AI Lab1

范翔宇 PB18000006

代码思路讲解

1.BFS

```
    visited = {}
    frontier = util.Queue()

    frontier.push((problem.getStartState(), None))

    while not frontier.isEmpty():
        state, prev_state = frontier.pop()

        if problem.isGoalState(state):
            solution = [state]
            while prev_state != None:
                solution.append(prev_state)
                prev_state = visited[prev_state]
            return solution[::-1]

        if state not in visited:
            visited[state] = prev_state

            for next_state, step_cost in problem.getChildren(state):
                frontier.push((next_state, state))
```

这部分代码其实很简单，跟DFS相比只是把**Stack**的数据结构更改为**Queue**。大致思路就是，按照入队列顺序处理节点，当处理某一具体节点n的时候，先判断n是不是目标节点，如果是则依次寻找父节点返回路径，如果不是则判断n之前是否处理过，没处理过则记录n的父节点，并且将n的子节点均入队列。整体的层次逻辑为先从左到右，再从上到下。(其实跟DFS相比真的只是改了一下数据结构)

2.A*

```
    # YOUR CODE HERE
    #openlist用优先队列表示
    openlist = util.PriorityQueue()
    #记录路径
    path = {}
    parent = {}
    F = {}
    #初始化 将起点加入open表
    start_state = problem.getStartState()
    parent[start_state] = None
    F[start_state] = heuristic(start_state) + 0 #g(n) = 0
    openlist.update(start_state,F[start_state])
    #开始循环
    while not openlist.isEmpty():
        current_state = openlist.pop()
        G = F[current_state] - heuristic(current_state)
```

```
        P = parent[current_state]
        #如果抵达目标状态
        if problem.isGoalState(current_state):
            #记录返回路径
            answer = [current_state]
            while P != None:
                answer.append(P)
                P = path[P]
            return answer[::-1]

        if current_state not in path:
            path[current_state] = P
            for next_state, step_cost in problem.getChildren(current_state):
                F_n = (heuristic(next_state) + step_cost + G)
                if next_state in F and F_n  >= F[next_state] :
                    continue
                else:
                    parent[next_state] = current_state
                    openlist.update(next_state, F_n)
                    F[next_state] = F_n
    if openlist.isEmpty():
        print("error!")
        util.raiseNotDefined()
```

排版问题，图在下一页~~

```
A* search {

closed list = [ ]
open list = [start node]

    do {
        if open list is empty then {
            return no solution
        }
        n = heuristic best node
        if n == final node then {
            return path from start to goal node
        }
        foreach direct available node do{
            if current node not in open and not in closed list do {
                add current node to open list and calculate heuristic
                set n as his parent node
            }
            else{
                check if path from star node to current node is
                better;
                if it is better calculate heuristics and transfer
                current node from closed list to open list
                set n as his parrent node
            }
        delete n from open list
        add n to closed list
    } while (open list is not empty)

}
```

大致思路参考这张图，但具体实现修改了部分细节，openlist采用**PriorityQueue**数据结构，用path记录路径，方便后续找到结果时返回，类似BFS用父子节点之间的关系链接，F[n]表示从初始节点由节点n到目标节点的代价估计。循环之前要进行初始化，父节点为空，此时G，即从初始节点到节点n的实际代价，为0，故F即为H(H意义为从节点n到目标节点的最佳路径的估计代价)，并且更新openlist。之后当openlist不为空时，要进行如下循环：用G表示初始节点到当前状态的实际代价，先判断是否抵达目标状态，如果抵达目标状态，则沿着path返回路径；如果不是则判断是否在path中(在的话说明之前处理过，没必要处理了，直接skip)，不在的话进行如下操作：计算初始节点通过当前节点到该子节点再到目标节点的代价估计F_{new}，更新记录到path，遍历当前状态的每个子节点，如果子节点还不在F中或者已经在F中但是F_{new}更优，那么我们就记录/更新子节点的父节点为当前节点，并且更新openlist和F值。

