15.1-4

```
PRINT-MEMOIZED-CUT-ROD（p, n）
1 let r[0..n] and s[0..n] be a new array
2 for i = 0 to n
3 r[i]=−∞
4 (r, s) = EXTENDED-MEMOIZED-CUT-ROD-AUX(p, n, r, s)
5 while(n>0)
6     print s[n]
7     n-=s[n]

EXTENDED-MEMOIZED-CUT-ROD-AUX(p, n, r, s)
1 if r[n]≥0
2     return r[n]
3 if n == 0
4     q = 0
5 else q=−∞
6     for i = 1 to n
7         q=max(q, p[i]+MEMOIZED-CUT-ROD-AUX(p, n-i, r, s))
8         s[i] = i_max
9 r[n]=q
10return r and s
```

15.3-4

这是贪心法，太好举反例了。

(30,35), (35,15), (15,10), (10,20), (20,25)

贪心法是 1(2((34)5))，数字代表第几个矩阵，这里指 34 先乘，然后跟 5 乘，然后跟 2 乘，最后跟 1 乘，总共需要 49875 次乘法。动态规划最少只需要 28250 次，23 先乘，再跟 1 乘，然后 45 乘，最后 1-3 的结果和 45 的结果乘。

15-8

a. 第一行可以随便选一个点开始，即有 n 种选择，从第二行开始，可以有 2 种或 3 种选择，2 种的在边界，3 种的在除边界外的中间点。所以最终有$n * 2^{m-1}$*到*$n * 3^{m-1}$种选择，即以 m 为指数的函数。感觉中文版没翻译好，“m 的指数函数”，一般都会理解成以 m 为底数的函数吧？

b.

设计描述

假设 r[i, j]表示以像素点 A[i, j]结尾的破坏度最低的接缝。如果 A[i, j]非边界点，r[i, j]的值与 d[i-1, j-1]，d[i-1, j]，d[i-1,j+1]这三个值和 d[i, j]的和有关，取他们和的最小值。如果 A[i, j]是边界点，则也可以得出类似结论，因此，状态转移方程为

$$r[i,j]=\begin{cases} d[i,j], i=1, 1\le j\le n \\ \min(r[i-1,j], r[i-1,j+1])+d[i,j], 1<i\le m, j=1 \\ \min(r[i-1,j], r[i-1,j-1])+d[i,j], 1<i\le m, j=n \\ \min(r[i-1,j-1], r[i-1,j], r[i-1,j+1]), 1<i\le m, 1<j<n \end{cases}$$

其中 $1\le i\le m, 1\le j\le n$，*行数和列数均从* 1 *开始*

将 r[i, j]填充完后，对最后一行进行遍历，取最小值即可。

PS：其实可以对 d 矩阵进行扩展，第一列前面和最后一列后面再加一列，赋值全为正无穷，这样就不用区分一个点是否是边界点了，工程上的一个 trick，这里还是照旧做。

伪代码

```
MIN-SEAM(A, d)
1 let r[1..m][1..n] be a new array
2 for k = 1 to n
3     r[1][k] = d[1][k]
4 for i = 2 to m
5     for j = 1 to n
6         if j == 1
7             r[i][j] = min(r[i-1][j], r[i-1][j+1]) + d[i][j]
8         else if j == n
9             r[i][j] = min(r[i-1][j], r[i-1][j-1]) + d[i][j]
10        else
11            r[i][j] = min(r[i-1][j-1], r[i-1][j], r[i-1][j+1]) + d[i][j]
12 q=+∞
13 for k = 1 to n
14        q = min(r[m][k])
```

15 return q

通过伪代码容易看出时间复杂度为$O(mn)$