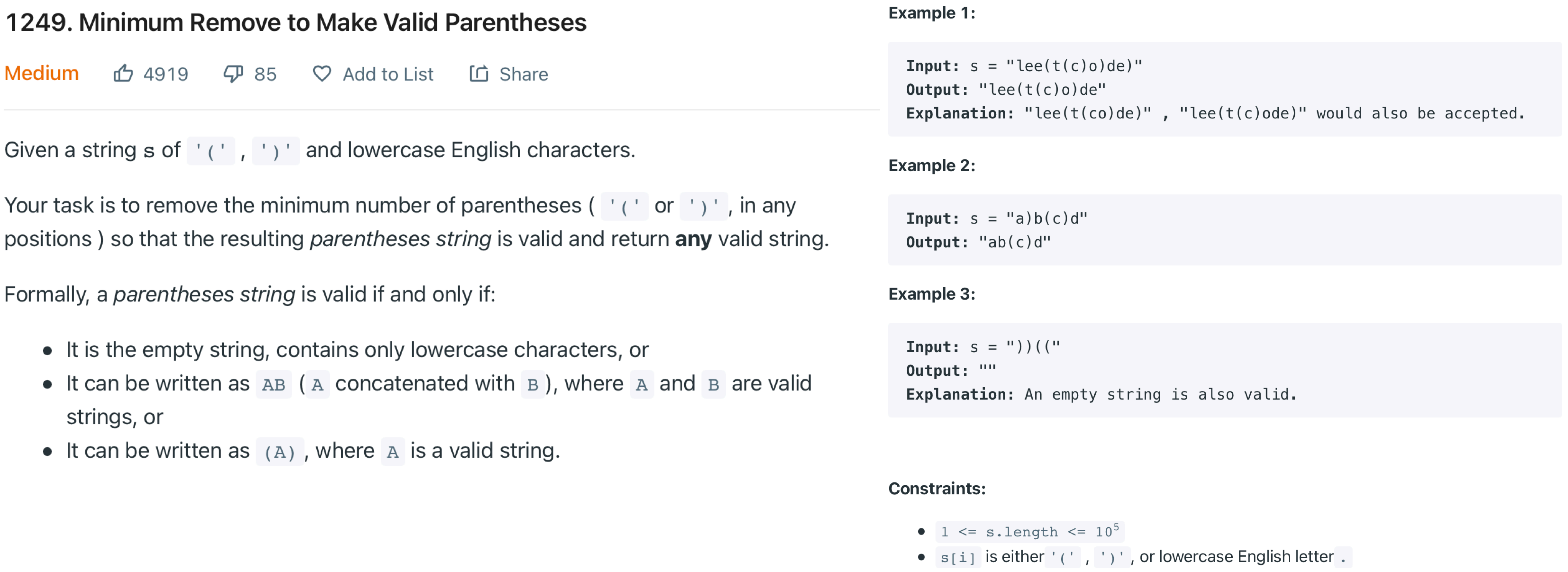


1249. Minimum Remove to Make Valid Parentheses

Given a string `s` of `'('` , `')'` and lowercase English characters.

Your task is to remove the minimum number of parentheses (`'('` or `')'`, in any positions) so that the resulting *parentheses string* is valid and return **any** valid string.

Formally, a *parentheses string* is valid if and only if:

- It is the empty string, contains only lowercase characters, or
- It can be written as `AB` (`A` concatenated with `B`), where `A` and `B` are valid strings, or
- It can be written as `(A)`, where `A` is a valid string.

Example 1:

```
Input: s = "lee(t(c)o)de)"
Output: "lee(t(c)o)de"
Explanation: "lee(t(co)de)" , "lee(t(c)ode)" would also be accepted.
```

Example 2:

```
Input: s = "a)b(c)d"
Output: "ab(c)d"
```

Example 3:

```
Input: s = "))(("
Output: ""
Explanation: An empty string is also valid.
```

Constraints:

- `1 <= s.length <= 10`^{5}
- `s[i]` is either `'('` , `')'`, or lowercase English letter `.`

