**武汉大学国家网络安全学院**

**研究生课程结课论文**

无人机配送路径规划问题

课程名称 ： 高级算法设计与分析

课程教师 ： 林海

姓 名 ： 刘华伟

学 号 ： 2023202210150

二零二四年六月

**目录**

[**1 问题描述** - 1 -](#_Toc170655762)

[**2 问题分析** - 2 -](#_Toc170655763)

[**3 实验设计** - 2 -](#_Toc170655764)

[**4 代码分析** - 3 -](#_Toc170655765)

[**5 结果分析** - 13 -](#_Toc170655766)

[**6 实验总结** - 16 -](#_Toc170655767)

**1 问题描述**

无人机可以快速解决最后10公里的配送，本作业要求设计一个算法，实现如下图所示区域的无人机配送的路径规划。在此区域中，共有j个配送中心，任意一个配送中心有用户所需要的商品，其数量无限，同时任一配送中心的无人机数量无限。该区域同时有k个卸货点（无人机只需要将货物放到相应的卸货点即可），假设每个卸货点会随机生成订单，一个订单只有一个商品，但这些订单有优先级别，分为三个优先级别（用户下订单时，会选择优先级别，优先级别高的付费高）：

- 一般：3小时内配送到即可；

- 较紧急：1.5小时内配送到；

- 紧急：0.5小时内配送到。

我们将时间离散化，也就是每隔t分钟，所有的卸货点会生成订单（0-m个订单），同时每隔t分钟，系统要做成决策，包括：

1. 哪些配送中心出动多少无人机完成哪些订单；

2. 每个无人机的路径规划，即先完成那个订单，再完成哪个订单，...，最后返回原来的配送中心；

注意：系统做决策时，可以不对当前的某些订单进行配送，因为当前某些订单可能紧急程度不高，可以累积后和后面的订单一起配送。

目标：一段时间内（如一天），所有无人机的总配送路径最短

约束条件：满足订单的优先级别要求

假设条件：

1. 无人机一次最多只能携带n个物品；

2. 无人机一次飞行最远路程为20公里（无人机送完货后需要返回配送点）；

3. 无人机的速度为60公里/小时；

4. 配送中心的无人机数量无限；

5. 任意一个配送中心都能满足用户的订货需求；

**2 问题分析**

该问题属于VRP问题。

车辆路径问题（Vehicle Routing Problem, VRP）, 车辆路线问题最早是由Dantzig和Ramser于1959年首次提出，它是指一定数量的客户，各自有不同数量的货物需求，配送中心向客户提供货物，由一个车队负责分送货物，组织适当的行车路线，目标是使得客户的需求得到满足，并能在一定的约束下，达到诸如路程最短、成本最小、耗费时间最少等目的。

由于难以对大规模车辆路径问题进行最优求解，因此大量研究致力于元启发式算法，如遗传算法、禁忌搜索、模拟退火和自适应大邻域搜索 (ALNS)。一些最新、最有效的车辆路径问题元启发式算法，对于包含数百或数千个交货点的问题实例，其解法与最优解的误差在 0.5% 或 1% 以内。这些方法也更加稳健，因为它们可以更容易地适应各种侧面约束。因此，对于具有复杂约束和决策集的大规模应用，元启发式技术的应用通常是首选。

本文采用遗传算法求解。

**3 实验设计**

**1** 创建初始种群

初始种群是随机选择的一组有效候选解（个体）。由于遗传算法使用染色体代表每个个体，因此初始种群实际上是一组染色体。

在本问题中，每个个体代表一种解决方案。

**2** 计算适应度

适应度函数的值是针对每个个体计算的。对于初始种群，此操作将执行一次，然后在应用选择、交叉和突变的遗传算子后，再对每个新一代进行。由于每个个体的适应度独立于其他个体，因此可以并行计算。

由于适应度计算之后的选择阶段通常认为适应度得分较高的个体是更好的解决方案，因此遗传算法专注于寻找适应度得分的最大值。如果是需要最小值的问题，则适应度计算应将原始值取反，例如，将其乘以值（-1）。

在本问题中，计算每个个体的适应度，即路径的总距离或成本。

**3** 选择、交叉和变异

将选择，交叉和突变的遗传算子应用到种群中，就产生了新一代，该新一代基于当前代中较好的个体。

选择（selection）操作负责当前种群中选择有优势的个体。

交叉（crossover，或重组，recombination）操作从选定的个体创建后代。这通常是通过两个被选定的个体互换他们染色体的一部分以创建代表后代的两个新染色体来完成的。

变异（mutation）操作可以将每个新创建个体的一个或多个染色体值（基因）随机进行变化。突变通常以非常低的概率发生。

4 算法终止条件

在确定算法是否可以停止时，可能有多种条件可以用于检查。两种最常用的停止条件是：

（1）已达到最大世代数。这也用于限制算法消耗的运行时间和计算资源。

（2）在过去的几代中，个体没有明显的改进。这可以通过存储每一代获得的最佳适应度值，然后将当前的最佳值与预定的几代之前获得的最佳值进行比较来实现。如果差异小于某个阈值，则算法可以停止。

