Java Data Structures

Pre-Course Assessment

1. Order the following algorithms in the ascending order of their best-case complexity:

Bubble sort

Selection sort

Quick sort

Merge sort

a. Bubble sort < Quick sort == Merge sort < Selection sort

b. Quick sort < Bubble sort == Merge sort < Selection sort

c. Selection sort < Quick sort < Merge sort < Bubble sort

d. Selection sort < Bubble sort < Merge sort < Quick sort

Analysis:

1. Correct. Bubble sort has a complexity of O(n) in the best-case scenario. Quick sort and merge sort have O(n log(n)), and selection sort has O(n^2). See Module 2: Sorting Algorithms and Fundamental Data Structures, Lesson 2.3: Using Merge Sort.
2. Incorrect. Bubble sort has a complexity of O(n) in the best-case scenario. Quick sort and merge sort have O(n log(n)), and selection sort has O(n^2). See Module 2: Sorting Algorithms and Fundamental Data Structures, Lesson 2.3: Using Merge Sort.
3. Incorrect. Bubble sort has a complexity of O(n) in the best-case scenario. Quick sort and merge sort have O(n log(n)), and selection sort has O(n^2). See Module 2: Sorting Algorithms and Fundamental Data Structures, Lesson 2.3: Using Merge Sort.
4. Incorrect. Bubble sort has a complexity of O(n) in the best-case scenario. Quick sort and merge sort have O(n log(n)), and selection sort has O(n^2). See Module 2: Sorting Algorithms and Fundamental Data Structures, Lesson 2.3: Using Merge Sort.

**… … …**

26. What is the Big-O for the following function?

void algo(int n, int x) {

for (int k = 0; k < n; ++k)

if (x < 99) {

for (int i = 0; i < n; ++i)

for (int j = 0; j < i; ++j)

System.out.println(”x = ” + x);

} else {

System.out.println(”x = ” + x);

}

}

a. O(n^3)

b. O(n log n)

c. O(n)

d. O(n^2)

Analysis:

1. Correct. The two arrays are traversed by the parameter n, and the outer loop traverses the array from I to n; so O(n) \* O(n^2), giving us O(n^3). See Module 1: Algorithms and Complexities, Lesson 1.2: Measuring Algorithmic Complexity with Big O Notation.
2. Incorrect. There is no division into halves by this algorithm function, so it is not logarithmic. See Module 1: Algorithms and Complexities, Lesson 1.2: Measuring Algorithmic Complexity with Big O Notation.
3. Incorrect. The array is traversed by multiple nested loops. See Module 1: Algorithms and Complexities, Lesson 1.2: Measuring Algorithmic Complexity with Big O Notation.
4. Incorrect. The array is traversed three times with loops over the length. See Module 1: Algorithms and Complexities, Lesson 1.2: Measuring Algorithmic Complexity with Big O Notation.