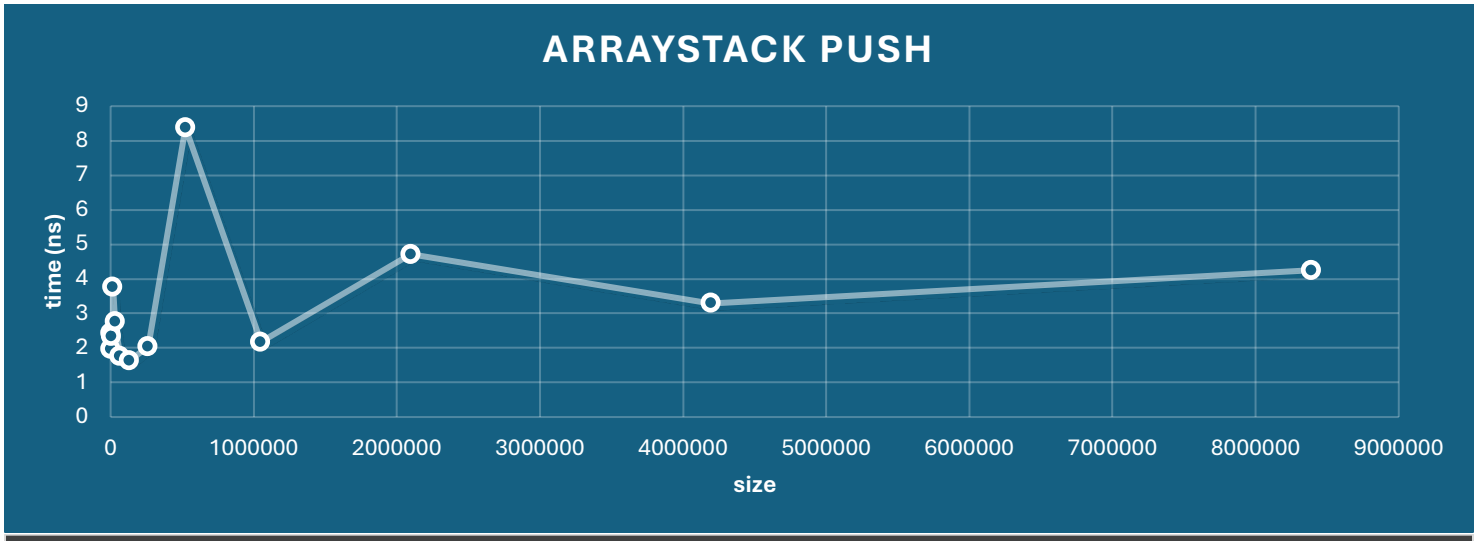
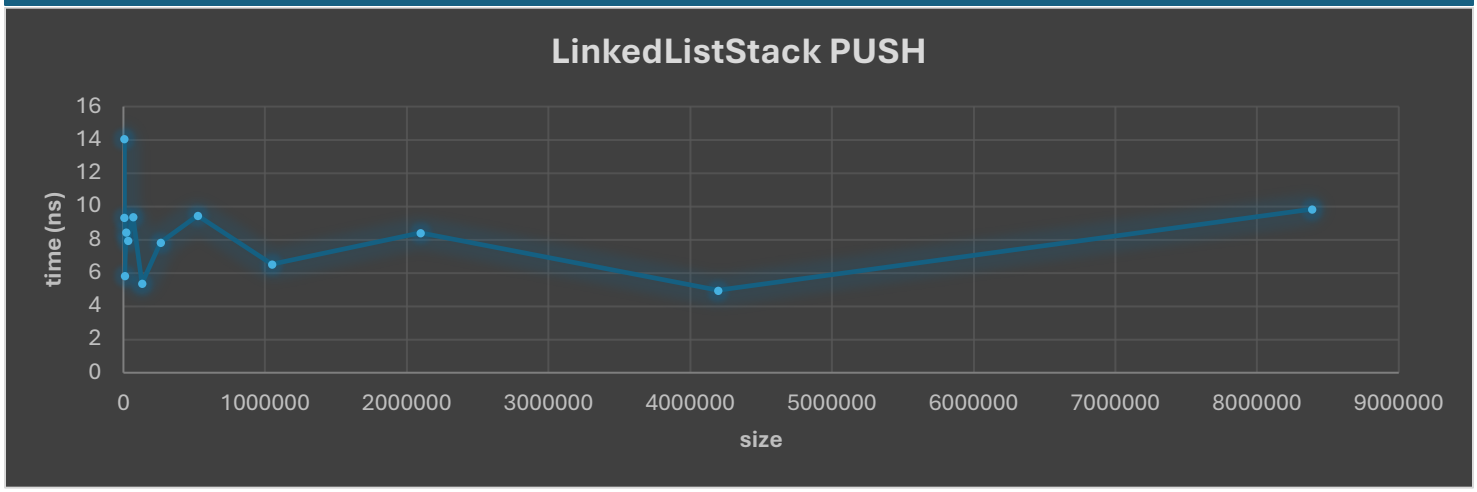
