2023 华为软件精英挑战赛

初赛任务书

文档版本	01
发布日期	2023-03-10

华为技术有限公司

目 录

1 更新记录

表1-1 更新记录

版本	修改说明	发布时间
01	第一次正式发布。	2023-03-10

2 背景信息

机器人已经在智能制造、物流仓储、递送、医疗等很多领域显现出巨大的市场需求与商业价值。在机器人领域，华为云聚焦于探索云计算如何给当前机器人产业带来增量价值，包括：极速开发、敏捷交付、以云助端、数据驱动、终身学习、高效运维。通过对机器人建图、仿真、技能开发、运行管理等方向云原生化，已验证云可为机器人递送、巡检等场景带来 TTM 和成本 80%的降低，并有望复制到电力、港口、煤矿等军团业务。例如，华为云数据中心运维机器人如图 2-1 所示。

图2-1 华为云数据库中心运维机器人

在机器人领域，如何规划多机器人的任务执行以实现最优调度，如何控制机器人的转向与前进速度以实现全程无碰的最优路径移动等都是非常有价值的算法难题。本次比赛通过软件模拟了多机器人的运行环境以及真实机器人的状态信息，由选手来挑战这些有价值的算法难题。

期待您的精彩解决方案。

3 题目

3.1 题目介绍

题目概述

- **目标：**

  赚取更多的资金。

- **程序操控方式：**

  选手程序操控 4 个机器人执行前进、后退、旋转、购买、出售等动作来完成物品递送任务，同时赚取差价获得利润。在运行结束时，选手拥有的资金数即为最终分数，所获得的资金越高越好。

- **程序交互方式：**

  选手程序通过标准输入和标准输出与判题器进行交互。判题器运行帧率为每秒 50 帧，对于每一帧，判题器都会把场上的实时信息通过标准输入传递给选手程序，同时从选手程序读取机器人的操控指令作用到各个机器人上。每一帧有 1000/50=20ms 的时间，由于判题器需保留 5ms 执行物理计算来模拟真实场景，故选手程序需要在 15ms 内做出每一帧的决策，如果超过 15ms 未做出决策，则系统将直接忽略这一帧的控制进入下一帧，并且在直到选手程序返回控制指令之前，不会再发送状态数据给选手程序。

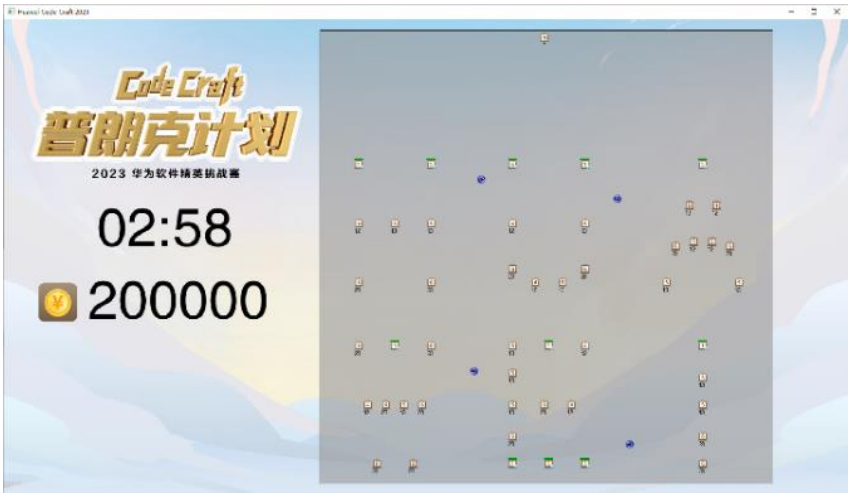
  注意，你不需要让自己的程序具备处理 50FPS 的性能也能正常运行（例如只处理 10FPS 也可以），但是处理更高的帧率可以让你实现更高精度的控制。

