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/1xOchl96coCllrkK3d8BqPQfT1qQB-bJN?usp=sharing

---

Student ID:B0929038 **Name**:吳安捷

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.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:
                if char in node.children:
                    node=node.children[char]
                else:
                    new_node=TrieNode(char)
                    node.children[char]=new_node
                    node=new_node
            node.finished=True
            node.counter+=1

        def dfs(self, node, prefix):
            if node.finished:
                self.output.append((prefix+node.char,node.counter))
            for child in node.children.values():
                self.dfs(child,prefix+node.char)

        def query(self, x):
            self.output=[]
```

```
            node=self.root
            for  char  in  x:
                if  char  in  node.children:
                    node=node.children[char]
                else:
                    return  []
            self.dfs(node,x[:-1])
            return  sorted(self.output,key=lambda  x:x[1],reverse=True)

# # 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)]
```


✓ 0 秒 完成時間：下午2:54