Average Rating: 3.20 (50 votes)

Solution

Approach 1: Brute Force

Intuition

We can try every possible `x`.

Algorithm

Since we divide the deck of `N` cards into say, `X` piles of `X` cards each, we must have `N % X == 0`.

Then, say the deck has `C_i` copies of cards with number `i`. Each group with number `i` has `X` copies, so we must have `C_i % X == 0`. These are necessary and sufficient conditions.

Java | Python — Copy

```
class Solution(object):
    def hasGroupsSizeX(self, deck):
        count = collections.Counter(deck)
        N = len(deck)
        for X in xrange(2, N+1):
            if N % X == 0:
                if all(v % X == 0 for v in count.values()):
                    return True
        return False
```

Complexity Analysis

- Time Complexity: $O(N^2 \log \log N)$, where N is the number of cards. It is outside the scope of this article to prove that the number of divisors of N is bounded by $O(N \log \log N)$.
- Space Complexity: $O(N)$.

Approach 2: Greatest Common Divisor

Intuition and Algorithm

Again, say there are `C_i` cards of number `i`. These must be broken down into piles of `X` cards each, ie. `C_i % X == 0` for all `i`.

Thus, `X` must divide the greatest common divisor of `C_i`. If this greatest common divisor `g` is greater than `1`, then `X = g` will satisfy. Otherwise, it won't.

Java | Python — Copy

```
class Solution(object):
    def hasGroupsSizeX(self, deck):
        from fractions import gcd
        vals = collections.Counter(deck).values()
        return reduce(gcd, vals) >= 2
```

Complexity Analysis

- Time Complexity: $O(N \log^2 N)$, where N is the number of votes. If there are C_i cards with number i, then each `gcd` operation is naively $O(\log^2 C_i)$. Better bounds exist, but are outside the scope of this article to develop.
- Space Complexity: $O(N)$.

Python3

Autocomplete

```
class Solution(object):
    def hasGroupsSizeX(self, deck):
        vals = collections.Counter(deck).values()
        return reduce(gcd, vals) >= 2
```