Student ID : #932262940 Name : Xuan He

Final Report of Projec

I have finish testing the BTree library as my plan. The “BTrees” library can be found in https://pypi.python.org/pypi/BTrees/4.1.4#downloads. There are five modules (OOBTree, IOBTree, OIBTree, IIBTree, and IFBTree), and for each modul there are four data structures (BTree, Bucket, TreeSet, and Set) in the library. My test just focuses on the OOBTree module and BTree data structure.

Accomplishment

Finally, I tested 5 function in the BTree library.

1. insert(key, value) : this function insert a key and value into the btree, If the key was already in the btree, then there is no change and 0 is returned. If the key was not already in the btree, then the item is added and 1 is returned. I use many different types of key and value to test this function, such as int, string, pair and btree. I also check whether the length of the btree is increased by 1 after insert operation (len(<btree,1>) == pre<(len(<btree,1>))> + 1) or the key that I want to insert already exists in the btree (<btree,1>.has_key(<key,1>) > 0) .

2. update ({<key>:<value>}): this function add or update the items from the given collection object to the B Tree. I use different types of 2-tuples(key, value) as input. I check whether the length of the btree is increased by 1 after update operation or the length of the btree does not change after update operation (len(<btree,1>) == pre<(len(<btree,1>))>), and I also define a new python function to check whether the data of the btree is updated after update operation (check_update(<btree,1>,<key,1>,<value,1>) == 1).

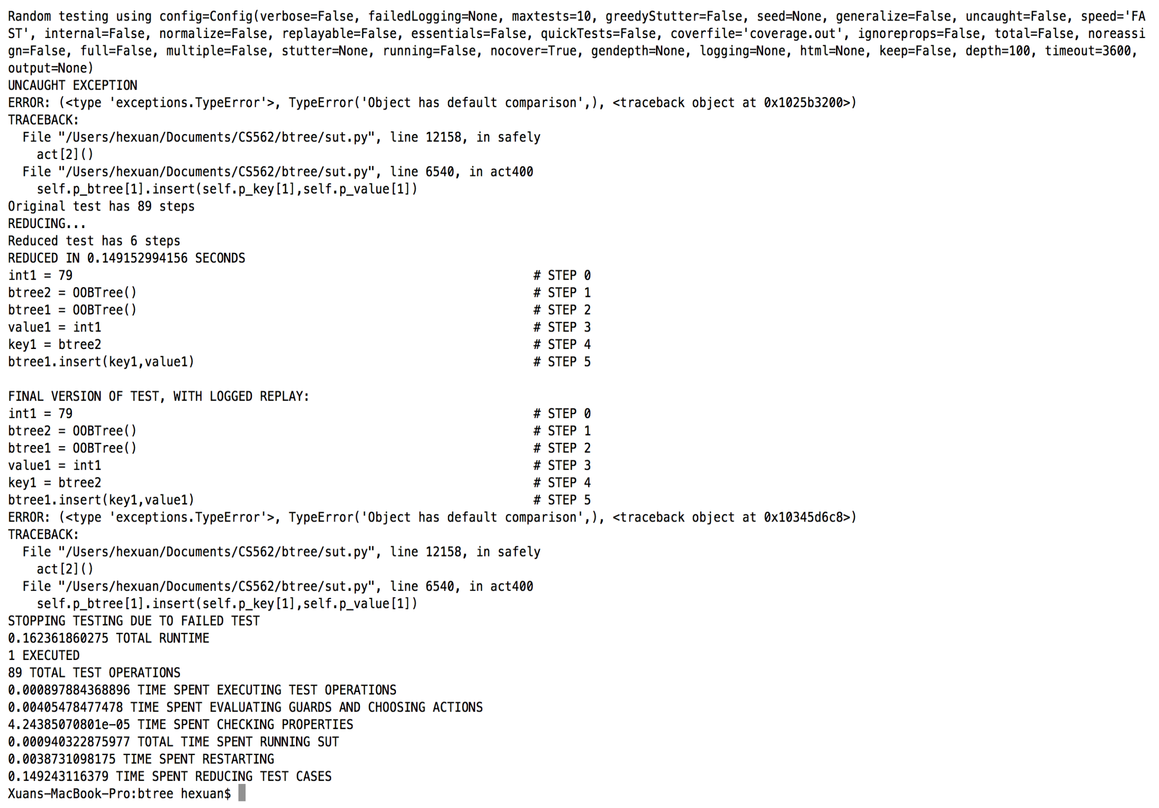
3. pop(<key>,0): this function remove key from the btree and return the corresponding value. If key is not found in the btree, 0 is returned. I use different types of key as input. I check whether the length of the btree is decreased by 1 after pop operation or the btree does no contain the key that I want to delete (<btree,1>.pop(<key,1>,0) == 0), and I also check whether the btree contains the key that is the input of the pop function after the pop operation (<btree,1>.has_key(<key,1>) == 0).

