2025/11/29 16:09 1/3 团队

团队

考试,摸了。

个人

zzh

pmxm

jsh

- 6/6 AtCoder Beginner Contest 163: pro: 6/6 rk: 14/?
- 6/7 Codeforces Round #648 (Div. 2): pro: 6/6/7 rk: ?/?
- 6/10 AtCoder Beginner Contest 164: pro: 5/6/6 rk: 18/?
- 6/11 Educational Codeforces Round 89 (Rated for Div. 2): pro: 5/5/7 rk: ?/?

本周推荐

zzh

pmxm

jsh

咱们的知识点里也有线段树合并,但是怎么没能证一下复杂度呢? 那下面我先证明一下线段树合并的复杂度,然后介绍一个稍微复杂点的应用情况。

线段树合并

线段树合并,即将两个区间一致的节点所对应的线段树,合并为新的线段树,返回出新的节点。

合并操作通常用递归来实现,所以一般非叶子节点的操作是合并两个分支后更新信息。 叶子则根据题目的需求来进行处理。

而为了保障复杂度,实现上,线段树需要懒惰申请节点,即只保存"有信息"的节点。所以当递归合并时, 某个节点为 nil□则可以不再继续递归。

线段树合并可行的前提是,在合并之后,被合并的两个树的根,不再会作为合并操作中被操作的节点。

递归过程中,若两个节点均不为 nill则说明要么是根,要么他们的父亲也均不为 nill 因此,合并时只要两个节点均不为 nill则说明这两个节点不再会作为合并操作中被操作的节点。

由于是二叉树,递归到的节点也都是包括根的一个连通子图,所以"递归的总次数"和"递归过程中两个

节点均不为 nil 的次数 " 同阶。

记第 \$i\$ 次合并前,可以用于合并的根的数量为 \$r i\$[记这些线段树总的节点数量为 \$k i\$ [重复也算 上)。显然 \$r_{i+1} = r_i - 2 + 1\$[

考虑递归合并的过程中,两个节点均不为 nil 的情况,由于它们不会再被拿来合并,对于 \$k {i+1}\$ 的贡 献就是 \$-2 + 1 = -1\$。 因此□\$k i\$ 是递减的,且"递归过程中两个节点均不为 nil 的次数",恰好等于 \$k_i - k_{i+1}\$[

综上, "递归过程中两个节点均不为 nil 的次数"的总和□\$(k_0 - k_1) + (k_1 - k_2) + \cdots\$□不会超过 最初的节点的数量。

因此总的时间复杂度和最初的节点数量等阶,通常就是 \$\mathcal{O}(n \log n)\$ 啦。

实现上如果确实合并掉的树没有用了,那合并时可以随便拿一个节点存信息,时间复杂度不变,空间也不 用新开。 而如果合并时均新建节点,旧的不删,则第 \$i\$ 次合并时,新的节点数量也等于 \$k_ik {i+1}\$□总的新建的节点不超过最初的节点的数量。所以可持久化时开两倍就好。

应用

容易发现,合并的两个线段树,如果存储的信息中有 key 相同的,则绝对会递归到存这个位置信息的叶子 上。

这是线段树合并最强大的性质,即我们能暴力对着公共 key 的信息进行操作。

例题 AtCoder Beginner Contest 163: F - path pass i

AtCoder Beginner Contest 163: F - path pass i

这个题目我的做法需要用到线段树合并。

题意

题意为给一个节点有颜色的树,对于每种颜色 \$k\$□计算一下有多少个不同的简单路径,至少经过一次颜 色为 \$k\$ 的节点。

我的做法

我的做法是在 DFS 的过程中,直接用线段树维护一下每个颜色作为 key□对应的不同的方案数。 每次 DFS 完一个子树后,就把返回的线段树和当前节点维护的线段树"想办法"合并一下。

对于线段树 \$A\$□记对于颜色 \$k\$ 记录的答案信息为 \$a {A,k}\$□ 为了能维护答案,还需要用线段树记录 当前根 \$u\$ 的子树中,所有"到当前根 \$u\$ 的路上经过有颜色为 \$k\$ 的节点"的不同节点数量。 记"到 \$u\$ 经过有颜色 \$k\$ 节点"的节点集合为 \$C {A, k}\$□大小为 \$c {A, k}\$□线段树只需要记录大小。

\$s u\$□ 再记刚 DFS 完的那个子树根为 \$v\$□子树节点集合为 \$S v\$□大小为 \$s v\$□返回的线段树为 \$A v\$□

Printed on 2025/11/29 16:09 https://wiki.cvbbacm.com/

2025/11/29 16:09 3/3 团队

记接下来合并好的线段树为 $A' u \cap A$ 如 分子的子树点数量为 S' u = S u + S v

对于维护的答案,此时需要新增 S_u 和 S_v 之间,每对点形成的简单路径的贡献。对于颜色 k 的答案[] a_{A_u} 的 a_{A_v} 为 $a_$

其中减掉一些信息,是因为 $C_{{A_u}, k}$ 和 $C_{{A_v}, k}$ 之间的简单路径被算了两次。

其中乘一个数更新答案,可以用懒惰标记来做,在合并、修改、询问时下传标记即可。

减掉的那部分答案,就需要用一下线段树合并的性质,来将公共颜色的节点之间的简单路径的贡献处理好。

特殊地,对于 $c_{A'_u}, \{k_u\} = s'_u$ \$的操作,由于只会进行边数乘 $\log n$ \$ 次,所以总复杂度是 $\mbox{mathcal}(0)(n \log n)$ \$\preceq\$

算上线段树合并的复杂度,即最初的节点数量,也即 DFS 最开始时,创建的只有一个节点信息的线段树,总节点数量为 \$\mathcal{O}(n \log n)\$[

所以这个算法的总的时间复杂度为 \$\mathcal{O}(n \log n)\$[]

后记

很麻烦?那就对了,因为这个不是正解,正解很简洁,而且是线性的时间复杂度,快去看 Tutorial 吧。

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

Permanent link:

Last update: 2020/06/13 01:09

