CSE143X Lecture Questions for Wednesday, 12/9/20

Time (e.g., 12:45)	Question	Answer
14:00	Super disappointed you didn't test the bogo sort with the cards - a true scientist would've checked to see if that was an efficient algorithm.	I'll take that as a suggestion for next time. :-)
38:30	Is there a way to make a static method like mergeInt(Queue<String> result, Queue<String> list1, Queue<String> list2) generic (able to take any type of Queue, not just Queue<String>) without putting it in a generic class? ~~That's it? You still are taking a Queue<String> argument…~~ Ok :)	Yes. Let me get a compiled version to show you. Change the header to this and it will be a generic sort: public static <T extends Comparable<T>> void mergeInto(Queue<T> result, …) Good point...change String to T in the header.
	When you refer to sorting algorithms as typically having complexity n^2 or n log (n), that is in reference to comparison sorting algorithms, only, right? E.g. Radix sort (here's a complete list).	What sorting algorithms don't involve comparisons? Radix requires finding specific locations, which is comparing values. I understand what you're talking about, but it's an odd distinction. What I can say is that sometimes when you know something about your data, you can do better than O(n log n), but it won't be a generic sort.
27:40	quicksort// If you know the minimum and max in some set/array, why dont you just pull the average and go through with quicksort? Seems efficient Re: ohh i see! Thank you One more thing, i dont really get how multiplying the area of the triangle is getting you the complexity of the algorithm (the whole n * n / 2 cut the 2 so O(n^2) thing)? Got ittt thank you!	Those situations are rare. You don't tend to know what the average value is going to be. It takes O(n) time to find the midpoint, which would defeat the purpose. But there are lots of interesting ideas people have come up with over the years to try to choose a good pivot. I was trying to appeal to your intuition (not a proof). I was putting a dot for each basic operation and I was arguing that the total number of dots you would end up with would fill half of an nXn square. So the total number of dots (total number of operations) would be half of n^2.

	Are you still here stuart (next page) The ratio goes from 1.7 to 2.6 just from adding <= ??? Hmm okay! Thank you again :-)	Yes I wouldn’t put too much emphasis on how those numbers turn out. The data is noisy. I don’t expect that you’d see a difference if you averaged over a lot of runs. Try running it on your own machine. It’s linked from the calendar.