

struct Edge

博客园 首页 新随笔 联系 订阅 管理

随笔 - 74 文章 - 0 评论 - 11

20161003 NOIP 模拟赛 T2 解题报告

Weed

duyege的电脑上面已经长草了，经过辨认上面有金坷垃的痕迹。

为了查出真相，duyege 准备修好电脑之后再进行一次金坷垃的模拟实验。

电脑上面有若干层金坷垃，每次只能在上面撒上一层高度为 v_i 的金坷垃，或者除掉最 新 v_i 层（不是量）撒的金坷垃。如果上面只留有不足 v_i 层金坷垃，那么就相当于电脑上 面没有金坷垃了。

duyege 非常严谨，一开始先给你 m 个上述操作要你依次完成。然后又对实验步骤进行了 q 次更改，每次更改都会改变其中一个操作为另外一个操作。每次修改之后都会询问最 终金坷垃的量有多少。

输入第一行为两个正整数 m 、 q ，接下来 m 行每行 2 个整数 k 、 v_i 。 k 为 0 时撒金坷垃，为 1 时除金坷垃。接下来 q 行每行 3 个整数 c_i 、 k 、 v_i ， c_i 代表被更改的操作是第 c_i 个，后面 2 个数描述更改为这样的操作。输出 q 行代表每次金坷垃的量为多少

对于 30%的数据， $m \leq 1000, q \leq 1000$ 。

对于另外 20%的数据，每次 $k=1$ 时都会将金坷垃清空。

对于 100%的数据， $m \leq 2 \times 10^5, q \leq 2 \times 10^5, v_i \leq 10^4$ 。

分割线

分析：

（30分）这道题朴素想法是每次对某个操作修改，然后依次算到最后一个元素。

（100分）朴素的想法在逐个计算上耗时太多，这时我们需要一种数据结构，能够对某个操作修改，又能够对某一段查询，那么不难想到线段树。

我们需要维护三个信息，即当前区间中操作的金坷垃数量，层数，以及需要删去更左边操作的层数。

显然左边的删除操作不会影响之后的操作，那么我们只需要对每个节点进行结算，统计即可。

下面代码的Push_up函数比较难懂，特此注释。

```
1 #include <bits/stdc++.h>
2 #define Never return
3 #define Explode 0
4
5 using namespace std;
6 struct SegTree { int l, r, Add, Cnt, Del; };
7 const int maxN = 2e5 + 100;
8
9 SegTree tr[ maxN << 2 ];
10 int arr[ maxN ];
11 bool op[ maxN ];
12
13 int INPUT ( ) {
14     int x = 0, f = 1; char ch = getchar ( );
15     while ( ch < '0' || ch > '9' ) { if ( ch == '-' ) f = -1; ch = getchar ( ); }
16     while ( ch >= '0' && ch <= '9' ) { x = ( x << 1 ) + ( x << 3 ) + ch - '0'; ch = getchar ( ); }
17     return x * f;
18 }
19
20 int Query_Tree ( const int i, const int Target ) {
```

公告

36,554 pageviews

昵称：SHHHS
园龄：1年8个月
粉丝：32
关注：34
+加关注

2018年4月						
<	日	一	二	三	四	五
	25	26	27	28	29	30
	1	2	3	4	5	6
	8	9	10	11	12	13
	15	16	17	18	19	20
	22	23	24	25	26	27
	29	30	1	2	3	4

搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类

DP(2)
Hash(1)
LCA
各类法师(13)
乱搞(3)
数据结构(3)

随笔档案

2016年10月 (51)
2016年9月 (23)

相册

哈哈(1)

