Name: Hengrui Guo

Student ID: 932504737

Date: May 5, 2016

Prof. Alex Groce

Project Milestone

Introduction：

In this project, I would like to implement the Adaptive Random test method in TSTL for this project. The random testing method is a black-box software testing technique [3]. So it will use a large number of test cases to test the programs. So I determine to implement the new generation based on the random test generation with some modifications. The Adaptive Random test generation's main idea it try to distribute test cases more evenly within the input space. It is based on the intuition that for non-point types of failure patterns, an even spread of test cases is more likely to detect failures using fewer test cases than ordinary random testing[1]. Experiments are performed using published programs. Results show that adaptive random testing does outperform ordinary random testing significantly. In other words, the distance from the selected test case in candidate set to every test cases in executed set is larger than distance from any other test cases in candidate set to every test cases in executed set. In this algorithm, the adaptive technique can reveal the failure quickly than the ordinary random test method.

Algorithm description：

Step 1. initilize
Step 2. init sut.sut
Step 3. init random.Random()
Step 4. init time.time()
Step 5. set variable depth, explore and savecoverage_test

Step 6. set parsing parameter: BUDGET, seed, width, faults, coverage ,running
Step 7. generate random STATEMENT
Step 8. collect random statemtnt
Step 9. caculate data coverage
Step 10. set coverage tolerance
Step 11. collect the statement lower than the tolerance.

Generation Testing:

Before test our generation, we should collect three files in one directory, which are tester.py, sut.py and AVL.py. The input the comment that shows below is used to compile the python file.
Command : python tester1.py 30 1 100 1 0 1 1
In this comment, there are 7 parameters and each of them has different meaning. The first one represents the running time. The second represents the seed, python random. Random objects used for random number generation code. The third parameter, which is 100 indicates the maximum length of the test generation. The next, which is 1, represents the maximum memory is basically a search width. The next three parameters can be set to 0 or 1. The following two parameters means that the final report will generate coverage and branch coverage will be generated

Improvement:

I have modified some code in the tester1.py and improved the algorithm. As a result, the new tester2.py have more branches and statements. Meanwhile, it can find more bugs.
The parameter I used: 40 1 100 1 1 1 1
Result:
(1) tester1.py:

```
, 219, 220, 221, 222, 223, 225, 227, 246, 247, 249, 250, 251, 252, 254, 258, 259
, 262, 263, 264, 267, 268, 270, 271, 272, 273, 275, 277, 278, 279, 281]
TSTL BRANCH COUNT: 185
TSTL STATEMENT COUNT: 139
(u'/Users/HengruiGuo/Desktop/SUTs/avl.py', 91) 20
```

(2) tester2.py:

```
TSTL BRANCH COUNT: 191
TSTL STATEMENT COUNT: 141
(u'/Users/HengruiGuo/Desktop/SUTs/avl.py', 254) 5
(u'/Users/HengruiGuo/Desktop/SUTs/avl.py', 93) 9
```

Reference

[1] Chan, F.T., Chen, T.Y., Mak, I.K., Yu, Y.T.: Proportional sampling strategy: guidelines for software testing practitioners. Information and Software Technology 38 (1996) 775–782

[2] T.Y. Chen, H. Leung, and I.K Mak. Adaptive Random Testing, 2004

[3] A. Groce, G. Holzmann, and R. Joshi. Randomized differential testing as a prelude to formal verification. *Software Engineering,* 2007. ICSE 2007. 29th International Conference on, pages 621-631. IEEE, 2007.