HSEA-第一次作业-实验报告

191250026 丁云翔

任务叙述

- 任务一：在search.py中的空函数aStarSearch中实现A*的图搜索代码和启发式函数代码。
- 任务二：基于任务一，当编写好aStarSearch函数后，现在我们将解决一个困难的搜索问题：在尽可能少的步骤中吃掉所有的豆子。 为此，我们引入一个新的搜索问题定义来形式化食物清理问题，即，searchAgents.py 中的FoodSearchProblem。 一个解决方案被定义为一条收集Pacman 世界中所有食物的路径。你需要完成 searchAgents.py 中foodHeuristic函数的编写，来帮助吃豆人进行游戏。
- 任务三：任务二中的解决方案解决方案只取决于墙壁、普通食物（小豆子）和吃豆人的位置,没有考虑任何幽灵和能量食物（大豆子)的作用。任务三中，我们将在minimaxClassic、originalClassic、powerClassic三个有幽灵的布局上试运行任务二中的A*搜索算法。

解决方法与必要分析

- 任务一：在search.py的aStarSearch方法中按照A*算法补充A*搜索代码，myHeuristic方法中补充启发式函数代码，启发式函数尝试了两种形式，分别为曼哈顿距离和欧氏距离。经过实际运行发现，曼哈顿距离的效果略好于欧氏距离，搜索中展开的节点数更少。分析可知，由于精灵只能上下左右移动，曼哈顿距离能更好地预估精灵离终点的距离，故效果会更好。

```
def aStarSearch(problem, heuristic=nullHeuristic):
    "*** YOUR CODE HERE ***"

    '''
    print("Start:", problem.getStartState())
    print("Is the start a goal?",
problem.isGoalState(problem.getStartState()))
    print("Start's successors:",
problem.getSuccessors(problem.getStartState()))
    '''

    # 特判，初始状态是否为目标状态
    startState = problem.getStartState()
    if problem.isGoalState(startState):
        return []

    # 声明搜索需要用到的变量
    actions = []
    stateQueue = util.PriorityQueue()
    stateQueue.push((startState, actions), heuristic(startState, problem))
    visited = []

    # A*搜索
    while ~stateQueue.isEmpty():
        nowState, nowActions = stateQueue.pop()
        if problem.isGoalState(nowState):
            return nowActions
        if nowState not in visited:
            visited.append(nowState)
            nowSuccessors = problem.getSuccessors(nowState)
```

```
            for newState, action, cost in nowSuccessors:
                newActions = nowActions + [action]
                stateQueue.push((newState, newActions),
problem.getCostOfActions(newActions) + heuristic(newState, problem))
    return actions
```

```
def myHeuristic(state, problem=None):
    "*** YOUR CODE HERE ***"
    x,y = state
    goalX,goalY = problem.getGoalState()
    # 使用当前位置与目标位置之间的曼哈顿距离作为启发式函数
    return abs(x - goalX) + abs(y - goalY)
    # 使用当前位置与目标位置之间的欧氏距离作为启发式函数
    # return math.sqrt(math.pow(x - goalX, 2) + math.pow(y - goalY, 2))
```

- 任务二：对于FoodSearchProblem，在searchAgent.py的foodHeuristic方法中补充对应的启发式函数代码。分析使用的三个地图，发现food均为线状分布，很自然会想到先走到食物构成的线上，再沿着线一路吃掉所有食物。故启发式函数设计的思路就是如果在线上，那沿着线走的启发式函数值最小，如果不在线上，则离线上最近的状态的启发式函数值最小。可以使用曼哈顿距离来统一定义，若在线上，下一个食物所在的位置到最近的食物曼哈顿距离是最小的；若不在线上，离线上越近的位置到最近的食物曼哈顿距离也越小。由于距离使用曼哈顿距离表示，该启发式函数满足满足admissible和consistent。因为由于精灵只能上下左右移动，且可能会受到墙的阻隔，曼哈顿距离一定小于等于实际的距离，该启发式函数满足admissible；还是因为精灵只能上下左右移动，原状态累计的开销加走一步的开销加上走过一步后的状态的启发式函数值不可能小于原状态累积的开销加原状态的启发式函数值，故该启发式函数也满足consistent。

```
def foodHeuristic(state, problem):
    position, foodGrid = state
    "*** YOUR CODE HERE ***"
    foodPositions = foodGrid.asList()
    # 没食物了就结束了
    if len(foodPositions) == 0:
        return 0

    # 先初始化为一个很大的值
    value = 1000000
    # 遍历所有食物，找当前位置到最近的食物的距离
    for food in foodPositions:
        if util.manhattanDistance(position, food) < value:
            value = util.manhattanDistance(position, food)

    return value
```

