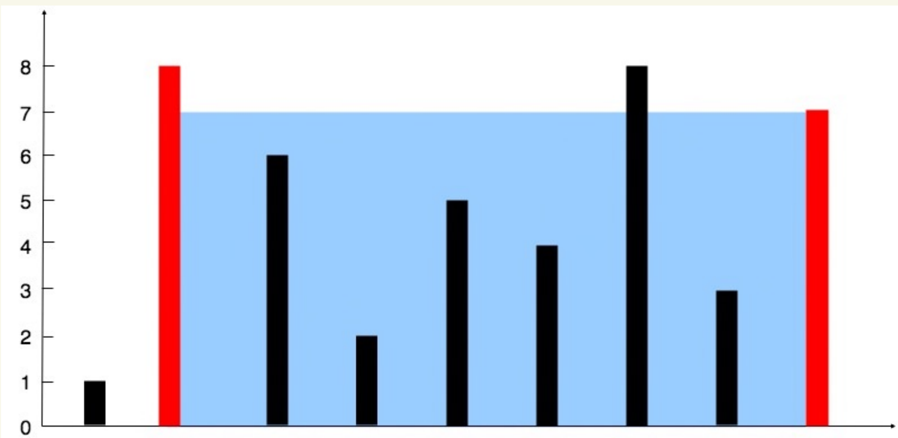
Problem Definition :

給定一大小為 n 的陣列為 height, 其中: height[i] 表示 i 時的高度, 欲找一 container 為 [i, j], 其中, 水量為最大的 container.

Example: height = [1, 8, 6, 2, 5, 4, 8, 3, 7]



Solution:

①. **brute force:**

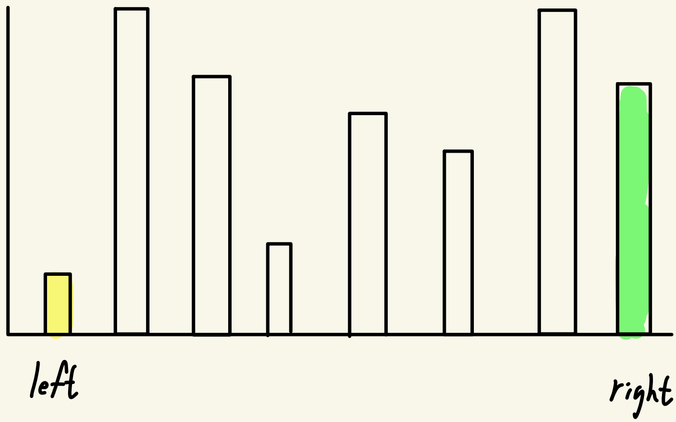
遍歷 $1 \le i < j \le n$, 計算: $(j-i) \times (\min(height[i], height[j]))$

儲最大的那個.