0	1	2	3	4	5	6
a	)	b	(	c	)	(

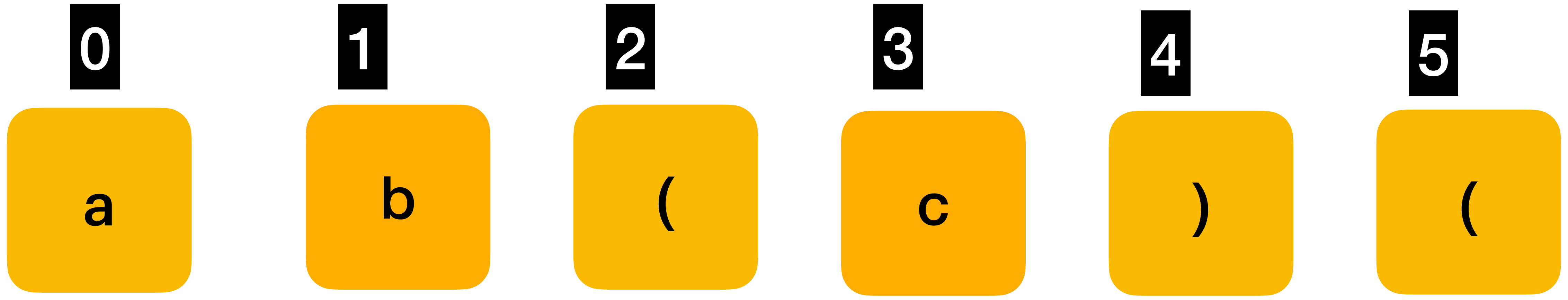
Input : a)b(c)(

Output : ab(c)

openBraceSeenCount = 0
openBraceCount= 0

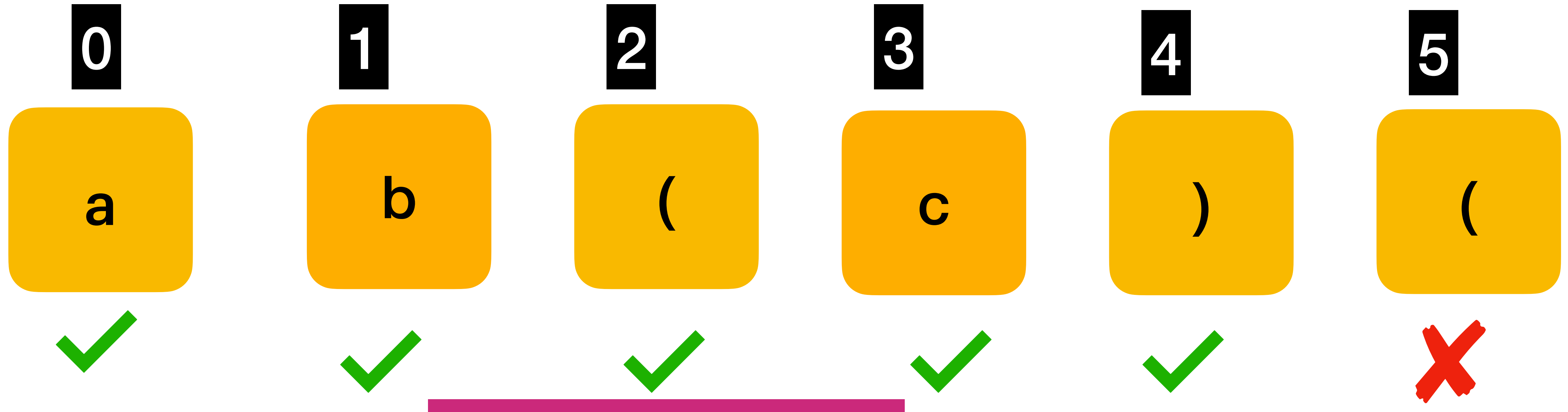
Let's Eliminate Invalid ')' Parentheses

0	1	2	3	4	5	6
a	)	b	(	c	)	(
✓	✗	✓	✓	✓	✓	✓
	As openBraceCount = 0		openBraceCount = 1 openBraceSeenCount = 1		openBraceCount - - openBraceCount = 0 openBraceSeenCount = 1	openBraceCount = 1 openBraceSeenCount = 2



