

Longest Word with All Prefixes

Valid word = every prefix is also a word ■

● MEDIUM

■ Hook

Given a list of words, find the longest word such that every prefix of it is also in the list. If 'cat' is the answer, then 'c', 'ca', 'cat' must all exist. Trie naturally tracks this: just check isEnd at every node along the path.

■ Core Intuition

Build a Trie. Then BFS/DFS from root, but only follow edges where the child node has isEnd=true. The deepest node reached (longest path of isEnd=true nodes) is the answer.

■ Code

```
public String longestWord(String[] words) {
    Trie trie = new Trie();
    for (String w : words) trie.insert(w);
    String result = "";
    // DFS using stack
    Deque<TrieNode> stack = new ArrayDeque<>();
    Deque<String> path = new ArrayDeque<>();
    stack.push(trie.root); path.push("");
    while (!stack.isEmpty()) {
        TrieNode node = stack.pop();
        String word = path.pop();
        if (word.length() > result.length() ||
            (word.length() == result.length() && word.compareTo(result) < 0))
            result = word;
        for (int i = 25; i >= 0; i--) {
            if (node.children[i] != null && node.children[i].isEnd) {
                stack.push(node.children[i]);
                path.push(word + (char)('a' + i));
            }
        }
    }
    return result;
}
```

■ TIME

$O(N \times L)$

■ SPACE

$O(N \times L)$

- ✓ Build Trie, then DFS only through isEnd=true nodes
- ✓ Every prefix of the answer must itself be a complete word
- ✓ Track longest (and lexicographically smallest on tie)
- ✓ Alternative: sort words, use HashSet of seen words
- ✓ Time: $O(NL)$ | Space: $O(NL)$

■ MENTAL MODEL — Remember This Forever

The Trie enforces the 'every prefix is a word' constraint naturally. A node can only be visited if its parent was isEnd=true. So any path you reach in this constrained traversal is guaranteed to have all prefixes as words.