Sean Connor Discussion Prompt 8 24 July 2018

1. What are some key factors that differentiate the various sorts you have seen?

Aside from the way that the sorts differ from each other in their methods, other ways they differ are in their worst-case, best-case, and average efficiency, and in their performance based on the initial order of the file.

2. What would you describe as the core idea behind merge sort?

Break down the initial file into smallest possible pieces, then rebuild the file in correct order.

3. What are some good methods for choosing a quicksort pivot, and why?

Random selection is a good method, because it minimizes the chances of impacting performance based on initial order of file. For example, if the first element is always chosen, the performance will be poor for nearly sorted or reverse sorted files.

4. How are subfiles created for shellsort?

A gap is chosen, where every i+gap is added to a subfile.

5. In what way does a natural merge exploit existing order in the dataset?

If the data is already ordered, natural merge will not bother to break down those chunks and redundantly reorder – it will simply merge the already-sorted piece. In the worst case of initially reverse-ordered file, there will be no difference. However, in the best case the performance is O(n), better than O(nlogn)

6. Describe in your own words the notion of a stable sort.

If there are items with identical keys in a file to be sorted, then a stable sort will not change the order of those elements during sorting. For items with identical keys, the order of those items in the sorted and unsorted files will be the same.

7. What difference might it make if the records you are sorting are very large or very small?

If very large records are being sorted, O(nlogn) performance is greatly preferred over less efficient $O(n^2)$ algorithms.

Sean Connor Discussion Prompt 8 24 July 2018

8. In considering a sorting problem, how important is it to you that some of the data is likely to be ordered?

In general, not very important. From what I understand, having presorted data nearly always results in faster overall performance. But for true optimization, it could be an important consideration.

9. Have you been in a pre-demo development crunch? What were the key decisions you faced, and how did you address them?

In the professional world – no. In the academic world – yes. As a biomedical engineering undergraduate and mechanical engineering graduate student, I frequently created prototypes for coursework, presentations, showcases, etc. The best solution is to always prepare well in advance. Of course, that was not always the case and sometimes I had to "cut corners" on some areas I would have liked to develop better. For example, I was creating a UI for a device to be used by surgeons during knee arthroplasty, and it certainly was not pretty for the initial presentation. In that case, functionality took precedence over aesthetics.

10. What are the risks associated with having the wrong algorithm in place as a long-term solution?

From what I understand, sorting is generally not the bottleneck in a program; however, in order to optimize for maximum performance, an efficient algorithm should be chosen based on the expectations of use. If a less than optimal algorithm is chosen, performance will be adversely affected.