Classes

Trie Class

The Trie class implements a tree-based classification structure with a root Node pointer and a count of total classifications. It manages all operations for inserting, classifying, and removing classification paths, storing them as comma-separated strings. The class validates input to ensure no uppercase characters are present and maintains proper memory management through its destructor. Each node in the trie can have up to 15 children, making child lookup operations effectively constant time.

Node Class

The Node class represents individual vertices in the trie, storing a string label and a vector of child node pointers (maximum 15 children). It provides methods for managing children (add, remove, get) and accessing/modifying the label. The class implements proper memory management through its destructor, which recursively deletes all child nodes. Each node can determine if it's terminal (leaf) by checking if it has no children.

Core Functions

insert(string classification)

Adds a new classification path to the trie, splitting the input string at commas and creating/traversing nodes for each label. Validates input for illegal characters, updates the classifications counter when appropriate, and returns status strings. Runtime is O(n) where n is the input string length, as it must process each character and traverse/create nodes for each label, with constant-time child operations due to the 15-child maximum.

classify(string input)

Traverses the trie based on provided input, building a classification path by selecting child nodes at each level using an external labelText function. Returns a comma-separated string of the classification path. Runtime is O(n) where n is the input length, as it processes the input and traverses at most n levels, with constant-time operations at each level due to the 15-child limit.

erase(string classification)

Removes a classification path from the trie by finding the target node, removing it if it's a leaf, and updating the classifications counter. Includes cleanup of empty branches and validation of the input string. Runtime is O(n) where n is the classification string length, as it must traverse to the target node and potentially clean up the path, with constant-time operations at each level.

print()

Performs a depth-first traversal of the trie, building and returning a string representation of all classification paths, separated by underscores. Runtime is O(n) where n is the number of classifications in the trie, as it must visit every node once and perform constant-time operations at each node.

empty()

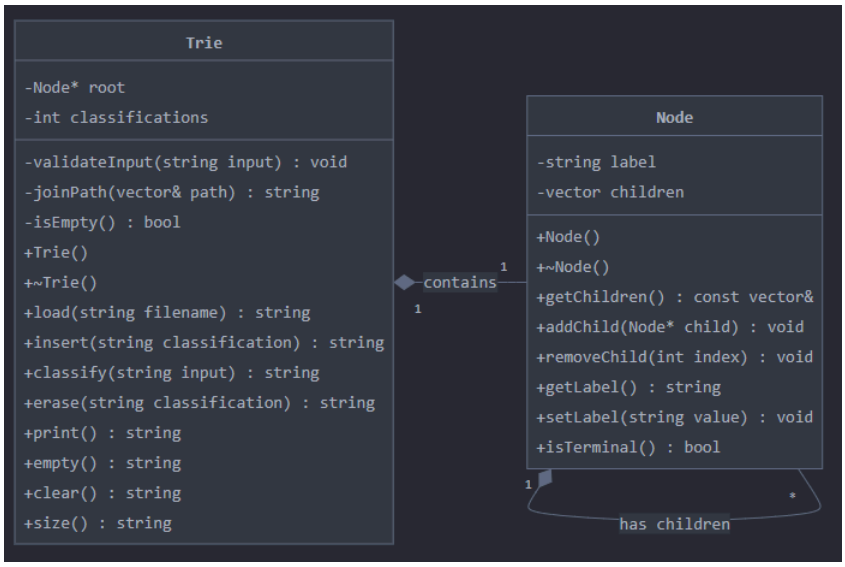
Returns "empty 1" if the trie has no classifications, "empty 0" otherwise. Runtime is O(1) as it simply checks the classifications counter.

clear()

Deletes all nodes in the trie and resets it to an empty state with a new root node and zero classifications. Runtime is O(n) where n is the number of nodes, as it must delete every node in the trie.

size()

Returns the current number of classifications in the trie. Runtime is O(1) as it simply returns the classifications counter value.


Trie
-Node* root
-int classifications
-validateInput(string input) : void
-joinPath(vector& path) : string
-isEmpty() : bool
+Trie()
+~Trie()
+load(string filename) : string
+insert(string classification) : string
+classify(string input) : string
+erase(string classification) : string
+print() : string
+empty() : string
+clear() : string
+size() : string
1
contains
1
Node
-string label
-vector children
+Node()
+~Node()
+getChildren() : const vector&
+addChild(Node* child) : void
+removeChild(int index) : void
+getLabel() : string
+setLabel(string value) : void
+isTerminal() : bool
1
*
has children