

2022级问题求解（四）

编程任务说明

Overcooked

林朗 徐沐杰

2024 年 4 月 24 日

目录

1 任务简介	1
1.1 任务内容	1
1.2 快速上手	1
2 场景及要素介绍	2
2.1 地图	2
2.2 容器	3
2.3 菜谱和工作台	3
2.4 拿取/放置规则	3
2.5 订单	5
2.6 上菜	5
2.7 小费系统	5
3 附加规则	6
3.1 悬崖	6

目录	2
3.2 洗碗池	6
3.3 过路车辆	6
3.4 分数损失	6
4 细节补充	7
4.1 运行时间	7
4.2 随机种子	7
5 程序交互	8
5.1 通用的字符串表示	8
5.1.1 坐标	8
5.1.2 实体	8
5.2 初始化	9
5.3 帧交互	10
5.3.1 输入格式	11
5.3.2 输出格式	11
5.4 程序运行	12
5.4.1 Windows	12
5.4.2 MAC	13
6 参数	14
7 框架代码	15
7.1 框架获取	15
7.2 框架构成	15
8 评分标准	16
8.1 进度安排及赋分	16
8.2 总评构成	16
8.3 加分	17
9 一些建议	18

目录	3
10 学术诚信	19

1 任务简介

相信你一定和 npy 玩过胡闹厨房（分手厨房）吧？在游戏中，玩家扮演厨师在充满各种障碍和危险的厨房里准备食材、烹饪、上菜和清理，需要在尽可能短的时间内完成客人的订单，赚到更多的钱。

若你以前没有接触过该游戏，可[点击此链接](#)直观感受下。

1.1 任务内容

在游戏中，两人的任务分配、调度是完成游戏的关键。我们需要你的程序模拟对玩家的控制，发出移动或操控指令，使得玩家在给定的时间内获取尽量高的分数。

需要说明的是，我们并不需要你像问题求解（二）课程项目中的泡泡堂一样，写一个 GUI。相反，为了帮助你直观地观测运行过程，我们为你提供 GUI。你只需要输出当前这一帧每个玩家的行为。更详细的内容将在后续介绍。

1.2 快速上手

第 6 部分获取代码，第 5.4 部分有压缩包内程序的使用说明。在手玩模式下玩一小会。

2 场景及要素介绍

(这一部分内容较多，建议在阅读时大致了解即可。快速上手：阅读 5.4 章节，运行代码，手玩一会。)

游戏在大小为 $N \times M$ 的二维空间上进行，共有两名可以操控的玩家。游戏目标是在有限的时间内做出指定的菜品并递交至上菜口，并获得尽可能多的利润。其中大部分设定与原版分手厨房相同，对于不一样的地方，我们会用红色特殊标注。

2.1 地图

- 地面：角色可以在上面自由移动，这也是唯一能够正常通行的区域。在图形化界面上表示为浅黄色格子。
- 垃圾桶：你可以随时将手上的菜品倒入垃圾桶中。这个操作只会清空锅、盘子等容器的内容，容器本身不会消失（也不会变脏）。(合理的策略不会有使用垃圾桶的必要)
- 工作台：工作台通常用于存放或加工食材。所有工作台都能够存放食材，而不同的工作台能以不同方式加工食材。我们会在后面详细解释工作台的表示和功能。工作台本身也会阻挡人物的移动。在图形化界面上表示为深绿色。
- 食材箱：食材箱是一种特殊的工作台。地图上会有多个不同种类的食材箱，但每一个食材箱只对应一种食材。角色单次可以从食材箱中取出一份食材，不限制拿取次数。与原版不同的是：取出食材会有一定的花费。在前几周的任务中，我们都保证花费为 0。在图形化界面上表示为画有食材形状的箱子。
- 出菜口：将做好的菜品用盘子盛好后放置到出菜口中，即可完成对应的订单。地图中可能有多个出菜口，将菜提交到任意一个出菜口都视为完成订单。在图形化界面上表示为粉色。
- 盘子回收处：上菜后，5 秒后脏盘子会返回于此。在图形化界面中的表示，与原版所用图片一致。

