程式作業 2

111504504　楊愷晴

一、程式碼

Way1: dynamic programming (Python)

```python
def lcs_len(str1, str2, m, n):
    c = [[0]*(n+1)]*(m+1)                   #c[0:m][0:n]

    for i in range(1, m+1):                 #1~m
        for j in range(1, n+1):             #1~n
            if str1[i-1] == str2[j-1]:
                c[i][j] = c[i-1][j-1] + 1
            elif c[i-1][j] >= c[i][j-1]:
                c[i][j] = c[i-1][j]
            else:
                c[i][j] = c[i][j-1]
    return c[m][n]
```

利用**動態規劃**的精神，c[i][j]表示 str1[1:i]和 str2[1:j]的 LCS 長度，從 c[1][1]開始找，最右下角的 c[m][n]即為所求。利用兩層 for 迴圈計算完 table c[][]。**Time complexity = O(n^2)**。

Way2: brute force (Python)

```python
def lcs_brute(str1, str2, m, n):
    if(m == -1 or n == -1):
        return 0
    if(str1[m] == str2[n]):
        return 1 + lcs_brute(str1, str2, m-1, n-1)
    return max(lcs brute(str1, str2, m-1, n), lcs brute(str1, str2, m, n-1))
```

從兩個字串的最後一個字元開始，一樣的話就去找 str1[0:m-1]和 str2[0:n-1]的 LCS（去掉兩邊最後一個字元後繼續找）；不一樣的話就找 Max{ str1 去掉最後一個字元、str2 不變的 LCS，str1 不變、str2 去掉最後一個字元的 LCS}。若其中一個 string 跑到 index -1 就終止遞迴。

Time complexity = O (2^n)。

二、實驗測試兩種方法

測試環境

CPU: 13th Gen Intel® Core™ i7-1360P

Operating System: Windows 11

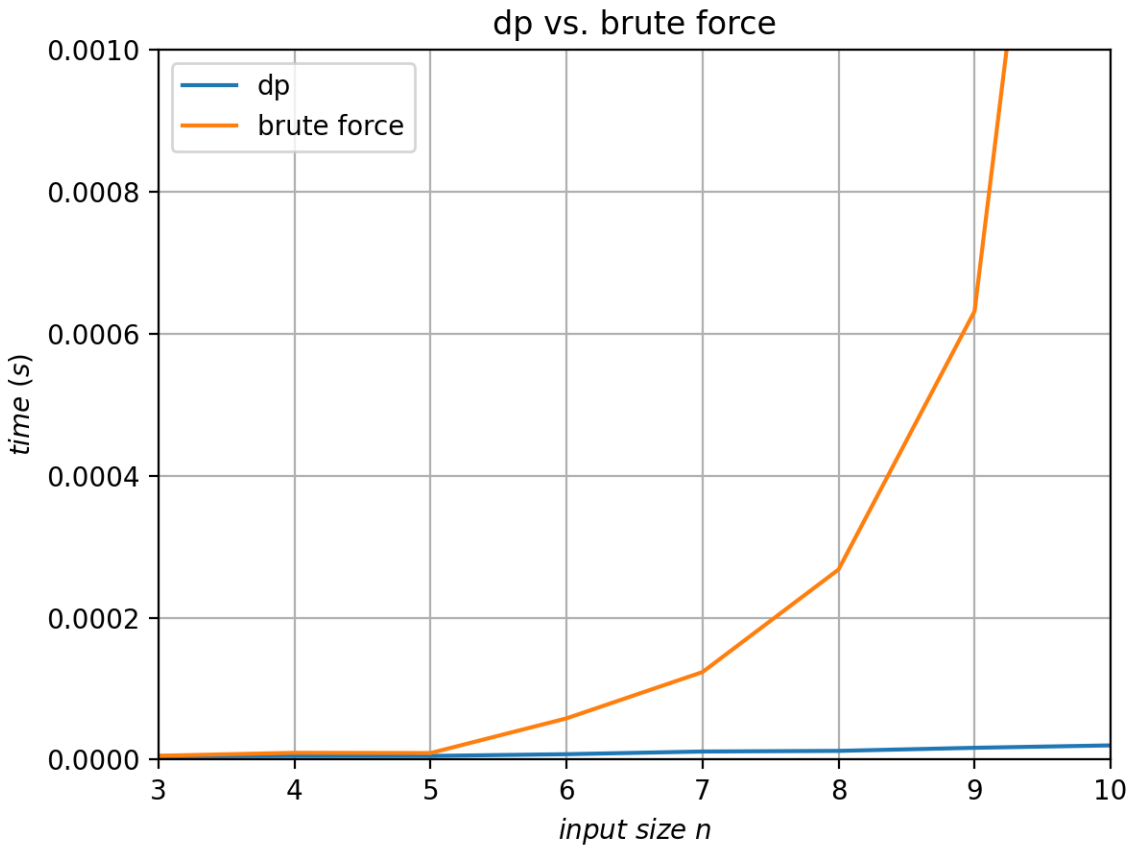
Python3.12

產生測資

手動產生 length=3~10 的英文字母字串，每種長度各 5 筆，存在’input.txt’，讀取該檔案，用 Python time module 取得 lcs_brute 和 lcs_len 兩個函式計算所需的時間，並寫到’output.txt’（在 google colab 執行）。

寫一個 plot.py，讀取’output.txt’，用 matplotlib 畫圖（在 google colab 執行）。

繪圖結果



根據圖表，大概在 n=5 後，dp 方法所需的時間明顯小於 brute force。