Solution 1

```
    foreach (string key in finalWords) {
      finalWordsArray.Add(key);
    }
    return finalWordsArray;
  }

  public static void explore(int i, int j, char[,] board, TrieNode trieNode, bool[,] visited,
    HashSet<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[letter];
    if (trieNode.children.ContainsKey('*')) {
      finalWords.Add(trieNode.word);
    }
    List<int[]> neighbors = getNeighbors(i, j, board);
    foreach (int[] neighbor in neighbors) {
      explore(neighbor[0], neighbor[1], board, trieNode, visited, finalWords);
    }
    visited[i,j] = false;
  }

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

  public class TrieNode {
    public Dictionary<char, TrieNode> children = new Dictionary<char, TrieNode>();
    public string word = "";
  }

  public class Trie {
    public TrieNode root;
    public 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[i];
        if (!node.children.ContainsKey(letter)) {
          TrieNode newNode = new TrieNode();
          node.children.Add(letter, newNode);
        }
        node = node.children[letter];
      }
      node.children[this.endSymbol] = null;
      node.word = str;
    }
  }
}
```