Project Proposal

He Zhang

zhangh7@oregonstate.edu

Test Object

B tree B-tree is a self-balancing tree data structure that keeps data sorted and allows searches, sequential access, insertions, and deletions in logarithmic time, which is a frequently-used data structure. B+ tree is an n-ary tree with a variable but often large number of children per node. In database management system class, we use B+ tree to store data efficiently. So it is useful and I am familiar with it. So I want to use TSTL to test pure-python B tree and B+ tree implementation. The source code is from https://gist.github.com/teepark/572734. Since this is not a standard python library, it is more likely to find some bugs during testing, which makes it a good choice for this project.
The file name is btree.py and programming language is Python 2.7. The Lines of Code of this file is 583, which contain 6 classes and 48 functions. The main body of this file is B tree and B+ tree initializing, inserting, deleting, presenting. It also contains two build-in unit test modules.

Test Criterion

Correctness:
It is obvious that btree.py should meet all the requirements of B tree and B+ tree.
For example, for B+ tree inserting, here is the right results. If the bucket is not full (at most b-1 entries after the insertion), add the record. Otherwise, split the bucket. If the root splits, create a new root which has one key and two pointers.
Also, the source code should not contains any functions that B tree or B+ tree does not have.
Coverage:
Since btree.py is not a big file, this test should be a coverage-based test. I would use TSTL to coverage most of the source code if it cannot reach 100%. If the coverage is not 100%, I need to analyze source code to see if it contains some unreachable branches.

Test Plan

This is a random test project by using TSTL. So what I need to do is analyze test object btree.py, write and execute tstl file, and report bugs I found in the source code. If something goes wrong with TSTL, it is also a useful information which should be recorded. This flow path should be iterative process. The testing result is an important feedback to modify tstl file.

Test Method

Equivalence Partitioning:

Partitioning inputs into valid parts and invalid parts, to see if btree.py can deal with different kinds of input sets. For example, the insert function is: insert(self, index, item, ancestors), first using some numbers as index, then using characters as index (not numbers).

Boundary Analysis: Focusing on boundary inputs, to see if btree.py can deal with boundary cases correctly. For example, when the tree is empty or just has one node, try to insert or delete.

Interface Test: Focusing on function calling, to see if the parameters are passing correctly.

Consistency Test: Executing some steps repeatedly, to see if it works well. For example, inserting and deleting several times.

Error guessing: Using some obvious error input, to see the robust of btree.py. For example, removing an empty B Tree, or inserting with same index.