Bug / Progress Report

Zixuan Zhao

In my project, I mainly test a function which is sorting dictionary. There are two different codes used to be tested. One is mainly written by C and another one is pure Python code. I have two goals in this testing. First one is to test if both codes can sort dictionary with variance elements correctly. Second is to test and compare the performance of both codes. I will explain my test in following aspects, the quality of testing codes including the the improved part and problematic part, coverage and my estimation for my project.

Testing Code

In the tstl files, I mainly create two types of testing codes. One is to generate the random dictionary. Another one is to test the function with the generated dictionary.

To generate the random dictionary, first is to create the element of dictionary. An element of dictionary consists of key and value. To make sure both sorting codes can deal with variance types of dictionary, I generate these combinations of key-value element:

1. Key: Numeric type; Value: Numeric type
2. Key: Numeric type; Value: String type
3. Key: String type; Value: Numeric type
4. Key: String type; Value: String type

From test, it can generate the element that I expect. However, there is one problem which is some special characters cannot be included in. I generate String by connecting character codes in ASCII. In this case, I expect that I can get some special symbols that we rarely use for normal working so that I could test whether these codes are good to support these special characters. But in the real testing, I realize it is a little difficult to generate these special characters. When I insert it into dictionary, these special words perform as the connected normal symbols. Until right now, I still cannot figure out how to solve it.

For the random dictionary, it can be generated as what I expected which is including all four combinations mentioned above. However, there are some repeated dictionaries that tested by several times. I am not sure if it is caused by random test from tstl or the random generator from Python. This does not affect a lot for my test. Interesting thing is even the same dictionary is tested repeated, the result varied a lot. I would like to collect data for it and compare the result to see how it changes.

Coverage

From the output of tstl test, it shows “3.23732444656 PERCENT COVERED”. In the coverage.out file, it also shows a comparably lower coverage. For code written with C, it shows 36% covered. For the pure python code, it is 10% covered. I think the reason may be the part I used to test occupies small amount of the whole code. If I add more functions, the coverage would be increased.

Estimation for My Project

To make it easy to test the performance, I just set pool as 1. In this case, each time it would just produce one same size dictionary at same time. On one hand, it is convenient. On another hand, it is hard to test whether these two sorting codes support all cases. So, until now, I did not find a bug from these two codes. Therefore, I may adjust my tstl testing code to adapt to these two test goals. Currently, I am collect the data and analysis the performs of these two codes so that I can see whether they can have similar abilities on sorting.