需求分析文档

一． 基本功能

实现一个打车系统，乘客在任意时间发出请求，如果在 3s 的时间窗口以内有出租车进入以呼叫点为中心的 4x4 方格区域，则该出租车抢单。最后 3s 时间窗口结束时，会在这些抢单的出租车中选择出最佳选项。

二．模型分析

（1）最佳相应出租车的选择方法：如果有车响应，系统为乘客从当前时间窗口中抢单的出租车中自动进行选择，在抢单时间窗口关闭时刻选择处于等待服务中信用度最高的出租车；如果有多辆信用度相同的出租车，则选择当前距离用户请求出发地最近的；如果仍有多辆满足条件的出租车，则从中随机选择一辆。

（2）出租车的运动模型：在等待服务状态时，出租车如果遇到道路分支，可随机选择一条分支边行走。 在接单状态，需以最短路径（本次作业为最短距离，有多条路径可以任选一条）前往乘客打车出发地。在运载乘客到目的地过程中，出租车必须按照最短路径（有多条最短路径任选一条）行走。

（3）出租车的状态转换规则：出租车处于等待服务状态持续 20s 后，停止运行 1s，然后再次运行立即进入等待服务状态。 出租车在抢到用户请求后，进入接单状态，需以最短路径（本次作业为最短距离，有多条路径可以任选一条）前往乘客打车出发地。一旦到达用户等待位置，停止运行 1s，然后再次运行立即进入服务状态。出租车完成当前服务（到达用户的目的地）后，停止运行 1s，然后再次运行立即进入等待服务状态。任何情况下，只要是停止运行，都处于停止运行状态。

（4）出租车的信用改变规则：出租车有信用积累，初始所有车信用为 0，每

抢单一次会使其信用度加 1，每成功服务顾客一次会使其信用度加 3

（5）地图模型：

a.城市地图通过 ASCII 编码的文本文件输入。文件内容为 80 行字符串，每行有 80 个数字字符（字符之间允许出现空格或制表符），每个数字字符为 0 到 3 之间的整数，表示一个 80×80 的邻接矩阵..80×80。输入文件中除数字 0,1,2,3，空格，制表符和回车换行外，出现任何其他字符都可判定为无效输入。

b.地图必须为连接图。

c.为了计算两个点之间的最短距离需要用到邻接矩阵。

（6）用户请求模型：程序可通过控制台来获得乘客请求，乘客请求格式为[CR,src,dst]，其中 CR 为标识符，src 和 dst 均为(i,j)形式的坐标位置,表示乘客请求的发出地和目的地。任何超出地图范围的坐标都视为无效请求而被直接忽略，不影响对其他有效请求的处理。乘客请求的产生时间自动从系统获得。系统的基本时间单位 100ms。需要解决的问题有：

a.对于用户请求有效性的判断

b.对于重复请求的判断

三．设计思路分析

（1）地图类 Map

本次作业的汽车行驶路径和用户请求都需要用到地图，所以需要一个单独的 Map 类来管理和解析这个地图。地图信息从文件读取，并不一定能保证其正确性。所以 Map 类需要几个方法：

a.从文件中读取地图并判断前输入格式是否正确

b.分析读取的地图是否为连通图

初次之外，出租车在行驶过程中需要判断计算最短路径。计算最短路径的方法要基于地图是和地图相关的，而且 Taxi 类和 Scheduler 类都需要调用该方法，所以应该把该方法放在 Map 类里，并且把其设为静态方法。由于计算最短路径需要使用 Dijkstra 算法，所以 Map 类里还应该管理一个邻接矩阵。由上述分析，Map 类需要新增几个方法：

c. 初始化地图的邻接矩阵

d.计算地图上两个点的最短距离

e.返回地图上两个点之间的最短路径

（2）用户类 Customer

由于程序需要不断的从控制台读取请求，并且和其他任务之间进行切换，所以 Customer 类需要被设计为一个线程。所需要的方法有：

a. 从控制台读取请求

b. 判断请求是否有效

Customer 类只负责接收请求，但其本身并不负责处理请求，所以 Customr 类需要将请求传递给请求队列。

（3）调度器 Scheduler

一个请求传进来需要判断应该由哪个出租车来最终相应这个请求，每一个请求有 3s 的相应时间窗口。现在考虑的问题是一个请求一个调度器还是所有请求共享一个调度器。考虑这种情况，所有请求共享一个调度器，如果同一时刻输入了上百条请求，那么调度器类的工作是非常繁重的，而且对于 3s 的时间窗口也是很难把控的。所以比较好的设计是一个调度器对应一个请求。

