哈爾濱工業大學

# 人工智能实验报告

题　　目　搜索策略

专　　业　计算机科学与技术学院

学　　号　1160300814

学　　生　姜思琪

指 导 教 师　李钦策

同 组 人 员　陈曦、谢将凤、冯传恒、丁明泽

# 一. 简介/问题描述

## 1.1 待解决问题的解释

实验 2 要求采用且不限于课程第四章内各种搜索算法，编写一系列吃豆人程序解决下面列出的 8 个问题。

**问题 1**：应用深度优先算法找到一个特定的位置的豆

本问题要求使用深度优先搜索给出吃豆人的路径，还要求写出一个完整的通用搜索算法。

**问题 2**：宽度优先算法

利用宽度优先算法实现解决以上问题。

**问题 3**：代价一致算法

很多情况下，路径中的代价是可以改变的。完成代价一致搜索方法(search.py 文件中的 uniformCostSearch 函数)。

**问题 4**：A* 算法

完成 A*搜索方法(search.py 文件中的 aStarSearch 函数)，利用曼哈顿距离作为启发函数。

**问题 5**：找到所有的角落

在角落迷宫的四个角上面有四个豆。这个搜索问题要求找到一条访问所有四个角落的最短的路径。完成 searchAgents.py 文件中的 CornersProblem 搜索问题，你需要重新定义状态，使其能够表示角落是否被访问。

**问题 6**：角落问题（启发式）

构建合适的启发函数，完成 searchAgents.py 文件中的 cornersHeuristic 角落搜索问题。

**问题 7**：吃掉所有的豆子

用尽可能少的步数吃掉所有的豆子。完成 searchAgents.py 文件中的 FoodSearchProblem 豆子搜索问题。

**问题 8**：次最优搜索

定义一个优先吃最近的豆子函数是提高搜索速度的一个好的办法。补充完成 searchAgents.py 文件中的 AnyFoodSearchProblem 目

标测试函数，并完成 searchAgents.py 文件中的 ClosestDotSearchAgent 部分，在此 Agent 当中缺少一个关键的函数：找到最近豆子的函数。

## 二. 算法介绍

2.1 所用方法的一般介绍以及解决方案（原理）

**问题 1：** 应用深度优先算法找到一个特定的位置的豆

通用的搜索算法：根据实验要求附录中所给出的算法伪代码，可以比较容易地写出实验要求的完整的通用搜索算法。使用了(state,actions)的二元组来表示每个节点，其中 state 为状态，对于问题 1 至问题 4，即为吃豆人所在的坐标(coord)，而对于之后的问题，除坐标之外还含其他参数。而 actions 为从初始结点到达本状态所要执行的操作序列，actions 即为保存构建路径的信息需要的结构，其即是路径信息本身。

深度优先搜索：首先扩展的是最深的结点。有了通用的搜索算法，实现深度优先搜索算法只需指定参数 data_struct_type 为实验提供栈结构 util.Stack 即可。

**问题 2：** 宽度优先算法

宽度优先算法是按层扩展的，遵循先生成的节点先扩展的策略，从初始节点 S0 开始，逐层地对节点进行扩展并考察它是否为目标节点，在第 n 层的节点没有全部扩展并考察之前，不对第 n＋1 层的节点进行扩展。将通用搜索算法中的 data_struct_type 设置为 util.Queue 即可（队列结构）。

**问题 3：** 代价一致算法

可以用 g(n)表示从初始节点 S0 到节点 n1 的代价，用 c(n1,n2)表示从父节点 n1 到 n2 的代价。这样，对节点 n2 的代价有：g(n2)=g(n1)+c(n1,n2)。

在代价一致搜索算法中，把从起始节点 S0 到任一节点 i 的路径代价记为 g(i)。从初始节点 S0 开始扩展，若没有得到目标节点，则优先扩展最少代价 g(i)的节点，一直如此向下搜索，所以我们选

择优先队列作为 open 表的数据结构。将通用搜索算法中的 data_struct_type 设置为 util.PriorityQueue，并将 usePriorityQueue 设置为 True，表示使用优先级队列。

