题目一（20 分）

根据以下说明和需求，设计一个系统 ***S***。

功能需求

1. 系统 ***S*** 的输入数据来源于**顺序队列 *QL* (FIFO)**；输出数据写入另一顺序队列 ***QR***。亦即 ***S*** 依次消费 ***QL*** 内数据，进行加工处理后将生成数据通过 ***QR*** 对外发布；
2. ***QL*** 数据定义为$X_{i,j}$，***i*** 表示该数据来自信息源 ***i***，***j*** 表示信息源 ***i*** 给定数据 ***X*** 的顺序号；
3. ***QR*** 内数据定义为$Y_{i,j}$，***i, j*** 含义同上；
4. ***S*** 对数据的处理逻辑定义为：$Y_{i,j} = F(Y_{i,j-1}, X_{i,j})$，

限制说明

a. 通过 ***QL*** 进入 ***S*** 的数据具有以下特性：
   - 信息源数量（***i*** 的最大取值）约 5000 个；
   - 每一个信息源产生的信息量（***j*** 的最大值范围）不等，约为 100000 ~ 500000；
   - 前置系统（信息源）向 ***QL*** 写入数据的速度较快，比如：这些数据是在 2 小时之内产生并压入 ***QL*** 的；
   - 前置系统（信息源）向 ***QL*** 写入数据的速度不均匀，具有一定的周期性。比如大部分信息源会集中在某一时间段产生大量数据，然后进入平缓期，经过一个时间周期，再次集中的产生大量数据。该行为使得系统 ***S*** 明显感知到数据流“洪峰”。

b. ***S*** 内的处理逻辑 ***F*** 具有一定的复杂性；

c. 通过 ***F*** 的定义可以得知 ***F*** 为递推表达，其必须严格按照 ***X*** 的顺序处理，亦即 ***S*** 未处理完$X_{i,j-1}$前不能处理$X_{i,j}$；

系统设计要求

1. 在保证功能正确前提下，尽可能提升 S 的性能，亦即尽可能降低数据$X_{i,j}$进入 ***QL*** 至$Y_{i,j}$

进入 ***QR*** 的整个时延。可以做理想化设定，认为 ***QL*** 和 ***QR*** 的读写性能极强，是理想的零延迟队列，**将重点放在 S 的优化设计上**。

2. 系统 ***S*** 运行于标准的 X86 服务器之上，GPU 不在考虑范围内。但对服务器内存、CPU 等配置不做具体限定，可根据你的设计给出你认为合理的选型。
3. 在要求 1 中，理想化设定了 ***QL***/***QR*** 为零延迟队列，回归到实际情况，你会以何种方案来设计 ***QL***/***QR***，尽可能提升其读写性能呢？

解答要求

说明展示形式不做规定，伪代码、流程图、系统框图等均可，要能够清晰展现设计思路。

题目二（25 分）

根据以下说明和需求，设计一个系统 ***S***。

<u>功能需求</u>

1. 系统 ***S*** 的输入数据来源于 ***N*** 个 csv 文件集合，记为 $\boldsymbol{F} = \{\boldsymbol{F_1}, \boldsymbol{F_2}, \ldots, \boldsymbol{F_N}\}$ ，每个文件的行数为 ***M***；
2. 输入文件内的每行为一条数据记录，该记录的第一个字段为高精度时间戳，表示该记录产生的时间，以 $\boldsymbol{T_{i,j}}$ 表示文件 $\boldsymbol{F_i}$ 的第 ***j*** 条记录时间戳，设定该时间戳在全部记录上不重复；
3. 记录在文件内是严格按照时间顺序排列的，但文件间顺序不保证，亦即：$\boldsymbol{T_{i,j} < T_{i,j+1}}$ 成立，但 $\boldsymbol{T_{i,j}}$ 与 $\boldsymbol{T_{k,j}}$ 的大小关系是不定的；
4. ***S*** 对外提供以下两个调用接口，供后置系统能够**<u>以时戳递增的顺序</u>**消费文件集合内的全部记录
   - *bool HasNext();*

     是否还有未消费的记录
   - *void Consume(Entry& entry);*

     消费一条记录，Entry 结构表示文本记录经过解析处理后的内存数据
5. ***S*** 的后置系统将采用以下逻辑消费 ***S*** 内的记录

```
while (S.HasNext()) {
    ...
    S.Consume(entry);
    ...
}
```

限制说明

a. 数据文件个数 ***N***和文件行数 ***M***都很大，***N***为 200000，***M***取值范围为[30000, 100000]；

b. ***S***运行环境有内存限制，亦即以下直接暴力的设计是**不允许**的

将 ***F***内的文件数据全部读取进内存，解析完毕后对所有 ***Entry***进行排序，而后 ***S***依次输出排序结果

系统设计要求

1. 先不考虑文件操作和记录解析耗时。在此基础上设计 S 的数据处理算法，给出算法的时间复杂度和空间复杂度。
2. 实际情况下文件操作和记录解析的耗时是不能直接忽略的。同等强度的算法复杂度，合适的文件操作能够有效降低绝对耗时。请给出你的优化方案。

解答要求

说明展示形式不做规定，伪代码、流程图、系统框图等均可，要能够清晰展现设计思路。

题目三（25 分）

根据以下功能需求设计报单流速控制组件 ***OrderController***

<u>功能需求</u>

1. OrderController 有两个设定参数 timeWindow（以 N 代表）和 upperLimit（以 M 代表）
2. 流控模块接收到一个客户端的报单请求（OrderRequest）后调用 OrderController::allow 来判断该请求是否可继续送达后端交易服务。
3. OrderController 的判断标准为：**<u>任意长度</u>**为 N 的时间窗口内（单位：毫秒），**<u>最多</u>**允许 M 个请求被送到后端交易服务（allow 返回 true），超过 M 数量的请求则被拒绝（allow 返回 false）
4. 接口定义如下

```
class OrderController {
public:
    struct OrderRequest {
      unsigned int reqTimestamp;
    };
public:
    OrderController(int timeWindow, int upperLimit);
    bool allow(const OrderRequest& order);
};
```

<u>设计要求</u>

1. 在功能正确的前提下，优化 OrderController::allow 的运行效率，给出时间复杂度

2. 能否做到 OrderController::allow 的时间复杂度与 N 和 M 无关？如果可以，给出方案；如果不可以，证明。

解答要求

源代码实现

题目四（30 分）

根据以下功能需求设计线程安全容器 ***ThreadSafeContainer***

功能需求

1. 支持 key-value 模式存储（为简化设计，约定 key: int, value: int）
2. 增删改查操作均为**线程安全**接口，亦即调用者可直接在多线程环境下使用 ThreadSafeContainer，不需要额外做任何线程安全处理。
3. 接口定义如下

```
class ThreadSafeContainer {
public:
    void Put(int key, int value); //有则更新，无则新增
    void Delete(int key);
    bool Get(int key, int& value); //key 存在返回 true，并通过 value 传出值；key 不存在则返回 false
};
```

设计要求

1. 在保证线程安全的前提下，尽可能提高 ThreadSafeContainer 的并发性能。

解答要求

说明展示形式不做规定，伪代码、流程图、系统框图等均可，要能够清晰展现设计思路。