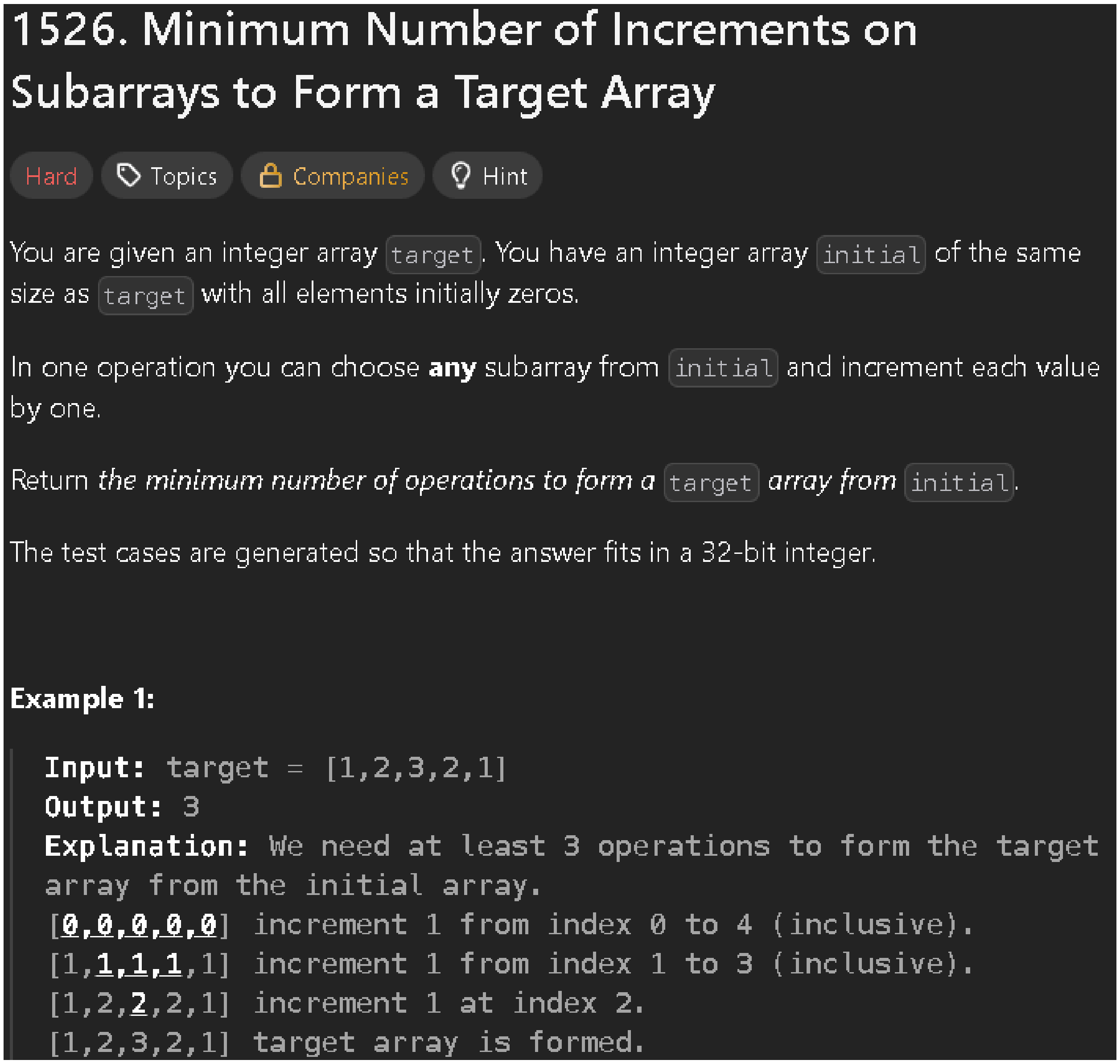


1526. Minimum Number of Increments on Subarrays to Form a Target Array

You are given an integer array `target`. You have an integer array `initial` of the same size as `target` with all elements initially zeros.

