输入一个按升序排序的整数数组（可能包含重复数字），你需要将它们分割成几个子序列，其中每个子序列至少包含三个连续整数。返回你是否能做出这样的分割?

示例 1:

输入: [1,2,3,3,4,5]
输出: True
解释:
你可以分割出这样两个连续子序列 :
1, 2, 3
3, 4, 5

示例 2:

输入: [1,2,3,3,4,4,5,5]
输出: True
解释:
你可以分割出这样两个连续子序列 :
1, 2, 3, 4, 5
3, 4, 5

示例 3:

输入: [1,2,3,4,4,5]
输出: False

这道题就用贪婪解法就可以了，我们使用两个哈希表map，第一个map用来建立数字和其出现次数之间的映射freq，第二个用来建立可以加在某个连续子序列后的数字及其可以出现的次数之间的映射need。对于第二个map，举个例子来说，就是假如有个连，[1,2,3]，那么后面可以加上4，所以就建立4的映射。这样我们首先遍历一遍数组，统计每个数字出现的频率，然后我们开始遍历数组，对于每个遍历到的数字，首先看其当前出现的次数，如果为0，则继续循环；如果need中存在这个数字的非0映射，那么表示当前的数字可以加到某个连的末尾，我们将当前数字的映射值自减1，然后将下一个连续数字的映射值加1，因为当[1,2,3]连上4后变成[1,2,3,4]之后，就可以连上5了；如果不能连到其他子序列后面，我们来看其是否可以成为新的子序列的起点，可以通过看后面两个数字的映射值是否大于0，都大于0的话，说明可以组成3连儿，于是将后面两个数字的映射值都自减1，还有由于组成了3连儿，在need中将末尾的下一位数字的映射值自增1；如果上面情况都不满足，说明该数字是单牌，只能划单儿，直接返回false。最后别忘了将当前数字的freq映射值自减1。退出for循环后返回true，

```
class Solution {
```

```
public:
    bool isPossible(vector<int>& nums) {
        map<int,int> fre,need;
        for(int num:nums){
            fre[num]++;
        }
        for(int num:nums){
            if(fre[num]==0){
                continue;
            }else if(need[num]!=0){
                need[num]--;
                need[num+1]++;
            }else if(fre[num+1]!=0&&fre[num+2]!=0){
                need[num+3]++;
                fre[num+1]--;
                fre[num+2]--;
            }else{
                return false;
            }
            fre[num]--;
        }
        return true;
    }
};
```