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/1-GioZOKSL0MA3umXs18fufvoAZ5GeLGm--

Student ID:B0928029

Name:毛謙鎧

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.

編集するにはダブルクリックするか Enter キーを押してください

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

class TrieNode:

    def __init__(self, char):
        self.char = char
        self.children = {}  # add children dictionary to store child nodes
        self.count = 0

class Trie(object):

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

    def insert(self, word):
        curr_node = self.root
        for char in word:
         if char not in curr_node.children:
                curr_node.children[char] = TrieNode(char)
                curr_node = curr_node.children[char]
        curr_node.count += 1

    def dfs(self, node, prefix):
         if node.count > 0:
            results.append((prefix, node.count))

         for child in node.children.values():
            self.dfs(child, prefix + child.char, results)

    def query(self, x):
        curr_node = self.root
        for char in x:
            if char not in curr_node.children:
```

```
                return []
            curr_node = curr_node.children[char]

        results = []
        self.dfs(curr_node, x, results)
        results.sort(key=lambda x: -x[1])
        return results

# 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)]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-96306a9363b7> in <module>
     52 # # DO NOT MODIFY THE BELOW LINE!
     53 # # THE RESULTS : [(words, count), (words, count)]
---> 54 print(obj.query("長"))
     55 # [('長庚', 2), ('長庚資工', 1), ('長庚大學', 1), ('長庚科技大學', 1), ('長大', 1)]
     56

<ipython-input-6-96306a9363b7> in query(self, x)
     37
     38         results = []
---> 39         self.dfs(curr_node, x, results)
     40         results.sort(key=lambda x: -x[1])
     41         return results

TypeError: dfs() takes 3 positional arguments but 4 were given
```

SEARCH STACK OVERFLOW