Software Testing 2022

S1833951 li fang

January 26 2022

3.1 Range of techniques

R1:

- Mutation test: I had changed the variable in the function, using null value for input, mixing the variable names to see if test still fail.
- Functional test: I test the function by comparing the expected result to the result provide by the function, and try to stimulate the real-world scenario that would happened
- Systematic test: I use the generated data to conduct the test, as the generated data is different from the real data, I generated them to fit different scenario, such as let the test line cross different restricted area so we know it works on different subject.
- Unit test: the test is design for a single function only.

R2:

- Mutation test: I had changed the variable in the function, using null value for input, mixing the variable names to see if test still fail.
- Systematic test: I use the generated data to conduct the test, I select the point in different direction so make sure all circumstance is covered.
- Functional test: the test is design for checking the function does achieve its expected result, which is find the nearest point for a point among a list of point.
- System test: the test does check on multiple function, checking the overall result is acceptable.
- Unit test: have an individual unit test for dis function, so easier to know which part of the code go wrong.

3.2 Evaluation criteria for the adequacy of the testing

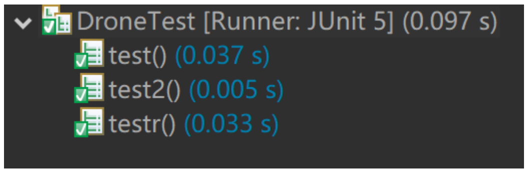
R1:

- I use generated data to test the function, since it is generated so I can cover the situation that may not covered by real data, such as testing each the side of the restricted area is not crossed, while in real world the whole path may only interact with one of the ides.
- Is it adequate as Unit test is suitable for testing single function like this one, it is fast and easily structured.
- The limitation of the testing method is that I use cross to check if the path does not get out the area. But if the input data point is outside the area, then the test will fail always, the test assumes all input data of the function is legitimate.

R2:

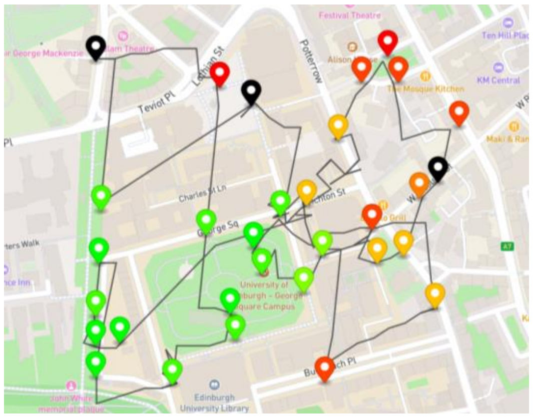
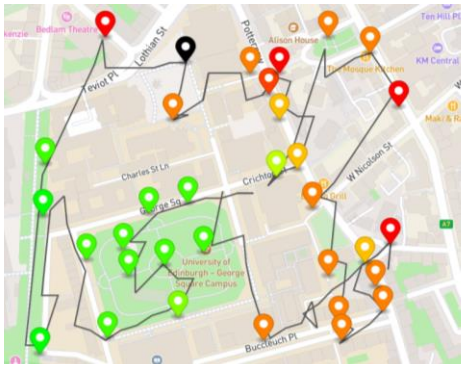
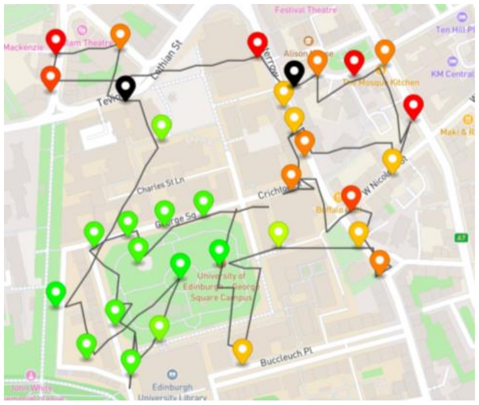
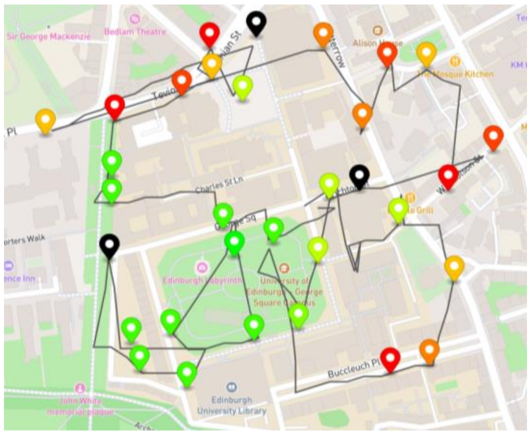
- The system test use for the R2 because it contains 2 functions at the same time, so we should use the system so it covers the general system of this sub branch.
- I use generated data, so it covers point from all directions, but it may be far lesser in number of points compare to the real circumstance
- Unit test for dis function is good, as measuring performance of a single math function is suitable, help to easily located the bug in the code.

