

莫队算法良心讲解

问题：有n个数组成一个序列，有m个形如询问L, R的询问，每次询问需要回答区间内至少出现2次的数有哪些。

朴素的解法需要读取 $O(nm)$ 次数。如果数据范围小，可以用数组，时间复杂度为 $O(nm)$ 。如果使用STL的Map来保存出现的次数，则需要 $O(nm\log n)$ 的复杂度。有没有更快的方法呢？

注意到询问并没有强制在线，因此我们可以使用离线方法。注意到一点，如果有计算完 $[L, R]$ 时的“中间变量”（在本题为每个数出现的次数），那么 $[L - 1, R]$ 、 $[L + 1, R]$ 、 $[L, R - 1]$ 、 $[L, R + 1]$ 都能够在“中间变量”的“基本操作时间复杂度”⁽¹⁾得出。如果能安排适当的询问顺序，使得每次询问都能用上上次运行产生的中间变量，那么我们将可以在更优的复杂度完成整个询问。

(1) 如果数据较小，用数组，时间复杂度为 $O(1)$ ；如果数据较大，可以考虑用离散化或map，时间复杂度为 $O(\log n)$ 。

那如何安排询问呢？这里有个时间复杂度非常优秀的方法：首先将每个询问视为一个“点”，两个点 P_1, P_2 之间的距离为 $\text{abs}(L_1 - L_2) + \text{abs}(R_1 - R_2)$ ，即曼哈顿距离，然后求这些点的最小生成树，然后沿着树边遍历一次。由于这里的距离是曼哈顿距离，所以这样的生成树被称为“曼哈顿最小生成树”。最小曼哈顿生成树有专用的算法⁽²⁾，求生成树时间复杂度可以仅为 $O(m\log m)$ 。

(2) 其实这里是建边的算法，建边后依然使用传统的Prim或者Kruskal算法来求最小生成树。

不幸的是，曼哈顿最小生成树的写法很复杂，考场上不建议这样做。

一种直观的办法是按照左端点排序，再按照右端点排序。但是这样的表现不好。特别是面对精心设计的数据，这样方法表现得很差。

举个例子，有6个询问如下：(1, 100), (2, 2), (3, 99), (4, 4), (5, 102), (6, 7)。

这个数据已经按照左端点排序了。用上述方法处理时，左端点会移动6次，右端点会移动移动 $98 + 97 + 95 + 98 + 95 = 483$ 次。右端点大幅度地来回移动，严重影响了时间复杂度——排序的复杂度是 $O(m\log m)$ ，所有左端点移动次数仅为 $O(n)$ ，但右端点每个询问移动 $O(n)$ ，共有m个询问，故总移动次数为 $O(nm)$ ，移动总数为 $O(m\log m + nm)$ 。运行时间上界并没有减少。

其实我们稍微改变一下询问处理的顺序就能做得更好：(2, 2), (4, 4), (6, 7), (5, 102), (3, 99), (1, 100)。

左端点移动次数为 $2 + 2 + 1 + 2 + 2 = 9$ 次，比原来稍多。右端点移动次数为 $2 + 3 + 95 + 3 + 1 = 104$ ，右端点的移动次数大大降低了。

上面的过程启发我们：①我们不应该严格按照升序排序，而是根据需要灵活一点的排序方法；②如果适当减少右端点移动次数，即使稍微增多一点左端点移动次数，在总的复杂度上看，也是划算的。

在排序时，我们并不是按照左右端点严格升序排序询问，而只是令其左右端点处于“大概是升序”的状态。具体的方法是，把所有的区间划分为不同的块，将每个询问按照左端点的所在块序号排序，左端点块一样则按照右端点排序。注意这个与上一个版本的不同之处在于“第一关键字”是左端点所在块而非左端点。

这就是莫队算法。为什么叫莫队算法呢？据说这是2010年国家集训队的莫涛⁽³⁾在作业里提到了这个方法。

(3) 由于莫涛经常打比赛做队长，大家都叫他莫队，该算法也被称为莫队算法。（感谢汝佳大神、莫队的指出）

莫队算法首先将整个序列分成 \sqrt{n} 个块（同样，只是概念上分的块，实际上我们并不需要严格存储块），接着将每个询问按照块序号排序（一样则按照右端点排序）。之后，我们从排序后第一个询问开始，逐个计算答案。



```
1 int len;    // 块长度
2
3 struct Query{
4     int L, R, ID, block;
5     Query(){} // 构造函数重载
6     Query(int l, int r, int ID):L(l), R(r), ID(ID){
```

