Hw 3

Page42：4.1-4，4.1-5

Page50：4.3-3，4.3-5，4.3-7

4.1

Page42：4.1-4，4.1-5

4.1-4 假定修改最大子数组问题的定义，允许结果为空子数组，其和为 0。你应该如何修改现有算法，使它们能允许空子数组为最终结果？

当算法返回负值时，用返回空子数组代替。

4.1-5 使用如下思想为最大子数组问题设计一个非递归的、线性时间的算法。从数组的左边界开始，由左至右处理，记录到目前为止已经处理过的最大子数组。若已知 $A[1..j]$的最大子数组，基于如下性质将解扩展为 $A[1..j+1]$的最大子数组：$A[1..j+1]$的最大子数组要么是 $A[1..j]$的最大子数组，要么是某个子数组 $A[i..j+1]$($1\leqslant i\leqslant j+1$)。在已知 $A[1..j]$的最大子数组的情况下，可以在线性时间内找出形如 $A[i..j+1]$的最大子数组。

分析如下代码写出算法思想和伪码：

```python
# =========== O(n)方法的两个版本 =============


def FindMaxSubArray(A, n):
    # 完全按照课上思路走的O(n)方法

    # max变量记录全局最大sum和下标
    # maxSum初始为A[0]而不是0，应对全为负数的数组
    maxSum = A[0]
    maxI, maxJ = 0, 0
    # current变量记录当前最大sum和下标
    curSum = 0
    curI, curJ = 0, 0
    for k in range(n):
        curSum += A[k]
        curJ = k
        if curSum > maxSum:
            maxSum = curSum
            maxI, maxJ = curI, curJ
        # curSum如果<=0，对后面的贡献非正，舍弃，从k+1重新开始
        if curSum <= 0:
            curI = curJ = k+1
            # 赋值为0，相当于舍弃之前的和
            curSum = 0
    print(f"max = {maxSum} in [{maxI},{maxJ}]")


def FindMaxSubArray2(A, n):
    # O(n)方法的另一个版本，与上个版本思想是一致的
    # 这个版本更直接反映题意
```

```python
    # max变量记录全局最大sum和下标
    # # maxSum初始为A[0]而不是0，应对全为负数的数组
    maxSum = A[0]
    maxI, maxJ = 0, 0
    # maxSumEndAtRightBound变量记录迭代过程中，以右边界结尾的最大子数组和
    maxSumEndAtRightBound = 0
    mseI, mseJ = 0, 0
    for k in range(n):
        # 由上次迭代的maxSumEndAtRightBound，确定这次迭代的maxSumEndAtRightBound
        if maxSumEndAtRightBound + A[k] > A[k]:
            maxSumEndAtRightBound += A[k]
            mseJ = k
        else:
            maxSumEndAtRightBound = A[k]
            mseI = mseJ = k
        # 由上次迭代的maxSum，即A[..k-1]的最大子数组和
        # 以及这次迭代的maxSumEndAtRightBound
        # 确定这次迭代的maxSum
        if maxSumEndAtRightBound > maxSum:
            maxSum = maxSumEndAtRightBound
            maxI, maxJ = mseI, mseJ
    print(f"max = {maxSum} in [{maxI},{maxJ}]")
```

非递归显然，就扫一遍，$T(n)=O(n)$，线性时间。

为什么符合题目所给的思想：第k次迭代开始，maxSum存的还是上次迭代的最大子数组和，计算以k结尾的最大子数组和maxSumEndAtRightBound，比较得第k次迭代的最大子数组和更新maxSum。

伪码略

4.3

Page50：4.3-3，4.3-5，4.3-7

4.3-3 我们看到 $T(n)=2T(\lfloor n/2\rfloor)+n$ 的解为 $O(n\lg n)$。证明 $\Omega(n\lg n)$ 也是这个递归式的解。从而得出结论：解为 $\Theta(n\lg n)$。

First, we guess $T(n) \le cn \lg n$,

$$\begin{aligned}
T(n) &\le 2c\lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor + n \\
&\le cn \lg(n/2) + n \\
&= cn \lg n - cn \lg 2 + n \\
&= cn \lg n + (1 - c)n \\
&\le cn \lg n,
\end{aligned}$$

where the last step holds for $c \ge 1$.

