**Module 7 Assessment**

|  |
| --- |
| extraLargeArray |
| Insert 1.3203735 s |
| Append 4.0505 ms |

|  |
| --- |
| LargeArray |
| Insert 11.4789 ms |
| Append 700 us |
|  |

|  |
| --- |
| smallArray |
| Insert 56.7 us |
| Append 125.4 us |

|  |
| --- |
| tinyArray |
| Insert 48.4 us |
| Append 114.7 us |

Key: s = seconds; ms = milliseconds; us = microseconds

|  |
| --- |
| mediumArray |
| Insert 267.1 us |
| Append 180.8 us |

For the medium, large & extra-large arrays we pass in through the helper function, the append function will perform better than the insert function. As the size of the array being passed in increases, we see the separation in efficiency between the append function and the insert function increase as well. The append function will scale better between the two functions due to this.

Extra Credit answer: Because the insert function is linear time, the efficiency of the function depends on the size of the array being passed in. Additionally, when a new number is passed in, the unshift method wants to place the new item at the first index which already exists. This means in order to unshift the new data, each piece of data that already exists has to be moved one spot to the right in order to make room. This task will affect runtime at scale.

The append function is constant time and has a simple task of adding data to the end of an array through the push method. This method does not have the same issue as the insert function when it comes to shifting data. We simply append the data to the end of the array and that’s it. This makes the append function better suited at scale.