3.MinMax

```
        #personal add here
        # YOUR CODE HERE
        #pacman - MAX
        Flag = state.isMe()
        if Flag == 1:
            depth = depth - 1
            if depth == -1:
```

```
                #不能继续递归就返回
                return None, state.evaluateScore()
            for successor in state.getChildren():
                # YOUR CODE HERE
                #调用minmax递归
                tmp, result = self.minimax(successor, depth)
                #pacman对应max
                best_score = max(best_score, result)
                #如果当前result更优
                if best_score == result:
                    best_state = successor
        else:
        #Ghost - MIN
            if depth == -1:
                #同样不能继续递归就返回
                return None, state.evaluateScore()
            for successor in state.getChildren():
                # YOUR CODE HERE
                #调用minmax递归
                tmp, result = self.minimax(successor, depth)
                #ghost对应min
                best_score = min(best_score, result)
                #如果当前result更优
                if best_score == result:
                    best_state = successor

        return best_state, best_score
```

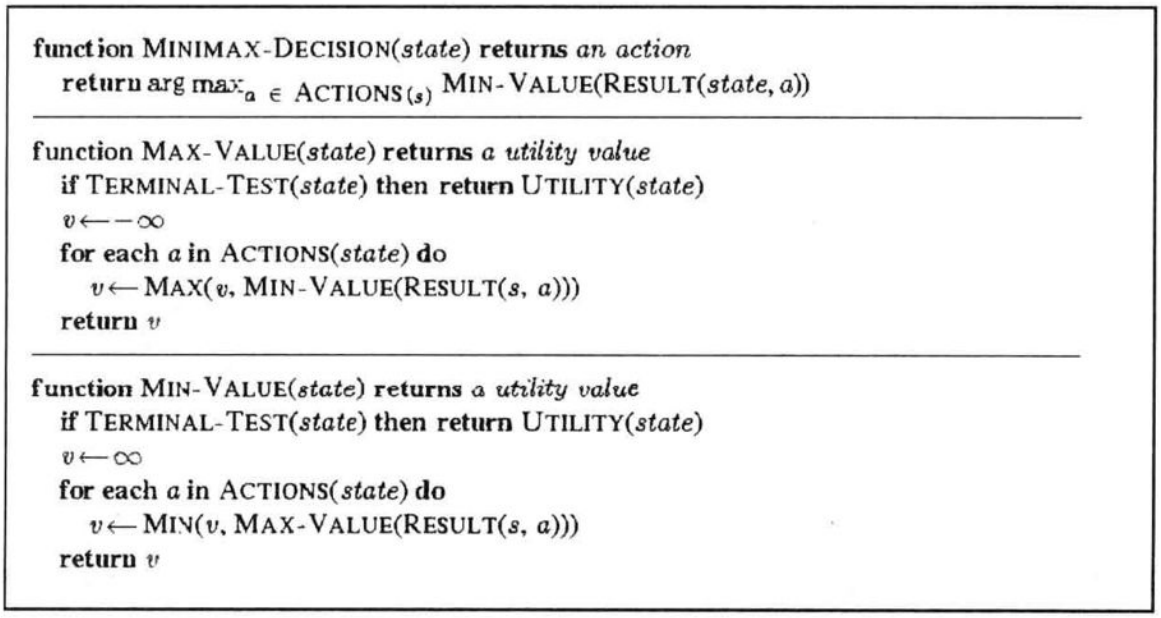

图 5.3　极小极大值决策算法

大致思路参考这张图，当然实现细节要自己补充。结合文档中“值得注意的是算法搜索的深度depth，它指的是每个agent所走的步数。例如depth=2，有1个pacman和2个ghost，则从搜索树的最顶层到最底层应该经过pacman->ghost1->ghost2->pacman->ghost1->ghost2，操作应该为max->min->min->max->min->min”。大致是一个递归算法，首先判断当前是否为目标agent在进行操作。如果是pacman，那么递归深度-1，并且判断当前深度是否为-1，如果为-1，说明不能继续递归即返回，否则，遍历当前节点的每一个后继(即子节点)，是调用minmax得到最后的评估值result，然后用max跟当前最优的分数进行比较，如果result更优那么更新最优状态为当前子节点。如果是ghost操作，那么递归深度不减，因为depth代表pacman所走步数，同样需要遍历当前节点的每一个子节点，调用minmax得到最后的评估值result，但是是用min对result和最优分数进行比较并更新。

