Final Report

In this project, I have tested the DataStucture python library downloaded from github (https://github.com/Daliprince/DataStructure). This python library is used to build some data structure, such as stack, queue and linked list. The final report includes six parts.

1 Accomplishment

After testing the DataStructure python lib, I have make a big progress on testing the python code using the tstl. In this process, I found some bugs existed in the DataStructure python library. I will first introduce some functions in the DatasStructure python library:

(1) Stack.push(self, data) and Stack.pop(self), these two function exists in the stack.py file. They are used to push and pop data in the stack.

(2) Queue.enqueu(self, data) and Queue.dequeue(self,data), these two function exists in the queue.py file. They are used to add data and delete data in the queue.

(3) LinkedList.insertEnd(self,data) and LinkedList.insertStart(self,data), these two function exists in the LinkedList.py file. The insertEnd() function means adding the data in the end of the linked list. The insertStart() function means adding the data in the beginning of the linked list

(4) Heap.heapsort(self), this function is used to sort the heap to make the datas in heap in right order.

(5) Heap.fixUp(self,index), this function is used to compare the index node to the head of the heap value. If the value of the index node is bigger than the head of the heap value, it will exchange their value.

2 Bugs

At first, I try to test the stack.py file in Datastructure library. However, the tstl kept printing error and said a variable in the python code is not existed no matter how I modified my tstl test file. Finally, I found there was a mistake in the Datastructure python library. In the Stack.py file, the author just define and initialize the self.numOfItems but he use self.numofitems in his pop() function. This capital letter mistake ruins the pop() function. After I change the self.numofitems into self.numOfItems the pop() function goes well in my Stest.tstl file. Except for the capital letter mistake, functions in Stack.py runs no

mistakes in Stest.tstl. Secondly, I tested the queue data structure with qtest.tstl. I tested enqueu() function and dequeue() function to make sure if the data added exists in the queue after enqueue(data) and the data deleted is not existed in the queue after dequeue(data) function. Meanwhile the size of the queue should get smaller after the dequeue() function and the size of the queue should get larger after enqueue() function. Finally, I tested the linked list. In the LinkedList.py file, there are two adding function, insertStart() and insertEnd(). The insertStart() function can add the data in the head of the linked list but the insertEnd() function has a bug (figure 1). That is if the linked list is empty and the insertEnd() function will fail to add the data into the linked list. Meanwhile, I consider the data can be inserted into any place in the linked list. But this linked list in the python library can only add data in head or the end of the linked list.

```
REDUCING...
Reduced test has 3 steps
REDUCED IN 0.0220141410828 SECONDS
data0 = 39                                                  # STEP 0
sll0 = sll.Linkedlist()                                     # STEP 1
sll0.insertEnd(data0)                                       # STEP 2

FINAL VERSION OF TEST, WITH LOGGED REPLAY:
data0 = 39                                                  # STEP 0
sll0 = sll.Linkedlist()                                     # STEP 1
sll0.insertEnd(data0)                                       # STEP 2
ERROR: (<type 'exceptions.AttributeError'>, AttributeError("'NoneType' object has no attribute 'nextNode'",), <traceback object at 0x104684f38>)
```

figure 1

3 Tester

When I tested this python library, I sometimes feel confused about its function. Because there is no instruction and I can just know how to use the function from the code. And some data structure are not completed, for example the linked list just allow the data inserted in the beginning or the end of the linked list. I have tested the heap.py. But the function heapsort() doesn't work and I cannot add a data into a heap with heap.py imported. As there are some little mistakes in some code, I have changed some code in some python code to make sure this python code can be complied successfully. In my source code I submit, I put the original code and the code modified by me in it.

In my opinion, the python library is good. But this python library still has some problems.

(1) Some datastucture function is not completed.

Some data structures have not necessary function. For example, in the AVL tree, there is no delete function. So it can only add data but it cannot delete data in the AVL tree. And the author used a lot of build –in function to build his own data structure. For example, in the stack.py file, the author used stack.pop(data) to pop the data.