Next, we guess $T(n) \ge c(n+2)\lg(n+2)$,

$$\begin{aligned}
T(n) &\ge 2c(\lfloor n/2 \rfloor + 2)(\lg(\lfloor n/2 \rfloor + 2) + n \\
&\ge 2c(n/2 - 1 + 2)(\lg(n/2 - 1 + 2) + n \\
&= 2c\frac{n+2}{2} \lg \frac{n+2}{2} + n \\
&= c(n+2)\lg(n+2) - c(n+2)\lg 2 + n \\
&= c(n+2)\lg(n+2) + (1-c)n - 2c \\
&\ge c(n+2)\lg(n+2),
\end{aligned}$$

where the last step holds for $n \ge \frac{2c}{1-c}, 0 \le c < 1$

4.3-5 证明：归并排序的“严格”递归式(4.3)的解为 $\Theta(n\lg n)$。

To show Θ bound, separately show O and Ω bounds.

- For $O(n \lg n)$, we guess $T(n) \le c(n-2)\lg(n-2)$,

$$
\begin{aligned}
T(n) &\le c(\lceil n/2 \rceil - 2)\lg(\lceil n/2 \rceil - 2) + c(\lfloor n/2 \rfloor - 2)\lg(\lfloor n/2 \rfloor - 2) + dn \\
&\le c(n/2 + 1 - 2)\lg(n/2 + 1 - 2) + c(n/2 - 2)\lg(n/2 - 2) + dn \\
&\le c(n/2 - 1)\lg(n/2 - 1) + c(n/2 - 1)\lg(n/2 - 1) + dn \\
&= c\frac{n-2}{2}\lg\frac{n-2}{2} + c\frac{n-2}{2}\lg\frac{n-2}{2} + dn \\
&= c(n-2)\lg\frac{n-2}{2} + dn \\
&= c(n-2)\lg(n-2) - c(n-2) + dn \\
&= c(n-2)\lg(n-2) + (d-c)n + 2c \\
&\le c(n-2)\lg(n-2),
\end{aligned}
$$

where the last step holds for $c > d$.

- For $\Omega(n \lg n)$, we guess $T(n) \ge c(n+2)\lg(n+2)$,

$$
\begin{aligned}
T(n) &\ge c(\lceil n/2 \rceil + 2)\lg(\lceil n/2 \rceil + 2) + c(\lfloor n/2 \rfloor + 2)\lg(\lfloor n/2 \rfloor + 2) + dn \\
&\ge c(n/2 + 2)\lg(n/2 + 2) + c(n/2 - 1 + 2)\lg(n/2 - 1 + 2) + dn \\
&\ge c(n/2 + 1)\lg(n/2 + 1) + c(n/2 + 1)\lg(n/2 + 1) + dn \\
&\ge c\frac{n+2}{2}\lg\frac{n+2}{2} + c\frac{n+2}{2}\lg\frac{n+2}{2} + dn \\
&= c(n+2)\lg\frac{n+2}{2} + dn \\
&= c(n+2)\lg(n+2) - c(n+2) + dn \\
&= c(n+2)\lg(n+2) + (d-c)n - 2c \\
&\ge c(n+2)\lg(n+2),
\end{aligned}
$$

where the last step holds for $d > c$.

4.3-7 使用 4.5 节中的主方法，可以证明 $T(n)=4T(n/3)+n$ 的解为 $T(n)=\Theta(n^{\log_3 4})$。说明基于假设 $T(n)\leqslant cn^{\log_3 4}$ 的代入法不能证明这一结论。然后说明如何通过减去一个低阶项完成代入法证明。

We guess $T(n) \le cn^{\log_3 4}$ first,

$$\begin{aligned} T(n) &\le 4c(n/3)^{\log_3 4} + n \\ &= cn^{\log_3 4} + n. \end{aligned}$$

We stuck here.

We guess $T(n) \le cn^{\log_3 4} - dn$ again,

$$\begin{aligned} T(n) &\le 4(c(n/3)^{\log_3 4} - dn/3) + n \\ &= 4(cn^{\log_3 4}/4 - dn/3) + n \\ &= cn^{\log_3 4} - \frac{4}{3}dn + n \\ &\le cn^{\log_3 4} - dn, \end{aligned}$$

where the last step holds for $d \ge 3$.