(r>tt[i].r) del(a[r--]);
        while(l>tt[i].l) add(a[--l]);
                while(l<tt[i].l) del(a[l++]);
        end[tt[i].num]=ans;
     }
    for(int i=1;i<=m;i++) printf("%d\n",end[i]);
 }
```

From:
https://wiki.cvbbacm.com/ - **CVBB ACM Team**

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:manespace:%E8%8E%AB%E9%98%9F&rev=1589028620

Last update: **2020/05/09 20:50**