CS2852 HW1-6 Feedback

Parenthetical numbers (m-n) indicate HW assigned on Week m, class day n. Due dates are not included here.

HW 3 (2-2)

Big-O complexity for our ArrayList.add

① For the purposes of this class, Big-O complexity, "asymptotic complexity", etc. always mean "worst-case.

② In addition to looking at the max iterations for a loop, we also need to look at the Big-O cost of the body of the loop. If one line on the loop takes O(f(n)), and the rest are O(1), and the loop loops O(n) times, the total time is O(n)*(O(f(n))+O(1)) = O(n*(f(n)+1)) = O(n f(n)).

HW4, prob 2 (2-3)

Impl. LinkedList.add(…)

① add(int, E) should use the int as an index indicating where to insert the node.

② New node not tied in correctly.

③ Does not handle empty-list case correctly (when head is null).

④ Solution attempts to use wrong internal data-structure. See class examples / memory map diagrams for correct approach.

HW5, (4-3)

① Please format code correctly.

② We can avoid O(n) work by not calling size() internally, since this is an O(n) operation. This does not change the order of an algorithm, but it does increase the efficiency of a "library" operation.

④ For loop could be better here for the simple counting loop.

⑤ Does not handle empty-list case correctly (when head is null).

⑥ Cannot return after throwing an exception. The calling method must handle the exception and will not use the return value.

⑦ Can handle end-of-list and middle-of-list code with the same code. Doesn't make any difference for a singly-linked list.

⑧ No need to update size (unless you are caching that). If you don't cache size, see ②.

⑨ Caching size can actually simplify the algorithms… as many of you show.

⑩ Can combine `if(curr.next != null) { prev.next = curr.next;} else {prev.next = null;}` into `prev.next = curr.next;`