可获得的最大点数

几张卡牌 **排成一行**，每张卡牌都有一个对应的点数。点数由整数数组 `cardPoints` 给出。

每次行动，你可以从行的开头或者末尾拿一张卡牌，最终你必须正好拿 `k` 张卡牌。

你的点数就是你拿到手中的所有卡牌的点数之和。

给你一个整数数组 `cardPoints` 和整数 `k`，请你返回可以获得的最大点数。

示例 1：

输入： `cardPoints = [1,2,3,4,5,6,1], k = 3`

输出： `12`

解释： 第一次行动，不管拿哪张牌，你的点数总是 `1` 。但是，先拿最右边的卡牌将会最大化你的可获得点数。最优策略是拿右边的三张牌，最终点数为 `1 + 6 + 5 = 12` 。

示例 2：

输入： `cardPoints = [2,2,2], k = 2`

输出： `4`

解释： 无论你拿起哪两张卡牌，可获得的点数总是 `4` 。

示例 3：

输入： `cardPoints = [9,7,7,9,7,7,9], k = 7`

输出： `55`

解释： 你必须拿起所有卡牌，可以获得的点数为所有卡牌的点数之和。

示例 4：

输入： `cardPoints = [1,1000,1], k = 1`

输出： `1`

解释： 你无法拿到中间那张卡牌，所以可以获得的最大点数为 `1` 。

示例 5：

输入： `cardPoints = [1,79,80,1,1,1,200,1], k = 3`

输出： `202`

```python
class Solution:
    def maxScore(self, cardPoints: List[int], k: int) -> int:
        window=len(cardPoints)-k
        #n-k 个剩余必定连续 以滑动窗口找到剩余连续最小和的子序列
        sum_num=sum(cardPoints[:window])
        min_sum=sum_num #以第一个滑窗作为初始化 min_sum
        for i in range(window,len(cardPoints)):
            #滑动过程：右加左减
            sum_num+=cardPoints[i]-cardPoints[i-window]
            min_sum=min(min_sum,sum_num)
        return sum(cardPoints)-min_sum #总和减去最小部分 剩余为取出的最大值
```