**Problem #2855: Minimum Right Shifts to Sort the Array (Easy)**

<https://leetcode.com/problems/minimum-right-shifts-to-sort-the-array/description/>

**My Solution:**

1. Initialize largestDigit as an empty dictionary -- the keys will be the maximum digit of the number and the numbers that have this key will be values that are in a list.
2. Iterate through each number num in nums. Initialize tempList as an empty list to hold the digits of num. Let x be equal to num. While x is greater than 0, append to templist the remainder obtained when x is divided by 10 and then do integer division of x by 10. Let maxNum be the maximum number in tempList. if maxNum is not in largestDigit keys, then add maxNum as the key and the number num as an entry in a list. Otherwise if maxNum is already in largestDigit keys, append num to the value list.
3. Let sortedKeys be the keys of largestDigit in descending order.
4. Initialize res to -1.
5. For key in sortedKeys, if the length of the value corresponding to key has a length greater than 2, sorted the values called sortedVals in descending order. Update res with the maximum of res and the sum of sortedVals at index 0 and 1.
6. Return res.

**Complexity**

* Time complexity: O(n \* log(n)) since lists are sorted
* Space complexity: O(n) since dictionary length depends on n

**Code**

class Solution:

def maxSum(self, nums: List[int]) -> int:

largestDigit = {}

for num in nums:

tempList = []

x = num

while x > 0:

tempList.append(x % 10)

x //= 10

maxNum = max(tempList)

if maxNum not in largestDigit.keys():

largestDigit[maxNum] = [num]

else:

largestDigit[maxNum].append(num)

sortedKeys = sorted(largestDigit.keys(), reverse = True)

res = -1

for key in sortedKeys:

if len(largestDigit[key]) >= 2:

sortedVals = sorted(largestDigit[key], reverse = True)

res = max(res, sortedVals[0] + sortedVals[1])

return res