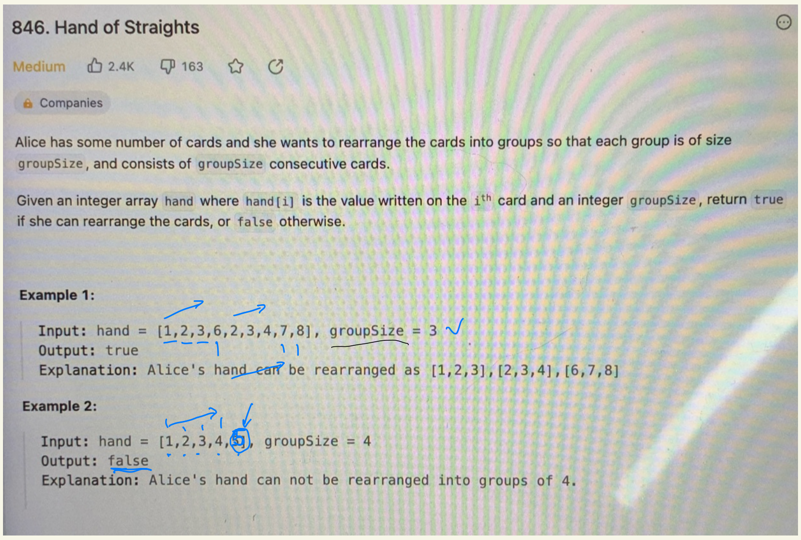
2.5



846. Hand of Straights

Alice has some number of cards and she wants to rearrange the cards into groups so that each group is of size `groupSize`, and consists of `groupSize` consecutive cards.

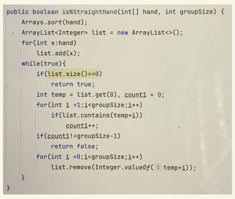
Given an integer array `hand` where `hand[i]` is the value written on the `i`th card and an integer `groupSize`, return `true` if she can rearrange the cards, or `false` otherwise.

Example 1:

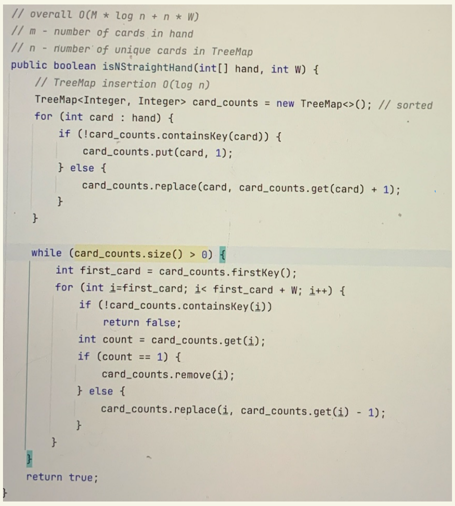
```
Input: hand = [1,2,3,6,2,3,4,7,8], groupSize = 3
Output: true
Explanation: Alice's hand can be rearranged as [1,2,3],[2,3,4],[6,7,8]
```

Example 2:

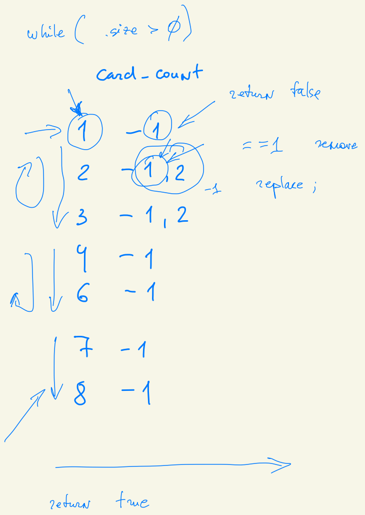
```
Input: hand = [1,2,3,4,5], groupSize = 4
Output: false
Explanation: Alice's hand can not be rearranged into groups of 4.
```

