Alvin Jasper Juan

CSS 143

Professor Peng Du

14 March 2020

Autocomplete Design

In this assignment we were asked to complete a code from scratch using all of the content and knowledge we’ve obtained from the class. We had to complete a similar code similar to google search auto complete meaning that when you start typing in the search bar, a list of suggested words is provided immediately. The time complexity for this code is O(n) and it was the fastest I could do because I needed to still loop through all of the nodes to be able to check if it’s the last one. First, the data structure I used is LinkedList because in a sense it was easier for me and I could pretty much use a tree data structure to complete the words and connect the characters and also it is easier to search for a string in a linked list. So, each trie node holds one character and the reference for the parent and child nodes and it is pretty good at storing and searching words and strings. To start off the code, when a word is initialized we put the characters in each try node for example if the word is “mouse”, my code would put the character “m” in a node then “o” then “u”, “s”, “e” and from the node “o” we can also connect more characters that could form a word that starts with “mo” for example “more”, “moo”, “month” and more. By having this way, more words can be added to the parent nodes and we can suggest words through the similarities of their prefixes. I created TrieNode which is pretty much the same as a TreeNode because it has references to its children and parent. We have the same implementation as a tree so we have add and search in the code but the only difference is the autocomplete method which is solely for this code. The autocomplete method is what completes the uncompleted word and find the last node of the given word and suggests multiple words that are associated with the end of the prefix. For example, if “ta” were given as a string input for the method then it would output “target”, “tactical”, “tablespoon” and more. To put all of the characters together, i added a function to concatenate the characters together to later form a word using recursion. I have a function called getWords which uses preorder traversal and it checks if the node is the end of the word then it adds words to the list and it also does the same for the child nodes. Autocomplete then uses the same process I just described which just loops through to the end of the prefix and then it returns all of the words that are connected with it. In the main function, I added words that would go with the prefix and some words that didnt and when I tested it out the outcome was this.

“h” = [hello, high, hollow, how]

“se” = [seattle, seatac, see]

“sea” = [seattle, seatac]

“ho” = [hollow, how]

“xyz” = []

Proof of the test:

