Final Report for Software Test Using TSTL

He Zhang
zhangh7@oregonstate.edu

1. Description of the test

1.1 Test process

The test process is not only the process of finding bugs in SUT, but also the process of familiar with TSTL and SUT itself. I read paper of TSTL and learn example tstl files on Github. Then transform the TSTL code from example to my project. Sometime, especially in the beginning, TSTL file always goes wrong. Without clear error information, I need to test source code in other ways, as running in Python, to help find the bugs. Step by step, more and more functions can be tested by TSTL file. Right now, I am more familiar with both TSTL and Python.

1.2 Test method

To test a specific function, I need to figure out the feature and return value of this function, as well as which functions it calls. Then find a way to test it with TSTL. When running TSTL file, if there is an error, it may either be a bug of SUT or an error in TSTL. Normally I would use Python to re-test this function with random inputs as in TSTL. If containing error again, it is more likely that it is caused by bugs in SUT. Analysis it, find it and modify it. Then do regression rest. If error does not appear when test in Python, it is wise to modify TSTL code.

1.3 Test Result

I have tested three class and part of their functions.

1. Stack Class
   Test push, pop, empty functions.
2. Node Class
   Test append, __len__, leaf_remove, search_child function.
3. BTree Class
   Test mid_order, insert, delete, search, min and max functions directly. And during test these functions, they also call __init__ , treesize, __split, full, search, __search, __delete, __successor, __check_brother_borrow, __merge_brother functions, so these functions are also tested indirectly.

The test shows that the quality of this software under test is not so good. Till now I have found 7 bugs. It has about 580 lines. So the bugs per thousand lines is about 12.5. This is not a low rate.

The main problem of this SUT is that it always forget to consider the special cases, for example empty B-Tree. This is a normal problem for software. Besides this, an bug shows that the code has out-of-bound index when using array.

2. Bugs in SUT

2.1 Bug 1: Do not consider special case

The first kind of bugs is in the mid_order function. This function is to output key values in the B-Tree into a list.
When I tested insert function in SUT I used mid_order function to build assertion in tstl as this:

```
~<BT>.insert(<int>) => \
    (len(<BT,1>.mid_order()) == pre<(len(<BT,1>.mid_order()))>+1)
```

When I ran tstl it reported error like this:

```
Traceback (most recent call last):
  File "randomtester.py", line 469, in <module>
    main()
  File "randomtester.py", line 330, in main
    if a[1]():
  File ".. \generators\sut.py", line 15895, in guard632
    return (self.p_int[2] != None) and (self.p_BT[1] != None) and
((len(self.p_BT[1].mid_order()) == 0) )
  File "..\generators\BTreenew.py", line 295, in mid_order
    if cur_node.is_leaf:
AttributeError: 'NoneType' object has no attribute 'is_leaf'
```

I tried in python, it also reported error.
I check the Source code and found min_order function does not consider the situation of empty B-Tree. So I added code below to modify this bug.

```
if (self.__size == 0):
    return result
```

Other five bugs are alike bug in mid_order function, which is caused by lack of considering the situation of empty B-Tree. They are located in delete, search, min, max functions in BTree Class and leaf_remove function in Node Class. I attach a form which record and analysis these bugs in the end of the report.

2.2 Bug 2: Out-of-bound index when using array

The second one is also in mid_order function.
When I ran tstl it reported error like this:

```
ERROR: (<type 'exceptions.IndexError'>, IndexError('list index out of range',),
<traceback object at 0x04F0CB98>)
TRACEBACK:
  File "..\generators\sut.py", line 27371, in safely
    act[2]()
  File "..\generators\sut.py", line 19825, in act779
    assert      ((self.p_BT[2].mid_order()) ==
[self.p_BT[2].min(),self.p_BT[2].min()])
```

```
  File "..\generators\BTreenew.py", line 291, in min
    return cur_node.keys[0]
```

According to the error information “list index out of range”, I checked in the source code. I found that in the code below, i may larger than the list index of cur_node.keys[]:

```
cur_node, i = stack.pop()
result.append(cur_node.keys[i])
```

So I modify the code by adding the condition and it corrects this bug.

```
cur_node, i = stack.pop()
if i<cur_node._size:
    result.append(cur_node.keys[i])
```

3. Discuss of TSTL

TSTL is a very useful tool for auto-random test. The good thing of TSTL is that it helps to build the core code of random test for python program. The testers just need to focus on SUT and its function if they have learnt how to use TSTL. And with my test project, TSTL truly help to find bugs. If there is something need to be improved, as a tester I think the error information of TSTL is not easy to understand. When running TSTL file but it report error, it hardly give me a clue of where my code is wrong. I need to try many times to fix it. Also, I got a problem once I ran randomtester. When I add the parameter from “-m 50” to “-m 100”, it just stopped after a few seconds, without any information.

