Bug/Progress Report

Class: CS -562

Name: Akshay Salvi

Date: March 3, 2016

1. **Bugs found in the SUT**

   I have performed tests on the SUT and till now I haven't found any bugs in the file. I have tested multiple functions/methods in the SUT by using them in a single tstl test statement and haven't found any noticeable bugs in the functionality of the methods.

2. **Progress till date**

   Below are the tstl test cases that I have used in my project:

   2.1 `<btree>.put(<key>,<value>) => (<btree,1>.is_empty()== False)`

   In this tstl test statement I have used the put() method to put a <key,value> pair into the binary tree. Then I just tried to check if the current object on which the <key,value> pair was inserted is it empty or not . ie Just checking whether the <key,value> pair was inserted or not. So when the random tester generates values and populates the binary tree, the assertion works properly i.e is_empty() method returns a false.

   2.2 `<btree>.put(<key>,<value>) ; <btree,1>.contains(<key,1>) == True ; <btree,1>.delete(<key,1>) => (<btree,1>.contains(<key,1>) == False)`

   To check if multiple methods interact with each other, I tried the above tstl test case. In this case I inserted a <key,value> pair into the binary tree using the put() method. After that I used the contains () methods to check if the binary tree object contains that <key,value> pair and later went on to delete the <key,value> pair. For the post condition, since the <key,value> pair does not exist now, so the assertion is checked with "false". So when the random tester generates values and performs the above test, the assertion is satisfied successfully.

   2.3 `<btree>.put(<key>,<value>) ; ((<btree,1>._max_node().key) == (<btree,1>.max_key()))`

   Here I have inserted a <key,value> pair and used the max_node() function which gives the maximum node in the binary tree and accessed its "key" value. And then used the max_key()

function to get the “key” value. After the random tester generates values, the assertion is passed successfully.

2.4 `<btree>.put(<key>,<value>) ; ((<btree,1>._min_node().key) ==(<btree,1>.min_key()))`

Similar to the test case I have used above, I have here used the min_node().key and min_key() functions to access the minimum “key” value. Here when the random tester generates values, the assertion is successfully satisfied.

3. By the end of the term I will test all the functions in the binary_search_tree.py and the singly_linked_list.py library and check if all the functions in both the libraries are working properly. Till now I haven’t found any bugs but I will still be looking out for bugs in the libraries and hence check all the methods. I am now testing the libraries of linked list, stack and queue and by the time of final submission I will be able to submit test cases for these files as well.

4. The SUT under test has most of its methods working properly. I have used methods from the binary_search_tree.py library and till now I haven’t found any bugs. So the quality of the SUT is good till this point. The code is very well written and if similar code quality is maintained throughout the library then, there would not be much bugs existing in the library.