You are given an integer array nums consisting of 2 \* n integers.

You need to divide nums into n pairs such that:

* Each element belongs to **exactly one** pair.
* The elements present in a pair are **equal**.

Return true *if nums can be divided into* n *pairs, otherwise return* false.

**Example 1:**

**Input:** nums = [3,2,3,2,2,2]

**Output:** true

**Explanation:**

There are 6 elements in nums, so they should be divided into 6 / 2 = 3 pairs.

If nums is divided into the pairs (2, 2), (3, 3), and (2, 2), it will satisfy all the conditions.

**Example 2:**

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

**Output:** false

**Explanation:**

There is no way to divide nums into 4 / 2 = 2 pairs such that the pairs satisfy every condition.

**Constraints:**

* nums.length == 2 \* n
* 1 <= n <= 500
* 1 <= nums[i] <= 500

class Solution {

public boolean divideArray(int[] nums) {

Map<Integer, Integer> mp=new HashMap();

for(int i:nums)

{

mp.put(i, mp.getOrDefault(i, 0)+1);

}

for(int i:mp.keySet())

{

if(mp.get(i)%2!=0)

return false;

}

return true;

}

}

