Problem 1681: Minimum Incompatibility

Problem Information

Difficulty: Hard
Acceptance Rate: 40.68%
Paid Only: No
Tags: Array, Dynamic Programming, Bit Manipulation, Bitmask

Problem Description

You are given an integer array `nums`■■■ and an integer `k`. You are asked to distribute this array into `k` subsets of **equal size** such that there are no two equal elements in the same subset.

A subset's **incompatibility** is the difference between the maximum and minimum elements in that array.

Return _the**minimum possible sum of incompatibilities** of the _`k` _subsets after distributing the array optimally, or return_`-1` _if it is not possible._

A subset is a group integers that appear in the array with no particular order.

Example 1:

Input: nums = [1,2,1,4], k = 2 **Output:** 4 **Explanation:** The optimal distribution of subsets is [1,2] and [1,4]. The incompatibility is (2-1) + (4-1) = 4. Note that [1,1] and [2,4] would result in a smaller sum, but the first subset contains 2 equal elements.

Example 2:

Input: nums = [6,3,8,1,3,1,2,2], k = 4 **Output:** 6 **Explanation:** The optimal distribution of subsets is [1,2], [2,3], [6,8], and [1,3]. The incompatibility is (2-1) + (3-2) + (8-6) + (3-1) = 6.

Example 3:

Input: nums = [5,3,3,6,3,3], k = 3 **Output:** -1 **Explanation:** It is impossible to distribute nums into 3 subsets where no two elements are equal in the same subset.

Constraints:

* `1 <= k <= nums.length <= 16` * `nums.length` is divisible by `k` * `1 <= nums[i] <= nums.length`

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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