4.AlphaBeta

```
    #add here
    #还是仿照minmax样式写比较好，方便递归
    def AlphaBeta(self, state, Alpha, Beta, depth):
        if state.isTerminated():
            return None, state.evaluateScore()

        best_state, best_score = None, -float('inf') if state.isMe() else
float('inf')


        Flag = state.isMe()
        if Flag:
            #pacman - MAX
            depth = depth - 1
            if depth == -1:
                #不能递归就返回
                return None, state.evaluateScore()
            for successor in state.getChildren():
                # YOUR CODE HERE
                tmp, result = self.AlphaBeta(successor, Alpha, Beta, depth)
                #pacman对应max
                best_score = max(best_score, result)
                #判断是否更优
                if best_score == result:
                    best_state = successor
                #如果大于β就返回结果
                if best_score > Beta:
                    return best_state, best_score
                #否则更新α
                Alpha = max(Alpha, best_score)

        else:
            #ghost - MIN
            if depth == -1:
                #不能递归就返回
                return None, state.evaluateScore()
            for successor in state.getChildren():
                # YOUR CODE HERE
                tmp, result = self.AlphaBeta(successor, Alpha, Beta, depth)
                #ghost对应min
                best_score = min(best_score, result)
                #判断是否更优
                if best_score == result:
                    best_state = successor
                #如果小于α就返回结果
                if best_score < Alpha:
                    return best_state, best_score
                #否则更新β
                Beta = min(Beta, best_score)


        return best_state, best_score

    def getNextState(self, state):
```

```
        best_state, best_score = self.AlphaBeta(state, -float('inf'),
+float('inf'), self.depth)
        return best_state
        #util.raiseNotDefined()
```

```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) 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(s,a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v

function MIN-VALUE(state, α, β) 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(s,a), α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

图 5.7　α-β搜索算法

这里并没有按照助教指示修改的部分修改，而是增加一个函数，因为毕竟是一个递归函数。整体部分其实是参考MinMax进行修改的，大致步骤类似，判断当前是否为pacman在操作，然后分别遍历子节点调用递归得到result，同样pacman对应max、ghost对应min，来与当前最优分数进行比较来决定是否更新最优节点。具体步骤可以参考MinMax(因为我觉得我在那里已经讲述的很详细了QAQ)。当然还是有与MinMax不同的地方：递归调用函数为AlphaBeta，而且传参不止传successor和depth，还要传alpha和beta，另外在判断是否更新最优节点之后，还要将最优分数与alpha/beta比较，如果小于/大于则返回最优节点与分数，否则同样利用max/min更新alpha和beta。大致与书上逻辑类似，但是有一点需要注意的是，书上的v与alpha/beta比较时用的是 $\leq$/ $\geq$，但是我们实际上比较时，不应该取那个等号，如果取了等号，会出现类似下面的错误，B由于也等于10，直接return了，而没有遍历到C。

```
*** FAIL: test_cases/q3/6-tied-root.test
***     Incorrect move for depth=3
***         Student move: Right
***         Optimal move: Left
***     Incorrect generated nodes for depth=3
***         Student generated nodes: A B max min1 min2
***         Correct generated nodes: A B C max min1 min2
***     Tree:
***           max
***          /   \
***     min1      min2
***      |        /  \
***      A       B    C
***     10      10    0
```

结果展示

```
(ustc-ai) fofo@ubuntu:~/桌面/AILab/LAB1$ ./test.sh
Starting on 5-23 at 20:20:31

