一、主要数据结构：

1.后台可见的数据：

送餐员（driverID，坐标，工作时间，状态，携带的订单列表，目前正在处理的订单，其他分配的订单）

状态分为：任务进行中／空闲／不可用。

携带订单指送餐员已经取到菜的订单。

其他分配的订单指upcoming task，实际上与核心算法无关。

订单（orderID，商家ID，取餐坐标，客户坐标，菜品列表，状态，分配的送餐员，预计取餐时间）

状态分为：尚未开始，取餐中，已取餐，已完成-成功，已完成-失败。

尚未开始和取餐中的区别在于，取餐中的订单具有分配的送餐员，并且此送餐员正在执行取餐任务。

当状态为已完成-失败时，可通过 订单->分配的送餐员->携带的订单 判断订单是否在送餐员手中（在重新配送时需要此信息）。

取餐坐标通常设为商家的坐标，在发生意外事故时则设为事故地点坐标

地图（某些坐标之间的距离、时间、建议路线）：需要跟地图模块进行交互。

2. 核心算法输入相关的数据结构：

送餐员（driverID，预计可用的时间，可用时的位置坐标，工作时间）

任务（taskID，deadline，readyTime，地点坐标，指定送餐员列表，完成收益，延迟代价）

readyTime表示如果提前到达则需等待至readyTime。

任务依赖关系（二元组（taskID1，taskID2））表明task2必须在task1完成之后才可由完成task1的送餐员进行。

地图（任务涉及坐标之间的旅行时间）

3. 后台根据其可见数据，在每次调用核心算法时准备以下数据：

3.1 检索所有状态为任务进行中或者空闲的送餐员， 建立一个可用送餐员的列表，表中的每一项为：送餐员（driverID，预计可用的时间，可用时的位置坐标，工作时间）

driverID和工作时间 直接copy自后台可见数据。

如果送餐员的状态为空闲，则预计可用时间设为当前时间，可用时的位置坐标设为目前的坐标；

如果送餐员的状态为任务进行中，则考察其正在进行的订单状态（应为以下两种情况之一）：

如果是取餐中，则将预计可用时间设为max {预期到达餐馆的时间，订单的预计取餐时间}，可用时的位置坐标设为餐馆的坐标；

如果是已取餐，则将预计可用时间设为预期到达客户的时间，可用时的位置坐标设为客户坐标。

3.2 对于每一个状态为尚未开始的订单，生成如下两个任务

取餐（readyTime＝订单的预计取餐时间，deadline＝readyTime＋一个预设值（可以为0），地点坐标＝取餐坐标，指定送餐员＝任意，完成收益，延迟代价 待定）

送餐（readyTime＝当前时间，deadline＝订单的预计取餐时间＋预设值，地点坐标＝客户坐标，指定送餐员＝任意（通过任务依赖关系保证取餐送餐为同一人））

并设置送餐任务依赖于取餐任务。

3.3 对于每一个状态为取餐中的订单，生成如下任务

送餐（readyTime＝当前时间，deadline＝订单的预计取餐时间＋预设值，地点坐标＝客户坐标，指定送餐员＝订单的分配送餐员）

//3.2和3.3中涉及的预计取餐时间需要考虑预计到达餐馆时间晚于订单的取餐时间

3.4 从地图信息中提取上述任务中涉及的坐标之间的旅行时间

4. 核心算法的返回：每个送餐员一个顺序任务列表。满足任务依赖关系和指定送餐员的限制。首先保证按时完成尽可能多的任务，其次让分配到任务的司机数目尽可能少。所有输入的任务都必须分配。

p.s. 算法应该考虑估计时间的不确定性，对于时间的变动应该健壮／稳定

p.p.s. 测试的时候可以写一个仿真模块。

二、功能实现：

1. 后台面向送餐员的接口：

//根据任务分配情况，合并显示取餐任务

取到餐（订单ID列表）：将相应订单状态置为已取餐，分配的送餐员置为此人，送餐员状态置为空闲，携带的订单中添加此订单。

送到餐（订单ID）：将相应的订单状态置为已完成-成功，送餐员状态置为空闲，携带订单中删除此订单。

取餐失败之未准备好（订单ID）：将相应的订单状态置为已完成-失败，送餐员状态置为空闲。//当送餐员决定不再等待时按这个

送餐失败之客户不在（订单ID）：将相应的订单状态置为已完成-失败，送餐员状态置为空闲，携带订单中保留此订单。

发生意外事故（事故地点or 交接地点坐标）：将送餐员的状态置为不可用，对其携带的每一个订单，将其状态设为未分配，取餐地点设为交接地点坐标，readyTime设为当前时间，deadline看心情设定吧☺

2. 后台面向商家的接口：

新建订单（取餐时间，客户坐标，所订菜单）：生成新订单并设置相应属性

重新配送（订单ID，取餐时间，客户坐标）：相应订单状态应为 已完成-失败。检查之前的配送员是否有携带订单来判断具体的失败于取餐阶段还是送餐阶段（也可通过添加订单状态来判断）。如果失败于取餐阶段，则新订单的状态置为尚未开始，指定送餐员设为任意（相当于一个新订单）。若失败于送餐阶段，则置为已取餐，指定送餐员维持原来那个不变。

3. 后台面向调度员的接口：

基本上可以在保证数据一致性的前提下，随意修改后台可见数据。大概要提供什么接口需要你列一下。

其他待解决的问题：

地图模块和后台／调度算法 交互机制

地图模块如何确定何时更新，更新哪些内容

Current Task (pick up / delivery)；Next Target address; deadline time

Coming Task 1 (pick up / delivery); Target address; expired time

Coming Task 2 (pick up / delivery); Target address; expired time

注意，每个司机都会有一个类似上面的任务列表

这个列表是否有必要显示给司机？