Difficulty: **Category:** Tries **Successful Submissions:** 9,901+

Suffix Trie Construction

Write a `SuffixTrie` class for a Suffix-Trie-like data structure. The class should have a `root` property set to be the root node of the trie and should support:

- Creating the trie from a string; this will be done by calling the `populateSuffixTrieFrom` method upon class instantiation, which should populate the `root` of the class.
- Searching for strings in the trie.

Note that every string added to the trie should end with the special `endSymbol` character: `"*"`.

If you're unfamiliar with Suffix Tries, we recommend watching the Conceptual Overview section of this question's video explanation before starting to code.

Sample Input (for creation)

```
string = "babc"
```

Sample Output (for creation)

```
The structure below is the root of the trie.
{
  "c": {"*": true},
  "b": {
    "c": {"*": true},
    "a": {"b": {"c": {"*": true}}},
  },
  "a": {"b": {"c": {"*": true}}},
}
```

Sample Input (for searching in the suffix trie above)

```
string = "abc"
```

Sample Output (for searching in the suffix trie above)

```
true
```

Hints

Hint 1

Building a suffix-trie-like data structure consists of essentially storing every suffix of a given string in a trie. To do so, iterate through the input string one character at a time and insert every substring starting at each character and ending at the end of the string into the trie.

Hint 2

To insert a string into the trie, start by adding the first character of the string into the root node of the trie and mapping it to an empty hash table if it isn't already there. Then, iterate through the rest of the string inserting each of the remaining characters into the previous character's corresponding node (or hash table) in the trie, making sure to add an endSymbol "*" at the end.

Hint 3

Searching the trie for a specific string should follow a nearly identical logic to the one used to add a string in the trie.

Optimal Space & Time Complexity

Creation: O(n^2) time | O(n^2) space - where n is the length of the input string Searching: O(m) time | O(1) space - where m is the length of the input string