HIKVISION CodeChallenge

登录注册

首页 赛事信息 赛题介绍 大赛中心 论坛

海康威视 2018 软件精英挑战赛

线上复赛截止：2018.06.12

本届大赛试题以无人机 AI 对战为背景，参赛选手以自建队伍为单位，在给定地图的三维空间与其他限定条件下，进行运送货物的对战比赛，每场比赛中获得价值最多者获胜。复赛赛题在初赛题目基础上增加条件，已在官网公布。复赛答案提交于 **6 月 12 日中午 12:00** 截止。**2018 年 6 月 6 日开放提交复赛代码调试。**

报名参加

海康威视 2018 软件精英挑战赛题目

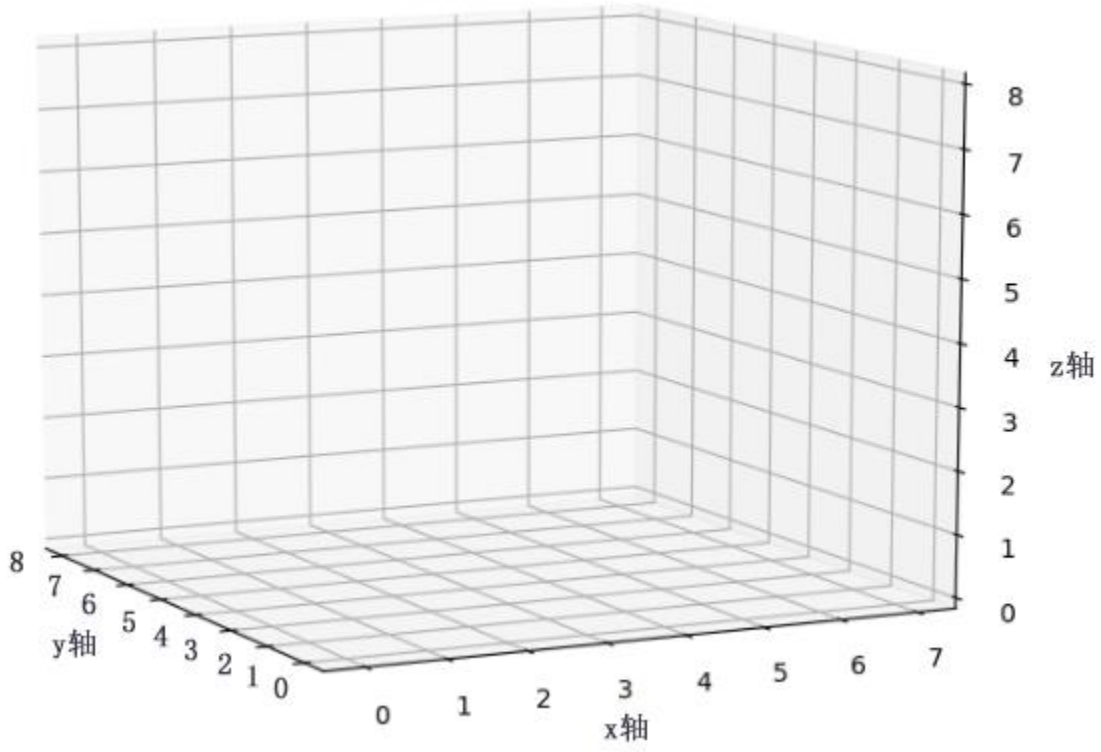
复赛题目

背景介绍

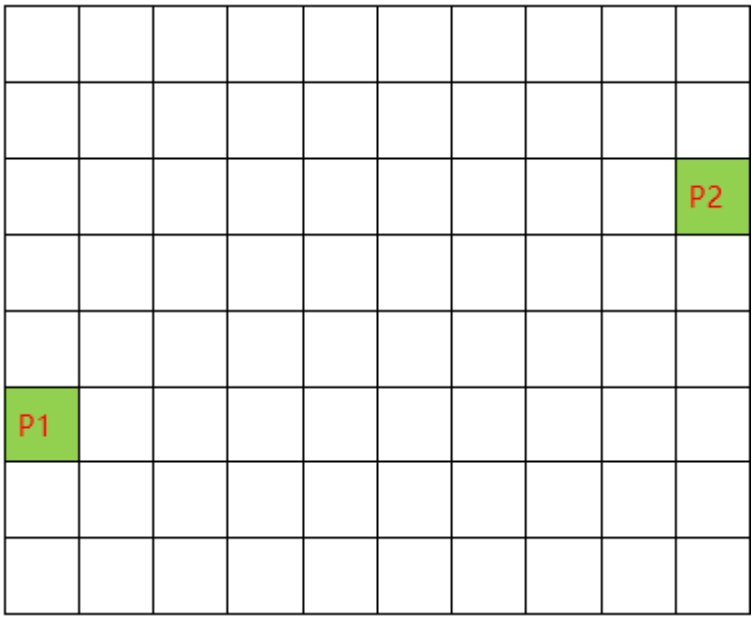
海康机器人行业级无人机业务依托海康威视视频技术的积累，进行跨技术领域的深度整合，以视频图像处理为核心、以产品安全为基石、以智能应用为导向、以满足行业需求为目标，自主研发了雄鹰系列无人机及保障低空空域安全的无人机干扰器，以丰富的产品为客户提供有针对性的行业解决方案，立足安防，专注行业。题目以无人机为主题，考察参赛队伍的综合能力。这个夏天，来一起 AI 对战吧！

了解更多海康机器人业务进入海康机器人官网

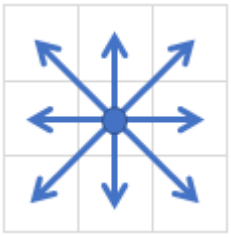
题目描述



在一个三维空间中，如上图所示，空间中的每个点对应一个坐标 P(x,y,z) 其中 x、y、z 为非负整数。x 和 y 为水平坐标， z 为垂直坐标。当 z=0 时，就是我们所说的地面，如下图所示：



在地面上存在二个停机坪 P1 和 P2，停机坪上各有 n（n > 0）架规格不一的无人机。无人机可以上下以及水平移动，每一步移动一个格子（包括斜着也是移动一个格子，例如下图中对角线的移动），同一时刻只能向一个方向移动，例如当前无人机的坐标为 P（4,4,4）则下一时刻无人机可以移动到 P(4,4,5) 、P（4,4,3）、P(3,4,4)、P(4, 3, 4)、P(5,4,4) 、P(4,5,4)、 P(5,5,4)、P(3,5,4)、 P(5,3,4), P(3,3,4)以及 P(4,4,4)。即无人机空间上不能水平和垂直同时运动。水平上可以向四周运动，如下图所示



