Lab#2, NLP@CGU Spring 2023

This is due on 2023/03/13 15:30, commit to your github as a PDF (lab2.pdf) (File>Print>Save as PDF).

IMPORTANT: After copying this notebook to your Google Drive, please paste a link to it below. To get a publicly-accessible link, hit the *Share* button at the top right, then click "Get shareable link" and copy over the result. If you fail to do this, you will receive no credit for this lab!

LINK: paste your link here

https://colab.research.google.com/drive/1klD1tQvLwTC0ZeuzRg71WYuTzaMvkYEK?usp=share_link

---

Student ID: B0928026

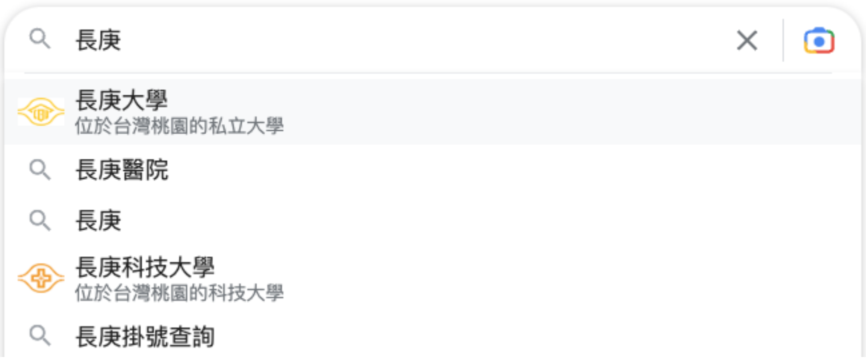
Name: 洪詩晴

Question 1 (100 points)

Implementing Trie in Python.

Trie is a very useful data structure. It is commonly used to represent a dictionary for looking up words in a vocabulary.

For example, consider the task of implementing a search bar with auto-completion or query suggestion. When the user enters a query, the search bar will automatically suggests common queries starting with the characters input by the user.



按兩下 (或按 Enter 鍵) 即可編輯

```
# YOUR CODE HERE!
# IMPLEMENTIG TRIE IN PYTHON

class TrieNode:

    def __init__(self, char):
        self.char = char
        self.value = char
        self.children = []
        self.finished = False
        self.counter = 0

class Trie(object):

    def __init__(self):
        self.root = TrieNode("")

    def insert(self, word):
        node = self.root
        for char in word:
            found_in_child = False
            for child in node.children:
                if child.char == char:
                    node = child
                    found_in_child = True
                    break
            if not found_in_child:
                new_node = TrieNode(char)
                node.children.append(new_node)
                node = new_node
        node.finished = True
        node.counter += 1
    def dfs(self, node, prefix, result):
        if node.finished:
            result.append((prefix, node.counter))
        for child in node.children:
            self.dfs(child, prefix + child.char, result)

    def query(self, x):
        node = self.root
        result = []
        for char in x:
            char_not_found = True
            for child in node.children:
                if child.char == char:
                    node = child
                    char_not_found = False
                    break
            if char_not_found:
                return []
        self.dfs(node, x, result)
        return result

# # DO NOT MODIFY THE VARIABLES
obj = Trie()
obj.insert("長庚資工")
obj.insert("長大")
obj.insert("長庚")
obj.insert("長庚")
obj.insert("長庚大學")
obj.insert("長庚科技大學")

# # DO NOT MODIFY THE BELOW LINE!
# # THE RESULTS : [(words, count), (words, count)]
print(obj.query("長"))
# [('長庚', 2), ('長庚資工', 1), ('長庚大學', 1), ('長庚科技大學', 1), ('長大', 1)]

print(obj.query("長庚"))
# [('長庚', 2), ('長庚資工', 1), ('長庚大學', 1), ('長庚科技大學', 1)]
```

```
[('長庚', 2), ('長庚資工', 1), ('長庚大學', 1), ('長庚科技大學', 1), ('長大', 1)]
[('長庚', 2), ('長庚資工', 1), ('長庚大學', 1), ('長庚科技大學', 1)]
```