  程序的输入和输出格式请参考输入与输出格式。

- **判题器使用：**

  今年的比赛判题器与数据集完全开放给大家下载，并且做了跨平台设计（Windows/Linux/MacOS），大家可以根据自身习惯选择对应版本下载。但是请注意，比赛平台使用 Linux，因此无论你选择何种平台开发调试，都必须确保你的代码可以在 Linux 下编译运行。

  运行判题器中的 run_simple_demo 可快速运行一个 DEMO，运行界面如下。



判题器还可以支持键盘操作来控制机器人，更详细的判题器使用说明请参考《判题器使用说明》

术语

表3-1 术语

名词	解释
地图	地图是一个 50 米*50 米的封闭区域。
坐标系	往右为 X 轴正方向，往上为 Y 轴正方向，单位为米。地图左下角坐标为原点(0,0)，右上角坐标为(50,50)。
物品	物品由工作台生产和收购，可由机器人在一处工作台购买物品之后出售给另一个工作台，由此产生的差价来赚取利润。
工作台	工作台是一个点（没有碰撞面积），坐落于地图的空地上，其作用是收购物品进行加工、并出售成品。机器人的主要任务就是在这些工作台之间流转，执行购买、运输、出售任务以获取利润。详情参考后文中的工作台机制。
机器人	机器人是一个半径 0.45 米的圆形物体，密度 20 kg/m^2，可携带一个物品。由选手程序进行操控，可以执行**前进、后退、旋转**三种移动操作，当机器人位于工作台附近时(圆心到点的距离小于 0.4 米)，可以花钱购买物品或出售物品获得收益。此外，机器人可以在任意时刻销毁手中的物品（但不会获得收益）。**机器人在携带物品时，其半径会变大为 0.53 米，由于质量=面积*密度，所以这意味着质量也会变大。**

3.2 工作台机制

3.2.1 物品和售价机制

本题中总共有 7 种物品，分别编号为 1-7。其中，1-3 是原材料物品，由 1-3 号工作台直接产生；4-7 是加工品，由 4-7 号工作台通过其他的物品加工而成。每一种物品都有一个购买价和原始售出价，价格表如表 3-2 所示。

表3-2 物品价格表

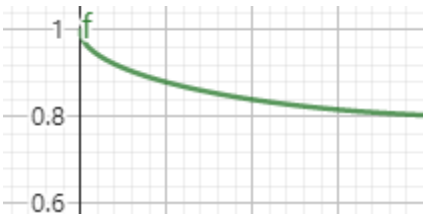
物品类型	生产配方	购买价	原始售出价	收益
1	无	3000	6000	3000
2	无	4400	7600	3200
3	无	5800	9200	3400
4	1+2	15400	22500	7100
5	1+3	17200	25000	7800
6	2+3	19200	27500	8300
7	4+5+6	76000	105000	29000

物品售价计算公式

- 物品售价 $=\lfloor$物品原始售价 $*$ 时间价值系数 $*$ 碰撞价值系数$\rfloor$
- 设 $f(x, maxX, minRate) = \begin{cases} \left[1-\sqrt{1-\left(1-\frac{x}{maxX}\right)^2}\right] * (1-minRate)+minRate, & x < maxX \\ minRate, & x \geq maxX \end{cases}$
- 时间价值系数 $=$ f(持有帧数, 9000, 0.8)
- 碰撞价值系数 $=$ f(持有时累计碰撞冲量, 1000, 0.8)

说明

- **公式含义**：物品持有时间越长，物品贬值越厉害。持有物品时碰撞的越厉害，物品贬值也越厉害。因此，想要获得高售价，就必须更快的运输，并且持有物品时尽可能避免碰撞。如碰撞不可避免，也应当尽量减轻碰撞冲量。
- **函数 f：**当 x 越接近 0 时，其单位 x 优化所带来的收益越高，这是考虑到 x 越接近 0 时越难优化，因此我们给予了极限优化更高的收益。f 函数曲线大致如下图所示。



