CS260 Quiz 1

Attempt to answer the questions with your mind alone and then go online or get together with other CS people to verify, expand, or possibly find the answers to the following questions:

1. What is an Abstract Data Type (ADT)? List a few examples of ADTs.

An abstract data type is a model of a kind of a structure intended for organization of data

List, tree, table

1. What is a data structure? List a few examples of data structures.

A way of organizing data so that it can be looked at as efficient.

Linked List, Binary Search Tree, HashTable

1. What are some important distinctions between ADTs and data structures?

The ADT is like an interface, (what the data does)

The data structure is the way it is implemented

1. What are the key differences between LIFOs and FIFOs? What are some common examples of structures with these behaviors?

FIFO – First In First Out – a real world example of this could be like a Line at the DMV, the first person in line (in most situations) is the first person to get service.

LIFO – Last In First Out – like a stack of plates, the last one you put onto the stack is the most accessible one.

1. What are some benefits of using an array to store the elements of a stack? Are there any downsides? Can any of those downsides be mitigated by some clever design or trick?  
     
   Linked List is better at adding new elements, and removing unwanted ones, whereas the Arraylist is better at querying the stored information. This is due to the Linkedlist needing to check nodes to find where it need to go next, however, the arraylist when it wants to add an element needs to expand where the linkedlist is sort of s free diagram where things can be added and removed, only changes needed are node locations.

From my research on this, it seems like these design problems are not something you can work around to make it better than the other, one will always be better at its respective method.

1. What are some benefits of using a list of linked nodes to store the elements of a stack? Are there any downsides? Can any of those downsides be mitigated by some clever design or trick?  
     
   Advantages: Insert and delete operations are more efficient, no need to pre-allocate memory.

Problems: Pointers require more memory for storage, there is no random access, you need to go through piece by piece to find data you need.

There didn’t seem to be ways around these problems

1. What is the goal of doing complexity analysis on a design, structure, or implemented behavior?  
     
   To find the efficiency of methods used on elements, that helps us calculate roughly how long the methods will take, how many data accesses we will need, and allows you to compare your current method against others that had been created before.
2. What is the benefit of doing complexity analysis on the time or operation count of some design or behavior?  
     
   to find the most efficient method of data storage, and sorting based on your needs.
3. What is the benefit of doing complexity analysis on the space or element count of some design or structure?
4. What are some of the symbols and ideas used in complexity analysis?

Big O notation – creates a formula based on number of operations needed to finish

O notation is used for Upper bounds

Ω Is used for lower-bounds