3 Analysis of Heap Sort

Heapify Method

Average-case Time Complexity

For both random and sorted arrays, the average-case time complexity for creating a max-heap is $O(n)$, where n is the size of the array. This is because the heapify operation runs in $O(\log n)$ time for each node, and there are approximately $n/2$ nodes in the heap. Therefore, the total time complexity is $O(n \cdot \log \frac{n}{2})$, which simplifies to $O(n)$. The total number of swaps required in heap creation and sorting contributes to this time complexity.

Worst-case Time Complexity

The worst-case time complexity for creating a max-heap using the heapify method is also $O(n \cdot \log n)$. This is because the heapify operation can take $O(\log n)$ time for each node, and there are n nodes in the heap. The total number of swaps required in heap creation and sorting contributes to this time complexity.

OneByOne Method

Average-case Time Complexity

The average-case time complexity for creating a max-heap using the OneByOne method is also $O(n \cdot \log n)$. Although inserting elements one by one takes $O(\log n)$ time for each insertion (due to moving the element up the heap), the total time complexity for n insertions is $O(n \cdot \log n)$. The total number of swaps required in heap creation and sorting contributes to this time complexity.

Worst-case Time Complexity

Similarly, the worst-case time complexity for creating a max-heap using the OneByOne method is also $O(n \cdot \log n)$. The total number of swaps required in heap creation and sorting contributes to this time complexity.

In summary, both methods have the same time complexity characteristics, with the average-case and worst-case time complexities being $O(n \cdot \log n)$ for both creating a max-heap and heap sort algorithm. However, the actual number of swaps required may vary between the methods and input arrays.