A5 Analysis Document
Samantha Pope

PAIR PROGRAMMING QUESTIONS:

* What is your programming partner's name and which of you submitted the program files to Gradescope?
I coded with Elisabeth Frischknecht. I submitted our final project to gradescope and added her to it as a partner on our final submission. She also has the final project on her computer so she can run analyses over the weekend. We both will commit our projects to github.

Briefly discuss the pair programming experience, including:
* How well did you apply the techniques of pair programming (as explained on this page) to this assignment?
When we got stuck on things we used one person's laptop and the other would use a notebook or the lecture slides to guide the other person through debugging or writing the trickier functions. This worked quite well and followed the recommendations for pair programming. When we weren't stuck, we divided and conquered the functions or writing tests to get through the assignment more quickly.

* What percentage of the program code did you write using these techniques?
I would say we wrote 70% of the code using the navigation and driver technique, and then the other 30% we both contributed to but did simultaneously.

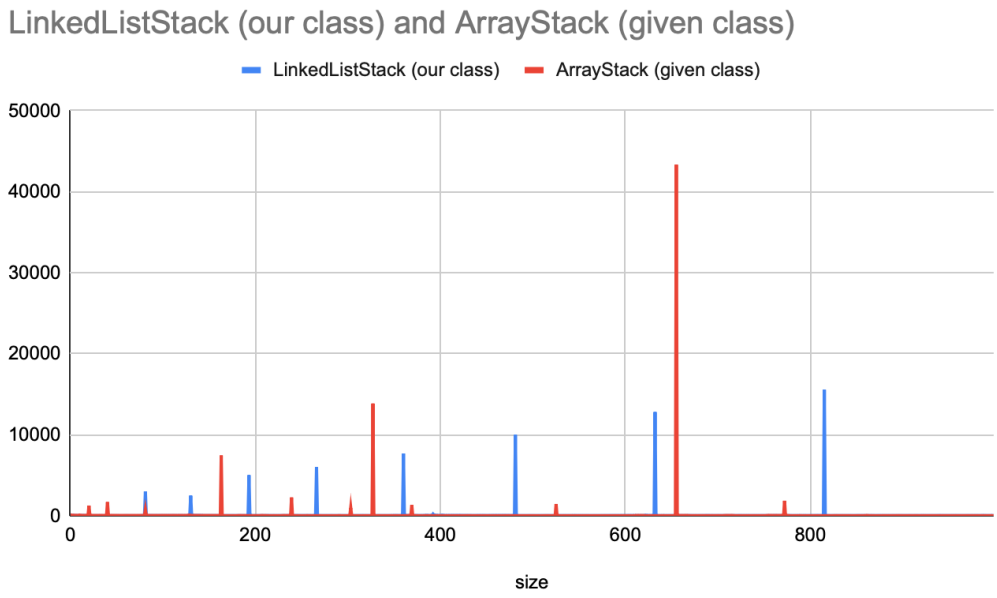
* About how many hours did you spend completing the programming and testing portion of this assignment?
It took us 8 hours on Friday to complete the coding for this assignment. That did not include any of the time analysis, which we are doing separately over the weekend.

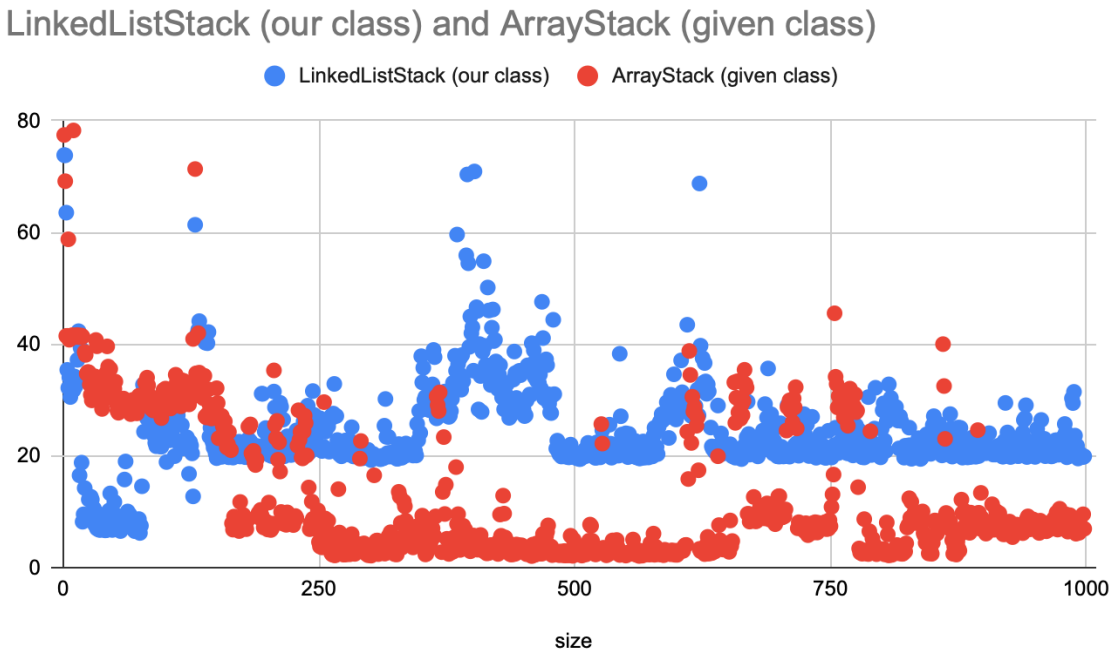
Evaluate your programming partner.
* Do you plan to work with this person again?
I love working with ziz! She is very kind and good at coding/problem solving. I also think our brains work through problems differently so she will find errors I won't think of and vice versa. It makes finding edge cases and simple mistakes much easier. I would love to work with her again!

