Assignment 5 analysis doc

Aiden Pratt

Partner: Melanie Prettyman

- What is your programming partner's name and which of you submitted the program files to Gradescope? Briefly discuss the pair programming experience, including:

  Melanie Prettyman

- How well did you apply the techniques of pair programming (as explained on this page) to this assignment?

  We applied the techniques well by alternating the navigator and the driver for each of the different classes we made. This allowed us to catch each others mistakes and discuss the next steps for moving forward.

- What percentage of the program code did you write using these techniques?

  We contributed equally to the program. I wrote about 50%.

- About how many hours did you spend completing the programming and testing portion of this assignment? Evaluate your programming partner.

  We spent about 15 hours total on this project and had great success working to problem solve together during the pair programming process.

- Do you plan to work with this person again?

  yes

- 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?

  Figure A

The growth rate for each stack class for the push method was linear. Both classes grow very similarly over the 11 input sizes provided in the tests. The time it took for each class was near identical, but based on my data in my linear graph, arraystack may be slightly better for larger arrays. These growth rates are linear because when there are more elements to add to an array, it will take longer to push them into the array. This is an example of O(N) growth complexity. While the process of pushing elements into the array is fairly constant, it takes longer with more elements.

- 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?

  figure B

The growth rate for each class's pop method is fairly constant due to the nature of this method. Pop removes the firstmost element from an array. It does not need to loop through the array, so the size will not impact the time it takes to perform the pop method. For this reason, the pop method is O(1) growth complexity. In my graph, the arraystack class consistently performed slightly faster than the linkedlist class.

- 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?

  figure C

The run times for the peek method in my experiment suggest that after the initial peek, the arraystack class is slightly quicker than the linkedlist class at peeking the element. The growth rate for the peek method is also O(1) because it only needs to look at the first element without needing to loop through the size of the array.

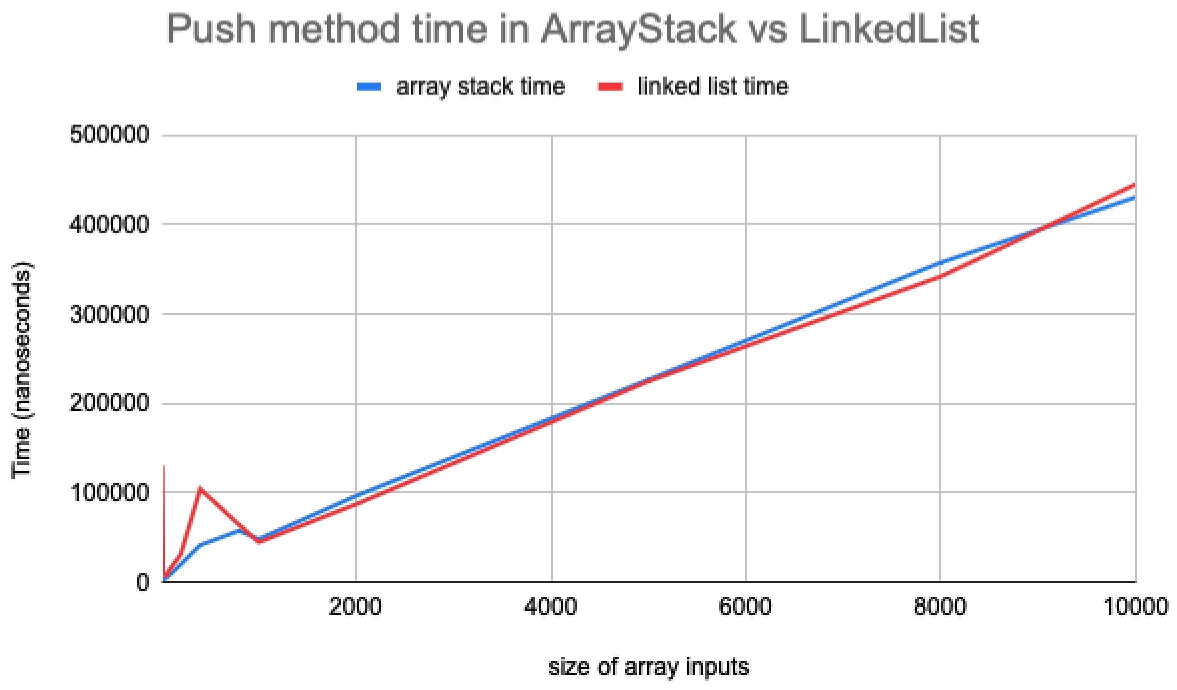
- Based on your timing experiments, which stack class do you think is more efficient for using in your WebBrowser application? Why?

