Tries

by Tian Cilliers

IOI Training Camp 3 (3-4 March 2018)

Definitions

Trie

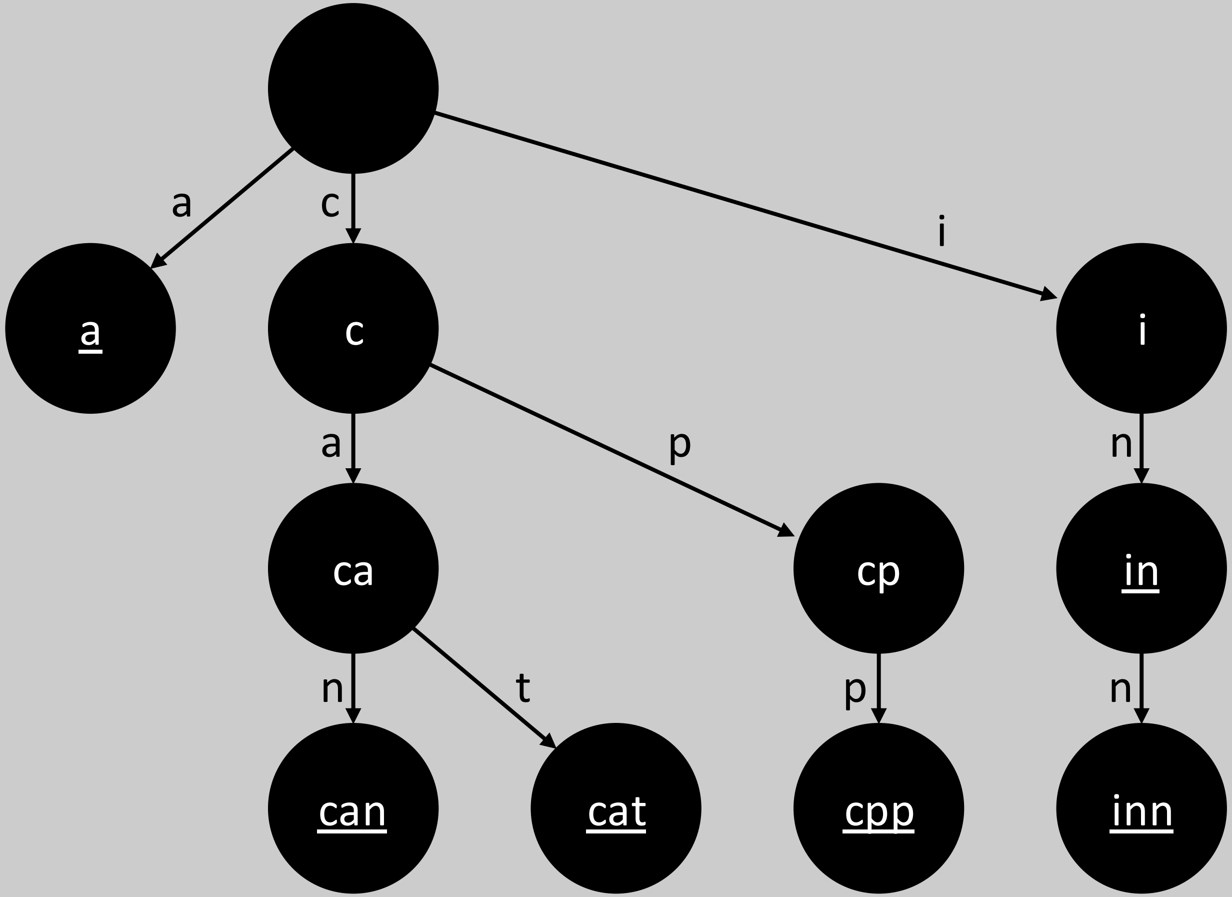
A trie (pronounced as in retrieval), also called a prefix or radix tree, is an ordered tree data structure used to store a dynamic set where the keys are usually strings.

- Can be used to store any associative data type
- Root node is empty
- Each node contains the prefix of all its children
- Not every node has to define a value, some can be intermediate nodes
- Can provide lexicographical sorting

Example

Storing the following values:

- cpp
- can
- cat
- in
- inn
- it
- a



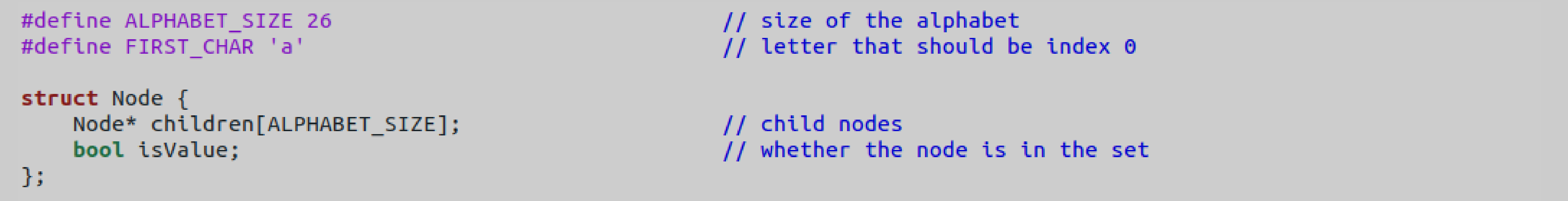
Implementation

Fundamental Structure

The first requirement is to setup a basic tree structure with the following properties:

- Each node can point to one child node for each letter in the alphabet
- Each node needs to store whether it represents a value in the dataset

Example:



```
#define ALPHABET_SIZE 26                                    // size of the alphabet
#define FIRST_CHAR 'a'                                      // letter that should be index 0

struct Node {
    Node* children[ALPHABET_SIZE];                          // child nodes
    bool isValue;                                           // whether the node is in the set
};
```

Implementation

Insert Value

Start at root node of tree.

For each character in string value:

- If child node corresponding to character doesn’t exist, add new empty node
- Descend to child node

Mark last node as valid value

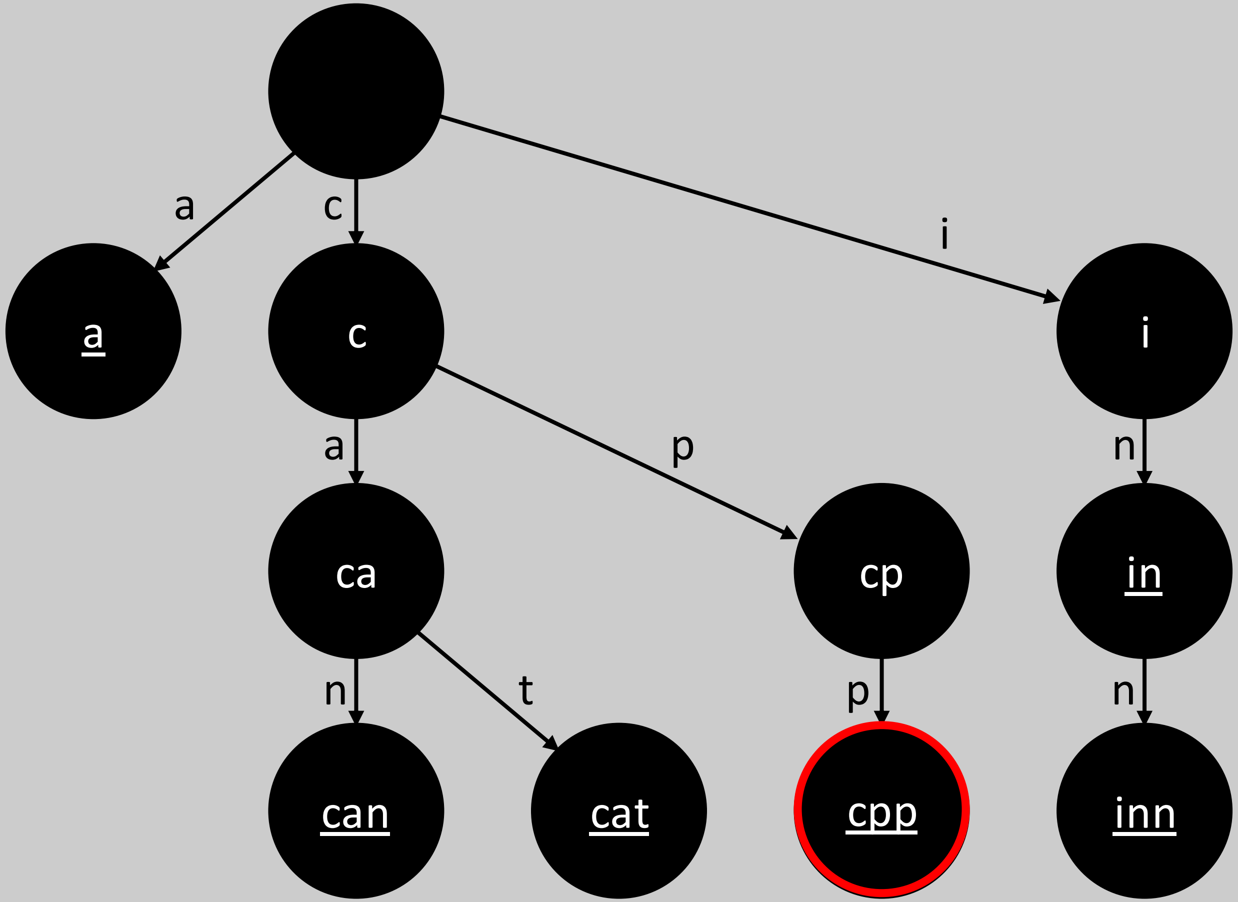
Implementation

Insert Value

Demonstration:

