AlgoExpert Quad Layout Java 12px Sublime Monokai 00:00:00

Prompt Scratchpad Our Solution(s) Video Explanation Run Code

Solution 1

```java

  public static void explore(
      int i,
      int j,
      char[][] board,
      TrieNode trieNode,
      boolean[][] visited,
      Set<String> finalWords) {
    if (visited[i][j]) {
      return;
    }
    char letter = board[i][j];
    if (!trieNode.children.containsKey(letter)) {
      return;
    }
    visited[i][j] = true;
    trieNode = trieNode.children.get(letter);
    if (trieNode.children.containsKey('*')) {
      finalWords.add(trieNode.word);
    }
    List<Integer[]> neighbors = getNeighbors(i, j, board);
    for (Integer[] neighbor : neighbors) {
      explore(neighbor[0], neighbor[1], board, trieNode, visited, finalWords);
    }
    visited[i][j] = false;
  }

  public static List<Integer[]> getNeighbors(int i, int j, char[][] board) {
    List<Integer[]> neighbors = new ArrayList<Integer[]>();
    if (i > 0 && j > 0) {
      neighbors.add(new Integer[] {i - 1, j - 1});
    }
    if (i > 0 && j < board[0].length - 1) {
      neighbors.add(new Integer[] {i - 1, j + 1});
    }
    if (i < board.length - 1 && j < board[0].length - 1) {
      neighbors.add(new Integer[] {i + 1, j + 1});
    }
    if (i < board.length - 1 && j > 0) {
      neighbors.add(new Integer[] {i + 1, j - 1});
    }
    if (i > 0) {
      neighbors.add(new Integer[] {i - 1, j});
    }
    if (i < board.length - 1) {
      neighbors.add(new Integer[] {i + 1, j});
    }
    if (j > 0) {
      neighbors.add(new Integer[] {i, j - 1});
    }
    if (j < board[0].length - 1) {
      neighbors.add(new Integer[] {i, j + 1});
    }
    return neighbors;
  }

  static class TrieNode {
    Map<Character, TrieNode> children = new HashMap<Character, TrieNode>();
    String word = "";
  }

  static class Trie {
    TrieNode root;
    char endSymbol;

    public Trie() {
      this.root = new TrieNode();
      this.endSymbol = '*';
    }

    public void add(String str) {
      TrieNode node = this.root;
      for (int i = 0; i < str.length(); i++) {
        char letter = str.charAt(i);
        if (!node.children.containsKey(letter)) {
          TrieNode newNode = new TrieNode();
          node.children.put(letter, newNode);
        }
        node = node.children.get(letter);
      }
      node.children.put(this.endSymbol, null);
      node.word = str;
    }
  }
}

```