Based on my timing experiments, I suspect that arraystack might be more efficient for the web browser application. arraystacks also allow for random access by the index, which can be advantageous when looking in a webbrowsers history. LinkedList may need to traverse through nodes which can result in slower run times. Arraystacks also have the constant time complexity of peeking the most recent element in this list, which is helpful for the back and forward buttons in a web browser.

Figures

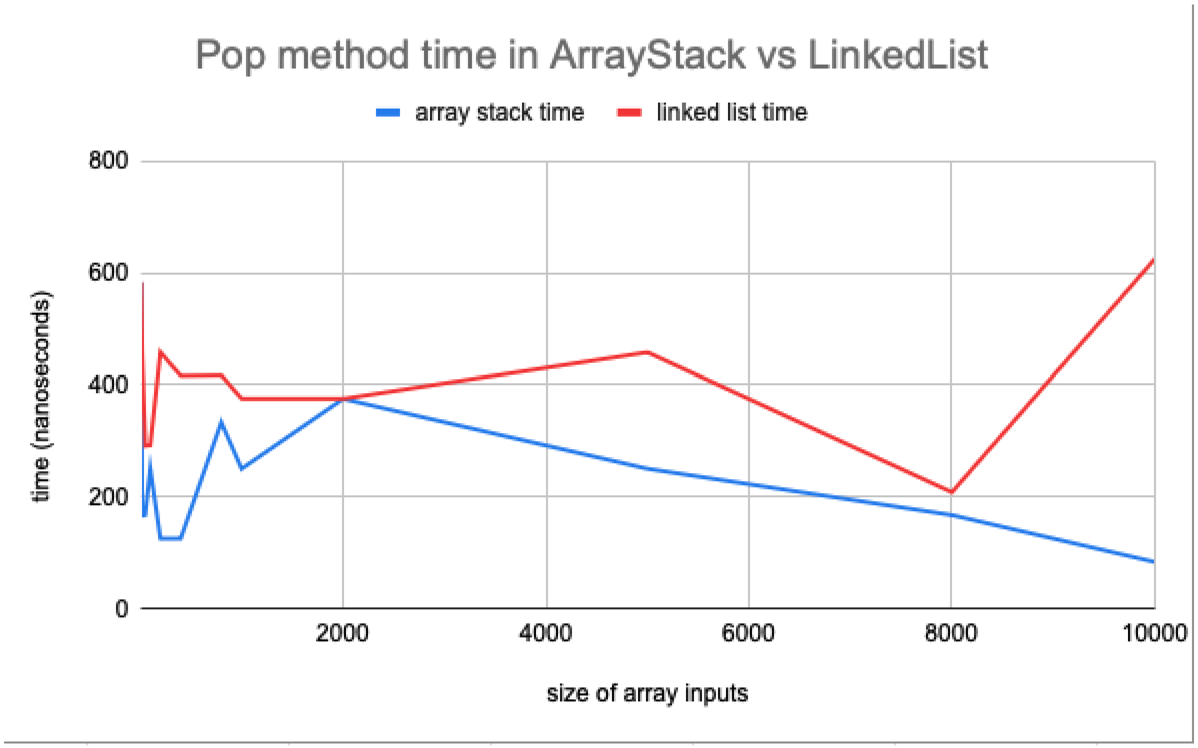
A:

size	array stack time	linked list time
10	4292	130292
20	2375	8667
50	4334	7333
100	9459	16167
200	19667	30708
400	41291	104250
800	57375	63917
1000	48042	44584
2000	96541	87166
5000	226709	225042
8000	357792	342166
10000	430708	445292


