上海交通大学安泰经济与管理学院

BUSS3620 人工智能导论

Project #1. 商业流程

刘佳璐 助理教授

商业流程

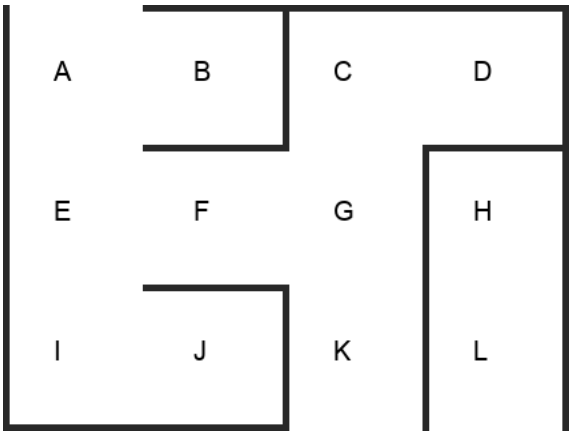
本项目将会制作一个优化仓库管理的 AI，旨在展示如何将课上所学知识运用到商业领域中。

```
$ python Warehouse.py Warehouse.txt A B,J
WareHouse:


█ ███████
█A B█C D█
█ ███ ███
█E F G█H█
█ ███ █ █
█I J█K█L█
█████ █ █
States Explored: 8
5 step solution: ([None, 'right', 'left', 'down', 'down', 'right'], ['A', 'B',
'A', 'E', 'I', 'J'], [(1, 1), (1, 3), (1, 1), (3, 1), (5, 1), (5, 3)])


█ ███████
█A B█C D█
█ ███ ███
█E F G█H█
█ ███ █ █
█I J█K█L█
█████ █ █
```

背景介绍

有一家线上零售公司希望借鉴京东的成功经验，自己建立自运营的仓库物流体系。这家公司的仓库每天收到来自各个省份的快递，需要分发至仓库所在省份的不同城市。为了更高效的完成分发工作，寄件地址相似的快递会储存在仓库中相同的位置。如下图所示，这个仓库一共有 12 个快递存放点，以 A 至 L 指代仓库的不同位置。

仓库平面图



在这次项目中，我们希望能够制作一个仓库 AI，他的主要功能是可以将多个快递用最短的路程放置到仓库中对应的位置。例如，假设仓库 AI 在 A 位置收到 2 个快递，这两个快递需要存放在仓库的 B 位置和 J 位置。那么从 A 到 B 和 J 的最短路线为：A → B → A → E → I → J。

开始

- 从课程中心平台 Canvas 上下 `Week3 搜索 I` 单元中的 `week3_project.zip` 并且解压缩
- 当处于本项目文件所在的工作目录中时，在终端上运行 `pip3 install -r requirements.txt` 用来安装这次项目需要的 Python 包（pillow 是为了安装 PIL 包）。

理解项目的相关文件

这个项目最主要的两个文件为：`Warehouse.txt` 和 `Warehouse.py`。

首先打开 `Warehouse.txt`。这个文件使用文本的形式展示上图的仓库平面图。其中，`#`代表墙壁，字母 `A-L` 代表仓库对应的位置。

其次，打开 `Warehouse.py`。在这个代码中，我们定义了一个类 `Warehouse`，是我们将要制作的仓库 AI，可以将多个快递用最短的路程送到仓库指定位置。在这个类中，`__init__`函数，`print` 函数，以及 `output_image` 函数已经为你写好了，无需变动。

`__init__`函数读取文本形式的仓库平面图。他记录了收快递的起点位置，以及快递需要储存的目标位置。他同时记录了墙壁的位置，并且在 `self.cell_to_letter` 中将字母位置 `A-Z` 和坐标位置`(i, j)`相对应。坐标位置`(i, j)` 中 `i` 是第几行（从 `0` 到 `height - 1`），`j` 是第几列（从 `0` 到 `width - 1`））。注意，墙壁的位置也是用`(i, j)`来表示。

`print` 函数可以将仓库平面图输出到终端窗口，其中起始位置为红色，路线经过的位置为绿色。

`output_image` 函数能够比较美观直接的画出仓库平面图，同时可以展示 AI 规划的路线以及已经探索过的位置。未来，同学们可能也需要将自己的代码结果，用一种易理解的方式展示给潜在听众。

