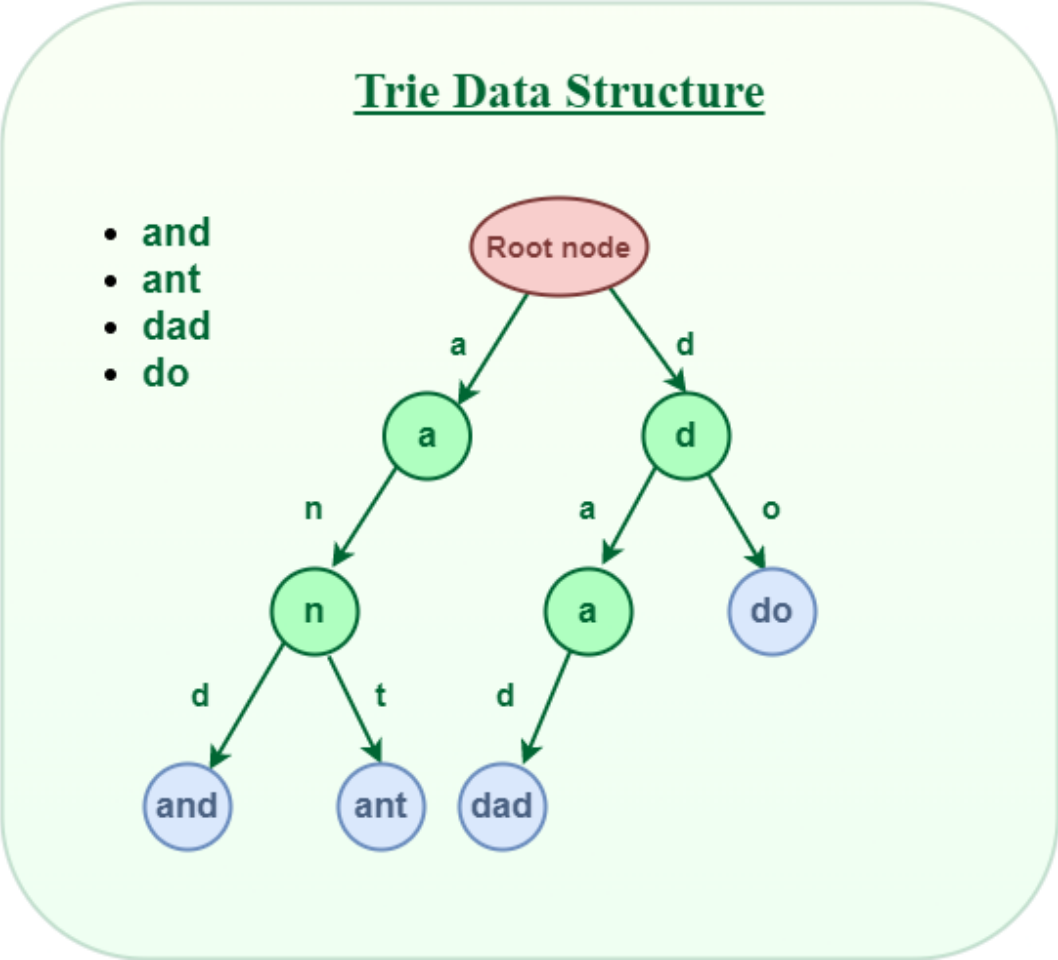
TRIE
DATA STRUCTURE

CODEUS_

WHAT IS A TRIE?



The **trie** data structure, also known as a **prefix tree**, is a **tree** variation used for efficient key/value retrieval.

Trie's **keys** are often strings, represented as a sequence of nodes with edges holding individual characters from the **keys**.

The main property of a Trie is that if two strings have a common prefix, they will have the same ancestor in the Trie.

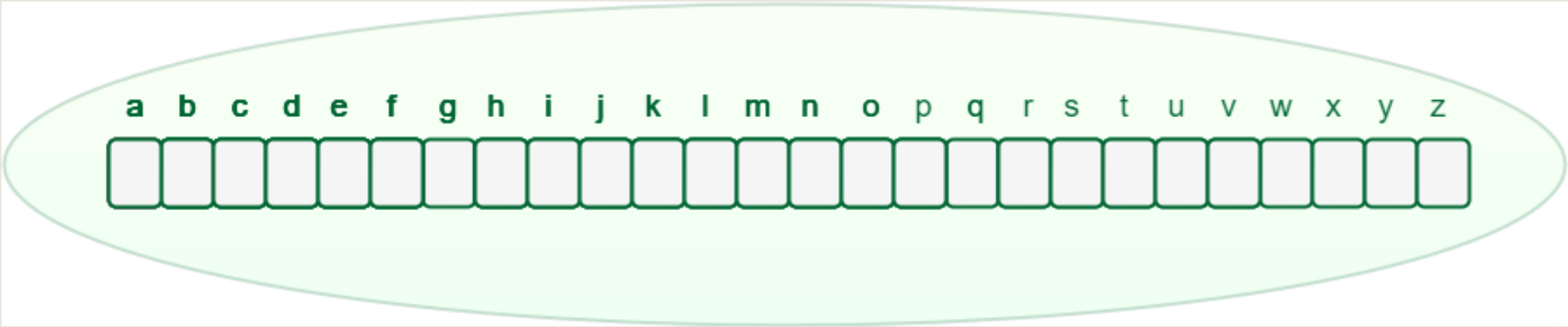
This particular property allows to find all words with a given prefix.

