

Part 4: Final Report on Project

Xinran Peng

CS562w16

Prof Groce Alex

2016/3/14

0. Abstract

This is the final report of my project in CS562 about coding test using TSTL. In this report, I will mainly talk about the library I found and what I have tested so far, the bugs I found, and evaluate the tester, and give some coverage summaries.

In this project, I tested a B Tree library finding from <https://pypi.python.org/pypi/BTrees/4.1.4>. This library is a very large library, there are five modules and four data structure for each module. In my test, I tested the OOBTree module and BTree data structure. I generated several B Trees and some strings including characters, numbers, and some symbols as keys and values. And using these keys and values to test some functions of the data structure to see if the functions operate correct or not.

1. What did you accomplish?

Because I am not very familiar with python, it is kind of hard for me to read the code I found. And I have accomplished three functions after I understand the code in my library. Compared the first version of my tssl code, I added some new functions and rewrite some of the functions. They are insert, pop, and contain. In the insert function, I check if the tree is not empty, and then check if the length of the tree has increase by 1 and if the tree has the key I just insert. In the pop function, I check if the tree is not empty and then check if the length of the tree has decrease by 1 and if the tree still has the key I just pop. And in the contains function, I check what the function will return if the value is in the tree or deleted from the tree or removed from the tree.

2. Did you find bugs?

So far, I have tested many functions of the library but haven't found any bugs.

I used characters, numbers and some symbols as my argument, and tested both strings and ints. I checked the output of each operation and didn't find any bugs. I think this is because this is a very good library. This library is written a few years ago and the developer updates it quite frequently. Many users use this library and I think most of the bugs have been found and repaired during these years.

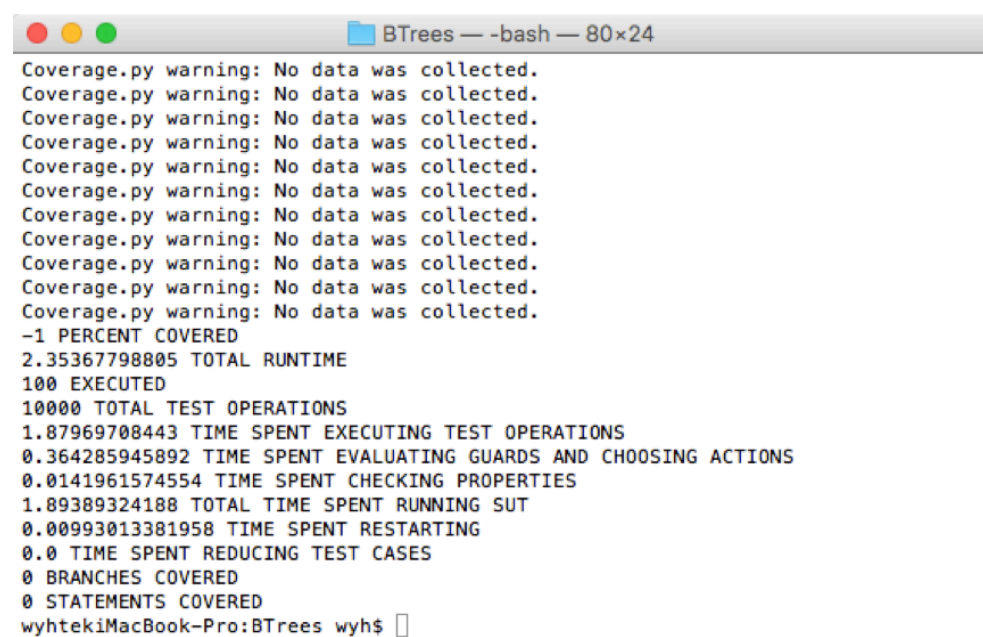
3. How well does the tester work?

At the beginning of using TSTL, it was very hard to understand it. There is not many books or introductions I can read and it is really difficult to start writing it. After reading and learning some test of AVL Tree posted online, I began to try to write some code of my tester. I think this tester does a well job. But maybe I cannot say so because I couldn't find any bugs in the code. Besides, there are still some of the parts I haven't tested, for example, the running time of each function. So I think this tester does a good job but not covered everything.

4. What is wrong with TSTL?

I think the most important thing TSTL need is a hand-book inducing how to use it with some code examples. Sometimes a little mistakes will stop me from continue the test and I just cannot find what is going wrong.

