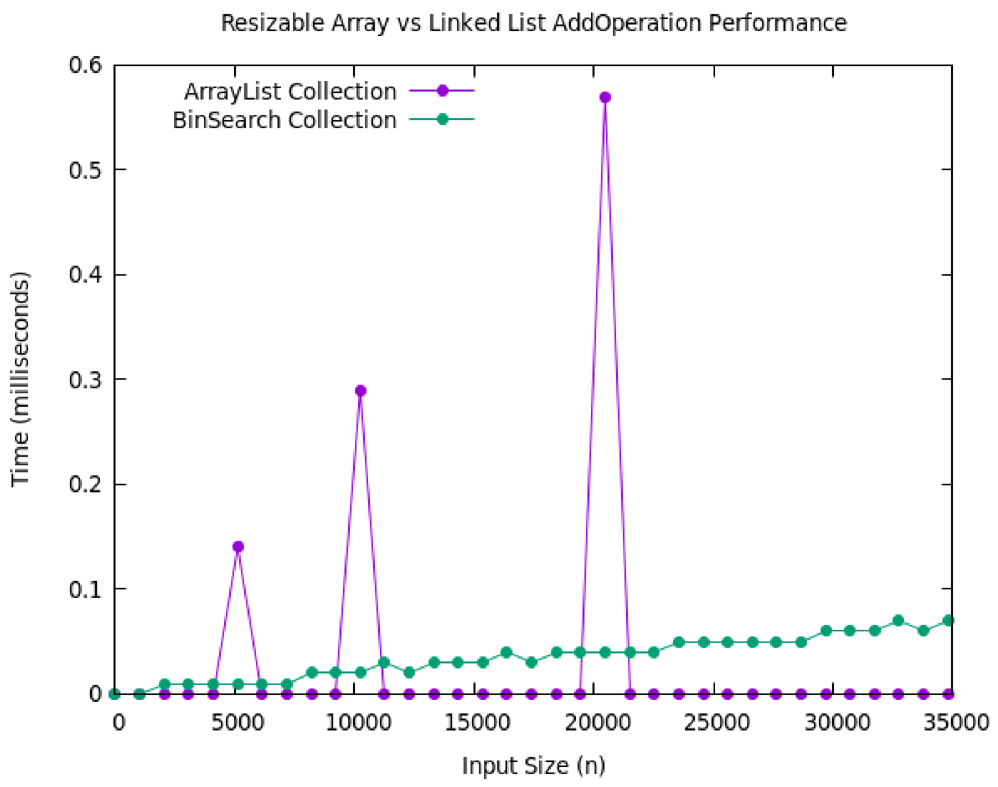
Matthew Moore

Mr. Bowers

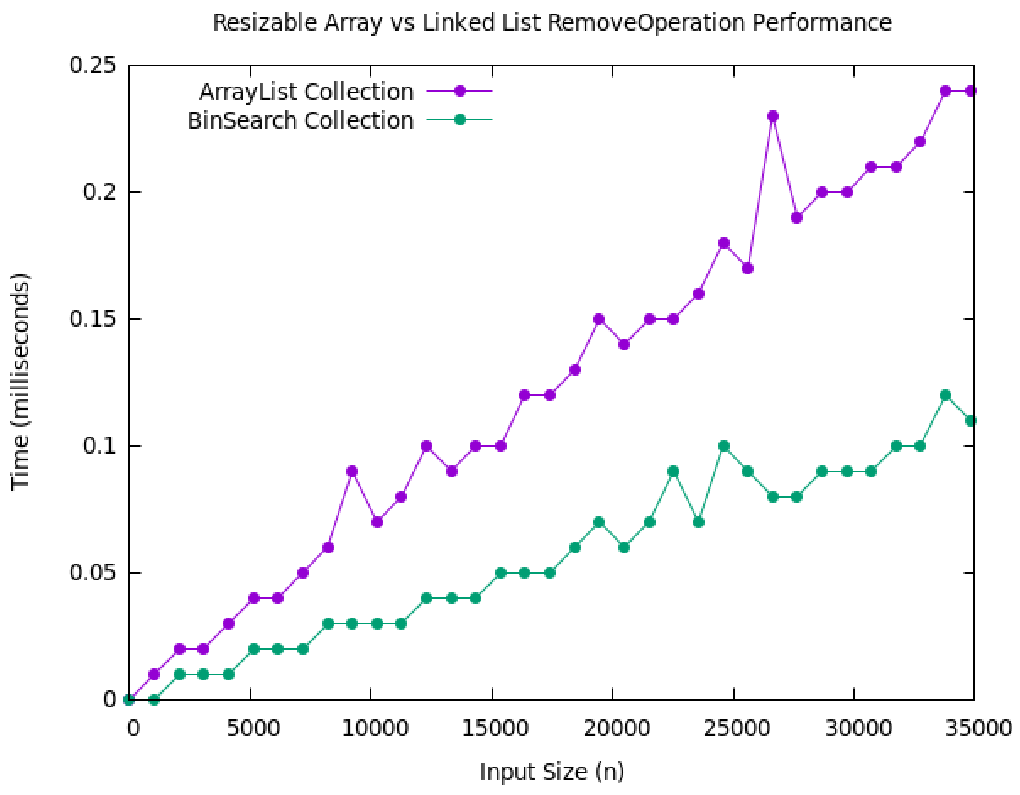
Comp Sci

10/22/20

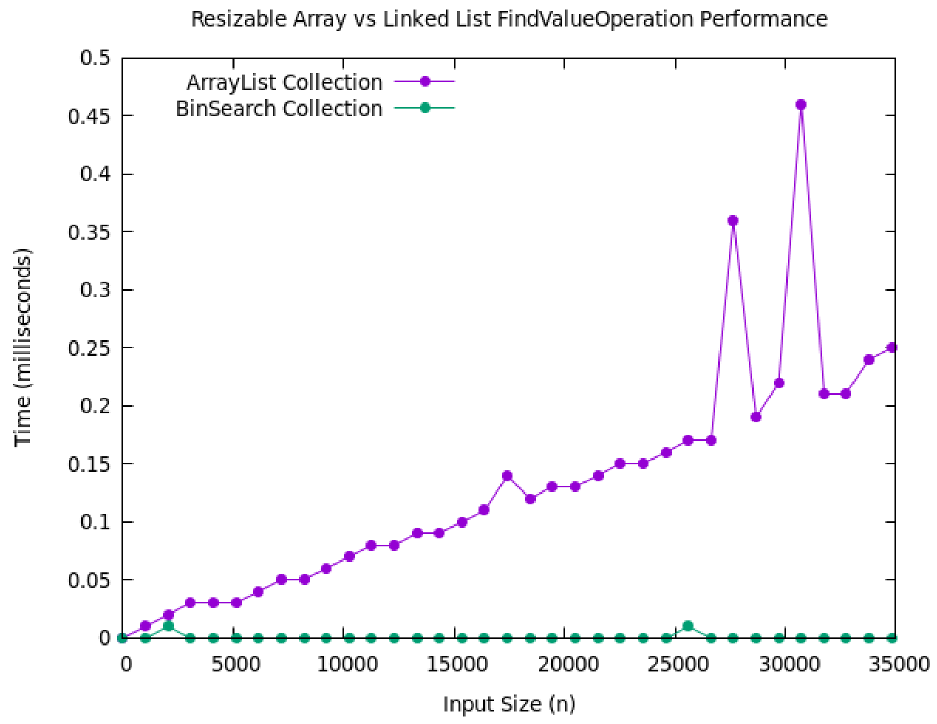
Graph Analysis



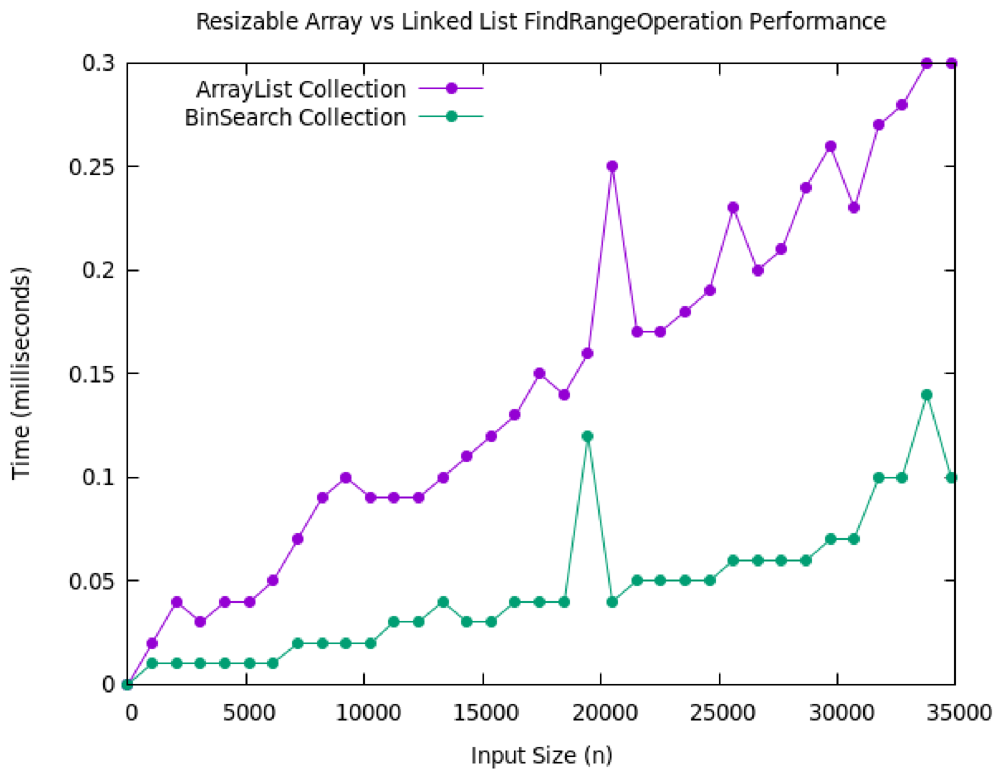
The add operation shows to be very efficient in both the Array List collection and the binary search collection. This is because both of the data structures allow for immediate addition of new elements without any traversal of the list. We can see large bumps in the complexity when resizing of the array occurs. In the case of the array list collection, there is a constant time complexity even with the huge bumps. We know this because after amortization, the graph averages out to have a constant complexity O(1). The binary search in the worst case is also constant O(1) because searching for an element at a clear space like the end or beginning makes it quite simple for the data structure to be successful, but in the worst case the function should be O(log(n)).



