Sean Penney
Project Proposal

I will be testing the bintrees 2.0.4 python package (https://pypi.python.org/pypi/bintrees/2.0.4). This package provides a binary tree, AVL-Tree, and Red-Black-Tree for storing key/value pairs. I plan on just testing the binary tree and Red-Black-Tree. The python standard library does not include these specific data structures, but does provide the dict implementation that would be useful for testing against. Bintrees can be installed using 'pip install bintrees.'

Both data structures I will be testing (BinaryTree and RBTree) have the same interface. Methods that will be tested include:

- __len__() <==> len(T)
- __max__() <==> max(T)
- __max__() <==> min(T)
- clear() -> None, remove all items from T
- copy() -> a shallow copy of T
- discard(k) -> None, remove k from T, if k is present
- get(k, [,d]) -> T[k] if k in T, else d
- is_empty() -> True if len(T) == 0
- pop(k[,d]) -> remove specified key and return the corresponding value
- pop_item() -> (k,v), remove and return some (key, value) pair as a 2-tuple
- foreach(f, [order]) -> visit all nodes of tree (0 = 'inorder', -1 = 'preorder' or +1 = 'postorder') and call f(k, v) for each node, O(n)
- remove_items(keys) -> None, remove items by keys

Other methods to test include set methods (intersection, union, difference, symmetric_difference).

Many of these methods (such as get() or pop()) could be tested by comparing against a reference from the python standard library, which would the dict class that also stores key/value pairs. There will be some methods in the bintree library that don't have equivalent methods for dict, however, which is why bintree is useful in the first place. These will include the max and min functions, traversal, and set methods.

Performance testing will also be done for the BinaryTree and RBTree, and performance will be compared against the dict class for functionality that they share. This would be useful for knowing how much of a performance hit is taken by using the bintrees library. This will also give some insight into how much a performance gain we are getting when inserting items into the red black tree versus the standard binary search tree. For insertion, the red black tree should have a worst case time of O(log(N)) whereas the binary search tree should have a worse case of O(N).

For all tests, I will make sure to use a variety of data types and values (ints from their lower to upper bounds, chars, strings).