doc

2022 年 3 月 31 日

软件 03 班陈启乾 2020012385

1 实验一：让吃豆人吃到一个食物

[10]:

```
# 小迷宫 + dfs
%run pacman.py -l tinyMaze -p SearchAgent -a fn=depthFirstSearch
```

```
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 9 in 0.001 seconds
Search nodes expanded: 15
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win
```

[9]:

```
# 中等迷宫 + BFS
%run pacman.py -l mediumMaze -p SearchAgent -a fn=breadthFirstSearch
```

```
[SearchAgent] using function breadthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.016 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
```

[8]:

```
# 中等迷宫 + UCS
%run pacman.py -l mediumMaze -p SearchAgent -a fn=uniformCostSearch
```

```
[SearchAgent] using function uniformCostSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.018 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
```

[7]:

```
# 大迷宫 + A*
%run pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=aStarSearch,heuristic=manhattanHeuristic
```

```
[SearchAgent] using function aStarSearch and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.031 seconds
Search nodes expanded: 549
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
```

1.1 实现解读

1.1.1 DFS

dfs 实现在 search.py 的 depthFirstSearch 中。

dfs 使用递归的实现，在递归变量中记录当前 pos ，返回值中记录 action 的序列。

1.1.2 BFS

bfs 实现在 search.py 的 breadthFirstSearch 中。

bfs 采用一个 FIFO 队列实现，每次从队列头中取出元素，然后将扩展的元素加入队列尾，action 的序列在“状态”中维护。

1.1.3 UCS

ucs 实现在 search.py 的 uniformCostSearch 中。

ucs 采用一个优先队列实现，每次从队列头取出元素，然后将还未从队列中取出扩展的元素加入队列，代价函数是给出的，这里是恒为 1 的函数。

1.1.4 A*

A* 搜索实现在 search.py 的 aStarSearch 中。

A* 的启发函数，直接使用了曼哈顿距离。即为 $d = |x_1 - x_2| + |y_1 - y_2|$ 。

容易发现这个启发函数是良定义的。

1. 满足启发值小于真实值：至少需要走 d 步才可能抵达终点。
2. 三角形不等式：曼哈顿距离显然拥有三角形不等式。

1.2 表现评估

```
[1]: %run pacman.py -l maze_gen_500 -p SearchAgent -a fn=breadthFirstSearch -q
```

```
[SearchAgent] using function breadthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 2796 in 0.355 seconds
Search nodes expanded: 3416
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作
业\lab1-search\pacman.py in <module>
    674
    675     args = readCommand( sys.argv[1:] ) # Get game components based on input
--> 676     runGames( **args )
    677 
    678     # import cProfile

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\pacman.py in runGames(layout, pacman, ghosts, display, numGames, record, numTraining, catchExceptions, timeout, save)
    632                 rules.quiet = False
    633             game = rules.newGame( layout, pacman, ghosts, gameDisplay, beQuiet, catchExceptions)
--> 634             game.run()
    635             if not beQuiet: games.append(game)
    636 

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\game.py in run(self)
    676                         return
    677                 else:
--> 678                     self.state = self.state.generateChild( agentIndex, action )
    679 
    680                 # Change the display

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\pacman.py in generateChild(self, agentIndex, action)
    107             state.data._agentMoved = agentIndex
    108             state.data.score += state.data.scoreChange
--> 109             GameState.explored.add(self)
    110             GameState.explored.add(state)
    111             return state

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\pacman.py in __hash__(self)
    228             Allows states to be keys of dictionaries.
    229             """
```

```
--> 230         return hash( self.data )
    231
    232     def __str__( self ):

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\game
 ↪py in __hash__(self)
    409                 print(e)
    410                 #hash(state)
--> 411         return int((hash(tuple(self.agentStates)) + 13*hash(self.food) +␣
 ↪113* hash(tuple(self.capsules)) + 7 * hash(self.score)) % 1048575 )
    412
    413     def __str__( self ):

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\game
 ↪py in __hash__(self)
    179                 if i:
    180                     h += base
--> 181                 base *= 2
    182         return hash(h)
    183

KeyboardInterrupt:
```