CODEUS_

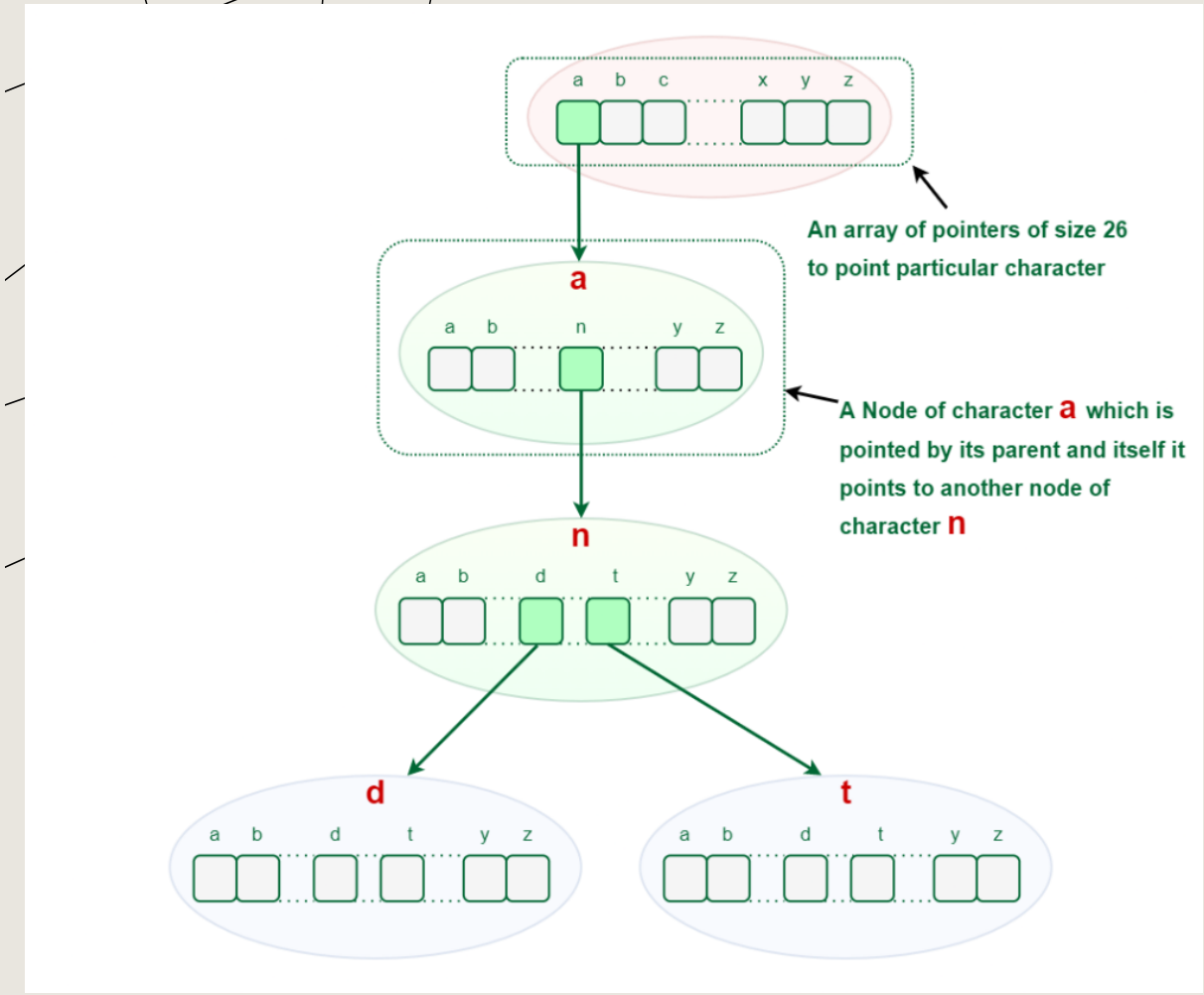
WHAT IS A TRIE?

Since **trie** is a **k-ary tree,** each **node** can have multiple children usually represented as an array.

In the simplest implementation, trie stores words as keys. For English alphabet trie, the size of the children array equal to 26.