但运行以后发现实际效果并不是很好，怀疑是继续沿食物线走的状态与离开食物线的状态区分不明显所致，故在原来的算法上做了一些修改，将两种情况完全区分开了，算法效果大大提升，但修改后的启发式函数既不满足admissible也不满足consistent，只能作为一种针对特定问题的特化算法。

```
def foodHeuristic(state, problem):
    position, foodGrid = state
    "*** YOUR CODE HERE ***"
    foodPositions = foodGrid.asList()
    # 没食物了就结束了
```

```
    if len(foodPositions) == 0:
        return 0

    # 先初始化为一个很大的值
    value = 1000000
    # 遍历所有食物，找当前位置到最近的食物的距离
    for food in foodPositions:
        if util.manhattanDistance(position, food) < value:
            # 当前位置在食物的线状分布上，此为最优选，继续沿食物线走即可
            if util.manhattanDistance(position, food) == 1:
                value = 1
            # 当前位置与离当前位置最近的食物的曼哈顿距离，加10000是为了与“当前位置在食物的线状分布上”完全区分开
            else:
                value = 10000 + util.manhattanDistance(position, food)

    return value
```

- 任务三：将算法直接应用于新的三个布局即可，不需要改动代码

实验效果

- 任务一：
  - 空启发式

```
(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l bigMaze -p SearchAgent -a fn=astar
[SearchAgent] using function astar and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.2 seconds
Search nodes expanded: 620
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win

(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l openMaze -p SearchAgent -a fn=astar
[SearchAgent] using function astar and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.1 seconds
Search nodes expanded: 682
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win

(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l smallMaze -p SearchAgent -a fn=astar
[SearchAgent] using function astar and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 19 in 0.0 seconds
Search nodes expanded: 92
Pacman emerges victorious! Score: 491
Average Score: 491.0
Scores:        491.0
Win Rate:      1/1 (1.00)
Record:        Win
```

  - 曼哈顿距离

```
(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l bigMaze -p SearchAgent -a fn=astar,heuristic=myHeuristic
[SearchAgent] using function astar and heuristic myHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.1 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
```

```
(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l openMaze -p SearchAgent -a fn=astar,heuristic=myHeuristic
[SearchAgent] using function astar and heuristic myHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.1 seconds
Search nodes expanded: 535
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win

(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l smallMaze -p SearchAgent -a fn=astar,heuristic=myHeuristic
[SearchAgent] using function astar and heuristic myHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 19 in 0.0 seconds
Search nodes expanded: 53
Pacman emerges victorious! Score: 491
Average Score: 491.0
Scores:        491.0
Win Rate:      1/1 (1.00)
Record:        Win
```

- 欧氏距离

```
(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l bigMaze -p SearchAgent -a fn=astar,heuristic=myHeuristic
[SearchAgent] using function astar and heuristic myHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.1 seconds
Search nodes expanded: 557
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win

(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l openMaze -p SearchAgent -a fn=astar,heuristic=myHeuristic
[SearchAgent] using function astar and heuristic myHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.1 seconds
Search nodes expanded: 550
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win

(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l smallMaze -p SearchAgent -a fn=astar,heuristic=myHeuristic
[SearchAgent] using function astar and heuristic myHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 19 in 0.0 seconds
Search nodes expanded: 56
Pacman emerges victorious! Score: 491
Average Score: 491.0
Scores:        491.0
Win Rate:      1/1 (1.00)
Record:        Win
```

- 对比

  由于同一个地图不同启发式函数运行的结果中Score均相同，故在表格中只展示Search nodes expanded

	bigMaze	**openMaze**	**smallMaze**
空启发式	620	682	92
曼哈顿距离	549	535	53
欧氏距离	557	550	56

  对比可知，使用非空启发式函数以后，搜索过程中扩展的节点数减少了12%-43%，且对于越小越简单的地图，减少的幅度越大，对于大且复杂的地图，曼哈顿和欧氏距离启发式函数的效果比较有限。

- 任务二：
  - h=0

```
(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l Search1 -p AStarFoodSearchAgent
Path found with total cost of 6 in 0.0 seconds
Search nodes expanded: 14
Pacman emerges victorious! Score: 534
Average Score: 534.0
Scores:        534.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l Search2 -p AStarFoodSearchAgent
Path found with total cost of 16 in 0.1 seconds
Search nodes expanded: 707
Pacman emerges victorious! Score: 614
Average Score: 614.0
Scores:        614.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Search3运行后无响应，超过三分钟仍未出结果。

- 实现后的启发式函数（未特化）

