Problem # 1337: The K Weakest Rows in a Matrix (Easy)

-----------------------------------------------------------------------

<https://leetcode.com/problems/the-k-weakest-rows-in-a-matrix/>

My Solution:

Runtime beats 97.05%.

1. Let m be the length if the matrix (i.e. number of rows in the matrix).

2. Initialize a dictionary called adict.

3. Iterate through the rows of the matrix and store the sum as key in adict and value as a list consisting of the row number.

4. Sort the keys of adict in a key called sorted\_keys

5. Initialize an empty list called result to store the result.

6. Initialize count to 0

7. Iterate through the keys in sorted\_keys and each key iterate through the values in adict for that key. Append the value to the result list. Increment count by 1.

8. If count is equal to k, then return the result since we have the k weakest rows.

--------------------------

class Solution:

def kWeakestRows(self, mat: List[List[int]], k: int) -> List[int]:

m = len(mat)

adict = {}

for i in range(m):

row\_sum = sum(mat[i])

if row\_sum not in adict:

adict[row\_sum] = [i]

else:

adict[row\_sum].append(i)

sorted\_keys = sorted(adict.keys())

result = []

count = 0

for key in sorted\_keys:

for val in adict[key]:

result.append(val)

count += 1

if count == k:

return(result)