HW4

第 3 組

104201025 張立欣

104303205 歐金榮

104303206 黃筱晴

104303542 林亦寧

105503512 趙德昊

105503516 游秉中

106503014 張秉洋

1.

長度 I	1	2	3	4
價格 pi	1	5	8	9
Density(pi/i)	1	2.5	2.67	2.25

考慮一鐵條，長度=4，

greedy strategy：切成長度 3+長度 1，總價值 8+1=9。

最佳解：切成長度 2+長度 2，總價值 5+5=10。

greedy strategy 未得到最佳解。

2.

題目要求在每切一刀都要再加一筆 cost，而其他部分都跟原本的 rod-cutting problem 相同，所以只需要做一些微調即可。

修改過的 pseudocode：

```
Modified rod-cutting (p, n, c)
    r[n] = {0}
    for j = 1 to n
        q = p[j]        //配合不切的情形所以先記住該長度的 price
        for i = 1 to j-1  // "j-1"即是為了避免不切的時候(i ==j)
                        //卻把 cost 算進去
            q = max (q, p[i] + r[j - i] - c)    // c = constant cost
        r[j] = q
    return r[n]
```

>如此一來同樣也是 double-nested loop，且 cost 不影響迴圈結束條件，所以複雜度維持在 $\theta(n^2)$。

3.

```
EXTENDED-MEMOIZED-CUT-ROD(p,n)
  let r[0...n] and s[0...n]be new arrays          // r records value and s records the first cut point for each size
  let sol be a dynamic size array
  for i = 0 to n
    r[i]= negative infinity
  value = EXTENDED-MEMOIZED-CUT-ROD-AUX(p,n,r,s)
  while n>0
    push s[n] in to sol                           // push length of left-hand rod to sol
    n = n-s[n]                                    // after pushing length of left-hand rod to sol n = length of right-hand rod
  return value and sol

EXTENDED-MEMOIZED-CUT-ROD-AUX(p,n,r,s)
  if r[n] >= 0                                    // if revenue of n is known return r[n]
    return r[n]
  r[n]=s[n]=0
  for i = 0 to n
    temp=p[i]+EXTENDED-MEMOIZED-CUT-ROD-AUX(p,n-i,r,s) // assign p[i]+r[n-i] to temp
    if temp > r[n]                                    // if temp is larger than r[n], then r[n] = temp
      r[n] = temp
      s[n] = i                                        // i will be the first cut point
  return r[n]
```

增加 s[n]這個陣列來紀錄在各個長度切第一刀的位置，然後利用 while 迴圈可以將最佳解一一切出來記錄在 sol 裡。而原本的

MEMORIZED-CUT-ROD-AUX 尋找 r[n]的方式與原本差不多，只是多了紀錄 s[n]。

4.

P_i ：(5,10,3,12,5,50,6)

遞迴式：$m[i,j]=\min_{i<=k<j}\{m[i,k]+m[k+1,j]+p_{i-1}p_kp_j\}$

$=0$,if $i=j$

表格如下：

	1	2	3	4	5	6
1	0	150	330	405	1655	2010
2		0	360	330	2430	1950
3			0	180	930	1770
4				0	3000	1860
5					0	1500
6						0

$m[1,2]= m[1,1] + m[2,2] + P_0P_1P_2 =150$，依此類推，由對角線往右上填表格，可

得最小值 2010

s[i,j]=a value of k that give min(紀錄 k 發生的位置)，表格如下

s[i,j]	1	2	3	4	5	6
1	0	1	2	2	4	2
2		0	2	2	2	2
3			0	3	4	4
4				0	4	4
5					0	5
6						0

s[1,6]=2 , s[3,6]=4 , s[5,6]=5 , 因此最佳解是 (A1A2)(A3A4)(A5A6)

6.

設 $bp[i]$ 為棍長 i 時的最佳解

bp(i)=$\begin{cases} 0\ ,\ n < i \leqq M \\ p(i),\ 0 < i \leqq n \end{cases}$

bp(i)=max{ $\mathrm{bp}(\mathrm{i}), \mathrm{bp}(\mathrm{i}-\mathrm{j})+\mathrm{p}(\mathrm{j})$ }

Pseudocode:

```
for i = 1 to M:
    if ( i > n ):   pi = 0
    else  :         dp[i] = pi
    for j = 1 to i-1:
        bp(i)=max{ bp(i), bp(i − j) + p(j) }
return bp[M]
```

7.

根據 Catalan Number，enumerating 的 time complex 為

$$C_n^{2n}/(n+1) = \Omega\left(4^n/n^{3/2}\right)$$

RECURSIVE-MATRIX-CHAIN 遞迴關係

$$\mathrm{T(n)} \le \begin{cases} c, & n = 1 \\ c + \sum_{k=1}^{n-1} (T(k) + T(n-k) + c), & n \ge 2 \end{cases}$$

$$\mathrm{T(n)} \le 2\sum_{i=1}^{n-1} T(i) + nc$$

我們只要找到 T(n) 的 upper bound 小於 $\Omega\left(4^n/n^{3/2}\right)$，即可證明

RECURSIVE-MATRIX-CHAIN 更有效率，在此我們假設 $\mathrm{T(i)} \le \mathrm{cn}3^{n-1}$，

for $n=1$

$$T(1) = c \le c \cdot 1 \cdot 3^{1-1} = c \text{ 成立}$$

for $n \ge 2$

$$\begin{aligned} T(n) &\le 2 \sum_{i=1}^{n-1} T(i) + cn \\ &\le 2 \sum_{i=1}^{n-1} c \cdot i \cdot 3^{i-1} + cn \\ &\le c \cdot \left(2 \sum i \cdot 3^{i-1} + n\right) \\ &= c \cdot \left(2 \cdot \left(\frac{n 3^{n-1}}{3-1} + \frac{1-3^n}{(3-1)^2}\right) + n\right) \\ &= cn3^{n-1} + \frac{c}{2}(2n+1-3^n) \\ &\le cn3^{n-1} \quad \text{成立} \end{aligned}$$

$$\boxed{* \sum_{i=1}^{n-1} i x^{i-1} = \frac{n x^{n-1}}{x-1} + \frac{1-x^n}{(x-1)^2}}$$

RECURSIVE-MATRIX-CHAIN 的 time complex 為 $\mathrm{O}(n3^{n-1})$，
故 RECURSIVE-MATRIX-CHAIN 比 enumerating 更有效率。

8.

Let A1=5x6, A2=6x4, A3=4x2

P = {5, 6, 4, 2}

$P_{i-1}P_kP_j$ is minimum if k=2

For Capulet’s method: (A1A2)A3= 160

Optimal answer: A1(A2A3)=108

So, Capulet’s method is a suboptimal solution