这是最基本的地图。后面的附加规则可能会增加新的地砖。

2.2 容器

游戏中仅有煮锅、煎锅、盘子三种容器。容器中可以同时装有多个同种或不同种的食材。

2.3 菜谱和工作台

菜品有三种基本的加工方式，对应两种特殊的工作台。

- 切：切菜板是一种特殊的工作台。角色需要将单个食材放到切菜板上，并对切菜板进行一次“交互”操作，此后会开始自动切菜。切菜时角色必须在切菜板附近且不能进行其他操作。若你离开切菜板太远或“交互”了其他工作台，当前操作便会停止。
- 煮：炉子是另一种特殊的工作台。角色需要将所需食材都放入煮锅（Pot）中，再将煮锅放置到炉子上加热。煮锅中存在任意食材时炉子就会开火。在煮菜期间，角色可以离开炉子进行其他操作，加热过程不会中断。不过需要注意，当食材煮好后，炉子不会自动关闭。如果你一段时间后没有取走容器或菜品，这道菜会被煮糊。即使你取出菜品后继续组合其他食材，最终价格都只有原来的 70%，该衰减不叠加（组合两个煮糊的食材，价格还是 70%）。
- 煎：角色需要将所需食材都放入煎锅（Pan）中，再将煎锅放置到炉子上加热。具体规则和“煮”类似。

2.4 拿取/放置规则

- 每一个角色手中都可以且仅可以持有一个食材或者容器。以下将食材和容器统称为物品。
- 为了卫生，任何物品都只能手持或放置于工作台上，不能放在地上。
- 当角色手中为空时，会优先选择拿取工作台上的物品。若工作台上没有物品，则视为空操作（食材箱除外）。

- 当角色手上有物品且工作台上没有物品时，角色会将物品放置在工作台上。

为了方便操作，当角色手上和工作台上都有物品时，角色会尝试“倾倒”。倾倒行为依然满足一处地方同时仅有一个食材或容器的限制，且不存在只倾倒一部分的情况。倾倒规则具体规则如下：

- 若工作台上存在容器，则角色会优先将手上的物品（食材或容器中的内容）全部倒入工作台上的容器中。角色不会交换手上的容器。
- 若工作台上存在容器但当角色手上的容器为空时，角色会将工作台上的菜品全部盛到手上的容器中。角色不会交换手上的容器。
- 若工作台上不存在容器（为单个食材），但是角色手上存在容器，则角色会将工作台上的食材放入容器中，再将容器放到工作台上。
- 若工作台和角色手上都不存在容器（都是单个食材），则“倾倒”失败，视为空操作。
- 为了使操作更符合直觉，当工作台上的容器为煎锅或煮锅且食物已经煮熟（绿色进度条已完成），且角色手上为盘子，角色会最优先将煮好的食物盛至盘子中。食物未煮熟和其他情况下角色依然会按照上述规则优先倒入容器。

此外，倾倒行为可能涉及食物的混合。我们规定：

- 对于未被加工过的食材和已经完成加工过的食材（没有加工进度条），二者可以随意混合。
- 对于正在烹饪中的食材（存在加工进度条），可以随时添加新的生食。每加入一次生食，烹饪进度会减半。你也可以使用盘子作为容器一次性加入多个生食。
- 对于未被烹饪完毕的食材，不能被盘子盛出。
- 对于两锅正在烹饪中的食材，若属于同一种加工方式，则可以混合。烹饪进度为二者烹饪进度的均值。

- 对于已经烹饪完毕但仍然留在锅中的食物，我们保留其烹饪进度（进度条依然存在），并且继续煮可能煮糊。但是在混合食材时会按刚刚烹饪完成时的烹饪进度计算。例如煮糊的食材和生食混合会视为烹饪进度已完成一半，但煮糊的价格衰减不会清零。
- 以不同方式烹饪的食材不能混合，会视为“倾倒”失败。

需要注意这一系列规则与原版不完全相同。这一块行为较为复杂，建议自行上手对照每一个情况玩一下。

2.5 订单

同一段时间共会有 4 个顾客的订单。每个订单可能会要求不同的菜品，且不同的菜品会有不同的价格，但所需菜品一定存在于食谱中。此外，每个订单都会有时间限制。自订单出现起在规定时间内完成订单即可获得一定的分数。当你完成一个订单或订单超时后，该订单会消失并出现新的订单。在设定中，更新的订单是随机的。

2.6 上菜

地图中可能有多个出菜口。将做好的菜品用盘子盛好后放置到任意一个出菜口中，即可提交菜品。游戏会自动匹配目前剩余时间最短的订单。若没有这个订单，会视为上错菜，菜品依然会消失。需要注意的是，提交菜品时使用的盘子将会一同消失，并一段时间后出现在盘子回收处。

2.7 小费系统

此处设定与原版一致：若你的订单未超时，且按订单顺序上菜，则可获取小费。连续上菜所获取的小费：8、16、24、32、32、32、32...

