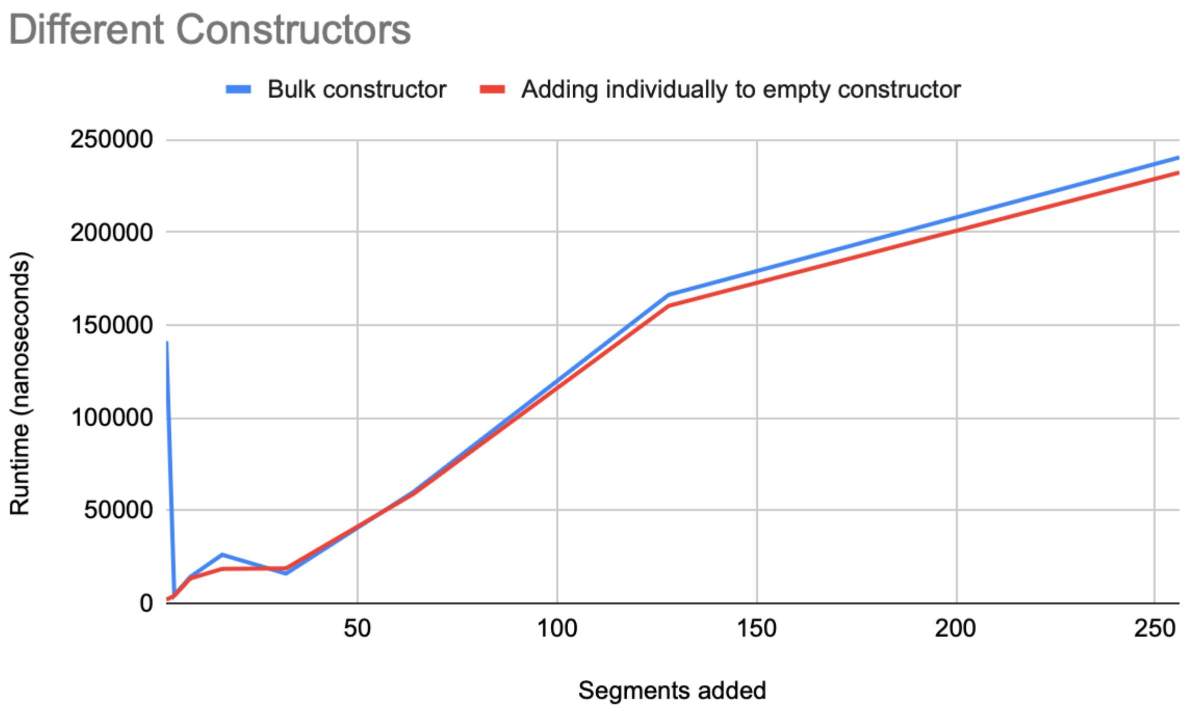
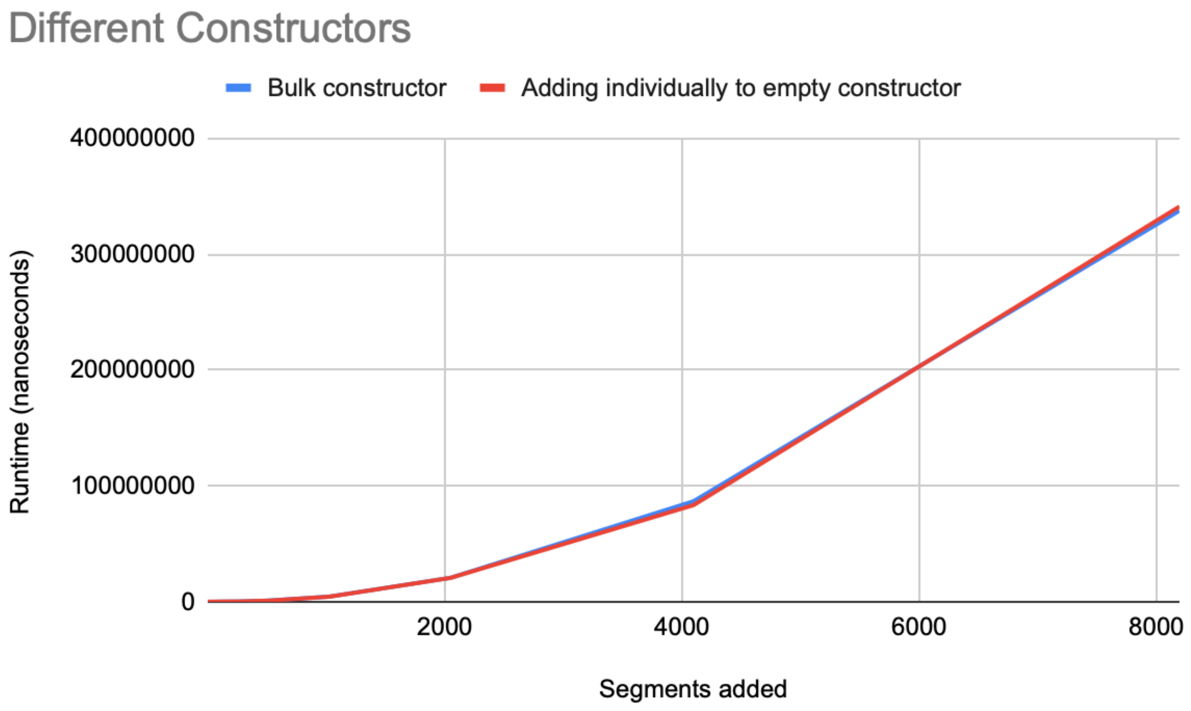
Assignment 9 Analysis
Samantha Pope