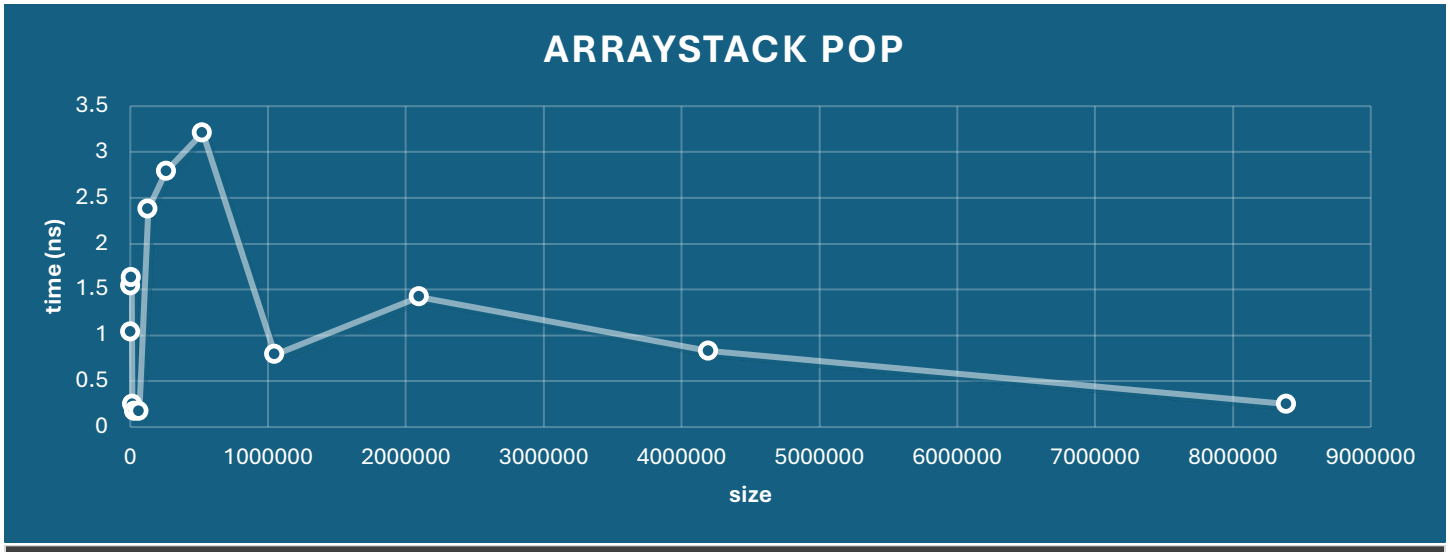
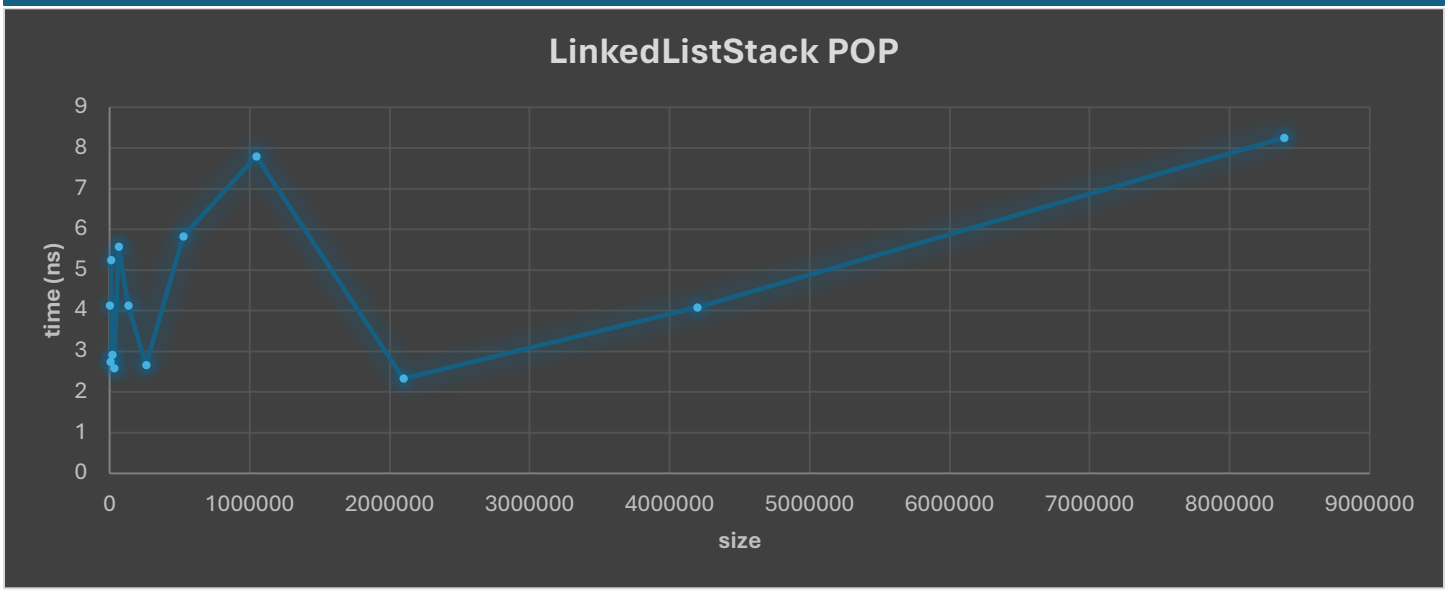




