哈爾濱工業大學 HARBIN INSTITUTE OF TECHNOLOGY （深圳）

实验报告

开课学期：2022 年春季

课程名称：人工智能

实验名称：搜索策略 pacman

实验性质：课内实验

实验时间：2022.4.2 地点：

学生专业：计算机科学与技术

学生班级：1901105

学生学号：190110509

学生姓名：王铭

同组成员：肖力炜、郑羿恺、陈柏江

授课教师：郑海刚

报告成绩：

一、实验内容

本次实验为实现 pacman 游戏的搜索算法，其中前四个问题是搜索特定位置食物的路径问题的不同经典算法的实现（包括 DFS、BFS、UCS 和 A*算法）。第五、六个问题是基于前面实现的 BFS 和 A*算法，搜索四个角落食物的路径问题。第七、八个问题是用 A*算法和次最优搜索算法寻找吃掉所有豆子的路径。总的来说，本次实验实现从单角落搜索到多角落搜索最后再到任意食物的搜索问题，层层递进，对同一问题的解决也采用了不同的搜索策略。

在小组合作实验中，我负责完成的是第四个和第八个问题，即 A*搜索算法的实现和次最优搜索的实现。

二、算法介绍

2.1 A*算法

A*算法是一种求解最短路径的搜索算法，其核心为 f(n)=g(n)+h(n)，其中 g(n)是在状态空间中从初始状态到状态 n 的最小代价，h(n)是从状态 n 到目标状态的估计代价。对于最短路径上的节点，其 f(n)=f*(n)，g(n)=g*(n)。当 A*算法使用的 h(n)满足下界要求和一致性时，找到的路径是最优且效率高的。

A*算法从初始状态起，每次选择当前未扩展结点中 f(n)最小的结点进行扩展，直至找到目标节点，其流程可用下图表示：

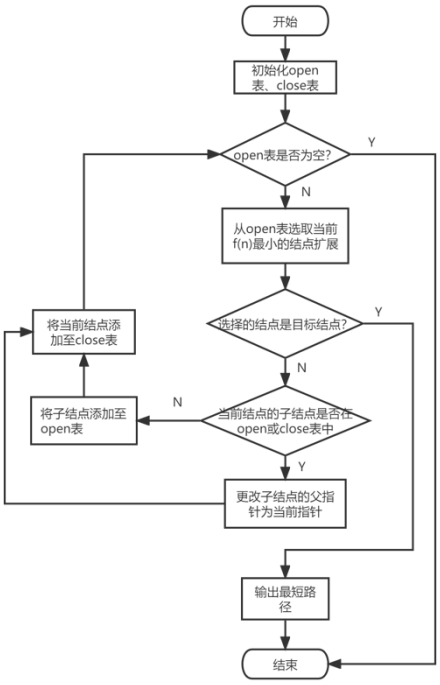


图 2.1 A*算法流程图

2.2 次最优搜索算法

当食物很多时，使用 A*算法搜索全局最优路径花费的时间过长，因此可以采用次最优搜索算法。该算法最先吃掉距离当前 pacman 最近的食物，可以降低搜索路径所需要的时间。

次最优搜索算法每次移动到距离当前结点最近的食物，直至地图上所有的食物均已被吃完结束。而寻找最短食物的路径可以用 BFS 实现。寻找距离当前结点最近食物路径的算法流程图和次最优算法的整体流程如图所示：

