Circular Buffer Splay Tree Delete Complexity Proof

Shyamal Suhana Chandra

1 Theorem: Circular Buffer Splay Tree Delete Complexity

Statement: Deleting from a Circular Buffer Splay Tree with n nodes takes $O(\log n)$ amortized time.

2 Proof

The delete operation consists of:

1. Find node to delete: $O(\log n)$
2. Splay node to root: $O(\log n)$ amortized
3. Delete node: $O(\log n)$ worst case
4. Deallocate from buffer: $O(1)$

2.1 Deletion Cases

2.1.1 Case 1: Leaf Node

- Remove from parent: $O(1)$
- Update subtree sizes: $O(\log n)$ (path to root)
- Deallocate: $O(1)$
- Total: $O(\log n)$

2.1.2 Case 2: One Child

- Replace with child: $O(1)$
- Update subtree sizes: $O(\log n)$
- Deallocate: $O(1)$
- Total: $O(\log n)$

2.1.3 Case 3: Two Children

- Find successor: $O(\log n)$ (height of right subtree)
- Replace node with successor: $O(1)$
- Remove successor: $O(\log n)$ (recursive)
- Update subtree sizes: $O(\log n)$
- Deallocate: $O(1)$
- Total: $O(\log n)$

2.2 Splay Operation

Splaying the node to root before deletion:

$$\text{Amortized cost} = O(\log n)$$

2.3 Buffer Deallocation

Deallocating from circular buffer:

- Clear buffer slot: $O(1)$
- Update size counter: $O(1)$
- Total: $O(1)$

2.4 Total Complexity

$$
\begin{aligned}
T(n) &= \text{Find} + \text{Splay} + \text{Delete} + \text{Deallocate} && (1)\\
&= O(\log n) + O(\log n) + O(\log n) + O(1) && (2)\\
&= O(\log n) \text{ amortized} && (3)
\end{aligned}
$$

Conclusion: Circular Buffer Splay Tree delete has $O(\log n)$ amortized time complexity.

3 Worst Case

In the worst case (linear tree), a single deletion may take $O(n)$, but amortized over a sequence of operations, the complexity is $O(\log n)$.