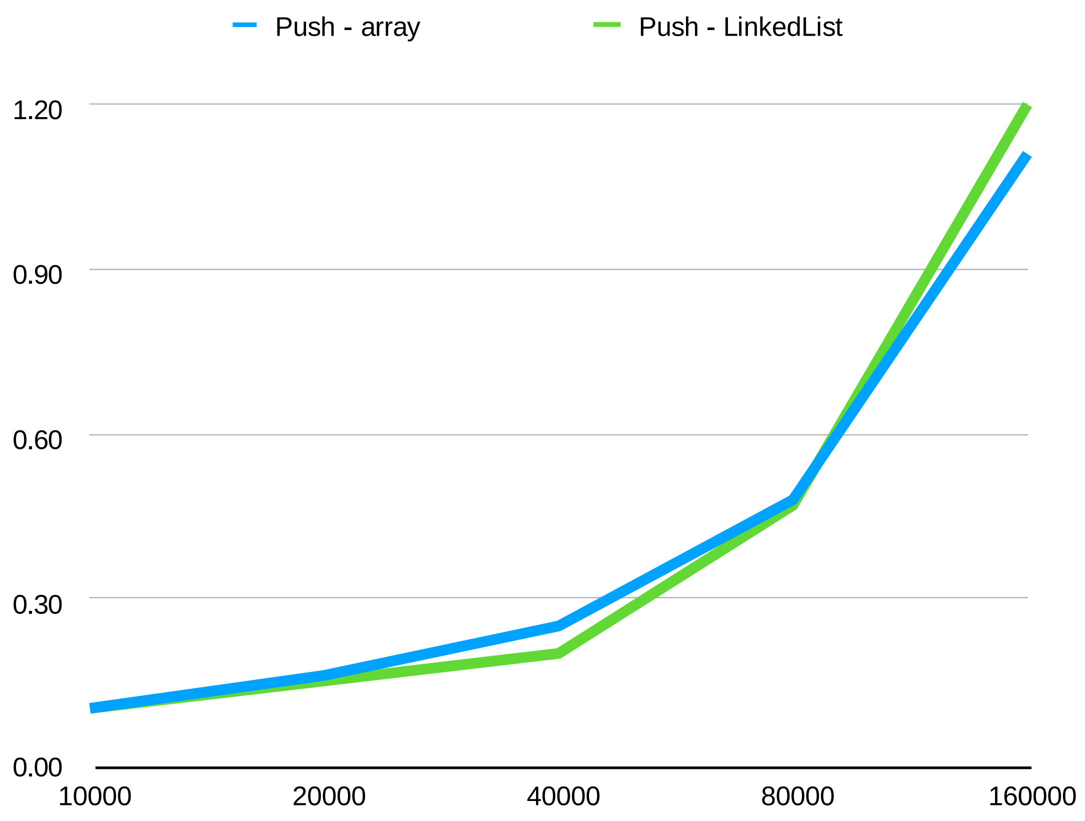
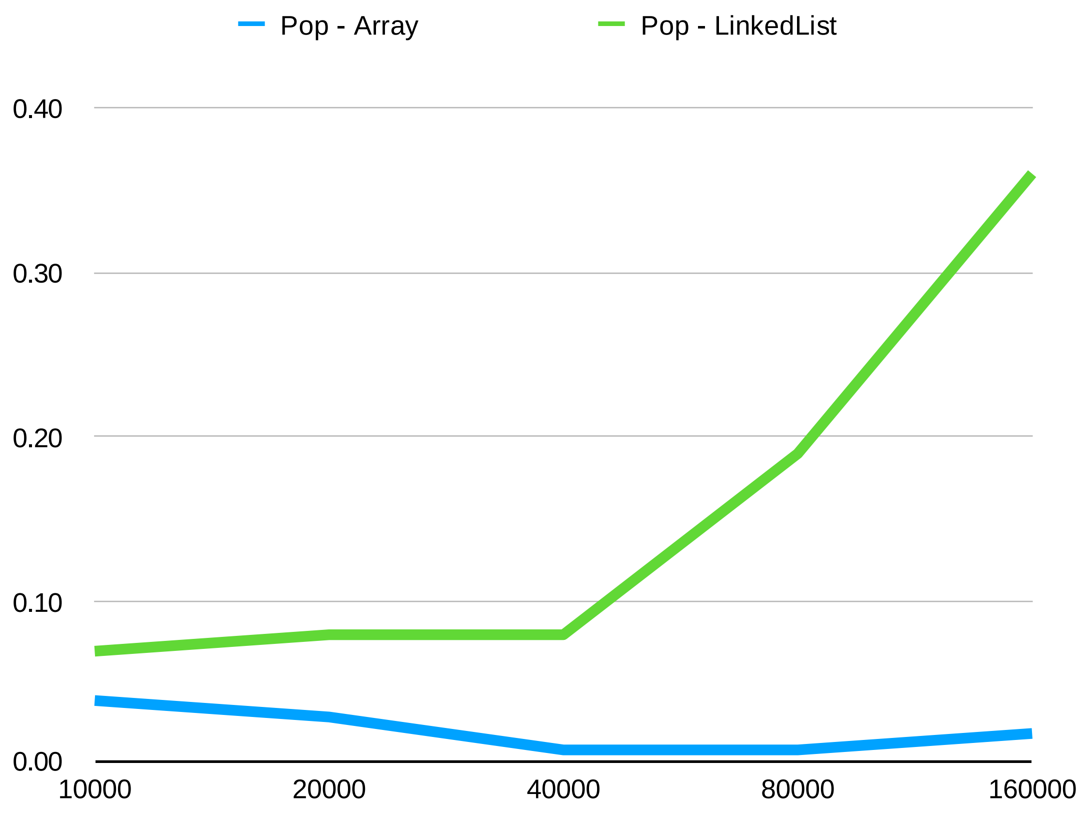
Compare the running time of the push method. What is the growth rate of the method's running time for each stack class, and why?