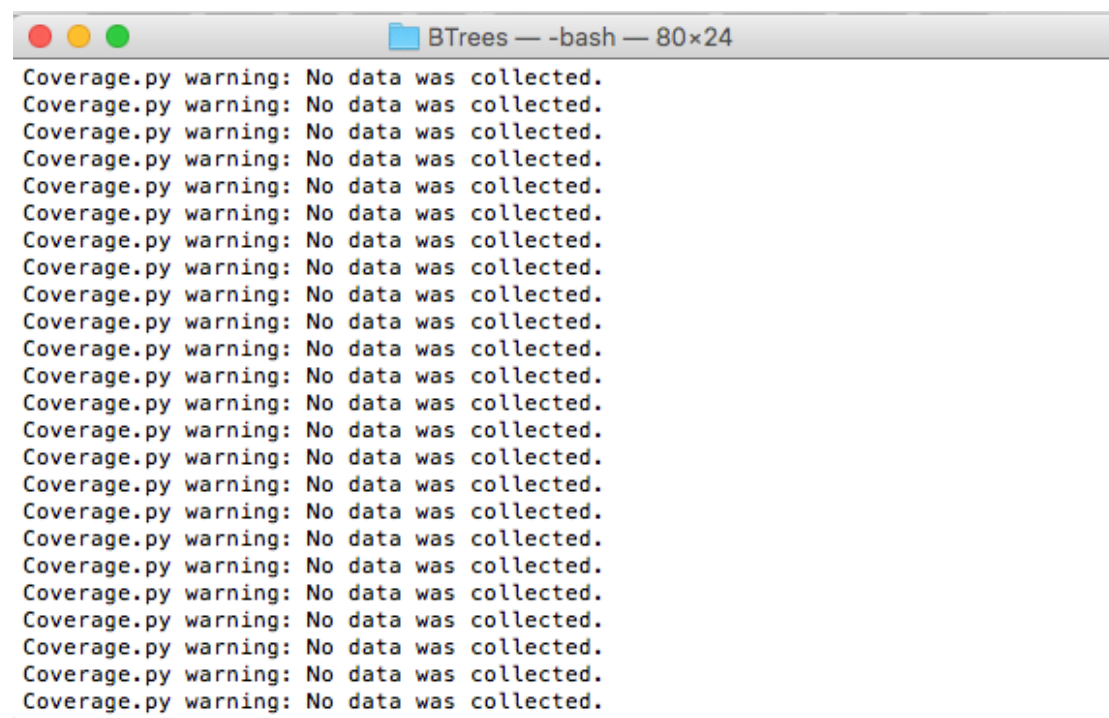
Besides, I had a strange output once I tried to get the coverage.

A terminal window titled "BTrees — -bash — 80x24" showing the output of a coverage command. The output consists of multiple warning messages, followed by summary statistics for test execution and coverage.

```
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.
-1 PERCENT COVERED
2.35367798805 TOTAL RUNTIME
100 EXECUTED
10000 TOTAL TEST OPERATIONS
1.87969708443 TIME SPENT EXECUTING TEST OPERATIONS
0.364285945892 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
0.0141961574554 TIME SPENT CHECKING PROPERTIES
1.89389324188 TOTAL TIME SPENT RUNNING SUT
0.00993013381958 TIME SPENT RESTARTING
0.0 TIME SPENT REDUCING TEST CASES
0 BRANCHES COVERED
0 STATEMENTS COVERED
wyhtekiMacBook-Pro:BTrees wyh$
```

I use the command “python randomtester.py -m 100” to check the coverage but the result turned out to be -1 percent covered, and got 0 branches and statements covered. I have totally no idea why this happened since I used to get a 5% coverage using the same code. And when we use the command “python randomtester.py -nocover” and quit when the printer is running, sometimes it will print like the coverage is 70% or something, but of course I don’t have that large coverage. I don’t have the picture of this because it just happened for one or two times. This really confused me when I trying to hand out my last time homework.

Another thing I think TSTL can improve is that when we didn’t find some bugs, TSTL will continually print. This is kind of annoyed.



```
BTrees — -bash — 80x24
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.
```

When we don’t define the numbers of we want to run the randomtester, I think this is okay. But when we define the numbers, I think it is better if TSTL can just output if there is anything wrong.

Besides, I think it will be better if TSTL can support some other widespread used language such as C++, Java, and C#.

5. What is good about TSTL?

I think TSTL is a good tool and is very useful and powerful. The best thing about TSTL is that I can use the random tester to run the test for the times I want

and using the random argument I defined. This is very helpful since we don't need to write a very long code to test everything we want.

Also, TSTL is very easy to install, and once we get familiar with TSTL, it is easy to use. The syntax of it is very clear and easy to understand, for example, the "pools", "logs", and "actions" syntax. This also makes the code very clear and easy to read.

The SUT file is another good thing generated by TSTL. Testers do not need to write bunches of hardness. This makes the test more efficiency and can also decrease the mistakes made by tester.

I think TSTL will be a very useful tool once it is well developed. And I hope I can use it to test in my work, not just for a project.

6. Coverage summaries, any other important data on the test effort.

Because I didn't test much function in the library and the library is very large, the coverage is just 12%. I want to improve this number in the future when I understand more about my library. But it is kind of hard for me now.