由于无人机低空水平飞行存在很大的安全隐患，因此无人机必须上升到一定高度 H_low(H_low > 0，包括 H_low)之后，才能水平飞行。同时无人机由于他自身性能问题，飞行的最大高度为 H_high (H_high > H_low， 包括 H_high)。不同规格的无人机载重量 L（L> 0）不同。（同一时刻 一个无人机只能载一个货物，即使载重量有多余也不能载多个货物）。

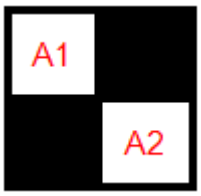
在某一时刻，水平地图上会出现不同价值的物品，（所有的物品 z=0，目的地也是 z=0，同一地点物品持续时间内不会出现多个，物品不会出现在建筑物内），物品的价值为 V（V>0），质量为（G>0），最大等待时间 T（T>0，如果物品在出现的时间为 t， 那么在 t+T 之后， 物品将消失，不能再运送，如果在物品消失之前被拾起，则物品不会再消失。）以及物品需要运送到的目的 D（x，y）。当无人机将物品送到目的地后，将获得物品的价值 V，物品不能分拆。在本次大赛的通讯协议 JSON 中，物品出现的时间用 start_time 表示，持续时间用 remain_time 表示（该值固定，不会随着时间减少），当前剩余时间用 left_time 表示， 每过一个单位时间，该值会减少 1。例如：某物品出现的时间为 60，持续时间为 5，那么第一次出现 start_time=60, remian_time=5, left_time=5，下一个单位的时候 start_time=60, remian_time=5, left_time=4。直到 left_time=0。注意 left_time=0 的时候，不能捡起物品。

选手可以用自己获取到的价值去购买无人机，以增加自己的运行效率，新购的无人机会在下一秒出现在自己的停机坪上。

在三维空间中，存在建筑物，以及雾区， 为了简化题目建筑物以柱型方式出现，即垂直方向建筑物大小不变。物品的体积可以忽略，同时雾区也是矩形方

式出现。无人机不能碰到建筑物，否则无人机将损坏不能再使用。对于雾区，对战双方不能看到对手的无人机。

对于建筑物（黑色部分表示建筑物），无人机允许如下图所示的情况进行飞行， 从 A1 位置飞行到 A2 位置。



复赛无人机说明

由于目前技术的限制，无人机的飞行过程中耗电量极大。为了方便做题，现在假设无人机空载的时候，耗电量极低，可以忽略不计。载货时，耗电量跟货物的重量成正比，比例系数为 1，例如货物重量为 100，那么单位时间的耗电量为 100 个单位电能。单位时间耗电量跟无人机有没有移动无关，即静止状态也需要耗能，包括无人机在地面的时候（停机坪除外）。不同的无人机电池容量不同。如果无人机在外面飞行过程中出现没电量（空载除外），即视为无人机坠毁（不论是否在地面）。选手可以在自己的停机坪对无人机进行充电，单位时间充的电量跟无人机机型有关。（无人机只能在空载的时候进行充电，如果载有货物，不能进行充电，否则视为犯规）。停机坪可以同时对多架无人机进行充电（充满电后，可以停留在停机坪）。初始无人机电量都为 0，（包

括比赛开始时，以及新购买的无人机），因此一开始需要充电（不打算运送货物除外）。

复赛对战协议 JSON 新增项说明

地图信息中 init_UAV 中新增剩余电量 remain_electricity，表示当前无人机还剩余的电量，由于一开始无人机没有电量，因此都为 0。

无人机价格表增加电池容量 capacity，和单位时间充电量 charge。

服务器发给参赛者一下一步指令中 UAV_we 和 UAV_enemy 中，增加剩余电量 remain_electricity，表示当前无人机还剩余的电量。同时增加 status 字段值定义 3（0 表示正常，1 表示坠毁，2 表示处于雾区），表示充电。

选手发给服务器指令 UAV_info 中，增加剩余电字段 "remain_electricity"。

复赛新增项数据说明

1. 当无人机飞到己方停机坪后，就默认开始充电，不需要选手设置，由于载货时不能充电，因此载货的时候，不能进入己方的停机坪。

2. 无人机剩余电量需要选手自己设置，例如某飞机（单位时间充电为1000）处于飞行载重状态，货物重量为 100，假如 t=40 时，服务器发送过来的数据中，该架无人机剩余电量为 500，那么这个时刻选手发送给服务器的数据中，剩余容量应该为 400。同理对于充电状态，如果服务器发送过来当前剩余容量为 1000，那么这个时刻选手发送给服务器的数据中，剩余电量为2000，（当然不能超过无人机的总容量，例如总容量为 1500，那么该时刻返回的应该是 1500）。如果当前时刻已经充满，那么每次返回只要返回充满的电量即可。

3. 进入停机坪临界说明，假设某无人机容量为 4000，单位充电量为2000，（x，y）坐标为停机坪坐标，需要飞向停机坪充电。当前服务器发送过来的数据 z=1，剩余电量为 1000，你返回给服务器时的数据为 z=0，剩余电量是 3000，随后服务器返回你的数据为 z=0，剩余电量为 3000；假如此时你还不打算飞走，你应该返回给服务器 z=0，剩余电量为 4000。

4. 离开停机坪临界说明，假设无人机 x，y 坐标为停机坪坐标，还没有充满电，就要离开停机坪，当前服务器发送过来的数据 z=0，剩余电量为1000，你返回给服务器的数据为 z= 1，剩余电量为 1000。

5. 取货进入临界值说明，假设无人机 x，y 坐标为货物坐标，货物编号为 50，重量为 100，当前服务器发送过来的数据 z=1，剩余电量为 1000，你返回给服务器的数据为 z=0，剩余电量为 1000，货物编号为 50。

6. 取货物离开临界值说明，假设无人机 x，y 坐标为货物坐标，货物编号为 50，重量为 100，当前服务器发送过来的数据 z=0，剩余电量为 1000，你返回给服务器的数据为 z=1，剩余电量为 900，货物编号为 50。

7. 放置货物进入临界值说明，假设 x，y 坐标为货物运送目标坐标，货物编号为 50，重量为 100，当前服务器发送过来的数据 z=1，剩余电量为 1000，你返回给服务器的数据为 z=0，剩余电量为 900，货物编号为 50。

8. 放置货物离开临界值说明，假设 x，y 坐标为货物运送目标坐标，货物编号为 50，重量为 100，当前服务器发送过来的数据 z=0，剩余电量为 1000，货物编号为-1，你应该返回给服务器的数据为 z=1，剩余电量为 1000，货物编号为-1。

9. 耗电完毕坠毁临界值说明，假如 t=t1 时刻，某无人机剩余电量 100，所载货物 no=10，货物重量为 100，则下一步，如果顺利到达送货点，则你应该发送给服务器 z=0，货物编号 10，剩余电量 0，随后服务器会返回给你 z=0，货物编号为-1，剩余电量为 0；如果没有到达送货点，假设为 z=1，则