剩余的两个函数，`neighbors` 和 `solve` 函数，将留给同学们你来完成。

`Week1_project.zip` 中还包含几个文件。`Arial.ttf` 是字体文件，用来绘制地图。`util.py` 文件提供了课上的 `Node`、 `StackFrontier` 和 `QueueFrontier` 相关代码。`autograde` 文件夹包含测试代码的相关文件。

要求

`neighbors` 函数

- 输入：一个坐标位置`(i, j)`
- 功能：给定一个坐标，找到这个位置能够到达的相邻位置以及到达该位置的行动（上下左右）。
  - 下一个可能的位置仅限于仓库中的 12 个位置（`A-L`），不可以是墙的坐标
  - 无法穿墙到达下一个位置。
    - 例如，`F` 的下一个可能位置仅有 `E` 和 `G`，而没有 `B` 和 `J`。
- 输出：一个列表，列表中每个元素是其中一个能够到达的位置，格式为元组`(action, cell)` 。`action` 是 `up, down, left, right` 中的其中一个方向。`cell` 是坐标位置`(i,j)`，其中 `i` 是第几行，`j` 是第几列。
  - 例如，`F` 的下一个可能位置是 `E` 和 `G`，所以会输出`[('left', (3, 1)), ('right', (3, 5))]`。

`solve` 函数

- 输入：无
- 功能：找到从起始位置到目标位置的最短路径，将其储存在类 `Warehouse` 中的 `self.solution`，并将已经搜索过多少个位置记录在 `self.num_explored`。
  - 假设最多两个快递（目标位置），不需考虑快递送达的前后顺序。(学有余力的同学可以考虑 3 个快递，或者 3 个快递以上的情况)
  - 如果有多个路径均为最短路径，那么其中任意路径均可。
  - 最短路径可以经过同一个位置多次。
    - 例如，假设起始位置是 `K`，目标位置是 `F` 和 `C`。那么你的最短路径解可以是 `K -> G -> F -> G-> C`，或者 `K -> G -> C -> G-> F`，这个 AI 可以经过位置 `G` 多次。
- 输出：
  - 如果从起始位置到目标位置无解，那么函数需要报错 `no solution`。
  - 假设从起始位置到目标位置有解，无需输出。只需找到最短路径并储存。最短路径的格式为`(actions, locations, cells)` ，其中 `actions` 是方向`(up, down, left, right)`列表，起始位置对应的 `actions` 为 `None`，`locations` 是字母位置（`A-L`）列表，`cells` 是坐标位置`(i, j)`列表。路径解以起始位置开始，需要含有所有的目标位置。
    - 例如，起始位置是 `A`，目标位置是 `B` 和 `J`，那么找到的最短路径解为`([None, 'right', 'left', 'down', 'down', 'right'], ['A', 'B', 'A', 'E', 'I', 'J'], [(1, 1), (1, 3), (1, 1), (3, 1), (5, 1), (5, 3)])`。

你可以借鉴参考我们课上的代码案例。我们已经在 `util.py` 中提供了课上的 `Node`、 `StackFrontier` 和 `QueueFrontier` 相关代码，你可以直接使用，你也可以修改他们，如果你修改，请在 `Warehouse.py` 中修改，请勿在 `util.py` 中修改。项目也不一定非要使用 `util.py` 中的代码。

你不应该修改 `Warehouse.py` 中除了 `neighbors` 和 `solve` 函数之外的已经写好的其他部分，但是你可以添加新的函数，或者使用 Python 中的一些标准的程序包。

测试代码

你可以使用代码 `check50 --d autograde -o ansi` 自行测试自己的代码是否满足要求。请先确保你的程序能够成功运行并输出结果。请确保你的工作目录中包含 `Warehouse.py`。您需要安装 `requirements.txt` 中的 `check50` 包。

特殊问题

有些同学终端中打出的地图起始点和终点不改变颜色，而是乱码。这种情况下，可以在 `Warehouse.py` 开头添加下面两行代码：

```
from colorama import init
init(autoreset=True)
```