其他停止条件：

（3）自算法过程开始以来已经超过预定时间。

（4）消耗了一定的成本或预算，例如CPU时间和/或内存。

（5）最好的解已接管了一部分种群，该部分大于预设的阈值。

在本问题中，算法终止条件为（1）达到最大世代数。

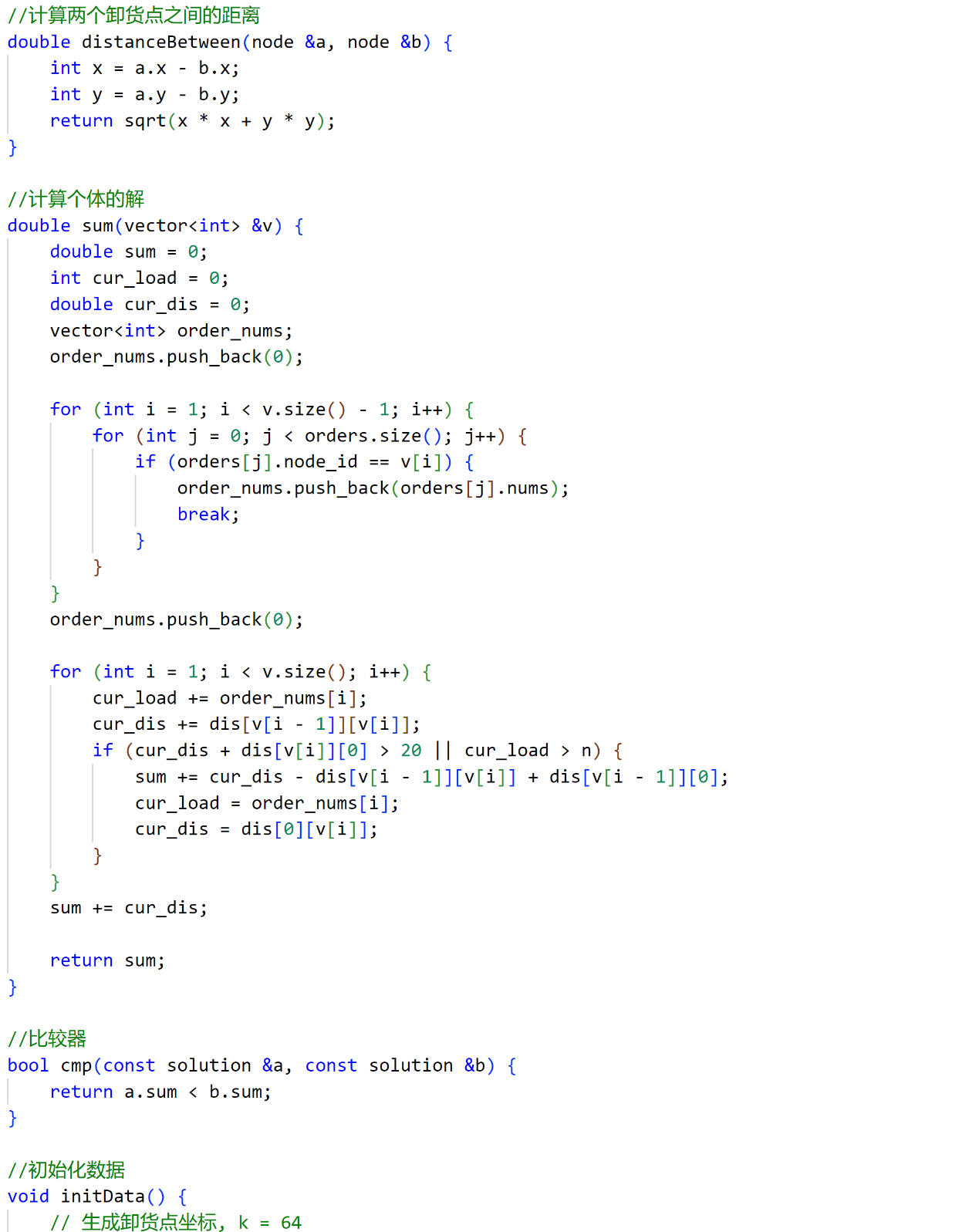
**4 代码分析**

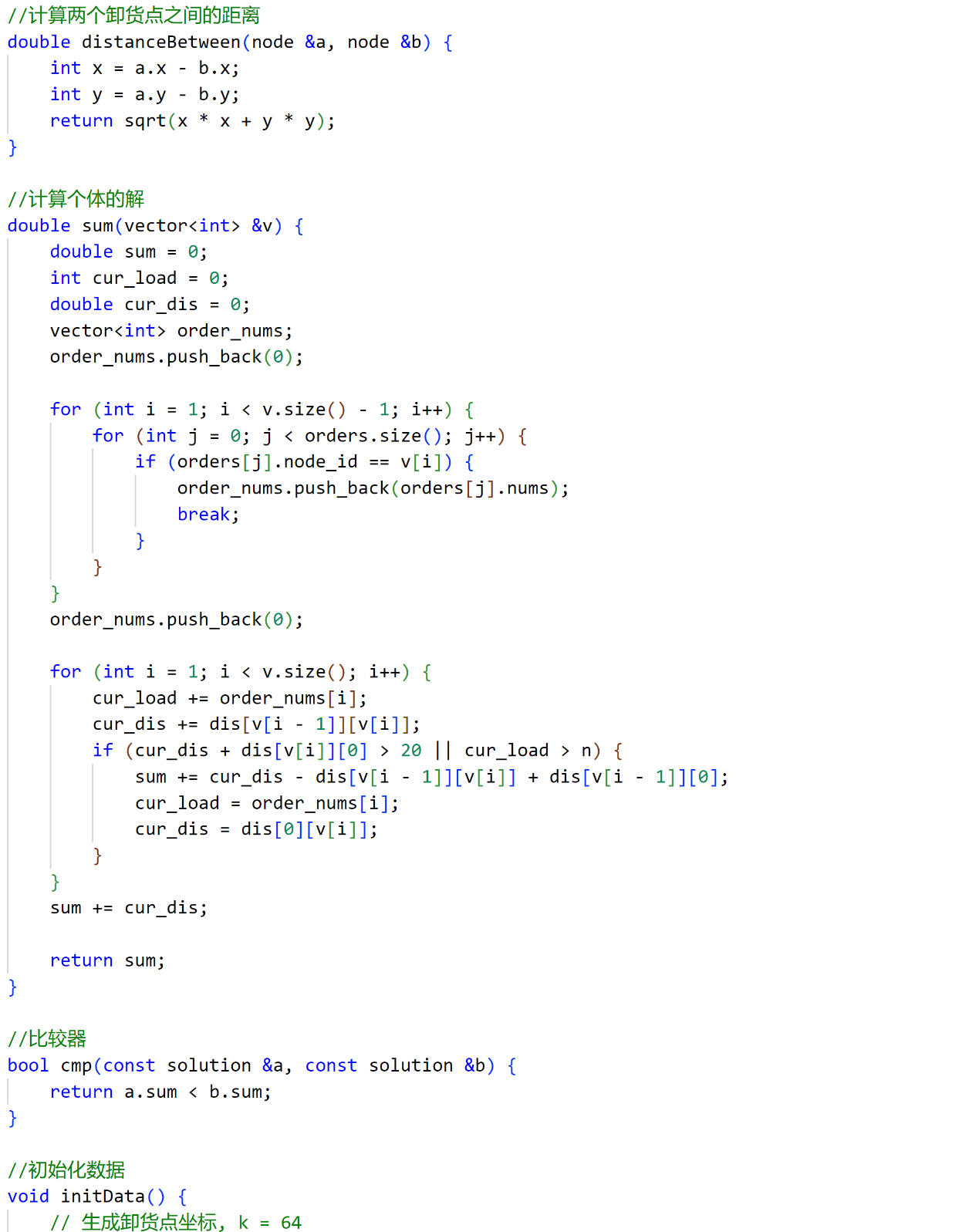
代码1：遗传算法中相关参数。其中node为卸货点，num为卸货点编号，x和y为卸货点坐标。Order为订单结构体，id为订单id，priority为订单优先级，1表示紧急，2表示较紧急，3表示一般，nums为卸货点订单数目，node\_id为卸货点编号。solution为遗传算法中的个体，该结构体包含了几个关键的成员变量：vector<int> path: 这是一个整型向量，用来存储表示该问题中路径的顺序。路径中存储了每个卸货点的编号，其中会插入0表示不同无人机路径，从起始配送中心开始，经过卸货点，最终回到起始配送中心。double sum: 表示路径的总长度或者成本，是所有无人机需要经过的总距离或成本。double F, P: 这两个变量表示适应度和累计适应度。int gen: 表示该个体所属的代数，即它是在遗传算法的第几代生成的。



代码 1

代码2：本段代码中，distanceBetween计算两个卸货点之间的距离，sum计算个体的解，cmp函数为比较器。在sum函数中，首先计算每个卸货点的订单数，然后根据每个无人机飞行距离的限制（20公里）和最大携带量的限制（m），计算所有无人机的飞行距离之和。代码如下所示。





