Michael Roseweir

07/23/2021

CS Assessment Runtime Analysis

1. After you have read over the file yourself and made some guesses on what's happening, hover below to read a summary.

    a. After reviewing the code, it seems as though both of them are going to be used to double the value of an index within an array. The key difference between the two would be using the .push() and the .unshift() method to push the new number into the new array. I believe that the .push() method will be much faster at executing the function. The reason I believe this is because the .unshift() method pushes the new number to the beginning of the array, which causes the index of every following number in the array to change. This will be inefficient with the larger arrays.

2. It's time to run the file! Run the command node runtime.js in the same directory as where the file lives. In your notes document, take note of the timing result for the **extraLargeArray** results– comparing when the **extraLargeArray** is passed to **doublerAppend** and **doublerInsert**.

Array Size	Function	Time
extraLargeArray	Insert	2.6s
extraLargeArray	Append	5ms

3. Next, edit the code in **runtime.js** to obtain timing results for calling the two functions with all of the differently sized arrays– **tinyArray**, **smallArray**, **mediumArray**, **largeArray**, and **extraLargeArray**. Notate these in your document in some kind table table so that you can easily compare the different values for the timers in relation to the size of the array that was passed into each function.

	Insert (.unshift)	**Append (.push)**	**Fastest**
tinyArray	48us	114us	Insert (.unshift)
smallArray	89us	139us	Insert (.unshift)
mediumArray	308us	380us	Insert (.unshift)
largeArray	14ms	848us	Append (.push)
extraLargeArray	2.3s	4.1ms	Append (.push)

4. 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?

    a. When looking at the data I was interested to see that the unshift method performed faster with the lower array sizes. Although it is only by a couple microseconds, it was interesting to see that it can be quicker with smaller array sizes. However, as soon as the array starts to scale the push method becomes far quicker. On the extraLargeArray it executes the function in half the time.

5. For extra credit, do some review / research on why the slower function is so slow, and summarize the reasoning for this.

    a. The reason the .unshift() method is so much slower when it comes to the larger arrays is due to the fact that the method affects every piece of the array. It forces all the indexes to shift by 1 each time it pushes into the array. Using the .push() method only adds to the end of the array, without changing anything in the beginning of the array. When the array is scaled, that effect is multiplied drastically.