**问题 4：** A* 算法

如果以 g(n)表示从起点到任意顶点 n 的实际距离，h(n)表示任意顶点 n 到目标顶点的估算距离，那么 A* 算法的估算函数为：f(n)=g(n)+h(n)。在 A* 搜索问题中，选择优先队列作为 open 表的数据结构。

将通用搜索算法的 data_struct_type 设置为 util.PriorityQueue,usePriorityQueue 设置为 True, heuristic 设置为 aStarSearch 传入的 heuristic。当执行 autograder.py 时，会将 manhattanHeuristic 传入 aStarSearch 的 heuristic，即可实现以曼哈顿距离为启发函数的 A*算法。

**问题 5：** 找到所有的角落

把状态 state 表示为一个形如(coord, foods_statebool)的二元组,其中 coord 为吃豆人所在的位置坐标(x,y)。foods_statebool 为四个角落食物的状态列表，初始值为[False, False, False, False], foods_statebool 与 self.corners((1,1), (1,top), (right,1),(right,top))位置一一对应,若对应位置的食物未被吃掉，则 foods_statebool 中对应项为 False,若已经被吃则 foods_statebool 中对应项为 True。getSuccessors 函数考虑生成的后继节点，若后继节点即是未到达过的角落位置，则在该后继的 foods_statebool 中，要将对应的位置置为 True，表示已经到达过把该角落的食物吃掉了。

**问题 6：**角落问题（启发式）

选择在无墙的 迷宫中从某位置到达目标状态的最小代价作为启发式函数值，定根据剩余角落的个数分类讨论。

当剩余 0 个食物时，说明已到达目标状态，这时应返回 0;当剩余 1 个食物时，最小代价即为当前坐标与剩余角落的曼哈顿距离;

当剩余 2 个食物时，应先到达距离当前结点较近的剩余角落（曼哈顿距离意义上的较近，下同），然后从这一角落到达另一剩余的角落。启发式函数值记为这两段路程的曼哈顿距离之和;当剩余食物个数为 3 时，启发式函数值取当前位置到较近的边缘点的曼哈顿距离加上一倍的迷宫长和宽;当剩余 4 个食物时，启发函数值也就等于吃豆人到距离其最近的角落的 曼哈顿距离加两倍的迷宫宽和一倍的迷宫长。

**问题 7**：吃掉所有的豆子

启发式函数值即为吃豆人与距离其最远的剩余食物之间的距离，因为 A* 算法总是选择 f 值最小的节点进行扩展，更倾向于扩展到剩余食物的中心位置，使节点距离剩余食物中的每一个都不至于很远。其次，这种启发式函数值选取方法还可以防止对某个食物节点的临近位置的过度探索，不会被局限在局部。最后定性地考虑一下，永远把离当前位置最远的那个豆子与当前位置之间的距离作为当前位置的启发式距离，也就是可以看做把最远的豆子作为目标点，这是合理的，因为肯定是吃完当前位置附近的豆子，最后再吃最远的豆子。

在课程已经给出的代码中，给出了一个 mazeDistance(point1, point2,gameState)函数，可计算在迷宫中从一个点到达另一个点所需要的真实最小代价。使用此距离函数可比使用曼哈顿距离函数产生一个更紧的下确界。

**问题 8**：次最优搜索

不断的使用 findPathToClosestDot 寻找到最近的食物的路径并将该路径加入到总路径之中，直到所有的食物均被吃完。因而在 isGoalState 函数中，要定义将吃掉一个食物作为目标状态，以使 findPathToClosestDot 函数在吃到最近的食物后结束并返回 action 列表。

在 findPathToClosestDot 函数中，使用广度优先搜索算法进行搜索。用已知的 PositionSearchProblem 可以拿到满分。

2.2 算法伪代码

**问题 1：** 应用深度优先算法找到一个特定的位置的豆

通用的搜索算法：

```
classGenericSearch:
    def__init__(self,problem,data_struct_type,usePriorityQu
eue=False, heuristic=nullHeuristic):
        self.problem=problem
        self.data_struct_type=data_struct_type
        self.usePriorityQueue=usePriorityQueue
        self.heuristic=heuristic
    defgenericSearch(self):
        closed <- an empty set
        fringe <- INSERT(MAKE-NODE(INITIAL-
        STATE[problem]),fringe)
        loop do
            if fringe is empty then return failure
            node <- REMOVE-FRONT(fringe)
            if GOAL-TEST( problem, STATE[node]) then return
        node
            if STATE[node] is not in closed then
                add STATE[ node] to closed
                for child-node in EXPAND(STAtE[node], problem)
            do
                fringe + INSERr( child-node, fringe)
            end
    end
```

