# Oanh Le

Instructor Dr. Valentina Korzhova

CIS4900

April 28th, 2023

# CIS4900 Independent Study Final Report

1. **Project Description:**

The project focuses on studying the use of the data structure trie (prefix tree) which is tailored for storing and retrieving strings. This report will cover my research and exploration of the core data structure of trie, its real-world applications, and the implementation of simple autocomplete search. It will cover the detail explanation and pseudocode of the autocomplete search for the USF class schedule based on the subject, course number, and title.

1. **Prefix Trees**

A prefix tree, also known as a trie, is a data structure tailored for storing and retrieving strings. The root node represents the empty string. Each possible next character branches to a different child node. Strings stored in the trie are inserted explicitly by the user; prefixes of such strings, although they occur along paths in the trie, are not considered to be stored in the trie unless they have been explicitly inserted. The concept of trie is used in many applications such as:

* Autocomplete and spell checking: Prefix trees can be used to efficiently implement autocomplete and spell-checking features in software applications. As a user types, the prefix tree can quickly suggest possible completions or correct misspelled words.
* IP routing: Prefix trees can be used to efficiently route IP traffic in computer networks. Each node in the tree represents a prefix of an IP address. The tree can be used to quickly determine the next hop for a given destination IP address.
* Dictionary search: Prefix trees can be used to store and search through word dictionaries. This can be useful in applications such as word processors or language translation software.
* DNA sequencing: Prefix trees can be used to efficiently search for patterns in DNA sequences. This can be useful in fields such as bioinformatics and genetic research.
* Text compression: Prefix trees can compress text by representing repeated substrings as a single node in the tree. This can be useful in applications such as data storage and transmission.
* Web indexing: Prefix trees can be used to index web pages, allowing for fast and efficient searching of large collections of web pages.

1. **Autocomplete Class Schedule Search**
2. **Overview**

As of now, the USF class schedule search system only allows manual searches. In other words, you need to enter the subject, the course number, and the title separately. The system has the following flaws:

* In the subject search, you can only search for the first letter of the word.
* Inserting a leading space accidentally will fail the title search even though the title is entered correctly.
* Search will fail when entering the subject+course number correctly but the input title is misspelled. (e.g, search will fail if user inputs CAP+4401+Images Processing instead of CAP+4401+Image Processing)

A better user experience can be achieved by implementing an autocomplete search for the subject, course number, and title (this can extend and apply to searching for instructors and other fields as well). We can use trie to store a sequence of values (subject+course number+title) in such a way that tracing the path from the root to any node yields a valid subset of that sequence. For example, if the current text S is “Co”, then a plausible completion found by prefix search could be “COP2510 Programming Concepts”. A multi-term prefix search could yield results such as "CDA4205 Computer Architecture" or "EML3041 Computational Methods".

1. **Design**

The autocomplete search for the class schedule will be based on the subject + course number (e.g. COP 4620), and the course title. The figure below shows the trie containing the strings ‘COP’, ‘COP4620’, and ‘Compliers’.

* When the search is ‘CO’, the result will contain ‘COP’, ‘COP4620’, and ‘Compliers’.
* When the search is ‘COM’, the result will contain ‘Compliers’.
* When the search is ‘COP’, the result will contain ‘COP’, ‘COP4620’.
* When the search is ‘COP4’, the result will contain ‘COP4620’.
* When the search is ‘COT’, it will yield no result.

Note that ‘COP4620’ is the subject + course number of the class title ‘Compliers’. When use searches for either one, it will yield the exact same result. The idea is that each node in the program stores the CRN (the unique number that defines an individual course), so that when searching for a specific subject + course number or the class title, the result can retrieve the course information based on that CRN.

**A picture containing shape

Description automatically generated**

Figure 1

1. **Algorithm**
2. **Insertion**

To insert a string S into trie, we start with the root node. We will choose a child or add a new child node depending on S[0], the first character in S. Then we go down to the second node and we will make a choice according to S[1]. Then we go down to the third node, so on and so for. Finally, we traverse all characters in S sequentially and reach the end. The end node will be the node which represents the string S.

1. **Search**

The algorithm to search a target word in a trie is as follows:

* Find the path of the word’s prefix in the tree, start from the root node.
* If search fails which means that no words start with the target word, the target word is not in the trie.
* If search succeeds, we need to check if the target word is only a prefix of words in trie or it is exactly a word. When constructing the Node, we will use a Boolean variable to signal to indicate if the string represented by this node is a word or not.

1. **Autocomplete**

This function utilizes the Search() function to obtain the list of words that start with the string input provided (as explained in the section (3A)).

1. **Program Delivery**

The technology stack of my program was Javascript, HTML, and CSS. The trie and the backend was implemented using Javascript, and the frontend was done using CSS and Tailwind library. The example data was retrieved from the actual class schedule data in Spring 2023. Figure 2 depicts the example of the program. The demo can be accessed via this link:

Graphical user interface, application

Description automatically generated

Figure 2