

## 3542. Minimum Operations to Convert All Elements to Zero

Solved 

Medium

 Topics

 Companies

 Hint

You are given an array `nums` of size `n`, consisting of **non-negative** integers. Your task is to apply some (possibly zero) operations on the array so that **all** elements become 0.

In one operation, you can select a **subarray** `[i, j]` (where `0 <= i <= j < n`) and set all occurrences of the **minimum non-negative** integer in that subarray to 0.

Return the **minimum** number of operations required to make all elements in the array 0.

### Example 1:

**Input:** `nums = [0,2]`

**Output:** 1

**Explanation:**

- Select the subarray `[1, 1]` (which is `[2]`), where the minimum non-negative integer is 2. Setting all occurrences of 2 to 0 results in `[0, 0]`.
- Thus, the minimum number of operations required is 1.

### Example 2:

**Input:** `nums = [3, 1, 2, 1]`

**Output:** 3

**Explanation:**

- Select subarray `[1, 3]` (which is `[1, 2, 1]`), where the minimum non-negative integer is 1. Setting all occurrences of 1 to 0 results in `[3, 0, 2, 0]`.
- Select subarray `[2, 2]` (which is `[2]`), where the minimum non-negative integer is 2. Setting all occurrences of 2 to 0 results in `[3, 0, 0, 0]`.
- Select subarray `[0, 0]` (which is `[3]`), where the minimum non-negative integer is 3. Setting all occurrences of 3 to 0 results in `[0, 0, 0, 0]`.
- Thus, the minimum number of operations required is 3.

### Example 3:

**Input:** `nums = [1, 2, 1, 2, 1, 2]`

**Output:** 4

**Explanation:**

- Select subarray `[0, 5]` (which is `[1, 2, 1, 2, 1, 2]`), where the minimum non-negative integer is 1. Setting all occurrences of 1 to 0 results in `[0, 2, 0, 2, 0, 2]`.
- Select subarray `[1, 1]` (which is `[2]`), where the minimum non-negative integer is 2. Setting all occurrences of 2 to 0 results in `[0, 0, 0, 2, 0, 2]`.
- Select subarray `[3, 3]` (which is `[2]`), where the minimum non-negative integer is 2. Setting all occurrences of 2 to 0 results in `[0, 0, 0, 0, 0, 2]`.
- Select subarray `[5, 5]` (which is `[2]`), where the minimum non-negative integer is 2. Setting all occurrences of 2 to 0 results in `[0, 0, 0, 0, 0, 0]`.
- Thus, the minimum number of operations required is 4.

### Constraints:

- `1 <= n == nums.length <= 105`
- `0 <= nums[i] <= 105`

## Python:

class Solution:

```
def minOperations(self, nums: list[int]) -> int:
    stack = []
    res = 0
    for n in nums:
        while stack and stack[-1] > n:
            stack.pop()
        if n == 0:
            continue
        if not stack or stack[-1] < n:
            res += 1
            stack.append(n)
    return res
```

## JavaScript:

```
const minOperations = nums => {
    const stk = [];
    let res = 0;
    for (const n of nums) {
        while (stk.length && stk.at(-1) > n)
            stk.pop();
        if (n === 0)
            continue;
        if (!stk.length || stk.at(-1) < n) {
            res++;
            stk.push(n);
        }
    }
    return res;
};
```

## Java:

```
class Solution {
    public static int minOperations(int[] nums) {
```

```

int n = nums.length;
int ans = 0;
boolean[] isNumEncountered = new boolean[100001];
int[] monoStack = new int[n];
int size = 0;

for (int i = 0; i < n; i++) {
    int curr = nums[i];

    if (curr == 0) {
        while (size > 0) {
            isNumEncountered[monoStack[--size]] = false;
        }
        continue;
    }

    while (size > 0 && monoStack[size - 1] > curr) {
        isNumEncountered[monoStack[--size]] = false;
    }

    if (!isNumEncountered[curr]) {
        ans++;
        isNumEncountered[curr] = true;
    }

    monoStack[size++] = curr;
}

return ans;
}

```