```
(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l Search1 -p AStarFoodSearchAgent
Path found with total cost of 6 in 0.0 seconds
Search nodes expanded: 12
Pacman emerges victorious! Score: 534
Average Score: 534.0
Scores:        534.0
Win Rate:      1/1 (1.00)
Record:        Win

(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l Search2 -p AStarFoodSearchAgent
Path found with total cost of 16 in 0.1 seconds
Search nodes expanded: 538
Pacman emerges victorious! Score: 614
Average Score: 614.0
Scores:        614.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Search3运行后无响应，超过三分钟仍未出结果。

- 特化后的启发式函数

```
(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l Search1 -p AStarFoodSearchAgent
Path found with total cost of 6 in 0.0 seconds
Search nodes expanded: 9
Pacman emerges victorious! Score: 534
Average Score: 534.0
Scores:        534.0
Win Rate:      1/1 (1.00)
Record:        Win

(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l Search2 -p AStarFoodSearchAgent
Path found with total cost of 16 in 0.0 seconds
Search nodes expanded: 120
Pacman emerges victorious! Score: 614
Average Score: 614.0
Scores:        614.0
Win Rate:      1/1 (1.00)
Record:        Win

(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l Search3 -p AStarFoodSearchAgent
Path found with total cost of 31 in 0.0 seconds
Search nodes expanded: 217
Pacman emerges victorious! Score: 779
Average Score: 779.0
Scores:        779.0
Win Rate:      1/1 (1.00)
Record:        Win
```

- 对比

由于同一个地图不同启发式函数运行的结果中Score均相同，故在表格中只展示Search nodes expanded

	Search1	**Search2**	**Search3**
h=0	14	707	——
实现后的启发式函数（未特化）	12	538	——
特化后的启发式函数	9	120	217

对比可知，使用实现后的启发式函数以后，搜索过程中扩展的节点数有所减少，但对于过大过复杂的地图，该算法拓展的节点仍然过多，无法在短时间内给出解。特化后的启发式函数效果更为明显，搜索过程中扩展的节点数大幅减少，且对于越大越复杂的地图，减少的幅度越大，对于在h=0和未特化时均无响应的Search3，特化后的启发式函数依然可以在0.1s内找到路

径，效果十分明显，但由于不满足admissible和consistent，该算法并不能称为一个好的启发式算法。

- 任务三：

```
(base) C:\Users\丁云翔\Desktop\启发式搜索与演化算法\作业\hw1\hw1\search-code>python pacman.py -l minimaxClassic -p AStarFoodSearchAgent -n 5
Path found with total cost of 4 in 0.0 seconds
Search nodes expanded: 7
Pacman emerges victorious! Score: 516
Path found with total cost of 4 in 0.0 seconds
Search nodes expanded: 7
Pacman emerges victorious! Score: 516
Path found with total cost of 4 in 0.0 seconds
Search nodes expanded: 7
Pacman emerges victorious! Score: 516
Path found with total cost of 4 in 0.0 seconds
Search nodes expanded: 7
Pacman emerges victorious! Score: 516
Path found with total cost of 4 in 0.0 seconds
Search nodes expanded: 7
Pacman emerges victorious! Score: 516
Average Score: 516.0
Scores:        516.0, 516.0, 516.0, 516.0, 516.0
Win Rate:      5/5 (1.00)
Record:        Win, Win, Win, Win, Win
```

originalClassic和powerClassic运行后无响应，超过三分钟仍未出结果。

由于任务二中的启发式函数设计思路是在食物为线状分布的前提下提出的，具有一定的局限性，而任务三中的后两个地图食物并非简单的线状分布，故该启发式函数的表现较差。

操作说明

依照实验效果截图中的命令（即发布的作业说明文档给出的命令）运行即可。

PS：对于任务一，若想使用空启发式函数不需指定heuristic参数；代码默认使用曼哈顿距离，若想使用欧氏距离需将return曼哈顿距离的行注释掉，并将return欧氏距离的行取消注释。

PS：对于任务二/三，代码默认使用未特化的启发式函数，若想使用h=0，将return 0取消注释并将之前的启发式代码注释掉即可；若想使用特化后的启发式函数，需将未特化的值更新公式注释掉，并将特化后的值更新公式取消注释即可。