入选理由

1. 堆和贪心也有”一腿“，我们来看看。

标签

- 堆
- 贪心

难度

- 中等

题目地址（1054. 距离相等的条形码）

https://leetcode-cn.com/problems/distant-barcodes/

题目描述

```
在一个仓库里，有一排条形码，其中第 i 个条形码为 barcodes[i]。

请你重新排列这些条形码，使其中两个相邻的条形码 不能 相等。 你可以返回任何满足该要求的答案，此题保证存在答案。



示例 1：

输入：[1,1,1,2,2,2]
输出：[2,1,2,1,2,1]
示例 2：

输入：[1,1,1,1,2,2,3,3]
输出：[1,3,1,3,2,1,2,1]


提示：

1 <= barcodes.length <= 10000
1 <= barcodes[i] <= 10000
```

前置知识

- 堆
- 贪心

分析

该题就是让数组重新排列使得相邻的元素互不相同，思考一下不难发现，如果某个元素超过了(len+1)/2，则必定不能实现，而题中说必有答案，因此我们只需要用贪心思想，每次安排元素的时候都取出现次数最多的前两个元素，这样多的解决了，少的自然也能解决掉。而我们每次取出后要更新一下元素频率，这样就会涉及到很多插入删除排序操作，堆再合适不过。

进阶： 你还能想到其他解决方法么？

代码：

Java

```java
class Solution {

    public int[] rearrangeBarcodes(int[] barcodes) {

        Map<Integer, Integer> counter = new HashMap<>();
        PriorityQueue<Integer> pq = new PriorityQueue<>((x, y) -> (counter.get(y) - counter.get(x)));

        for (int num : barcodes)
            counter.put(num, counter.getOrDefault(num, 0) + 1);

        for (int num : counter.keySet())
            pq.offer(num);

        int idx = 0;

        while (pq.size() > 1) {

            int first = pq.poll(), second = pq.poll();

            barcodes[idx++] = first;
            barcodes[idx++] = second;

            counter.put(first, counter.get(first) - 1);
            counter.put(second, counter.get(second) - 1);

            if (counter.get(first) > 0)
                pq.offer(first);
            if (counter.get(second) > 0)
                pq.offer(second);
        }
```

```
        if (pq.size() > 0)
            barcodes[idx++] = pq.poll();

        return barcodes;
    }
}
```

复杂度分析

设：元素个数为N，不重复元素个数为K

时间复杂度：$O(N\log K)$

空间复杂度：$O(K)$

上一页 下一页