Assignment – 5

Total Points: 80

Instructions:

- Download the attached python file assignment_5.py (make sure to NOT change the name of this file).
- Follow the instructions and replace all TODO comments in the scaffolding code.
- Test your code as much as you can to make certain it is correct.
- Run flake8 in addition to testing your code; I expect professional and clear code with minimal flake8 warnings and having McCabe complexity (<10) from all of you.
- Create a write up with formatted code and screenshots of your output, running the McCabe complexity command and error free console.
- Save the write-up as a PDF and submit it along with your python code (file name assignment_5.py) as separate attachments before the due date on canvas.

Note: Running flake 8

flake8 path/to/your/file (for warnings and errors)

flake8 --max-complexity 10 path/to/your/file (for complexity)

Problem 1 (Max points: 10)

Given a binary tree, return the *level order* traversal of its nodes' values. (i.e., from left to right, level by level).

For example:

Given binary tree

```
    3
   / \
  9  20
    /  \
   15   7
```

return its level order traversal as:

```
[
  [3],
  [9,20],
  [15,7]
]
```

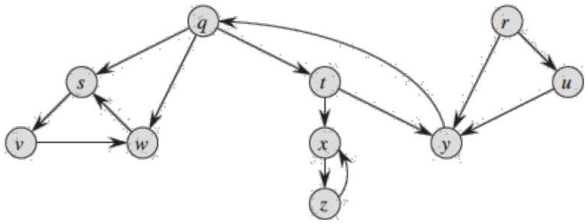
NOTE: The returned value is a list of lists, where each list represents nodes at respective levels.

Problem 2 (Max points: 10)

Given a directed graph and three vertices in it, check whether there is a path from the first given vertex (source) to second vertex (target) passing through the third vertex (intermediate).

For example:

Given the following graph:



Source = q, **Target** = z and **Intermediate** = t returns True, as there is a path from 'q' to 'z' via vertex 't'

(i.e. q -> t -> x -> z).

Source = z, **Target** = q and **Intermediate** = t returns False, as there is no path from 'z' to 'q' via vertex 't'.

NOTE: The returned value is a Boolean, representing whether a path exists or not. You do not need to output the path.

Problem 3 (Max points: 20)

Given a staircase consisting of 'n' steps, it takes n (> 0) steps to reach to the top.

Each time you can either climb 1, 2 or 3 steps. In how many distinct ways can you climb to the top?

For example:

Given n = 3 (number of steps)

Number of ways = **4,** as distinct ways to climb to the top are {(1,1,1), (1, 2), (2, 1), (3)}

NOTE: The returned value is the number of distinct ways and not the actual ways to climb to the top.

Problem 4 (Max points: 20)

Write a program that, given a set $\{X_1, X_2, ..., X_n\}$ of points on the real line, determines the size of the smallest set of unit-length closed intervals that contains all the given points.

For example: Given points {0.8, 4.3, 1.7, 5.4}, the size of the smallest set of unit-length closed intervals to cover them is 3.

NOTE: The returned value is the size of the smallest set, not the set itself.

Problem 5 (Max points: 20)

You are a Teaching Assistant tasked with the job of finding how similar two student submissions are. You have a program that gives you the number of characters in each line of both the submissions. Your job now is to find the length of the longest common subsequence between the two submissions.

A subsequence of a sequence is a new sequence generated from the original sequence with some characters (can be none) deleted without changing the relative order of the remaining characters. (e.g., "125" is a subsequence of "12345" while "153" is not). A common subsequence of two sequences is a subsequence that is common to both sequences.

If there is no common subsequence, return 0.

Example 1:

Input: submission1 = [1, 2, 3, 4, 5], submission2 = [1, 3, 5]

Output: 3

Explanation: The longest common subsequence is [1, 3, 5] and its length is 3.

Example 2:

Input: submission1 = [1, 2, 3], submission2 = [1, 2, 3]

Output: 3

Explanation: The longest common subsequence is [1, 2, 3] and its length is 3.

Example 3:

Input: submission1 = [1, 2, 3], submission2 = [4, 5, 6]

Output: 0

Explanation: There is no such common subsequence, so the result is 0.

NOTE:

1 <= submission1.length <= 1000

1 <= submission2.length <= 1000

Your method returns only the length of the longest subsequence and not the subsequence itself.
