Joel Turbi

Dr. Zavala

CS312 – Analysis of Algorithms

Homework Assignment 5 – MERGE-SORT Implementation - Questions

9/22/19

Questions:

1. In a sentence describe what you think the test in the first block of comments are supposed to do.

* The tests in the first block of comments are verifying that I have correctly implemented the isSorted() method.

1. Other than differences in the names of the left and right arrays and some indexes ( arrays start at 0 in C++ and the textbook starts them at 1), what significant differences do you notice between the MERGE pseudocode in the book and the *merge* function provided with the started code?

* The implementation of merge in the starter code forgoes the use of sentinels (i.e. The ∞ that the pseudocode uses to determine when it has reached the end of the sub-arrays it created). These sentinels allow the pseudocode to add all the elements from both subarrays to their proper positions in array A without having to implement two more for-loops to catch the remaining element left over in either the left or right subarrays.

1. In the pseudocode, what is the purpose of putting *sentinels* at the ends of the arrays L and R (lines 8-9)?

* The purpose of putting sentinels at the end of the arrays L and R is to verify that the end of the array has been reached.

1. Why aren’t lines 8-9 of the MERGE pseudocode implemented in the *merge* function?

* Lines 8-9 of the Merge pseudocode aren't implemented, because the merge function in the starter code chooses to use two additional loops in order to check for any remaining elements in the L and R arrays.

1. Why is a while loop used in the *merge* function instead of the for loop in lines 12-17 of the MERGE pseudocode?

* A while-loop is used in the starter code over a for-loop because the pseudocode already has a way to check if it has reached the end of the L and R arrays via sentinels and only needs to increment k, while the starter code needs to check if it's in the range of the L and R arrays, too.

1. Why are there two extra while loops at the end of the *merge* function? And why aren’t they in the MERGE pseudocode?

* There are two extra while loops in the merge function, because the function is checking for any leftover items in the L and R arrays. While the pseudocode already knows it has reached the end of the L and R arrays due to implemented sentinels.