面向对象电梯系列第二次指导书

摘要

本次作业，需要完成的任务为**单部多线程可捎带调度（ALS）电梯**的模拟

问题

电梯系统说明

本次作业需要模拟一个多线程实时电梯系统，从标准输入中输入请求信息，程序进行接收和处理，模拟电梯运行，将必要的运行信息通过输出接口进行输出。

本次作业电梯系统具有的功能为：上下行，开关门，每运行一层的时间为固定值，开关门的时间为**不同的固定值。**

电梯系统可以采用任意的调度策略，即上行还是下行，是否在某层开关门，都可自定义，只要保证在**系统限制时间内**将所有的乘客送至目的地即可。

电梯系统在某一层开关门时间内可以上下乘客，**开关门的边界时间都可以上下乘客。**

电梯运行基本概念

- 电梯系统时间：程序开始运行的时间
- 电梯楼层：-3层到-1层，1层到16层，共19层
- 电梯初始位置：1层
- 电梯数量：1部
- 电梯上升或下降一层的时间：0.4s
- 电梯开关门的时间：开门0.2s，关门0.2s，共计0.4s，到达楼层后方可立刻开门，关门完毕后方可立刻出发。

电梯请求说明

本次作业的电梯，为一种比较特殊的电梯，学名叫做**目的选层电梯**（可以百度一下，确实是存在的，且已经投入了主流市场）。

大概意思是，**在电梯的每层入口，都有一个输入装置，让每个乘客输入自己的目的楼层。**电梯基于这样的一个目的地选择系统进行调度，将乘客运送到指定的目标楼层。

所以，一个电梯请求包含这个人的**出发楼层和目的楼层**，以及这个人的id（保证人员id唯一），请求内容将作为一个整体送入电梯系统，在整个运行过程中请求内容不会发生改变。

ALS(可捎带电梯)规则介绍

- 可捎带电梯调度器将会新增**主请求**和**被捎带请求**两个概念
- 主请求选择规则：
  - 如果电梯中没有乘客，将请求队列中到达时间最早的请求作为主请求
  - 如果电梯中有乘客，将其中到达时间最早的乘客请求作为主请求
- 被捎带请求选择规则：

    - 电梯的主请求存在，即主请求到该请求进入电梯时尚未完成
    - 该请求到达请求队列的时间**小于等于**电梯到达该请求出发楼层关门的截止时间
    - 电梯的运行方向和该请求的目标方向一致。即电梯主请求的目标楼层和被捎带请求的目标楼层，两者在当前楼层的同一侧。
- 其他：
    - 标准ALS电梯不会连续开关门。即开门关门一次之后，如果请求队列中还有请求，不能立即再执行开关门操作，会先执行请求。
- 以上策略只是建议，如果有觉得更好的（更好写或者性能更高），欢迎自行探索。

输入输出

输入

- 输入一律在标准输入中进行。
- 本次电梯作业使用以人为单位的请求模式，即包含一个人的id，出发楼层，目标楼层，出发楼层和目标楼层不能相同。
- 输入标准格式为`id-FROM-x-TO-y`，其中一条指令一行，id是人的唯一标识，x，y是属于楼层范围的一个整数，分别代表出发楼层和目标楼层。
- 课程组会下发对应的输入解析（ElevatorInput）程序，自动解析标准输入之中的内容并转换为`PersonRequest`对象返回给使用者。如果输入中有格式错误，或者存在id重复，或是出发楼层和目标楼层相同，解析程序只会读取到正确的请求，错误的将跳过并在标准异常中输出错误信息（不影响程序本身运行，也不会引发RUNTIME_ERROR）。具体的接口使用信息请参考**电梯第二次作业输入接口文档。**
- 本次的输入为实时交互，评测机可以做到在某个时间点投放一定量的输入。~~也就是传说中的，强制在线。~~