(2) There are some small mistakes in the data structure.

These little mistakes are not obvious in the code but they ruin the code. But the tstl can find this mistakes easily. I think the author didn't run his own code for testing. For example, the author use a undefined variable numofitems (actually he defined the variable numOfItems).

(3) Lack of the instruction to help the user use the data structure

The whole python library lacks a instruction to guide the user how to use the data structures. Without the instruction, as a user I can just scan his code to find the way to test the code. It wastes a lot of time and I cannot tell the bug tstl show is caused by the code bug or by my wrong usage of the data structure function.

4 Some advice on TSTL

Though the TSTL is useful and powerful. I still have some advice to make it better. The way TSTL show the bug is not clear. When the TSTL detects a bug, it will show its test steps, which is good but it will also show things like "File "/Users/HengruiGuo/tstl/generators/sut.py", line 12502, in safely act[2]()". This information may be use if I check the sut.py file. But for me I am not familiar with the sut.py file. And the information provided before this, such as the test steps, can tell me what the bug is clear. Sometimes these information occupied the whole screen and confused me a lot. So maybe it will be better to simplified the bug information showing on the screen.

5 Advantage of the TSTL

TSTL is powerful and useful. Because it save a lot of test time. The user just need to design the frame of the test and the TSTL will test the code many times randomly. The TSTL does many repeated work under the user design.

6 Coverage Summary

For the stack.py, my coverage is 27 percent. Actually, I have make use of every function in the stack.py. But it's only 27 percent.

```
STOPPING TEST DUE TO TIMEOUT, TERMINATED AT LENGTH 49
STOPPING TESTING DUE TO TIMEOUT
26.6666666667 PERCENT COVERED
4.0058259964 TOTAL RUNTIME
116 EXECUTED
11549 TOTAL TEST OPERATIONS
2.0324587822 TIME SPENT EXECUTING TEST OPERATIONS
1.89099407196 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
0.0079140663147 TIME SPENT CHECKING PROPERTIES
2.04037284851 TOTAL TIME SPENT RUNNING SUT
0.0214083194733 TIME SPENT RESTARTING
0.0 TIME SPENT REDUCING TEST CASES
6 BRANCHES COVERED
4 STATEMENTS COVERED
```

For the Queue.py, my coverage is 18 percent.

```
STOPPING TEST DUE TO TIMEOUT, TERMINATED AT LENGTH 66
STOPPING TESTING DUE TO TIMEOUT
18.1818181818 PERCENT COVERED
4.00620388985 TOTAL RUNTIME
119 EXECUTED
11866 TOTAL TEST OPERATIONS
1.99280405045 TIME SPENT EXECUTING TEST OPERATIONS
1.93291831017 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
0.0078239440918 TIME SPENT CHECKING PROPERTIES
2.00062799454 TOTAL TIME SPENT RUNNING SUT
0.0194189548492 TIME SPENT RESTARTING
0.0 TIME SPENT REDUCING TEST CASES
4 BRANCHES COVERED
2 STATEMENTS COVERED
```

For the LinkedList.py, my coverage is 13 percent.

```
STOPPING TESTING DUE TO FAILED TEST
12.6984126984 PERCENT COVERED
0.0410900115967 TOTAL RUNTIME
1 EXECUTED
12 TOTAL TEST OPERATIONS
0.0019078254699 7 TIME SPENT EXECUTING TEST OPERATIONS
0.00123071670532 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
1.04904174805e-05 TIME SPENT CHECKING PROPERTIES
0.00191831588745 TOTAL TIME SPENT RUNNING SUT
0.000952959060669 TIME SPENT RESTARTING
0.0220601558685 TIME SPENT REDUCING TEST CASES
11 BRANCHES COVERED
8 STATEMENTS COVERED
sh-3.2#
```