4. maxKey(): this function return the maximum key in the btree. I used different btree as the input, so I insure that the btree is non-empty(~<btree> != None and <btree,1>.__nonzero__() == True) before testing this function. In order to check its correctness, I create a python function to check whether the return value is the maximum key int the btree(check_max(<btree,1>,<maxKey,1>) == 1).

5. minKey(): this function return the minimum key in the btree. Similar with maxKey(), I check the btree is non-empty before testing this function, and I also create another python function to check its correctness(check_min(<btree,1>,<minKey,1>) == 1).

Bugs

For the btree library, I found one bug. When I use btree as key, the insert, update, and pop operations do not work. It shows a type error that Object has default comparison. The result is shown as follow.



```
Random testing using config=Config(verbose=False, failedLogging=None, maxtests=10, greedyStutter=False, seed=None, generalize=False, uncaught=False, speed='FA
ST', internal=False, normalize=False, replayable=False, essentials=False, quickTests=False, coverfile='coverage.out', ignoreprops=False, total=False, noreassi
gn=False, full=False, multiple=False, stutter=None, running=False, nocover=True, gendepth=None, logging=None, html=None, keep=False, depth=100, timeout=3600,
output=None)
UNCAUGHT EXCEPTION
ERROR: (<type 'exceptions.TypeError'>, TypeError('Object has default comparison',), <traceback object at 0x1025b3200>)
TRACEBACK:
  File "/Users/hexuan/Documents/CS562/btree/sut.py", line 12158, in safely
    act[2]()
  File "/Users/hexuan/Documents/CS562/btree/sut.py", line 6540, in act400
    self.p_btree[1].insert(self.p_key[1],self.p_value[1])
Original test has 89 steps
REDUCING...
Reduced test has 6 steps
REDUCED IN 0.149152994156 SECONDS
int1 = 79                                                          # STEP 0
btree2 = OOBTree()                                                 # STEP 1
btree1 = OOBTree()                                                 # STEP 2
value1 = int1                                                      # STEP 3
key1 = btree2                                                      # STEP 4
btree1.insert(key1,value1)                                         # STEP 5

FINAL VERSION OF TEST, WITH LOGGED REPLAY:
int1 = 79                                                          # STEP 0
btree2 = OOBTree()                                                 # STEP 1
btree1 = OOBTree()                                                 # STEP 2
value1 = int1                                                      # STEP 3
key1 = btree2                                                      # STEP 4
btree1.insert(key1,value1)                                         # STEP 5
ERROR: (<type 'exceptions.TypeError'>, TypeError('Object has default comparison',), <traceback object at 0x10345d6c8>)
TRACEBACK:
  File "/Users/hexuan/Documents/CS562/btree/sut.py", line 12158, in safely
    act[2]()
  File "/Users/hexuan/Documents/CS562/btree/sut.py", line 6540, in act400
    self.p_btree[1].insert(self.p_key[1],self.p_value[1])
STOPPING TESTING DUE TO FAILED TEST
0.162361860275 TOTAL RUNTIME
1 EXECUTED
89 TOTAL TEST OPERATIONS
0.000897884368896 TIME SPENT EXECUTING TEST OPERATIONS
0.00405478477478 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
4.24385070801e-05 TIME SPENT CHECKING PROPERTIES
0.000940322875977 TOTAL TIME SPENT RUNNING SUT
0.0038731098175 TIME SPENT RESTARTING
0.149243116379 TIME SPENT REDUCING TEST CASES
Xuans-MacBook-Pro:btree hexuan$
```

Because the module I use is OOBTree, which means the types of the keys and values can be any object, the btree should work.

