人工智能基础 实验一

PB18151866 龚小航

【实验目的】

在给出的代码框架上完成吃豆人游戏的决策过程，需要完成静态目标下的广度优先搜索与 A^* 搜索算法实现寻找食物，在此基础上更进一步在多 agent 环境中实现 MaxMin 算法与 $\alpha-\beta$剪枝，吃完所有食物同时避开鬼。

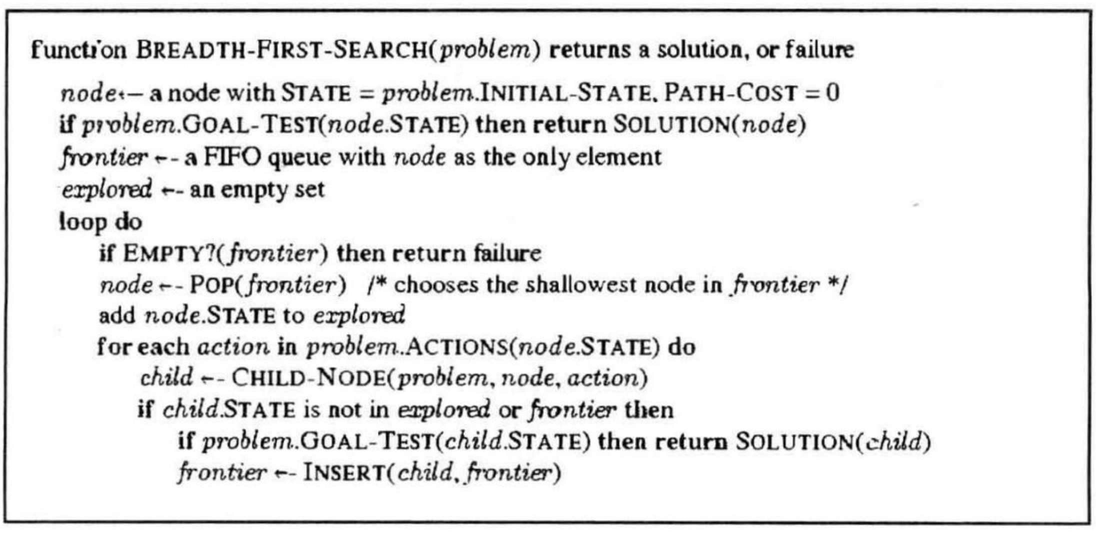
【实验内容】

本次实验有 2 个部分，分别是 Search 和 Multiagent。具体而言，Search 的目标是吃豆人仅仅是寻找食物；Multiagent 的目标是吃完所有食物，同时避开鬼。抽象而言，Search 实现的静态查找算法，Multiagent 的问题是在有对手的情况下做出下一步决策使自己的利益最大化。Search 部分需要实现 BFS 算法和 A*算法。Multiagent 部分需要实现 minimax 算法和 alpha-beta 剪枝。

【实验过程】

一 、BFS 算法实现静态搜索食物。只需将给出的深度优先搜索代码中的栈结构改为普通队列结构。

伪代码如下所示：



```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
    node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
    if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
    frontier ← a FIFO queue with node as the only element
    explored ← an empty set
    loop do
        if EMPTY?(frontier) then return failure
        node ← POP(frontier)   /* chooses the shallowest node in frontier */
        add node.STATE to explored
        for each action in problem.ACTIONS(node.STATE) do
            child ← CHILD-NODE(problem, node, action)
            if child.STATE is not in explored or frontier then
                if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
                frontier ← INSERT(child, frontier)
```

