Tries Data Structure

Understanding Tries: Implementation and Applications

What is a Trie?

- Trie (prefix tree) is a tree data structure used for storing strings efficiently.

- Used mainly in search applications such as autocomplete and dictionary lookups.

- Each node represents a character in a word.

Trie Structure

- Root node represents an empty string.

- Each edge represents a character.

- Nodes store a boolean flag to indicate the end of a word.

TrieNode

```
class TrieNode {
    TrieNode[] children;
    boolean isEndOfWord;

    public TrieNode() {
        children = new TrieNode[26]; // Assuming only
lowercase a-z
        isEndOfWord = false;
    }
}
```

Main Class

```
// Example Usage
public class Main {
    public static void main(String[] args) {
        Trie trie = new Trie();
        trie.insert("hello");
        System.out.println(trie.search("hello"));  //
Output: true
    }
}
```

Build the Trie

```
class Trie {
    private TrieNode root;

    public Trie() {
        root = new TrieNode();
    }

    public void insert(String word) {
        TrieNode node = root;
        for (char c : word.toCharArray()) {
            int index = c - 'a';
            if (node.children[index] == null) {
                node.children[index] = new TrieNode();
            }
            node = node.children[index];
        }
        node.isEndOfWord = true;
    }
```

Search in Java

```
 public boolean search(String word) {
        TrieNode node = root;
        for (char c : word.toCharArray()) {
            int index = c - 'a';
            if (node.children[index] == null) {
                return false;
            }
            node = node.children[index];
        }
        return node.isEndOfWord;
    }
}
```

Applications of Tries

- **Autocomplete & Dictionary Lookups**: Efficient prefix-based search.

- **Spell Checking**: Detecting misspelled words.

- **IP Routing**: Storing and querying routing tables.

- **DNA Sequencing**: Storing and matching DNA sequences.

- **Data Compression**: Storing common prefixes to reduce redundancy.

Conclusion

- Tries offer fast retrieval of strings.
- Useful in applications involving prefix searches.
- Alternative to Hash Maps for some problems where order matters.