代码 2

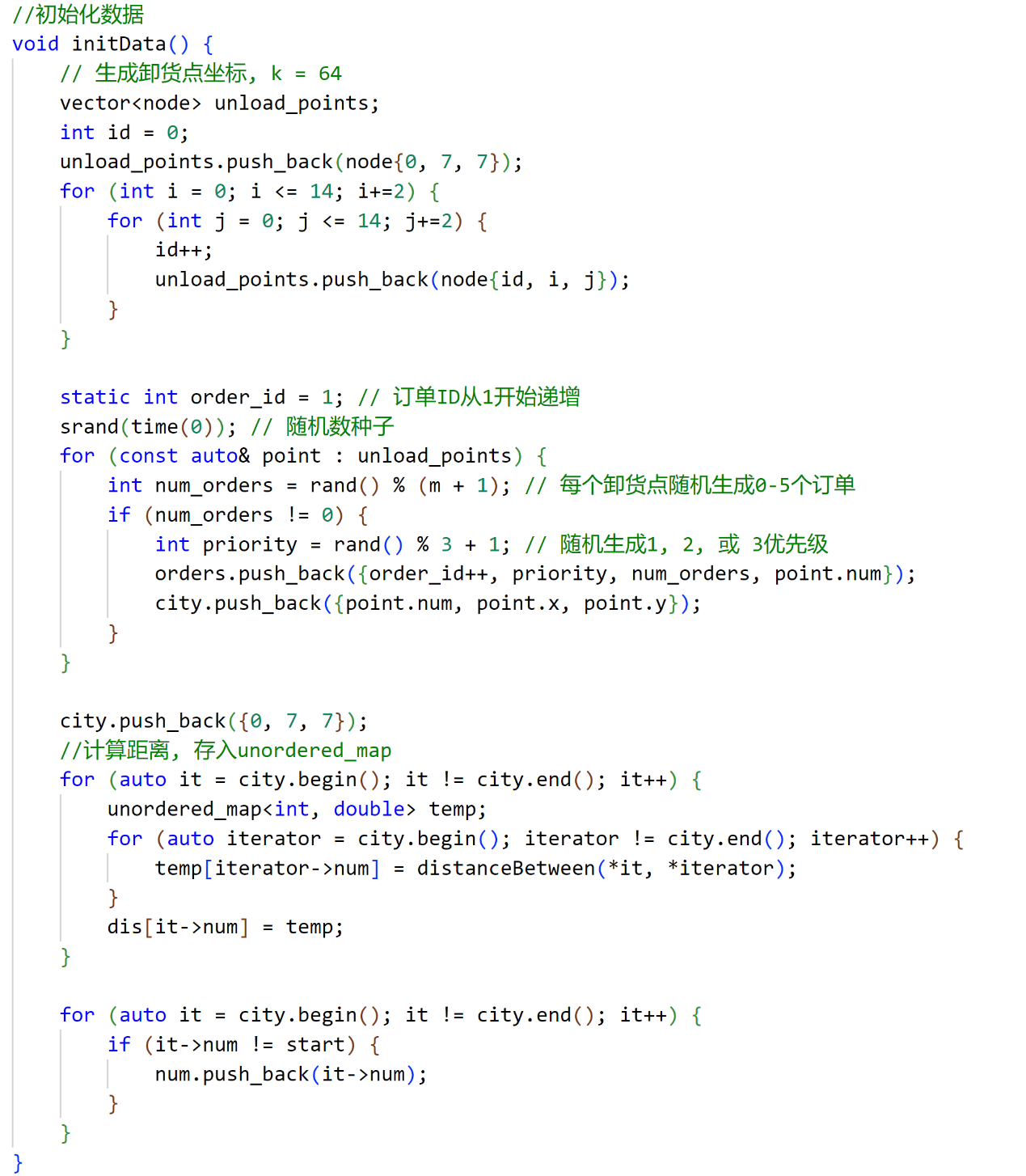
代码3：这段代码的主要功能是初始化相关数据。具体步骤如下：

生成卸货点坐标。创建一个存储卸货点的向量 unload\_points。将起点 (0, 7, 7) 加入 unload\_points，编号为0。使用嵌套的 for 循环，生成编号从1到64的卸货点，每个卸货点的坐标间隔为 (2, 2)，横纵坐标从0到14。

生成订单。使用 srand(time(0)) 以当前时间作为种子初始化随机数生成器。遍历所有卸货点，随机生成0到m个订单（num\_orders）：如果订单数量不为0，则为每个订单随机生成优先级（1到3），并创建订单（包含订单编号、优先级、订单数量、卸货点编号）存入 orders 向量。同时，将每个有订单的卸货点信息存入 city 向量。

计算并存储卸货点之间的距离：遍历 city 向量中的每个卸货点，计算它与所有其他卸货点之间的距离。使用嵌套循环，对于每个卸货点，创建一个临时的哈希表 temp 存储与其他卸货点的距离，利用 distanceBetween 函数计算距离。将每个卸货点与其他卸货点的距离存入全局哈希表 dis 中。