公告

昵称： 张瑯小强
园龄： 3年
粉丝： 6
关注： 5
[+加关注](#)

<					2019年9				
日					一				
1					2				
8					9				
15					16				
22					23				
29					30				
6					7				

搜索

我的标签

C++(3)

OI(1)

编译选项(1)

模拟退火(1)

莫队算法(1)

算法(1)

ACM(1)

C(1)

```
7     block = 1 / len;
8 }
9 bool operator < (const Query rhs) const {
10     if(block == rhs.block) return R < rhs.R;    // 不是if(L == rhs.L) return R < rhs.R; return L < rhs.L
11     return block < rhs.block;                  // 否则这就变回算法一了
12 }
13 }queries[maxm];
14
15 map<int, int> buf;
16
17 inline void insert(int n){
18     if(buf.count(n))
19         ++buf[n];
20     else
21         buf[n] = 1;
22 }
23 inline void erase(int n){
24     if(--buf[n] == 0) buf.erase(n);
25 }
26
27 int A[maxn];          // 原序列
28 queue<int> anss[maxm]; // 存储答案
29
30 int main(){
31     int n, m;
32     cin >> n;
33     len = (int)sqrt(n);    // 块长度
34     for(int i = 1; i <= n; i++){
35         cin >> A[i];
36     }
37     cin >> m;
38     for(int i = 1; i <= m; i++){
39         int l, r;
40         cin >> l >> r;
41         queries[i] = Query(l, r, i);
42     }
43     sort(queries + 1, queries + m + 1);
44     int L = 1, R = 1;
45     buf[A[1]] = 1;
46     for(int i = 1; i <= m; i++){
47         queue<int>& ans = anss[queries[i].ID];
48         Query &q = queries[i];
49         while(R < q.R) insert(A[++R]);
50         while(L > q.L) insert(A[--L]);
51         while(R > q.R) erase(A[R--]);
52         while(L < q.L) erase(A[L++]);
53
54         for(map<int, int>::iterator it = buf.begin(); it != buf.end(); ++it){
55             if(it->second >= 2){
56                 ans.push(it->first);
57             }
58         }
59     }
60     for(int i = 1; i <= m; i++){
61         queue<int>& ans = anss[i];
62         while(!ans.empty()){
63             cout << ans.front() << ' ';
64             ans.pop();
65         }
66         cout << endl;
67     }
68 }
```



随笔档案

2017年2月(1)

2016年11月(3)

2016年10月(4)

2016年9月(2)

2016年8月(2)

最新评论

1. Re:莫队算法良心讲解

你跟刘汝佳大神认识？

2. Re:莫队算法良心讲解

刘汝佳指导写的？？？

3. Re:莫队算法良心讲解

@
sorry楼主，我白痴了，
大>sqrt(n)，莫队算法
复杂度没有m是因为n和
级，所以统一写成了n？

阅读排行榜

1. 莫队算法良心讲解(7

2. 手把手教你模拟退

3. 我的G++编译选项(

4. 乘法取模(1050)

5. SQL注入方法之：获

评论排行榜

1. 莫队算法良心讲解(3

推荐排行榜

1. 莫队算法良心讲解(6

尽管分了块，但是我们可以对所有的“询问转移”一视同仁。上述的代码有几个需要注意的地方。

一是insert和erase，这里在插入前判断了是否存在、插入后判断是否为0，但这不是必须的（insert时会将新节点初始化为0，erase为0后对处理答案不影响）；