输出

- 输出一律使用课程组提供的接口进行输出，一条输出内容占一行，接口中会自动**附加输出时的电梯系统时间。**此外，任何其他信息**绝对不要**输出在标准输出中~~，更不要试图伪造假时间假输出蒙混过关~~，否则将会影响评测，且后果自负。
- 输出信息包括两部分，电梯的状态和人的状态
    - 电梯的状态，分为**三种**，OPEN(开门)，CLOSE(关门)，ARRIVE（到达）
        - 开门格式：OPEN-楼层（在开门刚开始输出）
        - 关门格式：CLOSE-楼层（在关门刚结束输出）
        - 到达格式：ARRIVE-楼层（在刚刚到达该楼层时输出）
    - 人的状态，分为两种，IN(进入电梯)，OUT(离开电梯)
        - 进入电梯格式：IN-id-楼层，这里的id是这个人自己的id
        - 离开电梯格式：OUT-id-楼层
- 具体输出时，使用者需要将格式字符串作为对应的参数传入官方输出接口中，接口内部会自动加上时间戳输出到标准输出。简单来说，和println使用方法大致类似，具体请参考**输出接口文档。**

样例

#	输入	输出	说明
1	[0.0]1-FROM-1-TO-2	[0.0100]OPEN-1 [0.0130]IN-1-1 [0.4110]CLOSE-1 [0.8120]ARRIVE-2 [0.8130]OPEN-2 [0.8130]OUT-1-2 [1.2130]CLOSE-2	显然
2	[0.0]1-FROM-1-TO-2 [0.0]2-FROM-1-TO-2	[0.0120]OPEN-1 [0.0160]IN-1-1 [0.0170]IN-2-1 [0.4120]CLOSE-1 [0.8120]ARRIVE-2 [0.8120]OPEN-2 [0.8120]OUT-1-2 [0.8130]OUT-2-2 [1.2120]CLOSE-2	电梯可以采用ALS调度策略，乘客1成为主请求，乘客2满足捎带条件故捎带

#	输入	输出	说明
3	[1.5]1-FROM-1-TO-2	[1.3870]OPEN-1 [1.3910]IN-1-1 [1.7930]CLOSE-1 [2.1940]ARRIVE-2 [2.1940]OPEN-2 [2.1940]OUT-1-2 [2.5940]CLOSE-2	输入的起始时间可以不从0.0开始 **（此处存在时间计测同步性问题，在样例说明部分会有详细的说明）**
4	[2.3]1-FROM-2-TO-5 [3.6]2-FROM-3-TO-4	[2.7060]ARRIVE-2 [2.7070]OPEN-2 [2.7100]IN-1-2 [3.1080]CLOSE-2 [3.5120]ARRIVE-3 [3.9120]ARRIVE-4 [4.3130]ARRIVE-5 [4.3140]OPEN-5 [4.3140]OUT-1-5 [4.7140]CLOSE-5 [5.2130]ARRIVE-4 [	起始楼层可以不是1层 **（此处存在时间计测同步性问题，在样例说明部分会有详细的说明）** 不满足捎带时间条件，故不捎带

#	输入	输出	说明
		5.6130]ARRIVE-3 [5.6140]OPEN-3 [5.6140]IN-2-3 [6.0150]CLOSE-3 [6.4160]ARRIVE-4 [6.4160]OPEN-4 [6.4180]OUT-2-4 [6.8170]CLOSE-4	

说明：

- 为了方便说明，指导书上提供的输入为这样的格式：`[x.xxxx]yyyyyyyy`。
- 意思是在`x.xxxx`这个时间点（相对于程序运行开始的时间，单位秒），输入`yyyyyyyy`这样的一行数据。
- 关于上面说到的时间计测同步性问题，在这里解释一下：
  - 直观地说，可能会观察到输入指令时间晚于做出反应的时间这一现象。
  - 这个实际上不是程序的错误，而是由于评测机投放数据的系统、和程序输出接口，这两边的时间可能存在不同步导致的。
  - **这一问题是由于多线程测试不可避免的波动性导致的计时同步性误差，属于正常现象，不会被认定为错误。**（~~当然，也不会出现故意提早的情况，因为程序本身就是实时交互，不可能做得到预知未来~~）

