# Tries & Prefix Trees in DSA: Beginner to Advanced Guide

**Subtitle:** Learn how to efficiently store and search strings for autocomplete, dictionary, and prefix-based problems.

**Website Name:** haas.dev
**Website Link:** https://dev-roast-app.vercel.app

## Introduction

A Trie (Prefix Tree) is a tree-like data structure used to store **strings in a way that allows fast prefix search**. It's widely used in **autocomplete, spell checkers, and dictionary problems**.

## Step 1: Trie Basics

- Each node represents a **character**
- Path from root → node = prefix
- **End of word marker** to indicate complete string
- Supports operations in **O(length of word)**

**Example (Node in JS):**

```
class TrieNode {
  constructor() {
    this.children = {};
    this.isEndOfWord = false;
  }
}

class Trie {
  constructor() { this.root = new TrieNode(); }
}
```

## Step 2: Insert a Word

```
function insert(trie, word) {
```

```
  let node = trie.root;
  for (let char of word) {
    if (!node.children[char]) node.children[char] = new TrieNode();
    node = node.children[char];
  }
  node.isEndOfWord = true;
}
```

**Example:** Insert "cat", "car", "dog"

# Step 3: Search a Word

```
function search(trie, word) {
  let node = trie.root;
  for (let char of word) {
    if (!node.children[char]) return false;
    node = node.children[char];
  }
  return node.isEndOfWord;
}
```

# Step 4: Prefix Search / Autocomplete

```
function startsWith(trie, prefix) {
  let node = trie.root;
  for (let char of prefix) {
    if (!node.children[char]) return false;
    node = node.children[char];
  }
  return true;
}
```

- Can also **collect all words starting with prefix** using DFS from node

## Step 5: Common Trie Problems

1. **Insert & Search words**
2. **Autocomplete / Suggestion systems**
3. **Longest Common Prefix among words**
4. **Word Search in 2D board using Trie**
5. **Prefix count / frequency of prefixes**

## Step 6: Mini Exercises

1. Count words starting with given prefix
2. Delete a word from Trie
3. Implement autocomplete for given prefix
4. Longest word in dictionary that can be built one character at a time

## Step 7: Common Mistakes

- Forgetting isEndOfWord → search fails
- Confusing prefix vs complete word
- Not handling overlapping prefixes
- Using array vs object incorrectly for children

## Step 8: Practice Plan for Tries

Day 1:

- Build trie, insert, search → 5 problems

Day 2:

- Prefix search, autocomplete → 5 problems

Day 3:

- Longest common prefix → 5 problems

Day 4–5:

- Word search / prefix count → 5–7 problems
- Mixed trie problems → 5–7 problems

## Key Takeaways

- Tries store **strings efficiently for prefix-based search**
- Operations are **O(word length)**, independent of number of words
- Essential for **autocomplete, dictionaries, and word problems**
- Practice helps **internalize traversal and prefix handling**

Visit haas.dev for more DSA trie guides, problem sets, and interview preparation resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app