深度优先算法：

```
    defdepthFirstSearch(problem):
        dfs=GenericSearch(problem,util.Stack)
```

```
        returndfs.genericSearch()
```

**问题 2：** 宽度优先算法

```
defbreadthFirstSearch(problem):
        bfs=GenericSearch(problem,util.Queue)
        returnbfs.genericSearch()
```

**问题 3：** 代价一致算法

```
defuniformCostSearch(problem):
    ucs=GenericSearch(problem,util.PriorityQueue,True)
    returnucs.genericSearch()
```

**问题 4：** A* 算法

```
defaStarSearch(problem,heuristic=nullHeuristic):
    astar=GenericSearch(problem,util.PriorityQueue,True,heu
ristic)
    returnastar.genericSearch()
```

**问题 5：** 找到所有的角落

```
defisGoalState(self,state):
returnstate[1][0]andstate[1][1]andstate[1][2]andstate[1][3]
defgetSuccessors(self,state):
        CONVERSATION foodBool TO foodXYList
        For action IN [Directions.NORTH,Directions.SOUTH,
        Directions.EAST, Directions.WEST]:
            dx,dy <-Actions.directionToVector(action)
            nextx,nexty<- int(x+dx),int(y+dy)
            hitsWall <-self.walls[nextx][nexty]
            IF NOT hitsWall:
                nextState <- (nextx,nexty)
                leftFoodBool<- ShallowCopy(foodBool)
                IF nextState IN foodXYList:
                leftFoodBool[self.corners.index((nextx,nexty
```

```
        ))] <- True
        successors.append(((nextState,leftFoodBool),action,
    1))
        self._expanded+=1
        returnsuccessors
```

**问题 6**：角落问题（启发式）

```
defcornersHeuristic(state,problem):
    IFfood_num==0:
        return0
    ELIFfood_num==1:
        return ManhattanDistanceOf CurrentLocation and
LeftfoodLocation
    ELIFfood_num==2:
        mindist<- min(ManhattanDistanceOf CurrentLocation
and LeftfoodLocations)
        return mindist Plus ManhattanDistanceOfLeftfood
Locations
    ELIFfood_num==3:
        leftedgefood<- GETtwoEdgesfoodLocation
    man_dist_lst<- min(ManhattanDistanceOf CurrentLocation
and LeftEdgesfoodLocations)
        returnman_dist_lstPlusMazeWidthSub2PlusMazeLengthth
Sub2
    ELIFfood_num==4:
   man_dist_lst <- min(ManhattanDistanceOf CurrentLocation
and LeftfoodLocations)
        return man_dist_lst Plus 2*(MazeWidth Sub 2) Plus
MazeLengthth Sub2
```

**问题 7**：吃掉所有的豆子

```
Class AStarFoodSearchAgent(SearchAgent):
    def __init__(self):
      self.searchFunction = lambda prob: search.aStarSearch(prob, foodHeuristic)
      self.searchType = FoodSearchProblem
```

**问题 8：次最优搜索**

```
defisGoalState(self,state):
  x,y=state
  returnself.food[x][y]
deffindPathToClosestDot(self,gameState):
    For x IN RANGE(food.width):
       For y IN RANGE(food.height):
          IF food.data[x][y]==True:
             dis <- manhattan(startPosition,(x,y))
             if dis<mindis:
                mindis=dis
                xindex=x
                yindex=y
 assertnotwalls[xindex][yindex],'point['+xindex+']['+yindex+']isawall.'
 prob=PositionSearchProblem(gameState,start=startPosition,goal=(xindex,yindex),warn=False,visualize=False)
 ans <- search.bfs(prob)
```

```
return ans
```

