实验二报告-202408040228-符航康

一、问题分析

要处理的对象（数据）

> - 本实验主要处理的对象是列车车厢的重排问题：有一组车厢按照特定的顺序进入入轨道，需要将这些车厢按正确的顺序**输出**到出轨道中。
> - **输入数据：** 入轨道上的车厢初始排列顺序（如：`5,8,1,7,4,2,9,6,3`）。
> - **输出数据：** 完成车厢重排所需的最小缓冲轨道数量。

要实现的功能

> - 根据车厢的初始顺序，模拟缓冲轨道的分配逻辑（FIFO→队列）；对车厢的顺序进行重排，使得每个车厢按照升序（从1到n）排出。
> - 动态判断车厢是否可以直接进入出轨道，或需要进入缓冲轨道。
> - 使用一个或多个缓冲轨道来存储无法直接进入出轨道的车厢，确保每个车厢的输出顺序正确。
> - 最终计算出至少需要的缓冲轨道数量。

结果

> - 输出一个整数，表示所需的最小缓冲轨道数。

样例求解过程

> #### 1. 初始输入队列：
>
> ```
> 输入队列：3, 6, 9, 2, 4, 7, 1, 8, 5
> 缓冲队列：空
> ```

```
输出队列：空
```

---

2. 按顺序处理车厢：（靠近冒号视为在前）

步骤 1: 处理车厢 3

- `3` 进入缓冲轨道 H1（无法输出）。

```
输入队列：6, 9, 2, 4, 7, 1, 8, 5
缓冲队列：H1: 3
输出队列：空
```

步骤 2: 处理车厢 6

- `6` 进入缓冲轨道 H1（`6 > 3`）。

```
输入队列：9, 2, 4, 7, 1, 8, 5
缓冲队列：H1: 3, 6
输出队列：空
```

步骤 3: 处理车厢 9

- `9` 进入缓冲轨道 H1（`9 > 6`）。

```
输入队列：2, 4, 7, 1, 8, 5
缓冲队列：H1: 3, 6, 9
输出队列：空
```

步骤 4: 处理车厢 2

- `2` 进入缓冲轨道 H2（无法输出）。

```
输入队列：4, 7, 1, 8, 5
缓冲队列：H1: 3, 6, 9
        H2: 2
输出队列：空
```

步骤 5: 处理车厢 4

- `4` 进入缓冲轨道 H2（`4 > 2`）。

```
输入队列：7, 1, 8, 5
缓冲队列：H1: 3, 6, 9
        H2: 2, 4
输出队列：空
```

步骤 6: 处理车厢 7

- `7` 进入缓冲轨道 H2（`7 > 4`）。

```
输入队列：1, 8, 5
缓冲队列：H1: 3, 6, 9
        H2: 2, 4, 7
输出队列：空
```

步骤 7: 处理车厢 1

- `1` 通过缓冲轨道 H3 直接进入出轨道（`current_output=1`）。

```
输入队列：6, 9, 2, 4, 7, 8, 5
缓冲队列：H1: 3, 6, 9
        H2: 2, 4, 7
输出队列：1
```

步骤 8: 处理车厢 2

- `2` 从缓冲轨道 H2 输出（`current_output=2`）。

```
输入队列：6, 9, 4, 7, 8, 5
缓冲队列：H1: 3, 6, 9
        H2: 4, 7
输出队列：1, 2
```

步骤 9: 处理车厢 3

- `3` 从缓冲轨道 H1 输出（`current_output=3`）。

```
输入队列：6, 9, 4, 7, 8, 5
```

```
缓冲队列：H1: 6, 9
        H2: 4, 7
输出队列：1, 2, 3
```

步骤 10: 处理车厢 4

- `4` 从缓冲轨道 H2 输出（`current_output=4`）。

```
输入队列：6, 9, 7, 8, 5
缓冲队列：H1: 6, 9
        H2: 7
输出队列：1, 2, 3, 4
```

步骤 11: 处理车厢 5

- `5` 通过缓冲轨道 H3 直接进入出轨道（`current_output=5`）。

```
输入队列：9, 7, 8
缓冲队列：H1: 6, 9
        H2: 7
输出队列：1, 2, 3, 4, 5
```

步骤 12: 处理车厢 6

- `6` 从缓冲轨道 H1 输出（`current_output=6`）。

```
输入队列：9, 7, 8
缓冲队列：H1: 9
        H2: 7
输出队列：1, 2, 3, 4, 5, 6
```