你应该发送给服务器 z=1，货物编号 10，剩余电量 0，随后，服务器会返回给你该无人机坠毁。

对战方式

每次比赛有二只队伍进行对战，每张地图比赛二次（交换停机坪），在停机坪上无人机可以重叠出现，即没有碰撞体积， 一旦离开停机坪，如果碰到其他无人机， 则二架无人机都将损坏（因此离开停机坪也必须一架一架的离开）。碰撞后， 无人机如果有载物品， 则物品也一起消失。碰撞的方式有二种，

1. 同一时刻，同一位置同时出现多架无人机；如下图所示：前一时刻无人机位置如图 1， 下一时刻无人机在同一个位置，如图 2。

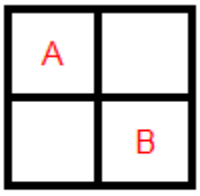


图 1

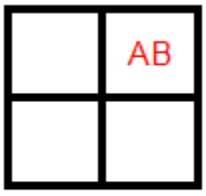


图 2

2. 无人机飞行过程中出现相遇。如下图所示 当前时刻无人机 A 和 B 在图 1 位置，下一个时刻变成了图 2 位置，就属于相遇，类似的情况也包括对角线的交换。

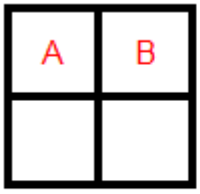


图 1

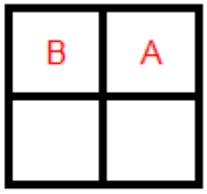
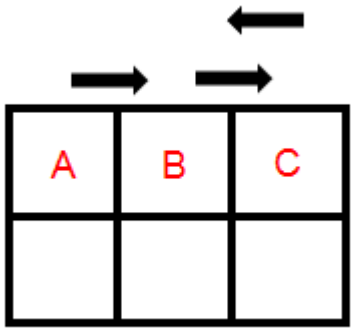


图 2

优先检测第二种情况，一旦飞机损坏，即刻从游戏中移除损坏的飞机，再检测第一种情况。例如，如下图所示，无人机 A B 向右飞行，无人机 C 向左飞行，那么下一时刻由于 B 和 C 存在相遇，因此先判断相遇，即 B 和 C 撞毁，从地图中移除，接下来判断位子重合，这个时候由于 B 和 C 已经消失，因此 A 无人机没有影响。



本次比赛不同于以往 ACM，需要存在网络交互。参赛者的程序需要去链接服务器，（服务器 IP，端口待定， 链接为长链接，中间不能关闭 socket，否则认为失败），链接到服务器后， 服务器会发送地图信息给二个参赛者，发送地图信息后表示比赛开始，该时刻为 0，参赛者程序收到地图信息，可以移动无人机准备接送货物。此时，需要参赛者将他们所控制的无人机时刻为 0 的位置发送给服务器， 服务器接受到二个参赛者的信息后，做一些处理，并将信息合并分别发送给参赛者程序，此时时刻为 1。同理参赛者需要将时刻为 1 的无人机

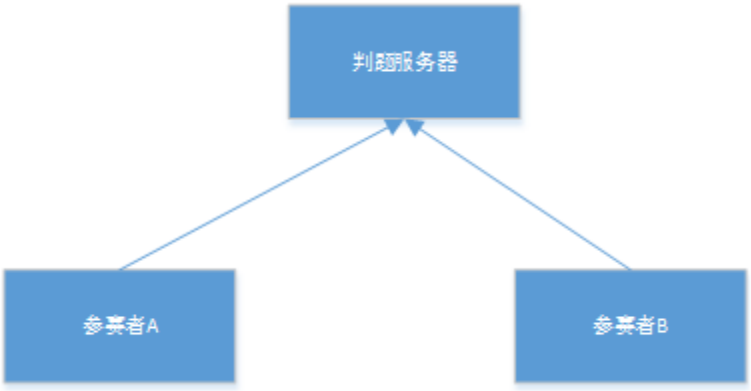
等信息发送给服务器。以此类推， 直到比赛结束。具体发送数据格式见协议说明。

当比赛结束时，获得的价值最多者获胜（这里的价值包括现有无人机价值+剩余价值），如果价值相同，那么所有运行时间最短的获胜，（运行时间指 服务器每次发送给参数者信息到接收到信息这段时间总和）。

