|  |  |  |
| --- | --- | --- |
| Array size | doublerInsert runtime | doublerAppend runtime |
| 100000 | 1.0456556 s | 2.8638 ms |
| 10000 | 11.2078 ms | 632.6 us |
| 1000 | 212.1 us | 180.5 us |
| 100 | 54.6 us | 107.5 us |
| 10 | 43.9 us | 102.7 us |

Read over the results, and write a paragraph that explains the pattern you see. How does each function “scale”? Which of the two functions scales better? How can you tell?

As the array scales to a larger size, the doublerAppend function scales better than the doublerInsert function. For significantly larger array size, the doublerAppend function has significantly faster runtime than the doublerInsert function.

In the doublerAppend function, within the for loop there is a one operation for each multiplication and one operation for appending each product into the array. Array.push() has a constant time complexity and is O(1). It adds an element and give it an index that is 1 greater than the index of the last element in the array. Array.push() takes one operation to append the element into the array regardless of the size of the array. Together with the for loop, if the list is n long, the doublerAppend function performs 2n operations. The function is O(n).

In the doublerInsert function, within the for loop there is one operation for each multiplication and approximately n operations for inserting each product into the array. For Array.unshift(), adding an element at the beginning of an array means the new element will have an index of 0. This means that the index of every other element must be incremented by 1. If the list is n long, Array.unshift() performs n operations to insert the element at the beginning of the array. Array.unshift() has a linear time complexity and is O(n). Together with the for loop, if the list is n long, the doublerInsert function performs n2 operations. The function is is O(n2).

Since doublerAppend function has linear time complexity while doublerInsert function has quadratic time complexity, doublerAppend scales better.