Adding string ‘cpp’ to tree

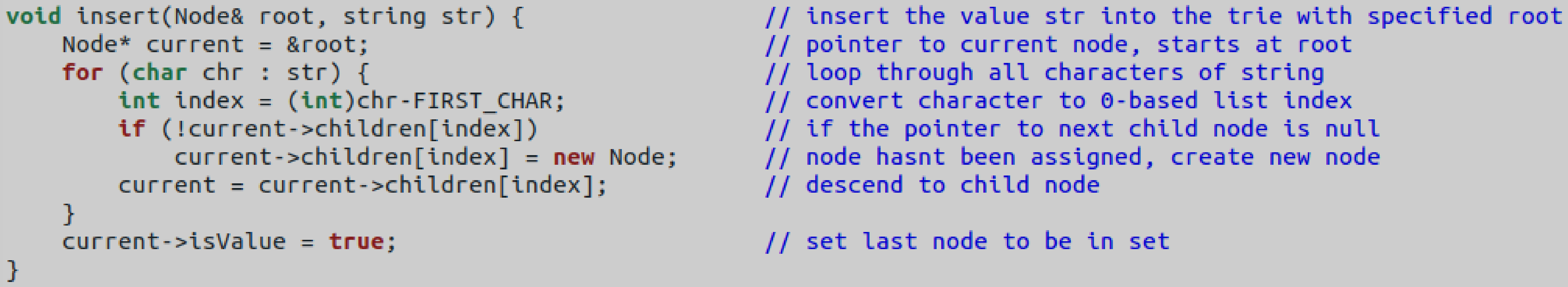
c p p



Implementation

Insert Value

Example:



```
void insert(Node& root, string str) {                 // insert the value str into the trie with specified root
    Node* current = &root;                            // pointer to current node, starts at root
    for (char chr : str) {                            // loop through all characters of string
        int index = (int)chr-FIRST_CHAR;              // convert character to 0-based list index
        if (!current->children[index])                // if the pointer to next child node is null
            current->children[index] = new Node;      // node hasnt been assigned, create new node
        current = current->children[index];           // descend to child node
    }
    current->isValue = true;                          // set last node to be in set
}
```

Implementation

Find Value

Start at root node of tree.

For each character in string value:

- If child node corresponding to character doesn’t exist, exit and return false
- Descend to child node

