

2025. Maximum Number of Ways to Partition an Array

Description

You are given a **0-indexed** integer array `nums` of length `n`. The number of ways to **partition** `nums` is the number of `pivot` indices that satisfy both conditions:

- `1 <= pivot < n`
- `nums[0] + nums[1] + ... + nums[pivot - 1] == nums[pivot] + nums[pivot + 1] + ... + nums[n - 1]`

You are also given an integer `k`. You can choose to change the value of **one** element of `nums` to `k`, or to leave the array **unchanged**.

Return *the maximum possible number of ways to partition `nums` to satisfy both conditions after changing **at most** one element*.

Example 1:

Input: `nums = [2,-1,2], k = 3`
Output: `1`
Explanation: One optimal approach is to change `nums[0]` to `k`. The array becomes `[3,-1,2]`.
There is one way to partition the array:
- For `pivot = 2`, we have the partition `[3,-1 | 2]`: `3 + -1 == 2`.

Example 2:

Input: `nums = [0,0,0], k = 1`
Output: `2`
Explanation: The optimal approach is to leave the array unchanged.
There are two ways to partition the array:
- For `pivot = 1`, we have the partition `[0 | 0,0]`: `0 == 0 + 0`.
- For `pivot = 2`, we have the partition `[0,0 | 0]`: `0 + 0 == 0`.

Example 3:

Input: `nums = [22,4,-25,-20,-15,15,-16,7,19,-10,0,-13,-14], k = -33`
Output: `4`
Explanation: One optimal approach is to change `nums[2]` to `k`. The array becomes `[22,4, -33,-20,-15,15,-16,7,19,-10,0,-13,-14]`.
There are four ways to partition the array.

Constraints:

- `n == nums.length`
- `2 <= n <= 105`
- `-105 <= k, nums[i] <= 105`