Push method time in ArrayStack vs LinkedList
array stack time
linked list time
Time (nanoseconds)
500000
400000
300000
200000
100000
0
2000
4000
6000
8000
10000
size of array inputs

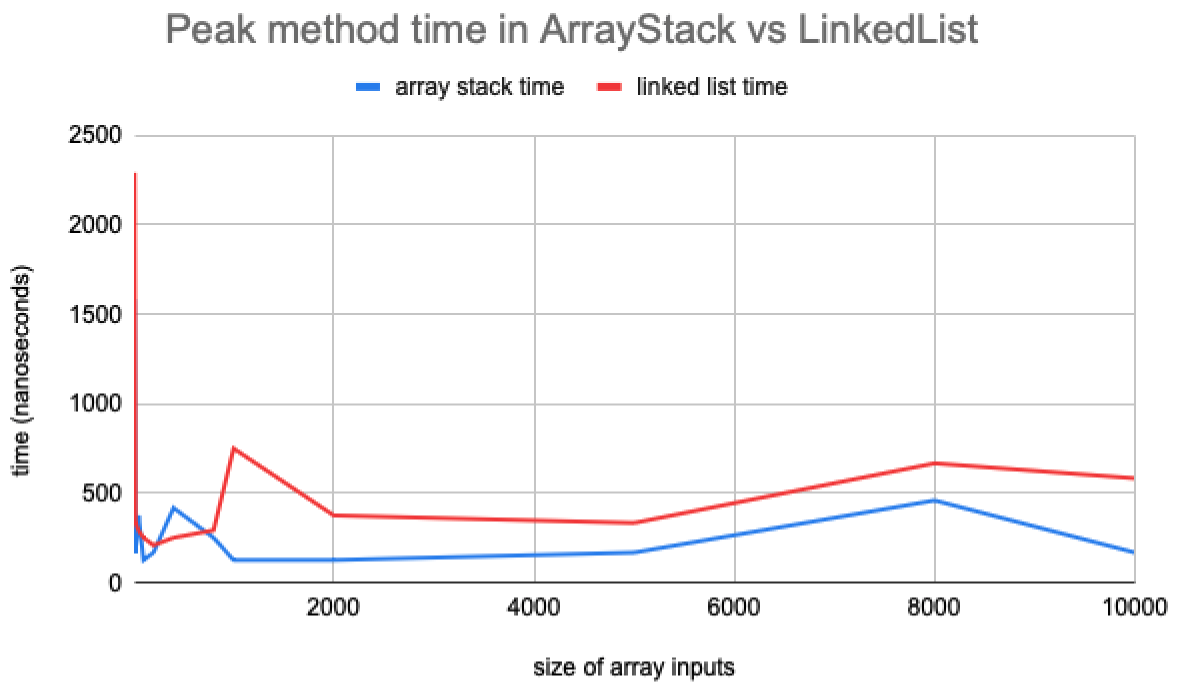
B:

size	array stack time	linked list time
10	583	583
20	166	458
50	167	291
100	250	292
200	125	458
400	125	416
800	333	417
1000	250	375
2000	375	375
5000	250	458
8000	167	208
10000	83	625



C:

size	array stack time	linked list time
10	1583	2292
20	166	333
50	375	291
100	125	250
200	167	208
400	417	250
800	250	292
1000	125	750
2000	125	375
5000	167	333
8000	459	667
10000	167	583


Peak method time in ArrayStack vs LinkedList
array stack time
linked list time
time (nanoseconds)
size of array inputs