

## 第2次作業-作業-HW2

學號：112111225

姓名：林芷羽

作業撰寫時間：180 (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():
    # 鍵盤下排
    alphabet1: List[List[chr]] = [
        ['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', ',', '.', '/'],
    ]
    # 鍵盤上排
    alphabet2: List[List[chr]] = [
        ['!', '@', '#', '$', '%', '^', '&', '*', '(', ')'],
        ['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("請輸入測試筆數: "))
    repeat = set() # 用來追蹤已經印出的符號，避免重複

    for i in range(n):
        value, direction = input("請輸入符號和方向(用空格隔開)(1:上，2:下，3:右，4:左): ").split()
        direction = int(direction)

        # 在 alphabet1 中尋找
        for j in range(len(alphabet1)):
            if value in alphabet1[j]:
                k = alphabet1[j].index(value)

                if direction == 1:
                    newValue = alphabet1[j - 1][k]
                elif direction == 2:
                    newValue = alphabet1[j + 1][k]
                elif direction == 3:
                    newValue = alphabet1[j][k + 1]
                elif direction == 4:
                    newValue = alphabet1[j][k - 1]
                else:
                    newValue = value

                if newValue not in repeat:
                    print(newValue)
                    repeat.add(newValue)

        # 如果沒在 alphabet1 中找到，則在 alphabet2 中尋找
        for j in range(len(alphabet2)):
            if value in alphabet2[j]:
```

```

        k = alphabet2[j].index(value)

        if direction == 1:
            newValue = alphabet2[j - 1][k]
        elif direction == 2:
            newValue = alphabet2[j + 1][k]
        elif direction == 3:
            newValue = alphabet2[j][k + 1]
        elif direction == 4:
            newValue = alphabet2[j][k - 1]
        else:
            newValue = value

        if newValue not in repeat:
            print(newValue)
            repeat.add(newValue)

    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:

```

def number(nums):
    n = len(nums)
    sum1 = n * (n + 1) // 2 # 完整範圍的總和
    sum2 = sum(nums) # 數組中已有數字的總和
    return sum1 - sum2

#輸入數組
nums = list(map(int, input("請輸入數組，以逗號分隔: ").split(',')))

# 計算缺失的數字
missing = number(nums)

#列印結果
print(f"nums = {nums}")
print(f"{missing}")
print(f"數組中包含了範圍 [0, {len(nums)}] 內的數字，其中缺少了 {missing}。")

```

3. 請回答下面問題：

Ans:

a.

$$f(n) = 2^{n+1} \Rightarrow O(2^n)$$

$$g(n) = 2^n$$

```

f(n) ≤ O(g(n))
2^{n+1} ≤ c * g(n)
2^{2n} ≤ c * 2^n
2^n * 2^1 ≤ c * 2^n
2^1 ≤ c
成立

```

b.

```

f(n) = 2^{2n} => O(2^n)
g(n) = 2^n
f(n) ≤ O(g(n))
2^{2n} ≤ c * g(n)
2^{2n} ≤ c * 2^n
2^{(n+n)} ≤ c * 2^n
2^n ≤ c
不成立

```

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.
def calculateTimes (number: int) -> None:
    while number >= 1: #(n + 1)
        counter:int = number #(n)
        while counter >= 1: #(n + 1)_n + (n)_n - 1 + ... + (2)_1 => (2 + (n +
1) * (n) / 2)
            print(number, counter) #((2 + (n + 1) * (n) / 2)) - 1
            counter = counter - 1 #((2 + (n + 1) * (n) / 2)) - 1
        number = number - 1 #n
    #(n^2 + 3n / 2) * 3 + 2n + n + 1 - 2
    #3/2(n^2) + 15n/2 - 1
    #O(T(n)) = n^2

```

(1)  $\frac{3}{2}n^2 + \frac{11}{2}n + 1$

(2)  $O(n^2)$

```

b.
def calculateTimes (number: int) -> None: #n = 16, 8, 4, 2, 1, 0 =>
floor(log_2^(16)) + 1
    while number >= 1: #floor(log_2^(16)) + 1
        print(number) #floor(log_2^(16))
        number = number // 2 #floor(log_2^(16))

```

```
#T(n) = 3 * floor(log_2(n)) + 1
#O(T(n)) = O(log_2(n)) = O(log n)
```

(1)  $3\lceil \log_2 n \rceil + 4$

(2)  $O(\log_2 n)$

```
c.
def calculateTimes (number: int, size: int) -> None:
    while number >= 1: #floor(log_2(n)) + 2
        while size >= 1: #(m + 1)(floor(log_2(n))+1)
            print(number, size) #m(floor(log_2(n))+1)
            size = size - 1 #m(floor(log_2(n))+1)
            number = number // 2 #floor(log_2(n))+1
```

(1)  $(3m + 3)\lceil \log_2 n \rceil + 3m + 4$

(2)  $O(m \log_2 n)$

```
d.
#m = n(最大值)
def calculateTimes (number: int, size: int) -> None:
    while number >= 1: #floor(log_2(n))+2
        while size >= 1: #(n+1)(floor(log_2(n))+1)
            print(number, size) #n(floor(log_2(n))+1)
            size = size - 1 #n(floor(log_2(n))+1)
            number = number // 2 #floor(log_2(n))+1

#m = n/2(最小值)
def calculateTimes (number: int, size: int) -> None:
    while number >= 1: #floor(log_2(n))+2
        while size >= 1: #(n/2+1)(floor(log_2(n))+1)
            print(number, size) #n/2(floor(log_2(n))+1)
            size = size - 1 #n/2(floor(log_2(n))+1)
            number = number // 2 #floor(log_2(n))+1
```

(1)  $(3n + 3)\lceil \log_2 n \rceil + 3n + 4 \geq T(n) \geq (\frac{3n}{2} + 3)\lceil \log_2 n \rceil + \frac{3n}{2} + 4$

(2)  $O(n \log_2 n)$

## 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

透過本次練習，可以掌握多維陣列的運算技巧及Big-O分析，  
計算執行次數與時間複雜度，透過題目練習，學習計算時間複雜度方法