Question q1
===========
*** PASS: test_cases/q1/graph_backtrack.test
***     solution:               ['1:A->C', '0:C->G']
***     expanded_states:        ['A', 'D', 'C']
*** PASS: test_cases/q1/graph_bfs_vs_dfs.test
***     solution:               ['2:A->D', '0:D->G']
***     expanded_states:        ['A', 'D']
*** PASS: test_cases/q1/graph_infinite.test
***     solution:               ['0:A->B', '1:B->C', '1:C->G']
***     expanded_states:        ['A', 'B', 'C']
*** PASS: test_cases/q1/graph_manypaths.test
***     solution:               ['2:A->B2', '0:B2->C', '0:C->D', '2:D->E2', '0:E2->F', '0:F->G']
***     expanded_states:        ['A', 'B2', 'C', 'D', 'E2', 'F']
*** PASS: test_cases/q1/pacman_1.test
***     pacman layout:          mediumMaze
***     solution length: 130
***     nodes expanded:         146

### Question q1: 4/4 ###


Finished at 20:20:31

Provisional grades
==================
Question q1: 4/4
------------------
Total: 4/4

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.

[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 130 in 0.0 seconds
Search nodes expanded: 146
Pacman emerges victorious! Score: 380
Average Score: 380.0
Scores:        380.0
Win Rate:      1/1 (1.00)
Record:        Win
Starting on 5-23 at 20:20:33

Question q2
===========
*** PASS: test_cases/q2/graph_backtrack.test
***     solution:               ['1:A->C', '0:C->G']
***     expanded_states:        ['A', 'B', 'C', 'D']
*** PASS: test_cases/q2/graph_bfs_vs_dfs.test
***     solution:               ['1:A->G']
```

```
Question q2
===========
*** PASS: test_cases/q2/graph_backtrack.test
***     solution:               ['1:A->C', '0:C->G']
***     expanded_states:        ['A', 'B', 'C', 'D']
*** PASS: test_cases/q2/graph_bfs_vs_dfs.test
***     solution:               ['1:A->G']
***     expanded_states:        ['A', 'B']
*** PASS: test_cases/q2/graph_infinite.test
***     solution:               ['0:A->B', '1:B->C', '1:C->G']
***     expanded_states:        ['A', 'B', 'C']
*** PASS: test_cases/q2/graph_manypaths.test
***     solution:               ['1:A->C', '0:C->D', '1:D->F', '0:F->G']
***     expanded_states:        ['A', 'B1', 'C', 'B2', 'D', 'E1', 'F', 'E2']
*** PASS: test_cases/q2/pacman_1.test
***     pacman layout:          mediumMaze
***     solution length: 68
***     nodes expanded:         269

### Question q2: 4/4 ###


Finished at 20:20:33

Provisional grades
==================
Question q2: 4/4
------------------
Total: 4/4

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.

[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 269
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win
Starting on 5-23 at 20:20:35

Question q3
===========
*** PASS: test_cases/q3/astar_0.test
***     solution:               ['Right', 'Down', 'Down']
***     expanded_states:        ['A', 'B', 'D', 'C', 'G']
*** PASS: test_cases/q3/astar_1_graph_heuristic.test
***     solution:               ['0', '0', '2']
***     expanded_states:        ['S', 'A', 'D', 'C']
*** PASS: test_cases/q3/astar_2_manhattan.test
***     pacman layout:          mediumMaze
***     solution length: 68
```

```
Question q3
===========
*** PASS: test_cases/q3/astar_0.test
***     solution:               ['Right', 'Down', 'Down']
***     expanded_states:        ['A', 'B', 'D', 'C', 'G']
*** PASS: test_cases/q3/astar_1_graph_heuristic.test
***     solution:               ['0', '0', '2']
***     expanded_states:        ['S', 'A', 'D', 'C']
*** PASS: test_cases/q3/astar_2_manhattan.test
***     pacman layout:          mediumMaze
***     solution length: 68
***     nodes expanded:         221
*** PASS: test_cases/q3/astar_3_goalAtDequeue.test
***     solution:               ['1:A->B', '0:B->C', '0:C->G']
***     expanded_states:        ['A', 'B', 'C']
*** PASS: test_cases/q3/graph_backtrack.test
***     solution:               ['1:A->C', '0:C->G']
***     expanded_states:        ['A', 'B', 'C', 'D']
*** PASS: test_cases/q3/graph_manypaths.test
***     solution:               ['1:A->C', '0:C->D', '1:D->F', '0:F->G']
***     expanded_states:        ['A', 'B1', 'C', 'B2', 'D', 'E1', 'F', 'E2']

### Question q3: 4/4 ###


Finished at 20:20:35

Provisional grades
==================
Question q3: 4/4
------------------
Total: 4/4

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.

