209. Minimum Size Subarray Sum

Medium | Topics | Companies

Given an array of positive integers `nums` and a positive integer `target`, return *the* ***minimal length*** *of a subarray whose sum is greater than or equal to* `target`. If there is no such subarray, return `0` instead.

Example 1:

> **Input:** target = 7, nums = [2,3,1,2,4,3]
> **Output:** 2
> **Explanation:** The subarray [4,3] has the minimal length under the problem constraint.

Example 2:

> **Input:** target = 4, nums = [1,4,4]
> **Output:** 1

Example 3:

> **Input:** target = 11, nums = [1,1,1,1,1,1,1,1]
> **Output:** 0

Constraints:

- `1 <= target <= 10^9`
- `1 <= nums.length <= 10^5`
- `1 <= nums[i] <= 10^4`

Follow up: If you have figured out the `O(n)` solution, try coding another solution of which the time complexity is `O(n log(n))`.

Approach 1: Brute force

$T(n) : O(n^2)$

$S(n) : O(1)$

Approach 2: Using sliding window

→ $l = 0$, $r = 0$

→ l is start of window and r is end of window

→ add a_r to window

→ if window sum is $\geq$ Target

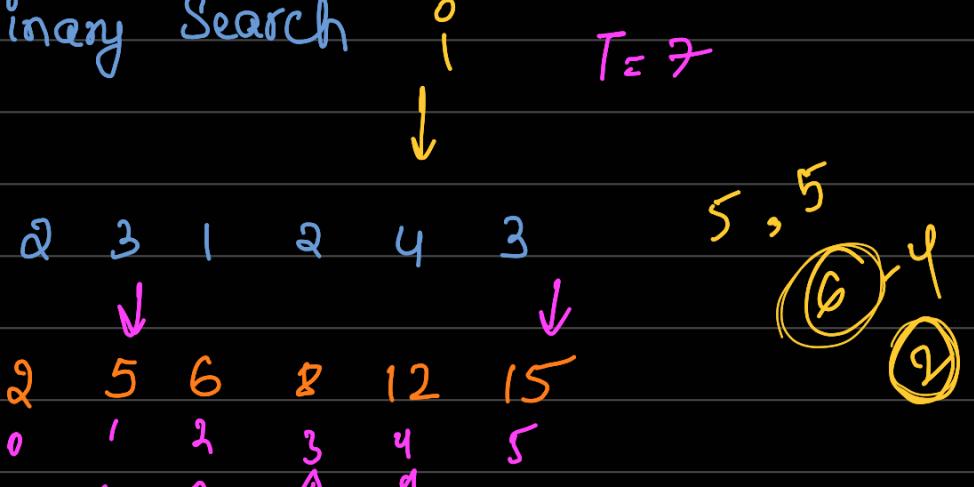
Then see if inner subarrays of window also has sum $\geq$ Target by shrinking the window