3.2.2 工作台工作机制

本题中工作台有 9 种，分别编号为 1-9。其中，1-3 为原材料工作台，它们负责生产 1-3 号原材料而不收购其他材料。4-7 是加工工作台，收购原料的同时也生产对应的成品。8、9 为消耗工作台，只收购不生产，各工作台的收购原材料情况和生成物品情况如表 3-3 所示。

表3-3 工作台类型说明

工作台类型	是收购的原材料编号	工作周期（帧数）	生产物品
1	无	50	1
2	无	50	2
3	无	50	3
4	1,2	500	4
5	1,3	500	5
6	2,3	500	6
7	4,5,6	1000	7
8	7	1	无
9	1-7	1	无

3.2.3 生产机制

物品格的存取机制

- 每个工作台根据其收购的原材料类型都有零或多个材料格，比如 7 号工作台就有 3 个材料格分别用于放置 4、5、6 号物品。同时，对于会生产物品的工作台，还有一个成品格，用于放置生产出来的成品。机器人购买物品是从工作台的成品格取走物品，而出售物品则是将物品放置到对应的材料格。
- 一个物品格只能放一个对应物品，直到被清空方可继续放入下一个。例如：您无法在 7 号工作台（材料为 4、5、6）放置 2 号物品，也无法在 7 号工作台已放置 4 号材料的时候再放一个 4 号材料。

生产物品过程

- **消耗型工作台**：只收购原材料、不生产物品的工作台。8 和 9 号工作台就属于消耗型工作台。
- **生产型工作台**：既收购原材料也生产物品的工作台。1-7 号工作台就属于生产型工作台。生产型工作台的工作流程如下：
  1. 等待材料格物品齐备（如无需材料则跳过该步骤）。
  2. 消耗所有材料格物品并进入生产周期。
  3. 周期到达后，等待成品格空出来。

4. 在成品格放入成品，生产完成。
5. 重复上述过程进行下一轮生产。

工作台在每帧开始的时候进行工作，因此在当前帧消耗或生产的材料，会在当前帧立刻体现出来。

生产物品的一个示例：

```
4 号工作台，初始状态均为空
400 帧时，放入 1 号材料
500 帧时，放入 2 号材料
501 帧开始时，判题器检测到材料齐备，消耗材料，进入生产。
501 帧放入 1 号材料（501 帧已经消耗掉材料，可以继续放入）
700 帧放入 2 号材料
1000 帧开始时，生产周期完毕，检查成品格为空，放入 4 号物品到成品格，生产完成
1001 帧开始时，检测到材料齐备，消耗材料，进入生产。
1500 帧开始时，生产周期完毕，检查成品格满，生产被阻塞
……
1800 帧，机器人买走成品
1801 帧，工作台检测成品格为空，放入成品，生产完成。
1802 帧，继续检查原材料格…
……
```

