Progress Report

Introduction

This is my primary report about the testing I had done on the library of data structures using TSTL so far.
The library was cloned from https://github.com/nryoung/algorithms .
Under the data structure module , I have tested binary search tree and queue.
The system under test is very well written and I assume that there are not many or no bugs.

Binary Search Tree is a binary tree where each node has a key and the value of each key should be larger than the keys on the left side of the node and should be smaller than the keys on the right side of the node.
In the module I have chosen, the binary search tree has the following functions –

1. Size - The size function returns the number of nodes in binary search tree
2. Is_empty – This function returns true if the bst is empty or else returns false
3. Get – Returns the value that is paired with the key
4. Contains – Returns true if the bst contains a key , else returns false
5. Put – It adds a new key value pair
6. Min_key – It returns the minimum value key in the bst
7. Max_key – It returns the maximum value key in the bst
8. Floor_key – It returns the biggest key that is less than or equal to the given key value
9. Ceiling_key – It returns the node with the smallest key that is greater than or equal to the given key
10. Select_key – Returns the key with rank equal to the rank that is passed as a parameter
11. Rank – It returns the number of keys less than the given key
12. Delete – Removes the node with the key equal to the key passed as a parameter
13. Delete_min – Removes the key value pair with the smallest key in the tree
14. Delete_max – removes the key value pair with the largest key in the tree
15. Keys – This function returns all the keys in ascending order

Testing

I have written three test cases for this module.
I have tested put , contains, is_empty , delete, delete_max functions of the binary search tree.
In the first test, I put a key value pair in the tree , and check if the inserted key contains in the tree and prove that the is_empty is false.
In the second test, I check if the tree is empty , I put a key value pair and delete the key. Then check if tree empty is true.
In the third test, put a key value pair, check contains is true , delete a key and check contains that key is false.

The second code , I have tested in the queue.
Queue has 4 functions

1. Add – Adds element as the last element of the queue
2. Remove – Removes element from the front of the queue and return the value
3. Is_empty – Returns a Boolean value that indicates if the queue is empty
4. Size – returns size of the queue

I have just tested the add , is_empty and remove functions.
In one test case , I added an element to queue.
And checked if the queue is not empty , remove the element that was added.

For further testing , I am planning to test LinkedLists and Stacks.

Code Coverage for queue test case

```
23.0769230769 PERCENT COVERED
6 BRANCHES COVERED
3 STATEMENTS COVERED
```

Coverage.out

```
Name       Stmts   Miss Branch BrPart  Cover   Missing
-----------------------------------------------------
queue.py      13     10      0      0    23%   13-19, 22, 29, 37-49
```

Code Coverage for binary search tree

```
27.3092369478 PERCENT COVERED
67 BRANCHES COVERED
46 STATEMENTS COVERED
```

Coverage .out

```
Name                   Stmts   Miss Branch BrPart  Cover   Missing
------------------------------------------------------------------
binary_search_tree.py     169    123     80      8    27%   14-19, 27-32, 35, 41-61, 63, 68, 72, 82, 92, 111, 121-149,
156, 163, 173, 177-304, 306, 308, 310, 315-325, 335-354, 362, 372-395, 62->63, 67->68, 155->156, 172->173, 305-
>306, 307->308, 309->310, 313->315
```