Usecase 1)**Identification of Unique Number of Occurrences**

Given an array of integers arr, return true *if the number of occurrences of each value in the array is****unique****or*false*otherwise*.

def count\_fre(arr):

count\_map={}

for num in arr:

count\_map[num]=count\_map.get(num,0)+1

unique\_counts=set(count\_map.values())

print(unique\_counts)

print(count\_map)

return len(unique\_counts) == len(count\_map)

arr=[1,2,2,3]

print(count\_fre(arr))

**Example 1:**

**Input:** arr = [1,2,2,1,1,3]

**Output:** true

**Explanation:** The value 1 has 3 occurrences, 2 has 2 and 3 has 1. No two values have the same number of occurrences.

**Example 2:**

**Input:** arr = [1,2]

**Output:** false

**Example 3:**

**Input:** arr = [-3,0,1,-3,1,1,1,-3,10,0]

**Output:** true

**Constraints:**

* 1 <= arr.length <= 100
* -100 <= arr[i] <= 100

**Usecase 2) Kth Largest Element in an Array**

Given an integer array nums and an integer k, return the kth largest element in the array.

Note that it is the kth largest element in the sorted order, not the kth distinct element.

Can you solve it without sorting?

import heapq

def find\_kth\_largest(nums, k):

min\_heap = []

for num in nums:

heapq.heappush(min\_heap, num)

if len(min\_heap) > k:

heapq.heappop(min\_heap)

return min\_heap[0]

nums = [3, 2, 1, 5, 6, 4]

k = 2

print(find\_kth\_largest(nums, k))

Example 1:

Input: nums = [3,2,1,5,6,4], k = 2

Output: 5

Example 2:

Constraints:

1 <= k <= nums.length <= 105

-104 <= nums[i] <= 104