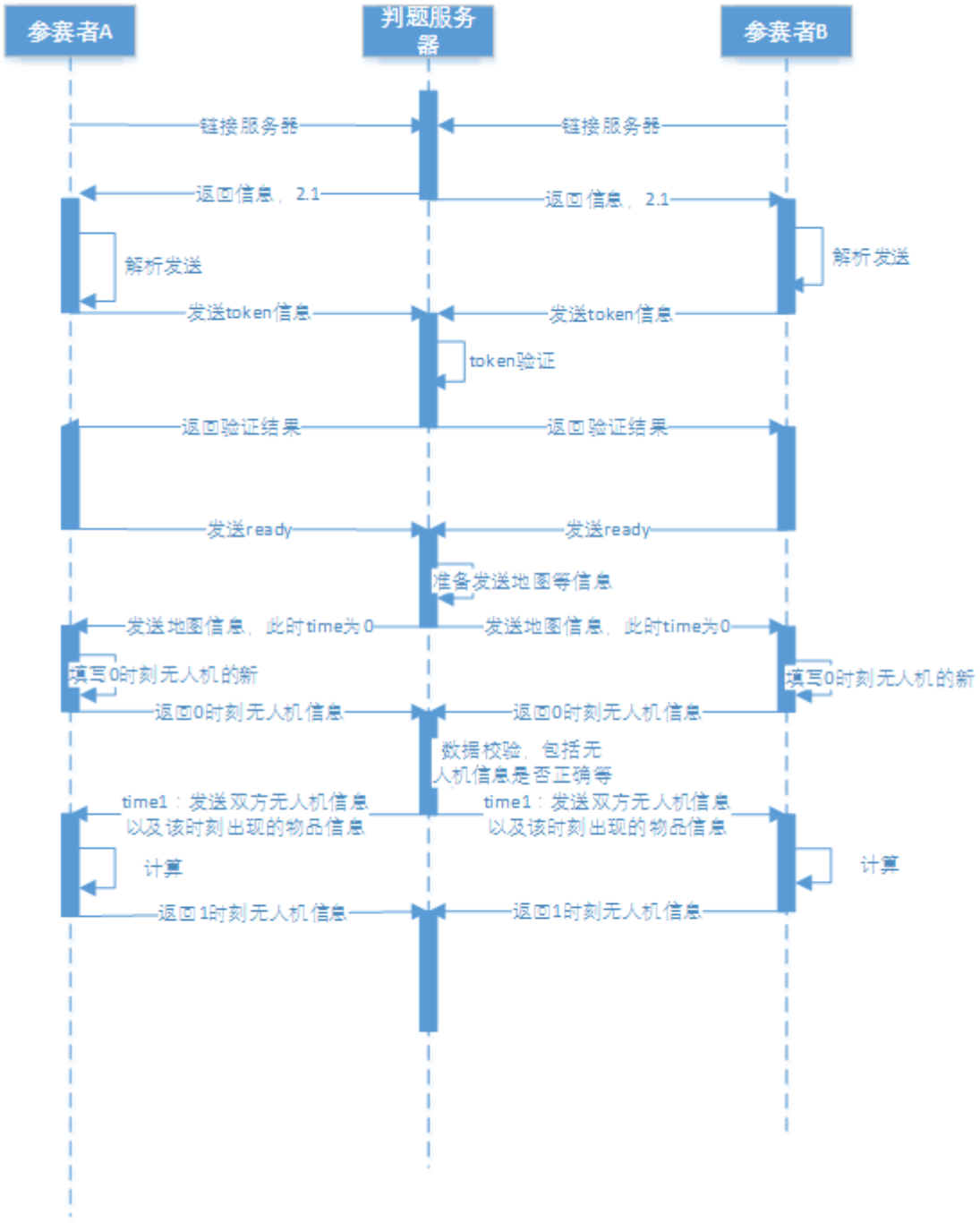
数据说明

比赛中关于链接服务器，以及协议的解析，我们已经提供相应的 demo，参赛者可以直接拿 demo 来使用。题目中所有的时间都是虚构的单位时间。为了使参赛者容易理解比赛过程中的流程，下面说明下这个过程的架构。

比赛过程中参赛者和服务器之间的链接如下（这部分 demo 中都已经实现，参赛者只要关注算法即可）：



对于他们之间的数据的时序图如下：



数据采用一问一答的模式， 当开始发送地图后，每次服务器发送给参赛者，时间+1。参赛者需要将对应 time 的无人机信息返回给服务器。注意：同一时刻可以移动多架无人机。每一架无人机每次最多移动一个格子（包括移动一个对角线）。

建筑物数据说明：建筑物用 x, y, l, w, h 表示，(x, y)表示建筑物的起始位置，l 表示长度，w 表示宽度，h 表示高度。在坐标中的位置因为 x 方向的范围为[x, x+l-1], y 方向的范围为[y, y+w-1], z 方向的范围为[0, h-1]。因此无人机可以到达例如[x,y, h] ,[x,y+w,h], [x+l, y, h],[x+l, y+w, h]等这些坐标，不算碰撞。

雾区用 x，y，l，w， b，t， 雾区的 x，和 y 方向跟建筑一样，z 方向的范围是[b, t]。

物品说明：物品只有到达时间点才会出现，对于物品的拾取，例如物品出现 time=60，出现位置为（0, 0, 0）；当前时刻为 time =59，无人机位置为（0, 0, 1）；下一时刻， 服务器发送给参赛者时，time=60，同时物品出现，这个时候，该无人机可以直接拾取物品，选手可返回无人机位置为（0, 0, 0），同时选手可设置 goods_no 为对应的物品。 相反，如果 time=59 时，无人机位置为（0, 0, 0）， 下一时刻，服务器发送给参赛者时，time=60，同时物品出现，这个时候，选手如果返回无人机位置为（0, 0, 1），并同时设置物品编号，则认为操作错误。因为你是先离开 0,0,0，再拾取物品。 对于无人机将货物送到目的地后，不需参赛者要设置 goodsno 为-1，服务器会自动将该字段设置为-1。 无人机空闲时刻可以飞到任何地方，（除了建筑物内）。由参赛者自己决定。

整个地图中，最多物品个数不会超过 256 个，建筑物个数不会超过 50 个，雾区不会超过 50 个，无人机种类不超过 64 种。

比赛结束的总价值 = 未撞毁无人机的价值+赚取物品的价值。

算法建议说明

由于该题目比较开发，要想考虑全面非常复杂，因此比较合适的做法是（仅做参考）

1.比赛开始的时候， 可以派出所有无人机，分散到地图的各个角落。

2.当物品出现后，派遣最近的无人机去拾取货物，然后根据货物的目的地进行路径规划。

3.送到货物，如果没有其他货物可以送，可以再次将无人机平均分散到地图上，如果有货物要接送，那么直接进行路径规划，去货物点拾取获取。

4.以上整个过程不要考虑对方无人机的干扰，否则问题又复杂化，会让选手无从下手。

5.如果上面的程序完成了， 接下来可以加入对方信息， 进行优化，例如用自己廉价无人机去撞毁对方昂贵无人机，或者带有重大价值的无人机等。

Demo 说明

以 C++demo 为例， 由于整个流程的架构 demo 都已经给出， 因此需要只需要关注 AlgorithmCalculationFun 函数，每一步服务器 demo 都会调用该函数，参数者只需要根据自己的算法填写 flyplane 结构体即可。对于 java 和 python 的 demo 也是类似。

C++

Demo: https://git.acmcoder.com/hikvision/UAVGoodsDemoForCPP

Java

Demo: https://git.acmcoder.com/hikvision/UAVGoodsAIDemoForJava8

Python2

Demo: https://git.acmcoder.com/hikvision/UAVGoodsAIDemoForPython2

Python3

Demo: https://git.acmcoder.com/hikvision/UAVGoodsAIDemoForPython3

协议说明

本次比赛评测时，会通过命令行启动您的程序，并附有 3 个命令行参数，第一个参数是对战控制器 IP，第二个参数是对战控制器端口，第三个参数是对战校验 Token；待 5 月中下旬开始调试时，可在我的对战中获取，获取后自行在本地启动程序并传入参数调试；正式比赛时，这 3 个参数由服务器自动生成。

本次比赛网络数据传输采用 json 格式，对于 json 格式不懂的同学，可以网上查找相应的资料，我们也会在提包里面提供相应的解析代码。数据发送和接受的格式为 前面 8 个字节表示 json 长度， 接下来为 json 数据，整个数据包长度为 8+json 的长度。

“前 8 个字节表示 JSON 长度”的说明：此 8 字节中的每个字节均为 ascii 码，表示长度对应的字符串的每一位，不够 8 位前面补零。

例如 JSON 长度为 1820，先将 1820 转换为字符串，长度为四位，然后再前面补四个 0，即变成字符串“00001820”，然后在 socket 中进行传递。

服务器发送给参赛者的 json 格式如下（赛题包中有相应的文件）：

参考

https://cdn.acmcoder.com/assets/hikvision2018/droneaisocket.html#jump6

参赛者发送到服务器的 json 格式如下：

参考

https://cdn.acmcoder.com/assets/hikvision2018/droneaisocket.html#jump5

关于 Socket 传输协议格式和 JSON 的示例，请参考：

https://git.acmcoder.com/hikvision/UAVRobotSimple

通讯以及 json 解析框架 DEMO 参考：

1、C++：

https://git.acmcoder.com/hikvision/UAVGoodsDemoForCPP

2、Java：

https://git.acmcoder.com/hikvision/UAVGoodsAIDemoForJava8

3、Python：

https://git.acmcoder.com/hikvision/UAVGoodsAIDemoForPython3 或 ht

tps://git.acmcoder.com/hikvision/UAVGoodsAIDemoForPython2

数据限制

题目中所有的数据都为非负整数，不存在浮点计算。

所有地图的长宽 0 < W,L <= 200， 高度 0 < H <=200.