3 附加规则

3.1 悬崖

你可能会从地图边界掉下去，则 5 秒后在原点复活。手中的食材会消失，盘子和锅会回到初始位置。在图形化界面上表示为白色格子。

3.2 洗碗池

在一些关卡中，盘子被回收后将不再是干净的盘子，而是脏盘子。你需要将脏盘子运送到洗碗池中清洗后才会变为干净的盘子。

洗碗时同样需要角色“交互”。交互方式与切菜相同。

3.3 过路车辆

(在最初阶段，我们不需考虑此内容。过路车辆将视任务进度决定是否加入)

可能会有一些车辆。若你被车辆撞到，则 5 秒后在原点复活。手中的食材会消失，盘子和锅会回到初始位置。

3.4 分数损失

在此，我们为你总结可能的分数损失。**以下三点均与原版不同。**

- 取食材时，需要一定的费用。
- 若灶台加工后一定时间未取走，则用该物品组合出的整体食材售卖价格 *0.7。
- 若拿着食材与另一玩家发生碰撞，则用该物品组合出的整体食材售卖价格 *0.8 (多次碰撞不累加)。

4 细节补充

4.1 运行时间

游戏的运行时间为 4 分钟。

4.2 随机种子

在游戏的过程，有许多随机出现的事件，例如：下一个订单、车辆等信息。在评测时，为了保障公平性，我们将使用统一的随机种子（不公布）进行评测。因此，你本地运行的结果，和平台上运行的结果会有一定差异。

5 程序交互

这一部分的内容有点长，不过不用担心，在框架代码里提供了处理输入的demo。但其仅能支持没有切/煎/煮时的读入，后续随着任务的深入，你需自行完善。

5.1 通用的字符串表示

5.1.1 坐标

在地图使用的坐标系中，x、y 坐标轴的正方向分别向右、向下。地图中的坐标使用两个浮点数表示，分别代表 x、y 坐标（横、纵坐标）。

5.1.2 实体

物品包括容器（如锅、盘子等）和单个的食材。在物品的字符串表示中，会首先输出当前物品的容器类型。若没有容器（为单个食材），则容器类型为空。若物品中有存在食材，则以“:”（冒号）隔开，其后跟随以空格分隔的食材类型。

食材类型不固定，以地图输入为准。容器有且仅有四种，“Pot” 蒸锅，“Pan” 煎锅，“Plate” 干净盘子，“DirtyPlate” 脏盘子。由于脏盘子是唯一允许堆叠的容器，因此其后面会紧跟一个数字代表当前堆叠的数量。

例如：煮锅中没有食材，其字符串表示为“Pot”。未使用容器装着的鱼，其字符串表示为“: fish”。盘子中装有鱼、海苔、煮好的米饭，其字符串表示为“Plate : s_rice c_fish kelp”（其中每个东西的字符串表示，将在输入中自行定义，例如这里的fish、s_rice等，不用纠结）。

3 个堆叠在一起的脏盘子，“DirtyPlates 3”。

除此之外，可能会有符号来表示是否碰撞（“@”）以及是否过度烹饪（“*”）。若当前食材发生过碰撞，则其前面的表示中会跟有一个“@”，例如“@ fish”。同理，过度烹饪表示为：“* p_fish”。又碰撞过、又过度烹饪的将被组合表示，例如“* @ p_fish”。

对于正在切的食材/洗的盘子，在表示中亦会给出当前任务的完成进度，将在前面实体部分输出完成后，以分号隔开。

例：“DirtyPlates 1 ; 120 / 180”表示当前洗盘子任务共有 180 帧，目前已经完成 120 帧。“Pot : s_rice ; 900 / 600”表示煮锅中食材已经煮熟且已经煮过了 300 帧。

