CD2 Study Guide

CD2 addresses topics from Chapters 3-5 in your textbook which were studied over weeks 5-10. It covers three topics/outcome sets in the course:

Topic 3 - Linear Data Structures

- Trace, identify and explain common "linear" data structures constructed using "arrays" (i.e., contiguous block of memory) and "linked nodes" as appropriate: stack, queue, and list. **[added deque]**

Topic 4 - Searching and Sorting Algorithms

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

Topic 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 5-10 you will see these ideas broken down into more specific detail.

Week Five and Six (Topic 3)

- Explain the main operators of a stack (understand and discuss the ADT description). push()
    - peek()
    - size()
    - is_empty()
    - pop()
- Explain the main operators of a queue (understand and discuss the ADT description).
    - enqueue()
    - dequeue()
    - size()

    - is_empty()
- Explain the main operators of a deque (understand and discuss the ADT description). add_front()
    - add_rear()
    - remove_front()
    - remove_rear()
    - size()
    - is_empty()
- Explain the pseudocode for the operators of a [stack | queue | deque]
- Identify the Big-Oh notation for the operators of a [stack | queue | deque]
- Explain how a Python list-based implementation of a [stack | queue | deque] fulfills the operators (ADT) of the stack.
    - **[Newly added] Consider how a change to the implementation of a [**stack | queue | deque] **might change the performance (Big-oh) of the operators.**
- Provide [or consider, or recognize] an example of where a [stack | queue | deque] could be used in programming
- **[Newly added] Given a sequence of actions on a [stack | queue | deque] determine the result(s).**

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.
- Explain how recursion is used in the merge and quick sorts
- **[New] Calculate the result of a simple recursive algorithm**
- **[New] Discuss why we use recursion (vs. just iteration)**
- **[New] Consider what happens if one of the three “laws” of recursion isn’t fulfilled in an algorithm**