CSC148 Summer 2018: Exercise 6

Due: ~~Thursday, July 12th @ 11PM~~ Friday, July 13th @ 11PM

In this exercise, you are to implement a function called get_nodes_connecting().

To start, download ex6.py and ex6_pyta.txt and read through the code provided in the `if __name__ == '__main__'` block.

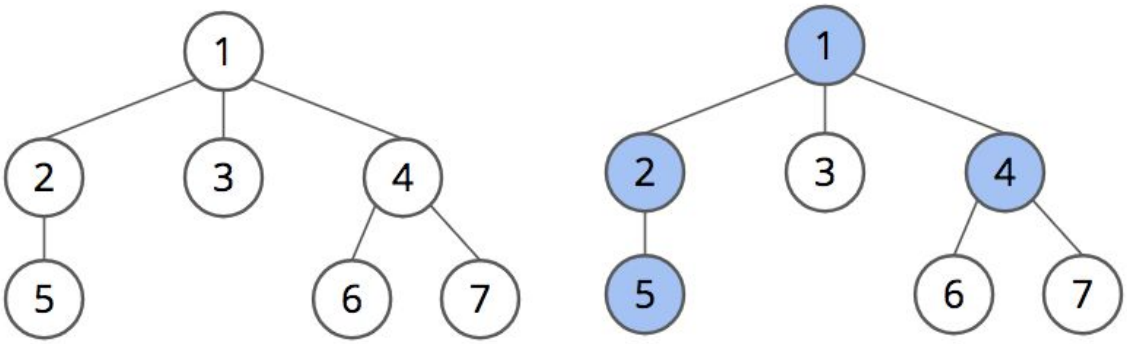
Read through the docstring for get_nodes_connecting() carefully and implement the method. We have provided the ___init___, contains, and ___str___ methods for convenience.

This exercise will require you to have PythonTA installed. If you haven't done so already, go through lab1 and the instructions on the Software page to install and set up PyCharm with PythonTA.

get_nodes_connecting

This method takes in 2 values, value1 and value2, and returns a list of values of nodes [v_1, v_2, ..., v_n], where v_1 is always value1 and v_n is always value2, such that v_i and v_{i+1} have an edge between them (either going from v_i to v_{i+1} or from v_{i+1} to v_i).

For example, consider the Tree t:



The nodes connecting the nodes with values 5 and 4 have the values 2 and 1, so if we called t.get_nodes_connecting(5, 4), we'd want to get [5, 2, 1, 4] back.

If we wanted the nodes connecting 5 and 2, we'd expect [5, 2] to be returned.

If we wanted the nodes connecting 2 and 2, we'd expect [2] to be returned.

If no path exists, for example if we called t.get_nodes_connecting(5, 10), we would want an empty list [] to be returned.

Hint: It might be helpful to create another method that returns the path from the root to a value (i.e. if you're looking for the value 5, then you'd want to get back the path [1, 2, 5] from this helper method).

Submission

Exercises are to be submitted through MarkUs in the ex6 folder. Submit only ex6.py.

To log in to MarkUs, use your UTORid as the log-in name. The password is your teaching labs password. If you have not set this up or have forgotten your password, go to the Teaching Lab's Account Management Page and (re)set your password.

Grading Scheme

This exercise will be graded out of 4 marks, broken down as follows:

- 2 marks for being able to run the client code without issue (no assertion errors raised)
- 1 mark for passing PythonTA
- 1 mark for passing hidden test cases (which use your client code in other ways)
    - Details on what the hidden test cases will/won't test are describe below.

All of these marks are 'all-or-nothing' (i.e. you'll either get 0 on that criteria, or full marks).

Hidden Test Cases

Things that the hidden test case might test:

- Cases where the values are in different subtrees
- Cases where the values are in a single subtree
- Cases where the root's value is one of the values being sought after
- Cases where no path exists

Things that the hidden test case will *not* test:

- A tree containing duplicate values