Circular Buffer Splay Tree Insert Complexity Proof

Shyamal Suhana Chandra

1 Theorem: Circular Buffer Splay Tree Insert Complexity

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

2 Proof

The insert operation consists of:

1. Find insertion point: $O(\log n)$
2. Allocate node from buffer: $O(1)$
3. Insert node: $O(1)$
4. Splay new node to root: $O(\log n)$ amortized
5. Handle buffer overflow (if needed): $O(\log n)$ amortized

2.1 Insertion Analysis

Finding the insertion point requires traversing from root to leaf:

$$\text{Height} \leq \log_2(n+1) = O(\log n)$$

2.2 Buffer Allocation

Buffer allocation is $O(1)$:

- Index calculation: $O(1)$
- Node creation: $O(1)$
- Buffer update: $O(1)$

2.3 Buffer Overflow Handling

When buffer is full ($n = \text{bufferSize}$):

- Find LRU node: $O(1)$ (using circular index)
- Remove LRU node: $O(\log n)$ (tree deletion)
- Occurs at most once per buffer size insertions
- Amortized cost: $\frac{O(\log n)}{\text{bufferSize}} = O(1)$ per insertion

2.4 Splay Operation

Splaying the newly inserted node to root:

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

2.5 Total Complexity

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

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

3 Space Complexity

Since the buffer size is fixed:

$$\text{Space} = O(\text{bufferSize}) = O(1) \text{ per node}$$

Total space: $O(n)$ where $n \leq \text{bufferSize}$.