图 3.11 图的宽度优先搜索

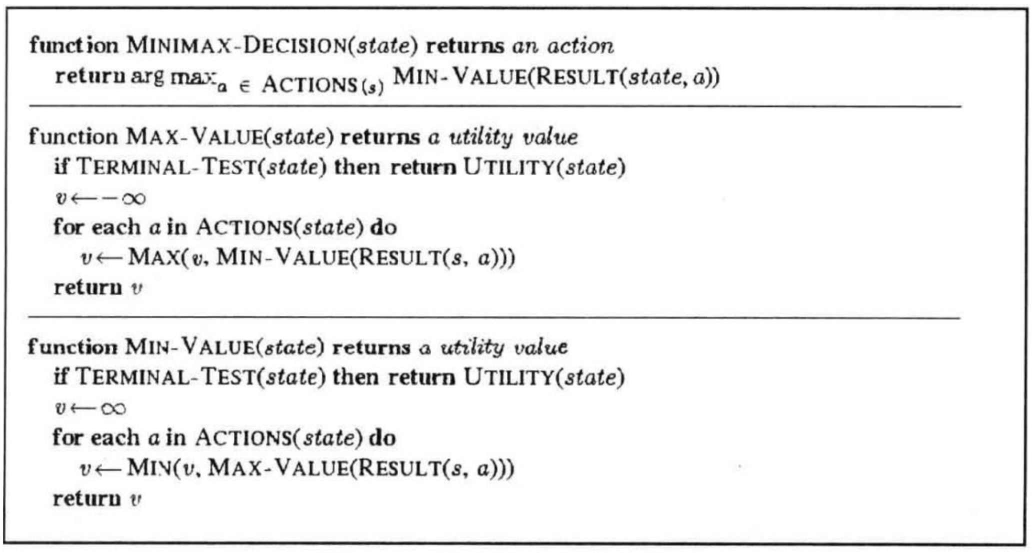
将其写为 python 代码：

```python
def myBreadthFirstSearch(problem):
    # YOUR CODE HERE
    visited = {}    #存储访问过的节点
    frontier = util.Queue() #队列结构存储待访问节点

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

    while not frontier.isEmpty():#还有待访问节点时继续
        #取出下一步访问的节点, 判断是否为目标节点, 返回或扩展
        state, prev_state = frontier.pop()

        if problem.isGoalState(state): #找到目标节点, 返回
            solution = [state]
            while prev_state != None:
                solution.append(prev_state)
                prev_state = visited[prev_state]
            return solution[::-1]

        if state not in visited:    #在BFS下, 拓展该节点
            visited[state] = prev_state
            for next_state, step_cost in problem.getChildren(state):
                frontier.push((next_state, state))
    return []
```

二、实现 A^* 静态搜索算法。和BFS类似，存储待访问节点的数据结构为优先队列，权值为g_n+h_n，g 存储从起始节点开始到当前待访问而未访问节点的实际消耗，在将某个节点插入优先队列的时刻生成。从优先队列取出数据时每个节点都已经有g_n域。

```python
def myAStarSearch(problem, heuristic): #预估函数 h 已经从外部给出, 调用即可
    # YOUR CODE HERE
    visited = {} #已访问节点列表
    frontier = util.PriorityQueue() #优先队列存储待访问节点
    frontier.__init__()
    #优先队列结构: 同时插入状态与权值, 权值为 g+h, 初始节点 g=0
    frontier.push(((problem.getStartState(),0), (None,0)),
                                         0 + heuristic(problem.getStartState()))

    while not frontier.isEmpty():
        (state,g), (prev_state,prev_g) = frontier.pop()
        #与 BFS 类似, 从优先队列中取出下个待访问的节点, 判断是否为目标, 返回或扩展它
        if problem.isGoalState(state):
            solution = [state]
            while prev_state != None:
                solution.append(prev_state)
                prev_state = visited[prev_state]
            return solution[::-1]

        #扩展节点, 同时将 gn 值赋予即将插入优先队列的待访问节点
        if state not in visited:
            visited[state] = prev_state
            for next_state, step_cost in problem.getChildren(state):
                next_g = g + step_cost
                frontier.push(((next_state,next_g), (state,g)),
                                        next_g + heuristic(next_state))
    return []
```

三、极小极大算法实现多 agent 环境中的动态决策。伪代码如下所示：



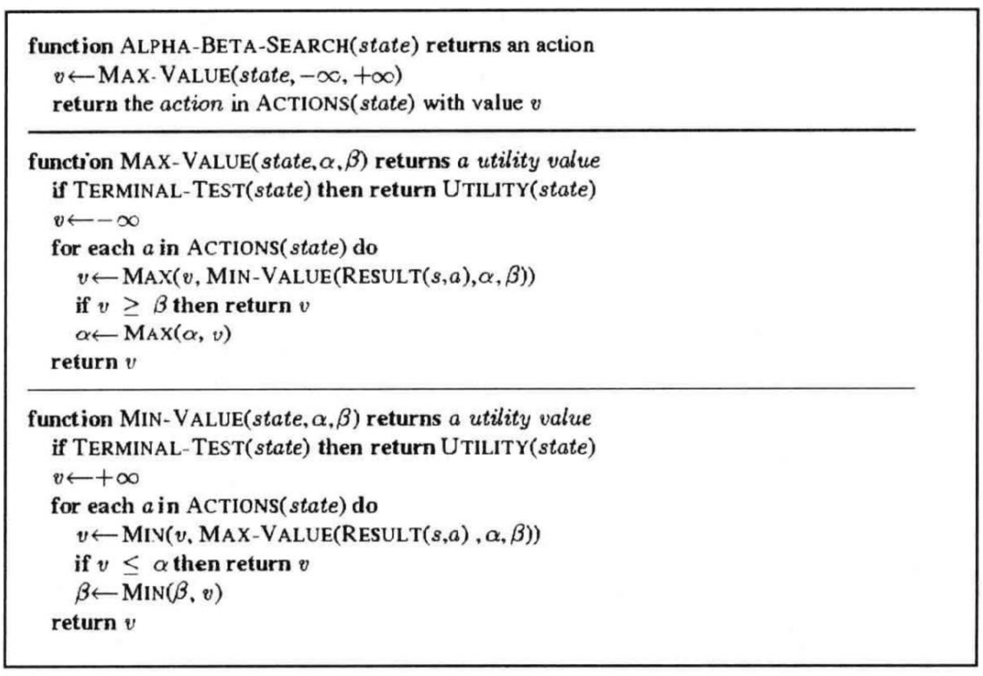
```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))

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(s, 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(s, a)))
  return v
```