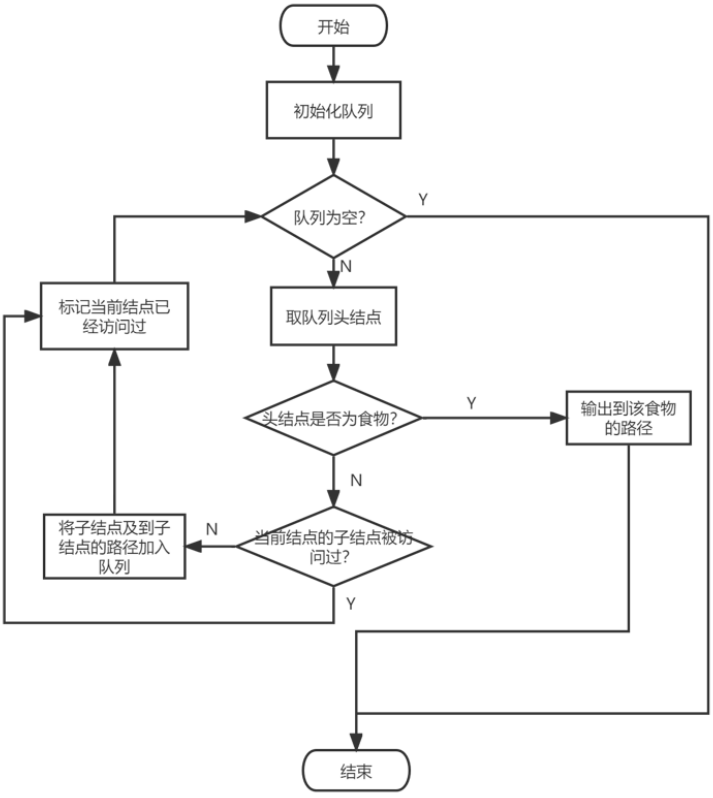


图 2.2 寻找距离最近的食物路径流程图

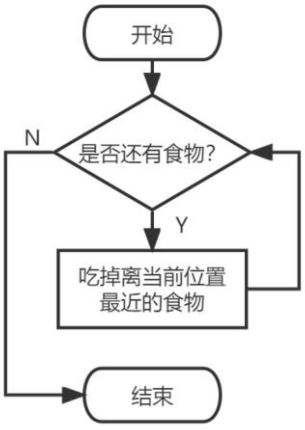


图 2.3 次最优搜索算法整体流程图

三、算法实现

3.1 实验环境与问题规模

实验环境为 windows10+pycharm+python3.8

问题规模为 bigMaze、bigSearch、trickySearch

3.2 关键代码、主要数据结构及说明

（1）问题四，A*算法实现

实现代码中所用的 queue 的主要数据结构为优先队列，该数据结构将按照 f(n)进行优先级排队，利用 push 和 pop 方法进行出入队列，从而便于选出当前未扩展结点中 f(n)最小的结点作为下一个要扩展的结点。

cost 用于记录结点的 g(n)和 h(n)便于排序。

path 用字典记录每个结点的父节点以及由父节点到当前结点的动作。

```
queue = util.PriorityQueue()
cost={}
path={}
result=[]
```

图 3.2.1 问题四主要数据结构

实现时，每次选出排队结点中 f(n)最小的结点进行扩展，对当前结点的子结点中未扩展的结点计算其 f(n)和 g(n)并用 cost 保存，并将子结点和其代价添加到优先队列 queue 中排队，同时记录其父节点及父节点到子结点的动作，用 path 保存。对于扩展过或在队列中排队的子结点，计算当前结点到这些结点的代价并判度是否需要更改指向父节点的指针。

最后根据 path 存储的信息，从目标结点返回初始结点，即可找到最短路径

实现关键代码如图所示：

```
while queue.isEmpty() == False:
    current_state = queue.pop()
    if problem.isGoalState(current_state):
        break
    candidate_state = problem.getSuccessors(current_state)
    for state in candidate_state:
        if state[0] not in path.keys():
            cost[state[0]]=[cost[current_state][0] + state[2], heuristic(state[0], problem)]
            queue.push(state[0],cost[state[0]][0]+cost[state[0]][1])
            path[state[0]]=[current_state,state[1]]
        else:
            newcost = state[2]+cost[current_state][0]
            if newcost < cost[state[0]][0]:# 更改父节点的指针并更新代价cost
                flag = False# 测试
                cost[state[0]][0] = newcost
                path[state[0]] = [current_state,state[1]]
                queue.update(state[0],newcost+cost[state[0]][1])
```

```
while path.get(current_state) and path[current_state][1] != 'Stop':
    result.append(path[current_state][1])
    current_state = path[current_state][0]
result.reverse()
return result
```