Let’s Eliminate Invalid ‘(’ Parentheses

Int validOpenParentheses = openBraceSeenCount - openBraceCount = 2 - 1 = 1

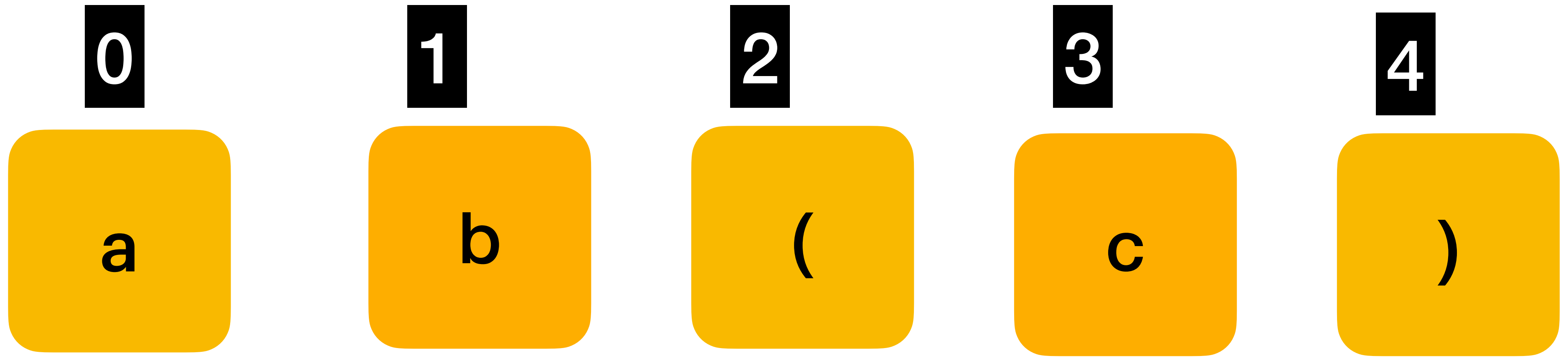


validOpenParentheses = 1

validOpenParentheses - -;
validOpenParentheses = 0

Time Complexity : O(n)

Space Complexity : O(1)



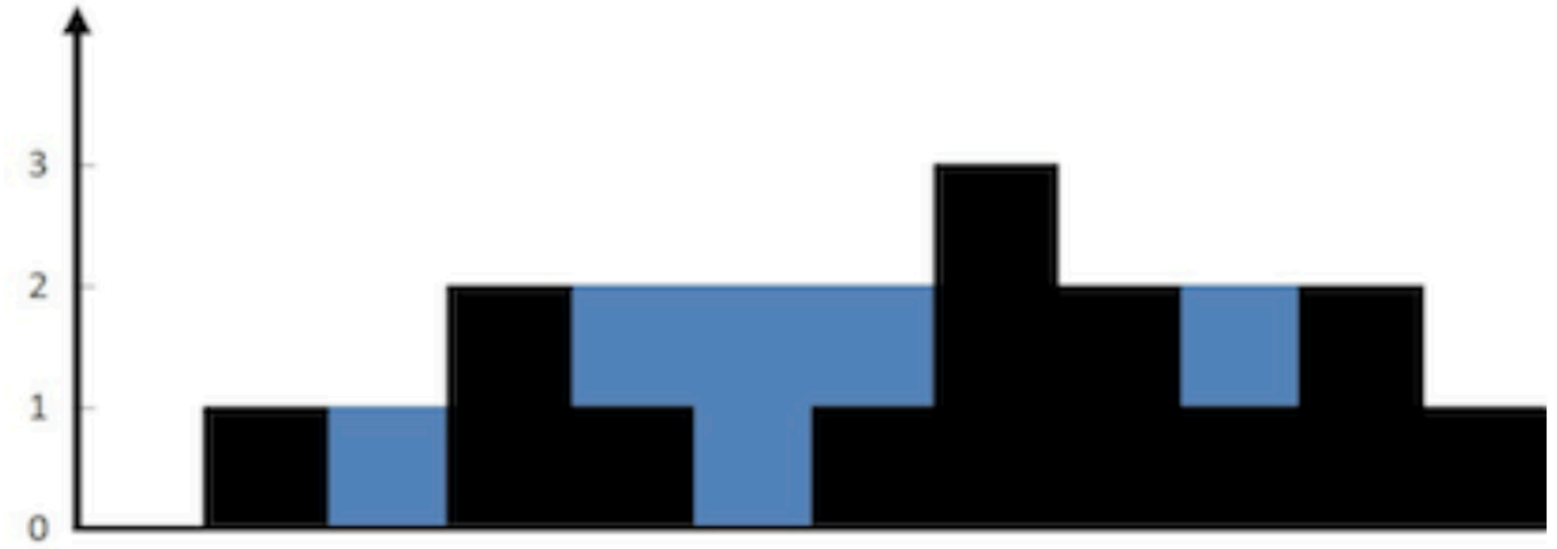
Output = ab(c)

42. Trapping Rain Water

Hard 19322 271 Add to List Share

Given `n` non-negative integers representing an elevation map where the width of each bar is `1`, compute how much water it can trap after raining.

