A Lecture on Splay Trees

Alex Waese-Perlman, Jun Kai Liao

March 26, 2023

This is the augmented transcript of a lecture given by Luc Devroye on the 9th of March, 2023 for a Data Structures and Algorithms class (COMP 252). The subject was splay trees.

Definitions

Definition 1. A *splay tree*[1] is a (not necessarily balanced) binary search tree that supports INSERT, DELETE, SEARCH queries in $O(\log n)$ amortized time. Invented by Sleator and Tarjan in 1985, it works by following each query with a *splay operation*. It is attractive since it can be implemented by using only the minimal information in each cell: LEFT, RIGHT and PARENT pointers, and KEY.

[1] Sleator and Tarjan [1985]

Definition 2. A *rotation* is an operation that swaps two adjacent nodes over their edge while preserving the properties of the search tree. Note that this is performed differently depending on whether the edge is pointing left or right.

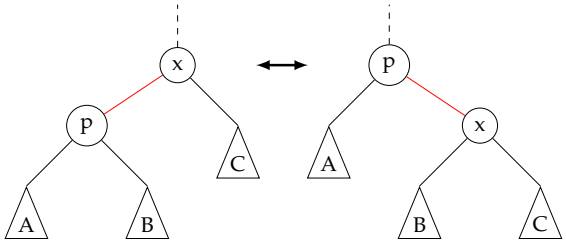

Figure 1: Rotating the red edge

Definition 3. A *splay step* uses rotations in a particular order to move a node one or two levels up. There are many different cases that define the *splay step* for a node x with parent p, and grandparent gp but we can reduce them into four cases. Let x be the node just inserted, or searched for, or the parent of of the deleted node.

Case 1. x is the root (p is undefined). Do not change anything.

Case 2. p is the root. Perform a rotation over the $x - p$ edge.

Case 3. The $x - p$ edge and the $p - gp$ edge point in the same direction. Rotate the $p - gp$ edge, followed by the $x - p$ edge. See figure 2.

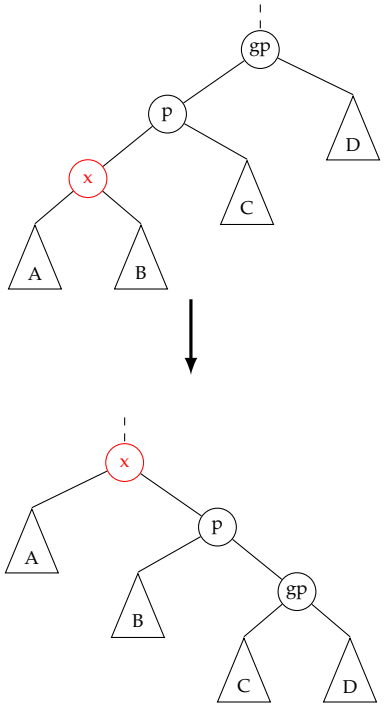

Figure 2: Case 3

Case 4. The $x - p$ edge and the $p - gp$ edge point in different directions. Here we will rotate the $x - p$ edge first, and then the (newly created) $x - gp$ edge. See figure 3.

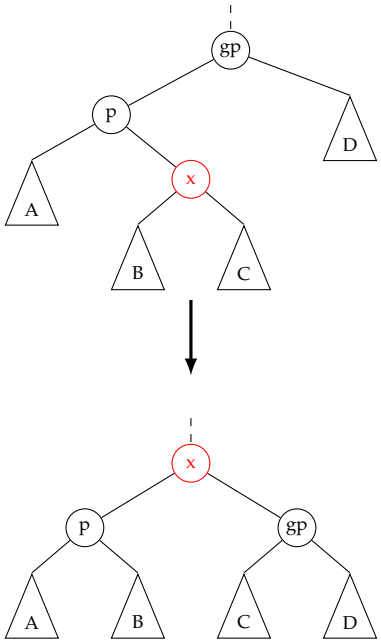

Figure 3: Case 4

Definition 4. A *splay operation* repeatedly performs *splay steps* on x until x is the root of the tree.

