Jackie Kelly

CSS 143 – Peng Du

Homework 7

Autocomplete

In order to create an autocomplete function to be implemented with a search engine, I will use an array data structure. This data structure will reduce the function runtime in comparison to other data structures because each character in each word of the text file/dictionary will have access to the next character in the given word if it exists. If there are no more characters in the word (signifying the given character is the end of the word) the character will point to a Boolean value, valued “true.” At first I was inclined to use a HashMap to implement the autocomplete, since autocomplete will work like a dictionary, and we used a HashMap for the dictionary/search implementation; however, after I analyzed the time complexity of using a HashMap, I quickly realized the concept of an array (like our Heap class) will improve performance because the array data structure offers random access and update capabilities.

This will work by indexing each character into the array, so the data structure will use one character as a parent, and its children will be the characters in the word that follow. More specifically, given the set of strings {"hello", "high", "seattle", "seatac", "see", "hollow", "how"}, if you were to search ‘ho,’ the letter ‘h’ will have children to ‘e,’ ‘i,’ and ‘o.’ After parsing the next character in the word ‘ho,’ which in this case is ‘o,’ the autocomplete will see that it is at the length of this passed in string, return the words that contain ‘o’ as a child of ‘h’ (which are ‘hollow’ and ‘how’) ending the parsing of the string. It will return the entirety of these two words by running through the rest of the words that contain ‘h’ and ‘o’ in order until it reaches the Boolean value of “true” marking the end of the word in the list of strings. The autocomplete class will also ignore casing by inserting the list of strings to the Trie after calling the “.toLowerCase()” function.