15-110 Hw4 - Written Portion

Name:

AndrewID:

#1 - Optimizing for Search - 5pts

You have been given a very large dataset of temperatures (represented as floats), and your task is to find how many of the data points fall into specific temperature ranges (such as 40 degrees to 50 degrees, or 75.7 degrees to 78.2 degrees). To do this, you want to store the data in a data structure so that, given any range, you'll be able to:

- find the smallest value in the structure that falls in that range
- find the largest value in the structure that falls in that range
- look between the two elements to quickly find all values that fall within the range

You want to optimize how quickly you can run the algorithm shown above, assuming the data structure has already been created. Choose the best search algorithm + data structure combination for the task. There might be multiple correct answers; you only need to choose one per question.

Search Algorithm:

- ☐ Linear Search
- ☐ Binary Search
- ☐ Hashed Search
- ☐ Breadth-First Search

Data Structure:

- ☐ List
- ☐ Dictionary
- ☐ Tree
- ☐ Graph

How would you organize the data in your chosen data structure to optimize the efficiency of your search algorithm?

#2 - Recognizing Data Structures - 10pts

For each of the following types of data, choose the data structure that would be the **best/most natural choice** to represent the data

Carnegie Mellon's college/department/major organization

- ☐ List
- ☐ Dictionary
- ☐ Tree
- ☐ Graph

A budget spreadsheet, with columns for cost and category, and rows for individual purchases

- ☐ List
- ☐ Dictionary
- ☐ Tree
- ☐ Graph

An attendance sheet of student names enrolled in a course

- ☐ List
- ☐ Dictionary
- ☐ Tree
- ☐ Graph

The subway map for London

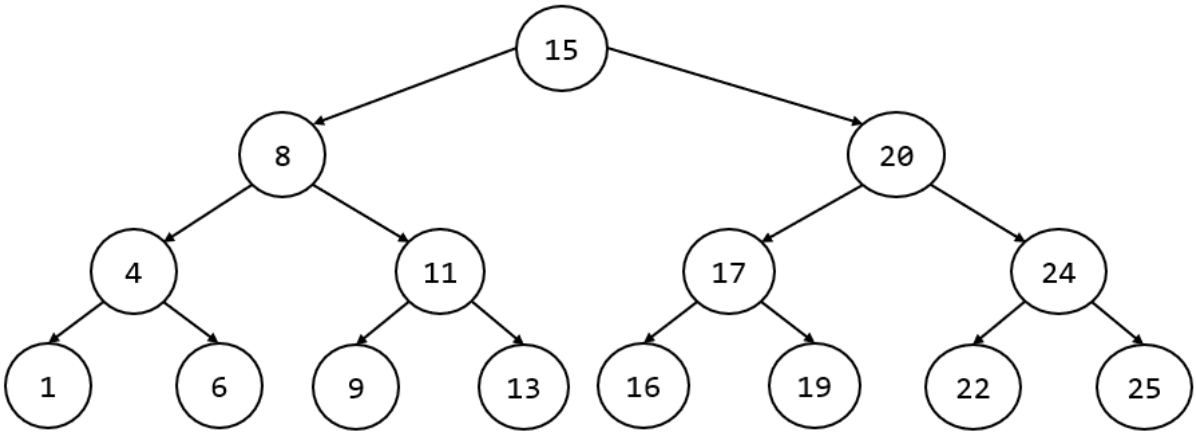
- ☐ List
- ☐ Dictionary
- ☐ Tree
- ☐ Graph

A deck of flash cards with words on one side and definitions on the other

- ☐ List
- ☐ Dictionary
- ☐ Tree
- ☐ Graph

#3 - Searching a BST - 9pts

Given the Binary Search Tree shown below:



What series of numbers would you visit if you ran a search algorithm that looked for **19**?

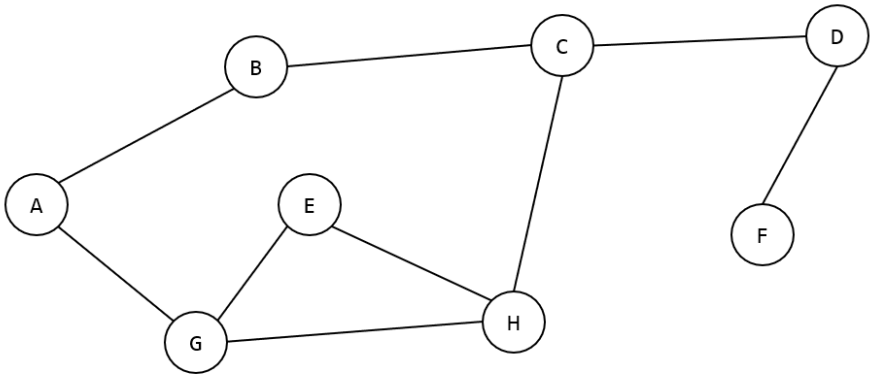
What series of numbers would you visit if you ran a search algorithm that looked for **4**?

What series of numbers would you visit if you ran a search algorithm that looked for **10**?

#4 - Searching a Graph - 12pts

For this problem, note that each prompt has multiple correct answers; you only need to include one.

Given the undirected graph shown below, where the letter A is the start node:



What series of letters could you visit if you ran Depth-First Search to find **H**?

What series of letters could you visit if you ran Breadth-First Search to find **H**?

What series of letters could you visit if you ran Depth-First Search to find **J**?

What series of letters could you visit if you ran Breadth-First Search to find **J**?

#5 - Advanced Hashing - 5pts

Recall our discussion of what hash functions are and what they are used for. Should it be possible to write a hash function for a tree, so we can use trees as dictionary keys? Select the best answer.

- ☐ Yes, we just need to map each of the node values to a number.
- ☐ Yes, but we can't change the data in the tree after adding it to the dictionary.
- ☐ No, because trees are mutable, and we can't have mutable values in a set.
- ☐ No, because only integers can be used as hash indexes.

#6 - P and NP Identification - 10pts

For each of the following problems, identify whether it is in the complexity class P, NP, or neither.

Finding the smallest value in a tree	☐ P ☐ NP ☐ Neither
Scheduling final exams for CMU so that there are no conflicts	☐ P ☐ NP ☐ Neither
Determining if an item is in a list	☐ P ☐ NP ☐ Neither
Finding the best next move to make in a chess game	☐ P ☐ NP ☐ Neither
Determining if there is a set of inputs that makes a circuit output 1	☐ P ☐ NP ☐ Neither

#7 - P and NP Comprehension - 9pts

In your own words (and not directly copied from the slides), briefly define each of the following terms. Each definition should be no longer than twenty words in length.

Complexity Class P:

Complexity Class NP:

Complexity Class NP-Complete:

Programming Problems

Each of these problems should be solved in the starter file available on the course website. They should be submitted to the Gradescope assignment Hw4 (Programming) to be autograded.

Most programming problems may also be checked by running the starter file, which calls the function testAll() to run test cases.

#1 - createAuthorMap(bookMap) - 10pts

Write the function createAuthorMap which takes bookMap, a dictionary that maps book titles (strings) to their authors (also strings), and returns a new dictionary that maps authors to lists of all the books they've written. Make sure not to destructively modify bookMap as you generate the new dictionary!

#2 - findFile(t, name) - 10pts

We can represent a filesystem as a tree by making each folder a node with children, where the children are the items in the folder. Files are then nodes with no children. We'll use the general dictionary tree structure we demonstrated in class for this problem, which maps the key "children" to a list of dictionaries (child nodes).

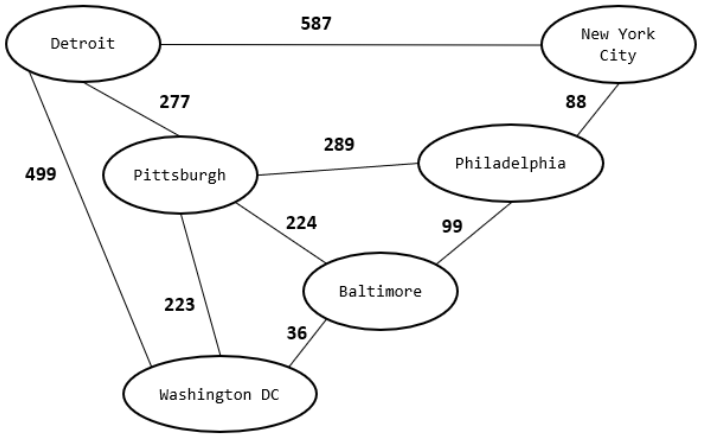
Given a tree t that represents a file system and a filename name (a string), write the function findFile(t, name) which returns True if that filename exists in the file system, and False otherwise. Note that the filenames will be in the **values** of the nodes.

