**程序报告**

学号：2211290 姓名：姚知言

1. **问题重述**

分别使用基础搜索算法和 Deep QLearning 算法实现机器人，使机器人自动走到迷宫的出口。

游戏规则为从起点开始，通过错综复杂的迷宫，到达目标点 (出口)。在任一位置可执行动作包括：向上走’u’、向右走’r’、向下走’d’、向左走’l’。执行不同的动作后，根据不同的情况会获得不同的奖励，具体而言，有以下几种情况：撞墙、走到出口、其余情况。

基础搜索算法：选择深度优先算法进行实现。输入迷宫，输出到达目标点的路径。

QLearning：与策略迭代（Policy Iteration）算法不同，值迭代算法会计算每个”状态“或是”状态-动作“的值（Value）或是效用（Utility），然后在执行动作的时候，会设法最大化这个值。因此，对每个状态值的准确估计，是值迭代算法的核心。考虑最大化动作的长期奖励，即不仅考虑当前动作带来的奖励，还会考虑动作长远的奖励。

1. **设计思想**

在深度优先算法实现中，我使用栈的结构，但并没有完全照搬传统的深度优先策略。在访问一个新的结点的时候，我将栈顶弹出之后将这个栈的所有子结点都压入栈。这样做的好处是在保证深度优先算法顺序的前提下，无需频繁的自底向上频繁访问，也无需维护子结点是否已经访问的问题。虽然这样的话没有办法直接存储查找到的路径，但每个结点都存储了父结点的位置，相比于频繁调用的开销，我认为单次的自底向上的开销还是可以接受的。

在QLearning的实现中，通过控制alpha,gamma,epsilon的参数，调整maze.reward，重新定义update\_parameter，并在train\_update和test\_update中实现来实现效果较好的QLearning算法。在起初的实验中，常常无法通过高级算法，甚至中途在部分修改后，出现无法通过较低级算法的情况（比如当撞墙惩罚过小的时候，epsilon过大的时候）。最终经过无数次的调参与测试，终于顺利通过了所有的测试。

1. **代码内容**

#深度优先算法实现

class SearchTree(object):

def \_\_init\_\_(self, loc=(), action='', parent=None):

"""

初始化搜索树节点对象

:param loc: 新节点的机器人所处位置

:param action: 新节点的对应的移动方向

:param parent: 新节点的父辈节点

"""

self.loc = loc # 当前节点位置

self.to\_this\_action = action # 到达当前节点的动作

self.parent = parent # 当前节点的父节点

self.children = [] # 当前节点的子节点

def add\_child(self, child):

"""

添加子节点

:param child:待添加的子节点

"""

self.children.append(child)

def is\_leaf(self):

"""

判断当前节点是否是叶子节点

"""

return len(self.children) == 0

def expand(maze, is\_visit\_m, node):

"""

拓展叶子节点，即为当前的叶子节点添加执行合法动作后到达的子节点

:param maze: 迷宫对象

:param is\_visit\_m: 记录迷宫每个位置是否访问的矩阵

:param node: 待拓展的叶子节点

"""

can\_move = maze.can\_move\_actions(node.loc)

for a in can\_move:

new\_loc = tuple(node.loc[i] + move\_map[a][i] for i in range(2))

if not is\_visit\_m[new\_loc]:

child = SearchTree(loc=new\_loc, action=a, parent=node)

node.add\_child(child)

def back\_propagation(node):

"""

回溯并记录节点路径

:param node: 待回溯节点

:return: 回溯路径

"""

path = []

while node.parent is not None:

path.insert(0, node.to\_this\_action)

node = node.parent

return path

def my\_search(maze):

"""

选择深度优先算法实现

:param maze: 迷宫对象

:return :到达目标点的路径 如：["u","u","r",...]

"""

path = []

start = maze.sense\_robot()

root = SearchTree(loc=start)

stack = [root]

h, w, \_ = maze.maze\_data.shape

is\_visit\_m = np.zeros((h, w), dtype=np.int) # 标记迷宫的各个位置是否被访问过

while True:

current\_node = stack[0]

is\_visit\_m[current\_node.loc] = 1

if current\_node.loc == maze.destination:

path = back\_propagation(current\_node)

break

if current\_node.is\_leaf():

expand(maze, is\_visit\_m, current\_node)

stack.pop(0)

for child in current\_node.children:

stack.insert(0,child)

return path

====================================================================

#QLearning算法实现

from QRobot import QRobot

import random

class Robot(QRobot):

def \_\_init\_\_(self, maze,alpha=0.5,gamma=0.95,epsilon0=0.8):

"""

初始化 Robot 类

:param maze:迷宫对象

"""

super(Robot, self).\_\_init\_\_(maze)

self.maze = maze

self.maze.reward = {

"hit\_wall": -3,

"destination": 10,

"default": -0.1,

}

self.alpha = alpha

self.gamma = gamma

self.epsilon0 = epsilon0

self.epsilon = epsilon0

def update\_parameter(self):

