Problem 3364: Minimum Positive Sum Subarray

Problem Information

Difficulty: Easy
Acceptance Rate: 44.60%
Paid Only: No
Tags: Array, Sliding Window, Prefix Sum

Problem Description

You are given an integer array `nums` and **two** integers `l` and `r`. Your task is to find the **minimum** sum of a **subarray** whose size is between `l` and `r` (inclusive) and whose sum is greater than 0.

Return the **minimum** sum of such a subarray. If no such subarray exists, return -1.

A **subarray** is a contiguous **non-empty** sequence of elements within an array.

Example 1:

Input: nums = [3, -2, 1, 4], l = 2, r = 3

Output: 1

Explanation:

The subarrays of length between `l = 2` and `r = 3` where the sum is greater than 0 are:

* `[3, -2]` with a sum of 1 * `[1, 4]` with a sum of 5 * `[3, -2, 1]` with a sum of 2 * `[-2, 1, 4]` with a sum of 3

Out of these, the subarray `[3, -2]` has a sum of 1, which is the smallest positive sum. Hence, the answer is 1.

Example 2:

Input: nums = [-2, 2, -3, 1], l = 2, r = 3

Output: -1

Explanation:

There is no subarray of length between `l` and `r` that has a sum greater than 0. So, the answer is -1.

Example 3:

Input: nums = [1, 2, 3, 4], l = 2, r = 4

Output: 3

Explanation:

The subarray `[1, 2]` has a length of 2 and the minimum sum greater than 0. So, the answer is 3.

Constraints:

* `1 <= nums.length <= 100` * `1 <= l <= r <= nums.length` * `-1000 <= nums[i] <= 1000`

Code Snippets

C++:

```
class Solution {
public:
int minimumSumSubarray(vector<int>& nums, int l, int r) {

}
};
```

Java:

```
class Solution {
public int minimumSumSubarray(List<Integer> nums, int l, int r) {
}
}
```

Python3:

```
class Solution:
def minimumSumSubarray(self, nums: List[int], l: int, r: int) -> int:
```