Hypothesis

If Merge sort and Insertion sort are combined into a Hybrid sort algorithm where the Hybrid sort uses Insertion sort for lists less than around 127 and Merge sort for lists greater than around 127, then the Hybrid sort algorithm will have a better overall performance than both the Merge sort and Insertion sort algorithms. I believe this as in the previous problem I got a crossover between Merge sort and Insertion sort at n = 127, so combining the two and switching them at n of around 127 logically should produce a better overall performance.

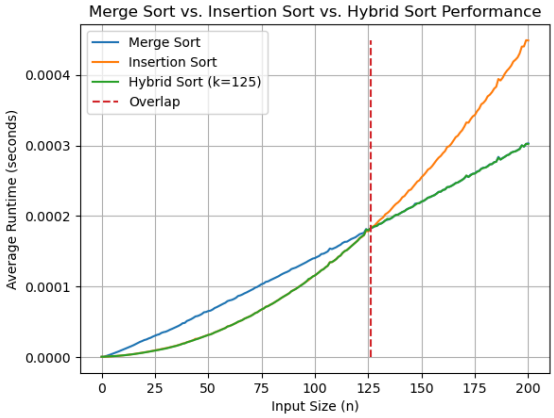
Methods

Using Python I first utilized the Merge sort and Insertion sort algorithms created by GeeksforGeeks and set the random seed to 42. I created an empty list to contain Merge sort average runtimes, an empty list to contain Insertion sort average runtimes, five empty lists to contain Hybrid sort runtimes for different k values 75, 100, 125, 150, and 175, and one final list containing each n integer from 0 to 200. For each n in the list, I created a random list of n values containing random numbers from 0 to 100,000, and copied the list so I would have seven identical lists, one for each sorting algorithm. I would then run each sorting algorithm on the data, measuring the time it took to complete each algorithm using timeit. I repeated this process 2,000 times for each n, storing the average runtime for each algorithm for each n in their corresponding lists. I then printed the total combined runtimes for each Hybrid sort of different k to compare which Hybrid sort performed the best. Finally, I created a graph showcasing the average runtimes for Merge sort, Insertion sort, and the best Hybrid sort based on the input size, with an extra line showcasing for what n Insertion sort starts taking longer than Merge Sort.

Source Code: https://github.com/Degancol/Colby-Degan-HW3-Q6-Submission

Results

By comparing the combined average times of Hybrid sort for k = 75, 100, 125, 150, and 175, I obtained total times of 0.135, 0.131, 0.129, 0.131, and 0.138 milliseconds, respectively. This means that Hybrid sort for k = 125 performed the best compared to the others.



As seen in the graph above, Hybrid Sort for k = 125 follows the same path as Insertion sort until it reaches the overlap (point where Merge sort starts performing better than Insertion sort), where it then follows the same path as Merge Sort.

Discussion

Since Merge sort performs best when k is located at the overlap point, it stands to reason that k is the same as the crossover point from Question 5. This follows what I expected would happen, as it makes sense that ensuring the smallest runtimes regardless of sorting method would produce the best performance.

Conclusions

Under the conditions tested, Hybrid sort produces a faster algorithm than both Merge sort and Insertion sort provided that k equals the crossover point between Merge sort and Insertion sort.

Sources

GeeksforGeeks. “Merge Sort.” GeeksforGeeks, www.geeksforgeeks.org/dsa/merge-sort/.

GeeksforGeeks. “Insertion Sort Algorithm.” GeeksforGeeks, www.geeksforgeeks.org/dsa/insertion-sort-algorithm/.