5.2 初始化

你首先需要加载地图文件，其后缀为 txt。

该文件的格式为：

- 第一行两个整数 *width* 和 *height*，代表地图的大小。
- 接下来 *height* 行，每行 *width* 个字符，描述整个地图。
- 第一行一个整数 *IngredientCount* 行，代表原料箱的个数。
- 接下来 *IngredientCount* 行，每行五个输入：字符串 “IngredientBox”，该原料箱位置的横坐标、该原料箱位置的纵坐标、该原料的名字、该原料的采购价格。
- 接下来一行，一个整数 *recipeCount*，代表该关卡的配方数目。
- 接下来 *recipeCount* 行，每行四个输入：进行该加工的时间、加工前食材的名称、加工方式（“-chop->”代表切，“-pan->”代表煎，“-pot->”代表煮），加工后食材的名称。
- 接下来一行三个整数，分别代表该关卡的帧数，采用的随机种子，以及该关卡可能的订单的种类总数 *totalOrderCount*。
- 接下来 *totalOrderCount* 行，每行三个整数，分别代表该订单的有效时间，该订单的售卖单价，该订单的权重（其出现概率：该订单的权重 / 所有订单的权重）。三个整数之后有若干个字符串，代表该订单需要的食材名称。
- 接下来一行，一个正整数 $k = 2$ ，表示出现的玩家数量。
- 接下来 $k(k = 2)$ 行，每行两个浮点数 (x, y) ，表示每个玩家的初始位置。

- 接下来一行，一个正整数 *entityCount*，表示出现的实体数量。
- 接下来 *entityCount* 行，每行两个整数，代表该实体的横坐标、纵坐标。整数后接一个字符串（“pan” 代表煎锅，“pot” 代表煮锅，“plate” 代表盘子）

对于其中的地图：

- . 代表地砖
- * 代表工作台
- _ 代表悬崖
- 不同的大写字母代表不同的原料箱（仅作为占位符，具体参数由 “Ingredient-Box” 部分描述）
- *t* 代表垃圾桶
- *c* 代表切菜板
- *s* 代表灶台
- *k* 代表洗水池
- *r* 代表盘子洗净后会出现的位置
- *p* 代表客人用餐一定时间后返还的脏盘子的位置。
- \$ 代表食材提交窗

之后，你有 5 秒的时间进行相关预处理。

5.3 帧交互

在初始化阶段结束后，程序运行时的交互，以每一帧为一个阶段。在每一帧到来时，我们的程序将会向你发送当前局面的状态信息，你需要在一帧的时间内作出回应，告知每个玩家的操作状态。若在一帧内，你的程序未有任何输出，将视为放弃当前这帧对于玩家的操控。

接下来是每一帧的输入格式与输出格式。

5.3.1 输入格式

第一行一个字符串“Frame”，之后接一个整数，代表当前的帧编号。游戏初始化后的第一帧为“Frame 0”。

接下来一行，三个整数，表示剩余的帧数，目前的收入，以及当前的订单个数 *OrderCount*。

接下来 *OrderCount* 行，每行前两个整数表示该订单的剩余帧数以及该订单的售出收入，之后若干个字符串，表示该订单的组成。

接下来一行，一个正整数 $k = 2$ ，代表玩家个数。

接下来 $k = 2$ 行，四个浮点数代表玩家的横坐标、纵坐标、x 方向速度、y 方向速度、剩余复活时间（若为 0，则代表其存活）。若该玩家手里有东西，则接下来一个分号，分号后一个空格，空格后为一个实体，实体的输入格式见 5.1.2

接下来一行，一个整数 *entityCount*，表示出现的实体数量。

接下来 *entityCount* 行，每行前两个浮点数，代表该实体的横、纵坐标。其后一个字符串，代表该实体类型及其内容，具体内容参考实体小节。

5.3.2 输出格式

一行必须为“Frame” + 整数 x，中间以空格隔开，代表你需要在第 x 帧中执行后续的操作。其中 x 必须和游戏当前进行到的帧相同，否则游戏会忽略这一次操作。

随后 $k = 2$ （玩家个数）行，每行输出一个符合以下任意一种格式的字符串，代表第 i 个玩家该帧的操作：

- Move [LRUD]*n
- Interact [LRUD]*n
- PutOrPick [LRUD]*n

在一帧中同一个角色仅能选择一种操作。

Move 表示移动操作，“Move LU”代表该玩家向左上方移动，若仅为“Move”而不加方向，视为“刹车”，角色会尽快停止运动。

Interact 视为交互, *Interact L* 视为该玩家与其左边的工作台交互 (例如切菜时, 需要触发, 等同于在手玩中玩家 1 对其左面的物品摠下了 J)。