3.3 Results of testing & evaluation

The test is all passed 

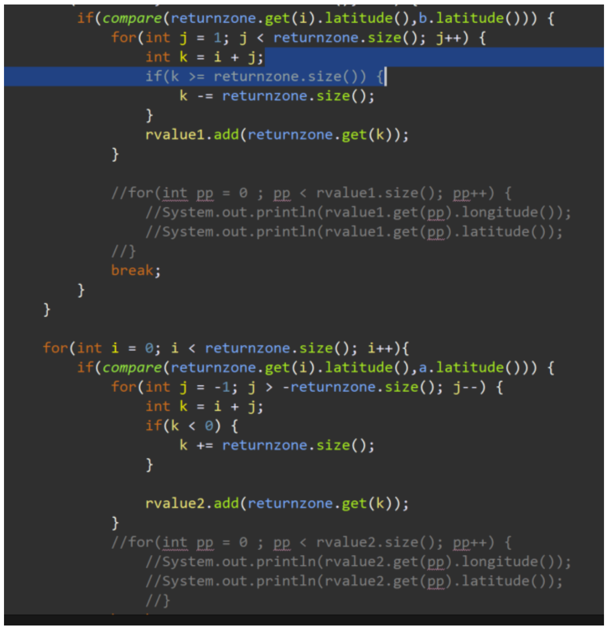
```
DroneTest [Runner: JUnit 5] (0.097 s)
    test() (0.037 s)
    test2() (0.005 s)
    testr() (0.033 s)
```

And comparing to the overall outcome, it does achieve the purposes:

None of the point exceed the restricted area, and none of the pass choose the route not based on nearest-point rule. So the test result is coherent with the system performance

Well, the result shows the test is efficiently showing the system sircumstances, but that builds on the fact that all the data feed in is correct. The test and system both does not check if the input data is legitimate. There would be security issue here.



```
    if(compare(returnzone.get(i).latitude(),b.latitude())) {
        for(int j = 1; j < returnzone.size(); j++) {
            int k = i + j;
            if(k >= returnzone.size()) {
                k -= returnzone.size();
            }
            rvalue1.add(returnzone.get(k));
        }

        //for(int pp = 0 ; pp < rvalue1.size(); pp++) {
            //System.out.println(rvalue1.get(pp).longitude());
            //System.out.println(rvalue1.get(pp).latitude());
        //}
        break;
    }
}

for(int i = 0; i < returnzone.size(); i++){
    if(compare(returnzone.get(i).latitude(),a.latitude())) {
        for(int j = -1; j > -returnzone.size(); j--) {
            int k = i + j;
            if(k < 0) {
                k += returnzone.size();
            }

            rvalue2.add(returnzone.get(k));
        }
        //for(int pp = 0 ; pp < rvalue2.size(); pp++) {
            //System.out.println(rvalue2.get(pp).longitude());
            //System.out.println(rvalue2.get(pp).latitude());
            //}
```

and the scaffolding is quite useful during the developing phase of the project, as it shows the states directly allowing me to fix the bug quicker and immediately, not after the function is generally done.