Results for the tinyArray
insert 134.599 µs
append 515.686 µs

Results for the smallArray
insert 101.996 µs
append 785.509 µs

Results for the mediumArray
insert 812.74 µs
append 1.428262 ms

Results for the largeArray
insert 39.246256 ms
append 2.430404 ms

Results for the extraLargeArray
insert 3.260770942 s
append 29.521677 ms

	Append	Insert
tinyArray	515.686 µs	134.599 µs
smallArray	785.509 µs	101.996 µs
mediumArray	1.428262 ms	812.74 µs
largeArray	2.430404 ms	39.246256 ms
extraLargeArray	29.521677 ms	3.260770942 s

Conclusion

While testing the two functions with the varied arrays I ran the code a number of times and there were some fluctuations between each run of any of the given arrays. With the results I gathered into the table above it seems that when passing in a smaller array through the Insert function the runtimes happened to be quicker than the Append function surprisingly (maybe just due to random collection while testing), but once computing the larger arrays it became quite clear that the Append function scaled much better then the Insert. Once there was a significant amount of data to compute it was obvious that Append was much more better at being scalable code.