4. Coverage Summary

The coverage of test until now is 27%. This is a fair number for this SUT, since it has a lot of branches, functions and classes.

Here is the data in coverage.out:

Name	BTreenew.py
Stmts	428
Miss	319
Branch	182
BrPart	9
Cover	27%
Missing	1-10, 15, 19, 28-31, 34-40, 50, 52, 55-60, 63, 78, 86-188, 195, 200, 228-231, 248-253, 261, 264, 267, 275, 277, 280-285, 290, 293, 298, 301, 307-324, 330-576, 51->52, 247->248, 274->275, 276->277, 278->280, 289->290, 297->298, 305->307, 328->330

Also, for some function, as mid_order function, I test it in different ways and get correct outcome. I think most of the statements in this function should be covered otherwise the outcome should be wrong. But I test a lot of time it still remains some branches uncovered. Otherwise the coverage should be higher than 30%.

Form of Bugs

function	Tstl code	Tstl running error	Bug and Modification	Regression Test

mid_order()	~<BT>.insert(<int>) => \
	(len(<BT,1>.mid_order()) == pre<(len(<BT,1>.mid_order()))>+1)
		Traceback (most recent call last):
		File "randomtester.py", line 469, in <module>
		main()
		File "randomtester.py", line 330, in main
		if a[1]():
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\sut.py", line 15895, in guard632
		return (self.p_int[2] != None) and (self.p_BT[1] != None) and ((len(self.p_BT[1].mid_order()) == 0))
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\BTreenew.py", line 295, in mid_order
		if cur_node.is_leaf:
		AttributeError: 'NoneType' object has no attribute 'is_leaf'
			Does not consider the situation of empty B-Tree.
			if (self.__size == 0):
			return result
				15.4330708661 PERCENT COVERED
				9.62599992752 TOTAL RUNTIME
				100 EXECUTED
				10000 TOTAL TEST OPERATIONS
				6.71798920631 TIME SPENT EXECUTING TEST OPERATIONS
				2.65600919724 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
				0.00699973106384 TIME SPENT CHECKING PROPERTIES
				6.72498893738 TOTAL TIME SPENT RUNNING SUT
				0.102000713348 TIME SPENT RESTARTING
				0.0 TIME SPENT REDUCING TEST CASES
				91 BRANCHES COVERED
				67 TATEMENTS COVERED

delete()	(len(<BT,1>.mid_order()) == 0) -> ~<BT>.delete(<int>) => \
	(len(<BT,1>.mid_order()) == pre<(len(<BT,1>.mid_order()))>)
		ERROR: (<type 'exceptions.TypeError'>, TypeError("object of type 'NoneType' has no len()",), <traceback object at 0x047D2F58>)
		TRACEBACK:
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\sut.py", line 21569, in safely
		act[2]()
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\sut.py", line 15716, in act626
		self.p_BT[0].delete(self.p_int[2])
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\BTreenew.py", line 342, in delete
		node, index = self.search(key)
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-
			Forget to consider special case.
			In delete() function, adding:
			if (self.__size == 0):
			return False
			in search() function, adding:
			if (self.__size == 0):
			return None, None
				20.9702660407 PERCENT COVERED
				9.03599977493 TOTAL RUNTIME
				100 EXECUTED
				10000 TOTAL TEST OPERATIONS
				6.16499853134 TIME SPENT EXECUTING TEST OPERATIONS
				2.53400087357 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
				0.00200009346008 TIME SPENT CHECKING PROPERTIES
				6.1669986248 TOTAL TIME SPENT RUNNING SUT
				0.112000226974 TIME SPENT RESTARTING
				0.0 TIME SPENT REDUCING TEST CASES

search()				

		master\generators\BTreenew.py", line 248, in search
		return self.__search(self.__root, instance)
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\BTreenew.py", line 260, in __search
		cur_len = len(cur_node)
				125 BRANCHES COVERED
				93 STATEMENTS COVERED

min()	(len(<BT,1>.mid_order()) == 0) -> ~<BT>.min() => \
	((<BT,1>.min()) == None)
		ERROR: (<type 'exceptions.AttributeError'>, AttributeError("'NoneType' object has no attribute 'is_leaf'",), <tracebac
		k object at 0x0496CEB8>)
		TRACEBACK:
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\sut.py", line 21908, in safely
		act[2]()
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\sut.py", line 16091, in act640
		self.p_BT[1].min()
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\BTreenew.py", line 277, in min
		while not cur_node.is_leaf:
			Forget to consider special case.
			In min() function, adding:
			if self.__root is None:
			return None
				21.2832550861 PERCENT COVERED
				1.01899981499 TOTAL RUNTIME
				10 EXECUTED
				1000 TOTAL TEST OPERATIONS
				0.566999673843 TIME SPENT EXECUTING TEST OPERATIONS
				0.286999940872 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
				0.0 TIME SPENT CHECKING PROPERTIES
				0.566999673843 TOTAL TIME SPENT RUNNING SUT
				0.063000202179 TIME SPENT RESTARTING
				0.0 TIME SPENT REDUCING TEST CASES
				127 BRANCHES COVERED
				94 STATEMENTS COVERED