- Compare the running time of the push method. What is the growth rate of the method's running time for each stack class, and why?

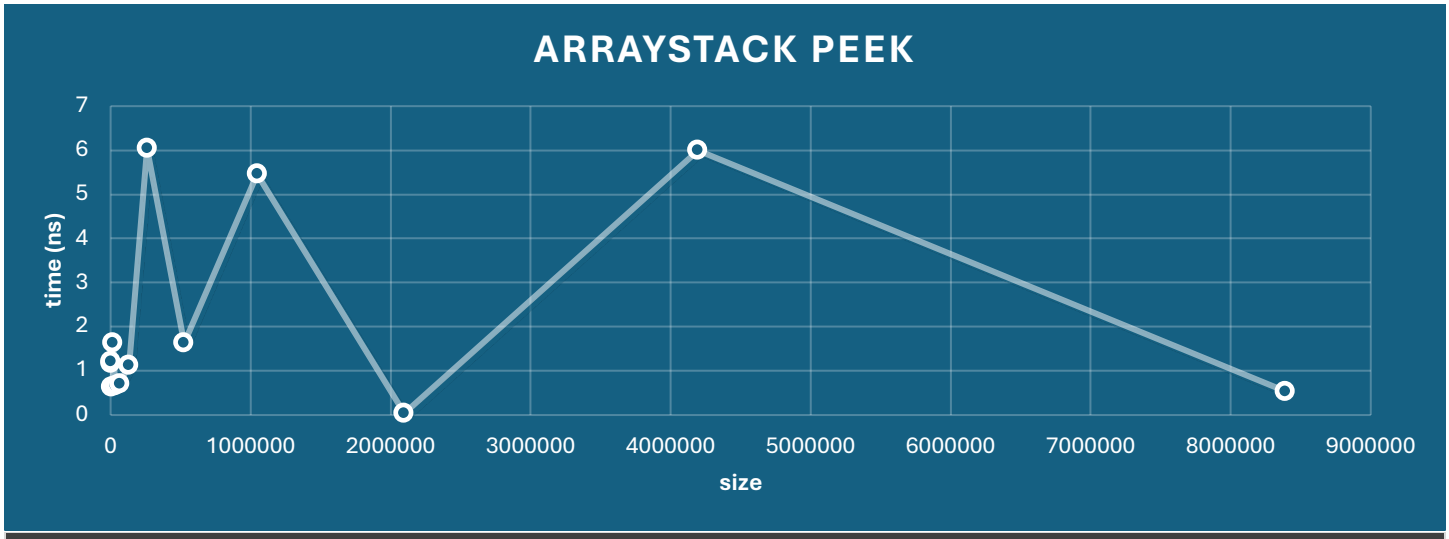
  - Both array stack and linked list stack's push method running time is O(1).
  - For array stack, the push method is adding an element which is typically just assigning a value to the next available index in the array.
  - For linked list stack, the push method is adding an element to the top of the stack which involves creating a new node and updating the head pointer.