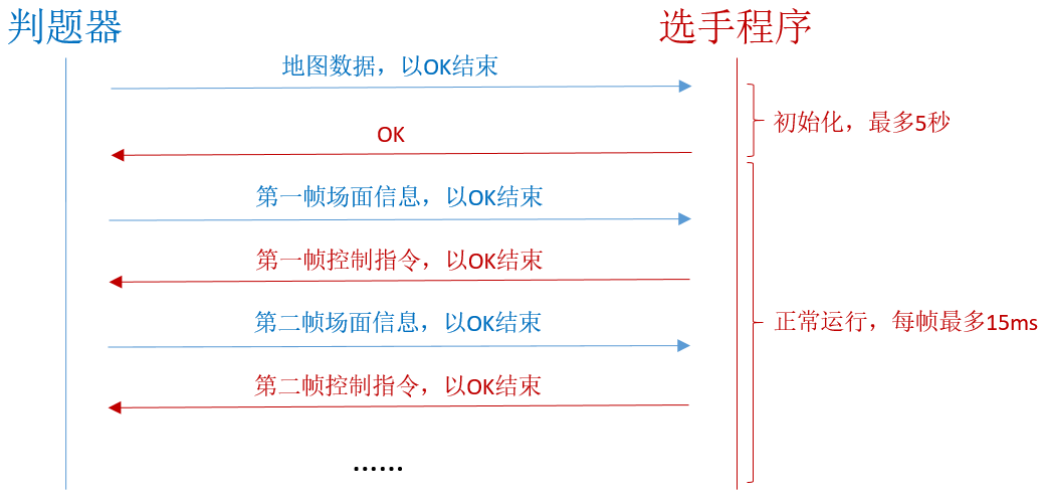
3.2.4 机器人所处工作台判定机制

当机器人与工作台距离小于 0.4 米时（圆心到点的距离），机器人可与工作台产生交互进行买卖操作。机器人最多只能身处一个工作台，倘若有附近有多个工作台满足距离小于 0.4 米，则以最近距离工作台为准，如果一样近，那么以顺序排在前面的工作台为准。

由于浮点计算都存在误差，可能会导致判题器和选手程序的计算结果不一样，因此，在输入数据中，判题器会把每一个机器人的所处工作台 ID 带给选手程序，请以此为准。

3.3 输入与输出格式

3.3.1 选手程序和判题器交互过程



3.3.2 输入格式

地图数据格式介绍

地图数据（可参考 maps/*.txt）是一个 100 行 100 列的字符矩阵。地图大小为 50 米*50 米。从此得知，每个字符对应地图的 0.5 米*0.5 米区域。地图数据中每个字符含义如下：

```
'.'： 空地。
'A'： 机器人起始位置，总共 4 个。
数字 1-9：工作台 1-9，总个数<=50，注意同一类工作台可能多次出现。
```

其余字符均为不合法字符，不会出现在地图数据中。

当地图上标记某个位置为机器人或者工作台时，则他们的坐标是该区域的**中心坐标**。地图第一行对应地图的最上方，最后一行对应地图的最下方,因此第一行第一列的中心坐标为：(0.25,49.75)。

判题器对地图数据按输入顺序逐行扫描，对遇到的机器人、工作台进行依次编号，选手可以在初始化期间即获取所有机器人、工作台的编号与位置信息。

约束：

- 由于一个格子大小容纳不下机器人，为避免区域重叠，保证机器人初始位置不会靠墙、相互不会相邻。
- 工作台个数<=50 个。

输入格式

所有数据采用文本格式通过标准输入和标准输出进行交互，数值之间用空格分隔。

- **初始化：**选手程序初始化时，将输入 100 行*100 列的字符组成的地图数据，然后紧接着一行 OK。
- **每一帧交互：**
  - 第一行输入 2 个整数，表示帧序号（从 1 开始递增）、当前金钱数。
  - 第二行输入 1 个整数，表示场上工作台的数量 K（K<=50）。
  - 紧接着 K 行数据，每一行表示一个工作台，分别由如下所示的数据构成，共计 6 个数字：

名称	数据类型	说明
工作台类型	整数	范围[1, 9]
坐标	2 个浮点 x,y	无
剩余生产时间（帧数）	整数	• -1：表示没有生产。 • 0：表示生产因输出格满而阻塞。 • >=0：表示剩余生产帧数。
原材料格状态	整数	二进制位表描述，例如 48(110000)表示拥有物品 4 和 5。
产品格状态	整数	• 0：表示无。 • 1：表示有。