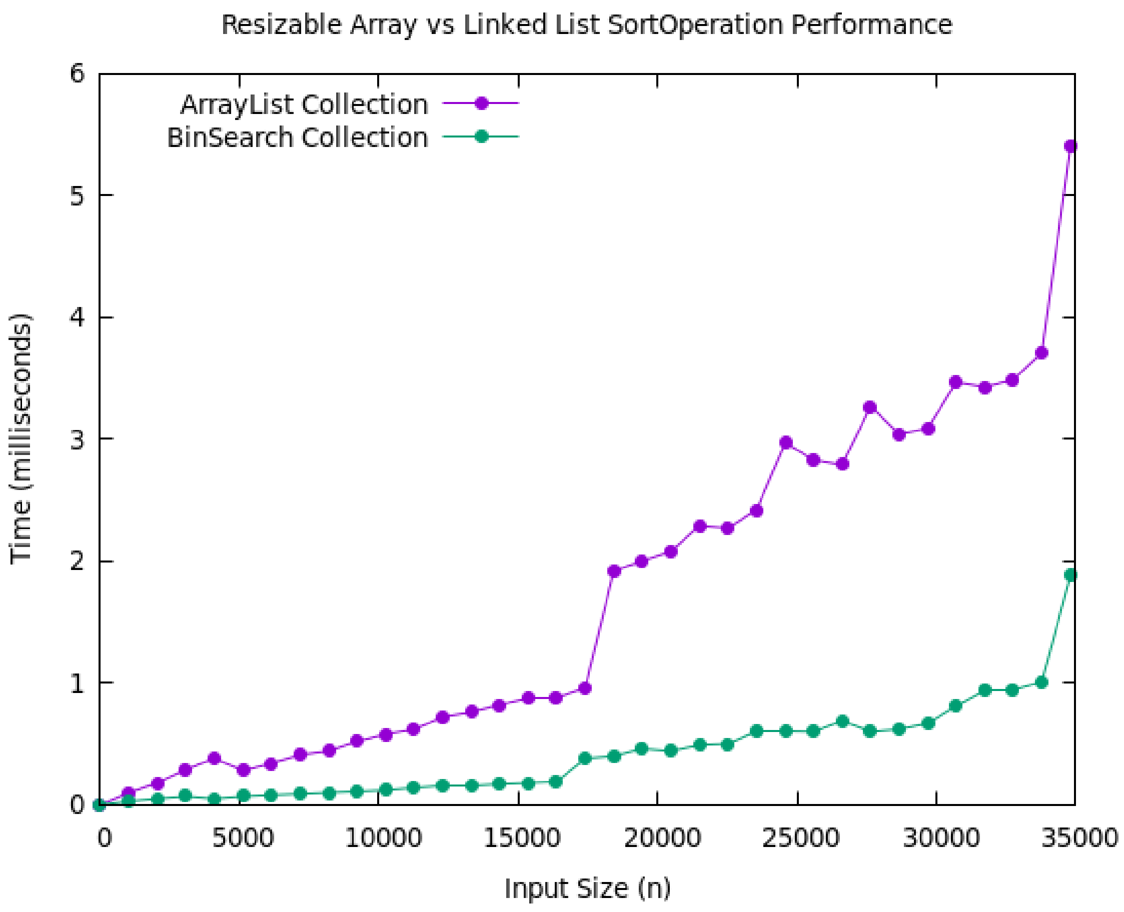
In the case of the remove operation, both of the array list and the binary search have some struggles. The array list has a lot of problems because it requires a searching of the array until the correct element is found to remove. So, the worst case here as well as the actual scenario in this example is O(n). The binary search is much faster though due to its quickness at searching for the correct item to remove. So, in this case and the worst case this data structure is O(log(n)).



For the Find Value Operation we notice that there is much better success with the binary search collection over the array list. Finding the value with binary search is extremely efficient with a worst case of O(log n), and in this case the function happens to be O(1). This is likely because the value being searched for is very easy to locate. Meanwhile, the array list collection has a lot of trouble searching linearly for the correct find value, which is O(n) in the worst case and likely n/2 in this specific case because of the average amount of elements needed to be searched in this situation. Some of the strange jumps could be due to other things running on my computer.



The finding range operation also is very advantageous for the Binary search collection over the array list collection due to the binary search being more successfully than the linear search of the array list collection. In both the worst case and this scenario, the time complexity is O(n) due to the inefficiency of the search for the initial key value. In both cases, the time to traverse to the second key value ends up being the same which is why the binary search has some problems. Binary search ends up being O(n log n) in the worst case and in this case it comes out to be around n log n.



Finally the sorting operation has a much more similar complexity between the binary search and the array list collection. The array list collection uses the quick sort method to sort the list which can be O(n^2) in the worst scenario, but in this case it comes out to being close to n log n which is close to the best case. Toward the end of the input size the array list collection jumps to something close to n^2 due to the sheer size of the array. Meanwhile, the binary search collection in the worst case comes out to O(log n), but in this specific situation we can see that the complexity is an average of log n and constant time. This is likely due to some of the added elements being placed in regions that easily fall into this data structure.

OVERALL PROBLEMS:

I think this homework was fairly simple for me to implement in comparison to the other homework, but there was still several problems I ran into. I struggled with the performance tests because my quick sort function was not working exactly how it needed to be. I ran into a seg fault due to a problem with how my quick sort function dealt with array lists that are only one element long. Along with that, I had some struggles with understanding why the add and remove functions acted the way they did with the performance test. It is not obvious where the elements are being added and removed, but once I picked up on the implications, it was easy to see.