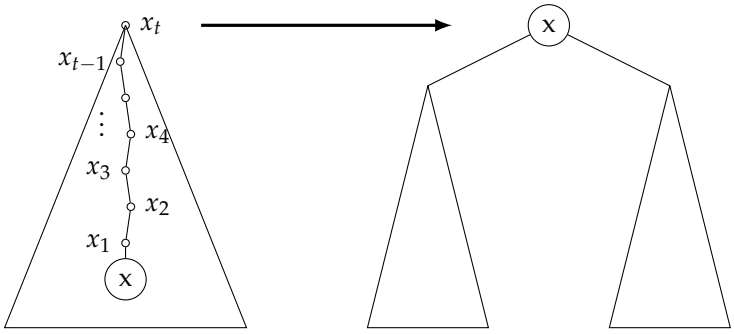

Figure 4: The splay operation

Potential function

We define the potential of the tree as

$$\begin{aligned}\Phi &= \sum_{u \in \text{tree}} \log_2 S_u \\ &= \log_2 \left(\prod_{u \in \text{tree}} S_u \right),\end{aligned}$$

where S_u is the size of subtree at u.

Φ is a measure of the unbalance of a tree, as illustrated in figures 5 and 6. When the tree is of size n, we have $\Phi = 0$ for $n = 0$ or 1. Φ is maximised for a chain, and equal to $\log_2 n! \approx n \log_2 n$. Φ is minimal for a perfectly balanced tree, which necessarily has height $\lfloor \log_2 n \rfloor$.

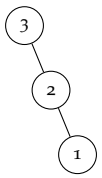

Figure 5:
$\Phi = \log_2 (3 \times 2 \times 1) = \log_2 6$.

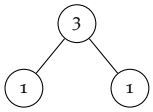

Figure 6:
$\Phi = \log_2 (3 \times 1 \times 1) = \log_2 3$.

Exercise 5. Show that $\Phi = \Theta(n)$ when the tree is perfectly balanced.

Analysis of Insert *and* Delete

Assume that rotations and comparisons both take one unit of time. If t denotes the path length to the element x that was inserted, deleted, or searched, the actual time of Insert, Delete, Search is $2t$, since we use t comparisons to find the path from the root to x, and t rotations to splay x.

We now verify how Φ changes for each operation, and denote the change $\log(\Delta\Phi)$. It is not hard to see that before we splay, SEARCH does not affect the structure of the tree, so it has $\Delta\Phi = 0$, and DELETE removes a node from its parents subtrees, so it has $\Delta\Phi \leq 0$.

For INSERT, things are more involved. Notice that only the nodes that are on the path down during INSERT see the size of their subtree increase. In particular, the size of the subtree for each of these nodes increases by 1. We can compute the differential of the potential $\Delta\Phi = \Phi\text{ (after)} - \Phi\text{ (before)}$, and refer to figure 4 for the notation $x_t, \ldots, x_1$.

$$\begin{array}{rl} \Phi\text{ (after)} = & \log_2(S_{x_t}+1) + \log_2(S_{x_{t-1}}+1) + \ldots + \log_2(S_{x_1}+1) + 0 \\ -\Phi\text{ (before)} = & \quad -\log_2(S_{x_t}) \quad\quad -\ldots-\log_2(S_{x_2}) \quad\quad -\log_2(S_{x_1}) \\ \hline \Delta\Phi \leq & \log_2(S_{x_t}+1) \end{array}$$

Notice that for all i, $\log_2(S_{x_{i-1}}+1) - \log_2(S_{x_i}) \leq 0$ since the size of the subtree at the ancestor is always at least one greater than its child. This allow us to cancel in the subtraction, and gives the upper bound of $\Delta\Phi \leq \log_2(S_{x_t}+1) = \log_2(n+1)$.

Analysis of the SPLAY *step*

For each case in definition 3, we need to compute $\Delta\Phi$. In what follows, a, b, c, d denote the sizes of subtrees A, B, C and D in figures 1, 2, and 3. Also p denotes the parent of x before the splay step, and gp the grandparent.

Case 1. $\Delta\Phi = 0$ since no rotation is performed.

Case 2. This step affects the subtree sizes of the node being rotated and its parent. Let A, B be the children of node x, and C be the sibling of node x.