图 5.3　极小极大值决策算法

本次实验中还需要实现同时返回下一步最佳的状态，只需要在 child 中找出 max 或 min 的同时将对应的状态记录下来即可。代码如下所示：

```
class MyMinimaxAgent():

    def __init__(self, depth):
        self.depth = depth

    #用于选择使用 MAX 方案还是 MIN 方案
    def minimax(self, state, depth):
        if state.isMe():
            return self.maxv(state,depth)
        return self.minv(state,depth)

    #MAX 节点操作
    def maxv(self,state,depth):
        beststate = None
        bestv = -float('inf')
        if state.isTerminated():
            return None, state.evaluateScore()
        if depth == 0:
            return state, state.evaluateScore()

        for child in state.getChildren():#MAX 行动一次 depth-1
            st,sc = self.minimax(child, depth-1)
            if sc >= bestv:
                bestv = sc
                beststate = child
        return beststate,bestv

    #MIN 节点的操作
    def minv(self,state,depth):
        beststate = None
        bestv = float('inf')
        if state.isTerminated():
            return None, state.evaluateScore()
        for child in state.getChildren():#MIN 行动一次 depth 不变
            st,sc = self.minimax(child, depth)
            if sc <= bestv:
                bestv = sc
                beststate = child
        return beststate,bestv

    #外部调用的接口，输入当前状态返回下一步的最佳行动状态
    def getNextState(self, state):
        best_state, _ = self.minimax(state, self.depth)
        return best_state
```

四、在极小极大算法的基础上实现 $\alpha-\beta$剪枝。类似 MAXMIN 算法，在其上增加每个节点的上下限 $\alpha-\beta$。伪代码实现如下所示：



```
function ALPHA-BETA-SEARCH(state) returns an action
  $v \leftarrow$ MAX-VALUE(state, $-\infty$, $+\infty$)
  return the action in ACTIONS(state) with value $v$

function MAX-VALUE(state, $\alpha$, $\beta$) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  $v \leftarrow -\infty$
  for each a in ACTIONS(state) do
    $v \leftarrow$ MAX($v$, MIN-VALUE(RESULT(s,a), $\alpha$, $\beta$))
    if $v \geq \beta$ then return $v$
    $\alpha \leftarrow$ MAX($\alpha$, $v$)
  return $v$

function MIN-VALUE(state, $\alpha$, $\beta$) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  $v \leftarrow +\infty$
  for each a in ACTIONS(state) do
    $v \leftarrow$ MIN($v$, MAX-VALUE(RESULT(s,a) , $\alpha$, $\beta$))
    if $v \leq \alpha$ then return $v$
    $\beta \leftarrow$ MIN($\beta$, $v$)
  return $v$
```