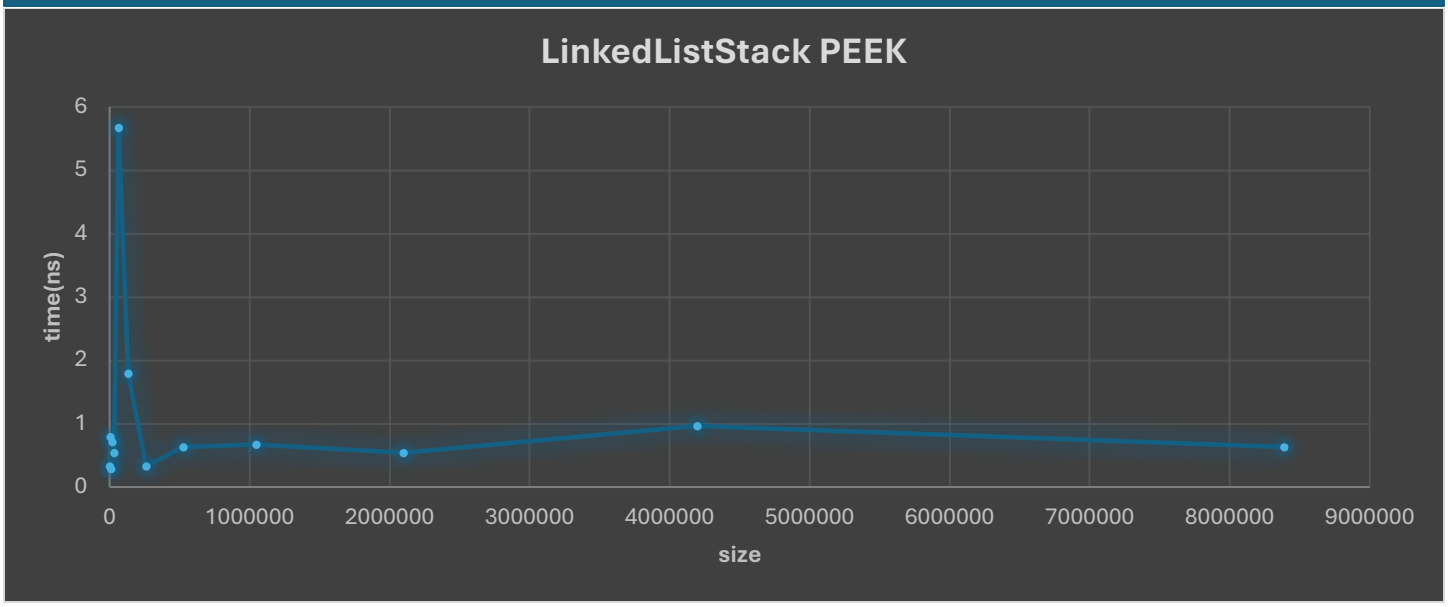




- Compare the running time of the pop method. What is the growth rate of the method's running time for each stack class, and why?

    - Both array stack and linked list stack's pop method running time is O(1).
    - For array stack, removing an element is simply decrementing the top pointer
    - For a linked list stack, adding an element to the top of the stack involves reassigning the head pointer to the next node.





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

  - Both array stack and linked list stack's peek method running time is O(1).
  - For array stack, accessing the top element involves reading the value at the top index without modifying the array.
  - For linked list stack, accessing the top element is simply returning the value stored in the head node.

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

  - It appears that array stack push and pop methods are slightly more efficient just because they seem to level off at fewer nanoseconds that linked list stack. The opposite is true though for the peek method