

Our Solution(s)

Run Code

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 import java.util.*;
4
5 class Program {
6     static class TrieNode {
7         Map<Character, TrieNode> children = new HashMap<Character,
8     }
9
10    static class SuffixTrie {
11        TrieNode root = new TrieNode();
12        char endSymbol = '*';
13
14        public SuffixTrie(String str) {
15            populateSuffixTrieFrom(str);
16        }
17
18        // O(n^2) time | O(n^2) space
19        public void populateSuffixTrieFrom(String str) {
20            for (int i = 0; i < str.length(); i++) {
21                insertSubstringStartingAt(i, str);
22            }
23        }
24
25        public void insertSubstringStartingAt(int i, String str) {
26            TrieNode node = root;
27            for (int j = i; j < str.length(); j++) {
28                char letter = str.charAt(j);
29                if (!node.children.containsKey(letter)) {
30                    TrieNode newNode = new TrieNode();
31                    node.children.put(letter, newNode);
32                }
33                node = node.children.get(letter);
```

Your Solutions

Run Code

Solution 1

Solution 2

Solution 3

```
1 import java.util.*;
2
3 class Program {
4     // Do not edit the class below except for the
5     // populateSuffixTrieFrom and contains methods.
6     // Feel free to add new properties and methods
7     // to the class.
8     static class TrieNode {
9         Map<Character, TrieNode> children = new HashMap<Character,
10    }
11
12    static class SuffixTrie {
13        TrieNode root = new TrieNode();
14        char endSymbol = '*';
15
16        public SuffixTrie(String str) {
17            populateSuffixTrieFrom(str);
18        }
19
20        public void populateSuffixTrieFrom(String str) {
21            // Write your code here.
22        }
23
24        public boolean contains(String str) {
25            // Write your code here.
26            return false;
27        }
28    }
29 }
30
```

```
1 class IngressTest {
2     String word = "test";
3     Ingress sufficient = new Ingress sufficient();
4     sufficient test() {
5         return sufficient;
6     }
7     String word = "test";
8     Ingress sufficient = new Ingress sufficient();
9     sufficient test() {
10         return sufficient;
11     }
12     String word = "test";
13     Ingress sufficient = new Ingress sufficient();
14     sufficient test() {
15         return sufficient;
16     }
17     String word = "test";
18     Ingress sufficient = new Ingress sufficient();
19     sufficient test() {
20         return sufficient;
21     }
22     String word = "test";
23     Ingress sufficient = new Ingress sufficient();
24     sufficient test() {
25         return sufficient;
26     }
27 }
```

Run or submit code when you're ready.