模拟OS作业调度

题目描述

请模拟实现操作系统的作业调度。给定的计算机具有一个CPU，该CPU有*n*个核（$n >= 1$）。CPU可以采用的作业调度算法有如下两种：（调度算法详解见附录）

1. 优先级调度算法（PSA）　　[非抢占式]
2. 最高响应比优先算法（HRRF）　[非抢占式]

现有*m*个任务在不同的时间进入该操作系统，请使用**面向对象**模拟操作系统对这些任务进行调度执行。

每个任务（task）具有如下属性

1. ID（大于等于0的整数）
2. 名称
3. 类型（是一个整数）
4. 优先级（是一个大于等于0的整数，数值越小代表优先级越大）
5. 抵达时间（整数类型，从0开始，单位为分钟）
6. 执行所需时长（整数类型，单位为分钟）

执行顺序

CPU对任务的分配是按照核的ID递增顺序查找的，核ID从0开始。比如有5核，则ID分别为 {0, 1, 2, 3, 4}，CPU对空闲核的查找从0开始。同样，对于同等层次的作业（在PSA下具有相同的优先级，在HRRF在具有相同的响应比）作业ID小的优先分配；当有多个空闲内核时，内核ID小的优先使用。

任务执行

该计算机接收的任务类型一共只有以下三种类型。每种类型的任务都有三个步骤，分为**load**，**execute**，**finish**三个步骤。其中**load**的职责是将所需资源加载进执行空间，**execute**为具体的执行步骤（不同任务类型的步骤不一样，步骤打印内容参考下面的表格），**finish**为将资源释放。

不同任务步骤

> LoadDatabase (type : 0)

行为顺序	行为名称	打印内容
1	加载驱动	Load driver
2	建立数据库连接	Establish a database connection
3	创建Statement对象	Create a Statement object
4	执行SQL语句	Execute SQL
5	关闭数据库	Close the database

> ETL (type : 1)

行为顺序	行为名称	打印内容
1	抽取	Read data from a file
2	转换	Data transform
3	加载	Load data into the database

> DataAnalyze (type : 2)

行为顺序	行为名称	打印内容
1	寻找中心点	Finding the center point
2	数据分类	Data Classification
3	数据比对	Data comparison
4	数据展示	Data Display

输入描述

```
2 PSA 4
1 task1 2 1 0 10
2 task2 0 2 0 8
3 task3 2 1 5 15
4 task4 1 3 5 1
```

> 每个输入的第一行分别代表
>
> CPU内核数量（*n*），操作系统的作业调度算法，任务数量（*m*）
>
> 接下来的 *m* 行分别描述了每个作业的内容
>
> 作业内容解读
>
> ID，名称，类型，优先级（每种调度算法都会给定这个字段），抵达时间，执行所需时间
>
> 每行输入的每一项分别用空格隔开（即每一项内部都不含空格）

输出描述

```
task 1 load: time 0 coreId 0 name task1 type DataAnalyze priority 0
task 1 execute: time 0 coreId 0
task 1 --- Finding the center point
task 1 --- Data Classification
task 1 --- Data comparison
task 1 --- Data Display
task 1 finish: time 10 coreId 0
```

> 每个作业的输出有多行，但是不一定是连续的多行。其中**前2行以及紧接着的执行步骤一定是紧邻的输出**。（比如上述示例的前6行是连在一起的）
>
> 第一行代表当前作业的装载信息
>
> ```
> task {taskId} load: time {currentTime} coreId {coreId} name {taskName}
> type {taskType} priority {taskPriority}
> ```

> *注：{taskType}为上述对应的类型文字，即“LoadDatabase”，“ETL”，”DataAnalyze“三种
>
> 第二行代表当前作业的开始执行状况

```
task {taskId} execute: time {currentTime} coreId {coreId}
```

> 紧接着的 *k* 行代表当前作业的执行步骤
>
> 如上述样例中任务类型给定为2，也就是DataAnalyze，对应的执行步骤有4步

```
task {taskId} --- {stepDescription}
task {taskId} --- {stepDescription}
task {taskId} --- {stepDescription}
task {taskId} --- {stepDescription}
```

> 倒数第一行代表作业结束状态

```
task {taskId} finish: time {currentTime} coreId {coreId}
```

> {}中是具体的信息内容，注意每个标点符号之后都有一个空格。（”{}“不用输出）

每个作业的开始部分语句与结束部分语句不一定连在一起，中间可能会有其他作业的相关打印语句！但是要注意顺序不能错乱！

> 输出规则

两点输出规则：① 任务开始按照 taskId ② 任务结束按照 coreId

① 每个task的开始部分的输出是根据作业的开始时刻先后输出的，同一时刻的所有task按照task的ID进行输出，ID小的优先输出。（任务开始按照 taskId ）
② 只有当task所需要的时间达到时，才会输出task的结束部分语句。同一时刻的结束语句则是按照core的ID小的优先输出规则。（任务结束按照 coreId ）

当同一时间有一个核的任务结束与另一个核任务的开始，则先打印结束信息的语句

> **请勿在程序中使用睡眠操作来进行时间控制，否则将会执行超时。给定的时间只是为了模拟先后顺序。**

> 示例1

输入

```
1 PSA 1
1 task1 1 0 0 4
```

输出

```
task 1 load: time 0 coreId 0 name task1 type ETL priority 0
task 1 execute: time 0 coreId 0
task 1 --- Read data from a file
task 1 --- Data transform
task 1 --- Load data into the database
task 1 finish: time 4 coreId 0
```

> 示例2

输入

```
2 HRRF 3
1 task1 2 1 0 2
2 task2 0 1 1 5
3 task3 0 0 1 4
```

输出

```
task 1 load: time 0 coreId 0 name task1 type DataAnalyze priority 1
task 1 execute: time 0 coreId 0
task 1 --- Finding the center point
task 1 --- Data Classification
task 1 --- Data comparison
task 1 --- Data Display
task 2 load: time 1 coreId 1 name task2 type LoadDatabase priority 1
task 2 execute: time 1 coreId 1
task 2 --- Load driver
task 2 --- Establish a database connection
task 2 --- Create a Statement object
task 2 --- Execute SQL
task 2 --- Close the database
task 1 finish: time 2 coreId 0
task 3 load: time 2 coreId 0 name task3 type LoadDatabase priority 0
task 3 execute: time 2 coreId 0
task 3 --- Load driver
task 3 --- Establish a database connection
task 3 --- Create a Statement object
task 3 --- Execute SQL
task 3 --- Close the database
task 3 finish: time 6 coreId 0
task 2 finish: time 6 coreId 1
```

附录

算法描述

> PSA

总是选择就绪队列中优先级最高者投入运行，在本次作业中采用非剥夺式策略。也就是说，即使有比当前执行任务优先级高的出现，也要等到当前任务执行才会再次从就绪队列中选取优先级高的执行。

> HRRF

响应比 = (作业等待时间 + 作业处理时间) / 作业处理时间

测试用例类型

1. 单核单任务
2. 多核单任务
3. 单核多任务
4. 多核多任务