0819823陳子祈 演算法導論 HW6

1. 測試演算法正確性

給定 X=[6,6,6,0,8,9,4,5,7,9,4,5,3], Y=[1,6,8,0,4,8,7,5,4,8,7,7,4,1,4]

請大家利用老師上課所講的得出Longest Common Subsequence的演算法，找出Z=LCS(X,Y)以及印出Longest Common SubsequenceZ的長度。

一張含有 文字 的圖片

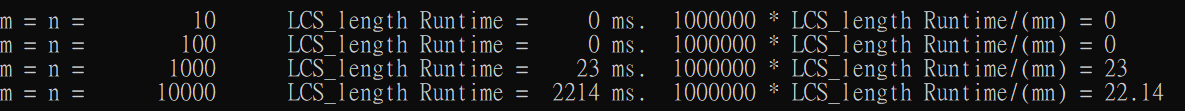
自動產生的描述

1. 時間複雜度試驗成果

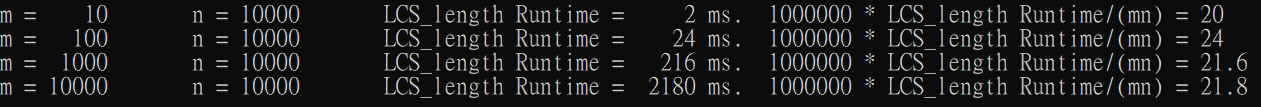
嘗試產生隨機元素為0~9不同長度的X與Y，並記錄下不同長度時所花費的時間，自行測試不同長度以驗證LCS-Length和Print-LCS時間複雜度是否和上課所講的一樣。

1. LCS\_length

根據上述基本要求，每個input array 所有元素都是從0~9隨機產生的，並使用for迴圈設定了m從10逐次成長10倍到10000，根據實測結果如下:



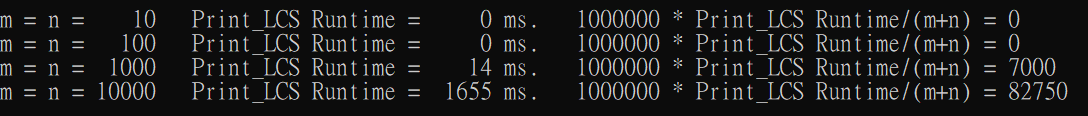
當m、n小於1000時，因為執行時間太短測不到，無法比較時間複雜度是否與上課講得一樣，因此我多測了幾項數據如下:



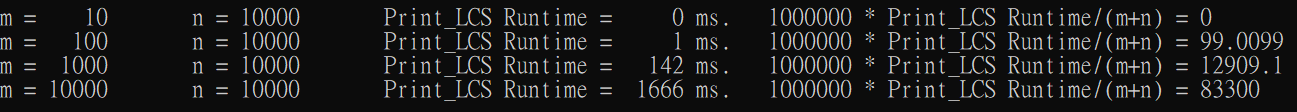
由上表可知，執行時間與m\*n有接近正比的關係，若n為input size，f(m,n)為此演算法執行的時間，符合Θ(f(m,n)) = m\*n的趨勢。

1. Print\_LCS

根據上述基本要求，每個input array 所有元素都是從隨機產生的，並使用for迴圈設定了m從10逐次成長10倍到10000，根據實測結果如下:



當m、n小於1000時，因為執行時間也會太短測不到，無法比較時間複雜度是否與上課講得一樣，因此我多測了幾項數據如下:

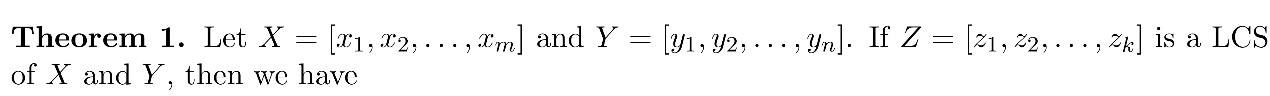


由上表可知，執行時間與m+n無正比的關係，若n為input size，f(m,n)為此演算法執行的時間，不符合Θ(f(m,n)) = m+n的趨勢。最大的原因是Print\_LCS以遞迴呼叫的方式實作，function call花的時間比iteration花的時間長很多，m、n線性成長不會讓執行時間也線性成長。

1. 討論

Q: 試著解釋原先窮舉法的時間複雜度為何，以及老師所講的演算法是如何改善時間複雜度。

A: 原先窮舉法的方法應該是找出X、Y所有的Subsequence，先將所有Subsequence比較是不是Common Subsequence，再找出最長的Common Subsequence。但是光X的所有的Subsequence就有2^m個，再兩兩比較是不是Common Subsequence就要花2m \* 2n= 2m+n次比較，執行時間會很長。因此老師所講的演算法使用Dynamic programming的技巧，提出以下構想:



一張含有 文字 的圖片

自動產生的描述

當xm = yn時，zk就是LCS其中的一個元素，因此可以讓i=1~m、j=1~n跑，當找到xi = yj，就把這個值存到z裡面，因此執行時間的複雜度就降為m\*n。只是講義上寫的是先找出LCS的長度再找出LCS每個元素的值，沒有直接把那個值存到z，不過方法大同小異，時間複雜度可以改善就好。

1. 心得

這一次作業與前幾次較時間複雜度的作業很像，不過這次要測試Longest Common Subsequence程式的執行時間。需要實作的函式只有LCS的兩個函式不多，其演算法也很簡單，因此這次作業主要就是要思考用什麼方法時做出來。我想到使用vector的方法實作，因為vector的創建與去除都比較靈活，可輕鬆創建隨機大小的vector拿去測試LCS。測試結果發現使用for loop的執行時間與預想一樣，但是使用遞迴呼叫的執行時間比預想長很多，我覺得如果把遞迴呼叫改成iteration的方法，執行時間應該會跟上課的一樣。