DIFFERENT CONSTRUCTORS:

I ran the runtimes for my bulk constructor to add in all of the items at once and then also the runtime for inserting all of those segments to my empty constructor. I did this for sizes up to 8000. I found that adding individually was faster than using the bulk constructor. I did pass the worst case scenario of different vertical lines to these constructors, which is why I think the difference is not as visible. The worst case scenario involves inserting segments at places where they would split the space of the past segments. I expected the bulk construction to have O(N^2) worst case and the other constructor to have a worst case of O(N).

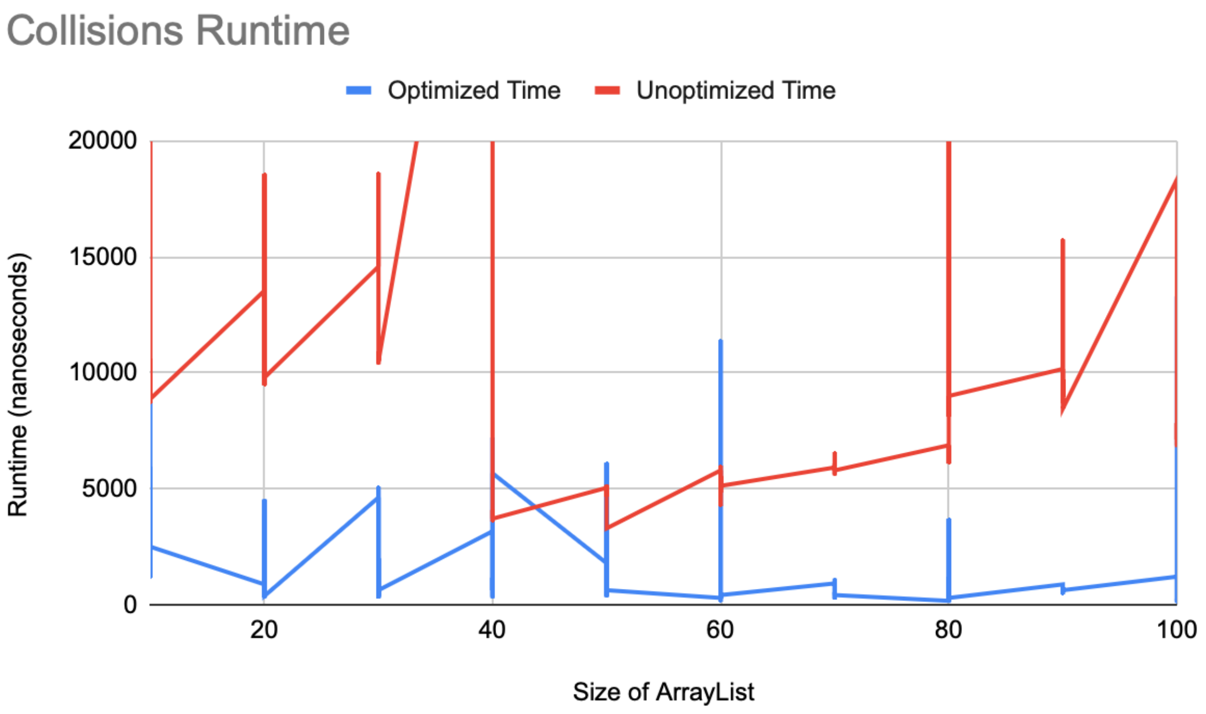
Below is a zoomed in graph of the second graph.



