Paul Carmichael
SER 222
Mod 5 Writeup

Hypothesis:

The run time of the bin array and the halved array will have the similar run times of O(n-1) this is because the date in both of these array are sorted. While the run time for the half random array will look closer to the average run time of insertion sort at $O(n^2/4)$ my prediction is it will be closer to $O(n^2/8)$ because half of the date is already sorted being that the first half of the data is 0s.

My prediction for the run times of shellsort will be similar to the insertion sort with the first two formats of the data being already sorted resulting in the same run time. The shellsort runtime will be close to the O(nlogn) since it is already sorted giving the best outcome for the date. While the run time for the half random input will likely be slightly worse than that probably around $O(n^{5/4})$ because it is not the worst case of a reverse sort, so somewhere between O(nlogn) and $O(n^{3/2})$.

Benchmark Results:

N = 65,568	Insertion	Shellsort
Bin	0.482	1.587
Half	0.985	0.714
RanInt	1.789	0.704

Evaluation:

Starting with the insertion sort I was not expecting there to be such a difference in for each of the different data inputs. I was anticipating bin and half being the same, but the b value is 50% larger for the

array that was halved. I would have expected the b values to be similar for both of the first to arrays, and for the difference to really show for the array that was half random integers.

The Shellsort struggled with the array that was half 0s and half 1s. I thought that the run times would have been affected similarly between the array that was halved until it was full. Instead, we see that the half array and the RanInt array both were affected by the size of the input similarly.

Scenario

From what I found in my analysis of the two algorithms, is that shellsort would be the better option. This is because the algorithm is much more consistent in terms of how it is affected by the change in input size. After running the program 20-30 times the b values continuously would come out around .7 while insertion sort would be much more sporadic. This would allow for better predictions when having to increase the size of the input and knowing how long it will take.