无人机飞行高度 0< h_low <= h_high < H

所有数据计算不会超过 32 位整数。

一架无人机上同时只能运载一种货物。

比赛说明

初始所拥有的无人机信息会在比赛命令开始时给出，货物信息在出现时才会给出；

每一步最长运行时间：服务器发送给参赛者后，参赛者必须在 1 秒内把结果返回给服务器，否则认为认输。初赛调试阶段由于是运行在选手个人电脑上，所以没有此 1 秒的限制，正式比赛时，选手的最后一次提交程序将会运行在一台 8 核 16G 的服务器上，此时才有 1 秒的限制。一次比赛不超过 15 分钟。

比赛结束时间，分三种情况：

1.所有物品都运送完毕或消失；

2.地图上没有无人机，无人机已全部撞毁（包括有价值购买无人机，但实际并没有购买）；

3.比赛超过 15 分钟。特别地，假如地图中已经没有未被拾起的物品，则已经在飞机上的物品必须在 T=地图的长*宽*高的时间内运送完毕，但是比赛总时间还是不能超过 15 分钟。

参数者每次计算的数据必须要确保准确，如果出现与实际不匹配，那么认为认输。不匹配的情况有如下几种：

1.无人机运行轨迹不正确，例如出现跳跃。（允许无人机撞向建筑物，或者其他无人机，这个判断由服务器做）；

2.载重量不符合要求，例如载重量小的无人机去运送超过它载重的物品；

3.当前剩余价值不够的情况下，购买无人机；

4.移动已经撞毁的无人机；

5.一些不符合题意的操作。

初赛晋级规则

如何进入复赛：

1、如果最终有效代码数大于 500，则随机两两分组进行 PK，每组胜者直接进入下一轮；

2、如果最终有效代码数小于等于 500，则随机每 15 个代码组成为 1 组，组内进行两两 PK，赢一次积 1 分，组内排名前 50%的进入下一轮。

什么是最终有效代码：

截止日 12：00 前，已经编译通过，且创建对战房间，通过与机器人或自己对战，能获取到货物价值的代码中的最后一个。

PK 方式：

一张地图，随机停机坪进行一局，然后交换停机坪进行一局，两局累加综合进行比较。

当比赛结束时，获得的价值最多者获胜（这里的价值包括现有无人机价值+剩余价值），如果价值相同，那么所有运行时间最短的获胜，（运行时间指 服务器每次发送给参数者信息到接收到信息这段时间总和）。

如何提交

本次大赛使用 git 进行代码管理，请在 https://git.acmcoder.com/hikvision 自行注册 git 账号进行代码版本管理。

注册完 git 账号后，请在“用户设置”-->“授权应用”-->“生成新的令牌”进行令牌的生成，大赛系统将会使用。

注意，生成的令牌只会显示一次，请生成后立刻拷贝进行妥善保存，已经用于提交代码评测的 token 请不要删除，否则会影响您的评测。

大赛将于 5 月中下旬左右开始代码的提交，届时，请输入令牌，即可选择对应的仓库和分支进行提交。

注意：git 账号和大赛账号不必一样，为了不泄漏您的 git 账号，我们在大赛系统中使用令牌进行代码的提交。

建议由队长新建一个私有仓库，并将队员加入为协作者。

对于要提交大赛系统进行评测的分支，建议通过分支保护功能保护起来，以免 push 时不小心被覆盖。

分支保护功能在您的代码仓库下，“仓库设置”-->“管理分支”-->“保护分支”