存储订单需配送的卸货点序列num, 用于shuffle。



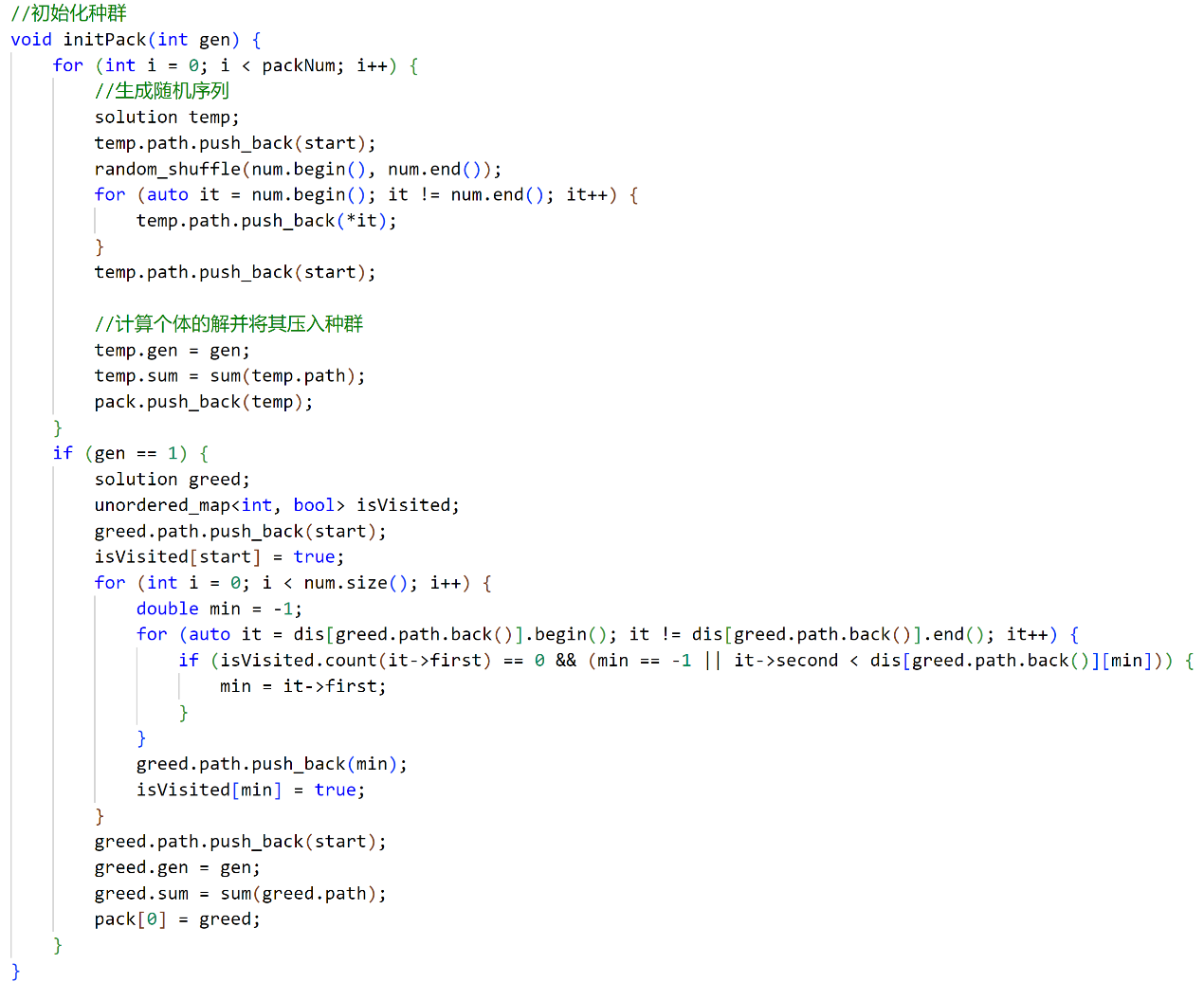
代码 3

代码4：initPack负责初始化种群，为遗传算法准备初始个体。它根据输入的gen参数生成一定数量的解决方案，并将其压入pack向量中。这段代码主要做了以下事情：

生成随机序列：对于种群中的每个个体，首先将城市编号进行随机排列，并添加到路径中，最后将起始城市添加到路径的末尾，形成一个完整的路径。

计算个体的解并将其压入种群：对于每个生成的个体，计算其路径的总长度或成本，并将该信息存储在sum中。然后将这个个体添加到种群pack中。

特殊处理第一个个体：如果gen为1，也就是初始生成的第一代个体，代码会额外生成一个基于贪婪算法的解决方案。它会从起始卸货点开始，每次选择未访问过且距离最近的卸货点，直到所有卸货点都被访问过，并回到起始卸货点形成一条完整路径。这个基于贪婪算法的解决方案会替换种群中的第一个个体，以提供一个较好的初始解。



代码 4

代码5：passOn 主要负责进行种群的传代操作。传代操作是指在找到一定数量的代数内得不到更优解时，通过一些策略对种群进行调整，以期望跳出局部最优解或增加搜索空间，从而有更多可能找到更优解。这段代码主要做了以下事情：

记录最优解的成本：将每一代中最优解的成本记录下来，并将记录存储在 record 队列中。

判断是否需要进行传代：如果记录队列中的最后一个值和前面的值相同，且这种情况持续了一段时间（这里设定为500代），则进行传代操作。

传代操作：在传代之前，保存当前最优解的路径、代数和总成本；清空原种群 pack；根据设定条件，可以对种群进行扩增或其他操作（如增加变异概率）；重新初始化种群并将保存的最优解放入种群中。进行排序，并更新当前的最优成本。最后清空记录队列。



代码 5

代码6：cross实现了遗传算法中的交叉算子，将两个父代个体进行交叉操作，生成一个新的子代个体。这段代码主要做了以下事情：

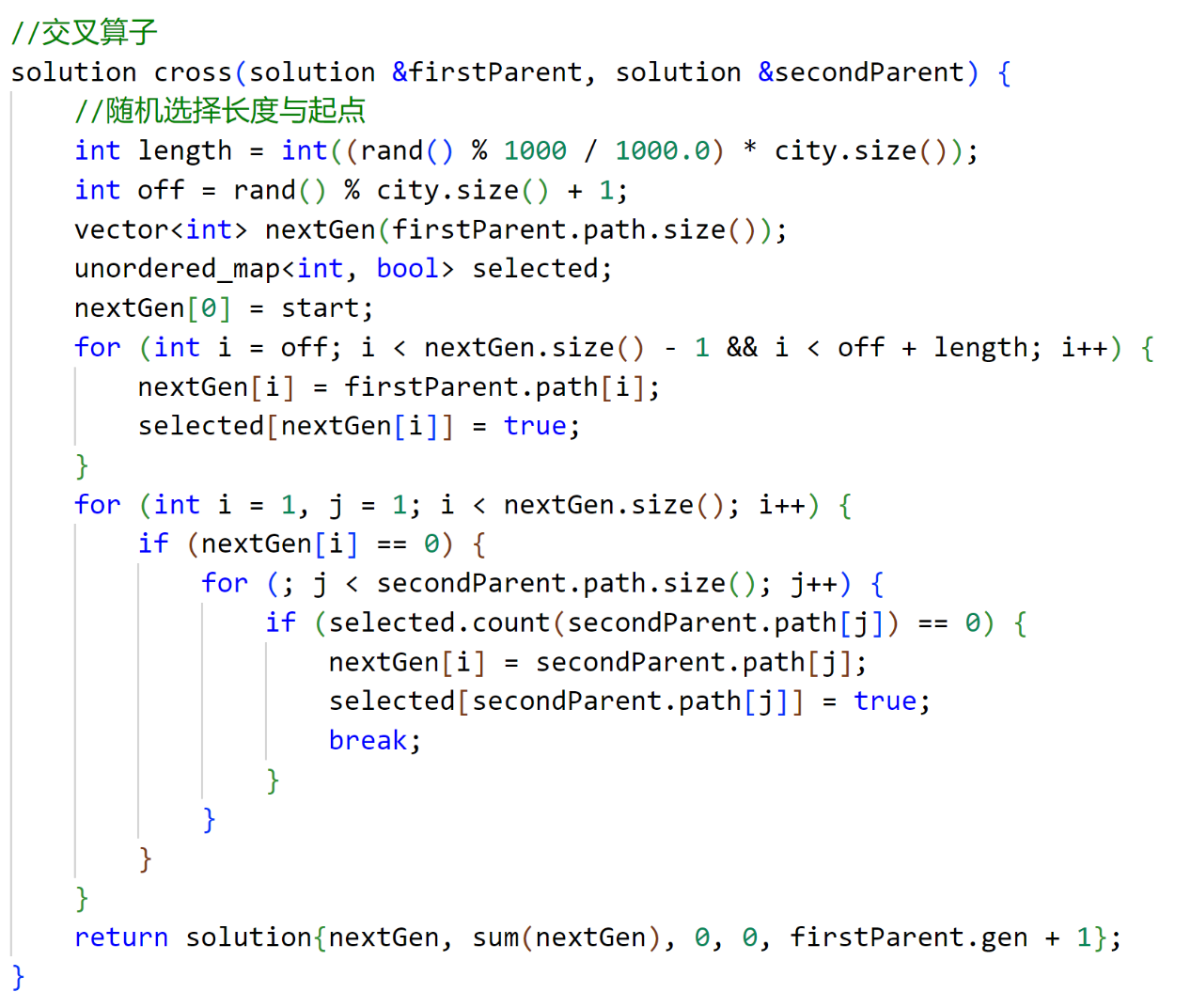
随机选择交叉的长度和起点：首先，随机选择一个长度 length（表示交叉的片段长度），同时随机选择一个起点 off。