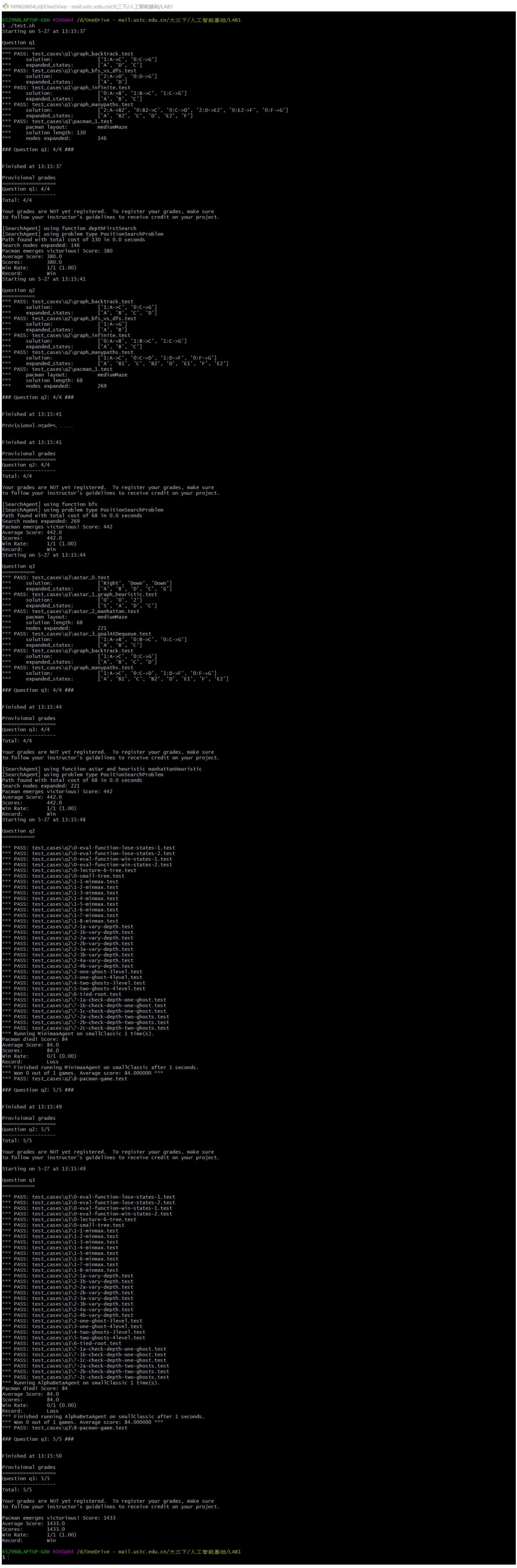
图 5.7 α-β搜索算法

将其实现为 python 代码，如下所示：

```python
class MyAlphaBetaAgent():

    def __init__(self, depth):
        self.depth = depth

    #外部调用接口，输入当前状态返回最佳后继状态
    def getNextState(self, state):
        # YOUR CODE HERE
        best_state, _ = self.alpha_beta_cut(state, self.depth,
                                            -float('inf'), float('inf'))
        return best_state

    #用于选择调用 MAX 还是 MIN 方法
    def alpha_beta_cut(self, state, depth, alpha, beta):
        if state.isMe():
            return self.alpha_beta_cutmaxv(state,depth, alpha, beta)
        return self.alpha_beta_cutminv(state,depth, alpha, beta)

    def alpha_beta_cutmaxv(self,state,depth, alpha, beta):
        beststate = None
        bestv = -float('inf')
        if state.isTerminated():
            return None, state.evaluateScore()
        if depth == 0:
            return state, state.evaluateScore()
        for child in state.getChildren():
            st,sc = self.alpha_beta_cut(child, depth-1, alpha, beta)
            if sc > bestv:
                bestv = sc
                beststate = child
            if bestv > beta:
                return beststate,bestv
            alpha = max(alpha, bestv)
        return beststate,bestv

    def alpha_beta_cutminv(self,state,depth, alpha, beta):
        beststate = None
        bestv = float('inf')
        if state.isTerminated():
            return None, state.evaluateScore()
        for child in state.getChildren():
            st,sc = self.alpha_beta_cut(child, depth, alpha, beta)
            if sc < bestv:
                bestv = sc
                beststate = child
            if bestv < alpha:
                return beststate,bestv
            beta = min(beta, bestv)
        return beststate,bestv
```

【实验结果】

运行 test.sh 测试文件，得到的结果如下所示。各测试结果均为 PASS，符合预期。



MINGW64:/d/OneDrive - mail.ustc.edu.cn/大三下/人工智能基础/LAB1

```
65296@LAPTOP-GXH MINGW64 /d/OneDrive - mail.ustc.edu.cn/大三下/人工智能基础/LAB1
$ ./test.sh
Starting on 5-27 at 13:15:37

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 13:15:37

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 5-27 at 13:15:41

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 13:15:41

Provisional grades


Finished at 13:15:41

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 5-27 at 13:15:44

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 13:15:44

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 5-27 at 13:15:48

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 1 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 13:15:49

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 5-27 at 13:15:49

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 1 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 13:15:50

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: 1433
Average Score: 1433.0
Scores:        1433.0
Win Rate:      1/1 (1.00)
Record:        Win

65296@LAPTOP-GXH MINGW64 /d/OneDrive - mail.ustc.edu.cn/大三下/人工智能基础/LAB1
$
```