Dalao们

Alpar
WA
阿飞
海蜇
卢表叔

```
21     if ( Target == tr[ i << 1 | 1 ].Add ) {
22         return tr[ i ].Cnt - tr[ i << 1 | 1 ].Cnt ;
23     }
24     else if ( Target < tr[ i << 1 | 1 ].Add ) {
25         return tr[ i ].Cnt - tr[ i << 1 | 1 ].Cnt + Query_Tree ( i << 1
| 1 , Target ) ;
26     }
27     else {
28         return Query_Tree ( i << 1 , Target - tr[ i << 1 | 1 ].Add + tr[
i << 1 | 1 ].Del ) ;
29     }
30 }
31
32 void Push_up ( const int i ) {
33     int lchild = i << 1 ;
34     int rchild = lchild + 1 ;
35     if ( tr[ rchild ].Del >= tr[ lchild ].Add ) { //右区间 删除数大于等于左边层数
36         tr[ i ].Del = tr[ lchild ].Del + tr[ rchild ].Del - tr[ lchild
].Add ;
37         tr[ i ].Add = tr[ rchild ].Add ;
38         tr[ i ].Cnt = tr[ rchild ].Cnt ;
39     }
40     else if ( !tr[ rchild ].Del ) { //右区间没有删除
41         tr[ i ].Add = tr[ lchild ].Add + tr[ rchild ].Add ;
42         tr[ i ].Cnt = tr[ lchild ].Cnt + tr[ rchild ].Cnt ;
43         tr[ i ].Del = tr[ lchild ].Del ;
44     }
45     else { //右边无法全部删去左边
46         tr[ i ].Del = tr[ lchild ].Del ;
47         tr[ i ].Add = tr[ lchild ].Add + tr[ rchild ].Add - tr[ rchild
].Del ;
48         tr[ i ].Cnt = tr[ rchild ].Cnt + Query_Tree ( lchild , tr[
rchild ].Del ) ;
49     }
50 }
51 void Build_Tree ( const int x , const int y , const int i ) {
52     tr[ i ].l = x ;
53     tr[ i ].r = y ;
54     if ( x == y ) {
55         if ( op[ x ] ) tr[ i ].Del = arr[ x ] ;
56         else if ( !op[ x ] ) {
57             tr[ i ].Add = 1 ;
58             tr[ i ].Cnt = arr[ x ] ;
59         }
60     }
61     else {
62         int mid = ( tr[ i ].l + tr[ i ].r ) >> 1 ;
63         Build_Tree ( x , mid , i<<1 ) ;
64         Build_Tree ( mid + 1 , y , i<<1|1 ) ;
65         Push_up ( i ) ;
66     }
67     return ;
68 }
69
70 void Update_Tree ( const int i , const int Target ) {
71     if ( tr[ i ].l == tr[ i ].r ) {
72         tr[ i ].Add = tr[ i ].Cnt = tr[ i ].Del = 0 ;
73         if ( INPUT ( ) ) {
74             tr[ i ].Del = INPUT ( ) ;
75         }
76         else {
77             tr[ i ].Cnt = INPUT ( ) ;
78             tr[ i ].Add = 1 ;
79         }
80     }
81     else {
82         int mid = ( tr[ i ].l + tr[ i ].r ) >> 1 ;
83         if ( Target > mid ) Update_Tree ( i << 1 | 1 , Target ) ;
84         else if( Target <= mid ) Update_Tree ( i << 1 , Target ) ;
85         Push_up ( i ) ;
86     }
87     return ;
88 }
89
90 int main ( ) {
91     int N , Q ;
92     freopen("weed.in" , "r" , stdin);
```

小红红

大神们

CtrlCV
Gster
y7070
晶姐

最新评论

1. Re:Tarjan 算法&模板
%%%

--蕭楓

2. Re:Tarjan 算法&模板
简洁明了的好文，顶！！
--代码中的爱因斯坦

3. Re:Tarjan 算法&模板
大佬
%%%%%%%%

--QSZIO

4. Re:《数据结构》线段树入...
真代码有毒
--今天没吃药

5. Re:20161022 NOIP模拟...
@にしきのまき向Dalao低头，
(orz..)...

--SHHHS

阅读排行榜

- 1. Tarjan 算法&模板(11703)
- 2. 匈牙利 算法&模板(5813)
- 3. 《数据结构》线段树入门...
- 4. SPFA算法(5304)
- 5. NOIP提高组2004 合并果...

评论排行榜

- 1. Tarjan 算法&模板(3)
- 2. NOIP 2013 货车运输【Kr...
- 3. 20161022 NOIP模拟赛 T...
- 4. 《数据结构》线段树入门...
- 5. Kosaraju 算法(1)

推荐排行榜

- 1. Tarjan 算法&模板(12)
- 2. 《数据结构》线段树入门...
- 3. 浅谈 LCA(4)
- 4. 匈牙利 算法&模板(4)
- 5. NOIP 2013 货车运输【Kr...

```
93     freopen("weed.out", "w", stdout);
94     scanf ( "%d%d" , &N , &Q ) ;
95     for ( int i=1 ; i<=N ; ++i ) scanf ("%d%d" , op + i , arr + i ) ;
96     Build_Tree ( 1 , N , 1 ) ;
97     while ( Q-- ) {
98         int tmp = INPUT ( ) ;
99         Update_Tree ( 1 , tmp ) ;
100         printf ( "%d\n" , tr[ 1 ].Cnt ) ;
101     }
102     fclose(stdin);
103     fclose(stdout);
104     Never Explode ;
105 }
```

NOIP_RP++;


2016-10-07 20:28:07

(完)

好文要顶

关注我

收藏该文

SHHHS
关注 - 34
粉丝 - 32
[+加关注](#)

00

« 上一篇：[20161005 NOIP 模拟赛 T3 解题报告](#)
» 下一篇：[20161007 NOIP 模拟赛 T1 解题报告](#)

posted @ 2016-10-07 12:48 SHHHS 阅读(120) 评论(0) 编辑 收藏

[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VC++源码: 大型组态工控、电力仿真CAD与GIS源码库！
- 【活动】2050 大会 - 博客园程序员团聚（5.25 杭州·云栖小镇）
- 【推荐】0元免费体验华为云服务
- 【活动】腾讯云招募自媒体，共享百万资源包

腾讯云

多规格高配云服务器

免费申请试用6个月

立即抢购



- 最新IT新闻:
- 滴滴被打乘客再次微博发长文：已找到司机嫌疑人
 - “天眼”发现毫秒脉冲星 引力波探测又添新可能
 - 王晓峰的体面告别：抵抗美团收购摩拜未果，25天后悲情出局
 - 知乎新隐私政策：不同意不能用 网友：不如换一键卸载？
 - 知名老赖贾跃亭堂而皇之广州拿地？国土部门被疑失职
- » 更多新闻...

阿里云

300+篇运维、数据库等
实战资料免费下载

点击获取

- 最新知识库文章:
- 如何识别人的技术能力和水平？
 - 写给自学者的入门指南
 - 和程序员谈恋爱

- 学会学习
- 优秀技术人的管理陷阱
- » 更多知识库文章...