Final Report

Student Name: Yipeng Wang

Student ID: 931903609

Date: Mar, 13, 2016

For this term final project, I choose the library which is implemented by Python language to achieve the treap data structure. This data structure has multiply attribute which combines the tree and the heap. In treap, each node should not only follow the rules which is same as the tree data structure, but also has the priority attribute which comes from the unique attribute of heap. This library could be found in GitHub: https://github.com/budabudimir/py-treap. By end of this term, all of the function which are mentioned in this library are tested by using TSTL debug tools.

What did I accomplish:

First of all, I test the insert function in this library. I built a new treap by using TSTL, then I added nodes which has random value and random priority as the root of the treap. After that, I started to insert new nodes in trap which follow the rules from the original python files. After I construct a complete treap, I started to check whether the priority of the leave nodes is less than the priority of his parent node. Next, I tested the delete function, I tried to construct a new treap with root and nodes, all of them have different value and unique priority. Then I used delete function in TSTL, and used contain() function to check whether this node had already been deleted from the treap or not. On the other hand, I checked the merge function, I built two treap data structure, then using merge function in TSTL. After that, I checked the new treap to see whether the priority of the left.treap is less than the priority of the right.treap, then I print the two new treap data structure to see whether it fulfill the requirement of the treap, such as the value of the left.node is larger than the value of the right.node and the priority of the parent.node is always larger than the priority of the child.node. Moreover, I checked the split function. In this part, I built a new treap by using TSTL developer. Then I called the split function by using TSTL debug tool, I got two new treap. For the two new treap, I checked whether they fulfill the requirement of the treap,. Such as te value of the left.node is large than the value of the right.node and the priority of the root is greater than the priority of his child.

Bugs:

After I tested all of functions in treap data structure by using TSTL debug tool, I did not find any bugs from it. In order to make sure that I did not make any mistakes, I try to record the log of each step, therefore, I could check the process of the library at each step. I set m = 10, which means it will provide 100 execution by TSTL. After using TSTL to generate different kinds of random numbers, I checked the log and got the result was there were no bugs exist in this library by using TSTL.

Tester Work:

For treap data structure, even though, it is not a big program, it has lots of useful functions. As the I said above, treap has multiple attribute which not only comes from the tree data structure, but also comes from the heap data structure, which mean, each node from treap should have the same attribute value same as the value attribute from tree data structure, it also have the priority attribute same as the priority attribute from the heap structure. For example, for the insert function, we not only need to consider about the value of the new node with the leave node, but also need to consider the priority of the new node and the original leave node. We need to let the node with high priority go upper as high as it could, and let the larger value of the node sit at the left of the smaller value of the node all the time. As far as I am concerned, this library is good enough to be a test library, but not perfect as a completely library.

For randomtester in TSTL, it is used to execute the SUT randomly by using TSTL. By using randomtester, it could generate random value or list or string or some other data type, and users could use that to test what they want, which could provide huge assistance to testers since random testing is much easier to find the bugs. Since I test the treap data structure, which only need the list and value, so it helps me a lot and does not lead to any drawback. However, some of my classmates who focus on the string testing or language testing meet some problems when using randomtester. Because of randomtester will generate data randomly, which will lead to one problem: users cannot control the order of the strings to create the string which they want. This drawbacks cause lots of troubles for them.

Good with TSTL

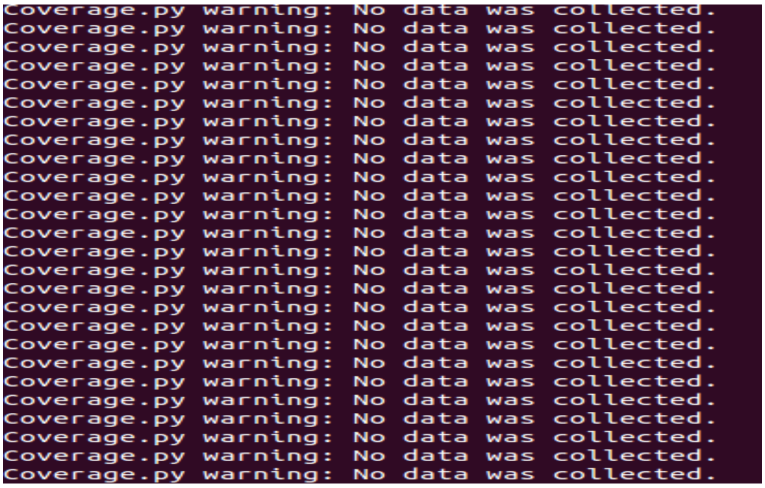
As far as I am concerned, the most useful part of TSTL is it could creating bunch of random data to help user testing, which will save lots of time. If testers do not use TSTL to test a program, they need to generate different kinds of random data manually, and sometimes, those random data is not really random. However, if testers choose to use TSTL, it could help to generate random data very fast.

Bad with TSTL

For the bad part, just as I mentioned above, I did not find any drawbacks from TSTL, since I test treap data structure which only need to consider Integer and List. However, some of my classmates who want to test string coverage or language similar percentage, they met some troubles when using TSTL. Even though, TSTL could generate strings randomly, users cannot control the order of the string. Therefore, when users want to create the string, they will meet some troubles due to the different order.

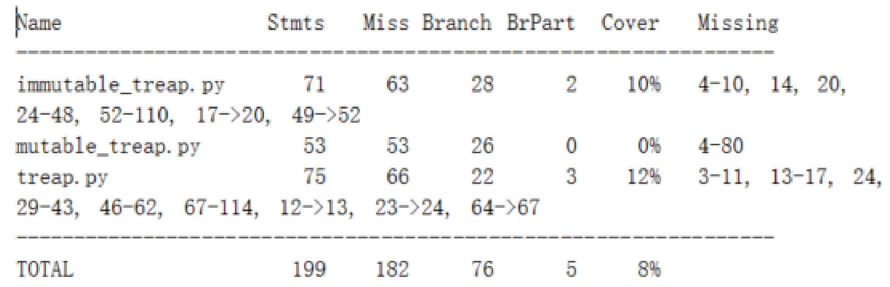
Coverage:

For the coverage, I implement TSTL during Linux, and TSTL always return the warning like below:



```
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
Coverage.py warning: No data was collected.
```

For the coverage.out, it shows like that:



```
Name                     Stmts    Miss Branch BrPart  Cover   Missing
---------------------------------------------------------------------
immutable_treap.py         71      63     28      2    10%    4-10,  14,  20,
24-48,  52-110,  17->20,  49->52
mutable_treap.py           53      53     26      0     0%    4-80
treap.py                   75      66     22      3    12%    3-11,  13-17,  24,
29-43,  46-62,  67-114,  12->13,  23->24,  64->67
---------------------------------------------------------------------
TOTAL                     199     182     76      5     8%
```

For the mutable_treap.py, it always shows the cover is 0%, even though I delete one of the function which I want to test in this file. I think the reason because this might be something wrong after TSTL detect the treap data structure.

Summary:

For the summary, TSTL is a quite useful debug application to help testers to find potential bugs in their programs. Even though, for the coverage part, it has some problems and might lead to order out of control by randomtester, it still could provide lots of assistance. From this project, I learn a lot about TSTL and better understand how to test a python program.