Our Solution(s) Run Code

Solution 1

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

#include <unordered_map>
using namespace std;

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

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

  SuffixTrie(string str) {
    this->root = new TrieNode();
    this->endSymbol = '*';
    this->populateSuffixTrieFrom(str);
  }

  // O(n^2) time | O(n^2) space
  void populateSuffixTrieFrom(string str) {
    for (int i = 0; i < str.length(); i++) {
      this->insertSubstringStartingAt(i, str);
    }
  }

  void insertSubstringStartingAt(int i, string str) {
    TrieNode *node = this->root;
    for (int j = i; j < str.length(); j++) {
      char letter = str[j];
      if (node->children.find(letter) == node->children.end()) {
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```cpp
#include <unordered_map>
using namespace std;

// Do not edit the class below except for the
// populateSuffixTrieFrom and contains methods.
// Feel free to add new properties and methods
// to the class.
class TrieNode {
public:
  unordered_map<char, TrieNode *> children;
};

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

  SuffixTrie(string str) {
    this->root = new TrieNode();
    this->endSymbol = '*';
    this->populateSuffixTrieFrom(str);
  }

  void populateSuffixTrieFrom(string str) {
    // Write your code here.
  }

  bool contains(string str) {
    // Write your code here.
    return false;
  }
};
```

Our Tests

Custom Output Submit Code

```
public:
  TrieNode *root;
  char endSymbol;

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

  void populateSuffixTreeFrom(string str) {
    for (int i = 0; i < str.length(); i++) {
      this->insertSubstringStartingAt(i, str);
    }
  }

  void insertSubstringStartingAt(int i, string str) {
    TrieNode *node = this->root;
    for (int j = i; j < str.length(); j++) {
      char letter = str[j];
      if (node->children.find(letter) == node->children.end()) {
```

Run or submit code when you're ready.