Course: CS569
Name: Yifan Shen
Content: competitive milestone 1

Introduction

The first version of the implementation of the test is based on the depth first search and it can find the bugs in avlbug1 efficiently for the bug in avlbug1 has a very special trigger mechanism which is easy for the depth first search to find. Depth first search is an algorithm for traversing or searching tree data structures. We usually randomly select some arbitrary node as the root and then explore the node as far as possible along each branch before backtracking. Depth first search is suitable for the situation that the data we need is along one branch.

Algorithm

In my code, I mainly use the functions mentioned in the bfs algorithm. In my code I use the random function to generate the random action sequence by using the rgen.shuffle. To keep the information of the actions, I use the S to keep the state and test content. We keep on checking the correctness of the action sequence when the S "pool" is not vacant. Each time, we just pop a state from the S "pool" and use the backtrack function to recover the state. To make the depth first search algorithm efficient, we check the state to see whether we have meet the state before and whether the length of the test exceeds the max depth we set. If not, we will use the enabled function to get all the actions that we can operate and use the rgen.shuffle to make the action sequence random and we use the sut.safely to check the correctness. The state will change when we use the safely function and we record this new state in the S "pool" to make the algorithm do the next cycle.

Result

To test the code, I use the avlbug1 as an example bug code. And use the avlefficient.tstl to compile the avlbug1. And finally we import the sut into my own dfs.py.
Compared with the bfs to use minutest to find the bug in the avlbug1, the dfs algorithm could find the bug very quickly. And I attach one of the bug that the dfs method find:

avl0 = avlbug1.AVLTree()	# STEP 0
val3 = 16	# STEP 1
val0 = 20	# STEP 2
avl0.insert(val0)	# STEP 3
Val0 = 14	# STEP 4
avl0.insert(val0)	# STEP 5
avl0.insert(val3)	# STEP 6
avl0.insert(val2)	# STEP 7

And the algorithm find the first bug just use 0.0211079120636 seconds. And I use the bfsnovisited method to find the bug and it uses 164.36986208 seconds to find the bug. From the comparison of time the bfs and dfs methods use, we can find that the efficiency of dfs is much higher than the bfs.

Discussion

I think the there are still several problems in my current code.
Firstly, we find that although the code can find many bugs but all these bugs are in the same pattern. Three insert will cause the bug. But the current algorithm could not find the pattern and it outputs lots of useless bug information. I think we can do some work in this part to make it wiser.
Secondly, we find the coverage by the dfs is not very high. Some other work will be done in the future to make up the disadvantage.