Practice 14: Amortized analysis of Red – Black trees

Mayra Cristina Berrones Reyes

May 1, 2020

1 Introduction

Amortized analysis, as its name portraits, is a worst case analysis of a sequence of operations. This is used for algorithms where we have a majority of fast operations, commonly known as cheap operations, but every now and then we find a very slow operation, or expensive operation. Amortized analysis allows us to have an average worst case time that is lower than the worst case time of a particular expensive operation. There are three techniques used for this analysis [CIS, 2011]:

1. **Aggregate method** where we analyse the total running time for a sequence of operations.
2. **Accounting method**, also known as the bankers method, is where we put an extra charge on cheap operations and use it to pay for the expensive operations later on.
3. **Potential method**, also known as the physicists method, we have to derive a potential function the extra amount of work we can do in every step. The potential either increases or decreases with each operation, but it can never go below zero.

2 Red – black tree

We previously worked with red – black trees. They are a self balancing binary search tree, where we need to follow some rules. It is important that we remember them, because they determine the time complexity of some of the operations.

1. Every node has a color, either red or black.
2. The root of the tree must always be black.

3. There can not be two adjacent red nodes, which means that a red node can not have a red parent or red child.
4. Every path from the root node to any of the null decedents (leafs) has to have the same number of black nodes.

Now, the red – black tree has three main operations. Insertion, deletion and search. For the insertion and deletion operation, we need to make some changes to the tree, so it keeps complying with the features we mentioned before, so we have other operations inside the insertion and deletion, which are rotate left, rotate right, an change the color of the node, and we use them depending on where the node we inserted or deleted is.

3 Amortized analysis of red – black tree

For this amortized analysis we are using the **potential method**. So, firstly we suppose that we can define the potential function Φ on states of a data structure with the following properties[Rhodes, 2020]:

- $\Phi(h_0) = 0$, where h_0 is the initial state of the data structure.
- $\Phi(h_m) \leq 0$, for all states h_m of the data structure during the computation.

The way this analysis works is similar to the concept of the bankers method in the way that keeps track of the balance we have available to pay for more expensive operations, but in this case we take into consideration the whole history of the computation of the state we are in.

Now we can define the amortized time of an operation as Equation 1 where T_i is the actual cost of the operation and $\Phi(h_i)$ and $\Phi(h_i - 1)$ are the states of the data after and before the operation respectively. Thus the amortized time is the actual time plus the change in potential. Ideally, the change in potential should be positive for low-cost operations and negative for high-cost operations. [Demaine, 2016]

$$A_i = T_i - \Phi(h_i) - \Phi(h_i) \tag{1}$$

Now consider a sequence of n operations taking actual times $c_0, c_1, c_2, ..., c_n$?1 and producing data structures $h_1, h_2, ..., h_n$ starting from h_0. The total amortized time is the sum of the individual amortized times:

Table 1: Credit assignment for the tree

Features	**Credits assigned**
A black node with 1 red child	0
A black node with 2 black children	1
A black node with 2 red children	2

Table 2: Rules for assigning credit in insertion operation

Features	**Credits assigned**
Attach a node to a black node and terminate insertion	-1
Update colors and move up	-1
Perform single/double rotation and terminate	+2

$$\begin{aligned}
&(c_0 + \Phi(h_1) - \Phi(h_0)) + (c_1 + \Phi(h_2) - \Phi(h_1)) + ... + (c_n - 1 + \Phi(h_n) - \Phi(h_n - 1)),\\
&= c_0 + c_1 + ...c_n - 1 + \Phi(h_n) - \Phi(h_0)\\
&= c_0 + c_1 + ... + c_n - 1 + \Phi(h_n).
\end{aligned} \tag{2}$$

Therefore the amortized time for a sequence of operations overestimates of the actual time by $\Phi(h_n)$, which by assumption is always positive. Thus the total amortized time is always an upper bound on the actual time. Now we can take Equation 2 and clean it to look like Equation 3 [Ashish Bahendwar et al., 2018].

$$\sum_{i-1}^{m} T_i = \sum_{i-1}^{m}(A_i - \Phi(h_n) - \Phi(h_n - 1)) = \Phi(h_0) - \Phi(h_m) + \sum_{i-1}^{m} A_i. \tag{3}$$

Now that we explained our equations, we need to establish the rules and the credits that we are going to assign.

