**Assessment 7**

|  |  |  |  |
| --- | --- | --- | --- |
| **Runtime** | | | |
| **Array Type** | **Insert Function** | **Append Function** | **Ratio** |
| extraLargeArray | 865.6146 ms | 3.2696 ms | |  | | --- | | 264.7463 | |
| largeArray | 9464.3 μs | 528.1 μs | 17.92142 |
| mediumArray | 273.2 μs | 309.7 μs | 0.882144 |
| smallArray | 9 μs | 8.7 μs | 1.034483 |
| tinyArray | 39.7 μs | 5.5 μs | 7.218182 |

The Insert function has a significantly longer runtime than the Append Function, with the exception of the medium array. I am not sure what is happening with the medium array and why the Insert Function is lower than the Append Function. When it comes to scaling, the Insert function’s differences are significantly larger than the Append Functions differences. As such, the Append function has a smaller difference, so the Append Function would be ‘better’.

(Extra Credit):

This makes sense because the Insert Function contains ‘unshift()’ which includes one other step other than adding to an array. That is, counting the array. This step creates a massive difference in time because there is another action going on.

Side note, there are some small inconsistencies with this report. Like the tinyArray’s Insert Function being larger than smallArray. And mediumArray’s Insert Function being smaller than it’s Append Function. I suspect there are many causes of this. Such as computer performance tasks in every given microsecond.