Example 1:



```
Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
Explanation: The above elevation map (black section) is represented
by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain
water (blue section) are being trapped.
```

Example 2:

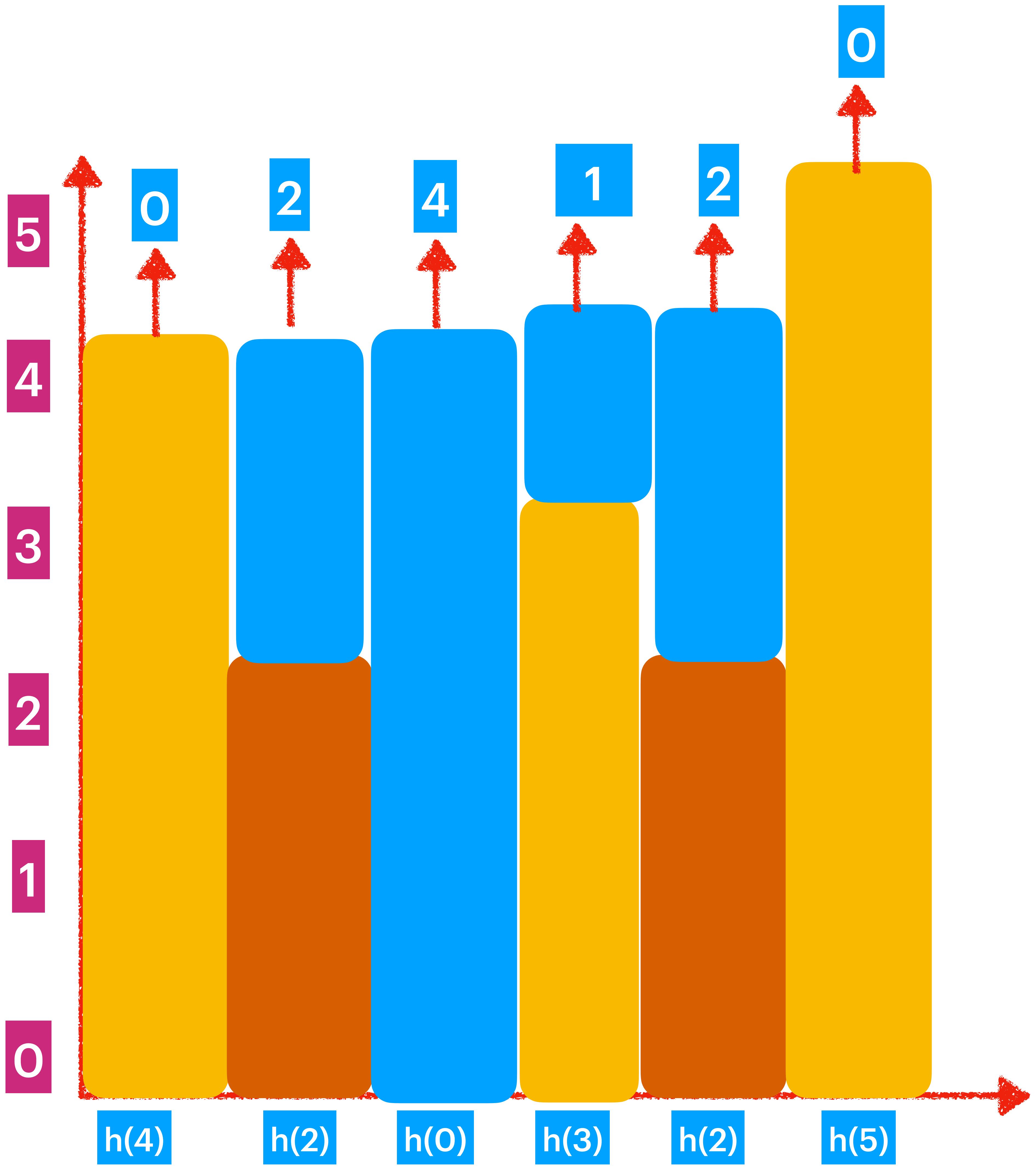
```
Input: height = [4,2,0,3,2,5]
Output: 9
```

