Algorithm 小考(一)

詳解與配分

Question 1 - (1)

- 詳解

```
n ← length[p] − 1
for i ← 1 to n do
    m[i, i] ← 0
end for
for ℓ ← 2 to n do
    for i ← 1 to n − ℓ + 1 do
        j ← i + ℓ − 1
        m[i, j] ← ∞
        for k ← i to j − 1 do
            q ← m[i, k] + m[k + 1, j] + p_{i−1} · p_k · p_j
            if q < m[i, j] then
                m[i, j] ← q
                s[i, j] ← k
            end if
        end for
    end for
end for
```

$$m[i,j] = \begin{cases} 0 & \text{if } i = j \\ \min_{i \le k < j}\{m[i,k] + m[k+1,j] & \text{if } i < j \\ +p_{i-1} \cdot p_k \cdot p_j\} & \end{cases}$$

We have three nested loops:

1. ℓ, length, $O(n)$ iterations
2. i, start, $O(n)$ iterations
3. k, split point, $O(n)$ iterations

Body of loops: constant complexity.

Total complexity: $O(n^3)$

- 配分(10%)
  - Algorithm(or Pseudocode) 8分
  - Time complexity 2分

Question 1 – (2)

- 詳解

M[i][j]

	1	2	3	4	5	6
1	0	15750	7875	9375	11875	15125
2		0	2625	4375	7125	10500
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0

S[i][j]

	2	3	4	5	6
1	1	1	3	3	3
2		2	3	3	3
3			3	3	3
4				4	5
5					5

  - So, ANS= $((A_1(A_2A_3))((A_4A_5)A_6))$ minimum number of scalar multiplications = 15125

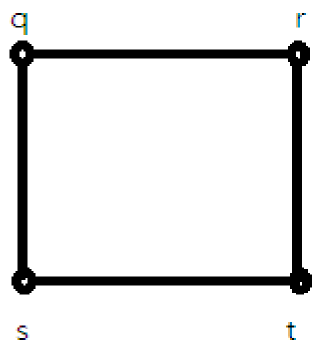
- 配分(10%)
  - Tables 5分 Ansers 5分

Question 2

- 詳解
  - Overlapping subproblem
    - A recursive algorithm revisits the same subproblem over and again.
  - Optimal substructure
    - An optimal solution to the problem contains optimal solutions to subproblems.
- 配分(10%)
  - 答案錯全錯 扣十分
  - 只寫出答案，未描述 扣四分

Question 3

- 詳解
  - Unweighted longest simple path problem does not satisfy the optimal substructure.



- 從圖可得知 q到t的最長path是q-r-t
  q-r並不是q到r的最長path (應該是q-s-t-r)
  而r-t也不是 r到t的最長path (應該是r-q-s-t)
  所以不具有optimal substructure

- 配分(20%)
  - 答案錯全錯
  - 例子敘述不完整，扣部分分數

Question 4

- 解答

(一)

step1. 將 A[1..n] 的資料排序存到 B[1..n]

step2. A 與 B 的 longest common subsequence (LCS)

step1. O(n lg n)

step2.O(n^2)

➔O (n^2)

Question 4

- 解答

(二)

Sequence : {A[1],A[2],…. A[n]}

L[i]: {A[1],A[2],…. A[i]}之Longest monotonically increasing subsequence的長度

P[i]:上一個字母位置

```
For(i=1;  i<=n; i++)
{
   L[i]= Max{ L[k] : 1<=k <=i-1  ,A[k]<=A[i] } + 1
   P[i] = k;
}
```

最後再找L[1…n]最大值 配合P列陣找回subsequence ➔O (nlogn)

- 配分(20%)

假如自己方法是對的且為O (n^2)，助教沒看出來可以找助教討論。

Question 5-(a)

- 詳解
  - 0-1 knapsack cannot be solved using the greedy strategy
- 配分(10%)
  - 答錯全錯 無部份給分

Question 5-(b)

- 詳解
  - Greedy: $\frac{8}{5} > \frac{5}{4} > \frac{4}{4}$
    - Choose item1 , Value = 8
  - If choose item2 & item3 , Value = 9

item	value	weight
1	8	5
2	4	4
3	5	4

Capacity = 8

- 配分(10%)
  - 範例不符合　斟酌給分
  - 未給範例　扣十分

Question 6

- 詳解

	yi	0	1	0	1	1	0	1	1	0
xi	0	0	0	0	0	0	0	0	0	0
1	0	0↑	1↖	1←	1↖	1↖	1←	1	1↖	1←
0	0	1↖	1↑	2↖	2←	2←	2↖	2←	2←	2↖
0	0	1↖	1↑	2↖	2↑	2↑	3↖	3←	3←	3↖
1	0	1↑	2↖	2↑	3↖	3↖	3↑	4↖	4↖	4←
0	0	1↖	2↑	3↖	3↑	3↑	4↖	4↑	4↑	5↖
1	0	1↑	2↖	3↑	4↖	4↖	4↑	5↖	5↖	5↑
0	0	1↖	2↑	3↖	4↑	4↑	5↖	5↑	5↑	6↖
1	0	1↑	2↖	3↑	4↖	5↖	5↑	6↖	6↖	6↑

- 注意:

在c[i - 1, j] =c[i, j - 1]case中 ，課本程式碼箭頭為:b[i, j] ← “↑”，
有些人箭頭標” ←”會算對，但希望還是依照課本為主。 但若沒統一，
一些標“↑”一些標”←”則會算錯。

Question 6

- 詳解
  - LCS長度 = 6

    <1,0,0,1,1,0> or <1,0,1,1,0,1> or <1,0,1,0,1,1>
  - 若兩序列擺相反則答案為 <0,1,0,1,0,1>
- 配分(10%)
  - 箭號沒畫、共同字沒有圈、只寫長度為6沒有寫答案扣部分分數
  - 有畫圖但整個圖畫錯，扣6~8分