Problem 3685: Subsequence Sum After Capping Elements

Problem Information

Difficulty: Medium
Acceptance Rate: 19.03%
Paid Only: No
Tags: Array, Two Pointers, Dynamic Programming, Sorting

Problem Description

You are given an integer array `nums` of size `n` and a positive integer `k`.

An array **capped** by value `x` is obtained by replacing every element `nums[i]` with `min(nums[i], x)`.

For each integer `x` from 1 to `n`, determine whether it is possible to choose a **subsequence** from the array capped by `x` such that the sum of the chosen elements is **exactly** `k`.

Return a **0-indexed** boolean array `answer` of size `n`, where `answer[i]` is `true` if it is possible when using `x = i + 1`, and `false` otherwise.

Example 1:

Input: nums = [4,3,2,4], k = 5

Output: [false,false,true,true]

Explanation:

* For `x = 1`, the capped array is `[1, 1, 1, 1]`. Possible sums are `1, 2, 3, 4`, so it is impossible to form a sum of `5`. * For `x = 2`, the capped array is `[2, 2, 2, 2]`. Possible sums are `2, 4, 6, 8`, so it is impossible to form a sum of `5`. * For `x = 3`, the capped array is `[3, 3, 2, 3]`. A subsequence `[2, 3]` sums to `5`, so it is possible. * For `x = 4`, the capped array is `[4, 3, 2, 4]`. A subsequence `[3, 2]` sums to `5`, so it is possible.

Example 2:

Input: nums = [1,2,3,4,5], k = 3

Output: [true,true,true,true,true]

Explanation:

For every value of `x`, it is always possible to select a subsequence from the capped array that sums exactly to `3`.

Constraints:

* `1 <= n == nums.length <= 4000` * `1 <= nums[i] <= n` * `1 <= k <= 4000`

Code Snippets

C++:

```
class Solution {
public:
vector<bool> subsequenceSumAfterCapping(vector<int>& nums, int k) {

}
};
```

Java:

```
class Solution {
public boolean[] subsequenceSumAfterCapping(int[] nums, int k) {

}
}
```

Python3:

```
class Solution:
def subsequenceSumAfterCapping(self, nums: List[int], k: int) -> List[bool]:
```