创建子代个体 nextGen：初始化一个新的个体 nextGen，其大小与父代个体的路径长度相同。

执行交叉操作：从 off 位置开始，将 firstParent 的部分路径片段复制到子代个体 nextGen 中。根据复制的路径片段，在子代中标记已选择的卸货点。

填充剩余部分：根据 secondParent 的路径，在子代中填充未被选择的卸货点，确保路径的完整性。

生成子代个体：构建一个新的 solution 结构体，包括子代的路径、总长度、适应度、累积概率和代数信息，并返回这个新的子代个体。

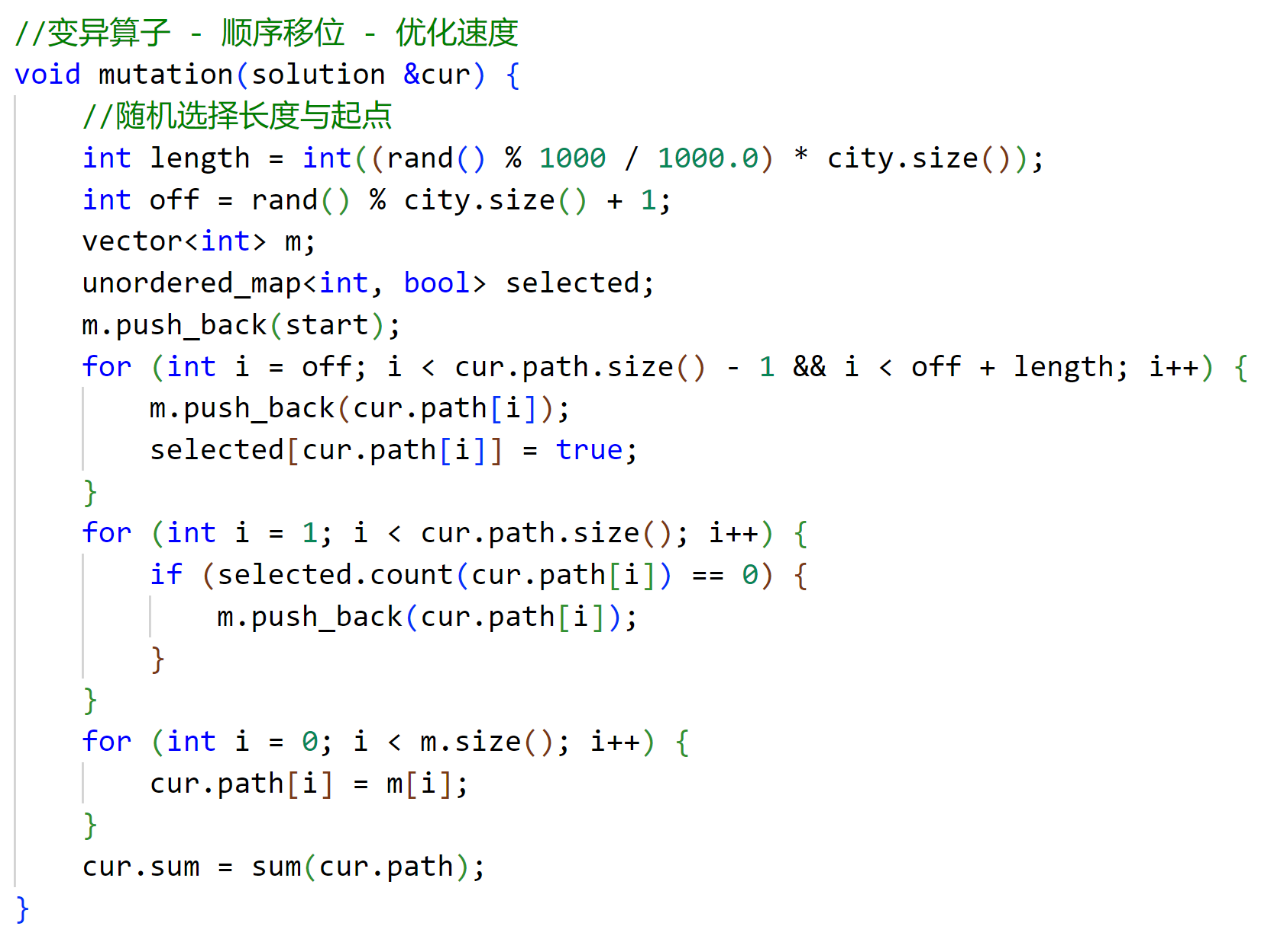


代码 6

代码7：mutation实现了遗传算法中一种变异算子——顺序移位。这个算子的目的是通过移动路径中的一部分城市顺序，以期望产生更好的解决方案。主要步骤如下：

随机选择移位的长度和起点；初始化一个临时向量 m；从 off 位置开始，将 cur 路径中一部分长度为 length 的片段复制到临时向量 m 中，根据复制的路径片段，在 selected 中标记已选择的卸货点；根据 cur 路径，将未被选择的卸货点填充到临时向量 m 中；更新路径和总长度：将临时向量 m 中的路径信息复制回 cur 的路径中。