"""

衰减随机选择动作的可能性

"""

self.t += 1

if self.epsilon < 0.03:

self.epsilon = 0.03

else:

self.epsilon -= self.t \* 0.1

return self.epsilon

def train\_update(self):

"""

以训练状态选择动作并更新Deep Q network的相关参数

:return :action, reward 如："u", -1

"""

self.state=self.sense\_state()

self.create\_Qtable\_line(self.state)

if(random.random()<self.epsilon):

action=random.choice(self.valid\_action)

else:

action=max(self.q\_table[self.state], key=self.q\_table[self.state].get)

reward = self.maze.move\_robot(action)

next\_state = self.sense\_state()

self.create\_Qtable\_line(next\_state)

self.update\_Qtable(reward, action, next\_state)

self.update\_parameter()

return action, reward

def test\_update(self):

"""

以测试状态选择动作并更新Deep Q network的相关参数

:return : action, reward 如："u", -1

"""

self.state = self.sense\_state()

self.create\_Qtable\_line(self.state)

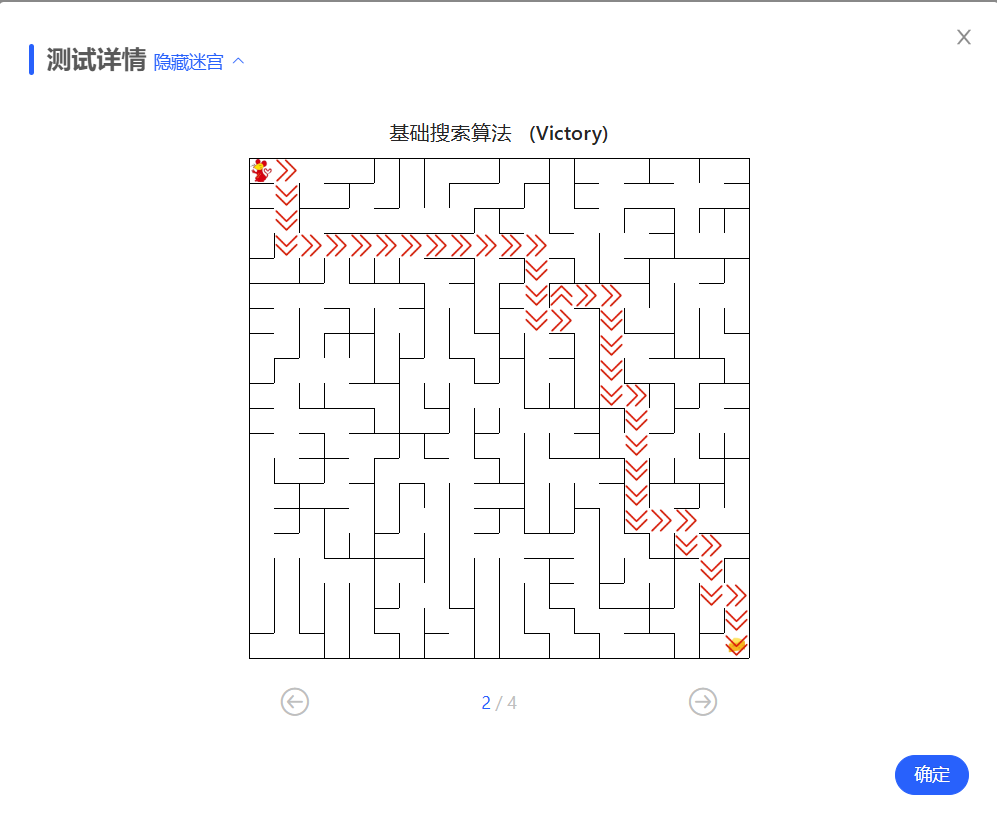
action = max(self.q\_table[self.state],key=self.q\_table[self.state].get)

