Algorithm Analysis

In a separate document (to be submitted as a PDF), write up your hypotheses (3 per algorithm): describe what you think the running time will look like O(n)? O(n^2)? O(n^3)? on each data set, and explain briely why you think that. As long as your ideas make sense, and you do the analysis prior to benchmarking , you will receive full credit on the hypotheses. There will be a separate submission link on Canvas. [5 points]

1. For Insertion Sort:

a. Given the data set which is divided in half with 0's and 1's, insertion sort would look for the minimum each pass through, and since half of the data set is equal to the minimum, then it would only require that half of the data members be swapped. Thus, the worst-case runtime complexity is ~O(n^2/4) for a reverse-sorted data set.

b. Given a data set where half of its members are equal to the minimum, that for each half of the remainder (recursively until 0), the previous data series' value is incremented by 1, and that the larger numbers will repeat ½ the number of times of the next lower number, it is likely that, once the lower numbers are in their final sorted position, that the higher numbers will be sorted as well. Therefore, I would hypothesize that the worst-case runtime complexity of insertion sort for this data set will be O(n^2/3).

c. Given a data set where half of its members are equal to the minimum and the other half are randomly generated integers, I would anticipate a worst-case runtime complexity of ~O(n^2/2).

2. For Shell Sort:

a. Given a data set which is half the minimum and the other half is the minimum + 1, Given that shell sort broadcasts exchanges over greater distance each pass, I would anticipate n/3 exchanges thus a worst-case runtime complexity of ~O(n log n).

b. Given a data set where half of its members are equal to the minimum, that for each half of the remainder (recursively until 0), the previous data series' value is incremented by 1, and that the larger numbers will repeat ½ the number of times of the next lower number; as the member keys grow, they repeat half as many times as the previous key. Given that shell sort casts its insertions over a greater distance with each pass, once the min values were sorted, it is likely that the higher values are nearly sorted. In other words, the number of exchanges could be expressed as: $\sum_{i=0}^{n-1} \left(\frac{1}{2}\right)^{k} * n$ for k in range (n // 2). This equivalates to O($n^{3/2}$).

c. Given a data set in which half of its members are equal to the minimum and the other half are randomly generated integers, this data sets worst-case runtime complexity is bounded by O($n^{3/2}$).