重新计算路径的总长度 cur.sum。



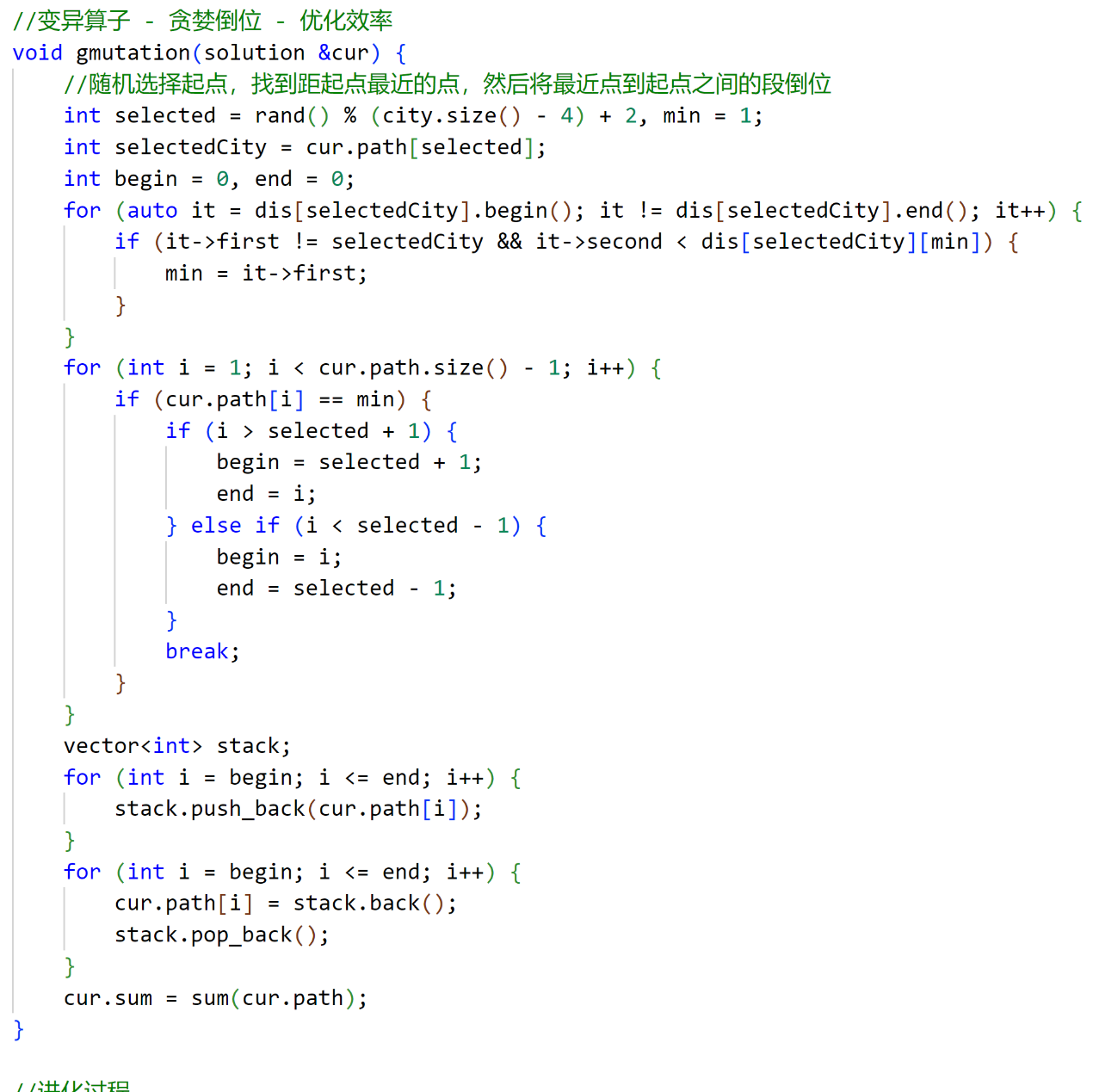
代码 7

代码8：gmutation实现了遗传算法中的另一种变异算子，使用了一种称为"贪婪倒位"的策略来优化路径。该算子旨在提高变异操作的效率，以改善种群的解。主要步骤如下：

选择起始点和最近点：首先随机选择一个起始点 selected，该点不是起始卸货点也不是终点卸货点。找到距离起始点最近的卸货点 selectedCity。

寻找倒位区间：遍历路径，找到最近点到起始点之间的段落，并记录下标 begin 和 end。

倒位操作：将记录的段落进行倒位操作，即将路径中 begin 到 end 这一段逆序排列。更新个体的路径 cur.path 和路径总长度 cur.sum。



代码 8

代码9：process实现了遗传算法的进化过程，包括选择、交叉和变异操作，生成下一代种群，以期望逐步优化路径的解。主要步骤如下：

计算总适应度：计算每个个体的适应度（it->F）和累积适应度（it->P），并将总适应度（total）累加。

保留最优个体：将最优个体直接放入下一代种群 nextGenPack 中，并将其代数加一。

选择交叉个体：使用轮盘赌选择个体进行交叉操作。针对种群中的每个个体，根据其适应度进行概率选择。选择的概率与个体的适应度成正比，高适应度个体被选中的概率更高。

交叉和变异：对于被选中的个体，进行交叉操作，生成新的个体，并加入到下一代种群中。对新生成的个体进行变异操作，以增加种群的多样性。在变异操作中，根据传代次数 Hayflick 的不同，选择了不同的变异算子，从简单的变异到更高效的贪婪倒位操作。