max()	(len(<BT,1>.mid_order()) == 0) -> ~<BT>.max() => \
	((<BT,1>.max()) == None)
		ERROR: (<type 'exceptions.AttributeError'>, AttributeError("'NoneType' object has no attribute 'is_leaf'",), <tracebac
		k object at 0x042523F0>)
		TRACEBACK:
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\sut.py", line 22118, in safely
		act[2]()
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\sut.py", line 16216, in act645
		self.p_BT[0].max()
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-
			Forget to consider special case.
			In max() function, adding:
			if self.__root is None:
			return None
				22.5352112676 PERCENT COVERED
				9.25 TOTAL RUNTIME
				100 EXECUTED
				10000 TOTAL TEST OPERATIONS
				6.3119893074 TIME SPENT EXECUTING TEST OPERATIONS
				2.66301035881 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
				0.0260007381439 TIME SPENT CHECKING PROPERTIES
				6.33799004555 TOTAL TIME SPENT RUNNING SUT

		master\generators\BTreenew.py", line 285, in max while not cur_node.is_leaf:		0.0819997787476 TIME SPENT RESTARTING 0.0 TIME SPENT REDUCING TEST CASES 134 BRANCHES COVERED 99 STATEMENTS COVERED
leaf_remove()	(len(<BT,1>.mid_order()) == 1) -> ~<BT>.delete(<BT,1>.min()) => \ (len(<BT,1>.mid_order()) == 0)	ERROR: (<type 'exceptions.AssertionError'>, AssertionError(), <traceback object at 0x047384E0>) TRACEBACK: File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\sut.py", line 26720, in safely act[2]() File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\sut.py", line 17651, in act698 assert ((self.p_NODE[2].__len__()) == 0)	Forget to consider special case.	--

search_child()	((<NODE,1>.__len__()) == 1) -> ~<NODE>.search_child(<int>) => \
	((<NODE,1>.search_child(<int>)) == None)
		ERROR: (<type 'exceptions.TypeError'>, TypeError("'NoneType' object has no attribute '__getitem__'",), <traceback obj
		ect at 0x02CD9A80>)
		TRACEBACK:
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\sut.py", line 27059, in safely
		act[2]()
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\sut.py", line 17710, in act701
		self.p_NODE[1].search_child(self.p_int[0])
		File "E:\Courses\2015-2016\Winter\CS562\tstl-master\tstl-master\generators\BTreenew.py", line 75, in search_child
		return self.childs[x]
			Forget to consider special case.
			In max() function, adding:
			if self.childs is None:
			return None
				26.5573770492 PERCENT COVERED
				96.5759999752 TOTAL RUNTIME
				1000 EXECUTED
				100000 TOTAL TEST OPERATIONS
				73.5460104942 TIME SPENT EXECUTING TEST OPERATIONS
				21.0259962082 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
				0.146000146866 TIME SPENT CHECKING PROPERTIES
				73.6920106411 TOTAL TIME SPENT RUNNING SUT
				0.953998565674 TIME SPENT RESTARTING
				0.0 TIME SPENT REDUCING TEST CASES
				154 BRANCHES COVERED
				113 STATEMENTS COVERED

mid_order()	~<BT>.insert(<int>) => \ (len(<BT,1>.mid_order()) ==	ERROR: (<type 'exceptions.IndexError'>, IndexError('list index out of range',), <traceback object at 0x04F0CB98>)	i may larger than the list index of cur_node.keys[]:	26.5573770492 PERCENT COVERED 96.5759999752 TOTAL RUNTIME

	pre<(len(<BT,1>.mid_order()))>+1)	TRACEBACK:
		File "..\generators\sut.py", line 27371, in safely
		act[2]()
		File "..\generators\sut.py", line 19825, in act779
		assert ((self.p_BT[2].mid_order()) == [self.p_BT[2].min(),self.p_BT[2].min()])
		File "..\generators\BTreenew.py", line 291, in min
		return cur_node.keys[0]
			cur_node, i = stack.pop()
			if i<cur_node._size:
			result.append(cur_node.keys[i])
				1000 EXECUTED
				100000 TOTAL TEST OPERATIONS
				73.5460104942 TIME SPENT EXECUTING TEST OPERATIONS
				21.0259962082 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
				0.146000146866 TIME SPENT CHECKING PROPERTIES
				73.6920106411 TOTAL TIME SPENT RUNNING SUT
				0.953998565674 TIME SPENT RESTARTING
				0.0 TIME SPENT REDUCING TEST CASES
				154 BRANCHES COVERED
				113 STATEMENTS COVERED