What is a Trie?

This lesson gives a brief introduction to Trie, its properties, and common applications.

We'll cover the following

Introduction

In the previous section, we covered several common types of trees like Red-Black trees, 2-3 trees, etc.

Now, we are going to look at a tree-like data structure that proves to be really efficient while solving programming problems related to **strings**.

This data structure is called a **trie** and is also known as a **Prefix Tree**. We will soon find out why.

The tree **trie** is derived from "retrieval." As you can guess, the main purpose of using this structure is to provide fast retrieval. Tries are mostly used in dictionary word searches, search engine auto-suggestions, and IP routing as well.

Common Applications of Tries

Let's have a look at some real-life examples to understand the role of tries.

Autocomplete Words

Today, the autocomplete feature is supported by almost all major applications. This feature can be very efficiently implemented with the help of tries as they reduce the overall cost of performance.

Spell-Checking

Tries come in handy when you need to perform a spell-check on a word entered by the user. This feature is really helpful when the user does not know the exact spelling of a word he or she is searching for.

Searching for a Phone Contact

Another real-life use of Tries is searching for contacts in our contact list. It provides auto-suggestions based on the combination of letters that we enter. As you'll see later on, this can also be performed with hash tables, but a hash table won't be as efficient as a trie.

Properties of a Trie

To maintain its overall efficiency, tries follow a certain structure:

- Tries are similar to graphs as they are a combination of nodes where **each node represents a unique letter**.
- Each node can point to `None` or other children nodes.
- The size of a trie depends upon the number of characters. For example, in the English alphabet, there are 26 letters so the number of unique nodes cannot exceed 26.
- The depth of a trie depends on the longest word that it stores.
- Another important property of a trie is that it provides the same path for words which share a common prefix. For example, "there" and "their" have a common prefix "the." Hence, they will share the same path till "e." After that, the path will split into two branches. This is the backbone of the trie functionality.

These are some of the basic properties that every trie must hold. In the next lesson, we will discuss the structure of tries in detail.

← Back
Trees Quiz: Test your understanding of...

Next →
Structure of a Trie

✓ Mark as Completed

Report an Issue

Ask a Question
(https://discuss.educative.io/tag/what-is-a-trie__trie__data-structures-for-coding-interviews-in-python)