返回下一代种群：返回经过选择、交叉和变异操作后生成的下一代种群 nextGenPack。



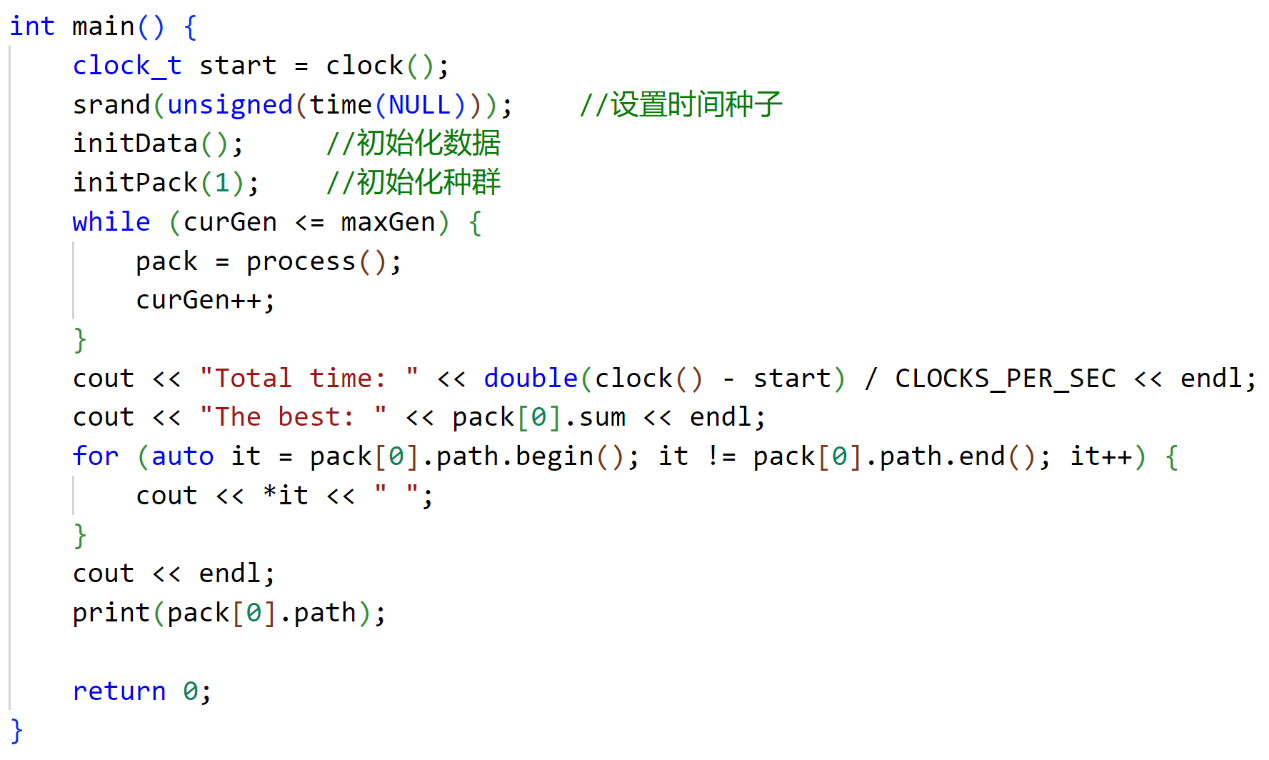
代码 9

代码10：main函数。主要步骤：

初始化操作：srand(unsigned(time(NULL))): 设置随机数生成器的种子，使用当前时间作为种子，以确保每次运行时的随机性。initData(): 初始化数据，设置64个卸货点，随机生成订单。initPack(1): 初始化种群，创建初始的个体群体，首代代数设为1。

进化过程：在没有找到满足条件的解且未达到最大代数限制时，执行进化过程；调用 process() 函数执行遗传算法的进化过程，更新种群。每完成一次进化，当前代数增加。

输出结果：在算法完成后，输出总运行时间和每个无人机的路径。

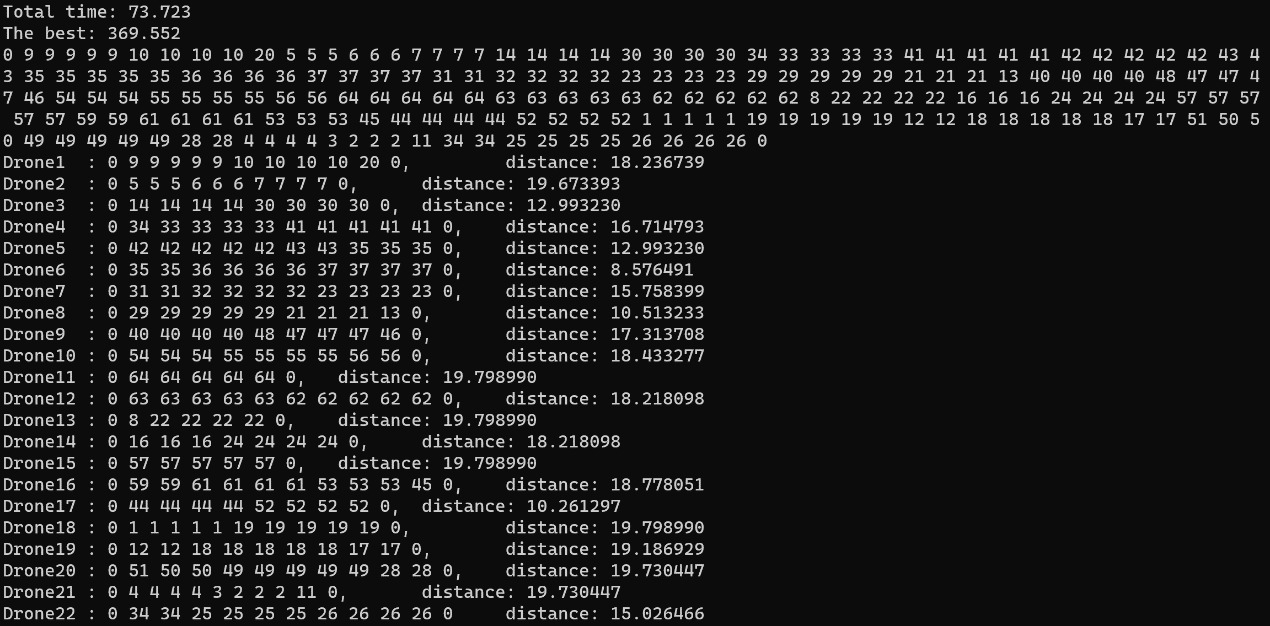


代码 10

**5 结果分析**

本文中配送中心假设为1个，坐标为（7，7），其覆盖范围为（0，0）-（14，14），配送中心到覆盖范围内最远点距离为10，满足无人机一次飞行最远路程为20公里。卸货点共有64个，在配送范围内均匀分布。

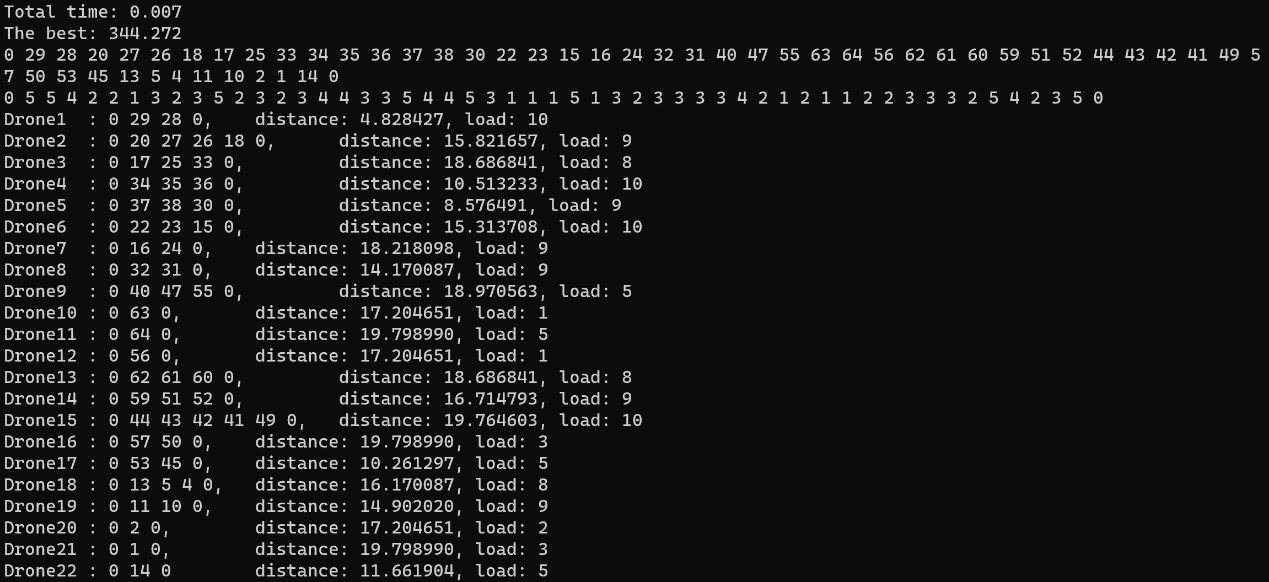
生成订单时，采用的是将该卸货点所有订单的订单数的优先级设为最高的，如果设置成一个一个的，结果1如下。



