Result of tinyArray:

- Insert = 35.41 μs (microsecond)
- Append = 95.153 μs

Result of smallArray:

- Insert = 46.36 μs
- Append = 106.566 μs

Result of mediumArray:

- Insert = 150.538 μs
- Append = 145.674 μs

Result of largeArray:

- Insert = 5.994891 ms
- Append = 539.557 μs

Result of extraLargeArray:

- Insert = 749.241307 ms
- Append = 3.750491 ms

After running all the tests, we see that the overall "winner" of the time complexity was the append method which essentially pushed the numbers in the array as they were being calculated.

WIth the unshift way (the insert method) initially it was able to be ahead since the size of itself was so small. But after the mediumArray size, it became the tipping point because it had a "further distance to run". Meaning that everytime the number was printed, it was immediately grabbed to be in the front of the array and that extra step added to the time complexity. As seen in the last test, which had 100,000 numbers, the simple action of adding them into the array and nothing more contributed to the faster processing of the function.