# 三. 算法实现

## 3.1 实验环境与问题规模

Python2。

## 3.2 数据结构

在深度优先搜索中 open 表是栈结构，最先进入的节点排在最后面，最后进入的节点排最前面，即先进后出。

宽度优先搜索的概念，open 表中的节点总是按进入的先后顺序排列，先进入的节点排在前面，后进入的排在后面，是一种队列结构，即先进先出。

代价一致算法和 A*算法使用优先级队列。

## 3.3 实验结果

## 3.4 系统中间及最终输出结果（要求有屏幕显示）

问题一：

```
D:\人工智能\lab1lab2\lab2>py -2 autograder.py
Starting on 10-29 at 22:11:16

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: 3/3 ###
```

问题二：

```
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: 3/3 ###
```

问题三：

```
Question q3
===========

*** 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_bfs_vs_dfs.test
***     solution:               ['1:A->G']
***     expanded_states:        ['A', 'B']
*** PASS: test_cases\q3\graph_infinite.test
***     solution:               ['0:A->B', '1:B->C', '1:C->G']
***     expanded_states:        ['A', 'B', 'C']
*** 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']
*** PASS: test_cases\q3\ucs_0_graph.test
***     solution:               ['Right', 'Down', 'Down']
***     expanded_states:        ['A', 'B', 'D', 'C', 'G']
*** PASS: test_cases\q3\ucs_1_problemC.test
***     pacman layout:          mediumMaze
***     solution length: 68
***     nodes expanded:         269
*** PASS: test_cases\q3\ucs_2_problemE.test
***     pacman layout:          mediumMaze
***     solution length: 74
***     nodes expanded:         260
*** PASS: test_cases\q3\ucs_3_problemW.test
***     pacman layout:          mediumMaze
***     solution length: 152
***     nodes expanded:         173
*** PASS: test_cases\q3\ucs_4_testSearch.test
***     pacman layout:          testSearch
***     solution length: 7
***     nodes expanded:         14
*** PASS: test_cases\q3\ucs_5_goalAtDequeue.test
***     solution:               ['1:A->B', '0:B->C', '0:C->G']
***     expanded_states:        ['A', 'B', 'C']

### Question q3: 3/3 ###
```

问题四：

```
Question q4
===========

*** PASS: test_cases\q4\astar_0.test
***     solution:               ['Right', 'Down', 'Down']
***     expanded_states:        ['A', 'B', 'D', 'C', 'G']
*** PASS: test_cases\q4\astar_1_graph_heuristic.test
***     solution:               ['0', '0', '2']
***     expanded_states:        ['S', 'A', 'D', 'C']
*** PASS: test_cases\q4\astar_2_manhattan.test
***     pacman layout:          mediumMaze
***     solution length: 68
***     nodes expanded:         221
*** PASS: test_cases\q4\astar_3_goalAtDequeue.test
***     solution:               ['1:A->B', '0:B->C', '0:C->G']
***     expanded_states:        ['A', 'B', 'C']
*** PASS: test_cases\q4\graph_backtrack.test
***     solution:               ['1:A->C', '0:C->G']
***     expanded_states:        ['A', 'B', 'C', 'D']
*** PASS: test_cases\q4\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 q4: 3/3 ###
```

问题五：

```
Question q5
===========

*** PASS: test_cases\q5\corner_tiny_corner.test
***     pacman layout:          tinyCorner
***     solution length:                28

### Question q5: 3/3 ###
```

问题六：

```
Question q6
===========

*** PASS: heuristic value less than true cost at start state
*** PASS: heuristic value less than true cost at start state
*** PASS: heuristic value less than true cost at start state
path: ['North', 'East', 'East', 'East', 'East', 'North', 'North', 'West', 'West', 'West',
'West', 'North', 'North', 'North', 'North', 'North', 'North', 'North', 'North', 'West', 'W
est', 'West', 'West', 'South', 'South', 'East', 'East', 'East', 'East', 'South', 'South',
'South', 'South', 'South', 'South', 'West', 'West', 'South', 'South', 'South', 'West', 'We
st', 'East', 'East', 'North', 'North', 'North', 'East', 'East', 'East', 'East', 'East', 'E
ast', 'East', 'East', 'South', 'South', 'East', 'East', 'East', 'East', 'East', 'North', '
North', 'East', 'East', 'North', 'North', 'East', 'East', 'North', 'North', 'East', 'East'
, 'East', 'East', 'South', 'South', 'South', 'South', 'East', 'East', 'North', 'North', 'E
ast', 'East', 'South', 'South', 'South', 'South', 'South', 'North', 'North', 'North', 'Nor
th', 'North', 'North', 'North', 'West', 'West', 'North', 'North', 'East', 'East', 'North',
 'North']
path length: 106
*** PASS: Heuristic resulted in expansion of 741 nodes

### Question q6: 3/3 ###
```

