

CS447 Computer Algorithms. Notes 5: Splay Trees. Fall 2021

Let T be a splay tree. For a node v in T , let $n(v)$ be the number of nodes in the subtree of T rooted at v . The *rank* of v is given by $r(v) = \log_2 n(v)$. The rank of T is defined as the sum of the ranks of all nodes in T :

$$r(T) = \sum_{v \in T} r(v).$$

A splay tree supports the basic operations that a binary search tree does, namely, *search*, *insert* and *delete*. In a splay tree, for each such operation, we apply the standard binary search tree operation followed by an operation that splays a *special* node to the root. This last maneuver is what makes splay tree distinct from a binary search tree. For *search*, the special node is the node that was found by the search or the last node before the search failed. For *insert*, the special node is the newly inserted node. For *delete*, the special node is the parent of the deleted node.

Splaying a node to the root involves a sequence of splay steps, where a splay step can be one of **ZigZig**, **ZigZag**, or **Zig**. The first two operations move a node to replace its grandparent while the last operation moves a node to replace its parent. These operations must preserve the underlying binary search tree ordering present before the operations. We assume that the costs of these steps are given as follows.

Operation	Cost
ZigZig	2
ZigZag	2
Zig	1

Thus, the cost of splaying a node at depth d to the root is $O(d)$. More crucially, the cost of an operation (such as search, insert, or delete) on a splay tree is bounded by the cost of splaying the special node to the root.

Claim 1. *Let $O_1, O_2, \dots, O_m$ be an arbitrary sequence of operations on a splay tree which is initially empty. Then,*

$$\frac{1}{m} \sum_{i=1}^m \text{cost}(O_i) = O(\log_2 n)$$

where n is the maximum size of the splay tree.

Proof. We use a potential function argument. So, we define the cost of a splay operation (search, insert or delete) as the true cost of the operation (which is the cost of splaying the special node to the root) plus the change in the rank of the splay tree. That is, consider the i -th operation O_i and let T_{i-1} be the tree before operation O_i and let T_i be the tree after operation O_i . Then,

$$\widehat{\text{cost}}(O_i) = \text{cost}(O_i) + \Delta_i$$

where $\Delta_i = r(T_i) - r(T_{i-1})$. Note that

$$\sum_{i=1}^m \widehat{\text{cost}}(O_i) = \sum_{i=1}^m \text{cost}(O_i) + \Delta_m - \Delta_0$$

because the summation on Δ_i 's is telescoping. Since $\Delta_0 = 0$ and $\Delta_m \geq 0$, we have

$$\sum_{i=1}^m \text{cost}(O_i) \leq \sum_{i=1}^m \widehat{\text{cost}}(O_i).$$

To prove the claim, it suffices to show that $\widehat{\text{cost}}(O_i) = O(\log_2 n)$ for every i .

Suppose that O_i involves splaying a node x at depth d all the way to the root. Thus, this involves a sequence of $p = \lceil d/2 \rceil$ splay steps (consisting of one of **ZigZig**, **ZigZag** or **Zig** moves). Let δ_j be the change in the tree rank caused by the j -th step. Suppose that the j -th step involves moving x up one or two levels. We claim (for now) that

$$\delta_j \leq \begin{cases} 3(r_j(x) - r_{j-1}(x)) - 2 & \text{for ZigZig or ZigZag} \\ 3(r_j(x) - r_{j-1}(x)) & \text{for Zig} \end{cases} \quad (1)$$

where $r_{j-1}(x)$ and $r_j(x)$ are the ranks of x before and after the j -th step, respectively. So, we have that

$$\begin{aligned} \Delta_i &= \sum_{j=1}^p \delta_j \\ &\leq \sum_{j=1}^p [3(r_j(x) - r_{j-1}(x)) - 2] + 2 \\ &= 3(r_p(x) - r_0(x)) - d + 2. \end{aligned}$$

This implies that

$$\Delta_i + d \leq 3(r_p(x) - r_0(x)) + 2 = O(\log_2 n)$$

since $r_p(x)$ is the rank of the current tree (note x is the root now) which is bounded by $O(\log_2 n)$.

It remains to show that (1) holds. We show this for **ZigZig** (and leave the other two as exercise).

Consider a subtree with z as root, y as the left-child of z , and x as the left-child of y . Let T_1 and T_2 be the left-subtree and right-subtree of x , respectively. Let T_3 be the right-subtree of y and T_4 be the right-subtree of z . After a **ZigZig** step which moves x to replace z (at the root), we have x at the root with T_1 as its left-subtree and y as its right-child. The left-subtree of y is T_2 and the right-child of y is z . The node z has T_3 and T_4 as its left-subtree and right-subtree, respectively. See Figure 1.

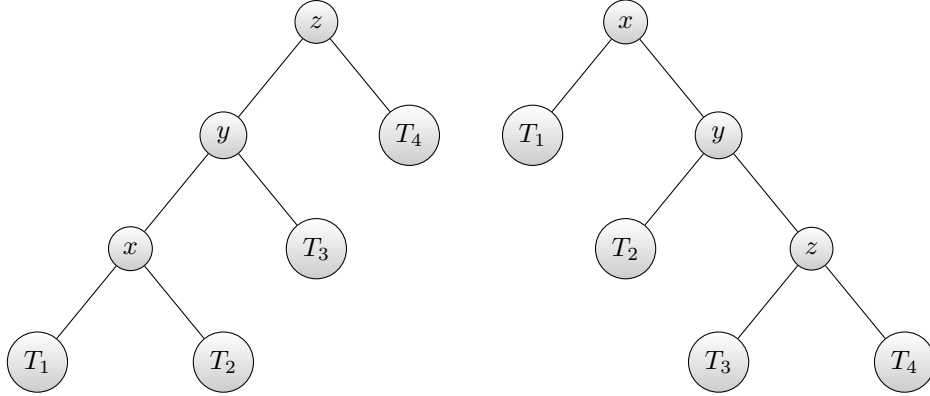


Figure 1: Left: before **ZigZig**. Right: after **ZigZig**.

Let the ranks of the nodes before the **ZigZig** step be denoted as $r(x)$, $r(y)$, and $r(z)$ while their ranks after are $r'(x)$, $r'(y)$ and $r'(z)$. Then, the variation in the tree rank is given by

$$\delta = r'(x) + r'(y) + r'(z) - r(x) - r(y) - r(z)$$

since the ranks of the other nodes did not change. Note that $r'(x) = r(z)$ because x is the root of the new tree and z is the root of the old tree. Also, note $r(y) \geq r(x)$ and $r'(y) \leq r'(x)$. Therefore,

$$\delta \leq r'(x) - 2r(x) + r'(z).$$

We now show that $r'(z) \leq 2r'(x) - r(x) - 2$, which proves the claim that $\delta \leq 3(r'(x) - r(x)) - 2$. It suffices to show

$$r'(z) + r(x) \leq 2r'(x) - 2.$$

By definition of rank, this is equivalent to

$$\log_2(n'(z)) + \log_2(n(x)) \leq 2\log_2(n'(x)) - 2.$$

By the rules of logarithms, we have

$$\log_2(\sqrt{n'(z)n(x)}) \leq \log_2(n'(x)/2).$$

By the monotonicity of logarithms,

$$\sqrt{n'(z)n(x)} \leq \frac{1}{2}n'(x). \tag{2}$$

Now, note that $n'(x) \geq n'(z) + n(x)$. Thus, (2) holds because $\sqrt{ab} \leq (a+b)/2$ for $a, b > 0$ (which is a special case of the Arithmetic-Geometric Mean inequality). $\square$