

CS562

Bowen Yu

01/26/2016

Proposal

For this project, I will use Template Scripting Testing Language(TSTL) to test a python program.

Source link: https://github.com/mirob2005/Python_Data_Structures. This structure contains ADTs, Graphic, HashTable, LinkedList, Sorting and Trees. I will choose programs from ADTs, LinkedList and Tress.

I will test this program in two steps. First, I will write a test file with many different data to test this program. Second, I will use some invalid data to test whether the program can return the right response when it receive wrong data.

For the LinkedList, there are three programs: LinkedList_Single.py, LinkedList_Double.py, and LinkedList_Circle.py. They are quite similar. I will choose LinkedList_Single.py for instance.

LinkedList_Single.py has insert, append, returnIndex, updateIndex, deleteIndex, insertBeforeIndex, insertAfterIndex, deleteData, deletePtr, deleteAllData, insertAfterEveryData, insertBeforeEveryData, deleteList, copyList, findMthToLastNode.

Insert function and append function are the function I will mainly test. I will define a function capacity. And then add an INT can check the capacity.

Delete function is likely the insert function, just need to define a function.

For the Tree, I will test BST.

BST.py has insert, rebalance, rotate, update_height, lookup, cmp, remove, remove_self, take_rightmost, become_child functions.

Insert function: this function is used to insert a new element in the tree and rebalance it if necessary. I will check when the node is leaf or not, the new data is inserted correctly, and whether the new data has been put the right side of the tree. Also, I will test what if I put in the invalid data. Finally, I will test what if the tree has no node, 1 node.

Rebalance function: this function is used to rebalance the tree if it is unbalanced. Since half of the possible rotations for an BST is a mirror of the other half, this function abstracts both cases in one using the variable "side" to refer to one child or the other. I will check is this function able to tell which side is bigger. Then I will test whether this function call the rotate() function correctly.

Rotate function: this function is used to rotate the tree replacing self with one of its children. The replace is done in place so that reference to the root node will continue pointing to the root of the updated tree. For this function, I will test if the function sort the data correctly and the relationship of self() and child(). Then I will test if the tree can be rotated and whether the rotation is correctly.

Update_height function: this function is used to recalculate the height of this tree.

Lookup function: this function is used to look for an item in the tree. It would return a tuple with the data if found, otherwise None. I will test when the data cannot be found, whether it returns None. Likewise, when the data can be found, what will return. Finally, I will test what if I lookup a invalid data.

Remove function: this function is used to remove a node from the tree. I will check its correctness, like when the tree has no node or need to be rotated.

Remove_self function: this function mostly like remove function.

Cmp function is used to wrap around the user_supplied comparison function. Take_rightmost function is used to take the data of the rightmost node in the tree, then destroy it..

Become_child function is used to replace self with one of its children. Those functions are quite simple, and they are called by other functions.