CSE143X Lecture Questions for Monday, 12/7/20

Time (e.g., 12:45)	Question	Answer
	Unrelated, but since you're using a Mac, do you have an opinion on the new M1 chips and their impact on computing? Different person (using M1 mac): It's pretty good they definitely weren't lying about the 20 hours of battery life.	I haven't followed it closely enough to have a strong opinion.
	In Big O, does log always mean log base 2? Very cool. Thanks.	In Big-O notation, we ignore constant multipliers. So we'd say O(n) rather than O(2n). For logarithms, every log is a constant multiple of every other log that uses a different base. So we don't include the idea that the log is to the base 2 because that's just a constant multiplier relative to every other log.
	What is E? Is it an interface or class that all wrapper types implement/inherit?	It's a placeholder that will be filled in with a type. So it's a kind of parameter. We wrote SearchTree<E> and in the client code I filled in the E with String and with Integer.
	How long would 50mil numbers take to run on Algorithm 1? Just out of curiosity. Is this code going to be on the calendar? Really good argument for optimization right there. Nice.	Yes, the code will be on the calendar. Running 50 mil numbers on algorithm 1? A LONG time. I ran it for 2500 items and it took 0.853 seconds. 50M is 20K times as big, which would take 8*10^12 times as long to run. A little over 216 thousand years.

	It seems to me that none of the algorithms are fully correct for two cases: an empty list and a list with all negative numbers. In ~~both~~ the first case, the algorithm results in a start of 0 and a stop of 0 (meaning the first element). I think you should initialize the index variables to -1 (or something similar) to indicate that no values of the array are being used. Yes, if the algorithm can't return an empty subsequence, it would work for negatives. But if it could return an empty subsequence, it should do so. Sounds good!	I'm revising my answer to this question. You make a good point. I think I should have said that the problem is to find the maximum sum for a nonempty subsequence. That would mean we wouldn't try to solve the problem for an empty array. For a nonempty list with all negatives, it will find the index with the largest negative value. For example, if the array stores [-3, -9, -14,-2, -7], it will set start/stop to 3 because -2 is the max sum.

	You mentioned that you have to take into account all lines of code (including those within library methods). I think it's fair to say that many operations in Java allocate memory (using data structures, for example). Would it be appropriate to factor in the time/complexity of the memory allocator/garbage collector when analyzing such algorithms?
	
	For a concrete example, the following code allocates an object n times, but finding memory to store that object is probably not an O(1) operation. Also, the garbage collector will have to run.
	
	for (int i = 0; i < n; i++) {
	  new Object(); // n times
	}
		Memory allocation should be amortized O(1). Yes, the garbage collector might be invoked for some specific call, but averaged over all of the calls on new, it will be O(1).
		
		I mentioned the special case for allocating an array because all of its elements must be initialized to 0.