[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 221
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win
Starting on 5-23 at 20:20:36

Question q2
===========

*** PASS: test_cases/q2/0-eval-function-lose-states-1.test
*** PASS: test_cases/q2/0-eval-function-lose-states-2.test
*** PASS: test_cases/q2/0-eval-function-win-states-1.test
*** PASS: test_cases/q2/0-eval-function-win-states-2.test
*** PASS: test_cases/q2/0-lecture-6-tree.test
```

```
Question q2
===========

*** PASS: test_cases/q2/0-eval-function-lose-states-1.test
*** PASS: test_cases/q2/0-eval-function-lose-states-2.test
*** PASS: test_cases/q2/0-eval-function-win-states-1.test
*** PASS: test_cases/q2/0-eval-function-win-states-2.test
*** PASS: test_cases/q2/0-lecture-6-tree.test
*** PASS: test_cases/q2/0-small-tree.test
*** PASS: test_cases/q2/1-1-minmax.test
*** PASS: test_cases/q2/1-2-minmax.test
*** PASS: test_cases/q2/1-3-minmax.test
*** PASS: test_cases/q2/1-4-minmax.test
*** PASS: test_cases/q2/1-5-minmax.test
*** PASS: test_cases/q2/1-6-minmax.test
*** PASS: test_cases/q2/1-7-minmax.test
*** PASS: test_cases/q2/1-8-minmax.test
*** PASS: test_cases/q2/2-1a-vary-depth.test
*** PASS: test_cases/q2/2-1b-vary-depth.test
*** PASS: test_cases/q2/2-2a-vary-depth.test
*** PASS: test_cases/q2/2-2b-vary-depth.test
*** PASS: test_cases/q2/2-3a-vary-depth.test
*** PASS: test_cases/q2/2-3b-vary-depth.test
*** PASS: test_cases/q2/2-4a-vary-depth.test
*** PASS: test_cases/q2/2-4b-vary-depth.test
*** PASS: test_cases/q2/2-one-ghost-3level.test
*** PASS: test_cases/q2/3-one-ghost-4level.test
*** PASS: test_cases/q2/4-two-ghosts-3level.test
*** PASS: test_cases/q2/5-two-ghosts-4level.test
*** PASS: test_cases/q2/6-tied-root.test
*** PASS: test_cases/q2/7-1a-check-depth-one-ghost.test
*** PASS: test_cases/q2/7-1b-check-depth-one-ghost.test
*** PASS: test_cases/q2/7-1c-check-depth-one-ghost.test
*** PASS: test_cases/q2/7-2a-check-depth-two-ghosts.test
*** PASS: test_cases/q2/7-2b-check-depth-two-ghosts.test
*** PASS: test_cases/q2/7-2c-check-depth-two-ghosts.test
*** Running MinimaxAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running MinimaxAgent on smallClassic after 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases/q2/8-pacman-game.test

### Question q2: 5/5 ###


Finished at 20:20:37

Provisional grades
==================
Question q2: 5/5
------------------
```

```
Total: 5/5

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.