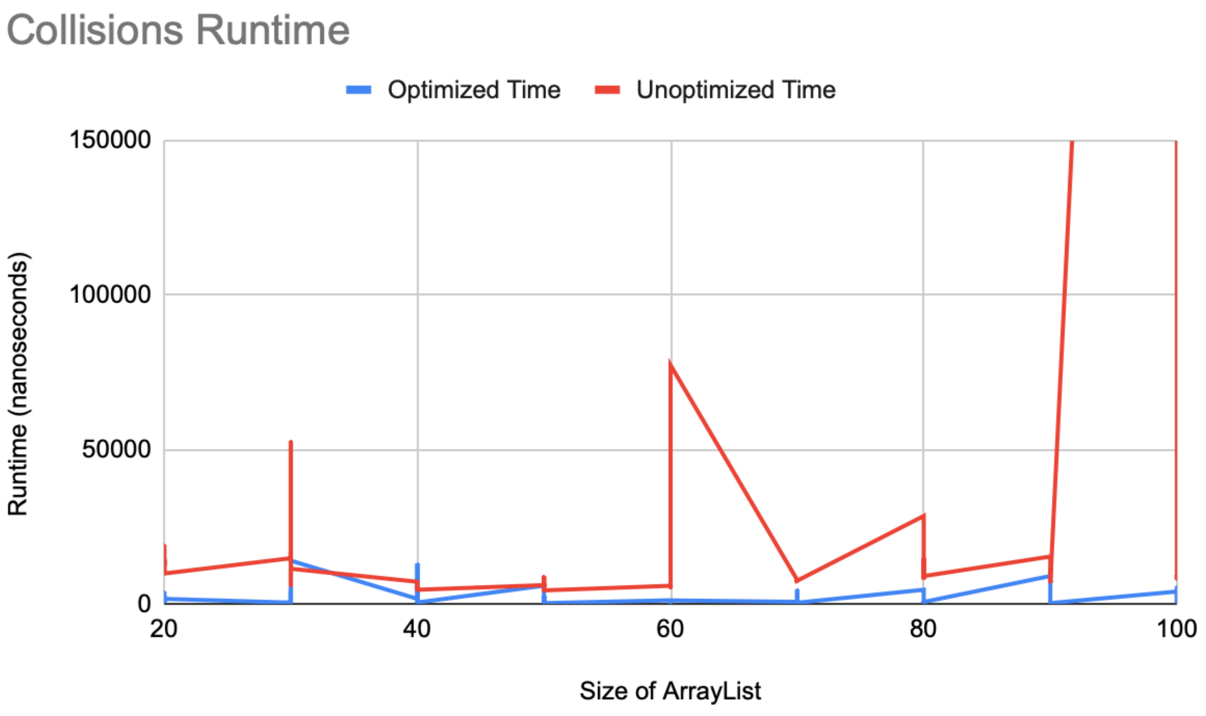


<u>COLLISIONS RUNTIME:</u>

I ran the runtime of both the optimized code that was given to us and the unoptimized which I originally wrote. I seeded two different random values and ran the test file twice and graphed the results. I found for both randomized sets of values, the optimized method ran much faster. When one spiked, they both spiked, but the optimized value was still much lower. I had a hard time looking at the BigO of these functions since they spike and go up and down so frequently. I do think that the unoptimized runtime follows around an O(N^value) to create the large jump at size 90 and it continues to grow. I don't think it was a large enough jump to be O(N^2). I think the optimized time follows a logarithmic growth rate O(Nlog(N)).



Collisions Runtime
Optimized Time
Unoptimized Time
Runtime (nanoseconds)
20000
15000
10000
5000
0
20
40
60
80
100
Size of ArrayList



Collisions Runtime
Optimized Time
Unoptimized Time
Runtime (nanoseconds)
150000
100000
50000
0
20
40
60
80
100
Size of ArrayList