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 nil则说明要么是根，要么他们的父亲也均不为 nil 因此，合并时只要两个节点均不为 nil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，“递归的总次数”和“递归过程中两个节点均不为 null 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 null 的情况，此时合并掉的两个节点，因为它们的父节点不再会被拿来合并，所以它们也不会再被拿来合并，对于 k_{i+1} 的贡献就是 $-2+1$。因此 k_i 是递减的，且“递归过程中两个节点均不为 null 的次数”，恰好等于 $k_i - k_{i+1}$，即被合并掉、后续不再会被递归到的节点的数量。

综上，“递归过程中两个节点均不为 null 的次数”的总和，不会超过最初的节点的数量。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，复杂度不变（空间还小）。