4 Example

We have a red – black tree in Figure 1 and then we use the insert function to insert the number 25. In Figure 1 we see the before and after of the tree, and in Table 5 we see how we can calculate de potential with the equations seen previously and following the rules on Table 1, 2, and 3. T_i takes the value of 4, because in Table 4 it says that this value is going to be as many nodes were

Table 3: Rules for assigning credit in deletion operation

Features	**Credits assigned**
Delete a black node and move up	-1
Update colors and move up	-2
Update colors and terminate deletion	-1
Perform single rotation and terminate deletion	-1 / +2
Perform double rotation and terminate deletion	+2

Table 4: Assumed value of T_i

Features	**Credits assigned**
Left rotate	1
Right rotate	1
Recolor node	Number of nodes recolored

Table 5: Potential before and after de insertion operation of the node 25

Nodes	8	5	17	15	18	25	Potential
$\Phi(h_0)$	1	1	2	-	-	-	4
$\Phi(h_1)$	0	1	-	1	0	-	2

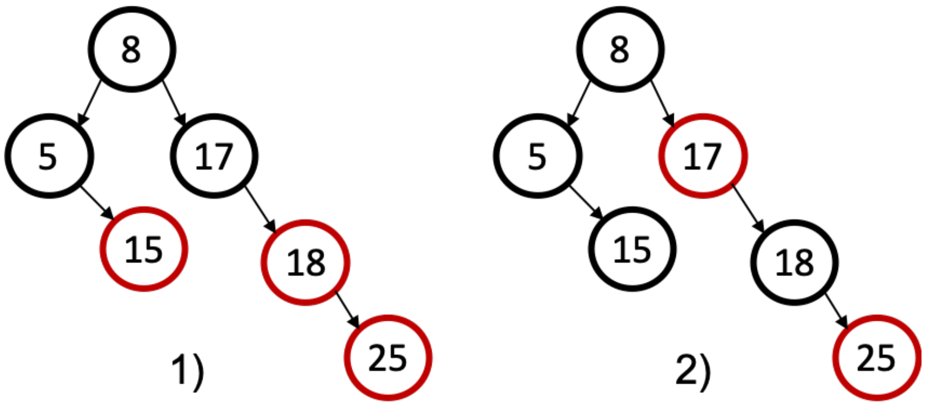

Figure 1: Example of an insertion in a red – black tree.

Table 6: Potential before and after the deletion of node 40

Nodes	8	5	17	15	25	18	40	Potential
$\Phi(h_0)$	0	1	-	1	2	-	-	4
$\Phi(h_1)$	0	1	-	1	0	1	-	3

recolored, and in the example we have 4 nodes that changed their color.

$$\begin{aligned} A_i &= T_i - \Phi(h_i) - \Phi(h_i) \\ &= 4 - 2 - 4 = -2 \end{aligned} \tag{4}$$

So now we have

$$\begin{aligned} &= \Phi(h_0) - \Phi(h_m) + \sum_{i-1}^{m} A_i \\ &= 4 - 2 + (-2) = 0 \end{aligned} \tag{5}$$

Now for the deletion function we have the potential from before and after in Table 5 and Figure 2 we can see how the tree accommodates to loose the node 40.

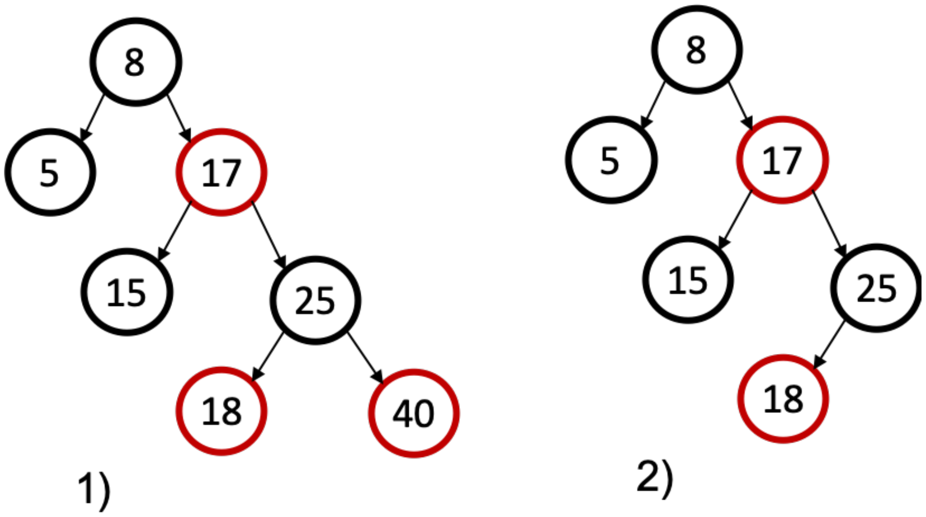

Figure 2: Example of a deletion in a red – black tree.

