

Problem 3117: Minimum Sum of Values by Dividing Array

Problem Information

Difficulty: Hard

Acceptance Rate: 27.39%

Paid Only: No

Tags: Array, Binary Search, Dynamic Programming, Bit Manipulation, Segment Tree, Queue

Problem Description

You are given two arrays `nums` and `andValues` of length `n` and `m` respectively.

The **value** of an array is equal to the **last** element of that array.

You have to divide `nums` into `m` **disjoint contiguous** subarrays such that for the `ith` subarray `[li, ri]`, the bitwise `AND` of the subarray elements is equal to `andValues[i]`, in other words, `nums[li] & nums[li + 1] & ... & nums[ri] == andValues[i]` for all `1 <= i <= m`, where `&` represents the bitwise `AND` operator.

Return _the**minimum** possible sum of the **values** of the _`m` _subarrays_`_nums` _is divided into_. _If it is not possible to divide_`nums` _into_`m` _subarrays satisfying these conditions, return_`-1`.

Example 1:

Input: `nums = [1,4,3,3,2]`, `andValues = [0,3,3,2]

Output: 12

Explanation:

The only possible way to divide `nums` is:

1. `'[1,4]` as `1 & 4 == 0`.
2. `[3]` as the bitwise `AND` of a single element subarray is that element itself.
3. `[3]` as the bitwise `AND` of a single element subarray is that element itself.
4. `[2]` as the bitwise `AND` of a single element subarray is that element itself.

The sum of the values for these subarrays is `4 + 3 + 3 + 2 = 12`.

Example 2:

Input: nums = [2,3,5,7,7,7,5], andValues = [0,7,5]

Output: 17

Explanation:

There are three ways to divide `nums`:

1. `[[2,3,5],[7,7,7],[5]]` with the sum of the values `5 + 7 + 5 == 17`.
2. `[[2,3,5,7],[7,7],[5]]` with the sum of the values `7 + 7 + 5 == 19`.
3. `[[2,3,5,7,7],[7],[5]]` with the sum of the values `7 + 7 + 5 == 19`.

The minimum possible sum of the values is `17`.

Example 3:

Input: nums = [1,2,3,4], andValues = [2]

Output: -1

Explanation:

The bitwise `AND` of the entire array `nums` is `0`. As there is no possible way to divide `nums` into a single subarray to have the bitwise `AND` of elements `2`, return `-1`.

Constraints:

* `1 <= n == nums.length <= 104` * `1 <= m == andValues.length <= min(n, 10)` * `1 <= nums[i] < 105` * `0 <= andValues[j] < 105`

Code Snippets

C++:

```
class Solution {  
public:  
    int minimumValueSum(vector<int>& nums, vector<int>& andValues) {  
  
    }  
};
```

Java:

```
class Solution {  
    public int minimumValueSum(int[] nums, int[] andValues) {  
  
    }  
}
```

Python3:

```
class Solution:  
    def minimumValueSum(self, nums: List[int], andValues: List[int]) -> int:
```