PutOrPick 视为拿东西, *PutOrPick L* 视为该玩家拿取/放下其左边的物品 (例如切菜时, 需要触发, 等同于在手玩中玩家 1 针对其左面的物品摠下了 space)。

在输出这一帧的操作之后, 不要忘记刷新输出缓冲区。

5.4 程序运行

因 linux 不同发行版的环境有较大差别, 我们不提供 linux 下的图形化程序。

我们提供的程序主要有 2 个: `QtOvercooked`和`runner`。

`QtOvercooked`为可视化程序, 其运行需要以下参数:

- `-l` 地图路径
- `-p` 你的策略代码的可执行文件

其中, `-l` 参数必填。若只有 `-l` 参数, 没有 `-p` 参数, 则进入了手玩模式, 此时的交互规则为:

- 玩家一: “WASD” 方向, “空格” 拿取/放下物品, “J” 交互 (洗盘子、切菜等)
- 玩家二: “上下左右” 方向, “回车” 拿取/放下物品, “ctrl” 交互 (洗盘子、切菜等)

当使用 `-p` 参数时, 游戏会运行你的程序并按照上一小节的方式获取玩家的操作。程序的交互记录会被保存在当前目录下的`cliilog.txt`中。

`runner`程序为快速测评程序, `-p`为必填参数。启动后会无视帧率限制尽快地运行游戏。运行过程中不会有任何输出, 最终输出的数字为游戏分数。

5.4.1 Windows

压缩包中含有`QtOvercooked`、`runner`以及其他依赖文件。运行方式见上。

在运行过程中遇到问题, 若你在 RTFM/RTFW 后仍无法解决, 请遵守提问的智慧咨询。

5.4.2 MAC

压缩包中含有runner, QtOvercooked 见:

QtOvercooked.app/Contents/MacOS/QtOvercooked

在运行过程中遇到问题, 若你在 RTFM/RTFW 后仍无法解决, 请遵守提问的智慧咨询。

6 参数

游戏一秒钟固定为 60 帧，一帧17ms。

地图中一格为1m。玩家半径为0.35m。玩家最大速度为5m/s。玩家最大加速度为 20m/s^2 （存在摩擦力，请不要依赖这个数据）。玩家最大交互距离为1.3m。

掉下悬崖/被车撞的复活时间：5 秒（300 帧），若手里持有容器，容器也会在 5 秒后回归原位。

煮/煎操作完成后，离过度烹饪的时间间隔：10 秒（600 帧）。

洗一个盘子的时间：3 秒（180 帧）。

用餐后归还盘子的时间：5 秒（300 帧）。

7 框架代码

7.1 框架获取

`git clone https://git.nju.edu.cn/psv/overcooked-2022.git`

含有QtOvercooked和runner的压缩包下载：

`https://box.nju.edu.cn/d/cac3186e5f034c9099c6/`

7.2 框架构成

```
.
├── include
│   ├── enum.h
│   └── framework.h
├── source
│   ├── framework.cpp
│   └── main.cpp
├── maps
├── makefile
├── CMakeLists.txt
└── README.md
```

source/main.cpp 为主函数。

source/framework.cpp 提供了一个处理输入的 demo。目前他只能兼容不用切/煮/煎功能的应用，建议至少对着输入格式阅读一遍。

其余信息以及提交方式见 README.md。

8 评分标准

8.1 进度安排及赋分

在每一周，我们会发布符合这周主题四个地图。

例如（**具体会视情况调整**）：

- 第一周（4.24 ~ 5.1）：在订单上不考虑食材的切、煎、煮及食材复合，在移动上不考虑障碍物及车辆。
- 第二周（5.1 ~ 5.15）：在移动上不考虑障碍物及车辆。
- 第三周（5.8 ~ 5.15）：存在障碍物及过路车辆（?）。
- 第四周（5.15 ~ 5.22）：优化——狭窄道路。
- 第五周（5.22 ~ 5.29）：优化——合作。
- 第六周（5.29 ~ 6.5）：完善。

