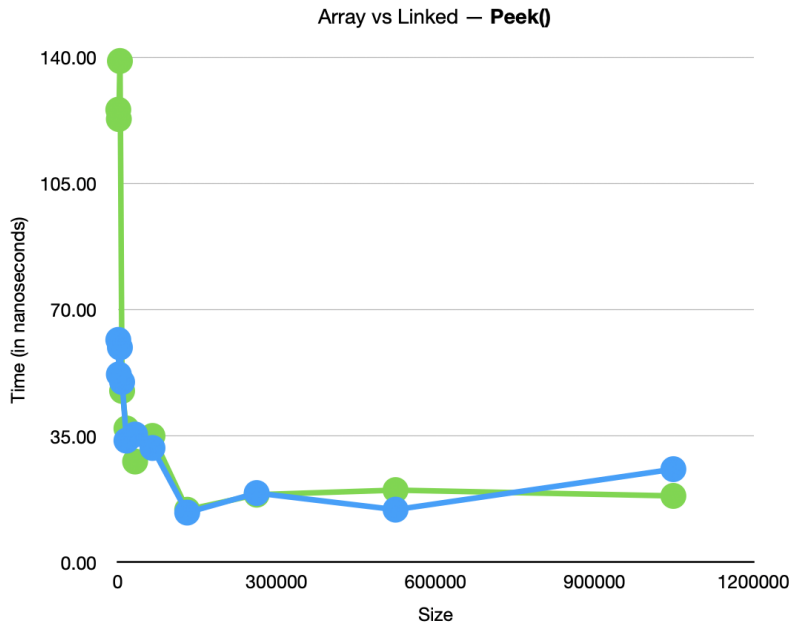
Assignment 5 Analysis
CS 6012
Lindsay Haslam

Peek()

For both ArrayStack and LinkedList, their growth rate resembles O(1) the most. There is a little bit of fluctuation, and a dramatic surge in the beginning, but they seem to level out as it increases in size. This isn't completely surprising to me, when considering the methodology behind Peek(), where the method is simply returning what is at the top of the stack.

ArrayStack	
1024	61.64
2048	52.07
4096	59.54
8192	49.96
16384	33.72
32768	35.42
65536	31.68
131072	13.77
262144	19.18
524288	14.58
1048576	25.8

LinkedList	
1024	125.46
2048	122.87
4096	138.96
8192	47.51
16384	37.09
32768	27.92
65536	35.02
131072	14.55
262144	18.73
524288	20.0
1048576	18.39

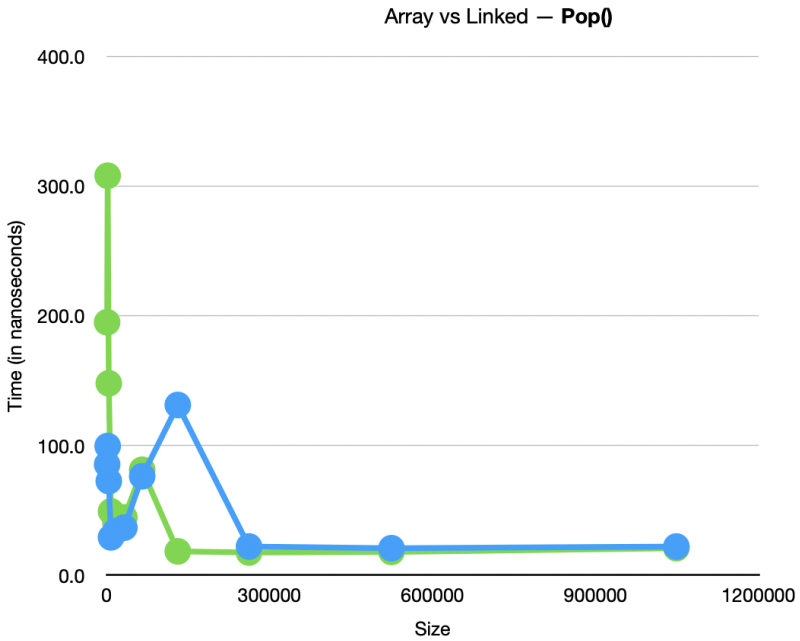


Pop()

Similar to Peek(), the growth rate of this method resembles closest to O(1). There isn't as much fluctuation as time passes, but there was a surge both at the beginning and somewhere around 131,072. Other than that, it remains pretty consistent throughout the rest of the iteration. This also isn't too surprising (I think the surges are the most surprising) as the code for pop() simply returns something.

Array	
1024	85.4
2048	99.59
4096	72.54
8192	29.15
16384	33.76
32768	36.67
65536	76.24
131072	131.24
262144	22.1
524288	20.81
1048576	22.08

Linked	
1024	195.01
2048	308.0
4096	147.93
8192	49.19
16384	40.43
32768	44.55
65536	81.26
131072	18.38
262144	17.47
524288	17.91
1048576	20.81



Push()

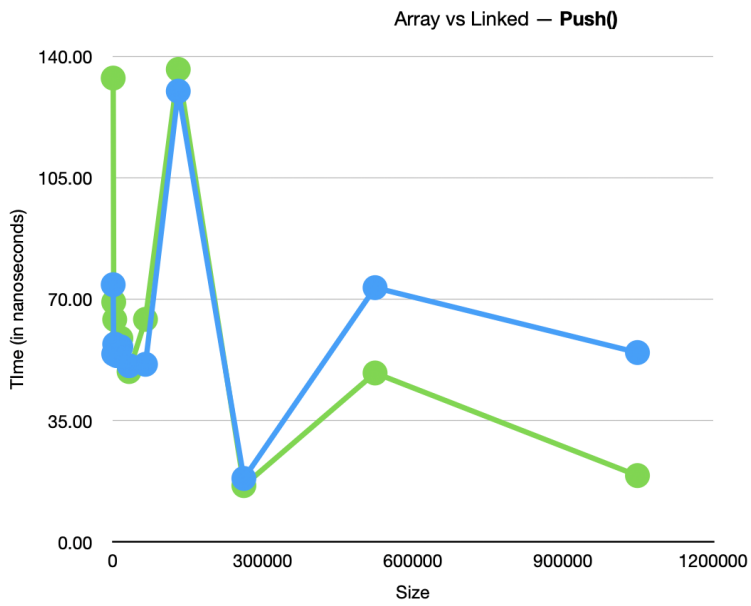
Push has much more variation in its growth rate. There is a large surge at the beginning, particularly with LinkedStack. Another surge happens for both at 131072, and a less dramatic one at around 524288. The push method does have a for loop in it, and is doing much more work and iteration than peek() and pop().

Array

1024	74.19
2048	54.23
4096	57.05
8192	53.75
16384	56.26
32768	50.85
65536	51.24
131072	129.99
262144	18.33
524288	73.34
1048576	54.63

Linked

1024	133.75
2048	69.19
4096	64.15
8192	56.27
16384	58.75
32768	49.16
65536	64.2
131072	136.26
262144	16.27
524288	48.74
1048576	19.16



As for which stack class is best for web browser, I think that ArrayStack would be best. Although it does vary quite a bit between the two, of which is most efficient, I think that Array has a more steady and simple growth rate. Especially considering that most web browsers don't have thousands upon thousands of URLs in their history, it would be best to go with Array since it's been the most efficient with smaller sizes.