Shortest Subarray With OR At Least K II

solved by	Senan
Platform	LeetCode
difficulty	Medium
Serial	3097
tags	Bit Manipulation
language	C++
solved on	@10/11/2024
link	https://leetcode.com/problems/shortest-subarray-with-or-at-least-k-ii/
Completion	✓

Intuition

The goal is to find the smallest subarray length in `nums` whose bitwise OR is greater than or equal to `k`. By calculating and updating the bitwise OR using bit manipulation, we can efficiently check each subarray.

Approach

1. **Bitwise Representation**: Use a vector `bits` of size 32 (for each bit position in a 32-bit integer) to keep track of the OR result of the current subarray. Each entry in `bits` represents whether the corresponding bit is set in the cumulative OR result.
2. **Sliding Window with Two Pointers**: Use two pointers, `i` (start) and `j` (end), to define the subarray:
   - Increment `j` to expand the subarray.
   - Update `bits` using the helper function `updateBits()` when adding or removing elements. This function updates each bit based on the binary representation of the number being added or removed.
3. **Checking Condition**: Once `getInt(bits)` (the integer representation of the current `bits` vector) meets or exceeds `k`, calculate the length of the subarray and update `answer` if this length is shorter. Then, increment `i` to potentially find a shorter subarray.
4. **Edge Case**: If no valid subarray is found, return -1.

Complexity

Time Complexity:

The time complexity is **O(n)**, where **n** is the length of `nums`. Each element is processed a constant number of times due to the sliding window approach.

Space Complexity:

The space complexity is **O(1)**, considering the `bits` array size is fixed at 32.

Code

```
class Solution {
    int getInt(vector<int>& bits) {
        int answer = 0;
        for (int i = 0; i < 32; i++)
            answer |= bits[i] ? 1 << i : 0;
        return answer;
    }

    void updateBits(int num, int factor, vector<int>& bits) {
        for (int i = 0; i < 32; i++)
            bits[i] += factor * ((num >> i) & 1);
    }

public:
    int minimumSubarrayLength(vector<int>& nums, int k) {
        int i = 0, j = 0;
        int answer = INT_MAX;
        vector<int> bits(32, 0);

        while (j < nums.size()) {
            updateBits(nums[j++], 1, bits);

            while (i < j && getInt(bits) >= k) {
                answer = min(answer, j - i);
                updateBits(nums[i++], -1, bits);
            }
        }

        return answer == INT_MAX ? -1 : answer;
    }
};
```