编程语言

支持 C，C++，java， python。允许使用第三方开源库，使用多线程等特性。

C

GCC 版本：7.2.0

MAKE 版本：GNU Make 4.1

APT 源：https://mirrors.aliyun.com/ubuntu/

操作系统(64 位)：Ubuntu 16.04.3 LTS 下使用 docker 运行 buildpack-deps:artful 镜像。

示例及注意事项请查看：
https://git.acmcoder.com/hikvision/UAVGoodsDemoForC

1、你的依赖包安装请写在可执行文件./configure 中，注意使用 apt-get install -y，以免被交互打断；

2、如果你没有额外依赖包，也要准备一个空的可执行文件./configure；

3、请提供 Makefile 进行编译生成，生成的目标文件名必须是 main.exe，备注：ubuntu 也可以是以.exe 结尾。

4、评测命令：./main.exe 39.106.111.130 4010 ABC123DEFG

参数 1：39.106.111.130，是裁判服务器的 IP，您在本地调试时，可在“我的对战”中获取；正式评测时，由评测调度程序生成；

参数 2：4010，是裁判服务器允许您连接的端口，您在本地调试时，可在“我的对战”中获取；正式评测时，由评测调度程序生成；

参数 3：字符串，是本次运行的令牌，以便校验您的资格，您在本地调试时，可在“我的对战”中获取；正式评测时，由评测调度程序生成。

C++

G++版本：7.2.0

MAKE 版本：GNU Make 4.1APT 源：https://mirrors.aliyun.com/ubuntu/

操作系统(64 位)：Ubuntu 16.04.3 LTS 下使用 docker 运行 buildpack-deps:artful 镜像

示例及注意事项请查看：
https://git.acmcoder.com/hikvision/UAVGoodsDemoForCPP

1、你的依赖包安装请写在可执行文件./configure 中，注意使用 apt-get install -y，以免被交互打断；

2、如果你没有额外依赖包，也要准备一个空的可执行文件./configure；

3、请提供 Makefile 进行编译生成，生成的目标文件名必须是 main.exe，备注：ubuntu 也可以是以.exe 结尾。

4、评测命令：./main.exe 39.106.111.130 4010 ABC123DEFG

参数 1：39.106.111.130，是裁判服务器的 IP，您在本地调试时，可在“我的对战”中获取；正式评测时，由评测调度程序生成；

参数 2：4010，是裁判服务器允许您连接的端口，您在本地调试时，可在“我的对战”中获取；正式评测时，由评测调度程序生成；

参数 3：字符串，是本次运行的令牌，以便校验您的资格，您在本地调试时，可在“我的对

战”中获取；正式评测时，由评测调度程序生成。

Java

Java 版本：1.8.0_162

mvn 版本：3.5.3

mvn 镜像：http://maven.aliyun.com/nexus

操作系统(64 位)：Ubuntu 16.04.3 LTS 下使用 docker 运行 maven:3.5.3-jdk-8 镜像

示例及注意事项请查看：
https://git.acmcoder.com/hikvision/UAVGoodsAIDemoForJava8

1、不要修改 pom.xml 中的 build 节内的任何内容；

2、入口程序在./src/main/java/Main.java，不要变更 main 函数所在的文件；

3、我们最后的打包命令是：mvn clean install package

4、评测命令：java -jar output/UAVGoodsAI.jar 39.106.111.130 4010 ABC123DEFG

参数 1：39.106.111.130，是裁判服务器的 IP，您在本地调试时，可在“我的对战”中获取；正式评测时，由评测调度程序生成；

参数 2：4010，是裁判服务器允许您连接的端口，您在本地调试时，可在“我的对战”中获

取；正式评测时，由评测调度程序生成；

参数 3：字符串，是本次运行的令牌，以便校验您的资格，您在本地调试时，可在“我的对战”中获取；正式评测时，由评测调度程序生成。

```
java -version
openjdk version "1.8.0_162"
OpenJDK Runtime Environment (build 1.8.0_162-8u162-b12-1~deb9u1-b12)
OpenJDK 64-Bit Server VM (build 25.162-b12, mixed mode)
```

```
mvn -version
Apache Maven 3.5.3 (3383c37e1f9e9b3bc3df5050c29c8aff9f295297; 2018-02-24T19:49:05Z)
Maven home: /usr/share/maven
Java version: 1.8.0_162, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-8-openjdk-amd64/jre
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "4.4.0-105-generic", arch: "amd64", family: "unix"
```

Python2

Python 版本：2.7.14

pip 镜像：https://pypi.tuna.tsinghua.edu.cn/simple

操作系统(64 位)：Ubuntu 16.04.3 LTS 下使用 docker 运行 python:2.7.14 镜像

示例及注意事项请查看：

https://git.acmcoder.com/hikvision/UAVGoodsAIDemoForPython2

1、入口 main 在 main.py 中，请不要变更此文件名，更不要将 main 入口变更至别的文件；

2、您的依赖包请写在 Makefile2 文件中，注意 M 大写；

3、Makefile3 文件格式：

每行一个依赖包

依赖包格式：库名字==版本号，例如 numpy==1.14.2

4、评测命令：python2 main.py 39.106.111.130 4010 ABC123DEFG

参数 1：39.106.111.130，是裁判服务器的 IP，您在本地调试时，可在“我的对战”中获取；正式评测时，由评测调度程序生成；

参数 2：4010，是裁判服务器允许您连接的端口，您在本地调试时，可在“我的对战”中获取；正式评测时，由评测调度程序生成；

参数 3：字符串，是本次运行的令牌，以便校验您的资格，您在本地调试时，可在“我的对战”中获取；正式评测时，由评测调度程序生成。

5、评测系统已经安装的依赖包

numpy==1.14.2

scipy==1.0.1

matplotlib==2.2.2

scikit-learn==0.19.1

simpleai==0.8.1

tensorflow==1.7.0

paddlepaddle==0.11.0

注意，由于正式评测时每位选手只有 8 核 16G 的服务器，请谨慎使用机器学习框架。

Python3

Python 版本：3.6.5

pip 镜像：https://pypi.tuna.tsinghua.edu.cn/simple

操作系统(64 位)：Ubuntu 16.04.3 LTS 下使用 docker 运行 python:3.6.5 镜像

示例及注意事项请查看：
https://git.acmcoder.com/hikvision/UAVGoodsAIDemoForPython3

1、入口 main 在 main.py 中，请不要变更此文件名，更不要将 main 入口变更至别的文件；

2、您的依赖包请写在 Makefile3 文件中，注意 M 大写；

3、Makefile3 文件格式：

每行一个依赖包

依赖包格式：库名字==版本号，例如 numpy==1.14.2

4、评测命令：python3 main.py 39.106.111.130 4010 ABC123DEFG

参数 1：39.106.111.130，是裁判服务器的 IP，您在本地调试时，可在“我的对战”中获取；正式评测时，由评测调度程序生成；

参数 2：4010，是裁判服务器允许您连接的端口，您在本地调试时，可在“我的对战”中获取；正式评测时，由评测调度程序生成；

参数 3：字符串，是本次运行的令牌，以便校验您的资格，您在本地调试时，可在“我的对战”中获取；正式评测时，由评测调度程序生成。

5、评测系统已经安装的依赖包

numpy==1.14.2

scipy==1.0.1

matplotlib==2.2.2

scikit-learn==0.19.1

simpleai==0.8.1

tensorflow==1.7.0

注意，由于正式评测时每位选手只有 8 核 16G 的服务器，请谨慎使用机器学习框架。

赛事交流讨论群

欢迎加入 QQ 群交流讨论大赛技术及赛制问题！

软件精英挑战赛全国 1 群：684146505

软件精英挑战赛全国 2 群：692600084

关注海康威视招聘官方公众号获取大赛及校园招聘实时信息！

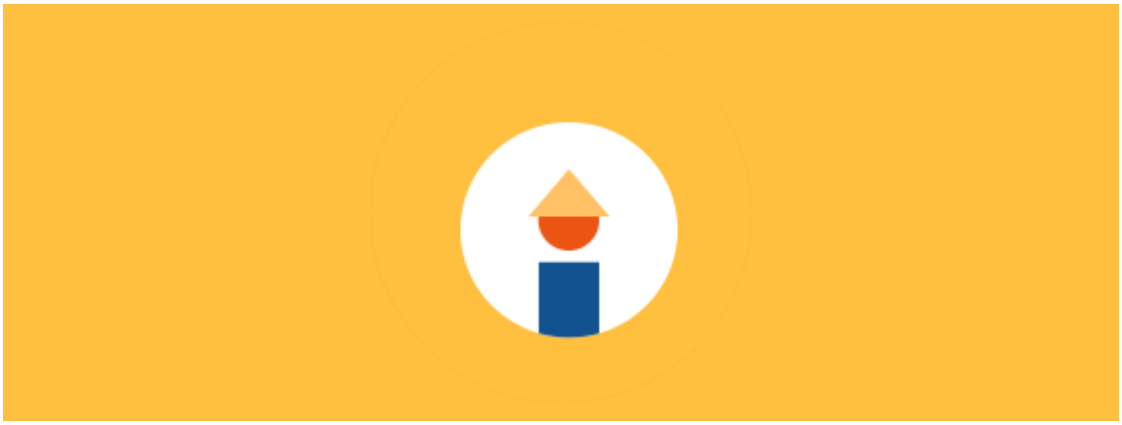
海康威视招聘官方公众号：hikvisioncareer

特别申明

1．参赛作品必须是原创作品，未侵犯第三方知识产权。且该作品未在报刊、杂志、网站（含开源社区）及其他媒体公开发表，未申请专利或进行版权登记的作品，未参加过其他比赛，未以任何形式进入商业渠道。

