Angelo Vita 3/28/23

Big-O is the standard way of measuring the time complexity of an algorithm. In this write up, I will be discussing two types of Big-O: Theoretical and experimental. Theoretical Big-O refers to the Big-O notation that is determined by analyzing and determining the time complexity based on the design and properties of an algorithm. This typically involves analyzing the algorithm’s code and determining the number of basic operations it takes to execute the code. The theoretical Big-O is often used to estimate the algorithm’s performance on larger inputs.

Experimental Big-O is determined by running the algorithm with various inputs and sizes and measuring the actual time it takes to run the algorithm. This typically involves running the algorithm multiple times with different input sizes and plotting the running time against the input size. The experimental Big-O is useful for verifying the theoretical Big-O and determining the actual performance of an algorithm on a specific computer.

In general, theoretical Big-O provides a useful estimate of the algorithm's performance, while experimental Big-O provides more accurate and realistic performance measurements. Both approaches are important for understanding the performance of an algorithm and making informed decisions about which algorithm to use in a given situation.

A fetch algorithm is used to retrieve a specific element from an array based on its index. For both unsorted and sorted arrays, the theoretical Big-O for a fetch algorithm is O(1). This is because it takes constant time to access any element in an array. However, the experimental Big-O notation for the fetch algorithm in an unsorted array is O(n). This is because in the worst-case scenario, the algorithm has to search through every element in the array to find the element you want. The experimental Big-O notation for the fetch algorithm in a sorted array is O(log n). This is because binary search can be used to find the element in a sorted array in a logarithmic amount of time.

An insert algorithm is used to insert an element into an array at any specific index. For an unsorted array, the theoretical Big-O is O(1) in the best case and O(n) in the worst case. This is because the algorithm may need to shift all elements in the array to make room for the new element. The theoretical Big-O for a sorted array is once again O(1) in the best case and O(n) in the worst case. The experimental Big-O notation for the insert algorithm in an unsorted array is difficult to determine precisely because it depends on the location of the insertion. However, on average, it is O(n/2) because the element may need to be inserted in the middle of the array, which requires shifting half of the elements. In a sorted array, the experimental Big-O notation for the insert algorithm is also O(n/2) on average, as half the elements may need to be shifted to find the correct position.

In conclusion, the theoretical and experimental Big-O notations for fetch and insert algorithms in sorted and unsorted arrays vary depending on the algorithm and the input size. Theoretical Big-O provides a theoretical analysis of the algorithm's time complexity based on the design and properties of that algorithm, while the experimental Big-O notation provides a more accurate representation of its performance in real-world scenarios by running the algorithm with various inputs and sizes and measuring the actual time it takes to run the algorithm. It is important to consider both the theoretical and experimental Big-O when designing and analyzing algorithms to ensure optimal performance in practical situations. Moreover, while the theoretical Big-O may give us an idea of how an algorithm performs, the experimental Big-O provides more concrete information, allowing for further optimization of the algorithm.