Proposal for project

of CS562

Student ID : 932699829
Student name : Yifan Shen

Reasons for choosing this library:
The library I want to test is a data structure library which deals with the red black tree problem. There are several reasons that I want to test this library. First of all, I am familiar with the tree data structure and know the AVL tree which is similar to the red black tree, especially the insert and delete method. Because of the similarity, I know what is correct and what is wrong which is very important for the testing work. Secondly, the implementation of the red black tree is not very easy and there are many functions that I can call in the program. So I find a library deals with the red black tree on the github. We can get access to the library by: https://github.com/headius/redblack

Introduction to red black tree:
Red black tree is a self-balancing binary tree. The mainly difference between a binary search tree and red black tree is the color of node. Each node has a color and two neighbouring nodes could not be red at the same time. There are five properties of red black tree as below:

1) the root node is black
2) a node is either red or black
3) all leaves nodes are black
4) if a node is red, then both its children are black
5) every path from a given node to any of its descendant leaves nodes contains that same number of black nodes.

The content of test:
Test of the library will be based on the five properties by the help of TSTL(Template Scripting Testing Language).

I will create a tree and check the color of the root node to guarantee this tree satisfies the property 1. I can check all the leaves nodes to see whether they are satisfied with the property 3. I will use the successor and predecessor function to get the children and parent of a node and to check whether they are satisfied with the properties of binary search tree.

I will add or delete a value to the red black tree, and test whether this value is

inserted into or deleted from the right place by the property of binary search tree. And I can test the color to check whether the color of node satisfy the property 4 above. I can count the black nodes on the paths to check whether this operation satisfy the property 5.
I can use the maximum, minimum function to get the smallest and biggest values in the tree and compare them with the values I insert into the tree to check the correctness of these two functions.
The correctness of left_rotate, right_rotate, insert_help and delete_help will be checked when I test the correctness of add and delete function.
Of course, all these operations will be operated by the TSTL language, and I need to tell the criterion to TSTL.

Functions in the library:

__init__(self)	To initialize the red black tree
__str__(self)	To return the size of the red black tree
add(self, key)	To add a value to the red black tree
insert(self, x)	To insert value into the red black tree
delete(self, z)	To delete the given value from tree
minimum(self, x = None)	To get the minimum number in the tree
maximum(self, x = None)	To get the maximum number in the tree
successor(self, x)	To get the successor of a given value
predecessor(self, x)	To get the predecessor of a given value
inorder_walk(self, x = None)	To do the in-order walk
reverse_inorder_walk(self, x = None)	To do the reverse in-order walk
search(self, key, x = None)	To search the value in the tree
is_empty(self)	To test whether the tree is empty
black_height(self, x = None)	To return the height of the black node
__left_rotate(self, x)	To do the left rotation
__right_rotate(self, x)	To do the right rotation
__insert_helper(self, z)	To help insert the given value
__delete_fixup(self, x)	To do the fix up when deleting value