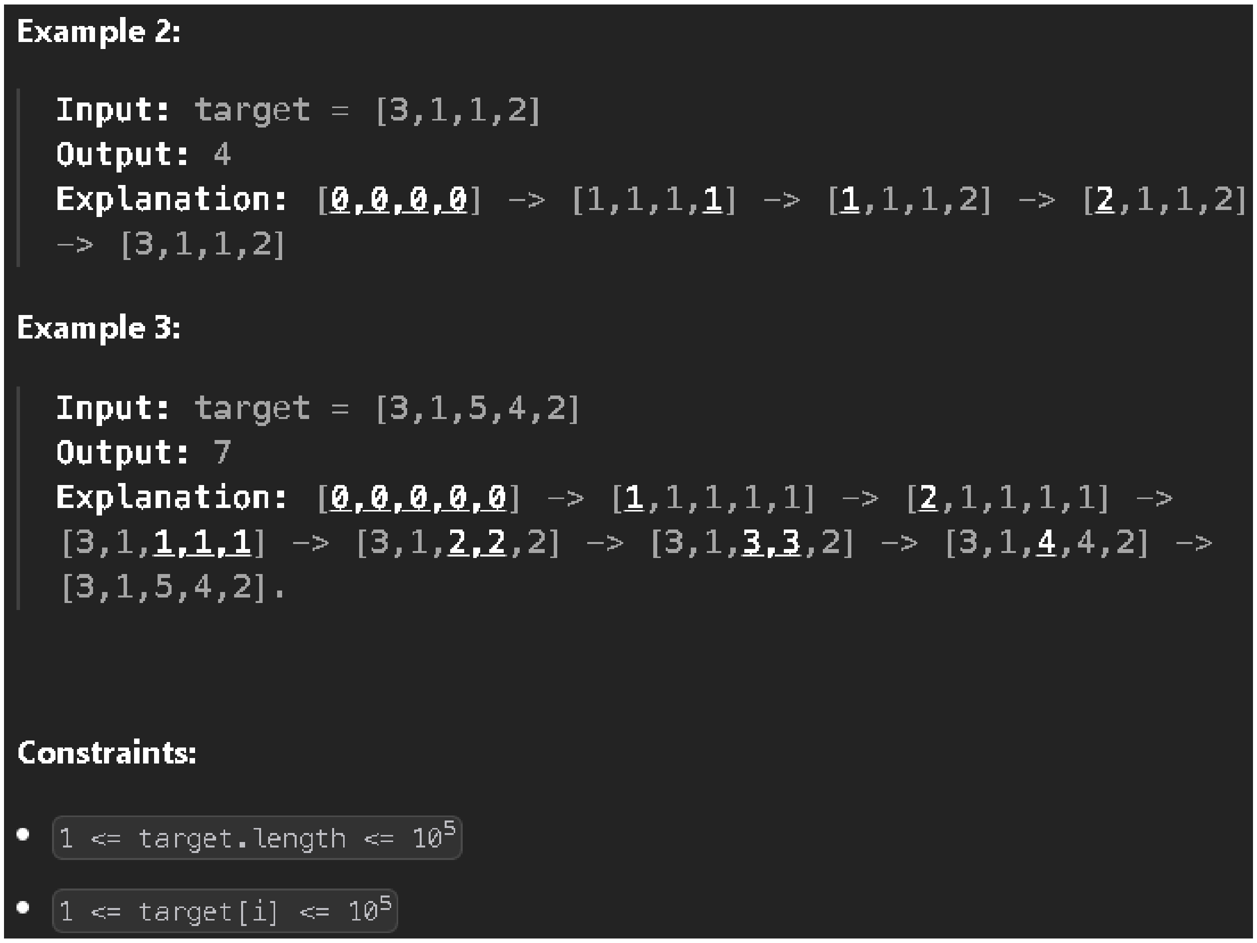
In one operation you can choose **any** subarray from `initial` and increment each value by one.

Return *the minimum number of operations to form a* `target` *array from* `initial`.

The test cases are generated so that the answer fits in a 32-bit integer.

Example 1:

```
Input: target = [1,2,3,2,1]
Output: 3
Explanation: We need at least 3 operations to form the target
array from the initial array.
[0,0,0,0,0] increment 1 from index 0 to 4 (inclusive).
[1,1,1,1,1] increment 1 from index 1 to 3 (inclusive).
[1,2,2,2,1] increment 1 at index 2.
[1,2,3,2,1] target array is formed.
```


Example 2:

```
Input: target = [3,1,1,2]
Output: 4
Explanation: [0,0,0,0] -> [1,1,1,1] -> [1,1,1,2] -> [2,1,1,2]
-> [3,1,1,2]
```

Example 3:

```
Input: target = [3,1,5,4,2]
Output: 7
Explanation: [0,0,0,0,0] -> [1,1,1,1,1] -> [2,1,1,1,1] ->
[3,1,1,1,1] -> [3,1,2,2,2] -> [3,1,3,3,2] -> [3,1,4,4,2] ->
[3,1,5,4,2].
```

Constraints:

- `1 <= target.length <= 10^5`
- `1 <= target[i] <= 10^5`

Python:

```
class Solution:
    def minNumberOperations(self, A):
        return sum(max(b - a, 0) for b, a in zip(A, [0] + A))
```

Javascript:

```
var minNumberOperations = function(target) {
        let totalOps = 0;
        let whereIAmNow = 0;
        for(let i = 0; i<target.length; i++){
                let whereINeedToBe = target[i];
                if(whereIAmNow <= whereINeedToBe){
                        // we only increment totalOps here
                        totalOps = totalOps + whereINeedToBe - whereIAmNow
                }
                whereIAmNow = whereINeedToBe

        }
        return totalOps
};
```

Java:

```
class Solution {
    public int minNumberOperations(int[] A) {
        int res = A[0];
        for (int i = 1; i < A.length; ++i)
            res += Math.max(A[i] - A[i - 1], 0);
        return res;
    }
}
```