二是区间变化的顺序，insert最好放在前面，erase最好在后面（想一想，为什么）；

三是insert总是使用前缀自增自减运算符，erase总是用后缀运算符；

2. 手把手教你模拟退
3. 编译器优化误解程序
4. SPFA最短路算法(1)
5. 我的G++编译选项(

四是在我们访问我们在“询问转移”前声明了Query的引用，来减少运行时寻址的计算量；

五是我们重载了Query的构造函数。为什么要重载呢？

我们希望在Query得到L, R, ID时自动计算块block，这就要写一个构造函数Query(int L, int R, int ID)来实现。但是,当结构体没有构造函数，实例化时不会初始化，有构造函数则一定会调用构造函数进行初始化。“托他的福”，queries数组建立时会对每个元素调用一次构造函数。可是我们只有有3个参数的构造函数，构造时一定要要有3个参数。而建立数组时却没有参数，编译器会报错。折中的办法是写一个没有参数的构造函数，可以避免这一问题。

这样排序有个特点。L和R都是“大概是升序”。不过L大概像爬山，总体上升但是会有局部的小幅度下降。R则有些难以形容，大概可以看出其由很多段快速上升，每段上升到顶端后下降到底。

下面是随机生成100个数据，将数据放到WPS表格后制成图表后的样子。



还有一个问题，为什么分块要分成 \sqrt{n} 块呢？我们分析一下时间复杂度。

假设我们每k个点分一块。

如果当前询问与上一询问左端点处在同一块，那么左端点移动为 $O(k)$ 。虽然右端点移动可能高达 $O(n)$ ，但是整一块询问的右端点移动距离之和也是 $O(n)$ （想一想，为什么）。因此平摊意义下，整块移动为 $O(k) \times O(k) + O(n)$ ，一共有 n / k 块，时间复杂度为 $O(kn + n^2 / k)$ 。

如果询问与上一询问左端点不处于同一块，那么左端点移动为 $O(k)$ ，但右端点移动则高达 $O(n)$ 。幸运的是，这种情况只有 $O(n / k)$ 个，时间复杂度为 $O(n + n^2 / k)$ 。

总的移动次数为 $O(kn + n^2 / k)$ 。因此，当 $k = \sqrt{n}$ 时，运行时间上界最优，为 $O(n^{1.5})$ 。

最后，因此根据每次insert和erase的时间复杂度，乘上 $O(1)$ 或者 $O(\log n)$ 亦或 $O(n)$ 不等，得到完整算法的时间复杂度（代码使用了map，为 $O(\log n)$ ）。

十分感谢汝佳大神对此文的指导orz。

标签：莫队算法， C++

好文要顶

关注我

收藏该文

张瑯小强

关注 - 56

粉丝 - 60

+加关注

« 上一篇：高精度模板

» 下一篇：乘法取模

posted @ 2016-09-24 23:03 张瑯小强 阅读(7964) 评论(3) 编辑 收藏

评论列表

#1楼 2018-08-03 10:52 Jinke2017

@

sorry楼主，我白痴了，应该是m要比较大>sqrt(n)，莫队算法才有优势，对吧？
复杂度没有m是因为n和m是同一个数量级，所以统一写成了n？

支持(0) 反对(0)

#2楼 2018-08-06 17:51 _明年今日

刘汝佳指导写的? ? ?

支持(1) 反对(0)

#3楼 2019-08-06 20:05 danzh

你跟刘汝佳大神认识?

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#) ， [访问](#) [网站首页](#)。

- 【推荐】超50万C++/C#源码: 大型实时仿真组态图形源码
- 【活动】阿里云910会员节多款云产品满减活动火热进行中
- 【推荐】新手上天翼云，数十款云产品、新一代主机0元体验
- 【推荐】零基础轻松玩转华为云产品，获赠礼加返百元大礼
- 【推荐】华为IoT平台开发者套餐9.9元起，购买即送免费课程

相关博文：

- [【学习笔记】莫队算法](#)
- [莫队算法~讲解【更新】](#)
- [莫队算法~讲解](#)
- [XOR and Favorite Number \(莫队算法\)](#)
- [bzoj 2038 小z的袜子 莫队例题](#)

最新 IT 新闻：

- 华为已获得50多份5G商用合同，5G基站发货超20万个
- 获800亿日元投资后，JDI将建OLED工厂，但两年后才能量产
- 搭载高通骁龙855移动平台的三星Galaxy A90 5G现已正式发布
- 历经30多年的努力，科学家终于得到了另一种高温超导材料
- 偿还30亿美元债务 退任CEO 贾跃亭宣布FF重大消息
- » [更多新闻...](#)