Mod 10 Prompt

Q1. Some key factors to differentiate the various sorts are: the use of temporary space or not(in-place sorting), basic methods for sorting(mergin, selection or insertion), split the main file or not.

Q2. The core idea of merge sort is to divide the collections to multiple ordered smaller sets and recursively merge back to the original size, by comparing the first elements from two smaller sets.

Q3. Since the worst case of the quicksort occurs when the pivot is set with an outlier value like the min/max from the collection, so choosing the median would be better. However, for a random distributed collection, there's no information about the distribution. So randomly choosing a pivot or choosing mid-point of the index is acceptable.

Q4. For shell sorting, the sub-files are created by using a specific gap. The gap, let's say m, creates sub groups of indices such as [0, m, 2m ..], [1, m+1 , 2m+1…] … and the values at those indices are considered as in these sub groups.

Q5. If the datasets has existing orders inside itself, a natural merge utilizes it and avoids over reducing the dataset which the normal merge sort would do.

Q6. A stable sort means after this type of sort is operated on the specific collection, the order of the records having the same rank (values used as the sorting criteria) are not changed.

Q7: Since the majoring the sorting algorithms cost O(nlogn) and O(n^2), once the datasets have a very large number of records, the difference between these complexities becomes more obvious, such as taking very long time to process the sorting. If the record size is very small, possibly the difference will be in milliseconds or microseconds, and it's hard to notice the difference.

Q8: Since ordered data possibly has positive/negative impact on the efficiencies of the sorting algorithms, this characteristic has to be taken into consideration.

Q9: Yes, I've worked extra hours to meet a deadline for my dev project. The key decision I addressed is to prioritize the functionality of the product over the quality, since the delay of my task would cause other consecutive delays on other features or parts of the product. However this is very risky for the future development, so at a later point, I still need to spend time on improvements or fixes.

Q10: Having a wrong algorithm might cause performance issues or logical bugs in the production environment once the dataset gets larger or tends to have some edge cases. In this case, the fix or the improvement would be more difficult than in the development stage.