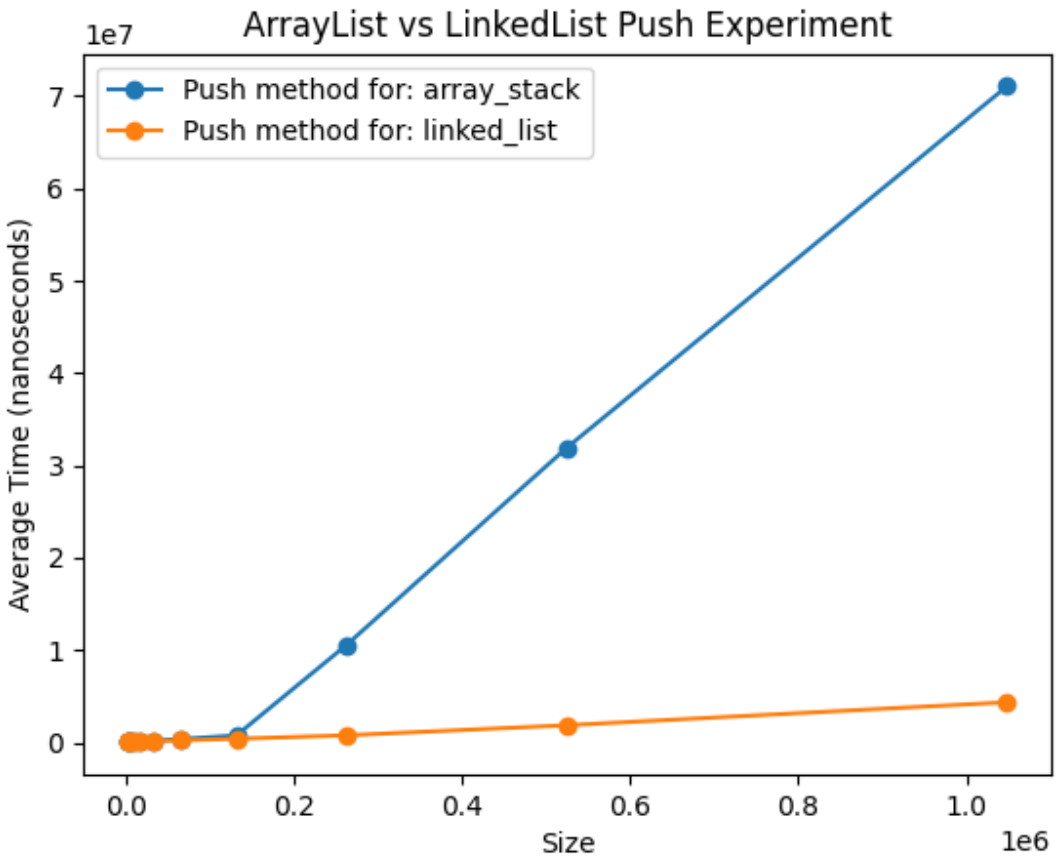
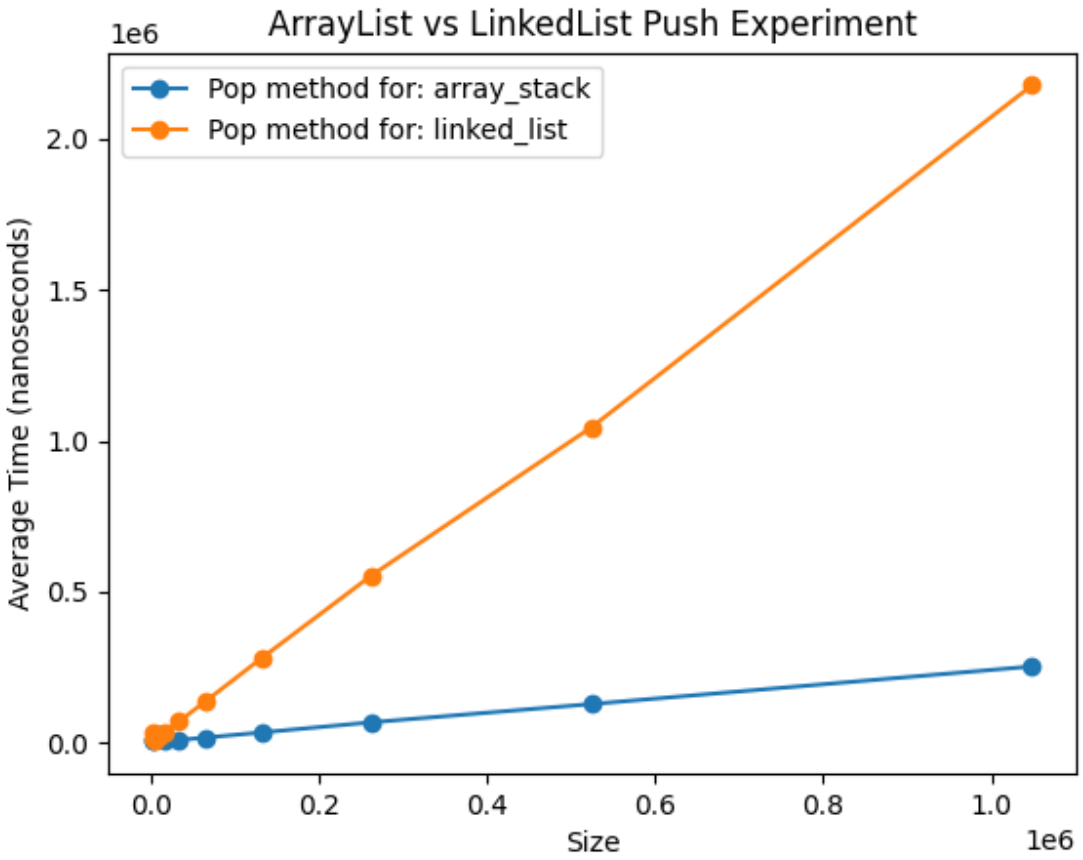
- Compare the running time of the push method. What is the growth rate of the method's running time for each stack class, and why?



