CS569

Yi-Chiao, Yang
932526065

Competitive milestone 2

In my milestone 2, the whole idea of my algorithm does not change, but I added more details in order to collect low coverage of all statements. In previous works, we tested our python code to know how many new branch statements and total statements it discovers. In these tests, I found that I did not have high coverage. For this weakness, I attempt to modify my algorithm and figure out why I have low coverage in my code.

My algorithm is related to Breadth Fast Search algorithm. The idea of Breadth Fast Search algorithm is to start a node to visit following its adjacent neighbors, and visits these adjacent neighbors until the node visits all of nodes in the pattern. Because we have the time limitation, how many adjacent neighbors nodes the node can visit it depends on how much time it can use. This algorithm takes $O(b^{d+1})$. b is the branching factor of the graph and the d is the distance from the start node. In my test, the depth is 9 when I use 30 seconds, and the depth is not great. In addition, I seldom find bugs when I use the time around 30 seconds to 50 seconds. That means that my algorithm is not efficiency.

For the improvement, I would like to design another section to test low coverage branches, and after collecting these low coverage branches, I might focus on running these parts. This section might increase the overall coverage of SUT. Fist, I use Breadth Fast Search algorithm to travel all statements to find bugs. Second, I have a function to separate all statements into two parts, including lower coverage and high coverage through the median coverage, but in this function, it uses an approximate median coverage. Third, statements have low coverage that means they do not be visited by system many times. This low coverage might contain software bugs, so I spend more time testing this part.

In my current python code, it has not finished yet. I am still dealing with many challenges in order to make my test generation more efficient. I have already collected statements that have low coverage, but I have not tested them yet. I cited many tasks on my current python code because these parts are unfinished.

I consider that what kind of method I can test them. If I still test tem through Breadth fast Search algorithm, it might also waste time. If I use a new method to test these low coverage statements, I need to spend more time designing this new method. A program with high code coverage has more probability to find software bugs. Thus, my future works will focus on improving the code coverage. If I have enough time, I might test the statements that they have high coverage more times. Generally, these statements have already testes many times, but they might have potential failures. Whatever, the most important thing is to enhance the efficiency of finding software bugs.