对于每个地图，我们会发布其一颗星、两颗星、三颗星、四颗星、五颗星对应的分值。

每次作业的 ddl：从发布之日起，到 6 月 9 日 23 点 59 分结束（暂定）。你可以在后面写出更好的策略后，重新提交以前的地图，进行“刷星”。

但与此同时，为了防止懒狗行为，若在该地图发布的一周内（到下个星期三的 23 点 59 分之前），若你在四个图中的 x 个图没有拿到一星的成绩，将会被扣除 $0.15 * x$ 分的总评（约为原来 OJ 一周内不交扣除 20% 分数）。

除了 $4*6=24$ 个公开地图以外，为了防止你过度地硬编码，我们保留有 8 个隐藏地图。该地图不公开，将于第五周左右发布测试。我们保证：隐藏地图不会涉及前四周未出现过的情况。

8.2 总评构成

项目的总评由以下方式构成：

实验报告（原则上不扣分）。提交方式及截止时间待通知。除非特殊情况，本次实验的实验报告不建议超过 2 页 A4 纸。请在实验报告中描述你在实验中遇到的特别值得一提的事件，例如你代码的架构设计、特别精巧的实现、遇到印象深刻的 bug 等。

程序运行情况。设 x 为你的程序在所有地图（公开地图 + 隐藏地图）上的总星数，在项目结束后，我们会根据整体情况进行赋分。例如：总计 160 星，80 星以上得满分，80 星以下线性下降（仅作为例子）。因此，我们允许你在部分地图跑的较差。最后取值将会参考我们所写的 simple demo 以及前三个学期 OJ 作业得分的分布。在期望情况下，你的得分不会劣于前三个学期 OJ 作业部分的得分的平均值。

若你每在所有的地图上都只有一颗星，则分数不超过总得分的 35%。

8.3 加分

项目结束时，我们会按所有关卡的总星数进行排序。若总星数相同，则依据总分数进行排序。第 1 名同学课程总评 +2 分，第 2~3 名同学课程总评 +1.5 分，第 4~6 名同学课程总评 +1 分。

课程总评的加分在最后统计完其他所有项（作业、期末等）得分后，会独立计算。不会因为别人有加分而导致你的分数降低。我们期望有兴趣、有能力的同学可以做的更加好一些，而不是为了加分而内卷。

对于 ACM 银牌及以上的 1 位同学，超过非 ACM 同学的第 6 名，即可课程总评 +1 分。

课程总分不溢出 100。

9 一些建议

实验的推进按周分任务进行，但总体来说只有两条主线：**更复杂的任务策略与更复杂的移动策略**。因此，可以考虑将决策和移动分为不同的两个部分实现，决策获取机器人的目标点后，移动部分给出朝向目标点的当前帧移动方案。

最开始选取的架构是值得深思的，也是需要费时的。例如：在设计第一周实验的时候，对于决策部分，只需考虑单一的不用加工的食材，以及对盘子的清洗。但是面对再往后的复杂需要加工组合食材，**你的代码是否具有可拓展性？**如果你没有想清大概的框架，你甚至可能需要在新的几周，面对新的功能时重构你的代码。若你选取了良好的框架，则可能仅需通过添加几十行甚至十几行，就可以应对新一周的任务。

写具有可读性的代码。这一点就不必多说了，当你的代码有几百行甚至上千行时，就会深刻感受到写可读代码的重要性了。

多写 `assert`。经历过 PA 的折磨，如果你还选着 jyy 的 OS，一定对这点有很多的感悟。

10 学术诚信

若仍有同学对学术诚信有疑问，请阅读 MIT 版的学术诚信。在项目结束后，我们会对所有同学的代码进行查重。若存在抄袭现象，则按以下规则进行处理：

- 第一次发现：课程总评扣 10%
- 第二次发现：课程总评扣 10%
- 第三次发现：课程总评不及格

查重的最后判断将由所有老师和助教共同进行，若被判定为抄袭，其结果仅作为通知，不接受任何解释。若对结果有异议，请向所在学院提出申诉。

若你的代码中，有极少量的参考，请在实验报告中给出，这将影响查重的判断结果。（如果存在大量相同，即使在实验报告中给出也无效）

在本学期项目结束之前，禁止将项目上传至任何公开仓库。否则，因此造成的抄袭，双方按照违反学术诚信处理。