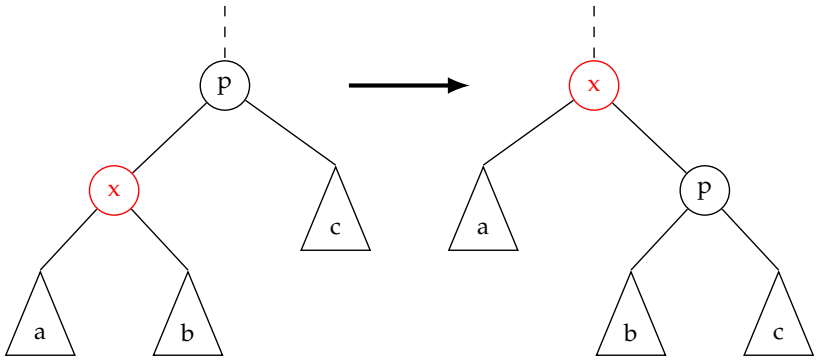

Figure 7: Case 2 — a, b, c denote subtree sizes.

$$\begin{aligned}\Delta\Phi &= \log_2(a+b+c+2) + \log_2(b+c+1) \\ &\quad - \log_2(a+b+c+2) - \log_2(a+b+1) \\ &= \log_2(b+c+1) - \log_2(a+b+1) \\ &\leq \log_2(S_p) - \log_2(S_x)\end{aligned}$$

Case 3. This affects the subtree sizes of x, p, and gp. We can compute their sizes before and after by looking at figure 8.

$$\begin{aligned}\Delta\Phi &= \log_2(c+d+1) + \log_2(b+c+d+2) \\ &\quad - \log_2(a+b+1) - \log_2(a+b+c+2)x'\end{aligned}$$

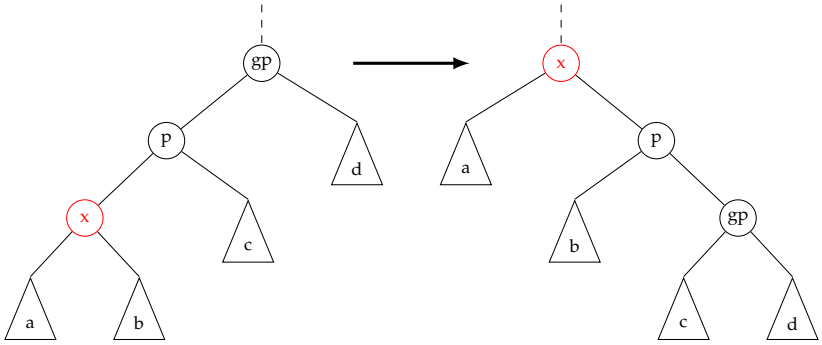

Figure 8: Case 3 — a, b, c, d denote subtree sizes.

We can upper bound positive terms

$$\begin{aligned}&\log_2(c+d+1) + \log_2(b+c+d+2) \\ &\quad \leq \log_2(c+d+1) + \log_2(a+b+c+d+2) \\ &\quad \leq \log_2(c+d+1) + \log_2(a+b+c+d+2) + \log_2(a+b+1) \\ &\quad \leq \log_2(a+b+1)(c+d+1) + \log_2(a+b+c+d+2) \\ &\quad \leq \log_2\left(\frac{a+b+1+c+d+1}{2}\right)^2 + 2\log_2(a+b+c+d+2) \\ &\quad \leq 3\log_2(a+b+c+d+2) - 2 \\ &\quad \leq 3\log_2 S_{gp} - 2\end{aligned}$$

by using the arithmetic/geometric mean inequality $ab \leq \left(\frac{a+b}{2}\right)^2$. Likewise, we can bound the negative terms by

$$\begin{aligned}&-\log_2(a+b+1) - \log_2(a+b+c+2) \\ &\qquad \leq -2\log_2(a+b+1) - \log_2(a+b+c+2) \\ &\qquad \leq -3\log_2(a+b+1) \\ &\qquad \leq -3\log_2 S_x.\end{aligned}$$

Hence

$$\Delta\Phi \leq 3\left(\log_2(S_{gp}) - \log_2(S_x)\right) - 2$$

Case 4. Looking at figure 9, we have

$$\begin{aligned}\Delta\Phi = \log_2(a+b+1) + \log_2(c+d+1) - \log_2(b+c+1) \\ - \log_2(a+b+c+2).\end{aligned}$$