5 Conclusions

Again with this practice, it wasn't until I made an example by hand that I fully understood the problem of amortization. In this case, I had to re read the red – black trees, and I found that the time complexity for this algorithm is:

- Insert – $\mathcal{O}(\log n)$
- Delete – $\mathcal{O}(\log n)$
- Search – $\mathcal{O}(\log n)$
- Rotate right – $\mathcal{O}(1)$
- Rotate left – $\mathcal{O}(1)$

And as I understood, the purpose of the amortized cost is to be sort of like a upper bound, to have a more accurate prediction of the performance the algorithm may have. In this case, and after reading about the Accounting method, it sounded like I have to give a bigger cost to the "cheaper" operations, so that I have like saved credit for the more expensive ones. In my case, rotate left and right are the cheapest ones, and they also happen quite often, because they are used in the insertion and delete operations.

According to Figure 3 the next best notation is $\mathcal{O}(\log n)$.

At this point I don't know if I am understanding it correctly, but my amortized costs are:

- Insert – $\mathcal{O}(\log n)$
- Delete – $\mathcal{O}(\log n)$
- Search – $\mathcal{O}(\log n)$
- Rotate right – $\mathcal{O}(\log n)$
- Rotate left – $\mathcal{O}(\log n)$

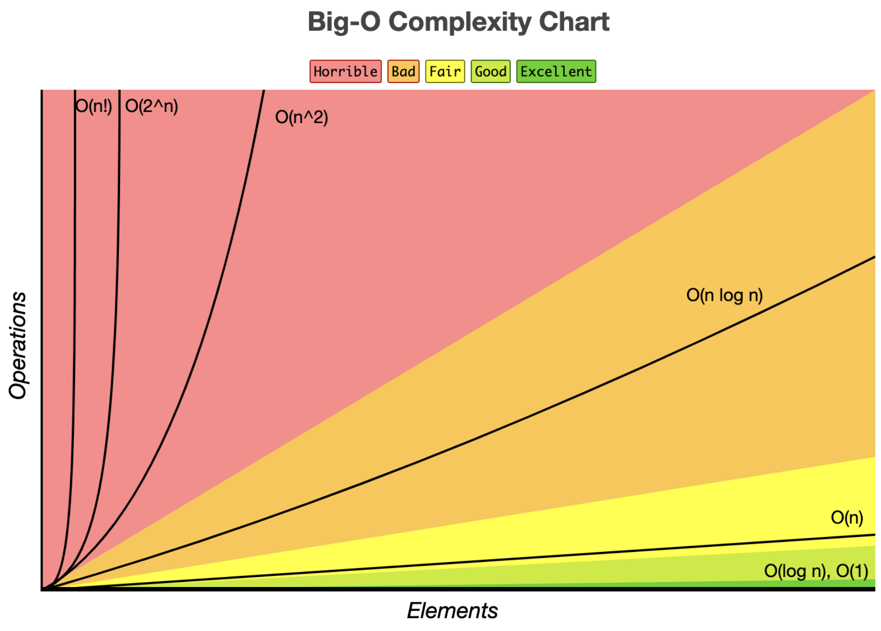

Figure 3: Big-O Complexity Chart from `https://www.bigocheatsheet.com/`.

References

[Ashish Bahendwar et al., 2018] Ashish Bahendwar, I., Bhardwaj, R., and Mundada, S. (2018). Amortized complexity analysis for red-black trees and splay trees. *Isha Ashish Bahendwar, RuchitPurshottam Bhardwaj, Prof. SG Mundada (2018) Amortized Complexity Analysis for Red-Black Trees and Splay Trees IJIRCST*, 6.

[CIS, 2011] CIS, C. U. (2011). Lecture 20: Amortized analysis. Available: `"http://www.cs.cornell.edu/courses/cs3110/2011sp/Lectures/lec20-amortized/amortized.htm"`.

[Demaine, 2016] Demaine, E. (2016). Amortization: Amortized analysis. Available: `"https://www.youtube.com/watch?v=3MpzavN3Mco&t=237s"`.

[Rhodes, 2020] Rhodes, N. (2020). Amortized analysis: Bankers method. Available `"https://www.coursera.org/lecture/data-structures/amortized-analysis-bankers-method-X6a5I"`.