问题七：

```
Question q7
===========

*** PASS: test_cases\q7\food_heuristic_1.test
*** PASS: test_cases\q7\food_heuristic_10.test
*** PASS: test_cases\q7\food_heuristic_11.test
*** PASS: test_cases\q7\food_heuristic_12.test
*** PASS: test_cases\q7\food_heuristic_13.test
*** PASS: test_cases\q7\food_heuristic_14.test
*** PASS: test_cases\q7\food_heuristic_15.test
*** PASS: test_cases\q7\food_heuristic_16.test
*** PASS: test_cases\q7\food_heuristic_17.test
*** PASS: test_cases\q7\food_heuristic_2.test
*** PASS: test_cases\q7\food_heuristic_3.test
*** PASS: test_cases\q7\food_heuristic_4.test
*** PASS: test_cases\q7\food_heuristic_5.test
*** PASS: test_cases\q7\food_heuristic_6.test
*** PASS: test_cases\q7\food_heuristic_7.test
*** PASS: test_cases\q7\food_heuristic_8.test
*** PASS: test_cases\q7\food_heuristic_9.test
*** PASS: test_cases\q7\food_heuristic_grade_tricky.test
***     expanded nodes: 4137
***     thresholds: [15000, 12000, 9000, 7000]

### Question q7: 5/4 ###
```

问题八：

命令提示符

```
Question q8
===========

[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
*** PASS: test_cases\q8\closest_dot_1.test
***     pacman layout:          Test 1
***     solution length:                1
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
*** PASS: test_cases\q8\closest_dot_10.test
***     pacman layout:          Test 10
***     solution length:                1
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
*** PASS: test_cases\q8\closest_dot_11.test
***     pacman layout:          Test 11
***     solution length:                2
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
*** PASS: test_cases\q8\closest_dot_12.test
***     pacman layout:          Test 12
***     solution length:                3
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
*** PASS: test_cases\q8\closest_dot_13.test
***     pacman layout:          Test 13
***     solution length:                1
[SearchAgent] using function depthFirstSearch
```

```
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
*** PASS: test_cases\q8\closest_dot_2.test
***     pacman layout:          Test 2
***     solution length:                1
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
*** PASS: test_cases\q8\closest_dot_3.test
***     pacman layout:          Test 3
***     solution length:                1
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
*** PASS: test_cases\q8\closest_dot_4.test
***     pacman layout:          Test 4
***     solution length:                3
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
*** PASS: test_cases\q8\closest_dot_5.test
***     pacman layout:          Test 5
***     solution length:                1
```

```
***     pacman layout:          Test 5
***     solution length:                1
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
*** PASS: test_cases\q8\closest_dot_6.test
***     pacman layout:          Test 6
***     solution length:                2
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
*** PASS: test_cases\q8\closest_dot_7.test
***     pacman layout:          Test 7
***     solution length:                1
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
*** PASS: test_cases\q8\closest_dot_8.test
***     pacman layout:          Test 8
***     solution length:                1
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
*** PASS: test_cases\q8\closest_dot_9.test
***     pacman layout:          Test 9
***     solution length:                1

### Question q8: 3/3 ###
```

总成绩：

```
Finished at 22:11:49

Provisional grades
==================
Question q1: 3/3
Question q2: 3/3
Question q3: 3/3
Question q4: 3/3
Question q5: 3/3
Question q6: 3/3
Question q7: 5/4
Question q8: 3/3
------------------
Total: 26/25

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

## 四. 总结及讨论、

通过本次实验，我知道了要多思考，多钻研，多证明。一种思路可以解决问题，但不一定是最优的。要经过严谨的思维逻辑证明，才能判断是否是最优的。