步骤 13: 处理车厢 7

- `7` 从缓冲轨道 H2 输出（`current_output=7`）。

```
输入队列：9, 8
缓冲队列：H1: 9
        H2: 空
```

```
输出队列：1, 2, 3, 4, 5, 6, 7
```

步骤 14: 处理车厢 8

- `8` 通过缓冲轨道 H3 直接进入出轨道（`current_output=8`）。

```
输入队列：9
缓冲队列：H1: 9
输出队列：1, 2, 3, 4, 5, 6, 7, 8
```

步骤 15: 处理车厢 9

- `9` 从缓冲轨道 H1 输出（`current_output=9`）。

```
输入队列：空
缓冲队列：H1: 空
输出队列：1, 2, 3, 4, 5, 6, 7, 8, 9
```

---

3. 使用的缓冲轨道

- **缓冲轨道 H1**：包含了车厢 3, 6, 9。
- **缓冲轨道 H2**：包含了车厢 2, 4, 7。
- **缓冲轨道 H3**：被用作临时存储，确保车厢可以直接进入出轨道。

最终的最小缓冲轨道数量是 **3**。

二、数据结构和算法设计

抽象数据类型设计

- **队列**：分析逻辑，本题目需要使用FIFO的数据处理方式，用于模拟入轨道和出轨道的车厢顺序（逆序入队），所以采用队列模拟，保证了每个车厢在恰当的时机出轨。
- **缓冲轨道**：使用多个队列来表示缓冲轨道，每个缓冲轨道按FIFO规则都存储着不能立即进入出轨道的车厢。

- **当前应该输出的编号**：记录下一个应输出的车厢编号（从1开始）。

物理数据对象设计

- **输入队列**：一个标准队列（queue），用于存储车厢的输入顺序。
- **缓冲轨道数组**：一个存储多个队列的数组，每个队列表示一个缓冲轨道，存储正在等待的车厢。
- **当前输出车厢编号**：一个整型变量，用于记录当前正在输出的车厢编号。

算法思想的设计

- **递归检查输出**：使用递归函数来处理车厢的输出，通过不断检查输入队列和缓冲轨道，决定是否能输出当前车厢，且优先尝试从输入队列或缓冲队列头部直接输出车厢。
- **缓冲轨道分配**：当某个车厢不能立即进入出轨道时，将其存放到一个合适的缓冲轨道中。

关键功能的算法步骤

输入阶段：将车厢按照给定顺序放入入轨道队列。

输出阶段：利用递归函数 `process_output()` 检查是否可以输出车厢。如果可以，则从输入队列或缓冲轨道中移除车厢。

- 检查输入队列头部是否等于 `current_output`，若是则直接输出。
- 遍历所有缓冲轨道，检查头部是否等于 `current_output`，若是则输出。

缓冲轨道分配：若车厢无法输出，按照车厢顺序，将其放入适当的缓冲轨道。

- 选择尾部车厢编号小于当前车厢的轨道（且满足缓冲轨道中尾部车厢编号是最大的，目的是为了尽可能减少缓冲轨道的使用）。
- 继续进行输出操作，直到所有车厢都成功输出。

统计缓冲轨道数：

- 通过标记已使用的缓冲轨道，统计最终需要的数量。

三、算法性能分析

1. **时间复杂度分析**

- 每次递归调用过程中检查所有缓冲轨道和输入队列，每个车厢最多遍历所有缓冲轨道一次，因此时间复杂度为O(n)。
- 在最坏情况下，每个车厢都需要经过所有的缓冲轨道和入轨道进行检查，总体时间复杂度为$O(n^2)$。

2. **空间复杂度分析**

   - 空间复杂度主要取决于缓冲轨道的数量，最多为n，因此空间复杂度为O(n)。

3. **瓶颈及优化分析**

   - 算法的瓶颈在于每次递归时需要对输入队列和所有缓冲轨道进行检查。
   - 进一步优化可以通过减少缓冲轨道数量来提高效率，如通过哈希维护每一个车厢号所在某一个缓冲轨道的首位，避免每次遍历所有缓冲轨道，而是从是否存在最近的缓冲轨道开始遍历，可以在数据规模较大时提高时间效率。

四、实验总结

1. **核心收获：**通过队列模拟缓冲轨道，理解FIFO规则的应用；递归思想简化了输出逻辑的判断。
2. **改进方向：**可优化缓冲轨道分配策略（如哈希标记），将时间复杂度降低；增加对输入合法性的检查（如车厢编号是否连续）；采用其它效率更高的数据结构（使用更加快速的模拟，而非用现有STL库）。