[11]:
```
%run pacman.py -l maze_gen_500 -p SearchAgent -a fn=depthFirstSearch -q
```

```
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
```

[2]:
```
%run pacman.py -l maze_gen_500 -p SearchAgent -a␣
 ↪fn=aStarSearch,heuristic=manhattanHeuristic -q
```

```
[SearchAgent] using function aStarSearch and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 2796 in 0.362 seconds
Search nodes expanded: 3368
```

```
---------------------------------------------------------------------------
```

```
KeyboardInterrupt                         Traceback (most recent call last)
d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作
业\lab1-search\pacman.py in <module>
    674
    675     args = readCommand( sys.argv[1:] ) # Get game components based on␣
 ↪input
--> 676     runGames( **args )
    677
    678     # import cProfile

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作
业\lab1-search\pacman.py in runGames(layout, pacman, ghosts, display, numGames,␣
 ↪record, numTraining, catchExceptions, timeout, save)
    632             rules.quiet = False
    633         game = rules.newGame( layout, pacman, ghosts, gameDisplay,␣
 ↪beQuiet, catchExceptions)
--> 634         game.run()
    635         if not beQuiet: games.append(game)
    636

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\game
 ↪py in run(self)
    676                     return
    677             else:
--> 678                 self.state = self.state.generateChild( agentIndex,␣
 ↪action )
    679
    680             # Change the display

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作
业\lab1-search\pacman.py in generateChild(self, agentIndex, action)
    108         state.data.score += state.data.scoreChange
    109         GameState.explored.add(self)
--> 110         GameState.explored.add(state)
    111         return state
    112
```

```
d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作
业\lab1-search\pacman.py in __hash__(self)
    228         Allows states to be keys of dictionaries.
    229         """
--> 230         return hash( self.data )
    231 
    232     def __str__( self ):

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\game
 ↪py in __hash__(self)
    409                 print(e)
    410                 #hash(state)
--> 411         return int((hash(tuple(self.agentStates)) + 13*hash(self.food) +
 ↪113* hash(tuple(self.capsules)) + 7 * hash(self.score)) % 1048575 )
    412 
    413     def __str__( self ):

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\game
 ↪py in __hash__(self)
    179                 if i:
    180                     h += base
--> 181                 base *= 2
    182         return hash(h)
    183 

KeyboardInterrupt:
```

[3]:
```
%run pacman.py -l maze_gen_500 -p SearchAgent -a fn=uniformCostSearch -q
```

```
[SearchAgent] using function uniformCostSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 2796 in 0.158 seconds
Search nodes expanded: 3416
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
```

```
d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作
业\lab1-search\pacman.py in <module>
    674
    675     args = readCommand( sys.argv[1:] ) # Get game components based on␣
 ↪input
--> 676     runGames( **args )
    677
    678     # import cProfile

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作
业\lab1-search\pacman.py in runGames(layout, pacman, ghosts, display, numGames,␣
 ↪record, numTraining, catchExceptions, timeout, save)
    632             rules.quiet = False
    633         game = rules.newGame( layout, pacman, ghosts, gameDisplay,␣
 ↪beQuiet, catchExceptions)
--> 634         game.run()
    635         if not beQuiet: games.append(game)
    636

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\game
 ↪py in run(self)
    616                 self.unmute()
    617             else:
--> 618                 observation = self.state.deepCopy()
    619
    620             # Solicit an action

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作
业\lab1-search\pacman.py in deepCopy(self)
    215     def deepCopy( self ):
    216         state = GameState( self )
--> 217         state.data = self.data.deepCopy()
    218         return state
    219

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\game
 ↪py in deepCopy(self)
```

```
    374         state = GameStateData( self )
    375         state.food = self.food.deepCopy()
--> 376         state.layout = self.layout.deepCopy()
    377         state._agentMoved = self._agentMoved
    378         state._foodEaten = self._foodEaten

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\layout.py in deepCopy(self)
     78 
     79     def deepCopy(self):
---> 80         return Layout(self.layoutText[:])
     81 
     82     def processLayoutText(self, layoutText):

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\layout.py in __init__(self, layoutText)
     20         self.agentPositions = []
     21         self.numGhosts = 0
---> 22         self.processLayoutText(layoutText)
     23         self.layoutText = layoutText
     24         self.totalFood = len(self.food.asList())

d:\seafile\陈启乾\我的资料库\课程资料\5-大二春\人工智能导论\作业\lab1-search\layout.py in processLayoutText(self, layoutText)
     97             for x in range(self.width):
     98                 layoutChar = layoutText[maxY - y][x]
---> 99                 self.processLayoutChar(x, y, layoutChar)
    100         self.agentPositions.sort()
    101         self.agentPositions = [ ( i == 0, pos) for i, pos in self.agentPositions]

KeyboardInterrupt:
```

