AlgoExpert Quad Layout C++ 12px Sublime Monokai 00:00:00

Prompt Scratchpad Our Solution(s) Video Explanation Run Code

Solution 1 Solution 2 Solution 3

73

```
_{\rm 1} \, // Copyright @ 2020 AlgoExpert, LLC. All rights reserved.
   #include <vector>
   #include <unordered_map>
   using namespace std;
   class TrieNode {
   public:
      unordered_map<char, TrieNode *> children;
10
      string word;
11 };
12
13
   class Trie {
14
    public:
15
      TrieNode *root;
      char endSymbol;
16
17
18
      Trie() {
19
       this->root = new TrieNode();
20
        this->endSymbol = '*';
21
22
23
      void insert(string str) {
24
        TrieNode *current = this->root;
25
        for (int i = 0; i < str.length(); i++) {</pre>
26
          char letter = str[i];
          if (current->children.find(letter) == current->children.end()) {
27
28
            TrieNode *newNode = new TrieNode();
29
            current->children.insert({letter, newNode});
30
31
          current = current->children[letter];
32
33
        current->children.insert({this->endSymbol, NULL});
34
        current->word = str;
35
36
    };
37
    void findSmallStringsIn(string str, int startIdx, Trie *trie,
38
39
                             unordered_map<string, bool> *containedStrings);
40
    // O(ns + bs) time | O(ns) space
41
    vector<bool> multiStringSearch(string bigString, vector<string> smallStrings) {
43
      Trie *trie = new Trie();
      for (string smallString : smallStrings) {
44
        trie->insert(smallString);
45
46
      unordered_map<string, bool> containedStrings;
47
      for (int i = 0; i < bigString.length(); i++) {</pre>
48
49
        \verb|findSmallStringsIn| (\verb|bigString|, i, trie|, \&containedStrings|); \\
50
51
      vector<bool> solution;
52
      for (string smallString : smallStrings) {
53
        solution.push_back(containedStrings.find(smallString) !=
54
                            containedStrings.end());
55
56
      return solution;
57
58
59
    \begin{tabular}{ll} \textbf{void findSmallStringsIn} (\textbf{string str, int startIdx, Trie *trie, } \\ \end{tabular}
60
                             unordered_map<string, bool> *containedStrings) {
      TrieNode *currentNode = trie->root;
61
      for (int i = startIdx; i < str.length(); i++) {</pre>
62
63
        if (currentNode->children.find(str[i]) == currentNode->children.end()) {
64
          break;
65
66
        currentNode = currentNode->children[str[i]];
67
        if (currentNode->children.find(trie->endSymbol) !=
68
            currentNode->children.end()) {
69
          containedStrings->insert({currentNode->word, true});
70
71
72 }
```