28 第十一讲 作业记录（一）

笔记本： 浙江大学《数据结构》
创建时间： 2025/5/27 22:04 **更新时间：** 2025/5/27 22:07
作者： panhengye@163.com
URL： https://pintia.cn/problem-sets/1873565885118418944/exam/problems/typ...

11-散列1 电话聊天狂人

代码记录

```
# 统计每个号码出现的次数 def count_numbers(numbers):     """     Args:
numbers: 一个包含多个号码的列表     Returns:         count: 一个字典，键为号码，
值为该号码出现的次数     """     count = {}     for number in numbers:
count[number] = count.get(number, 0) + 1     return count def
read_call_records():     """     读取通话记录数据     Returns:
all_numbers: 包含所有号码的列表     """     n = int(input())     all_numbers =
[]         for _ in range(n):         line = input().strip()         numbers
= line.split()         caller = numbers[0]         receiver = numbers[1]
    all_numbers.append(caller)         all_numbers.append(receiver)
return all_numbers def find_chat_king(number_count):     """     找出聊天狂人
（通话次数最多且号码最小的人）     Args:         number_count: 号码统计字典
Returns:         tuple: (聊天狂人号码, 通话次数, 并列人数)     """     if not
number_count:         return None, 0, 0         max_count =
max(number_count.values())         # 找出所有通话次数最多的号码     candidates
= [number for number, count in number_count.items() if count == max_count]
  candidates.sort()  # 按字典序排序以找到最小号码         chat_king =
candidates[0]  # 最小的号码     tie_count = len(candidates)  # 并列人数
  return chat_king, max_count, tie_count def main():     # 读取数据
all_numbers = read_call_records()         # 统计每个号码出现的次数
number_count = count_numbers(all_numbers)         # 找出聊天狂人
chat_king, max_count, tie_count = find_chat_king(number_count)         # 输出
结果 - 根据是否有并列情况决定输出格式     if tie_count == 1:         print(f"
{chat_king} {max_count}")     else:         print(f"{chat_king} {max_count}
{tie_count}") if __name__ == "__main__":     main()
```

整体思路

问题分解：

├── 数据收集：读取所有通话记录

├── 频次统计：统计每个号码的出现次数

├── 最值查找：找出最大通话次数

└── 结果筛选：处理并列情况并输出

关键点解释1

```
# 1. 找出最大通话次数 max_count = max(count.values()) # 2. 筛选所有达到最大次数的候选者 candidates = [num for num, cnt in count.items() if cnt == max_count] # 3. 按字典序排序，选择最小号码 candidates.sort() chat_king = candidates[0]
```

关键点解释2

```
# 使用字典进行计数，时间复杂度O(n) count[number] = count.get(number, 0) + 1
```