Xu Zheng
932-509-227
January 26, 2016

CS562 Project Proposal

I. Introduction

This project is to test a third party Python library called "Algorithms" by using TSTL. "Algorithms", which gained more than 1300 stars on GitHub, is a library of algorithms and data structures implemented in Python. "Algorithms" includes 8 modules: Data Structures, Dynamic Programming, Factorization, Math, Random, Searching, Shuffling and Sorting. Due to the limit of this project, I decided focus on testing Data Structures and Sorting modules.

GitHub page of "Algorithms": https://github.com/nryoung/algorithms

Documentation of "Algorithms": http://algorithms.readthedocs.org/en/latest

II. Technical Details

Data Structures module includes several common data structures such as Binary Search Tree, Directed Graph, Queue, Stack, Linked List and so on. As for the details of testing all the data structures, I want to take the testing of Queue as an example. As we know, a queue is a First-In-First-Out data structure. It means that the first element added to the queue will be the first one to be removed. There are four function calls of Queue: *add(value), is_empty( ), remove( ) and size( ).* So that I can generate several inputs to see whether the outputs are right or not based on the understanding of queue data structure. After calling multiple functions calls, I will compare the actual outputs with the expected outputs to evaluate the correctness. All the test cases will be generated automatically by using TSTL. As for the rest of the data structure modules, I will use the same strategy but with some adjustments for different data structures.

Sorting module includes several common sorting algorithms such as Bubble Sort, Merge Sort, Heap Sort, Insertion Sort, Quick Sort and so on. As for the details of testing all the sorting algorithms, I want to take the testing of Heap Sort as an example. Before the sorting, we need to construct a max heap with Heap Sort based on the inputs. There are three function calls of Heap Sort: *build_heap(seq), max_heapify(seq, i, n), and sort(seq).* First of all, I will test the correctness of building the max heap. Then I will compare the actual

sorting outputs with the expected sorting outputs to evaluate the correctness of sorting algorithms. Beside this, I can use different sorting algorithms to deal with the same inputs since there are several sorting algorithms here. In this way, the correctness of different sorting algorithms could be tested in fewer inputs. All the test cases will be generated automatically by using TSTL.

III. Checkpoints

1. Initial version of testing system in TSTL
2. Bug/progress report
3. Final report
4. Final version of TSTL code

	Checkpoint 1	Checkpoint 2	Checkpoint 3	Checkpoint 4
February 15				
March 1st				
March 14				