TIMING ANALYSIS QUESTIONS:

*** 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 growth rate for each class is about O(1). I tracked the time it took to push 1000 items into a stack, and the time to add any item was about the same. I believe the peaks in my first graph are just noise. I zoomed in on the second graph to get rid of the noise. Both of the stacks had a time to add that was independent of size (not affected by N). This is because the stack only adds in one spot, so it does not need to iterate through the list of objects. In our method a new node is created, the header is set to the new node and the size is increased. The ArrayStack given to us performed the push element at a smaller time, on average, than our class. This is shown by the second graph that zooms in on the data points, and also the calculated averages in the table.

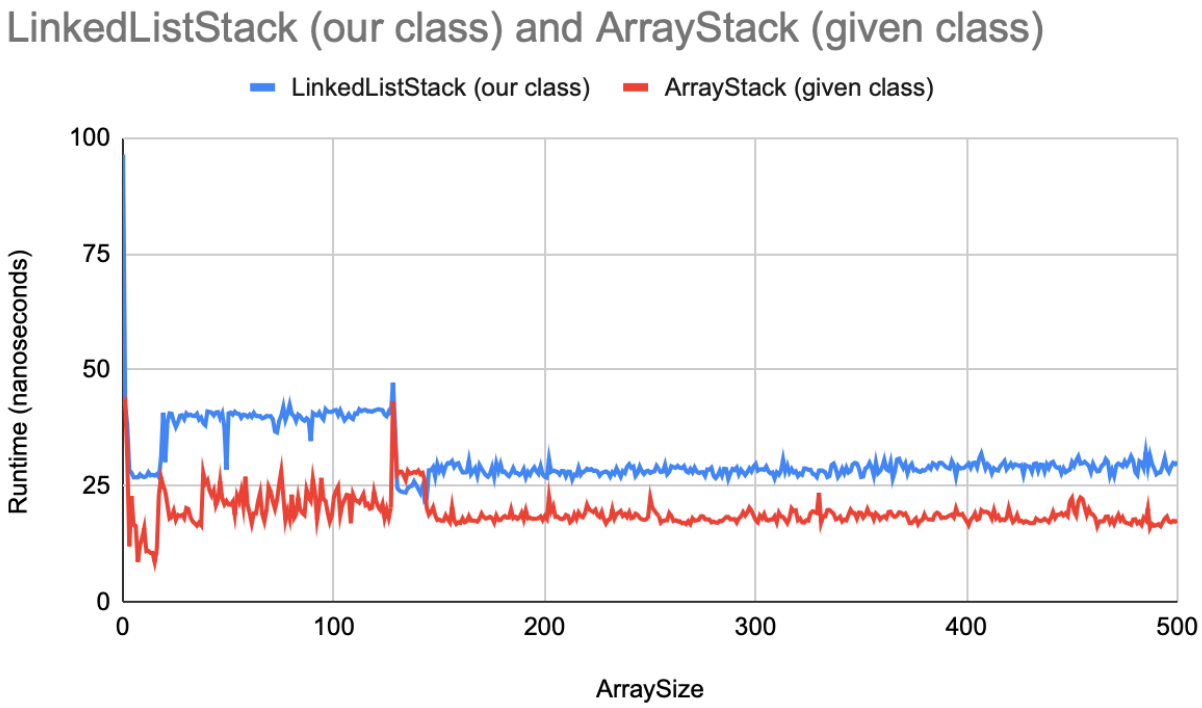




Push Method	Average	Median
LinkedListStack (our class)	111.8592051	27.385
ArrayStack (given class)	65.36356244	7.6139

