Justify For Assignment 4

Zhuowen Song & Keming Chen

U1313034 & U

University of Utah, School of Computing, CS3505

Oct 3, 2022

REPO: https://github.com/University-of-Utah-CS3505/a4-refactor-and-test-UKeming

In a3, we used ***Node* childNode[26]*** to store the childs of the node. We need to initialize every element of the array after it is created. Every Trie in the childNode array should be recursively destroyed after the Trie which holds the childNode was destroyed. Also, in the Trie destructor, the childNode will be destroyed since it is allocated to the heap.

Previously, we are forced to allocate the Trie on the heap since we are only holding the address of the new created object, and we want it still to be there after leaving the scope. Since it must be allocated on the heap, we must manually destroy it. The destruction of Trie is somewhat complex because Trie contains the address of many other Tries. If one of them is destroyed, every Tries in the Trie should also be destroyed. Therefore, we must do recursive destruction in the Trie destructor.

However, with the map structure, which can hold the copy of the Trie instead of the Trie's address, we can ensure that the Trie is still there after leaving the scope. Therefore, the heap allocation is unnecessary. We can simply use the assign operator to save the copy of the Trie into the map, and the map will keep the object for us. Therefore, there will be nothing on the heap. The destructor can be empty.

We optimized all of the variable names in our project. In detail, for example, we used ***pointer** as the iterator to go deeper in the tree. This name can not describe this variable is descending the tree. Now we are using scanningNode to represent that.

We cleaned up the classes. I (Zhuowen) used two classes to implement the trie tree: Trie and Node. After we grouped up and discussed the structure of the code, we decided to use only one Trie class to finish this assignment.