演算法導論 HW6

110511254 徐煜絨

一、程式架構

這次作業要實作 longest common subsequence (LCS)，假設有兩個序列分別為<2, 4, 6, 8, 10>和<1, 2, 5, 4, 9, 10>，則其 LCS 為<2, 4, 10>。

課本提供的演算法是用來找出兩個序列的 LCS，然而這次作業需要輸入三個序列（X, Y, Z），一開始想到兩種方法，第一種是建立三維的 c 和 b，並 print 出結果，第二種則是先將前兩個序列（X, Y）的 LCS 找出並設為 XY，再找出 XY 和 Z 的 LCS 並 print 結果。最後我選擇使用後者。

使用者輸入 seq.的大小和值。

↓

LCS_Length(): 建立 X 和 Y 的 c and b table.

↓

Produce(): 做出 X 和 Y 的 LCS，並存入新的 seq.(XY)。

↓

LCS_Length(): 建立 XY 和 Z 的 c and b table.

↓

Print_LCS(): 算出 XY 和 Z 的 LCS，即 X, Y, Z 的 LCS。

二、執行範例

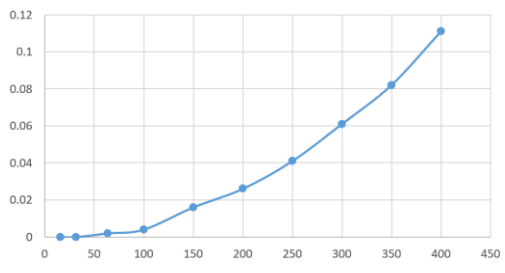
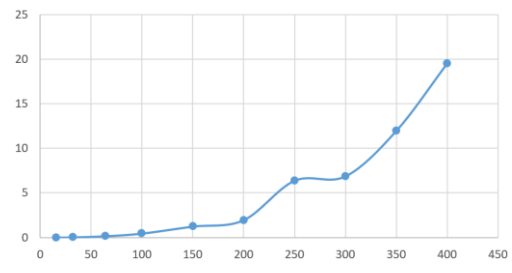
```
Input N1, N2, N3 (1<=length<1000)
N1: 3
N2: 5
N3: 2
x[1] = 1.90
x[2] = 2.3
x[3] = 5.0
y[1] = 1.9
y[2] = 2
y[3] = 5
y[4] = 6
y[5] = 7
z[1] = 1.900
z[2] = 5
LCS: 1.9 5
Length of LCS: 2
```

三、時間複雜度

前面有提及我選擇使用第二種方法，原因為下，當時考量前者可能會讓 LCS_Length 的時間複雜度升至$\theta(n1 \cdot n2 \cdot n3)$（其中 n1 為 X 的 size、n2 是 Y 的、n3 是 Z 的），所以最後選擇後者，希望把時間複雜度縮小到$\theta(n1 \cdot n2 + n4 \cdot n3)$（其中 n4 是 XY 的 size）。但原先 Print_LCS 的時間複雜度為$\theta(n1 + n2 + n3)$，可能會增加到$\theta((n1 + n2) + (n3 + n4))$。不過 n^3 降為 $2n^2$ 和 3n 升成 4n 相比，可以發現降低 LCS_Length 的時間複雜度會有較大效益。

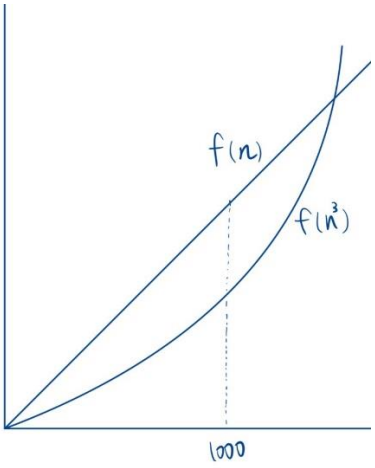
LL: LCS_Length, PL: Print_LCS（PL X and Y 用來做出 XY）

Size			Running time					
N1	N2	N3	LL X and Y	LL XY and Z	Total LL	PL X and Y	PL XY and Z	Total PL
16	16	16	0	0	0	0	0	0
32	32	32	0	0	0	0.014	0.009	0.023
64	64	64	0.002	0	0.002	0.092	0.057	0.149
100	100	100	0.002	0.002	0.004	0.136	0.311	0.447
150	150	150	0.011	0.005	0.016	0.561	0.667	1.228
200	200	200	0.02	0.006	0.026	1.396	0.544	1.94
250	250	250	0.017	0.024	0.041	4.338	2.016	6.354
300	300	300	0.036	0.025	0.061	6.003	0.839	6.842
350	350	350	0.045	0.037	0.082	10.329	1.599	11.928
400	400	400	0.066	0.045	0.111	17.578	1.89	19.468
1000	1000	1000	0.226	0.5	0.726	508.655	24.456	533.111

Total running time of two LCS_Length processes.	Total running time of Produce and Print_LCS processes.
	

四、討論

除了上面表格所列，有另外測試其他 seq. size 導致的不同執行時間，觀察下發現在一定 size 之內，整個程式在生成 XY 會花最多時間，猜測是因為一開始 n1 和 n2 都很大，所以從 c and b table 要找出 XY 就會花大量時間，而當 XY 的大小遠小於 n3，就會讓找出 XY 和 Z 的 LCS 的時間減少許多。由此可以推測當 n1, n2, n3 不大時，用三維陣列做出 c and b table 並 print 可能會有較好的 performance，因為儘管三維會讓 LCS_Length 的時間複雜度變成 $\theta(n^3)$，但在 size<10^4 前不會明顯上升，反而是$\theta(n)$的 Print_LCS 會上升許多。



老師上課提供的方法可以讓時間複雜度大幅下降，原先窮舉法要列出所有子序列並比較，需要$O(2^{n1}\cdot 2^{n2}\cdot 2^{n3})$，如果改成前兩者先比較，再取後兩者比較會降成$O(2^{n1}\cdot 2^{n2}+2^{n1+n2}\cdot 2^{n3})$，依舊比老師上課教的 DP 方法的$O(n1\cdot n2+n3\cdot n1)$（假設 worst case，X 和 Y 完全相同且 X.size()較小）大不少。在 Print_LCS 方面，時間複雜度難以從$O(n)$上獲得改善，不過空間複雜度能藉由 c and b table 的建立方法變小。