Proposal For Testing Project

For the Testing Project, I will use TSTL to test a Python library called **Algorithms**. Algorithms is a library of algorithms and data structures implemented in Python. This library contains a couple of pretty important data structures and sorting algorithms. In this system, there are totally eight modules can be tested: Math, Data Structures, Dynamic Programming, Factorization, Random, Searching, Shuffling and Sorting. Each of them are a submodule of Algorithms. This library can be found at https://github.com/nryoung/algorithms. Some of my previous programs has used this module for more conveniently dealing with data.

I will first test the implementation of data structure and whether those functions inside module can be robust to some strange input such as unexpected string or binary etc. And for a data structure, the most important thing is that whether they follow their properties. that is another part I will test for this data structure module.

The sorting module will be the second submodule to test. I will use TSTL to try different type of input to see if sorting algorithms are correctly sort the input or not. And probably I will use python build-in sorted function as reference to see if it gets the same result. If possible, the time complexity will also be taken into account for the correctness of sorting module in Algorithms.

For the Math submodule, each Math algorithm calculate a specific value for different input. In order to test this module, one of the choose will be to guard the result by the value that we expect to get. But this means we have to implement the completely same algorithm again to compare. That seems not reasonable. So instead of compare with our expected value, I will set a maximum limit to guard all results. Maximum limit will be computed by hand using the maximum value that I set in TSTL variable.

The Searching submodule has six search algorithms. For the correctness of this parts, I will use python “in” operate as reference to compare the result for list input. For non-list input such as tree or graph, the implementation of contains function can be used as reference for correctness of these searching algorithms.

There is only one algorithm for random module which is Mersenne Twister. The correctness of random algorithm is pretty blurred. Since this function does not have input, I will set a maximum random value guard for it to test the correctness.

The Factorization submodule takes a integer as input and return a list of factorization. For example, N = a*a-b*b = (a-b)*(a+b), the list will be returned as [a-b, a+b]. So for testing the correctness, I will first to see if return value is a list or not then compare (a-b)*(a+b) with the original input integer.

For other submodules such as shuffling or dynamic programming, there may be no way to test the correctness. I will think about a way to test the their correctness if there is. But basically, the main part of testing this system will be the data structures, sorting and searching algorithms. If time allows me to test more system, I will also try to test a couple of functions in Numpy.