PROPERTIES



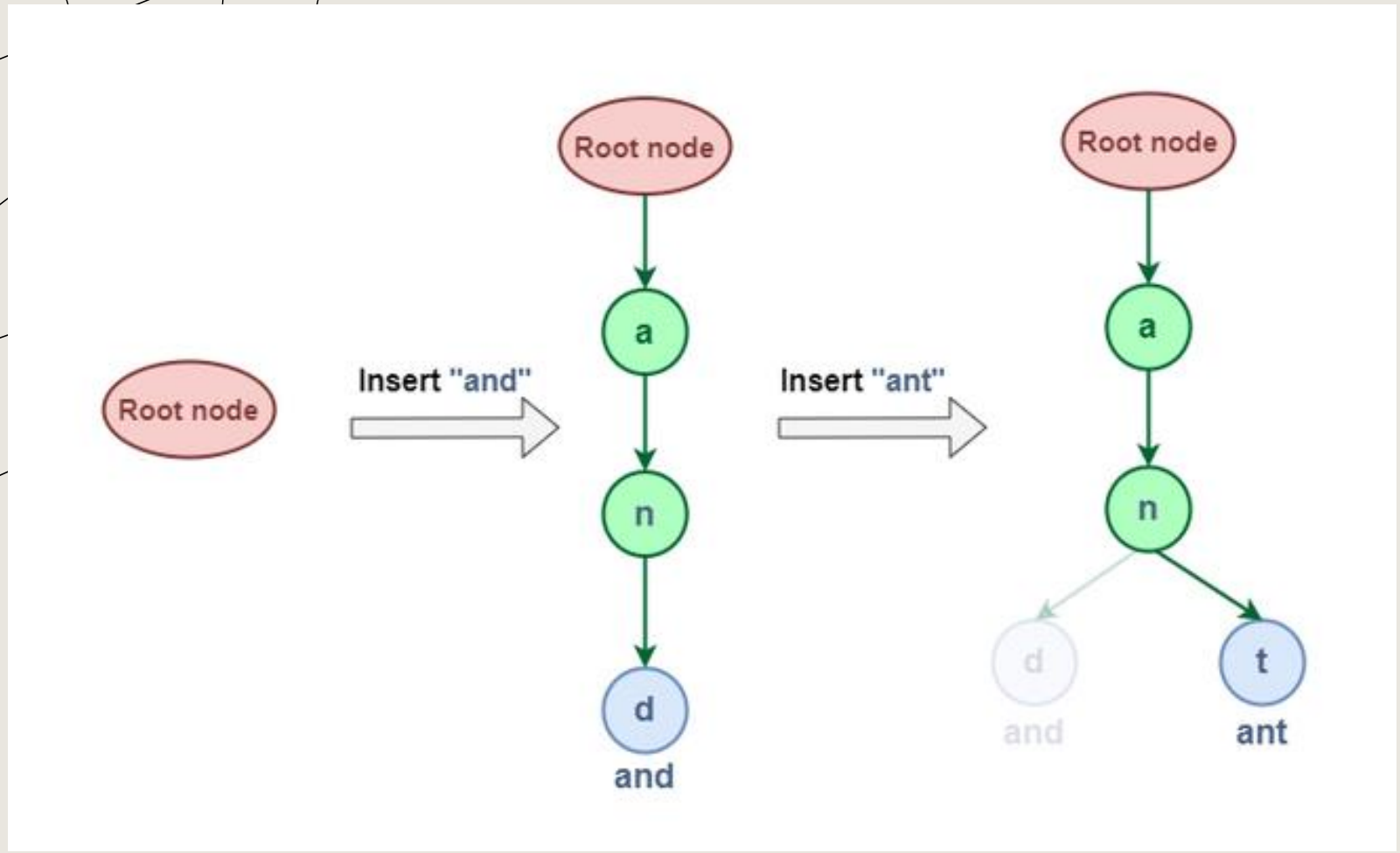
- Each Trie has an empty-value root node (red-color node), with links (or references) to child nodes.
- Index of each child node represents a character of a word (**key**).
- The index of each child node represents a character of a word that was inserted into a Trie.
- Each child node consists of a **value**, an **array** (or hashmap) of child nodes pointers, and a **flag** (blue-color nodes) to indicate if any string ends at the current node.
- Each path from the root to any node represents a word (**key**) or string (typically a prefix).