For the push method, **LinkedListStack** class has a **consistent O(1)** time complexity which is independent of size since elements are added at the beginning of the linked list, there is no need to resize an array. On the other hand, **ArrayStack** push method has a **constant O(n)** time where n is the current size of the array. The graph shows visually the difference between a O(n) and O(1) behavior from the two different methods from the different classes.

- 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?



In this graph, the two lines are very similar and close to each other. There is a minimal difference in the total run time of both methods. times for both methods. Where **both methods** exhibit a **O(1)** time complexity. In the data collected, the difference of the ArrayStack and LinkedListStack in nanoseconds to seconds is 0.00217776543 of a second meanwhile ArrayStack does the same in 0.00025212415 of a second.

ArrayStack data:

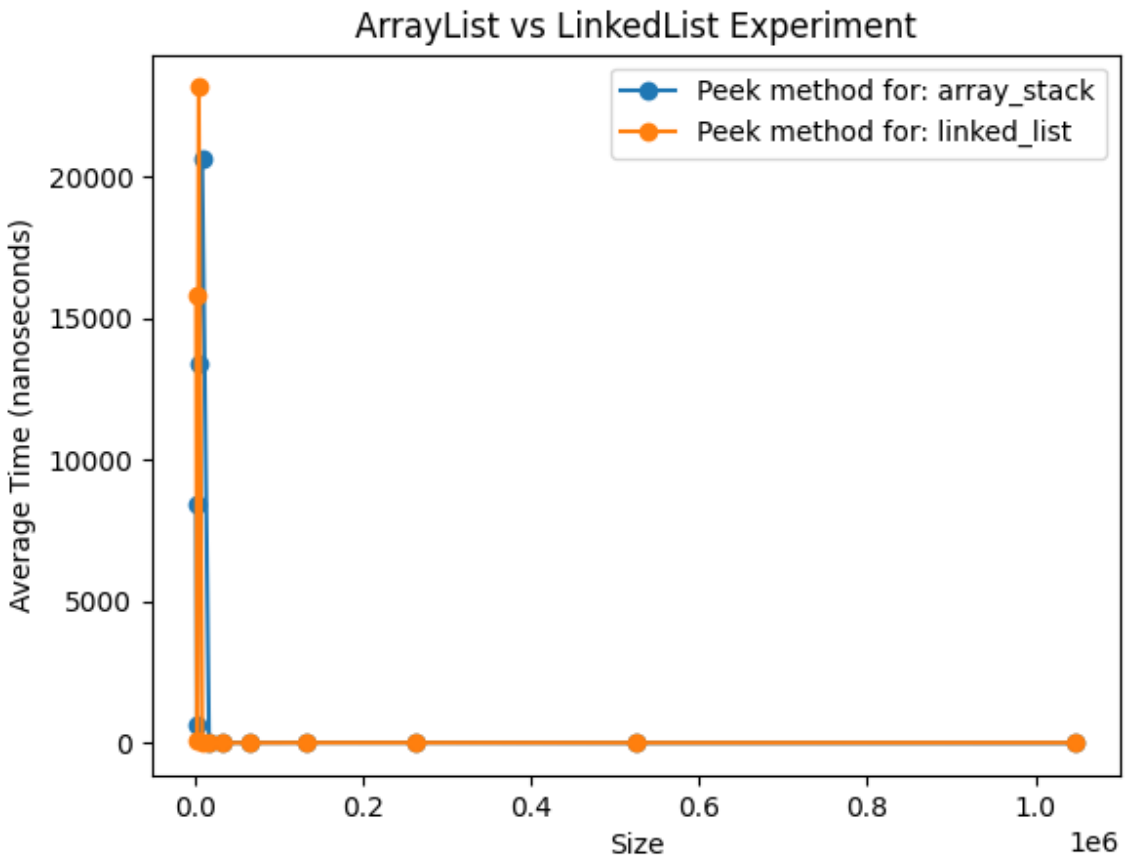
```
1024    10372.12
2048    8492.88
4096    22793.78
8192    28249.13
16384   4661.68
32768   8423.33
65536   16760.03
131072  33311.15
262144  67066.27
524288  127527.56
1048576 252124.15
```

LinkedList data:

```
1024    15862.81
```

```
2048    29956.29
4096    8561.26
8192    18172.51
16384   35016.21
32768   69570.88
65536   138620.83
131072  278996.6
262144  553498.33
524288  1045067.88
1048576 2177765.43
```

- 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?



In this graph, **both methods** have a **O(1)** time complexity. The peek operation access the top elements of the stack and both data structures are designed to allow constant-time access to the top element. In the **ArrayStack** accessing an element in the array by index is a constant-time operation meanwhile using the **LinkedListStack** access the first element of the linked list(the head) is also a constant-time operation.

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

Based on the timing experiments, the LinkedListStack is more efficient than using ArrayStack. The LinkedListStack has a consistent O(1) time complexity meanwhile the ArrayStack can possible have O(n) for the push method meaning that the created class run time does not depend at all in the size of the stack size while ArrayStack can have a max runtime of O(n).