

浅谈静态数据结构的合并与分裂

黄洛天

中国人民大学附属中学

January 30, 2024

线段树合并

线段树合并大家都了解，那么什么样的信息是线段树合并可以维护的？

为了维护信息，对于暴力的合并的节点我们有两种选择：

- ① 在合并完成子树后，从合并的后的两个儿子的信息合并上来。
- ② 直接合并对应节点的信息。

Examples

对于 01 序列，维护区间中 $[010, 01, 10, 0, 1]$ 子序列的出现次数。

Examples

线段树套平衡树，平衡树支持对 *key* 的平移和区间查询。

线段树合并

对于带懒标记的线段树合并，由于动态开点线段树在下放标记时会建立新的节点从而影响时间复杂度，因此我们需要用一些方法限制住新建点的数量。

点分树、边分树合并

类似于动态开点线段树，我们可以定义动态开点的点分树、边分树。为了方便叙述，我们只讨论点分树，边分树也是类似的。

由于点分树每个节点有很多儿子，我们难以支持自上而下的检索，一些信息也难以维护。因此考虑将点分树的儿子建哈夫曼树，不难证明树高是 $O(\log n)$ 的。

点分树、边分树合并

对于一类点分树问题，经常需要维护重心到当前连通块其他点的信息。那么对于每个节点 u ，他会影响的节点就是点分树上的父亲。由此可见如果我们做单点修改操作只有 $O(\log n)$ 个节点在点分树上维护的信息会变化。

静态 Top Tree 合并

Definition

对于无向树 T ，定义 T 上一个簇为一个三元组 $C = (V, E, B)$ ，其中 (V, E) 连通且是 T 的子图，集合 $B \subseteq V, |B| \leq 2$ ，且对于每个 $x \in V$ ，若 $\exists y \notin V, (x, y) \in E(T)$ ，则 $x \in B$ 。 B 中元素称为 C 中的界点。我们用 $V(C), E(C)$ 分别表示 C 的点集和边集，用 $B(C)$ 表示 C 中的界点集合。

静态 Top Tree 合并

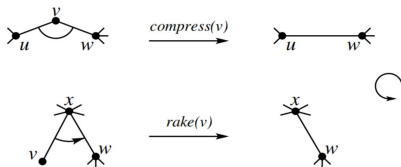
Definition

compress 操作是指将两个簇合并为一个簇的操作。其中，将两个界点集合交为 1 的簇合并成一个簇，新的界点集合为两个界点集合的对称差。

Definition

rake 操作是指将两个簇合并为一个簇的操作。其中，将两个界点集合交为 1 的簇合并成一个簇，新的界点集合为其中一个簇的界点集合。

静态 Top Tree 合并



对于一棵树来说，我们可以用上面两种操作，将整棵树进行收缩。具体来说，可以将一个簇看成是缩成一条界点 (u, v) 之间的边，初始每一条 (u, v) 的边都看成一个簇，通过用上面两种操作不断合并簇，最后可以把整棵树合并成一个簇。

收缩过程中，簇的合并结构形成了一个树，我们把这个树称为 Top Tree。

静态 Top Tree 合并

如何构建静态 Top Tree ?

对原树轻重链剖分，dfs 地自底向上处理每一个重链，并认为每一个轻子树都已经合并成一个簇。对于重链上的每一个节点，将这个节点的父亲的轻子树的簇分治地 rake，得到的总簇再用一次 rake 操作合并到这个节点的父边。需要保证深度不会太大，考虑每次分成两半，rake 完两半之后再 rake 成一个簇。我们称这样操作形成的树为 rake tree。然后，我们得到了排成一条链的簇。接着我们就可以将这些簇 compress 为一个，同样需要分治操作来保证深度，得到的树称为 compress tree。