  - 接下来的 4 行数据，每一行表示一个机器人，分别由如下表格中所示的数据构成，每行 10 个数字。

名称	数据类型	说明
所处工作台 ID	整数	• -1：表示当前没有处于任何工作台附近 • [0,工作台总数-1] ：表示某工作台的下标，从 0 开始，按输入顺序定。当前机器人的所有购买、出售行为均针对该工作台进行。
携带物品类型	整数	范围[0,7]。 • 0 表示未携带物品。 • 1-7 表示对应物品。
时间价值系数	浮点	携带物品时为[0.8, 1]的浮点数，不携带物品时为 0
碰撞价值系数	浮点	携带物品时为[0.8, 1]的浮点数，不携带物品时为 0。

名称	数据类型	说明
角速度	浮点	单位：弧度/秒。 • 正数：表示逆时针。 • 负数：表示顺时针。
线速度	2 个浮点 x,y	由二维向量描述线速度，单位：米/秒
朝向	浮点	弧度，范围[-π,π]。方向示例： • 0：表示右方向。 • π/2：表示上方向。 • -π/2：表示下方向。
坐标	2 个浮点 x,y	无

判题器保证工作台和机器人的**输出顺序和下标均始终一致**。

- 最后，判题器会输出一行 OK，表示所有数据已经写入完毕。
- 示例：合法的一帧输入可能是这样的：

```
1144 199346
9
1 43.75 49.25 0 0 1
2 45.75 49.25 0 0 1
3 47.75 49.25 0 0 1
4 43.75 47.25 -1 0 0
5 45.75 47.25 168 0 0
6 47.75 47.25 -1 0 0
7 44.75 45.25 -1 0 0
8 46.75 45.25 -1 0 0
9 46.25 42.25 -1 0 0
5 3 0.9657950401 1 0 0 0 -0.3755806088 47.5760498 47.40252686
-1 0 0 0 0 0 0 -0.006108176429 43.75140762 48.23157501
-1 0 0 0 0 0 0 0 3.25 2.25
-1 0 0 0 0 0 0 0 45.75 1.75
OK
```

- **判题结束：**

  判题结束时，判题器会关闭输入管道，同时选手程序会读到 EOF(end of file)，此时应当退出程序。

说明

由于浮点计算可能存在误差，判题器和选手针对机器人与工作台的距离判定不一定完全相同，因此请以机器人输入字段“所处工作台 ID”来判定当前机器人是否已经位于指定工作台，以此来决定当前帧的购买和出售行为。

3.3.3 输出格式

- **初始化：**读入地图数据并完成初始化后，选手程序应当输出一行 OK，告诉判题器已就绪。
- **每一帧交互：**
  - 第一行，选手程序需要输出一个整数，表示帧 ID 。
  - 紧接着，每行一个指令，按照 "指令 <机器人 ID> [参数 2]"的格式进行输出，机器人 ID 的取值范围是[0,3]，对应机器人的输入顺序。支持的指令如下所示：

指令	参数 1	参数 2	说明
forward	机器人 ID 取值[0, 3]	[-2,6]之间的浮点数	设置前进速度，单位为**米/秒**。 • 正数表示前进。 • 负数表示后退。
rotate	机器人 ID 取值[0, 3]	[-π,π]之间的浮点数	设置旋转速度，单位为**弧度/秒**。 • 负数表示顺时针旋转。 • 正数表示逆时针旋转。
buy	机器人 ID 取值[0, 3]	无	购买当前工作台的物品，以输入数据的身处工作台 ID 为准。
sell	机器人 ID 取值[0, 3]	无	出售物品给当前工作台，以输入数据的身处工作台 ID 为准。
destroy	机器人 ID 取值[0, 3]	无	销毁物品。

输出指令将按顺序依次执行，同一个机器人可以在同一帧内执行多条指令，因此机器人可以同一帧内出售后立刻买入。

当你输出完所有指令后，紧跟一行 OK，表示输出结束。例如，一个可能的输出是：

```
1140
forward 0 4.5
rotate 0 3.14159
forward 2 5
forward 3 -5
sell 1
buy 1
OK
```