关于判定

数据基本限制

- 可输入电梯系统的指令数上限：30条。
- 电梯系统总执行时间上限为T_{max}，这里的总执行时间是由课程组下发的`datacheck`程序确定，具体是指电梯执行完最后一条指令关门时的时间。
- 在互测时，输入数据的基准时间（T_{base}）不得超过170s。
- 类似的，强测数据也会保证正常的程序不会超过时间上限，也**会避免使用对边界情况较为敏感的数据。**
- **对于中测和强测数据，运行真实时间（real time），将会严格限制为略大于T_{max}的值，即超过这个值会被直接判定为REAL_TIME_LIMIT_EXCEED。**~~所以，请不要试图用超长的sleep来回避线程安全停止的问题。~~
- 对于互测数据，真实时间的限制依然为210s，但是也会做T_{max}（或者200s）的检查。超过这一限制的也会被判定为不通过。

正确性判断

电梯系统运行的正确性判断只根据以下三点：

- 电梯运行逻辑合理，即
  - 电梯在两个楼层之间的运行时间**大于等于**理论运行时间
  - 电梯开关门的时间**大于等于**理论开关门时间
  - 电梯**必须停下才能开门**，开门到关门期间，禁止移动，电梯**关门完毕后才可以继续移动**
  - 电梯在开门到关门的时间闭区间内，才可以进行上下人的行为
  - 而且，上下人行为的输出顺序必须介于开门和关门之间
    - 即对于这样的例子，即便输出时间上符合要求
      - `OUT-1-1`
      - `OPEN-1`
      - `CLOSE-1`
    - 也是属于错误的，`IN`、`OUT`**必须在一对相邻的**`OPEN`、`CLOSE`**之间**
  - 关于ARRIVE
    - 当且仅当电梯到达每个楼层（即楼层发生改变）时，必须立刻输出且仅输出一次ARRIVE信息。
    - 忽略其他输出的话，每两个相邻ARRIVE之间，必须刚好只差一层楼。即，**禁止跳层**，即便时间是对的。
    - 在电梯的开关门之间，不允许ARRIVE。
    - **当且仅当电梯ARRIVE到了某一层，才可以在这个楼层上进行开门、上下人、关门的动作。**
- 人员运行逻辑合理，假设请求为`ID-FROM-X-TO-Y`，则
  - 发出指令时，人员处于电梯外的X层
  - 人员进入电梯后，人员将处于电梯内，且必须是**电梯外的当前停靠层的人员才可以进入电梯**
  - 人员离开电梯后，人员将处于电梯外的当前停靠层，且必须是**电梯内的人员才可以离开电梯**
  - 人员不在电梯内的时候，不会有自行移动的行为。即，可以理解为人员会一直待在原地
- 在电梯运行结束时，所有的乘客都已经到达各自的目标楼层电梯外，且电梯都已经关上了门。
- 故如果有程序运行结果正确但总时间超过了电梯系统总执行时间上限，或者未达到性能下限要求（T_{max}），那么会视为错误。
- **满足上述四条要求的一切调度结果都是正确的**

性能要求

- 我们以电梯系统的总运行时间作为性能指标。
- 本次电梯系统的参考调度策略为ALS(可捎带电梯)，我们提供的`datacheck`模拟器就是使用这一策略，`datacheck`模拟出的运行时间作为基准时间T_{base}。
- 定义一个值，$T_{max} = \max\left(T_{base} + 3, 1.05 \cdot T_{base}\right)$。
- 对于本次所有的数据（中测、强测），运行结果正确但总时间超过T_{max}的程序将也会视作未通过该测试点。
- 在本次的互测中，运行结果正确但是总时间超过200s的程序也将视为未通过测试点。**互测的重点还是放在正确性本身上，同时也是为了避免有些极端边界数据造成的WRONG ANSWER。**
- 关于这个T_{base}的求解方式，即为根据输入来进行ALS策略的模拟。不过值得注意的是，**我们会在模拟的时候随机加入一些扰动**（比如电梯运行一层的时间随机加长一定量，请求接收的时间也随机前后波动一定量等。**这些扰动在实际运行时也都是多少会存在的，不可能做到绝对意义上的精确，这也正是加入扰动的原因**），然后进行5000次运行，取一个最大值，向上取整后作为基准时间T_{base}。