因为上面的程序运行完需要很久的时间，因此在搜索出道路后就停止了程序。

A* 算法搜索节点数最少，但是所需时间较长，可能是启发函数带来的额外常数所致；

UCS，BFS 搜索节点较多，但是所需时间较短。

DFS 则因递归层数过多而无法运行。

2 实验二：迷宫中存在多个食物，甚至怪物，找到一条尽可能获得高分的路径。

[5]:
```
# 有怪物的，只有一个食物
%run pacman.py -l mediumScaryMaze -p MySearchAgent -a␣
 ↪fn=aStarSearch,heuristic=manhattanHeuristic,prob=MediumScarySearchProblem
```

```
[SearchAgent] using function aStarSearch and heuristic manhattanHeuristic
[SearchAgent] using problem type MediumScarySearchProblem
Path found with total cost of 6086 in 0.008 seconds
Search nodes expanded: 125
Pacman emerges victorious! Score: 418
Average Score: 418.0
Scores:        418.0
Win Rate:      1/1 (1.00)
Record:        Win
```

[6]:
```
# 有很多食物的，没有怪物
%run pacman.py -l foodSearchMaze -p MySearchAgent
```

```
[SearchAgent] using function aStarSearch and heuristic nullHeuristic
[SearchAgent] using problem type FoodSearchProblem
Path found with total cost of 36 in 0.002 seconds
Search nodes expanded: 36
Pacman emerges victorious! Score: 594
Average Score: 594.0
Scores:        594.0
Win Rate:      1/1 (1.00)
Record:        Win
```

2.1 设计思路

其实这里并非应该换 agent，而是应该换 problem，因为这里已经不是一个 position search problem（路径搜索问题），而分别是躲怪物和吃食物的两个问题。

我们在 searchAgents.py 中继承了 PositionSearchProblem，分别派生了 FoodSearchProblem 和 MediumScarySearchProblem，完成了这两个问题。

我们在不同的问题中改变了**目标的位置坐标和位置的代价函数**，来让吃豆人走出我们需要的路线。

2.1.1 第一个问题：MediumScary Maze

这个问题要求我们避开所有的怪物，到达右下角的终点。我们经过观察发现，怪物集中在右下角，因此我们就把右下侧的点的 cost 设置为 1000，其他店设置为 1。

这样我们就可以绕过怪物，从左上侧吃到食物

2.1.2 第二个问题：FoodSearch

这个问题要求我们吃到所有的食物。我们注意到：食物分布在上、下、左三侧边上。因此，我们将中间部分和右边部分的 cost 设置为 1000，其他位置设置为 1 。同时，我们将目标设置在右下角。

这样我们就可以依次经过：右上-左上-左下-右下角，吃到所有怪物。

3 实验三：地图中存在一些聪明的怪物的情况，吃豆人的目标是获取尽量高分

[2]:

```
# mini-max 搜索
%run pacman.py -p MinimaxAgent -l mediumClassic.lay -a␣
↪depth=3,evalFn="myScoreEvaluationFunction"
```

```
Pacman emerges victorious! Score: 1701
Average Score: 1701.0
Scores:        1701.0
Win Rate:      1/1 (1.00)
Record:        Win
```

