< Back

# Clean Python Trie Solution

BoboBiggins ★ 15 August 27, 2019 2:31 AM 530 VIEWS

9

```
class TrieNode:
    def __init__(self):
        self.children, self.is_word = {}, False

    @staticmethod
    def construct_trie(words):
        root = TrieNode()
        for word in words:
            node = root
            for c in word:
                node.children.setdefault(c, TrieNode())
                node = node.children[c]
            node.is_word = True
        return root

class Solution:
    def indexPairs(self, text, words):
        res, trie = [], TrieNode.construct_trie(words)
        for l in range(len(text)):
            node = trie
            for r in range(l, len(text)):
                if text[r] not in node.children:
                    break
                node = node.children[text[r]]
                if node.is_word:
                    res.append((l, r))
        return res
```

Comments: 3 Best Most Votes Newest to Oldest Oldest to Newest

Type comment here... (Markdown is supported)

Post

JummyEgg ★ 151 Last Edit: March 7, 2020 1:24 PM

Great solution ! Here is my 2 Pythonic solutions:

- Iterative solution

```
import functools
def indexPairs(self, text: str, words: List[str]) -> List[List[int]]:
    Trie = lambda: collections.defaultdict(Trie)
    trie = Trie()
    for word in words:
        functools.reduce(dict.__getitem__, word, trie)["END"] = word
```

Read More

0 Reply

Ord ★ 1 November 19, 2019 1:39 PM

hey, great solution! notice that you can use `dict`'s `setdefault` and `get` to save some lookups:

```
class Solution:
    def indexPairs(self, text: str, words: List[str]) -> List[List[int]]:
        # n - len(text)
        # m - sum(len(word))

        # build words trie - O(m)
        class TrieNode:
            def __init__(self):
```