Return true if last node is marked as valid

Implementation

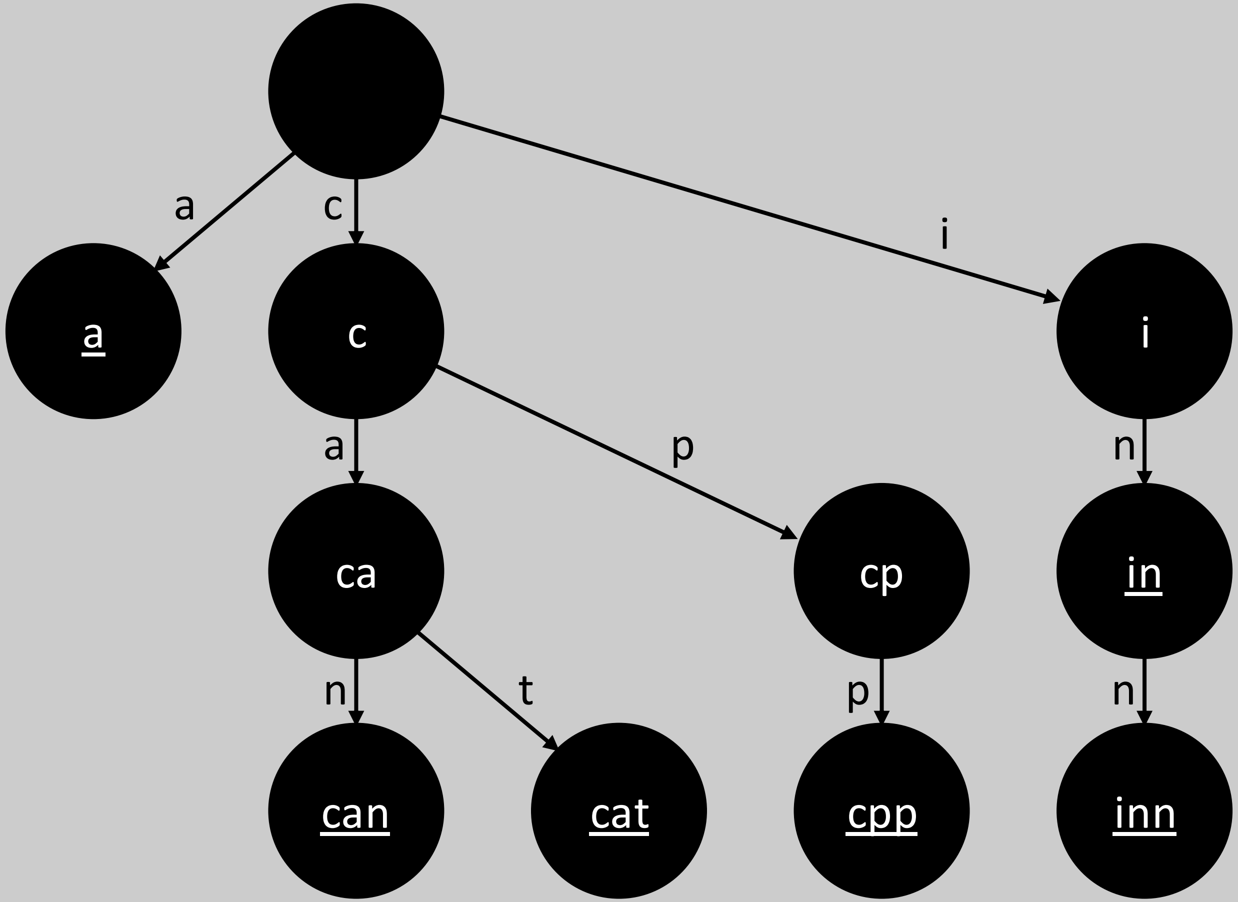
Find Value

Demonstration:

Finding string ‘cpr’ in tree

```
c p r
```

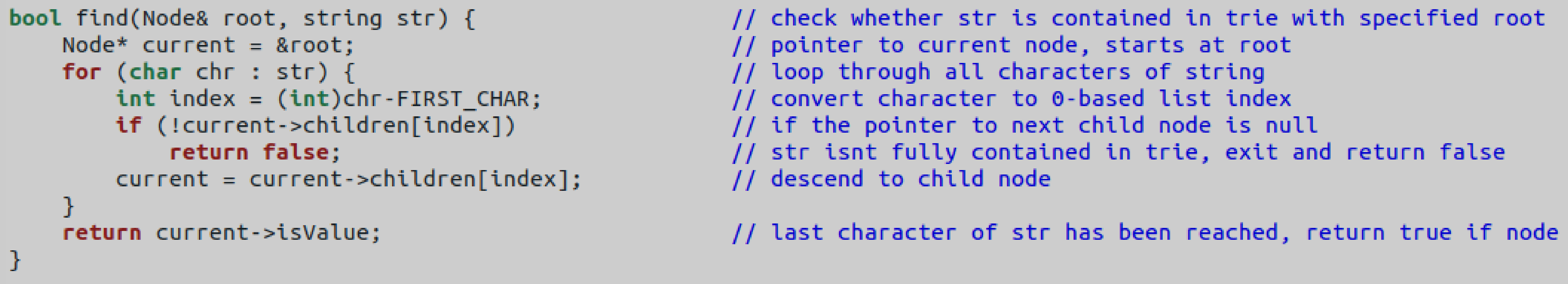
FALSE



Implementation

Find Value

Example:



```
bool find(Node& root, string str) {                   // check whether str is contained in trie with specified root
    Node* current = &root;                            // pointer to current node, starts at root
    for (char chr : str) {                            // loop through all characters of string
        int index = (int)chr-FIRST_CHAR;              // convert character to 0-based list index
        if (!current->children[index])                // if the pointer to next child node is null
            return false;                             // str isnt fully contained in trie, exit and return false
        current = current->children[index];           // descend to child node
    }
    return current->isValue;                          // last character of str has been reached, return true if node
}
```

Analysis

Time Complexity:

- Insert: O(L)
- Find: O(L)

Space Complexity: O(NL)

Example

Longest Prefix (IOI 1996)

Given a set of short strings `P` and a longer string `S`, calculate the length of the longest prefix of `S` such that the prefix equals to a concatenation of some (possibly repeated) elements of `P`

Sample IO:

```
Input                                              Output

A AB BA CA BBC                                     11
.
ABABACABAABC
```

Example

Longest Prefix (IOI 1996)

Solution:

We use a DP solution to find which characters are reachable by constructing a prefix from some elements of `P`. Let `DP[i]` denote whether it is possible to construct a prefix of length `i`. Initially, `DP[0] = true`. Firstly, let's construct a trie containing all the elements of `P`.

Now, loop through all `i` for which `DP[i]` is `true`, and for each loop we do the following:

- Start at the root of the trie
- Run a while loop with iterator `n`, and each time descend down the trie to character `S[i+n]` setting `DP[i+n] = true`. If the node doesn't exist, terminate the inner loop

Time complexity: $O(|S|^2)$

Questions?