2．参赛者或参赛团体同意，获奖作品相关知识产权归海康威视所有。获奖者不以同一作品形式参加其他的设计比赛或转让给他方；否则，海康威视将取消其参赛、入围与获奖资格，收回奖品及并保留追究法律责任的权利。

3．因参赛作品而引发的第三方侵权纠纷，由参赛者或参赛团体承担全部法律责任。



无人机 AI 线路规划大赛接口文档

1. 概述

本次大赛设置有裁判服务器 Judger，所有比赛选手 Player 需要使用 Socket TCP 和 Judger 进行通讯，所有通讯数据均为 **JSON** 格式。

调试服务器 IP 地址及 TCP 端口：

```
Judger's IP:    39.106.111.130
Judger's port:  4010
```

2. Socket 通讯协议

2.1 选手连接裁判服务器 Judger

连接成功后，Judger 会返回一条消息：

```
{
    "notice": "token",
    "msg": "hello, what's your token?"
}
```

2.2 选手向裁判服务器表明身份(Player -> Judger)

```
{
    "token": "eyJ0eXAiOiJKV1",
    "action":"sendtoken"
}
```

其中：
token：选择对战选手后，会生成一个身份验证令牌，可以在“我的对抗”页面找到；
action：客户端连接服务器的事件类型，sendtoken 为进行身份验证。

2.3 身份验证结果(Judger -> Player)

```
{
    "token": "eyJ0eXAiOiJKV1",
    "notice": "tokenresult",
    "result": 0,
    "roundId": "vvvvv",
    "yourId": "player01"
}
```

其中：
token：身份验证令牌
notice：tokenresult 为进行身份验证结果事件
result：0 正常， -1 非法
roundId：对战房间 id
yourId：我的唯一 id

2.3.1 选手向裁判服务器表明自己已准备就绪(Player -> Judger)

```
{
    "token": "eyJ0eXAiOiJKV1",
    "action":"ready"
}
```

2.4 对战开始通知(Judger -> Player)，您需要 1s 之内返回，否则直接判负，调试阶段无此限制。

类型：json 字符串，服务器通知选手端

```
js 示例

{
    "token": "eyJ0eXAiOiJKV1",
    "notice": "sendmap",
    "time": 0,
    "map": {***}
```

```
}
```

其中：
token： 身份验证令牌
notice：发送地图信息后,表示比赛开始，该时刻为 0，参赛者程序收到地图信息，可以移动无人机准备接送货物。此时，需要参赛者将他们所控制的无人机时刻为 0 的位置发送给服务器
map：地图数据，参见地图数据格式章节。

2.5 选手返回下一步无人机坐标(Player -> Judger)

收到“比赛下一步骤指令”指令后，需要 1s 之内返回，否则直接判负，调试阶段没有此限制。取货放货规则：1、必须到了取货点(start_x, start_y, 0)，才能设置 goods_no 进行取货； 2、到了放货点(end_x, end_y, 0)，服务器会自动放货，千万不要自己将 goods_no 改为-1，否则直接违规，服务器会在这一步的结果中自动将 goods_no 设置为-1。

```
{
    "token": "eyJ0eXAiOiJKV1",
    "action": "flyPlane",
    //无人机信息：必须包含我方控制的所有的无人机的信息（除了撞毁的），信息内容包括无人机编号，xyz 坐标， goods_no 货物编号，-1 表示没有载货。当无人机飞到停机坪后，就默认开始充电，不需要选手设置，由于载货时不能充电，因此载货的时候，不能进入停机坪。
    "UAV_info": [
        { "no": 0, "x": 10, "y": 20, "z": 80, "remain_electricity": 1000,"goods_no": 0},
        { "no": 1, "x": 10, "y": 20, "z": 90, "remain_electricity": 1000, "goods_no": -1},
        { "no": 2, "x": 10, "y": 30, "z": 40, "remain_electricity": 1000,"goods_no": 2 },
        { "no": 4, "x": 70, "y": 20, "z": 20, "remain_electricity": 1000,"goods_no": 3 }
    ],

    //请求购买无人机：该字段可以没有，如果有表示要购买无人机，提交购买请求后，下一次收到的服务器发送过来的我方无人机信息中会增加相应的无人机信息，初始位置为停机坪,若购买 2 架 F1，一架 F2，写法如下：。
    "purchase_UAV": [
        { "purchase": "F1" },
        { "purchase": "F1" },
        { "purchase": "F2" }
    ]
```

```
}
```

2.6 比赛下一步骤指令(Judger -> Player)

