Course: CS569
Name: Kai Shi
Project Proposal

Proposal For Testing Project

Introduction

When a program is finished, we need to test the functions of this program. Therefore, we need to use variables of test cases to test it. And it's important for tester to design test cases; maybe we can call it test suite. Test case is one execution of a program, and it can help tester to find a bug. I think random testing is a good way to test programs. Random testing is also called "fuzzing"[1]. But how to generate random test cases, that's a big problem. Because, random is not really random, if tester uses a huge amount of meaningless test cases, the efficiency will be low. So, testers need to come up with some algorithms to generate effective test cases to test programs. That's the test generation. The algorithm testers come up with should be able to cover more branches and use more effective random data but with minimization of test cases. In my opinion, to generate minimization of effective test cases is test generation, no matter using manual testing or automated testing. In the class, professor introduces random test method, DFS, BFS checker algorithms to cover most branches. These two algorithms like to traversal tree or graph; so, they have high efficiency and high coverage of branches.

New Test Generation Algorithm (FAR)

And different from these testing methods, I plan to use a new test method Fast Antirandom (FAR)[2]. This method is presented in the paper Fast Antirandom (FAR) Test Generation, and the authors are Tom Chen, Amjad Hajjar, and Andre Bai. They present and empirically evaluate a technique to generate anti-random vectors that is computationally feasible for large input vectors and long sequences of tests. The authors also claim this algorithm has high coverage of branches and the most advantage is this algorithm has high efficiency. The anti-random test vector is

defined as a vector with maximum distance from all previous vectors, which were applied during a test. The Fast Anti-Random (FAR) approach generates new test sequences by centralizing all existing input test vectors into one test vector. Then, it finds an anti-random vector with maximum distance from the centroid vector. FAR can be used by following three steps: Centralization, Doing vector calculation, and maximum distance calculation. In the paper, authors present the results of FAR's branches coverage. It can cover more branches than Random test generation in the same time. At the first step of this algorithm is similar to the random testing, but referring to the sequence of testing, it doesn't use the random way. It calculates a new way by treating existing testing sequences as vectors. Therefore, it has higher efficiency than random testing. I plan to use this algorithm to generate test cases.

Project Plan

First step is reading some random testing papers to figure out what is the sequence of the random testing. After knowing about existing testing sequence, I can calculate new sequence by the algorithm. Also, I need to know about how randomtester of TSTL works.

The next step is reading the example given in this paper. I think this example can give me a direction about how to calculate the new testing sequence rather than using random sequence.

Then, using TSTL API in this algorithm. And I plan to use it to test AVL tree (with bug).

References

[1] W.Howden,"SystemsTestingandStatisticalTestDataCov- erage", COMPSAC '97, Washington, DC, August 1997, p. 500-504.

[2] C. Tom, B. Andre, H. Amjad, "Fast Antirandom (FAR) Test Generation", Procs. Quality Week 98, May 1998, San Francisco