Starting on 5-23 at 20:20:37

Question q3
===========

*** PASS: test_cases/q3/0-eval-function-lose-states-1.test
*** PASS: test_cases/q3/0-eval-function-lose-states-2.test
*** PASS: test_cases/q3/0-eval-function-win-states-1.test
*** PASS: test_cases/q3/0-eval-function-win-states-2.test
*** PASS: test_cases/q3/0-lecture-6-tree.test
*** PASS: test_cases/q3/0-small-tree.test
*** PASS: test_cases/q3/1-1-minmax.test
*** PASS: test_cases/q3/1-2-minmax.test
*** PASS: test_cases/q3/1-3-minmax.test
*** PASS: test_cases/q3/1-4-minmax.test
*** PASS: test_cases/q3/1-5-minmax.test
*** PASS: test_cases/q3/1-6-minmax.test
*** PASS: test_cases/q3/1-7-minmax.test
*** PASS: test_cases/q3/1-8-minmax.test
*** PASS: test_cases/q3/2-1a-vary-depth.test
*** PASS: test_cases/q3/2-1b-vary-depth.test
*** PASS: test_cases/q3/2-2a-vary-depth.test
*** PASS: test_cases/q3/2-2b-vary-depth.test
*** PASS: test_cases/q3/2-3a-vary-depth.test
*** PASS: test_cases/q3/2-3b-vary-depth.test
*** PASS: test_cases/q3/2-4a-vary-depth.test
*** PASS: test_cases/q3/2-4b-vary-depth.test
*** PASS: test_cases/q3/2-one-ghost-3level.test
*** PASS: test_cases/q3/3-one-ghost-4level.test
*** PASS: test_cases/q3/4-two-ghosts-3level.test
*** PASS: test_cases/q3/5-two-ghosts-4level.test
*** PASS: test_cases/q3/6-tied-root.test
*** PASS: test_cases/q3/7-1a-check-depth-one-ghost.test
*** PASS: test_cases/q3/7-1b-check-depth-one-ghost.test
*** PASS: test_cases/q3/7-1c-check-depth-one-ghost.test
*** PASS: test_cases/q3/7-2a-check-depth-two-ghosts.test
*** PASS: test_cases/q3/7-2b-check-depth-two-ghosts.test
*** PASS: test_cases/q3/7-2c-check-depth-two-ghosts.test
*** Running AlphaBetaAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running AlphaBetaAgent on smallClassic after 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases/q3/8-pacman-game.test

### Question q3: 5/5 ###
```

```
*** PASS: test_cases/q3/1-3-minmax.test
*** PASS: test_cases/q3/1-4-minmax.test
*** PASS: test_cases/q3/1-5-minmax.test
*** PASS: test_cases/q3/1-6-minmax.test
*** PASS: test_cases/q3/1-7-minmax.test
*** PASS: test_cases/q3/1-8-minmax.test
*** PASS: test_cases/q3/2-1a-vary-depth.test
*** PASS: test_cases/q3/2-1b-vary-depth.test
*** PASS: test_cases/q3/2-2a-vary-depth.test
*** PASS: test_cases/q3/2-2b-vary-depth.test
*** PASS: test_cases/q3/2-3a-vary-depth.test
*** PASS: test_cases/q3/2-3b-vary-depth.test
*** PASS: test_cases/q3/2-4a-vary-depth.test
*** PASS: test_cases/q3/2-4b-vary-depth.test
*** PASS: test_cases/q3/2-one-ghost-3level.test
*** PASS: test_cases/q3/3-one-ghost-4level.test
*** PASS: test_cases/q3/4-two-ghosts-3level.test
*** PASS: test_cases/q3/5-two-ghosts-4level.test
*** PASS: test_cases/q3/6-tied-root.test
*** PASS: test_cases/q3/7-1a-check-depth-one-ghost.test
*** PASS: test_cases/q3/7-1b-check-depth-one-ghost.test
*** PASS: test_cases/q3/7-1c-check-depth-one-ghost.test
*** PASS: test_cases/q3/7-2a-check-depth-two-ghosts.test
*** PASS: test_cases/q3/7-2b-check-depth-two-ghosts.test
*** PASS: test_cases/q3/7-2c-check-depth-two-ghosts.test
*** Running AlphaBetaAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running AlphaBetaAgent on smallClassic after 0 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases/q3/8-pacman-game.test

### Question q3: 5/5 ###


Finished at 20:20:38

Provisional grades
==================
Question q3: 5/5
------------------
Total: 5/5

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.

Pacman emerges victorious! Score: 1889
Average Score: 1889.0
Scores:        1889.0
Win Rate:      1/1 (1.00)
Record:        Win
(ustc-ai) fofo@ubuntu:~/桌面/AILab/LAB1$
```

可以看到，全部pass!

pacman yyds