收到“比赛下一步骤指令”指令后，需要 1s 之内返回，否则直接判负，调试阶段无此限制,与 2.5 形成循环直到比赛结束。

```
{
    "token": "eyJ0eXAiOiJKV1",
    "notice": "step",
    //比赛状态: 0 表示正常比赛中，1 表示比赛结束，收到为 1 时，参赛者可以关闭连接,
    "match_status": 0
    //当前时间: 当前的时间，每次给比赛者都会比上一次增加 1
    "time": 16,

    //我方无人机信息": "不同时间，数据不同。 我方无人机的当前信息，根据我方传递给服务器后，服务器经过计算后得到的数据， goods_no 货物编号， -1 表示没有载货物，否则表示装载了相应的货物,剩余电量 remain_electricity，表示当前无人机还剩余的电量
    //状态说明: "无人机状态 0 表示正常， 1 表示坠毁， 2 表示处于雾区，3 表示正在充电 其他数据暂时未定义"
    "UAV_we": [
        { "no": 0,  "type": "F1","x": 10, "y": 20, "z": 80, "goods_no": -1,"load_weight": 100, "remain_electricity": 1000,"status": 0 },
        { "no": 1,  "type": "F1","x": 10, "y": 20, "z": 90, "goods_no": 0,"load_weight": 100, "remain_electricity": 1000,"status": 0 },
        { "no": 2,  "type": "F1","x": 10, "y": 30, "z": 40, "goods_no": 3,"load_weight": 100,"remain_electricity": 1000,"status": 0 },
        { "no": 3,  "type": "F1","x": 50, "y": 20, "z": 30, "goods_no": 5,"load_weight": 100,"remain_electricity": 1000,"status": 0 },
        { "no": 4,  "type": "F1","x": 70, "y": 20, "z": 20, "goods_no": -1,"load_weight": 100,"remain_electricity": 1000, "status": 1 }
    ],

    //"我方目前总价值": "不同时间，数据不同，表示当前时刻，我方所取到的运送物品价值",
    "we_value": 10000,

    //"敌方无人机信息": "不同时间，数据不同。 敌方无人机的当前信息，根据敌方传递给服务器后，服务器经过计算后得到的数据，如果敌方无人机在雾区，状态为 2， x，y，z 坐标都为-1，表示无效。"
    //"状态说明": "无人机状态 0 表示正常，2 表示处于雾区，3 表示正在充电,坠毁的不返回。"
```

```
    "UAV_enemy": [
        { "no": 0, "type": "F1","x": 40, "y": 20, "z": 80, "goods_no": -
1,"load_weight": 100,"remain_electricity": 1000,"status": 0 },
        { "no": 1,"type": "F1", "x": 20, "y": 20, "z": 90, "goods_no":
7,"load_weight": 100,"remain_electricity": 1000,"status": 0 },
        { "no": 2, "type": "F1","x": 80, "y": 30, "z": 40, "goods_no": -
1,"load_weight": 100,"remain_electricity": 1000,"status": 0 },
        { "no": 3, "type": "F1","x": 90, "y": 20, "z": 30, "goods_no": -
1,"load_weight": 100,"remain_electricity": 1000,"status": 0 },
        { "no": 5, "type": "F1","x": -1, "y": -1, "z": -1, "goods_no": -
1,"load_weight": 100,"remain_electricity": 1000,"status": 2 }
    ],
    //"敌方目前总价值": "不同时间，数据不同，表示当前时刻，敌方获取到的运送物
品价值",
    "enemy_value": 30000,

    //物品信息：不同时间，数据不同，no 货物唯一编号， startxy 表示货物出现的
地面坐标，endxy 表示货物需要运送到的地面坐标， weight 表示货物的重量，value 表
示运送到后货物的价值,start_time:货物出现的时间,remain_time:货物从开始出现到
消失的持续时长,left_time: 货物可被捡起的剩余时长，这是个冗余字段，您可以从
start_time+remain_time-step 得出；一旦被捡起，remain_time 和 left_time 字段
无效。status 为 0 表示货物正常且可以被拾起,status 为 1 表示已经被无人机拾起，
status 为 2 表示已经运送到目的地，status 为 3 表示无效（无效包括运送过程中撞毁、
货物超时未被拾起等，被删除),其实您只能看见 0 和 1 状态，因为其他状态的货物会被删
除,status 为 0 时，left_time 才有意义,已经消失或送到的货物会在列表中被删除。

    "goods": [
        { "no": 0, "start_x": 3, "start_y": 3, "end_x": 98, "end_y": 3,
"weight": 55, "value": 100, "start_time":15,"remain_time": 90,
"left_time": 89,"status": 1},
        { "no": 1, "start_x": 98, "start_y": 13, "end_x": 3, "end_y": 3,
"weight": 51, "value": 90,"start_time":15, "remain_time": 9,
"left_time": 8,"status": 0},
        { "no": 2, "start_x": 15, "start_y": 63, "end_x": 81, "end_y":
33, "weight": 15, "value": 20,"start_time":15, "remain_time": 7,
"left_time": 6,"status": 0},
        { "no": 3, "start_x": 3, "start_y": 3, "end_x": 98, "end_y": 3,
"weight": 55, "value": 100, "start_time":15,"remain_time": 330,
"left_time": 329,"status": 0},
        { "no": 5, "start_x": 3, "start_y": 3, "end_x": 98, "end_y": 3,
"weight": 55, "value": 100,"start_time":15, "remain_time": 3,
"left_time": 2,"status": 0}
    ]
}
```

3. 数据格式描述

3.1 地图数据格式

```
{
    "map": {          //地图信息
        "x": 100,
        "y": 100,
        "z": 100     //天空最大高度
    },
    "parking": {     // 停机坪
        "x": 0,
        "y": 0
    },
    "h_low": 60,     //"飞行最低高度": 固定值
    "h_high": 99,  //"飞行最高高度": 固定值
    "building": [   //xy 表示建筑物的起始位置
                  // l 表示长度，w 表示宽度，h 表示高度
                  //因此水平上坐标位置为 x->x+l-1, y->y+w-1",
        { "x": 10, "y": 10, "l": 10, "w": 10, "h": 80 },
        { "x": 40, "y": 40, "l": 10, "w": 10, "h": 60 }

    ],
     //雾区: 固定值，整个比赛过程中不变，雾区个数根据地图而不同，
     //xy 表示雾区的起始位置，l 表示长度，w 表示宽度，b 表示雾区最低高度，
     //t 表示雾区的最大高度，水平上坐标为 x->x+l-1, y->y+w-1，垂直区间为
b->t",
    "fog": [
        { "x": 60, "y": 60, "l": 10, "w": 10, "b": 55, "t": 90 },
        { "x": 35, "y": 47, "l": 15, "w": 20, "b": 60, "t": 99 }
    ],
     //"一开始停机坪无人机信息": "固定值，整个比赛过程中不变，无人机个数根据地
图而不同，无人机信息包括 编号和最大载重量，编号单方唯一,剩余电量
remain_electricity，表示当前无人机还剩余的电量,初始值为 0"
    "init_UAV": [
        { "no": 0, "x":0,"y":0,"z":0,"load_weight": 100,"type": "F1",
"status": 0, "remain_electricity": 0,"goods_no":-1},
        { "no": 1, "x":0,"y":0,"z":0,"load_weight": 20 ,"type": "F3",
"status": 0, "remain_electricity": 0,"goods_no":-1},
        { "no": 2, "x":0,"y":0,"z":0,"load_weight": 20 ,"type": "F3",
"status": 0, "remain_electricity": 0,"goods_no":-1}
    ],
```

```
    //"无人机价格表": "固定值，整个比赛过程中不变，no 表示无人机购买编号，价格表根据载重不同，价值也不同，初始化的无人机中的载重必定在这个价格表中，方便统计最后价值,电池容量 capacity，单位时间充电量 charge"
    "UAV_price": [
        { "type": "F1", "load_weight": 100, "value": 300,"capacity": 9000,"charge": 1000},
        { "type": "F2","load_weight": 50, "value": 200,"capacity": 8000 ,"charge": 800},
        { "type": "F3","load_weight": 20, "value": 100,"capacity": 5000 ,"charge": 400},
        { "type": "F4","load_weight": 30, "value": 150,"capacity": 6000 ,"charge": 500},
        { "type": "F5","load_weight": 360, "value": 400,"capacity": 20000,"charge": 3000}
    ],

}
```