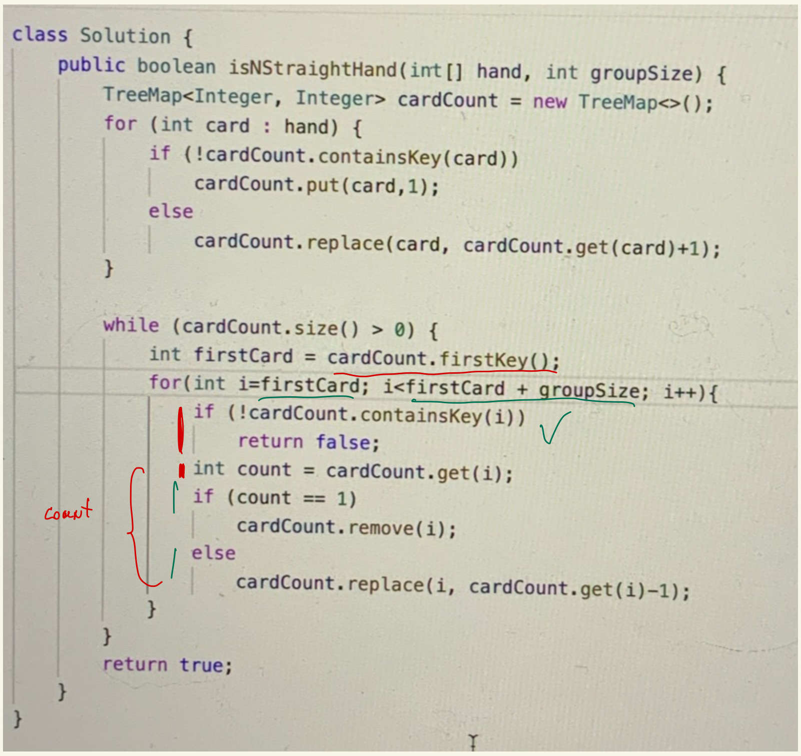


```java
public boolean isNStraightHand(int[] hand, int groupSize) {
    Arrays.sort(hand);
    ArrayList<Integer> list = new ArrayList<>();
    for(int x:hand)
        list.add(x);
    while(true){
        if(list.size()==0)
            return true;
        int temp = list.get(0), count1 = 0;
        for(int i =1;i<groupSize;i++)
            if(list.contains(temp+i))
                count1++;
        if(count1!=groupSize-1)
            return false;
        for(int i =0;i<groupSize;i++)
            list.remove(Integer.valueOf( i: temp+i));
    }
}
```

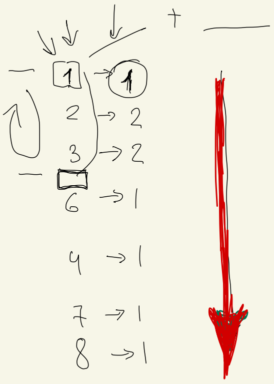


```java
// overall O(M * log n + n * W)
// m - number of cards in hand
// n - number of unique cards in TreeMap
public boolean isNStraightHand(int[] hand, int W) {
    // TreeMap insertion O(log n)
    TreeMap<Integer, Integer> card_counts = new TreeMap<>(); // sorted
    for (int card : hand) {
        if (!card_counts.containsKey(card)) {
            card_counts.put(card, 1);
        } else {
            card_counts.replace(card, card_counts.get(card) + 1);
        }
    }

    while (card_counts.size() > 0) {
        int first_card = card_counts.firstKey();
        for (int i=first_card; i< first_card + W; i++) {
            if (!card_counts.containsKey(i))
                return false;
            int count = card_counts.get(i);
            if (count == 1) {
                card_counts.remove(i);
            } else {
                card_counts.replace(i, card_counts.get(i) - 1);
            }
        }
    }
    return true;
}
```





```java
class Solution {
    public boolean isNStraightHand(int[] hand, int groupSize) {
        TreeMap<Integer, Integer> cardCount = new TreeMap<>();
        for (int card : hand) {
            if (!cardCount.containsKey(card))
                cardCount.put(card,1);
            else
                cardCount.replace(card, cardCount.get(card)+1);
        }

        while (cardCount.size() > 0) {
            int firstCard = cardCount.firstKey();
            for(int i=firstCard; i<firstCard + groupSize; i++){
                if (!cardCount.containsKey(i))
                    return false;
                int count = cardCount.get(i);
                if (count == 1)
                    cardCount.remove(i);
                else
                    cardCount.replace(i, cardCount.get(i)-1);
            }
        }
        return true;
    }
}
```



```
while (size > Ø)
for (firstCard, < firstCard + groupSize, ++)
        1  2  3  4
    if (! cont. Key)
                return false;
    count = map.get(i);
    if count == 1
        card. remove (i);
    else
        card. replace (card, map. get(i) -1)
return true
```