Time Complexity: $O(n^2)$

② Two Pointers 方法：

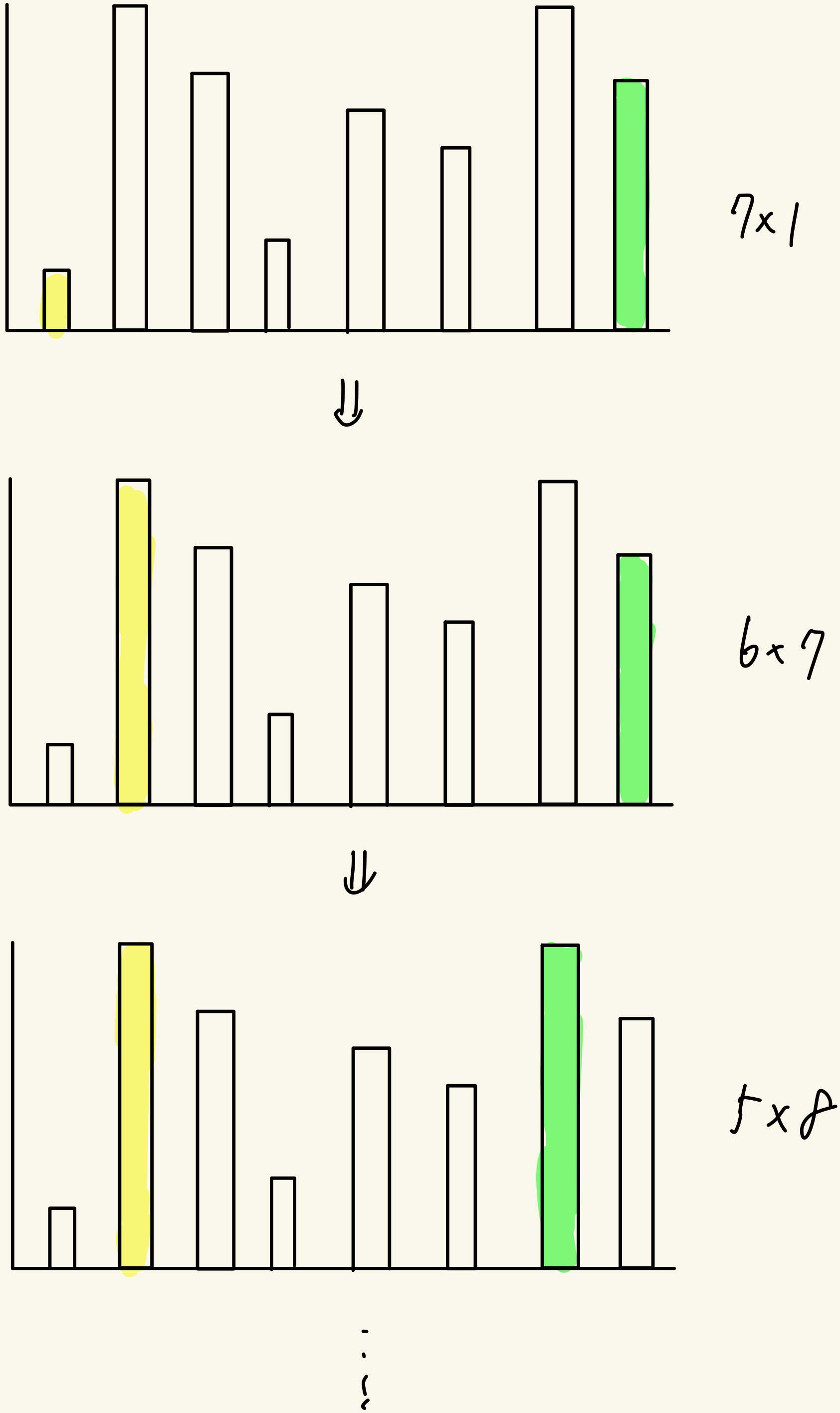
用左右指標記錄當前 i, j



問題在每一回合，如何決定 left 向右或 right 向左可涵蓋到 optimal solution.

＊若 height[left] ≤ height[right] 則移动 left

height[left] > height[right] 則移动 right



7x1
6x7
5x8

為何此方法會 work?

已知當前 amount 為:

$$(right - left) \times \min(height[left], height[right]) \text{ — } ①$$

下一个選擇可為:

(1). $(right - left - 1) \times \min(height[left+1], height[right])$ — ②

(2). $(right - left - 1) \times \min(height[left], height[right-1])$ — ③

Optimal solution 為 max(①, ②, ③)

但若已知 $height[left] \leq height[right]$

① = $(right - left) \times height[left]$

③ = $(right - left - 1) \times height[left]$

則可保證 ③ < ①

故可不考慮 ③, ∵ ③ 不會使解變好.

同理，當 height[left] > height[right]

可不考慮 ②

又當 left ≥ right 時，解答為 0，必不為解

故可用 two pointers 來減少遍歷次數！