性能分的计算

- 在强测时，则会根据总运行时间（即电梯调度完成全部请求的时间，而非程序运行时间，设为$time$）来计算相应的性能分。
- 设全体通过该数据点的同学的运行时间的平均值为$mean$，标准差为std。
- 在区间内部的情况下，设$x = -\frac{time-mean}{std}$，则

$$r\left(x\right) = 100\% \cdot \begin{cases} 1 & x \geq 1 \\ 0.17361111x^3 - 0.05952381x^2 + 0.73710317x + 0.15119048 & -0.2 < x < 1 \\ 0 & x \leq -0.2 \end{cases}$$

- 简而言之，大致是如下的分布

x	$r\left(x\right)$
-0.2	0%
0.0	15%
0.2	30%
0.4	45%
0.6	60%
0.8	80%
1.0	100%

- 本次强测性能分占强测总分的20%

其他评测时要求

- 从本次作业开始，严格限制CPU时间，**即程序运行CPU时间限制为10s，超出即视为该测试点未通过。**目的是限制暴力轮询的行为，建议**尽量使用notify、wait来解决问题。**

互测输入要求

互测样例输入数据需要额外附加时间项，评测系统将使用请求发射器在对应的时间将请求送入标准输入。

即互测时的输入格式为`[time]id-FROM-x-TO-y`，其中time是一个保留一位小数的浮点数，比如`0.0`,`1.5`,`2.2`等

互测样例输入基本限制总结

- 输入严格符合标准格式，不存在格式错误
- 整个输入序列中不存在重复的人员id，且人员id为int范围内的非负整数
- 整个输入序列中不存在出发楼层和目的楼层相同的请求指令
- 整个输入序列中的时间必须单调递增（不一定严格递增，但是禁止出现递减的情况）
- 满足指令数上限要求
- 互测时为防止特殊边界情况，限制样例的`datacheck`基准时间（T_{base}）必须小于170s。
- 课程组下发的`datacheck`程序将会检测是否符合如上6条基本限制，如果不符合的样例将会禁止进入互测。

提示

- 如果还有人不知道标准输入、标准输出是啥的话，那在这里解释一下
  - 标准输入，直观来说就是屏幕输入
  - 标准输出，直观来说就是屏幕输出
  - 标准异常，直观来说就是报错的时候那堆红字
  - 想更加详细的了解的话，请去百度
- 关于ALS调度策略
  - 本次作业的目标是希望每位同学都至少可以完成一个ALS电梯的设计和运行。
  - 如果有自己觉得更好（更好写、或效率更高）的调度方式，欢迎自行探索和研究，可以获得更大的性能分收益。
- 关于这次的架构，可以采用这样的模式：
  - 主线程进行输入的管理，使用ElevatorInput，负责接收请求并存入队列
  - 构建一个调度器（本次的调度器可以和队列是一体的）
    - 用于管理请求
    - 和电梯进行交互并指派任务给电梯
    - 且**需要保证调度器是线程安全的**
  - 构建一个电梯线程，负责和调度器进行交互（比如每到一层进行一个交互）接收任务，并按照一定的局部策略进行执行。
  - 设计的重点依然在于**最大限度降低耦合，每个对象只应该管自己该管的事。**
  - 以上只是建议，如果有自己觉得更好的设计，欢迎自行探索和研究。
- 当然，对于追求更强调度性能的同学们，可以**百度搜索“电梯调度常见算法”，自行寻找并实现更优的调度算法。**
- 关于接口的一些使用，可以在idea里面，按Ctrl+Q。将可以看到类和方法的使用格式以及注释。
- 友情提醒：**请不要为了那点性能分而在架构设计上大开倒车。**所有人的最终目的都是学到知识而不是紧扣分数。而且，冒着出现大量BUG的风险，拿一个工程性极差的架构来赌博式地追求性能分，很可能并不划算，一旦正确性错了就会血本无归。**真正靠谱的架构，一定是可以做到兼顾正确性和性能优化的。**长远来看，好好优化架构才是拿高分唯一正确的思路，而不是剑走偏锋甚至于本末倒置。