**Course: CS569**
**Name: Yifan Shen**
**Content: competitive milestone 2**

## *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*

By using the part2 code, I can find some bugs. But the number of bugs that the code finds is few. I think it has two reasons. Firstly, the program just run one time, the dfs search just run for a time. So it could only find the bugs during a very short period. Secondly, the code uses the visited pool. When some states can be find in the pool, the code will skip this branch. But in fact , many different bugs could be find in the same or similar states. So I am going to do one work is

to collect the test and states when the tester find bugs and then run these test and states in a random sequence. I think many more bugs could be find by this way.
In the figure below we can find the results are really different.
Part2 result:

```
TSTL BRANCH COUNT: 147
TSTL STATEMENT COUNT: 110
TOTAL NUMBER OF BUGS 4
188
```

part3 result:

```
TSTL INTERNAL COVERAGE REPORT:
/Users/shinsuguruhan/PycharmPr
/Users/shinsuguruhan/PycharmPr
TSTL BRANCH COUNT: 181
TSTL STATEMENT COUNT: 130
TOTAL NUMBER OF BUGS 1103
```

## *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.