[4]:

```
# alpha-beta 剪枝
%run pacman.py -p AlphaBetaAgent -l mediumClassic.lay -a␣
 ↪depth=4,evalFn="myScoreEvaluationFunction"
```

```
Pacman emerges victorious! Score: 1919
Average Score: 1919.0
Scores:        1919.0
Win Rate:      1/1 (1.00)
Record:        Win
```

3.1 实现解读

这里分别实现了 MiniMax 对抗搜索和 Alpha-Beta 的剪枝算法。

在 multiAgents.py 文件中，我们在 MinimaxAgent 类和 AlphaBetaAgent 中，分别实现了两种算法，重写了 getAction 接口。

```
def myScoreEvaluationFunction(currentGameState: GameState):
    # considering the food and the ghost's relative position
    ans = 0
    for food in currentGameState.getFood().asList():
        ans += 20 / (abs(food[0] - currentGameState.getPacmanPosition()[0]) +
                     abs(food[1] - currentGameState.getPacmanPosition()[1]) + 10)
    ans += currentGameState.getScore()
    return ans
```

这里还重写了 evaluation 函数，如上所示，这里将食物也计入考虑，具体的原因在后面会阐述。

3.2 表现评估

[1]:

```
# mini-max 搜索
%run pacman.py -p MinimaxAgent -l mediumClassic.lay -a␣
 ↪depth=3,evalFn="myScoreEvaluationFunction" -n 10
```

```
Pacman emerges victorious! Score: 1526
Pacman emerges victorious! Score: 1332
Pacman emerges victorious! Score: 1729
Pacman emerges victorious! Score: 1728
Pacman died! Score: -26
Pacman emerges victorious! Score: 1527
Pacman died! Score: -29
Pacman emerges victorious! Score: 1726
Pacman emerges victorious! Score: 2122
Pacman emerges victorious! Score: 2125
Average Score: 1376.0
Scores:        1526.0, 1332.0, 1729.0, 1728.0, -26.0, 1527.0, -29.0, 1726.0,
2122.0, 2125.0
Win Rate:      8/10 (0.80)
Record:        Win, Win, Win, Win, Loss, Win, Loss, Win, Win, Win
```

[1]:
```
# alpha-beta 剪枝
%run pacman.py -p AlphaBetaAgent -l mediumClassic.lay -a␣
↪depth=4,evalFn="myScoreEvaluationFunction" -n 10
```

```
Pacman emerges victorious! Score: 1539
Pacman emerges victorious! Score: 1706
Pacman died! Score: 440
Pacman emerges victorious! Score: 1731
Pacman emerges victorious! Score: 1733
Pacman emerges victorious! Score: 1925
Pacman emerges victorious! Score: 1329
Pacman emerges victorious! Score: 1938
Pacman died! Score: 462
Pacman emerges victorious! Score: 1909
Average Score: 1471.2
Scores:        1539.0, 1706.0, 440.0, 1731.0, 1733.0, 1925.0, 1329.0, 1938.0,
462.0, 1909.0
Win Rate:      8/10 (0.80)
Record:        Win, Win, Loss, Win, Win, Win, Win, Win, Loss, Win
```

3.2.1 Minimax 算法

Minimax 算法最多跑三层，大概这样的话一步 1 秒左右。可以看到，我们的胜率在 80% 左右，平均分在 1376。

3.2.2 Alpha-Beta 剪枝

Alpha-Beta 剪枝让我们的程序可以在相同的时间内多跑一层。可以看到，我们的胜率仍然在 80%，但平均分来到了更高的 1471.2。

3.3 摆烂问题的解决

值得提到的是，如果直接使用提供的 `scoreEvaluationFunction` 函数，则很容易因为没有怪物在旁边，搜索函数觉得往哪里走都差不多，很多步之后总能吃到附近的豆子，所以在原地不断打转的问题。为了解决这种问题，我们需要让 pacman 对整体局面有一个了解。因此，我们重写了 `myScoreEvaluationFunction`，给每个食物都赋予了一个权值，近处的很高而远处的较低，这样我们的吃豆人就可以更好完成任务。

所以我们得到这样的人生经验，某一个特定方向诱惑越大、欲望越多，就越不容易摆烂。