CS Data Structures 2

Concept Review

Recursion

- Recursion occurs when a function calls itself
- Any loop can be written instead with recursion; any recursion can be written instead with a loop
- Recursion is often useful in graph and tree problems

Graphs

- Graphs are like trees, except they can contain loops (“cycles”), and can be non-directed
- Nodes (or vertices) are connected by edges (or arcs).
- Graphs are useful for (among other things) tracking connections for a set of data, describing dependencies and finding efficient ways to get from one state to another.

CompSci Data Structures

- Python List allocates contiguous space for items
  - Pre-allocates extra space to grow
- Python Dictionaries and Sets are hashed
  - hash: stable “one-way” conversion of data to fixed-size result
- Queues are good for Linked List and Double-Linked List
- Stacks are good for any, but tricky for Linked-List

Sorting

- Sorting is commonly needed method

- Factors when choosing Sorting Algorithms:
  - Runtime
  - Space Requirements
  - Likely Structure of your data:
    - Random?
    - Almost reversed?
    - Almost sorted?
    - Likely duplicates?

Git Branching

- Git Branching is good for:
  - Testing
  - Maintaining separate releases
- When you finish working on a branch, you merge it to master
- Many software companies use branching and pull requests to manage multiple developers on the same project

Part 1: Discussion Questions

Recursion

1. In your own words, what is Recursion?
2. Why is it necessary to have a Base Case?

Graphs

1. What is a Graph?
2. How is a Graph different from a Tree?
3. Give an example of somthing that would be good to model with a Graph.

Performance of Different Data Structures

Fill in the missing spots in the chart with the correct runtimes. Do this by reasoning through how the data structures work, NOT by looking up the solution. Add-R means add to the right/end/top and Add-L means add to the left/beginning. There are Xs in the spots where that operation doesn't make sense for that data

structure (for instance, you can't index a Stack, or pop from the end of a Queue). We've provided the first few answers for you.

Fill in the runtimes for the following actions for the table below:

Data Structure	***Index***	***Search***	***Add-R***	***Add-L***	***Pop-L***	***Pop-R***
Python List (Array)	*O(1)*	*O(n)*	*O(1)*			
Linked List						
Doubly-Linked List						
Queue (as Array)	*X*	*X*		*X*		*X*
Queue (as LL or DLL)	*X*	*X*		*X*		*X*
Stack (as Array, LL, or DLL)	*X*	*X*		*X*	*X*	
Deque (as DLL)	*X*	*X*				

- ***Index:*** *Find an item in the structure when you know its position*
- ***Search:*** *Find an item in the structure when you know its data*
- ***Add(R/L):*** *Set a key in set/dictionary or add node to tree*
- ***Pop(R/L):*** *Remove a key or node*

Fill in Runtime and Memory:

The answers for Dictionary have been provided; you should fill in the rest:

Data Structure	***Get***	***Add***	***Delete***	***Iterate***	***Memory***
Dictionary (Hash Map)	*O(1)*	*O(1)*	*O(1)*	*O(n)*	*medium*
Set (Hash Map)					
Binary Search Tree					
Tree					

- ***Get:*** *Find an item in the structure*
- ***Add:*** *Set a key in set/dictionary or add node to tree*
- ***Delete:*** *Remove a key or node*
- ***Iterate:*** *Find next item in data structure*
- ***Memory:*** *Relative to data, how much memory is used? (Choices: a little, medium, or a lot)*

Sorting

1. Describe in words how the Bubble Sort algorithm works.
2. Describe in words how the Merge Sort algorithm works.
3. Describe in words how the Quick Sort algorithm works.

Git Branching

1. Give an instance when you would use git branching.
2. What is a pull request?

Part 2: Skills Assessment

Recursion

Finish the functions in the **recursion.py**.

1. Print a list recursively.
2. Print tree data recursively.
3. Find the length of a list recursively.
4. Find the number of nodes in a tree recursively.

Sorting

Finish the functions in the **sorting.py**.

1. Write a bubble sort algorithm.
2. Write a function that merges two already sorted lists.

Advanced

1. Implement merge sort.