静态 Top Tree 合并

Lemma

如果我们按簇的大小选择带权中点分治，那么这样建立出来的 Top Tree 深度为 $O(\log n)$ 。

无论是 rake tree 还是 compress tree，每向上跳常数层，簇的大小至少会翻倍，因此深度为 $O(\log n)$ 。

静态 Top Tree 合并

现在有了动态开点的静态 Top Tree，于是考虑怎么合并两个动态开点的静态 Top Tree。我们仍然自上而下的去合并 T_1, T_2 。

对于非叶子节点之间的合并，沿用之前线段树合并的思想，暴力递归下去。

我们要面对叶子节点和一个子树的合并。此时叶子节点上面带有两种标记——簇界点之间路径的标记和整个簇除了前者的标记。较为常见的情况是可以直接将两种标记打给另一个静态 Top Tree 的对应节点上。但如果不行需要针对簇界点之间路径的标记和整个簇除了前者的两套信息设计一个势能函数。

线段树分裂

线段树分裂支持把线段树分裂成两个值域不交的部分。注意
线段树分裂与线段树合并并不是逆操作的关系。

点分树、边分树分裂

点分树、边分树的局限性

数据结构分裂之后信息从儿子合并而来是很简单的。但如果想直接用原来的信息分裂，则必须维护的信息支持分裂。因此点分树边分树分裂对信息的要求高并且应用范围窄，不作为本文讨论的重点。

静态 Top Tree 分裂

考虑静态 Top Tree 的分裂出子树和子树补。

假设我们要分裂出 u 子树，可以在定位出 u 子树在静态 Top Tree 上对应的子树。即 u 节点所在 compress tree 中深度在 u 后面的节点，及其的 rake tree 子树，这一部分可以完整的分裂出来。对于 u 在静态 Top Tree 上的祖先节点由于子树内一部分在原树上的 u 子树，另一部分不在，因此需要复制到两棵树上。每次只会新建 $O(\log n)$ 个节点。

静态 Top Tree 分裂

处理完了子树的分裂，另一个想法是处理链的分裂。

面临到的问题

也就是说要将静态 Top Tree 中的一部分 compress tree 复制到 T' 的对应位置，但是这些 compress tree 连接的 rake tree 并不应该复制到 T' 的对应位置，而是应该保留在原来的位置。因此 T' 需要复制的节点并不是若干子树的形式，暴力剥离掉 compress tree 连接的 rake tree 的时间代价是不可接受的。

静态 Top Tree 分裂

由于静态 Top Tree 本身的结构导致了此问题，因此需要扩展出一个新的结构。考虑只支持修改查询链信息的静态 Top Tree。因此对于 compress node 的 rake tree 子树内的信息并不关键。也就是说不一定要让 rake tree 作为当前 compress node 的儿子。

考虑维护双层静态 Top Tree，将原本的每个点划分为 u 和 u' ，分别表示第一层和第二层。第一层内部的结构同静态 Top Tree 一样。第二层内部只建立 compress tree。对于每个 compress tree 的根节点 rt 有额外的一条边指向 rt' 。

静态 Top Tree 分裂

只有第二层维护链信息，第一层仅作为定位和重量平衡的用途。



不难发现上述结构，可以很好的维护链修改查询，并且具有每个节点儿子个数 $O(1)$ 且树高为 $O(\log n)$ 的性质。

静态 Top Tree 分裂

有了动态开点的双层静态 Top Tree，就可以合并和支持分裂出一个链了。前者是类似于普通静态 Top Tree 的。后者考虑我们要分裂的链在第二层的若干 compress tree 的区间上，而第二层的 compress tree 并不包含多余的 rake tree，因此这些子树可以被直接复制为 T' 的对应节点。这些 compress tree 上共用的节点，包括第一层静态 Top Tree 上公用的节点总数为 $O(\log n)$ ，可以新建出来。对于其余节点则保留了除了链以外的信息。

势能函数设计为 $\Phi = \sum |T_i|$ ，即静态 Top Tree 的节点个数，可以分析得到时间复杂度为 $O(n \log n)$ 。

参考文献

-  2023 年集训队论文, 程思元, 《浅谈静态 Top Tree 在树和广义串并联图上的应用》
-  赵雨扬, 《Top tree 相关东西的理论、用法和实现》
, <https://uoj.ac/blog/4912>