Exercise #3

REPORT

I. Environment

Mac OS, clang-1316.0.21.2, Xcode

II. Results

- Solutions

整個code可以簡單拆分成三個迴圈

– 輸入數值

因為總共有n組，每組有 m+1 個 profit 數值(可以取0～m個，所以個數為m+1)需要紀錄，所以開一個二維陣列 profit[n][m+1] 去紀錄在每個row取不同數量個 project 時的 profit值。

```
int profit[n][m+1];
for(int i=0; i<n; i++){
    for(int j=0; j<m+1; j++){
        cin >> profit[i][j];
    }
}
```

– 動態規劃

因為並不是每次都選擇最大利潤會組合出最終總和的最大利潤，因此我們需要把每次的結果分開來討論再進行比較並選擇最大利潤的組合。

有組數據(n)就拆分成幾個 stage，每後一個stage就多加入一個 project進行考量。舉例而言：Stage1 –> Project1, Stage2 –> Project1+Project2, Stage3 –> Project1+Project2+Project3，以此類推一直到 Stage(n) –> Project1+ … +Project(n)。然後再從Stage(n)中找最大值。

Concept example:

求賣出6盒水果能得到的最大利潤

No. of boxes		0	1	2	3	4	5	6
Profits	Store 1	0	4	6	7	7	7	7
	Store 2	0	2	4	6	8	9	10
	Store 3	0	6	8	8	8	8	8

Stage 1

No. X_1	0	1	2	3	4	5	6
Profit $F_1(X_1)$	0	4	6	7	7	7	7

Exercise #3

Stage 2

Store 1	No. X_1	0	1	2	3	4	5	6
	$F_1(X_1)$	0	4	6	7	7	7	7
Store 2		$F_1(X_1)+F_2(X_2)$						
No. X_2	$F_2(X_2)$							
0	0	0	4	6	7	7	7	7
1	2	2	6	8	9	9	11	
2	4	4	8	10	11	11		
3	6	6	10	12	13			
4	8	8	12	14				
5	9	9	13					
6	10	10						

Stage 3

No. of boxes		0	1	2	3	4	5	6
Max. $[F_1(X_1)+F_2(X_2)]$		0	4	6	8	10	12	14
Boxes in Store 1 + 2		0	1+0	2+0 1+1	2+1 1+2	2+2 1+3	2+3 1+4	2+4
Store 3		**Max. $[F_1(X_1)+F_2(X_2)]$**						
No. X_3	$F_3(X_3)$							
0	0	0	4	6	8	10	12	14
1	6	6	10	12	14	16	18	
2	8	8	12	14	16	18		
3	8	8	12	14	16			
4	8	8	12	14				
5	8	8	12					
6	8	8						

MAX

No. of boxes	0	1	2	3	4	5	6
Max. $[F_1(X_1)+F_2(X_2)+F_3(X_3)]$	0	6	10	12	14	16	18

Maximum profit: 18

– Reference

Exercise #3

- Time Complexity

```
for(int i=0; i<n; i++){

    //stage 1
    if(i==0){
        for(int j=0; j<m+1; j++)
            idx[j] = profit[0][j];
    }

    //other
    else{
        for(int j=0; j<m+1; j++){
            for(int k=0; k<m+1; k++){
                temp = profit[i][j] + idx[k];
                count = j+k;
                if(count>m)
                    break;
```

這次的程式碼中，每一個for迴圈都需要 n 的時間，而程式碼中最多有三層迴圈（動態規劃的部分），因此最後的時間複雜度為 n^3。