Problem # 11 : Container with most water (Medium)

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

<https://leetcode.com/problems/container-with-most-water/>

My solution:

Runtime beats 75.03%.

1. Let n be the length of height array.

2. Initialize left pointer to index 0 and right pointer to index n-1 ( i.e. index of the last element in height array).

3. Initialize max\_area to 0.

4. While left pointer is less than right pointer (they cannot cross over), area will be the product of the difference of right and left, and the minimum of height at index left and index right. So max\_area is the max of max\_area and the area just calculated.

5. If the height at the left index is greater than the height at the right index, then move the right index to the left by 1 unit, i.e. decrement the right pointer by 1. Otherwise, move the left pointer to the right by 1 unit, i.e. increment left pointer by 1.

6. Finally return max\_area.

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

class Solution:

def maxArea(self, height: List[int]) -> int:

n = len(height)

left, right = 0, n-1

max\_area = 0

while (left < right):

max\_area = max(max\_area, (right - left) \* min(height[left], height[right]))

if height[left] > height[right]:

right -= 1

else:

left += 1

return max\_area