Solution Review: Word Formation From a Dictionary Using Trie

This review provides a detailed analysis of the solution to the Word Formation From a Dictionary Using a Trie Challenge.

We'll cover the following

Solution: Iterative Word Matching

main.py

Trie.py

TrieNode.py

```
from Trie import Trie
from TrieNode import TrieNode

def is_formation_possible(dct, word):

    # Create Trie and insert dctionary elements in it
    trie = Trie()
    for x in range(len(dct)):
        trie.insert(dct[x])

    # Get Root
    current_node = trie.root

    # Iterate all the letters of the word
    for i in range(len(word)):
        # get index of the character from Trie
        char = trie.get_index(word[i])

        # if the prefix of word does not exist, word would not
        if current_node.children[char] is None:
            return False

        # if the substring of the word exists as a word in tri
        # check whether rest of the word also exists,
        # if it does return true
        elif current_node.children[char].is_end_word:
            if trie.search(word[i+1:]):
                return True

        current_node = current_node.children[char]

    return False

keys = ["the", "hello", "there", "answer",
        "any", "educative", "world", "their", "abc"]
print(is_formation_possible(keys, "helloworld"))

```

The algorithm can be divided into three parts. The first and simplest part is making a trie for the words in the dictionary.

The second part is to check if there is a `word` in the trie which can become a prefix for the query word. In the case of `“helloworld”`, we can find `“hello”` in the trie. Since there can be multiple prefixes of a word, we have to check for every such prefix. As we iterate through the trie, looking for prefix, whenever we find a prefix that exists as a word in the trie, we lookup the remaining word in the trie using the search function. If this substring exists we have found a solution

Time Complexity

We perform the insert operation **m** times for a dictionary of size **m**. After that, the search operation runs on the `word` in the sequence:

```
"h", "he", "hel", "hell"...
```

If the size of the word is **n**, the complexity for this turns out to be n^2. Hence, the total time complexity is $O(m + n^2)$. We will solve this challenge again in the hashing chapter (https://www.educative.io/collection/page/5642554087309312/5634727314718720/5710682603388928/).

Back

Challenge 4: Word Formation From a ...

Next

Trie Quiz: Test Your Understanding of ...

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/solution-review-word-formation-from-a-dictionary-using-trie__trie__data-structures-for-coding-interviews-in-python)