CODEUS_

BASIC OPERATIONS

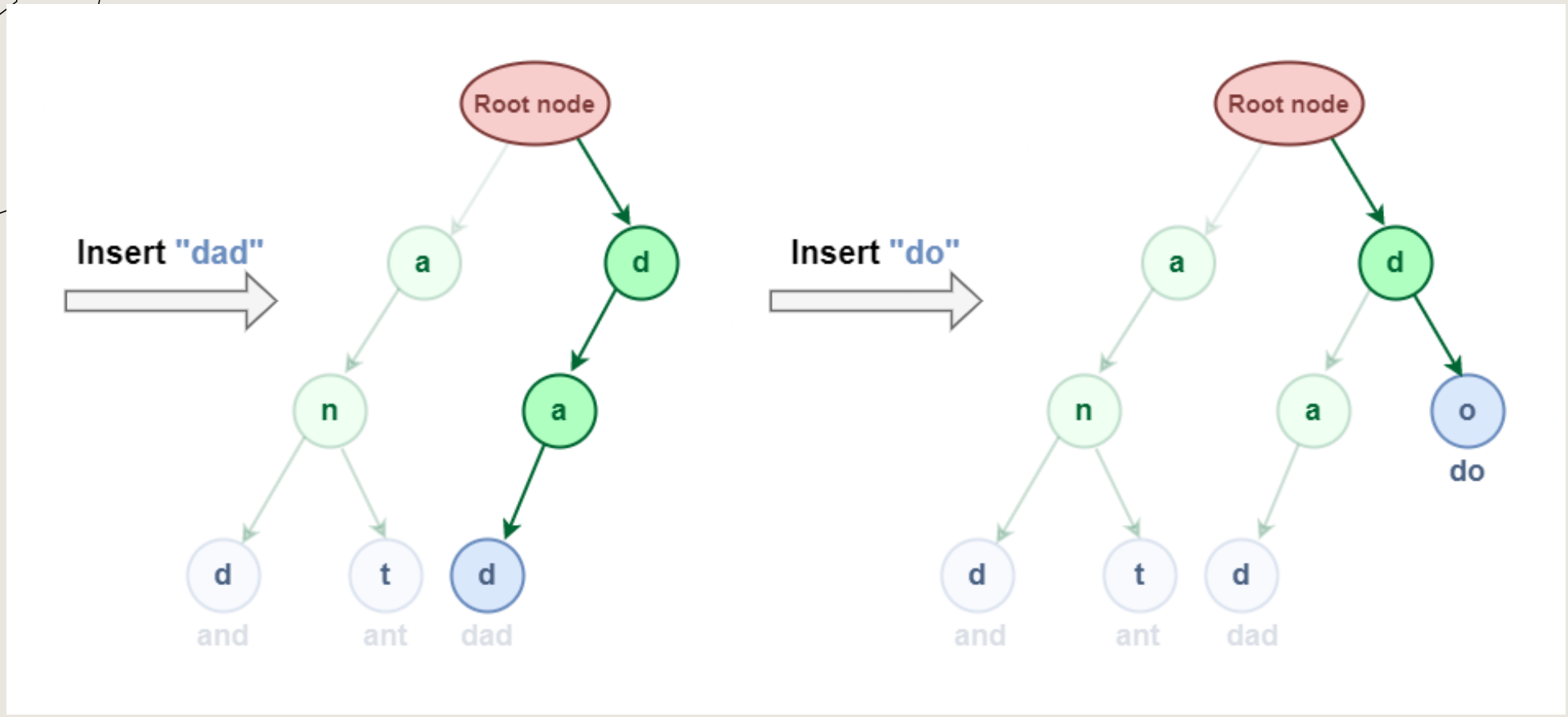
1. Insertion
2. Search
3. Deletion

BASIC OPERATIONS