前进和旋转指令本质上是两条设置指令，这意味着如若某一帧未对某个机器人做出前进或旋转指令（包括跳帧），将保留之前的前进或旋转参数。

说明

大多数语言会默认对输出数据做缓冲，因此在输出完一帧控制数据时，你应该主动 **flush** 标准输出以避免判题器读不到数据导致超时。此外，由于标准输出已用于比赛交互，因此不要往标准输出打日志等其他内容，以免造成命令解析错误。平台上没有任何写文件权限，故正式提交版本不要写日志文件，你可以使用 stderr 将日志输出到控制台以方便调试。

3.4 其他设定

3.4.1 随机种子设定

在真实的业务场景中，由于生产工艺的限制，任意两个机器人不可能完全相同，或多或少会存在一定的差异。故在本次比赛中，机器人的密度、最大前进速度、最大后退速度、最大旋转速度、最大牵引力、最大力矩这 6 项参数均存在万分之一的随机误差，该误差由判题器的-s 参数设定的随机种子决定，机器人之间相互不同。由于误差仅仅只有万分之一，对选手的算法影响很小，因此选手几乎可以忽略这个误差。

平台使用一个不公开的固定随机种子判题，这意味着选手本地跑的结果和提交平台之后结果会有所不同，该设计的初衷是为了避免相同指令序列在平台和本地得到完全一致的结果，从而防止有选手针对地图直接提交指令序列等影响比赛公平性的行为。

说明

相同的指令序列+相同的随机种子可以得到完全一致的结果。由于平台使用固定的随机种子，故相同的代码提交到平台理论上会得到相同的结果，除非选手的算法自身包含随机因子或是出现了超时跳帧等随机因素导致了不同的指令序列。所以一般情况下而言，没必要反复提交相同的代码浪费宝贵的提交次数。

3.4.2 常数定义

表3-4 常数定义

名称	数值	说明
比赛时长	3 分钟	无
地图尺寸	50 米*50 米	无
机器人数量	4	无
每秒帧数	50 FPS	无
初始资金	200000	无
机器人-工作台判定距离	0.4 米	二者距离小于该值时，视为机器人位于工作台上。
机器人半径（常态）	0.45 米	无
机器人半径（持有物品）	0.53 米	无
机器人密度*	20	单位：kg/m^2。质量=面积*密度，

名称	数值	说明
		故机器人变大后质量也会变大
最大前进速度*	6 米/s	forward x 6
最大后退速度*	2 米/s	即：forward x -2
最大旋转速度*	π/s	rotate x [-π, π]
最大牵引力*	250 N	机器人的加速/减速/防侧滑均由牵引力驱动
最大力矩*	50 N*m	机器人的旋转由力矩驱动

*：标记星号的 6 项参数均存在万分之一的随机误差。

3.5 异常判定与处理

- **程序异常：**判为 0 分。

  包括：程序崩溃、输出格式错误、指定机器人下标越界、输出数据量超长（1 帧内超 8KB）。
- **逻辑异常的命令：**忽略该指令，不扣分。

  在无法执行某个指令的时候执行了某个指令，该指令被忽略，不扣分。例如，无法购买、出售、销毁的时候执行了对应指令。
- **越界的命令参数：**取最接近的上下界

  例如设置前进速度超出最大速度，则取最大速度。
- **初始化响应超时：**直接进入比赛

  5 秒内未收到选手的 OK，则直接进入比赛，并且直到收到选手的 OK 之前，不会给选手下发数据。
- **控制帧响应超时：**忽略该帧控制指令

  选手程序没有在 15ms 内返回控制指令，则忽略该帧的控制行为（即：跳帧），直到收到控制指令之前，不会继续给选手程序下发数据。

  注意：跳帧后收到的控制数据会作用在当前最新的一帧，例如第 1000 帧下发的数据，选手程序直到 1010 帧才做出响应，那么该控制指令会作用于 1010 帧，此时有可能机器人已经不在原来的工作台位置，从而导致一些购买、出售指令失败（失败的指令被忽略）。