Constraints:

- `n == height.length`
- $1 <= n <= 2 * 10^4$
- $0 <= height[i] <= 10^5$

{4,2,0,3,2,5}

Total Units Of water we can store is 9

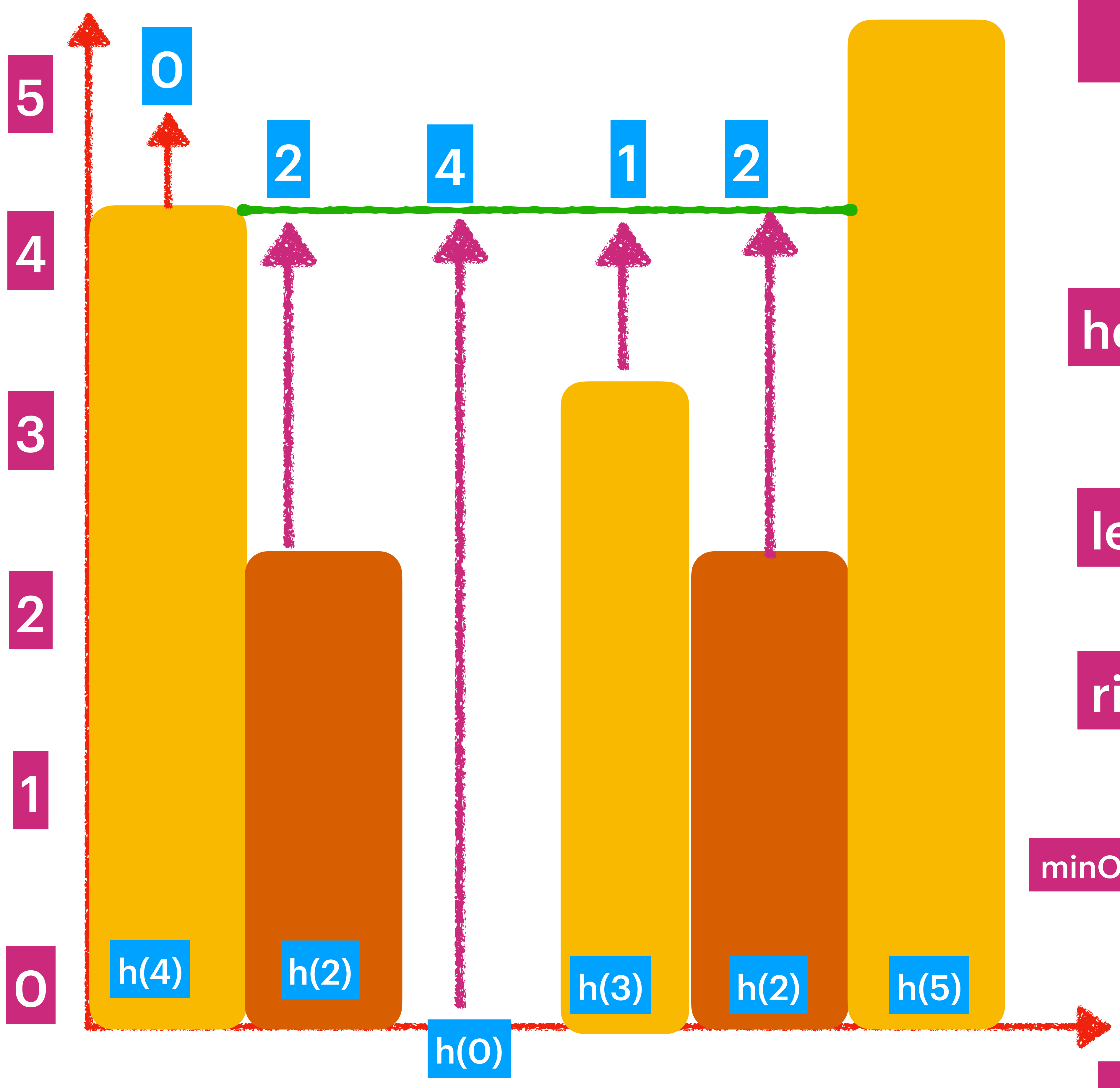


🟦 → Water

{4,2,0,3,2,5}

🟦 → Water

Total Units Of water we can store is 9

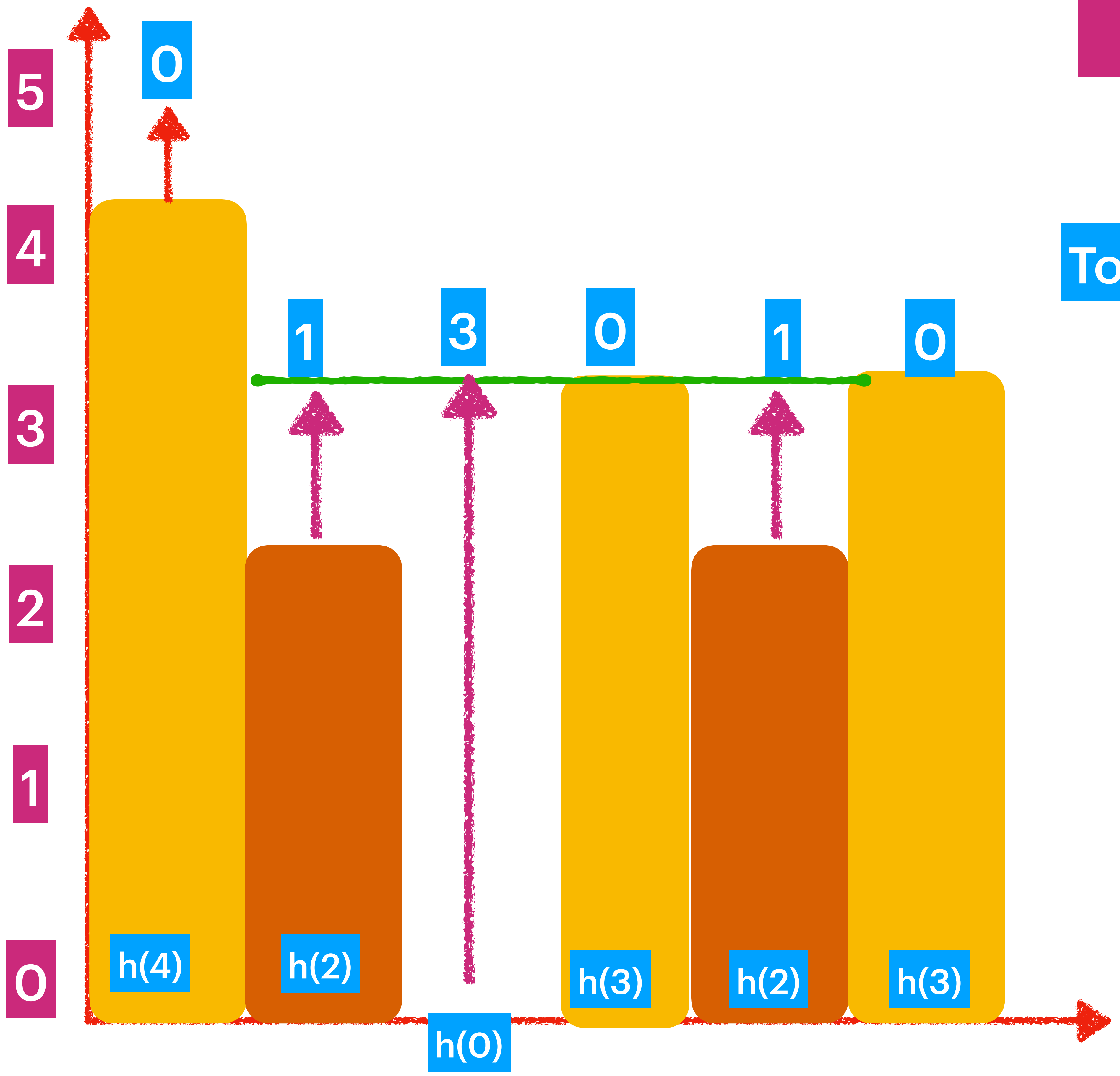


For any Index -> Min (leftMax, rightMax) - height(i) ;
If and Only If the value > 0

	0	1	2	3	4	5
height[]	4	2	0	3	2	5
leftMax[]	0	4	4	4	4	4
rightMax[]	5	5	5	5	5	0
minOfLeftRightMax[]	0	4	4	4	4	0

Time Complexity : O(n)
Space Complexity : O(n)

{4,2,0,3,2,3}



For any Index -> Min (leftMax, rightMax) - height(i) ;
If and Only If the value > 0

Total Units Of water we can store is 5