720. Longest Word in Dictionary

Given a list of strings words representing an English Dictionary, find the longest word in words that can be built one character at a time by other words in words. If there is more than one possible answer, return the longest word with the smallest lexicographical order.

If there is no answer, return the empty string.

Example 1:

```
Input:
words = ["a", "banana", "app", "appl", "ap", "apply", "apple"]
Output: "apple"
Explanation:
Both "apply" and "apple" can be built from other words in the dictionary. However, "a
pple" is lexicographically smaller than "apply".
```

```
class Solution {
    public String longestWord(String[] words) {

        // trie, a prefix tree
        Trie trie = new Trie();
        int index = 0;
        for (String word : words) {
            trie.insert(word, ++index);  // index from 1
        }
        trie.words = words;
        return trie.dfs();
    }
}

class Node {
    char c;
    HashMap<Character, Node> children = new HashMap<>();
    int end;

    public Node (char c) {
        this.c = c;
    }
}

class Trie {
    Node root;
    String[] words;

    public Trie() {
        root = new Node('0');
    }

    public void insert(String word, int index) {
        Node cur = root;
        for (char c : word.toCharArray()) {
            cur.children.putIfAbsent(c, new Node(c));
            cur = cur.children.get(c);
        }
        cur.end = index;
    }

    public String dfs() {
        String ans = "";
        Stack<Node> stack = new Stack<>();
        stack.push(root);

        while (!stack.isEmpty()) {
            Node node = stack.pop();

            if (node.end > 0 || node == root) {
                if (node != root) {
                    String word = words[node.end -1];
                    if (word.length() > ans.length() ||
                       word.length() == ans.length() && word.compareTo(ans) < 0) {
                        ans = word;
                    }
                }

                for (Node n : node.children.values())
                    stack.push(n);
            }
        }

        return ans;
    }

}
```