*** 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?**

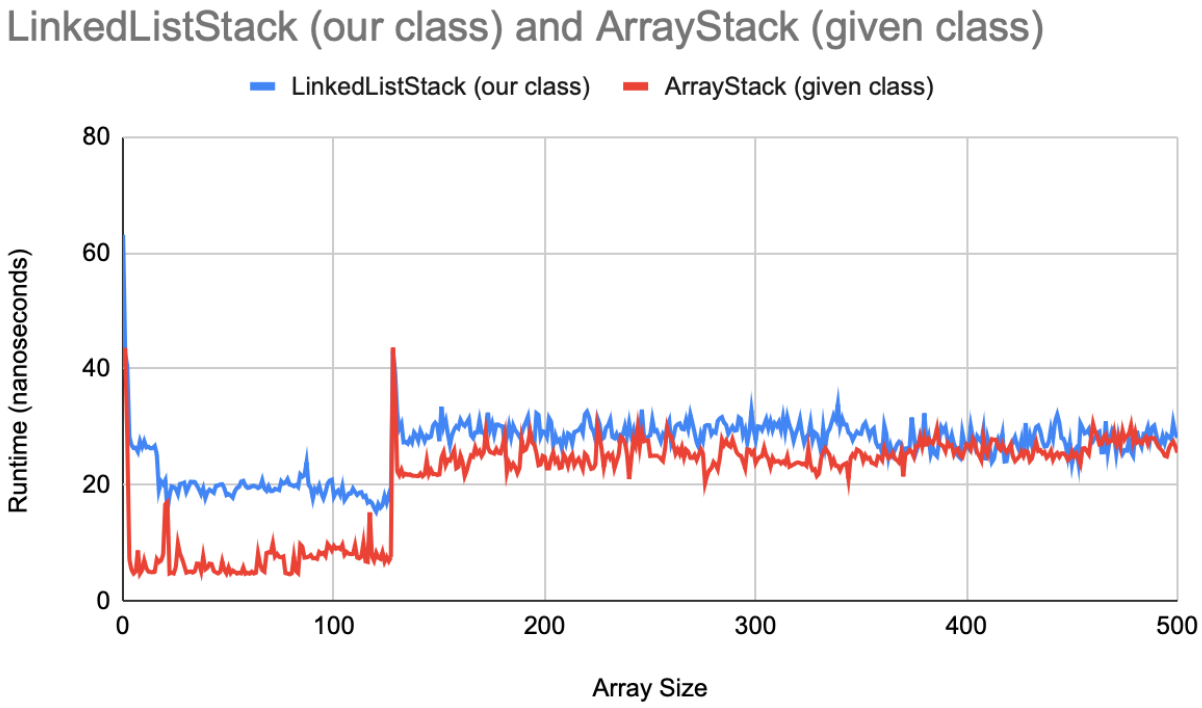
The growth rate for each class was again O(1), meaning that it was independent of the array size. There is nothing in the method calls that makes us need to access each member of the array. The pop method only needs to access the most recently pushed element of the array, making it O(1) behavior. The ArrayStack class performed this method faster than the Linked list stack. This difference became less significant at size 130. I am not certain why this behavior is shown. It could possibly be noise in our experiment. There are also fewer lines of code in this method, which is an indication that there will be less noise as there are less edge cases to be aware of. This is also why the means and medians of each class are more closely related to each other, they do not have these large “noisy” data points. Our method call creates and returns a temporary variable, which could lead to a slightly larger runtime than the given class which just returns something the last value and decrements the size.



Pop Method	Average	Median
LinkedListStack (our class)	31.13711996	28.8606
ArrayStack (given class)	19.21267186	18.3729

*** 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?**

The growth rate for each class was again O(1), meaning that it was independent of the array size. There is nothing in the method calls that makes us need to access each member of the array. The peek method does not need to change anything in the list/array, it needs to just show what the most “recent” value is. The ArrayStack class performed this method only slightly faster than the Linked list stack. This difference became less significant at size 130, like in the pop method. I am not certain why this behavior is shown. Since peek is not changing the data of the array/list at all, I expected this to be the most stable. It has average and mean values that are extremely close to each other and appears more stable than the other two methods. It also has a similar runtime between classes. Both classes just call the most “recent” data pushed into the stack and return its value.



Peek method	Average	Median
LinkedListStack (our class)	26.67442275	27.9846
ArrayStack (given class)	20.81375968	24.5437

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

I think the ArrayStack class will be more efficient for using in our WebServer class. The two classes both had O(1) runtimes for all of the three methods. However, the arraystack class was noticeably faster on the push and pop methods. I would argue that the peek method, while the arraystack was still faster, had an insignificant difference in runtimes for the two methods after the array size was greater than 150. I think that this difference comes partially from our methods being nested inside of another class. I think that this would add some runtime differences, however I am not sure that it would account for the total difference in the push or pop method.