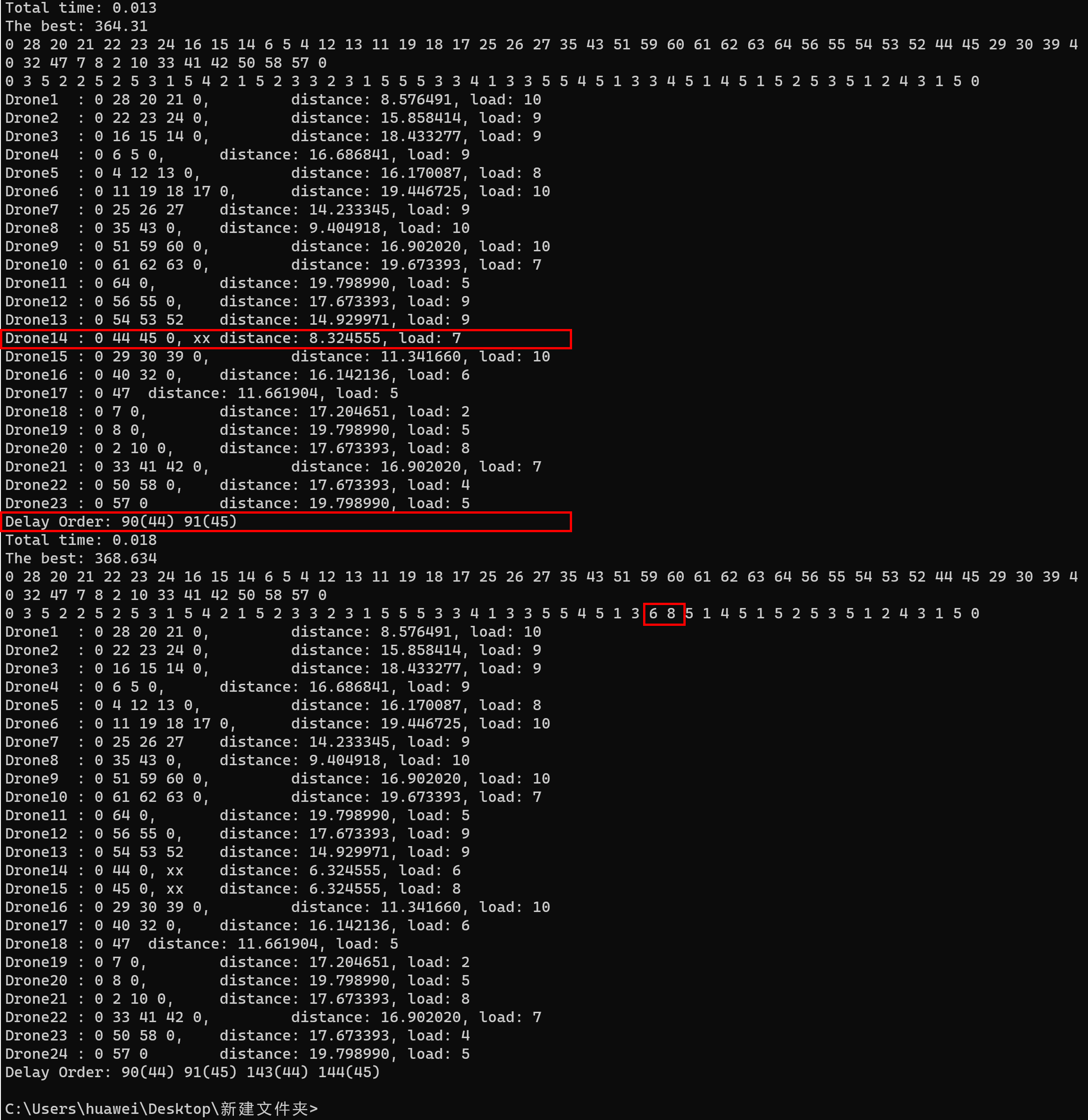
结果 1

该结果是某个时刻生成的订单，以及相应的配送方案，第一行是算法运行耗费的总时间，第二行是所有订单配送完的最短距离，第三行是无人机经过的所有卸货点，最后是每个无人机分别经过的卸货点以及飞行距离。可以看出无人机配送时，如果不考虑优先级，无人机配送也是一次配送完一个卸货点所有订单，因此后续生成订单时，假设一个卸货点生成的所有订单优先级相同，这样算法运行时间会大大减少，相应结果2如下。

结果 211

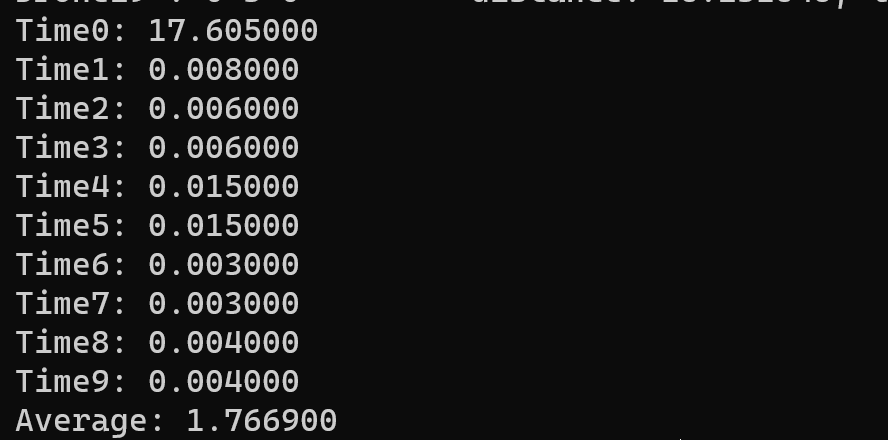


从结果2可看出，当不考虑优先级时，即所有订单生成时即完成配送，而不考虑延期配送，使用该算法会出现一些无人机飞行距离过短且配送物品较少，如无人机22，还可飞行约9公里，还可配送5个顶单。如果这种情况下订单的优先级不是紧急，即将该无人机配送的订单延期到30分钟后。因为订单生成和决策时间均为30分钟间隔，紧急订单需立即配送，较紧急订单可延期2次，第三次必须配送，一般订单可延期5次，第6次必须配送。考虑优先级结果3如下：



结果 3

本段结果截取了两次订单生成与配送策略，第一个订单无人机14配送的订单飞行距离过短，负载为7个订单，且根据优先级可延期，故进行延期操作，xx表示延期配送。可看出，第二次订单生成时，两个订单的容量超过了5，说明第一次订单已延期。最后执行时间结果4如下。



结果 4

除第一次执行时间较长，后续都较短，执行10次的平均执行时间为1.76s。执行100次的平均执行时间是0.17s。

**6 实验总结**

本次实验为vrp问题，需在满足优先级的情况下，所有无人机配送的总距离最短。本次实验使用遗传算法解决，实验结果显示系统决策平均时间为0.17s，且在满足优先级的情况下，配送完所有订单，并使得总距离最短。可进一步改进的地方在于订单配送，本文是将每个卸货点的订单作为整体来进行配送，即使是延期的订单，也会加入当前卸货点的订单作为整体进行配送，这样做简化了问题，后续改进方案可将各个订单单独配送，应能进一步改进算法，提升效率。