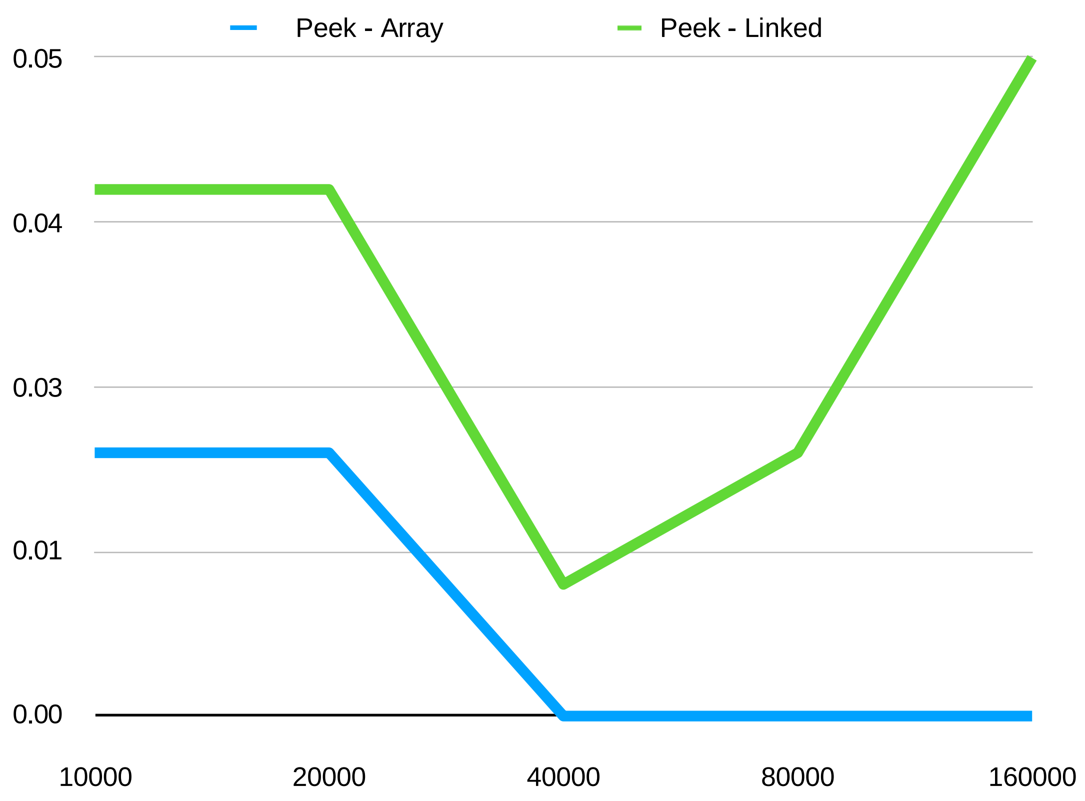
The push method was tested for both the array-based stack and the linked list-based stack. For sample sizes between 10,000 and 40,000, both performed similarly. After 40,000, the runtime increased quickly, but the array-based stack was slightly more stable with larger sizes.The array stack is usually fast for push, but it can slow down O(n)when the array resizes. The linked list stack is always O(1) for push, but it uses more memory, which may make it less stable with larger sizes.

Compare the running time of the pop method. What is the growth rate of the method's running time for each stack class, and why?



The pop method was tested for both the array stack and the linked list stack. The array stack performed much better, because it works with continuous memory, making it faster and more efficient. The array stack only needs to move the top pointer and access nearby memory, which is quick. The linked list stack, even though it's also O(1), is slower because it has to handle scattered memory and node references, which makes it less efficient for the CPU.

Compare the running time of the peek method. What is the growthrate of the method's running time for each stack class, and why?



The peek method was tested for both the array stack and the linked list stack, and the array stack performed better. For the array stack, peek accesses the top element in continuous memory, making it very fast and efficient. The linked list stack also has (O(1)growth for peek, but it is slightly slower because it involves accessing a node, which is less efficient for the CPU.

Based on your timing experiments, which stack class do you think is more efficient for using in your WebBrowser application? Why?

Maybe array stack will be more efficient to using in the WebBrowser application, he linked list stack is less efficient because it uses more memory and has slower memory access due to node locations, making it less suitable for the frequent stack operations required in a web browser.