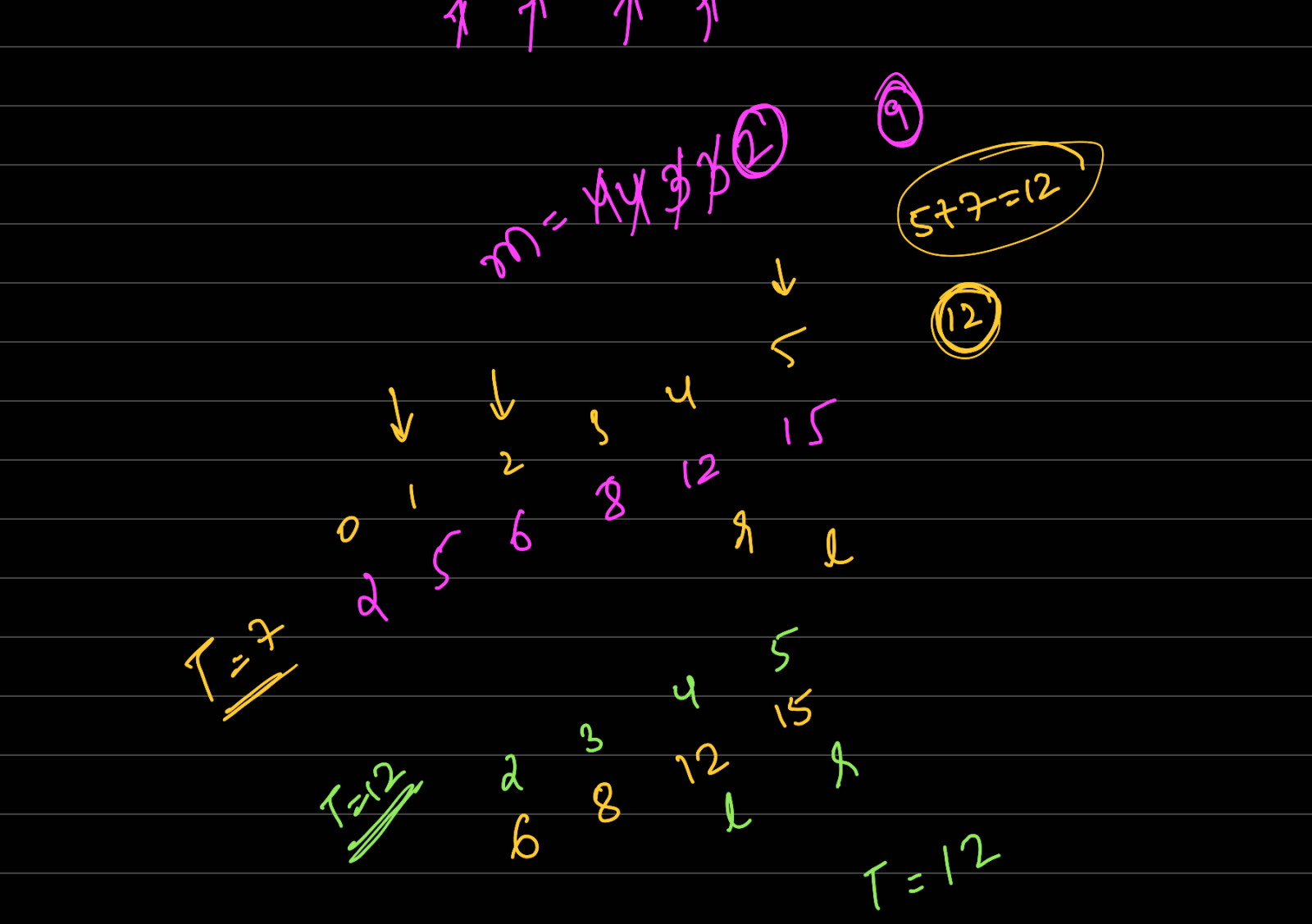
also has sum ≥ Target by shrinking the window and update answer accordingly.

→ if window sum is less than Target then shrinking the window further also doesn't has sum ≥ Target. So increase the size of window.

```
l = 0, r = 0, sum = 0, ans = INT_MAX

while (r < nums size)
{
    sum = sum + nums[r]

    while (l < nums size && sum ≥ Target)
    {
        ans = min(ans, r - l + 1)
        sum = sum - nums[l]
        l++
    }

    r++
}

return ans == INTMAX ? 0 : ans
```

Approach 3: binary Search




m = 4 4 3 3 2
9
5+7=12
12
T=7
T=12
T=12
T=16

2 3 1 2 4

0 1 2 3 4 5

2 5 6 8 12 15

T = 7

$a_T = 15$

$(l < n-1) \&\& (a_m \leq Target \;\&\&\; a_{m+1} \leq Target)$

T = 4

T = 9

0 1 2

1 4 4

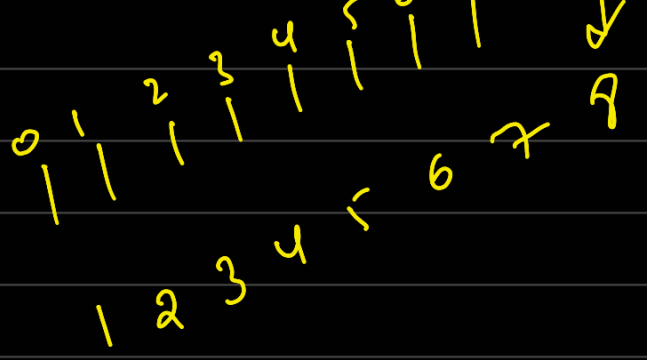
1 5 9

l r

ans = 2

T = 11

1 2 3 4 5 6 7


0 1 2 3 4 5 6 7 8 9
1 2 3 4 5 6


T=11