Solving this problem for a tree with no children (a file) should be easy. To solve the problem for a tree with children (a folder), consider this: if you can find the filename in **any** of the child trees, then you can return True right away! On the other hand, if you search **all** the children and the file is in none of them, you can return False.

#3 - nearestNeighbor(nodeList, edgeMatrix, node) - 10pts

We can represent cities in the United States as nodes in a graph, and use edges to connect the cities that offer flights from one to the other. We can also give those edges values- the distance from one city to another.

Write the function nearestNeighbor(nodeList, edgeMatrix, node) which takes a graph in adjacency matrix format (nodeList and edgeMatrix) and a string, node (a city name), and returns the closest neighbor to node. For example, in the graph shown below, Pittsburgh's nearest neighbor would be Washington DC.



Recall that in adjacency matrix format, a graph's **node list** is a 1D list that maps each index to a node value (in this case, the name of a city), and an **edge matrix** is a 2D list, where matrix[i][j] represents the value of the edge between nodeList[i] and nodeList[j], or None if they are not connected.

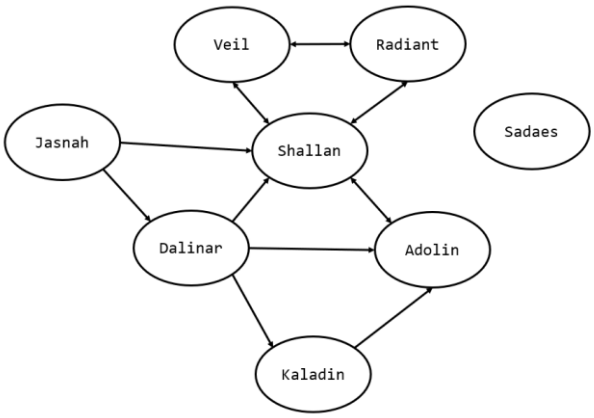
To write this function, you will need to take the following steps:

1. Identify the index in nodeList of the given city, **node**.
2. Keep track of two data values - the current closest city and the current shortest distance. These can start as None and a very large number, like 1000.
3. Iterate over each city index in the edge matrix.
    a. If there exists an edge between that city index and our node city, compare the distance between the two to the current shortest distance.
    b. If the edge distance is smaller, update the current closest city and current shortest distance.
4. Once you have checked all the cities, return the current closest city.

#4 - getAllFollowers(network, name) - 10pts

It's easy to determine the number of followers you have on a social media platform like Twitter, but the displayed number does not account for people who follow your followers, but do not (yet) follow you. We want to determine how many **eventual-followers** a person has. An eventual-follower of X is either a) someone who follows X, or b) someone other than X who is a follower of another person, Y, who is an eventual-follower of X.

For example, in the social network shown below, an arrow leading from Jasnah to Dalinar means that Dalinar is a follower of Jasnah; Jasnah sends posts to Dalinar. A double-headed arrow means that two people follow and send posts to each other. Kaladin only has one direct follower- Adolin- but through Adolin, he has three more eventual-followers (Shallan, Veil, and Radiant). getAllFollowers(network, "Kaladin") would return ["Adolin", "Shallan", "Veil", "Radiant"].



Write the function getAllFollowers(network, name) which takes a graph in the dictionary format, *network*, and a string, *name*, and returns a list of all eventual-followers of name. The graph shown above would be stored in the dictionary format as follows:

```
{ "Jasnah" : [ "Shallan", "Dalinar" ],
  "Dalinar" : [ "Shallan", "Adolin", "Kaladin" ],
  "Kaladin" : [ "Adolin" ],
  "Adolin" : [ "Shallan" ],
  "Shallan" : [ "Adolin", "Veil", "Radiant" ],
  "Veil" : [ "Shallan", "Radiant" ],
  "Radiant" : [ "Shallan", "Veil" ],
  "Sadaes" : [ ] }
```

Major Hint: This problem is a variant on the idea of searching a graph! Instead of searching for an item, it's searching for ALL the nodes connected to the start.

If you're lost, try starting with breadth-first search or depth-first search from the class notes, then modify the code to return a list of followers instead of True or False. You'll need to change two parts of the function: the part that actually checks if an item has been found, and the value that is returned at the end.