1438. 绝对差不超过限制的最长连续子数组

ITCharge 大约 2 分钟

- 标签：队列、数组、有序集合、滑动窗口、单调队列、堆（优先队列）
- 难度：中等

题目链接

- 1438. 绝对差不超过限制的最长连续子数组 - 力扣

题目大意

给定一个整数数组 `nums` ，和一个表示限制的整数 `limit` 。

要求：返回最长连续子数组的长度，该子数组中的任意两个元素之间的绝对差必须小于或者等于 `limit` 。

如果不存在满足条件的子数组，则返回 `0` 。

解题思路

求最长连续子数组，可以使用滑动窗口来解决。这道题目的难点在于如何维护滑动窗口内的最大值和最小值的差值。遍历滑动窗口求最大值和最小值，每次计算的时间复杂度为 $O(k)$ ，时间复杂度过高。考虑使用特殊的数据结构来降低时间复杂度。可以使用堆（优先队列）来解决。这里使用 `Python` 中 `heapq` 实现。具体做法如下：

- 使用 `left` 、 `right` 两个指针，分别指向滑动窗口的左右边界，保证窗口中最大值和最小值的差值不超过 `limit` 。
- 一开始， `left` 、 `right` 都指向 `0` 。
- 向右移动 `right` ，将最右侧元素加入当前窗口和大顶堆、小顶堆中。
- 如果大顶堆堆顶元素和小顶堆堆顶元素大于 `limit` ，则不断右移 `left` ，缩小滑动窗口长度，并更新窗口内的大顶堆、小顶堆。
- 如果大顶堆堆顶元素和小顶堆堆顶元素小于等于 `limit` ，则更新最长连续子数组长度。
- 然后继续右移 `right` ，直到 `right >= len(nums)` 结束。

- 输出答案。

代码

```py
import heapq

class Solution:
    def longestSubarray(self, nums: List[int], limit: int) -> int:
        size = len(nums)
        heap_max = []
        heap_min = []

        ans = 0
        left, right = 0, 0
        while right < size:
            heapq.heappush(heap_max, [-nums[right], right])
            heapq.heappush(heap_min, [nums[right], right])

            while -heap_max[0][0] - heap_min[0][0] > limit:
                while heap_min[0][1] <= left:
                    heapq.heappo  ap_min)
                while heap_max[0  <= left:
                    heapq.heappop(heap_max)
                left += 1
            ans = max(ans, right - left + 1)
            right += 1

        return ans
```