# 题目

给你一个下标从0开始的整数数组nums和一个整数threshold。

请你从 nums 的子数组中找出以下标 l 开头、下标 r 结尾 (0 <= l <= r < nums.length) 且满足以下条件的 最长子数组 ：

nums[l] % 2 == 0

对于范围[l, r - 1]内的所有下标i，nums[i] % 2 != nums[i + 1] % 2

对于范围[l, r]内的所有下标i，nums[i] <= threshold

以整数形式返回满足题目要求的最长子数组的长度。

注意：子数组 是数组中的一个连续非空元素序列。

示例 1：

输入：nums = [3,2,5,4], threshold = 5

输出：3

解释：在这个示例中，我们选择从l = 1开始、到r = 3结束的子数组=> [2,5,4]，满足上述条件。

因此，答案就是这个子数组的长度3。可以证明3是满足题目要求的最大长度。

示例 2：

输入：nums = [1,2], threshold = 2

输出：1

解释：

在这个示例中，我们选择从l = 1开始、到r = 1结束的子数组=> [2]。

该子数组满足上述全部条件。可以证明 1 是满足题目要求的最大长度。

示例 3：

输入：nums = [2,3,4,5], threshold = 4

输出：3

解释：

在这个示例中，我们选择从l = 0开始、到r = 2结束的子数组=> [2,3,4]。

该子数组满足上述全部条件。

因此，答案就是这个子数组的长度3。可以证明3是满足题目要求的最大长度。

提示：

1 <= nums.length <= 100

1 <= nums[i] <= 100

1 <= threshold <= 100

# 分析

## 方法一：枚举

class Solution {

public:

bool isSatisfied(vector<int> &nums, int l, int r, int threshold) {

if (nums[l] % 2 != 0) {

return false;

}

for (int i = l; i <= r; i++) {

if (nums[i] > threshold || (i < r && nums[i] % 2 == nums[i + 1] % 2)) {

return false;

}

}

return true;

}

int longestAlternatingSubarray(vector<int>& nums, int threshold) {

int res = 0, n = nums.size();

for (int l = 0; l < n; l++) {

for (int r = l; r < n; r++) {

if (isSatisfied(nums, l, r, threshold)) {

res = max(res, r - l + 1);

}

}

}

return res;

}

};

复杂度分析

时间复杂度：O(n^3)，其中 n是 nums的长度。三重循环需要O(n^3)。

空间复杂度：O(1)。

## 方法二：动态规划