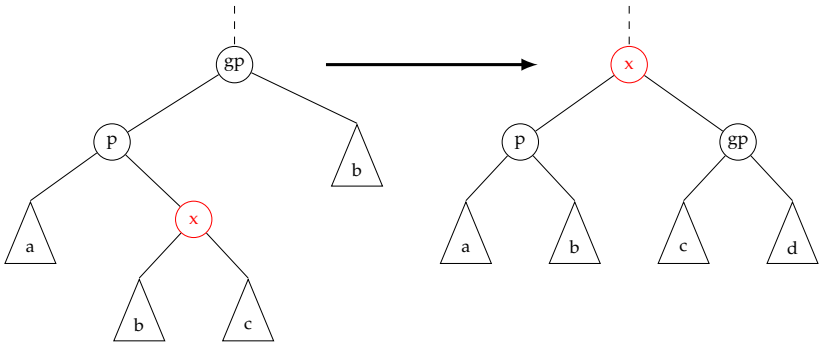

Figure 9: Case 4 — a, b, c, d denotes subtree sizes.

We can upper bound positive terms by

$$\begin{aligned}\log_2(a+b+1) + \log_2(c+d+1) &= \log_2\left((a+b+1)(c+d+1)\right) \\ &\leq \log_2\left(\frac{a+b+c+d+2}{2}\right)^2 \\ &\leq 2\log_2(a+b+c+d+3) - 2 \\ &= 2\log_2(S_{gp}) - 2.\end{aligned}$$

Likewise, the negative terms can be bounded easily

$$\begin{aligned}-\log_2(b+c+1) - \log_2(a+b+c+2) &\leq -2\log_2(b+c+1) \\ &= -2\log_2(S_x).\end{aligned}$$

This gives $\Delta\Phi \leq 2\left(\log_2 S_{gp} - \log_2 S_x\right) - 2$.

Global analysis

From the previous computation, for a splay step located at node x with parent p, and grandparent gp,

$$\Delta\Phi \leq \begin{cases} 0 & \text{if } x \text{ is the root} \\ \log_2(S_p) - \log_2(S_x) & \text{if } p \text{ is the root} \\ 3\left(\log_2(S_{gp}) - \log_2(S_x)\right) - 2 & \text{otherwise} \end{cases}$$

so $\Delta\Phi \leq 3\left(\log_2(S_{gp}) - \log_2(S_x)\right) - 2$. Since we perform $t/2$ splay steps to move a node to the root of the tree during the splay operation, the total $\Delta\Phi$ for the splay operation is

$$\begin{aligned} \sum_{\text{steps}} \Delta\Phi &\leq 3 \sum_{i=0}^{\lceil t/2 \rceil} \left(\log_2(S_{x_{2(i+1)}}) - \log_2(S_{x_{2i}})\right) - \sum_{i=0}^{\lceil t/2 \rceil} 2 \\ &\leq 3\log_2 n + 2 - t. \end{aligned}$$

The sum of all the splay operations telescopes to be less than or equal to $3(\log_2(n) - \log_2(S_x)) + 2 - 2t$. The plus two comes from the last step of the splay at the root. Notice that the $-t$ in our estimation is not sufficient to cancel the $2t$ from the actual cost of each operation. However, since we have arbitrary control over the choice of the potential function, we can define our potential to be twice the initial definition, and it will give the right bound to conclude that each operation is $O(\log n)$. The $\Delta\Phi$ of the splay operation is then $\leq 6\log_2 n - 2t + 4$.

We conclude by giving the table for the amortized cost of each equation:

Operation	Actual cost	$\Delta\Phi$	Amortized cost
Insert + Splay	$2t$	$\log_2(n+1)$ $+6(\log_2(n) - \log_2(S_x))$ $-2t + 4$	$7\log_2(n+1) + 4$
Delete + Splay	$2t$	$6\left(\log_2(n) - \log_2(S_x)\right)$ $-2t + 4$	$6\log_2(n) + 4$
Search + Splay	$2t$	$6\left(\log_2(n) - \log_2(S_x)\right)$ $-2t + 4$	$6\log_2(n) + 4$

Table 1: Amortized cost of Insert, Delete, Search + Splay in a splay tree of size n.

References

Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, jul 1985. ISSN 0004-5411. DOI: 10.1145/3828.3835. URL https://doi.org/10.1145/3828.3835.