2244. 完成所有任务需要的最少轮数

给你一个下标从 **0** 开始的整数数组 `tasks` ，其中 `tasks[i]` 表示任务的难度级别。在每一轮中，你可以完成 2 个或者 3 个 **相同难度级别** 的任务。

返回完成所有任务需要的 **最少** 轮数，如果无法完成所有任务，返回 `-1` 。

示例 1：

```
输入：tasks = [2,2,3,3,2,4,4,4,4,4]
输出：4
解释：要想完成所有任务，一个可能的计划是：
- 第一轮，完成难度级别为 2 的 3 个任务。
- 第二轮，完成难度级别为 3 的 2 个任务。
- 第三轮，完成难度级别为 4 的 3 个任务。
- 第四轮，完成难度级别为 4 的 2 个任务。
可以证明，无法在少于 4 轮的情况下完成所有任务，所以答案为 4 。
```

示例 2：

```
输入：tasks = [2,3,3]
输出：-1
解释：难度级别为 2 的任务只有 1 个，但每一轮执行中，只能选择完成 2 个或者 3 个相同难度级别的任务。因此，无法完成所有任务，答案为 -1 。
```

提示：

- `1 <= tasks.length <= 105`
- `1 <= tasks[i] <= 109`

思路：

1. 通用

首先统计不同难度级别的任务各自出现的频率，然后对频率 (≥ 1) 进行分类讨论：

- 频率是 1，说明这种任务无法完成。
- 频率是 $3 \times k$，k 为 ≥ 1 的整数。每次完成 3 个，k 轮完成。
- 频率是 $3 \times k + 2$，k 为 ≥ 0 的整数。其中 $3 \times k$ 个任务需要 k 轮完成，剩下 2 个任务需要 1 轮完成。
- 频率是 $3 \times k + 1$，k 为 ≥ 1 的整数。其中 $3 \times (k - 1)$ 个任务需要 $(k - 1)$ 轮完成，剩下 4 个任务需要 2 轮完成。

对这些情况求和即可。

Python

```python
class Solution:
    def minimumRounds(self, tasks: List[int]) -> int:
        # 贪心思考即可
        """
        1. c = 1 无法完成
        2. c = 3k k 轮
        3. c = 3k + 1 3(k - 1) + 2 * 2
        4. c = 3k + 2 3 * k + 2 * 1
        """
        cnt = Counter(tasks)
        res = 0
        for v in cnt.values():
            if v == 1:
                return -1
            if v % 3 == 0:
                res += v // 3
            else:
                res += (1 + v // 3)
        return res
```

Java

```java
class Solution {
    public int minimumRounds(int[] tasks) {
        Map<Integer, Integer> cnt = new HashMap<Integer, Integer>();
        for (int task : tasks) {
            cnt.put(task, cnt.getOrDefault(task, 0) + 1);
        }
        int res = 0;
        for (int v : cnt.values()) {
            if (v == 1) {
                return -1;
            }
            if (v % 3 == 0) {
                res += v / 3;
            } else {
                res += 1 + v / 3;
            }
        }
        return res;
    }
}
```

2. 灵神

每轮完成的都是相同难度级别的任务，假设难度为 1 的任务有 c 个，问题变成：

每轮可以把 c 减少 2，或者减少 3。把 c 减少到 0 最少要多少轮？

例如 $c=10$ 时，$3+3+2+2=10$，最少要 4 轮。

贪心地想，尽量多地使用「减少 3」，可以让轮数尽量少。

分类讨论：

- 如果 $c = 1$，无法完成，返回 -1。
- 如果 $c = 3k (k \geq 1)$，只用「减少 3」就能完成，轮数为 $\frac{c}{3}$。
- 如果 $c = 3k + 1 (k \geq 1)$，即 $c = 3k' + 4 (k' \geq 0)$，我们可以先把 c 减少到 4，然后使用两次「减少 2」，轮数为 $\frac{c-4}{3} + 2 = \frac{c+2}{3} = \lceil \frac{c}{3} \rceil$
- 如果 $c = 3k + 2 (k \geq 1)$，我们可以先把 c 减少到 2，然后使用一次「减少 2」，轮数为 $\frac{c-2}{1} + 1 = \frac{c+1}{3} = \lceil \frac{c}{3} \rceil$

综上所述，对于 $c (c \geq 2)$ 个相同难度级别的任务，最少需要操作

$$\left\lceil \frac{c}{3} \right\rceil = \left\lfloor \frac{c+2}{3} \right\rfloor$$

次。

用哈希表统计不同难度任务的个数，按照上式计算轮数，累加轮数即为答案。

Python

```python
class Solution:
    def minimumRounds(self, tasks: List[int]) -> int:
        # 贪心思考即可
        """
        1. 如果 c = 1 无法完成 返回 -1
        2. 如果 c = 3k (k >= 1) 只用减少 3 就能完成 轮数为 k / 3
        3. 如果 c = 3k + 1 (k >= 1) 我们可以先把 c 减少到 4 然后
        使用两次减少 2 轮数为 (c - 4) / 3 + 2 = (c + 2) / 3 = c / 3 上取整
        4. 如果 c = 3k + 2 (k >= 1) 我们可以先把 c 减少到 2 然后
        使用一次减少 2 轮数为 (c - 2) / 3 + 1 = c / 3 上取整
        综上 对于 c (c >= 2) 个相同难度级别的任务 最少需要操作
        c / 3 上取整 = (c + 2) / 3 下取整
        """
        cnt = Counter(tasks)
        if 1 in cnt.values():
            return -1
        return sum((c + 2) // 3 for c in cnt.values())
```

Java

```java
public class Solution {
    public int minimumRounds(int[] tasks) {
        Map<Integer, Integer> cnt = new HashMap<>();
        for (int t : tasks) {
            cnt.merge(t, 1, Integer::sum);
        }
        int ans = 0;
        for (int c : cnt.values()) {
            if (c == 1) {
                return -1;
            }
            ans += (c + 2) / 3;
        }
        return ans;
    }
}
```