Competency #4 and #5 Study Guide

Competencies #4 and 5 will be assessed at the same time since they were presented as combined topics and there is overlap in their topic coverage. These address material from chapters 4 and 5 which I present slightly out of order over weeks 7-10

Competency #4 - Searching and Sorting Algorithms

- Trace, explain, and analyze common search/sort techniques such as linear search, binary search
- Explain and analyze simple and advanced sorts such as bubble, selection, insertion, merge, and quick sorts.

Competency #5 - Recursive Algorithms

- Define recursion and identify the components of a recursive function, including the base case and the recursive case.
- Trace the execution of a recursive function, demonstrating understanding by outlining the calls and return values step-by-step.

If you go back through the “Outcomes” sections of weeks 7-10 you will see these ideas broken down into more specific detail.

Weeks 7, 9, and 10 (Topic 4)

- Describe the algorithm for a linear search
- Describe the algorithm for a binary search
- Recognize/identify the code for a linear or binary search.
- Identify the [best-case | worst-case | average-case] Big-oh runtime analysis for a linear or binary search.
- Explain where O(log n) fits into the "rankings" of O(1), O(n) and O(n^2)
-
- Describe the algorithm for a bubble sort
- Describe the algorithm for an insertion sort
- Describe the algorithm for a selection sort
- Explain why the runtime for each of the previous sorts is O(n^2)
- Explain why, even though each of the previous sorts is O(n^2), that the bubble sort is still considered the worst of the three.

-
- Describe the algorithm for a merge sort
- Describe the algorithm for a quck sort
- Explain why the runtime for each of the previous sorts is O(n log n)
- Include O(n log n) in discussion about Big-Oh notation (including
    - where it falls in the rankings of other categories
    - what it means compared to O(n) and O(n^2) (the two categories that come before and after it)
- Read existing Python code and connect it to corresponding pseudocode [We begin the process and continue over the next few weeks]

Weeks 8 and 10 (Recursion)

- Describe the three "laws" of recursion.
- Identify where each of the three "laws" of recursion are observed in a recursive algorithm. [Both what triggers them and what actions they perform]
- Identify the Big-oh runtime analysis for a recursive algorithm.
-
- Describe the algorithm for a merge sort
- Describe the algorithm for a quick sort
- Explain how recursion is used in the two previous sorts
- Explain why the runtime for each of the previous sorts is O(n log n)
- Include O(n log n) in discussion about Big-Oh notation (including
    - where it falls in the rankings of other categories
    - what it means compared to O(n) and O(n^2) (the two categories that come before and after it)
- Calculate the result of a simple recursive algorithm
- Discuss why we use recursion (vs. just iteration)
- Consider what happens if one of the three “laws” of recursion isn’t fulfilled in an algorithm