Ryan Khamneian

Dr. Du

CSS 143B

March 14, 2020

Problem 3 Design

Problem 3 of homework 7 asked us to create a search engine autocomplete (like google searching). At first, I thought this sounded way too simple and just used substrings and a for loop. Of course, it was not that easy, it never is! After seeing the last lecture, I realized that the data type we are supposed to use is a Trie. However, unlike what we saw in the lecture, I used a linked list to implement it instead of an array. The reason for this is that it is faster. I will get to the reason why and the implementation in a second, however I would like to cover the complexity first (Big O Notation). The searching aspect is O(1) because it is not dependent on the number of nodes, and instead depends on the length of the query. “findEndings”, which is the name of the function I built to find which words it can be, is O(N) because it visits every node once to build the ending. Overall it ends up being O(1).

To create the autocomplete I had to create two classes, one for the Trie and one for its Node. The Node function instantiates the Trie and creates it having a value and a linked list of tries called its “children”. It also has a bunch of get functions which we need, most importantly, the recursive function, “get endings” which I mentioned earlier. The Trie class has a few functions. The first is to initialize the tree by taking in all of the first set of children (the first letters) and then giving each individual letter the next set of children. For example, if the dictionary has just two words (“Seattle” and “SeaTac”), the first list is just “s”, the second is just “e”, and so on until we get to the 5th which breaks off into “t” and “a” etc. Then there is a search function which takes in the query and grabs only the words which it fits in, and finally autocomplete which creates the possible words and builds the string in the least expensive way possible.