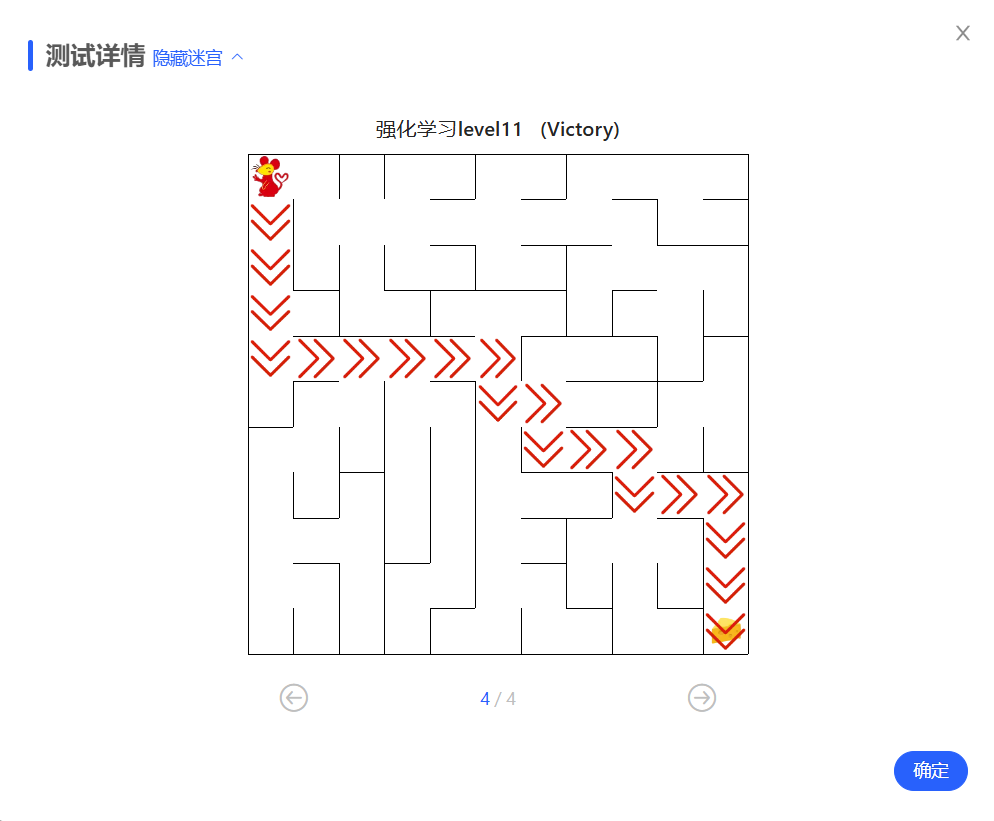
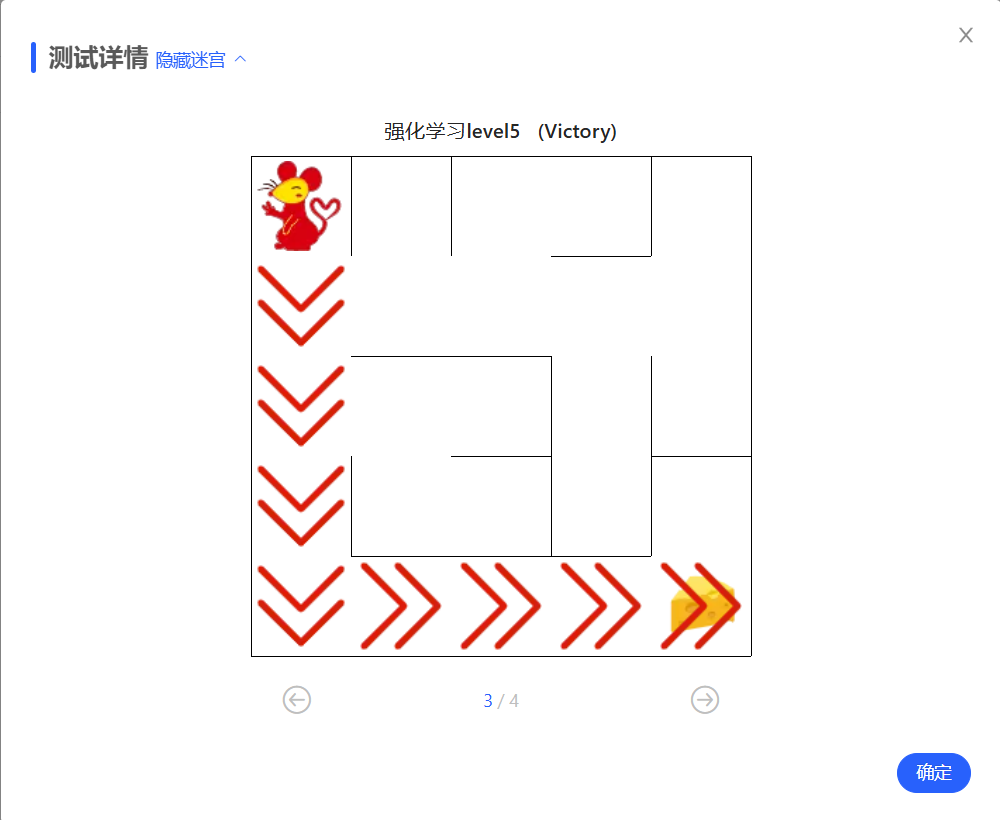
reward = self.maze.move\_robot(action)

return action, reward

1. **实验结果**

如图所示，所有迷宫成功完成，用时1s。







1. **总结**

在本次实验中，通过对机器人走迷宫模型的学习，完成了深度优先搜索算法，并学习了强化学习中的 Q-Learning算法，成功加以运用在实验中来。强化学习能够广泛应用于解

决实际生活中常见的决策问题，作为一种通用的策略学习框架，向人们展示了其强

大的能力和应用前景。

在这学期人工智能实验的学习中，我深刻了解了人工智能的各种算法，以及部分常用的模型框架。这为我入门人工智能领域打下了良好的基础，在未来我也会继续努力。