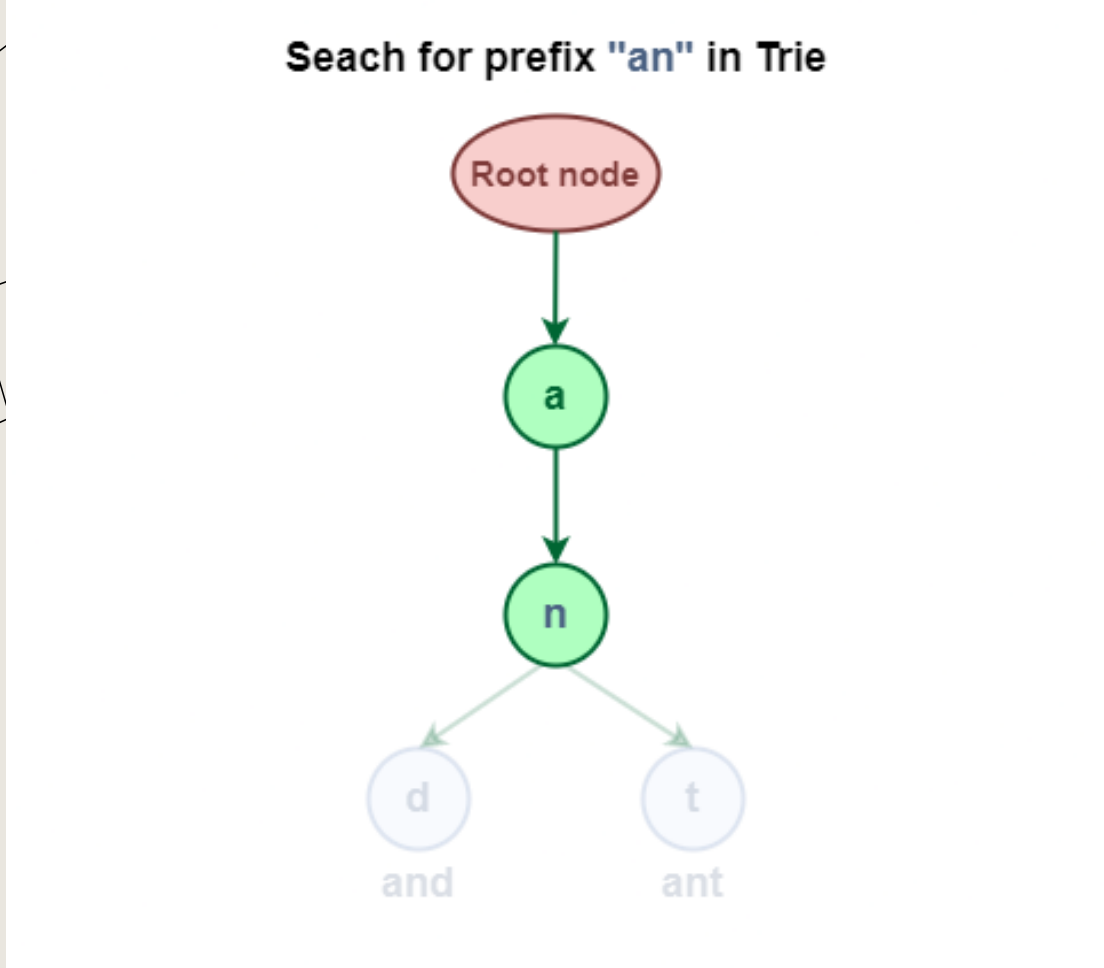


1.Insertion

BASIC OPERATIONS



BASIC OPERATIONS

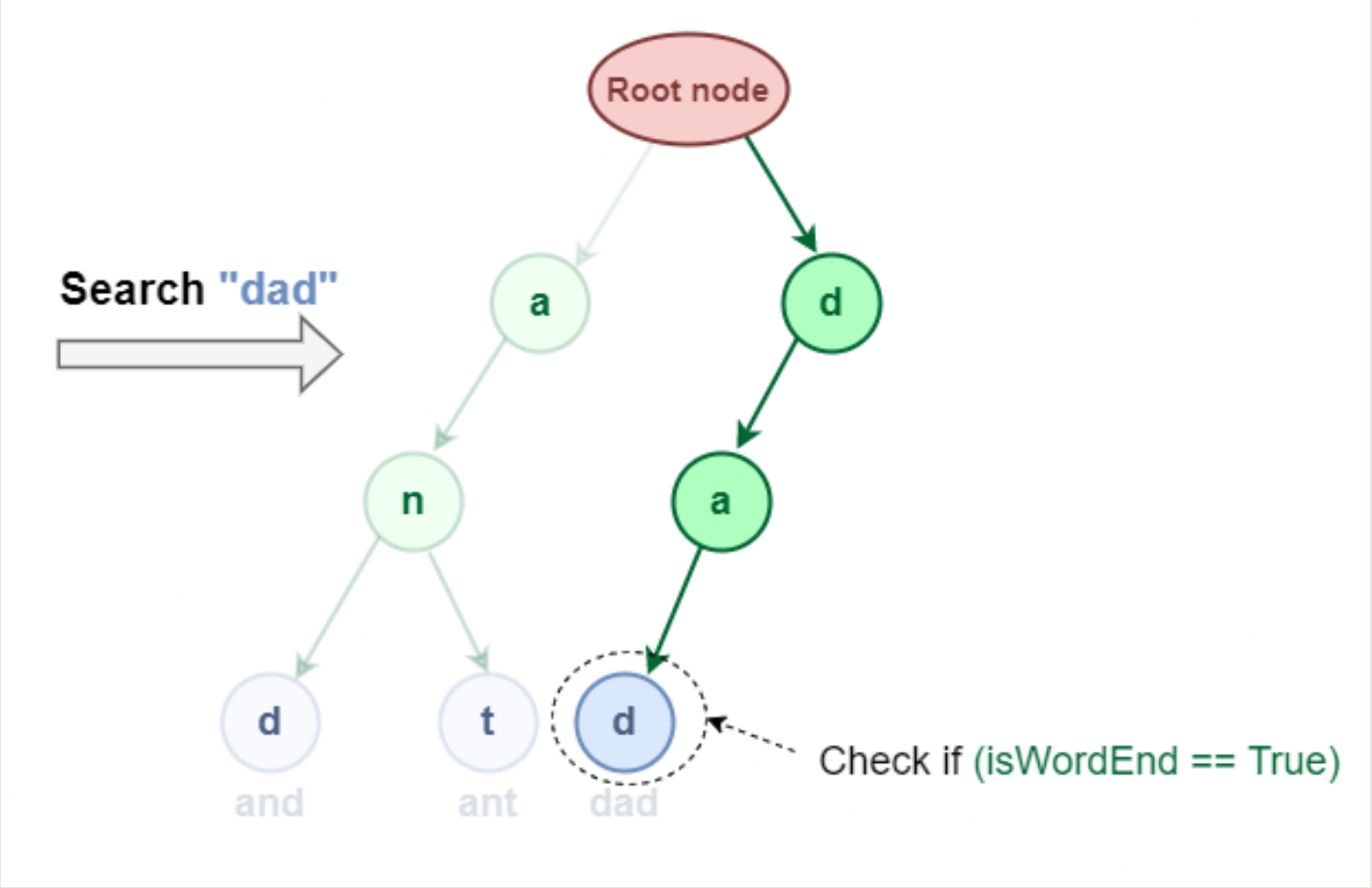


