```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <vector>
#include <unordered_map>
using namespace std;

class TrieNode {
public:
  unordered_map<char, TrieNode *> children;
  string word;
};

class Trie {
public:
  TrieNode *root;
  char endSymbol;

  Trie() {
    this->root = new TrieNode();
    this->endSymbol = '*';
  }

  void insert(string str) {
    TrieNode *current = this->root;
    for (int i = 0; i < str.length(); i++) {
      char letter = str[i];
      if (current->children.find(letter) == current->children.end()) {
        TrieNode *newNode = new TrieNode();
        current->children.insert({letter, newNode});
      }
      current = current->children[letter];
    }
    current->children.insert({this->endSymbol, NULL});
    current->word = str;
  }
};

void findSmallStringsIn(string str, int startIdx, Trie *trie,
                        unordered_map<string, bool> *containedStrings);

// O(ns + bs) time | O(ns) space
vector<bool> multiStringSearch(string bigString, vector<string> smallStrings) {
  Trie *trie = new Trie();
  for (string smallString : smallStrings) {
    trie->insert(smallString);
  }
  unordered_map<string, bool> containedStrings;
  for (int i = 0; i < bigString.length(); i++) {
    findSmallStringsIn(bigString, i, trie, &containedStrings);
  }
  vector<bool> solution;
  for (string smallString : smallStrings) {
    solution.push_back(containedStrings.find(smallString) !=
                       containedStrings.end());
  }
  return solution;
}

void findSmallStringsIn(string str, int startIdx, Trie *trie,
                        unordered_map<string, bool> *containedStrings) {
  TrieNode *currentNode = trie->root;
  for (int i = startIdx; i < str.length(); i++) {
    if (currentNode->children.find(str[i]) == currentNode->children.end()) {
      break;
    }
    currentNode = currentNode->children[str[i]];
    if (currentNode->children.find(trie->endSymbol) !=
        currentNode->children.end()) {
      containedStrings->insert({currentNode->word, true});
    }
  }
}
```