Taster’s work

The tester does a very good work. The random-tester can generate a mount of random case and execute them. When it found bugs, it can reduce a lot of test case that could not generate the bug, which let me find the bug easier and faster. What I need to do is just providing the range and the form for the random data. I also can add some property to the test case in order to check its correctness. The random-tester not only help me find the bug, but also can show the process of the test by using log, which is very helpful for me to find how the bug generates. Overall, the random-tester is a very good tester.

Wrong with TSTL

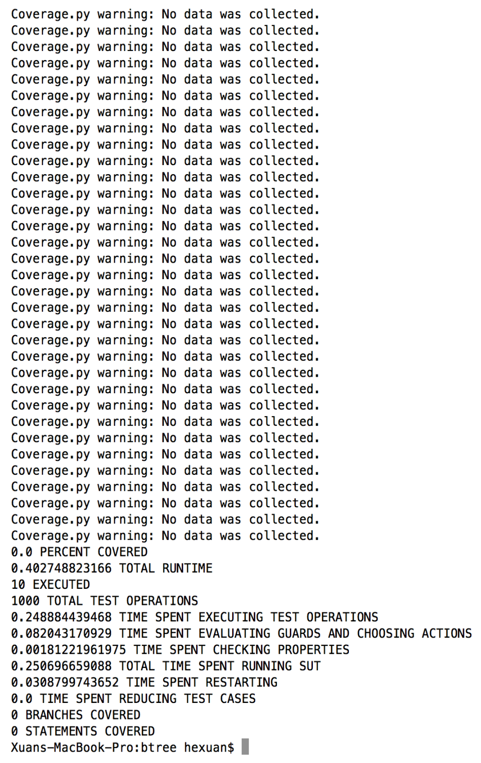
TSTL is a very good and convenient program and language, but it is very hard to learn. We have too little information about it. We cannot get help from the Internet. We just have two paper and some examples, which is hard to understand for the people who first use TSTL to test program. Therefore, I thing a good API is very useful for fresh. In addition, the program language that TSTL support is too little. In future, it should support more program language, such as c, c++.

Good with TSTL

TSTL is very easy to use. We just need to write little code, such as the type of the data, the number of the data, the function I want to test and so on, then it can generate the whole test file. It can save a lot of time for testing. Writing a test file is not a very easy thing. We should consider a lot of condition and branch. By using TSTL, we don't need to worry about that, and all the things is done by TSTL. We just provide data. It is wonderful and convenient.

Coverage

The Btree library is a very big library. There are five modules, and for each module there are four data structures. My test just focuses on the OOBTree module and BTree data structure. In this part, it also contains many functions. I select 5 main functions to test, which coverage about 30 percent of the whole functions in this part. But when I test the coverage percent of my test, the TSTL tell me it is 0 percent. I don’t know why it happens. I guess maybe something is wrong for importing the module, because it sometimes shows that the module I test does not import. The result of my test is shown as follow:



```
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.
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.
0.0 PERCENT COVERED
0.402748823166 TOTAL RUNTIME
10 EXECUTED
1000 TOTAL TEST OPERATIONS
0.248884439468 TIME SPENT EXECUTING TEST OPERATIONS
0.082043170929 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
0.00181221961975 TIME SPENT CHECKING PROPERTIES
0.250696659088 TOTAL TIME SPENT RUNNING SUT
0.0308799743652 TIME SPENT RESTARTING
0.0 TIME SPENT REDUCING TEST CASES
0 BRANCHES COVERED
0 STATEMENTS COVERED
Xuans-MacBook-Pro:btree hexuan$ 
```

I also do not understand the source target. Whether use it is no effect for the test, but when I deleted the source the covered percent changes to -1. I think the reason is that TSTL cannot find object to compare. Overall, the coverage test of TSTL does not work for me.