|  |  |  |
| --- | --- | --- |
| **Timing Results** | | |
| Array Size: | Insert Method | Append Method |
| Tiny (10) | **48.5** μs | **117.9** μs |
| Small (100) | **64.8** μs | **140.7** μs |
| Medium (1000) | **239.8** μs | **192.8** μs |
| Large (10000) | **12.587** ms | **0.748** ms |
| Extra Large (100000) | **1.425** s | **4.56** ms |

At first glance, the data appears to favor the Insert method over the Append method, for arrays smaller than 1000 elements. However, as the input data increases beyond 1000, the append method yields a much shorter runtime. The upward slope of Append is much more gradual, which bodes well for runtime given exponentially bigger databases. This would make Append the clear choice for scalability, and can be rationalized by considering the procedures required for each. While Append is merely adding an element into the end of the new array via .push, the Insert method is adding the element to the beginning of the new array and subsequently moving every previous element over by one index. As the input array increases, this can greatly affect runtime. The difference between the append(tiny) vs. append(extraLarge) is a mere 4.4 ms, while the difference using Insert is a whopping 1.4249 s. (I tried running an array with 1 mil elements, the execution-time pkg was not having it…)