由于调度器只需要工作 3s 来判读最后的出租车选项，所以可以将其设为 1

个线程，在这 3s 的时间窗口内不断的扫描有无新进入 4x4 区域的出租车，并记录其相关信息。3 秒结束后，run 函数执行完，线程自动终止。

由于调度类要访问和设置出租车的信息，所以 Scheduler 和 ControlCenter 应该共享出租车对象。

（4）控制中心 ControlCenter

控制中心需要完成对请求的分配处理，并且通知出租车完成相应的请求。所以控制中心应该管理着一个出租车队列和用户请求队列，并且完成它们之间的关联和相应。

为了访问出租车的地址，需要有一个对象类记录当前某个位置所有出租车的编号，也即记录出租车的位置信息。所有出租车应当共享这个对象，所以应该让 ControlCenter 来管理这个对象，作为 ControlCenter 和 Taxi 的共享资源。综上需要设计的方法有：

a.初始化出租车队列

b.初始化记录出租车物理位置的地图

刚才说到 Customer 类把请求传递给请求队列，但是还存在一种情况就是重复请求，为了略去重复请求也要有一个方法，综上控制中心应该有方法：

c. 到请求队列中寻找有无和当前请求相同的请求

d. 将请求加入到请求队列（这个方法由 Customer 类来调用）

为了访问出租车的地址，需要有一个对象类记录当前某个位置所有出租车的编号，也即记录出租车的位置信息。所有出租车应当共享这个对象，所以应该让 ControlCenter 来管理这个对象，作为 ControlCenter 和 Taxi 的共享资源。

（5）出租车类 Taxi

根据出租车的功能，很容易想到它有如下几个方法：

a. 出租车闲逛，即走 20s，睡 1s

b. 出租车去某个地点接乘客上车，并且睡 1s

c. 出租车载乘客去某个地点，并且睡 1s

d. 出租车随机移动一步

e. 出租车更新自身信息若坐标和状态

此外出租车还和其他类有一定的交互，如 Scheduler 类，Scheduler 类判断某辆出租车进入抢单区域后应该将车的信用加 1，同时 Scheduler 需要访问出租车信用信息，所以需要给 Scheduler 类设计两个接口：

f. 增加出租车的信用

g. 访问出租车的信用

当一个请求执行完后，出租车的状态和信用信息发生改变，所以还需要两个方法：

h.请求执行完后更新出租车的状态和信用

i.将请求信息传给 Result 类，使其将信息输出

（6）Result 类

该类的任务很明确，就是将请求的响应信息输出。

请求的响应信息有两类，一是可执行且执行完的请求，二是无出租车响应的请求。执行完的请求是和 Taxi 类发生交互，无出租车相应的请求是和 Scheduler 类发生交互。所以该类可以设计两个静态方法：

a. 输出来自 Scheduler 的无出租车相应的请求信息

b. 输出来自 Taxi 类的已执行完的出租车的请求信息

（7）RequestFile 类

该类就是记录请求的相关信息，通过类名里的 File 就可以得知，该类记录的信息较多，并不单单是请求本身这么简单。因为用户需求指出：针对每个乘客请求，需要记录的数据包括：请求发出时，处于以请求 src 为中心的 4×4 区域中的所有出租车状态、信用信息；在抢单时间窗内所有抢单的出租车；系统选择响应相应请求的出租车；出租车响应相应请求过程中的实际行驶路径。

一个可执行请求从发出，到执行完需要经过 3 个类。

Step1: Customer 类读入请求，并且将其传入请求队列。在这一步记录的有请求的发出时间，和起始和终止地点坐标。

Step2:由请求建立相应的调度器类，并完成 3s 时间窗口内的调度分析。在这一步记录的有：

a. 请求发出时，4x4 区域的出租车信息

b. 抢单窗口内所有出租车编号

c. 响应请求的出租车编号

Step3: 调度器完成任务分析后将请求分配给具体的出租车，出租车根据始末地点坐标规划路径。在这一步记录的有响应请求过程中的出租车行驶路径。