人工智能作业

学生姓名：董安宁

班 学 号：111171-05

指导教师：叶雅琴

中国地质大学信息工程学院

2020 年 3 月 1 日

目录

作业二 自动吃豆人

1、实现 MINIMAX

说白了这题就是要实现 MINIMAX，算法的伪代码如下(借用 PPT 的内容):

Minimax Algorithm 最小最大算法

```
function MINIMAX-DECISION(state) returns an action
  return argmax_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))

function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do v ← MAX(v, MIN-VALUE(RESULT(state, a)))
  return v

function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do v ← MIN(v, MAX-VALUE(RESULT(state, a)))
  return v
```

图 1 算法伪代码流程

算法的核心在于 MIN 和 MAX 的套娃调用，以及判断退出的条件，这也是我自己在实现的时候遇到的坎。

遇到的问题:

① MAX 和 MIN 相互调用的返回值是什么

由于算法本身只是计算每一层树的最佳状态，因此只需要进行值的传递就可以了，但在处理这个问题的时候，由于想着需要在得到最佳值的同时，也要得到最佳的行动(action)，因此我就在每次递归的时候以(value,action)的形式进行值的返回，但这样就导致递归的过程中值会变成这种形式：(((value,action) ,action) ,action),这样就会一来在求 max 和 min 的时候出现值比较的问题，二来在扒开括号的时候也会出现问题，因此在思考这种数据结构的时候想了很长时间。后来发现伪代码中还有 MINIMAX-DECISION 这个函数是专门处理我的这个疑惑的，也就是只在最外面一层返回 action，在下面别的层只需要返回值就可以了。

② 递归深度

在运行测试的时候发现自己递归的深度并没有达到预期深度，因此在这上面也花费了一些时间来思索。

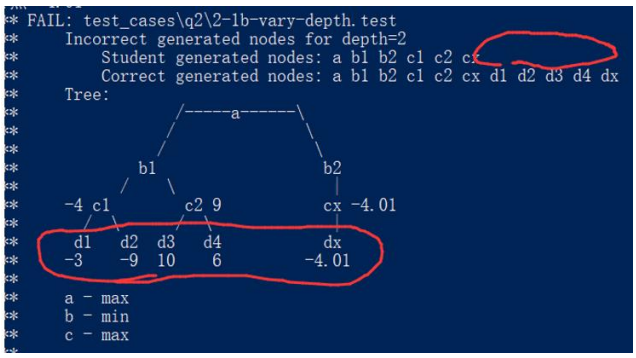

图 2 递归深度不够

③ 空节点的情况

当吃豆人四周都被包围住的时候，那么能够走得 action 列表就是空，因此需要对这种情况进行特殊处理，即直接返回该状态的值。

2、结果展示

其实解决完上述问题后，就可以通过所有的测试了，结果展示如下图所示：

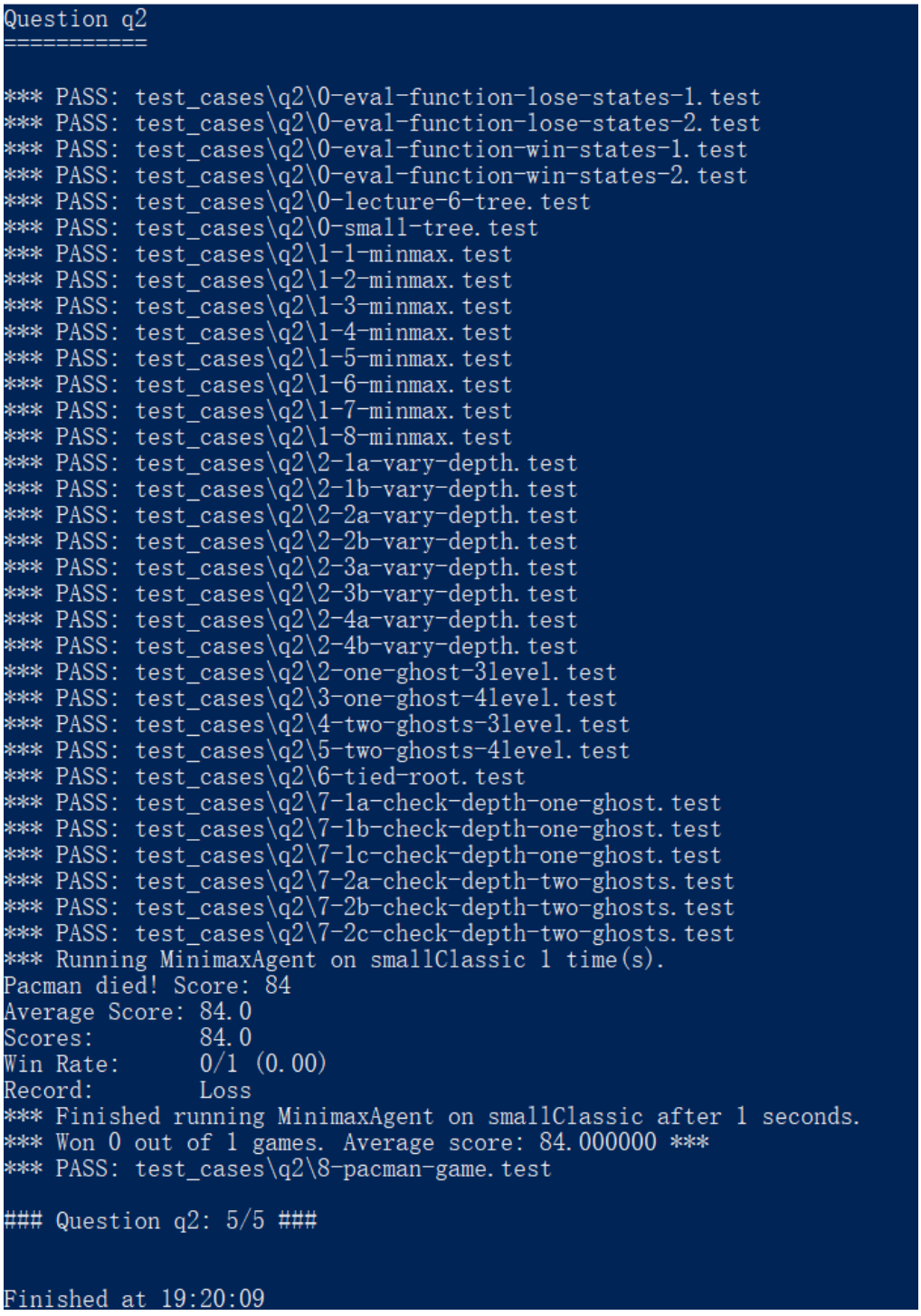

图 3 运行 python autograder.py -q q2 --no-graphics 的结果展示

最终吃豆人还是死了 hhh，不过和同学交流后发现都死了，并且得到的分数都是 84 分，因此应该可以说明是一个普遍性的情况（毕竟原文也说了可能会死）。

3、总结

本次实习着实花了我不少时间去完成，其实最开始打倒我的是心理暗示，我一直认为实现自动吃豆人很难，要考虑很多情况，因此有些畏惧。后来在实现的过程中发现，MINIMAX的算法设计是真的巧妙，将这个博弈的难题迎刃而解。在实现算法的时候我也和同学进行了很多的交流，我们之间没有交换代码，只是交换了思路，也在很多程度了给了我启发。代码我也添加了必要的注释，在伪代码的指导下能写出如此简洁的结构也是很令我感到惊奇的。学无止境呀。