768. Max Chunks To Make Sorted II

You are given an integer array `arr`.

We split `arr` into some number of **chunks** (i.e., partitions), and individually sort each chunk. After concatenating them, the result should equal the sorted array.

Return *the largest number of chunks we can make to sort the array*.

Example 1:

```
Input: arr = [5,4,3,2,1]
Output: 1
Explanation:
Splitting into two or more chunks will not return the required result.
For example, splitting into [5, 4], [3, 2, 1] will result in [4, 5, 1, 2,
3], which isn't sorted.
```

Example 2:

```
Input: arr = [2,1,3,4,4]
Output: 4
Explanation:
We can split into two chunks, such as [2, 1], [3, 4, 4].
However, splitting into [2, 1], [3], [4], [4] is the highest number of
chunks possible.
```

Constraints:

- `1 <= arr.length <= 2000`
- `0 <= arr[i] <= 10<sup>8</sup>`
-
  ```
  import sys
  class Solution:
      def maxChunksToSorted(self, arr: List[int]) -> int:
          leftMax = [0]*len(arr)
          rightMin = [sys.maxsize]*(len(arr)+1)
          tempMax = 0
          tempMin = sys.maxsize
          for i in range(len(arr)):
              tempMax = max(tempMax,arr[i])
              leftMax[i] = tempMax
  ```

```
        for i in range(len(arr)-1,-1,-1):
            tempMin = min(tempMin,arr[i])
            rightMin[i] = tempMin

        count = 0
        for i in range(0,len(arr)):
            if leftMax[i]<=rightMin[i+1]:
                count = count+1
        return count
```