

第2次作業-作業-HW2

學號：112112105

姓名：李佩琪

作業撰寫時間：170 (mins，包含程式撰寫時間)

最後撰寫文件日期：2024/10/27

本份文件包含以下主題：(至少需下面兩項，若是有多者可以自行新增)

- ☒ 說明內容
- ☒ 個人認為完成作業須具備觀念

說明程式與內容

開始寫說明，該說明需說明想法，並於之後再對上述想法的每一部分將程式進一步進行展現，若需引用程式區則使用下面方法，若為.cs檔內程式除了於敘述中需註明檔案名稱外，還需使用語法```語言種類 程式碼```，其中語言種類若是要用python則使用py，java則使用java，C/C++則使用cpp，下段程式碼為語言種類選擇csharp使用後結果：

```
public void mt_getResult(){  
    ...  
}
```

若要於內文中標示部分網頁檔，則使用以下標籤```html 程式碼```，下段程式碼則為使用後結果：

```
<%@ Page Language="C#" AutoEventWireup="true" ...>  
  
<!DOCTYPE html>  
  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head runat="server">  
<meta http-equiv="Content-Type" ...>  
    <title></title>  
</head>  
<body>  
    <form id="form1" runat="server">  
        <div>  
            </div>  
    </form>  
</body>  
</html>
```

更多markdown方法可參閱<https://ithelp.ithome.com.tw/articles/10203758>

請在撰寫"說明程式與內容"該塊內容，請把原該塊內上述敘述刪除，該塊上述內容只是用來指引該怎麼撰寫內容。

1. 問題如下圖所述，並回答下面問題。

Ans:

```
from typing import List
def getResult():
    alphabet1: List[List[str]] = [
        ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0'],
        ['Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P'],
        ['A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', ';'],
        ['Z', 'X', 'C', 'V', 'B', 'N', 'M', ',', '.', '/']
    ]
    alphabet2: List[List[str]] = [
        ['!', '@', '#', '$', '%', '^', '&', '*', '(', ')'],
        ['Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P'],
        ['A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', ':'],
        ['Z', 'X', 'C', 'V', 'B', 'N', 'M', '<', '>', '?']
    ]
    n = int(input("輸入資料組數: "))
    repeat = set()
    def find_and_print(value, direction, alphabet):
        for j in range(len(alphabet)):
            if value in alphabet[j]:
                k = alphabet[j].index(value)
                new_j, new_k = j, k
                if direction == 1 and j > 0:
                    new_j -= 1
                elif direction == 2 and j < len(alphabet) - 1:
                    new_j += 1
                elif direction == 3 and k < len(alphabet[j]) - 1:
                    new_k += 1
                elif direction == 4 and k > 0:
                    new_k -= 1
                else:
                    continue # 若移動超出邊界則跳過
                ans = alphabet[new_j][new_k]
                if ans not in repeat:
                    print(ans)
                    repeat.add(ans)
                break
    for _ in range(n):
        value, direction = input("輸入按鍵與方向: ").split()
        direction = int(direction)
        find_and_print(value, direction, alphabet1)
        find_and_print(value, direction, alphabet2)
    getResult()
```

2. 給定一個包含 n 個不同數字的數組，這些數字的範圍是從 0 到 n 。找出數組中缺失的那一個數字。 Ans:

```
# 接收使用者輸入，例如：nums = [3, 0, 1]
num = input("請輸入數組(格式為 nums = [3, 0, 1]): ")
# 解析輸入的數組，只取出等號後的數組部分，並轉換為列表
nums = eval(num.split('=')[1].strip())
# 取得數組的長度
n = len(nums)
# 計算從 0 到 n 的總和
sum1 = n * (n + 1) / 2
# 計算數組中所有數字的總和
sum2 = sum(nums)
# 取得缺少的數字
miss = sum1 - sum2
# 輸出缺少的數字，轉為整數形式
print("缺少的數字為:", int(miss))
```

3. 請回答下面問題：

Ans:

a. 成立

將 $T(n)=2^n+1$ 與 2^n 比較：

在 $T(n)=2^n+1$ 中，常數1是一個很小的額外項，相較於 2^n 在 n 很大時的增長率，它變得可以忽略不計。可以寫成：

$$2^n+1 \leq 2 \cdot 2^n$$

驗證大 O 條件：

對於所有 $n \geq 1$ 成立。因此，只要我們設定 $c=2$ ，就有 $T(n) \leq c \cdot 2^n$

選擇 $c=2$ 和 $n_0=1$ ，對於所有 $n \geq n_0$ ，不等式 $T(n)=2^n+1 \leq 2 \cdot 2^n$ 成立，因此根據大O定義，我們可以說 $T(n)=O(2^n)$

結論：

當 $T(n)=2^n+1$ 時，其時間複雜度為 $O(2^n)$

b. 不成立

將 2^{2^n} 表示為 2^n 的冪次方形式：

$$2^{2^n} = (2^n)^2 \text{ 也就是 } 2^{2^n} \text{ 等於 } 2^n \text{ 的平方。}$$

增長速度比較：

由於 $T(n)=2^{2^n}$ 等於 2^n 的平方，因此 $T(n)$ 的增長速度遠大於 2^n 。這意味著即使 n 稍大， 2^{2^n} 的數值也會大幅超過 2^n 。

反證法：

若 $T(n)$ 的時間複雜度為 $O(2^n)$ ，則應存在常數 c 和 n_0 使得當 $n \geq n_0$ 時，有： $T(n) \leq c \cdot 2^n$ ，即 $2^{2^n} \leq c \cdot 2^n$

兩邊同除 2^n 後得到： $2^n \leq c$

這個不等式對於任何固定的常數 c 都無法成立，因為當 n 增加時， 2^n 可以變得無限大。

結論：

當執行次數為 2^{2^n} 時，其時間複雜度不是 $O(2^n)$ 。實際上， $T(n)=2^{2^n}$ 的時間複雜度應為 $O(2^{2^n})$ 。

4. 請問以下各函式，在進行呼叫後，請計算(1)執行次數 $T(n)$ ，並(2)透過執行次數判斷時間複雜度為何(請用 Big-Oh 進行表示)？ Ans:

a.

(1) 執行次數 $n(n+1)/2$

(2) 時間複雜度 $T(n)=O(n^2)$

b.

(1) 執行次數 T 約為 $\lfloor \log_2(n) \rfloor + 1$

(2) 時間複雜度為 $O(\log n)$

c.

(1) 執行次數 $T(n, m) = (\lfloor \log_2(n) \rfloor + 1) \times m$

(2) 時間複雜度為 $O(m \log n)$

d.

(1) 執行次數 $T(n, m) = O(m \log n)$ 。

(2) 時間複雜度為 $O(m \log n)$ ，這意味著當 n 和 m 都增長到無窮大時，這段程式碼的運行時間會隨之增長。

個人認為完成作業須具備觀念

開始寫說明，需要說明本次練習需學會那些觀念 (需寫成文章，需最少50字，並且文內不得有你、我、他三種文字) 且必須提供完整與練習相關過程的 notion 筆記連結

時間複雜度的概念：了解時間複雜度的定義及其重要性。

執行次數的計算：學習如何計算演算法中各個函式的執行次數。

多層迴圈的分析：掌握多層迴圈對執行次數的影響，並進行分析。

大 O 符號表示法：學會使用大 O 符號表示演算法的時間複雜度。

效能優化：認識如何透過時間複雜度分析來優化程式的執行效率