Lower Bound on Compare-Based MinPQ Implementation

Introduction

We aim to prove that it is impossible to develop a compare-based implementation of the MinPQ (Minimum Priority Queue) API such that both the `insert` and `delete the minimum` operations can be performed using $O(\log\log N)$ comparisons. We will show that this contradicts the theoretical lower bounds for comparison-based algorithms.

MinPQ API and Comparison-Based Algorithms

- The **MinPQ API** consists of:
  - `Insert` operation: Adds a new element to the priority queue.
  - `Delete the minimum` operation: Removes and returns the smallest element from the priority queue.
- **Comparison-based algorithms** are algorithms that rely on comparing elements using comparison operators such as $<$, $>$, or $=$. The efficiency of such algorithms is analyzed in terms of the number of comparisons made.
- Known bounds for comparison-based priority queues:
  - For comparison-based priority queues, the **worst-case time complexity** for both the `insert` and `delete-min` operations (in data structures such as binary heaps, Fibonacci heaps, etc.) is $O(\log N)$.

Theoretical Limits: $O(N \log N)$ Bound

Insert Operation

The `insert` operation adds an element while maintaining the priority queue's order property. In a binary heap, this is done by placing the new element at the end of the heap and then "bubbling up" the element to restore the heap order. This takes $O(\log N)$ comparisons, where N is the number of elements in the priority queue, because the height of the binary heap is $O(\log N)$.

Delete the Minimum Operation

The `delete-min` operation removes the smallest element, replaces it with the last element, and then "bubbles down" to restore the heap property. This operation also requires $O(\log N)$ comparisons since the height of the heap is $O(\log N)$.

Attempt to Achieve $O(\log \log N)$

Attempting to achieve $O(\log \log N)$ comparisons for both operations contradicts the established lower bounds for comparison-based priority queues:

Logarithmic Depth of Trees

In a binary heap, the height of the tree is $O(\log N)$. Reducing the comparisons to $O(\log \log N)$ would require a data structure with height $O(\log \log N)$, which is shallower than a binary tree. However, no comparison-based data structure can achieve such a shallow height while maintaining the necessary properties of a priority queue.

Information-Theoretic Lower Bound

The information-theoretic lower bound for comparison-based sorting is $\Omega(N \log N)$, which implies that at least $\Omega(\log N)$ comparisons are required for each insert or delete-min operation. A priority queue can be used to sort N elements by inserting all elements and performing N delete-min operations, so any comparison-based priority queue must perform at least $\Omega(\log N)$ comparisons per operation. Reducing the time complexity to $O(\log \log N)$ would violate this lower bound, as it would allow sorting to be performed in $O(N \log \log N)$, which is faster than the known lower bound for comparison-based sorting.

Conclusion

It is **impossible** to develop a comparison-based implementation of the MinPQ API such that both `insert` and `delete the minimum` can be performed in $O(\log \log N)$ comparisons. The reason is that comparison-based algorithms have a well-established lower bound of $\Omega(N \log N)$ for sorting and priority queue operations. Reducing the complexity of these operations to $O(\log \log N)$ would violate this fundamental lower bound. Therefore, the best achievable time complexity for both `insert` and `delete the minimum` operations is $O(\log N)$.