Zehuan Chen; ID: 932541517

CS562: Alex Groce

Project Proposal

Introduction

The “pygtrie” is a Python library implementing a trie data structure. Trie data structure, also known as radix or prefix tree, is a tree associating keys to values where all the descendants of a node have a common prefix (associated with that node).
The trie module contains `Trie`, `CharTrie` and `StringTrie` classes each implementing a mutable mapping interface. The module also contains PrefixSet class which uses a trie to store a set of prefixes such that a key is contained in the set if it or its prefix is stored in the set. So this library could be found at GitHub https://github.com/google/pygtrie .
This library have some approaches that show advantages to another data structure, first approach is pygtrie is a full mutable mapping implementation, also it supports iterating over as well as deleting a subtrie and prefix checking as well as shortest and longest prefix look-up. Which is also extensible for any kind of user-defined keys, can store any value including None.

The testing Plan

This library contains four basic classes: class Trie(*args, **kwargs), class CharTrie(*args, **kwargs), class StringTrie(*args, **kwargs) and class PrefixSet(iterable, factory, **kwargs) . The only difference between pygtrie.CharTrie and pygtrie.Trie is that when pygtrie.CharTrie returns keys back to the client (for instance in keys() method is called), those keys are returned as strings. As for StringTrie, The trie accepts strings as keys which are split into components using a separator specified during initialization. pygtrie.PrefixSet works similar to a normal set except it is said to contain a key if the key or it’s prefix is stored in the set. The set supports addition of elements but does not support removal of elements. This is because there’s no obvious consistent and intuitive behaviour for element deletion.
All these classes have several functions. Keys used with the pygtrie. Trie must be iterable, yielding hashable objects. In other words, for a given key, fromkeys() must be valid. For class Trie, I am planning to test fromkeys(key) first, because this method can

judge whether this Trie is legal or not, the return value have to be a new Trie where each key from keys has been set to the given value. By using TSTL, we could see if the structure of Trie match with the description above, also we could test the robust propertied of this call, the TSTL can help us to see how many keys we can add maximum, see the time consuming and performance when we continues adding elements to the Trie.
Besides, the function has_node(key) returns whether given node is in the trie, so using TSTL to test if the return value is equal to type of HAS_VALUE or HAS_SUBTRIE.
For other classes such as StringTrie(), the input data is a string with separator as keys.