Course 569
Instructor: Alex Groce
Name: Zehuan Chen

project 2 report
The first tryout of random tester improved algorithm

Introduction

The algorithm I applied is base on the fundamental idea of random test, this algorithm could bring some improvement on current method. In random testing, we consider the system testing as a tree traversal problem, the test case are nodes in this big tree, the algorithms is not randomly pick up a operations or test case in a pool, but with deep traversal first instead. So the the Depth First Search Algorithm that we used to solve searching elements in tree structure is significan useful in this specific situation.

Algorithm

In general thinking, the random tester could keep working till it find the bug, the processing could also be infinity because the bug could be multiple.
The random source is derived from random.Random(), when every time the iteration is done, we shuffle the sequence of sut.state(), the stop condition could be vary, in my program, in order to meet the course requirement, maximun time restrict, depth restrict and width restrict can be applied. When the time you set up is not run out, the testing goes down to the button of the tree using backtrack(), then when the current depth of the traversal bigger than the max depth, the testing iteration is over this round. Using the sut.safely(actions) and test if there are any bug against the restriction write in TSTL file.

Result

For this project, I using avl tree as my example, the sut.py is generated by tstl, when it runs it show the result like the chart show below.
The test library is an AVL tree with some bug in it called avlbug1.py, the tstl is avlefficient.tstl, which contains the pool of test case qualification.

```
bill@bill-laptop:~/CS569/cs569sp16-master/SUTs$ python tester1.py 40 1 100 1 1 1
 1
use0.0151119232178
FOUND A FAILURE
```

Figure 1, the input of tester, 40 1 100 1 1 1 1 indicates that take 40 seconds maximum time budget to run, the random seed is 1, length 100, width 1, report for failure, internal coverage report and new coverage branches discovered while running.

Conclussion

This is a milstone of my project this term, but I think it still have a lot of problems, the algortihm of feedback-directedd is still hard to implement it all, but the dfs way is the first step to the final, this method shows a good performance that can find bug very quickly, the sample of avlbug.py can be detected.

```
FOUND A FAILURE
(<type 'exceptions.AssertionError'>, AssertionError(), <traceback object at 0x7f
9c4a25c5f0>)
REDUCING
val1 = 14                                                          # STEP
0
avl0 = avlbug1.AVLTree()                                           # STEP
1
avl0.insert(val1)                                                  # STEP
2
val3 = 5                                                           # STEP
3
val1 = 12                                                          # STEP
4
avl0.insert(val3)                                                  # STEP
5
avl0.insert(val1)                                                  # STEP
6
this is fault:(<type 'exceptions.AssertionError'>, AssertionError(), <traceback
object at 0x7f9c4a25c5f0>)
```

Figure 2, the test cases that tester ran and the failure report

```
TSTL BRANCH COUNT: 171
TSTL STATEMENT COUNT: 127
TOTAL NUMBER OF BUGS 3
```

Figure 3, the final result report

So, the final report shows that the branch count as 171, statement count 127, find 3 bug eventually. The time comsuming is pretty short.

This result is not performance good, because it could be more bugs shown in this time budget, but the benefit of this algorithm is directly and simple to implement, but the drawback is also obviously, this algorithm is too simple that could not improve the performance very much, so it absolutely need to improve it again.

For the future work, I need try the algorithm that I choose at the first beginning, the feedback-directed random testing algorithm. This algorithm could reduce the time of generate the test cases more efficient instead of in chaotic way, the test routine will be more precise and accurate, so it may detect the bug in shorter period of time.