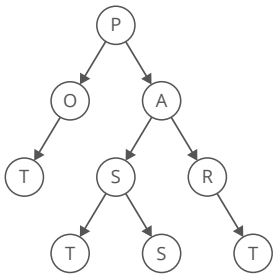
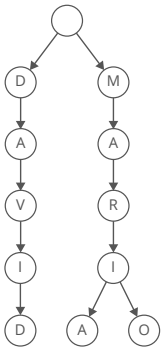
Tries in C++

Final Project Proposal – CSPB 2270 Spring 2024

By Nathan J Harris – naha3153





Introduction

For my final project I have decided to take the second programming assignment, "Dictionary Lookup and Autocomplete Using Tries", from CSPB 3104 Algorithms and adapt it from python to C++ in the testing environment we have used for our homework assignments in this course so far.

Data Structure Description and Application

Tries, also called Prefix Trees, are an interesting data structure with many useful applications. We can use them to store a set of words in a treelike structure where prefixes of the words share nodes and edges.

Tries have three different types of nodes. The root node, an internal node, and a word ending node. The root node stores edges to the internal nodes. Internal nodes store pointers to the next characters in the tree and a word ending nodes hold the number of times that word has been inserted into the trie. In a trie, the edges store the characters of the alphabet. For example, the struct for a trie node would include a word end count number, initialized as 0, and a map that holds the entire alphabet as keys, and the values would be the next node in the tree.

Inserting a word involves traversing the trie when the new word shares the prefix of an existing word in the trie and branches off when the new word is different than anything present in the trie. Inserting a word always increments the word count on the word ending node. In addition to inserting, we can also look up the frequency of a word, by traversing and returning the word count of the word ending node. We can also have a method called autocomplete, where with a given prefix, we can provide all the words that share the prefix.

In addition to autocomplete, tries can be used for full-text search, web search engines, and internet routing. There are some variations, but in this case, I will be focused on a 26-ary tree, storing the characters in the alphabet. In some cases, tries can replace hash tables since they do not have hash collisions nor require a hash function.

Motivation and Implementation Details

I enjoyed learning and exploring tries for the previous assignment and I am curious about exploring and using them further. I am also interested in growing my C++ skills by adapting a completed python assignment as well as practicing test driven development methods and using build tools. For this project I will be writing the tests myself and not adapting them from the Algorithms course assignment. My end goal is to have a complete project that looks like any of the other homework assignments in CSPB 2270 Data Structures but for the tries data structure.

Challenges and Hurdles

In setting out for this project I do not foresee any significant technical hurdles since the knowledge required is within reach. The most foreign task of the project would be writing the tests in C++ for the data structure and its methods. My main hurdle will be completing the project on time alongside my other courses' projects and exams. The scope of the project will focus on the three methods, insert, word count, and autocomplete. If I finish the project with extra time, I would explore more methods and perhaps adapting it to a radix tree (very similar to tries but takes up less space by compressing the nodes).

Conclusion

It is my hope by choosing a project with a limited scope it will enable me to focus more using C++, writing tests, and having a complete working project within my time constraints. In researching some fascinating data structures, I realized that I needed a focused project in order to achieve my desired results in the course. I am looking forward to this project.

References

Information for this proposal was taken from the mentioned CSPB 3104 assignment as well as the Wikipedia article on Tries.

https://en.wikipedia.org/wiki/Trie

Images were taken from this informative article on interview cake.

https://www.interviewcake.com/concept/python/trie