11. Container With Most Water

Hint

Medium 26.3K 1.4K

Companies

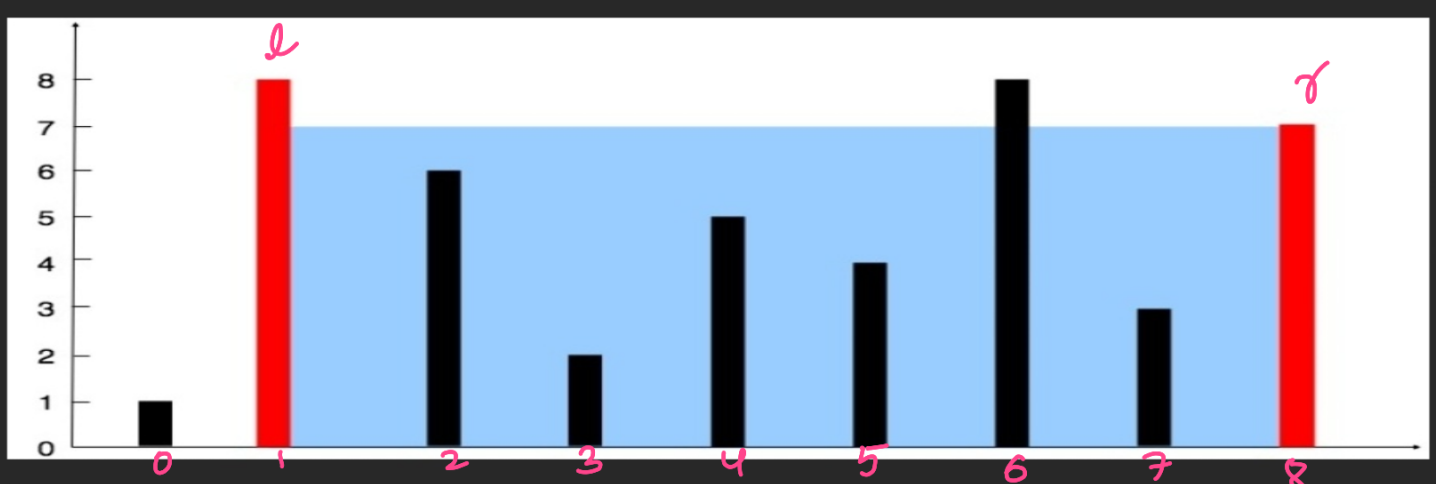
You are given an integer array `height` of length `n`. There are `n` vertical lines drawn such that the two endpoints of the `i^th` line are `(i, 0)` and `(i, height[i])`.

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return *the maximum amount of water a container can store*.

Notice that you may not slant the container.

Example 1:




```
Input: height = [1,8,6,2,5,4,8,3,7]
Output: 49
Explanation: The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.
```

Example 2:

```
Input: height = [1,1]
Output: 1
```

Constraints:

- `n == height.length`
- `2 <= n <= 10^5`
- `0 <= height[i] <= 10^4`

Accepted **2.3M** | Submissions **4.3M** | Acceptance Rate **54.1%**

Bruteforce:

The main thing that we need to figure out is

Q: how to calculate amount of water that can be there b/w two lines indexed i & j?

The amount of water that can placed b/w two lines indexed i and j is

$$w = (j - i) * \min(h[i], h[j])$$

Thats it, now we can easily come up with a brute force solution.

```
maxwater = Currwater = 0

for( i : 0 to n-1)
{  for(j : i+1 to n-1)
   {
      w = (j - i) * min(h[i], h[j])

      maxwater = max(currwater, maxwater)
   }
}
return maxwater
```

$T(n) : O(n^2)$

Approach 1: Using Two pointer approach

Q: while using two pointer approach, how can we be sure on which pointer should be moved?

Lets say

$l = 1$ $\quad r = 6$

$h[l] = 2$ $\quad h[r] = 5$

now

$w = 5 * 2$

$= 10$

now lets say we move r pointer, no matter how big is $h[r]$ the value of w is always going to be less than 10

beoz $\min(2, h[r])$ will always be ≤ 2 and r value also decreasing. So the value of w will be < 10.

"So we must always move from the smaller line side." so that value of w will always be $\geq$ previous w.

```
maxwater = currwater = 0
l = 0 , r = n-1

while ( l < r)
{
    currwater = (j-i) * (min(h[l], h[r]))
    maxwater = max(currwater, maxwater)

    h[l] < h[r] ? l++ : r--
}

return maxwater.
```

$T(n) : O(n)$