K 个不同整数的子数组

给定一个正整数数组 A，如果 A 的某个子数组中不同整数的个数恰好为 K，则称 A 的这个连续、不一定不同的子数组为*好子数组*。

（例如，[1,2,3,1,2] 中有 3 个不同的整数：1，2，以及 3。）

返回 A 中*好子数组*的数目。

示例 1：

输入： A = [1,2,1,2,3], K = 2

输出： 7

解释： 恰好由 2 个不同整数组成的子数组：[1,2], [2,1], [1,2], [2,3], [1,2,1], [2,1,2], [1,2,1,2].

示例 2：

输入： A = [1,2,1,3,4], K = 3

输出： 3

解释： 恰好由 3 个不同整数组成的子数组：[1,2,1,3], [2,1,3], [1,3,4].

```
class Solution:
    def subarraysWithKDistinct(self, nums: List[int], k: int) -> int:
        return self.mostK(nums,k)-self.mostK(nums,k-1)

    def mostK(self,A,k):
        #左右指针
        left=0
        right=0
        #字典 统计某数字出现次数
        counter=collections.defaultdict(int)
        #统计不同数字个数
        keynum=0
        #统计最终满足条件子数组个数
        res=0
        while right<len(A):
            if counter[A[right]]==0:
                keynum+=1#右指针指向为新数字，第一次出现需要将其统计为不同数字
            counter[A[right]]+=1
            while keynum>k:#调整区间
```

```
                counter[A[left]]-=1
                if counter[A[left]]==0:
                    keynum-=1
                left+=1#左指针右移 调整区间
            #上面求的是 A 中由 K 个不同整数组成的最长子数组的长度，如果问 A 中由最多 K 个不同整数组成的子数组的个数，该怎么办呢？答：只用把 res = max(res, right - left + 1) 改成 res += right - left + 1。我们要求由最多 K 个不同整数组成的子数组的个数,那么对于长度 [left, right] 区间内的每个子数组都是满足要求的，res 要累加的其实是 符合条件的并且以 right 为右端点的子数组个数，这个数量就是 right - left + 1，即区间长度。例如 [2,4,3,5] 满足条件，要累加的其实就是 [2,4,3,5], [4,3,5], [3,5], [5] 四个子区间
            res+=right-left+1
            right+=1
        return res
```