Circular Buffer Splay Tree Search Complexity Proof

Shyamal Suhana Chandra

1 Theorem: Circular Buffer Splay Tree Search Complexity

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

2 Proof

The search operation consists of:

1. Finding the node: $O(\log n)$ worst case (tree height)
2. Splay operation: $O(\log n)$ amortized (from standard splay tree analysis)
3. Buffer lookup: $O(1)$ (direct index access)

2.1 Splay Tree Search Analysis

Using the potential method with potential function:

$$\Phi(T) = \sum_{v \in T} \log(\text{size}(v))$$

where $\text{size}(v)$ = number of nodes in subtree rooted at v.
For a search operation:

$$\begin{aligned}
\text{Amortized cost} &= \text{Actual cost} + \Delta\Phi && (1)\\
&= O(\log n) + O(\log n) && (2)\\
&= O(\log n) && (3)
\end{aligned}$$

2.2 Circular Buffer Overhead

The circular buffer adds $O(1)$ overhead:

- Buffer index access: $O(1)$
- Node allocation/deallocation: $O(1)$
- LRU eviction: $O(1)$ per operation (amortized)

2.3 Total Complexity

$$\begin{aligned} T(n) &= \text{Find node} + \text{Splay} + \text{Buffer operations} \quad (4) \\ &= O(\log n) + O(\log n) + O(1) \quad (5) \\ &= O(\log n) \text{ amortized} \quad (6) \end{aligned}$$

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

3 Best Case

When the searched node is at the root: $O(1)$

4 Worst Case

When the tree is a linear chain: $O(n)$ for a single operation, but amortized over a sequence of operations: $O(\log n)$