15.2-5 R(i,j)表示一次调用中计算其他表项时访问m[i,j]的次数

证明 $\sum_{i=1}^{n}\sum_{j=i}^{n} R(i,j) = \frac{n^3-n}{3}$　　$\sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$ (A.3)

$\sum_{i=1}^{n}\sum_{j=i}^{n} R(i,j) = \sum_{l=2}^{n}\sum_{i=1}^{n-l+1}\sum_{k=i}^{i+l-2} 2$　　即代码10行运行次数×2

$= \sum_{l=2}^{n}\sum_{i=1}^{n-l+1} 2(l-1)$

$= \sum_{l=1}^{n-1} 2l(n-l)$

$= n^3-n^2-\frac{2n^3-3n^2+n}{3} = \frac{n^3-n}{3}$

15.4-5 $O(n^2)$ find the longest monotonically increasing subsequence.

刻画最优解的结构特征

对于数组A=⟨$a_1, a_2 \cdots a_n$⟩，以a_1开头，a_i结尾的子数组的最长单调增子序列长度

= 以a_1开头，a_j结尾的子数组的最长单调增子序列长度+1.

其中j是满足$j<i$, $a_j<a_i$的最大值。

index:	1	2	3	4	5	6
num:	1	4	3	4	2	3
length:	1	2	2	3	2	3
track:	0	1	1	3	1	5

利用length数组记录长度

track数组记录轨迹

```
LIS-LENGTH(A)
    n = A.length
    let L[0...n] and T[0...n] be new arrays
    L[0] = 0  T[0] = 0
    for i = 2 to n
        j = i-1
        while A[j] >= A[i] and j != 0
            j = j-1
        L[i] = L[j]+1
        T[i] = j
    return L and T
```

构造LIS:

```
PRINT-LIS(L,T,A)
    max = 0
    for i = n to 1
        if L[i] > max
            max-index = i
            max = L[i]
    print(A[max-index])
    while T[max-index] != 0
        max-index = T[max-index]
        print(A[max-index])
```

15.5-3 计算w(i,j)代价为$\Theta(j-i)$，算法中已有3重循环，不影响，仍为$O(n^3)$