2. Search

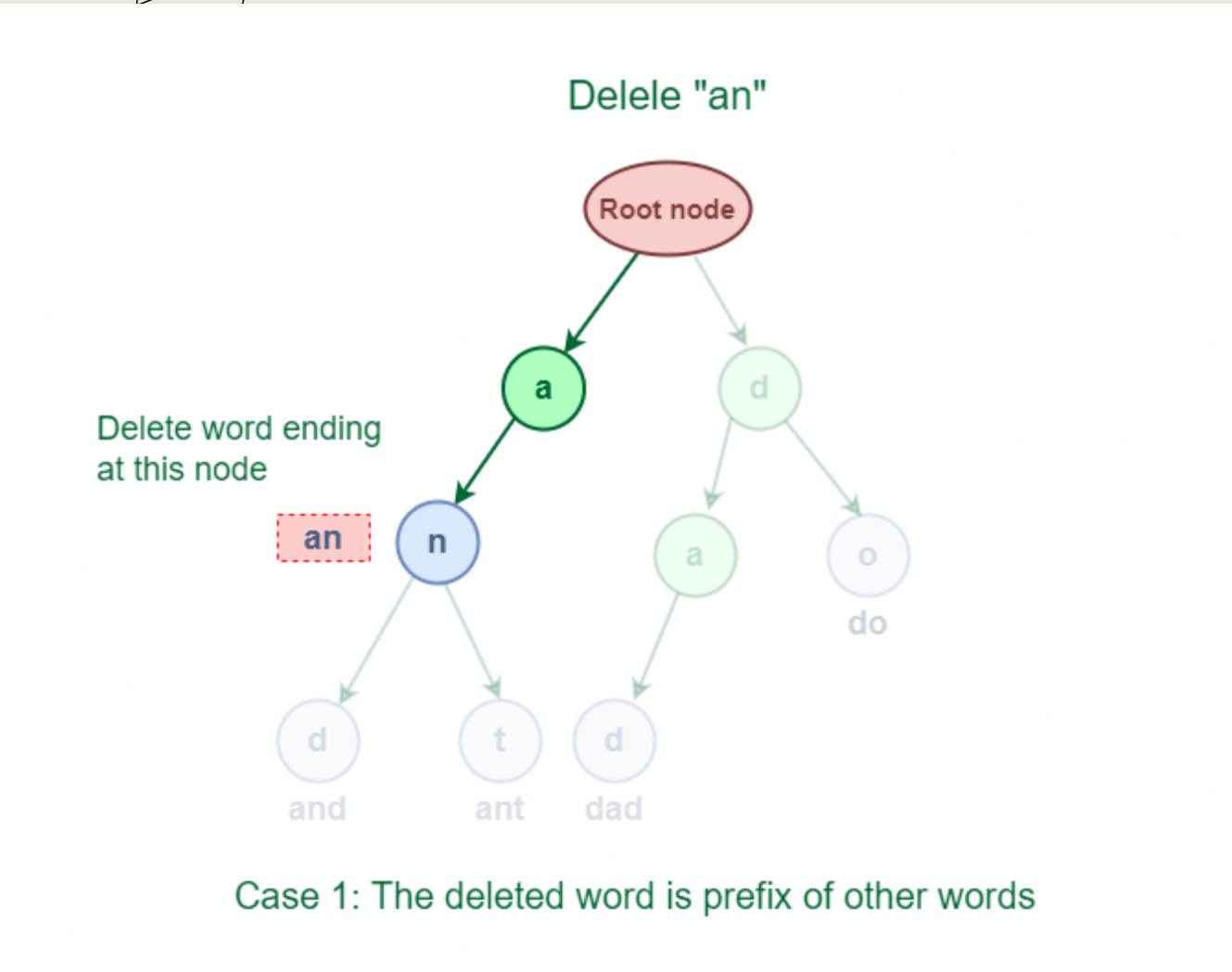
There are two types of search in the Trie:

1. Search for prefix.
2. Search for complete word.

BASIC OPERATIONS



BASIC OPERATIONS

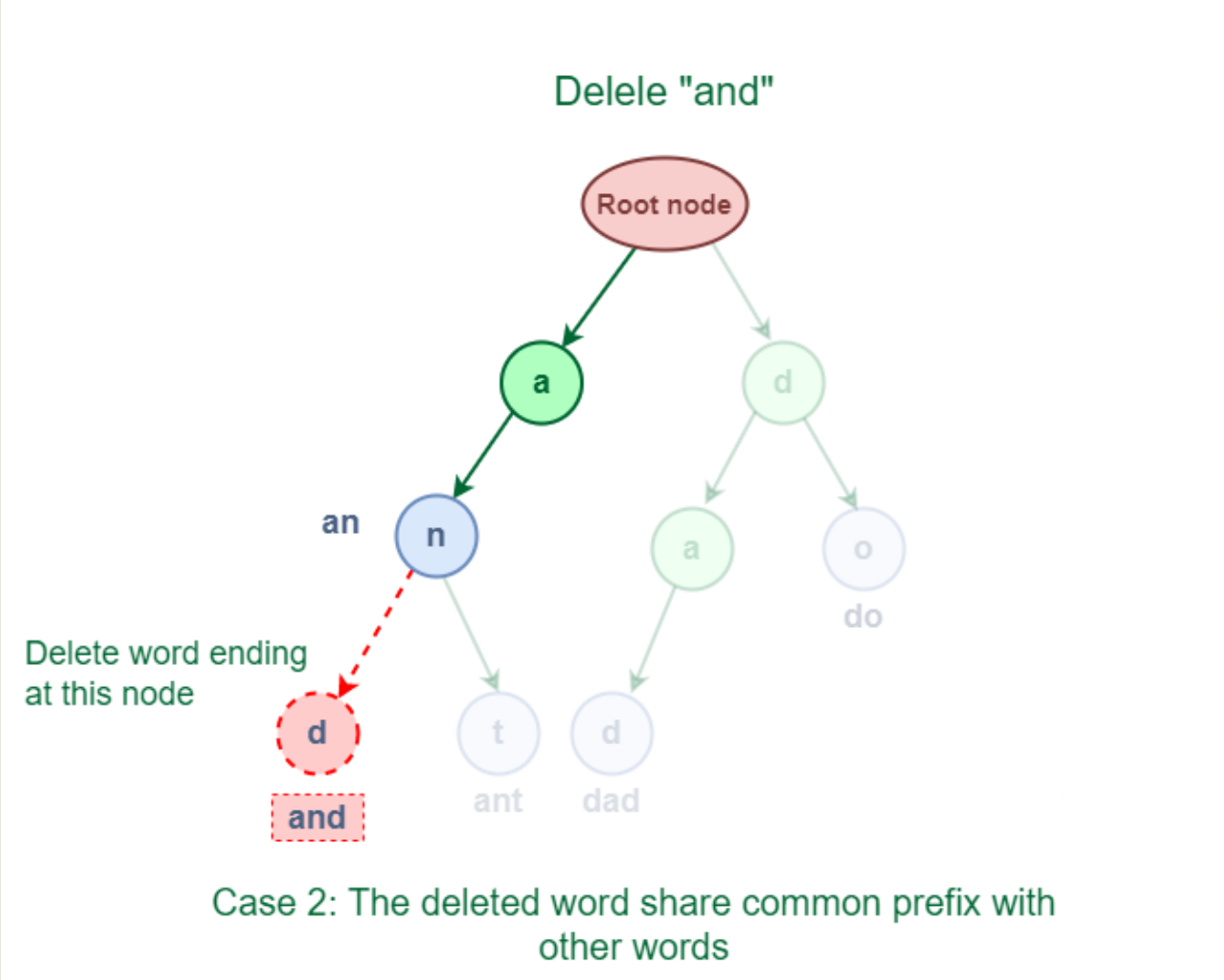


3. Deletion

There are three cases when deleting a word from Trie:

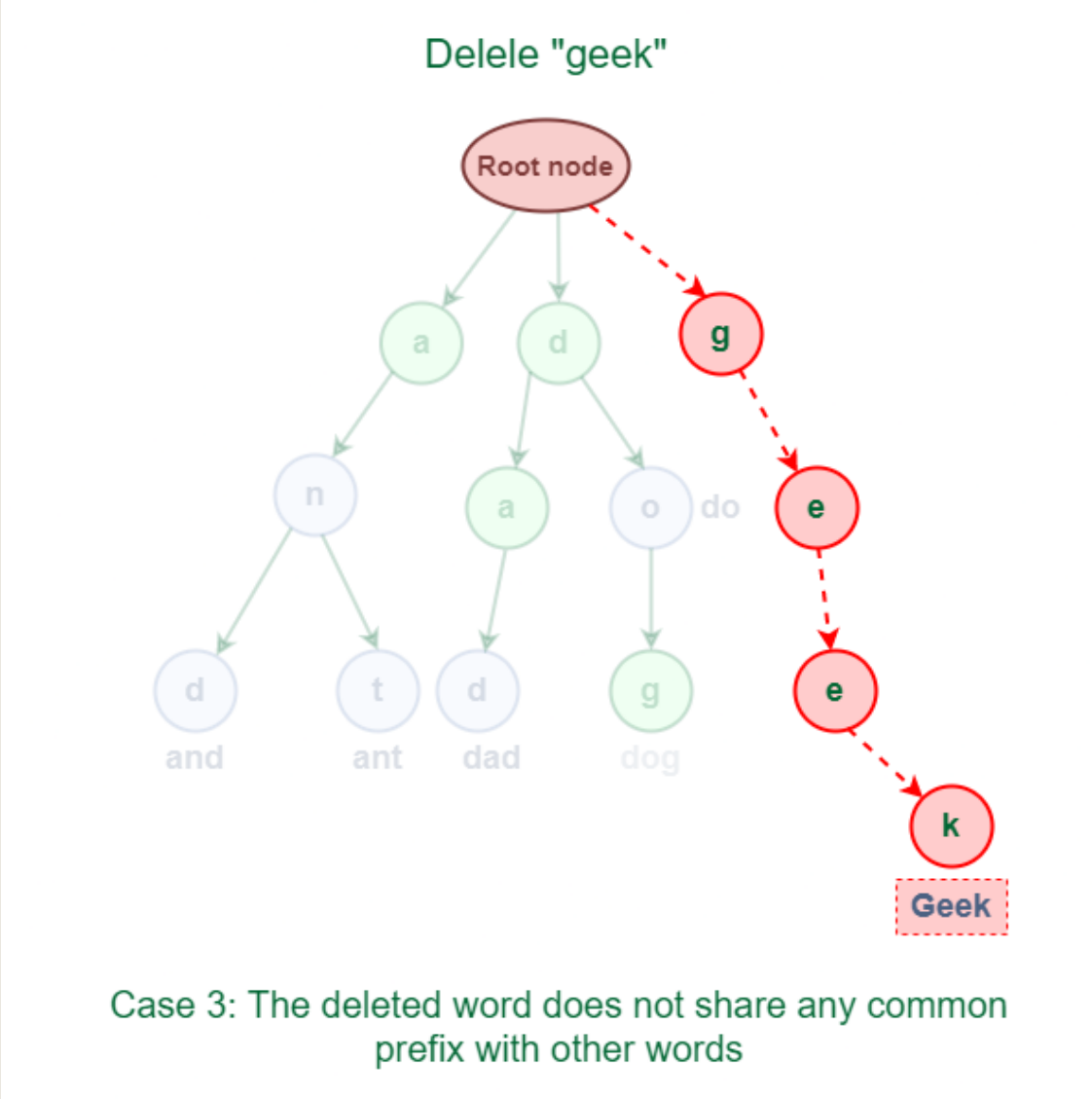
1. The deleted word is a prefix of other words in Trie.
2. The deleted word shares a common prefix with other words in Trie.
3. The deleted word does not share any common prefix with other words in Trie.

BASIC OPERATIONS



CODEUS_

BASIC OPERATIONS



CODEUS_

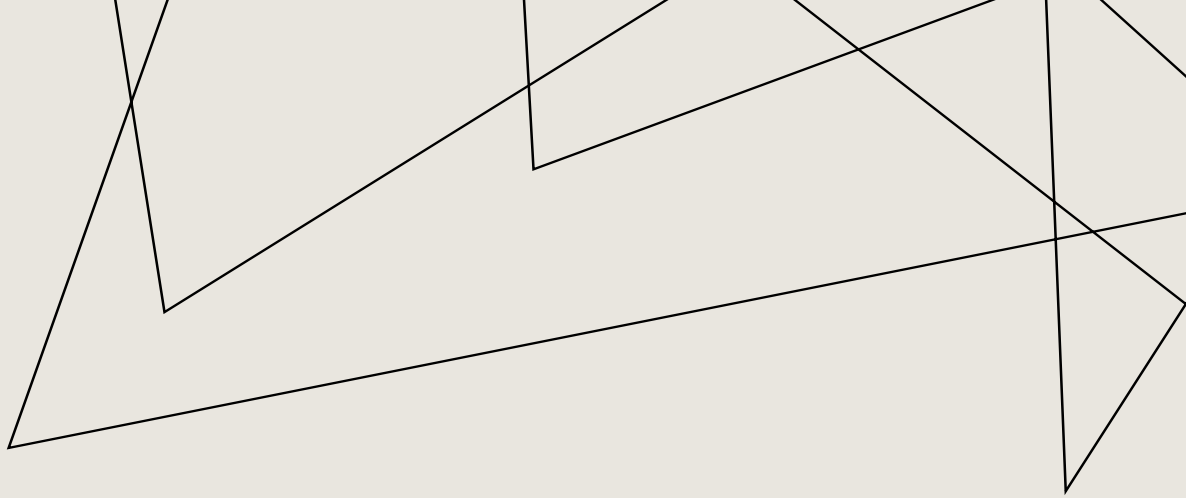
KEY FEATURES

Benefits:

- can be more effective than a **Hash Table** (as a result of the internal structure of data storage)
- efficient prefix-based searching
- sorted traversal of all words
- relatively fast search by value
- memory efficiency for common prefixes

Drawbacks:

- high memory usage for sparse data
- not optimized for arbitrary key types
- can be tricky to implement
- slower lookup for long words



RESOURCES

- Trie Data Structure Tutorial