图 3.2.2 A*算法关键代码实现

（2）问题八，次优先搜索算法实现

寻找最近食物路径时采用 BFS 算法，找到的第一个食物即为离当前最近的食物，使用的数据结构主要为队列 queue。每次取队列的头结点并进行扩展，将未访问过的子结点及其路径送至队列进行排队，直至找到第一个食物。

其主要实现代码如图所示：

```
def registerInitialState(self, state):
    self.actions = []
    currentState = state
    while(currentState.getFood().count() > 0):
        nextPathSegment = self.findPathToClosestDot(currentState) # The missing piece
        self.actions += nextPathSegment
        for action in nextPathSegment:
            legal = currentState.getLegalActions()
            if action not in legal:
                t = (str(action), str(currentState))
                raise Exception('findPathToClosestDot returned an illegal move: %s!\n%s' % t)
            currentState = currentState.generateSuccessor(0, action)
    self.actionIndex = 0
    print('Path found with cost %d.' % len(self.actions))
```

```
"*** YOUR CODE HERE ***"
walked = []
queue = util.Queue()
queue.push((startPosition,[]))
while not queue.isEmpty():
    current_state,actions = queue.pop()
    if problem.isGoalState(current_state):
        return actions
    candidates = problem.getSuccessors(current_state)
    for candidate in candidates:
        if candidate[0] not in walked:
            queue.push((candidate[0],actions+[candidate[1]]))
    walked.append(current_state)
return []
```

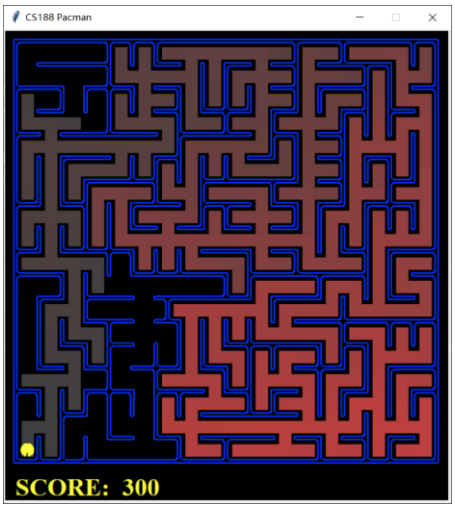
```
def isGoalState(self, state):
    """
    The state is Pacman's position. Fill this in with a goal test that will
    complete the problem definition.
    """
    x,y = state
    "*** YOUR CODE HERE ***"
    return self.food[x][y]
```

图 3.2.3 次优先搜索算法关键代码实现

3.3 实验结果

（1）问题四

运行 python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic，成功通过




```
C:\Users\86133\Downloads\QQ\人工智能\LAB\search-p3>python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuri
stic
[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
True
Path found with total cost of 210 in 0.0 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
```

图 3.3.1 问题四结果截图

与 DFS、BFS、代价一致搜索比较：

①DFS:

```
C:\Users\86133\Downloads\QQ\人工智能\LAB\search-p3>python pacman.py -l bigMaze -z .5 -p SearchAgent
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 212 in 0.0 seconds
Search nodes expanded: 391
Pacman emerges victorious! Score: 298
Average Score: 298.0
Scores:        298.0
Win Rate:      1/1 (1.00)
Record:        Win
```

图 3.3.2 DFS 结果截图

②BFS：

```
C:\Users\86133\Downloads\QQ\人工智能\LAB\search-p3>python pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.0 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
```

图 3.3.3 BFS 结果截图

③代价一致：

```
C:\Users\86133\Downloads\QQ\人工智能\LAB\search-p3>python pacman.py -l bigMaze -p SearchAgent -a fn=ucs -z .5
[SearchAgent] using function ucs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.0 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
```

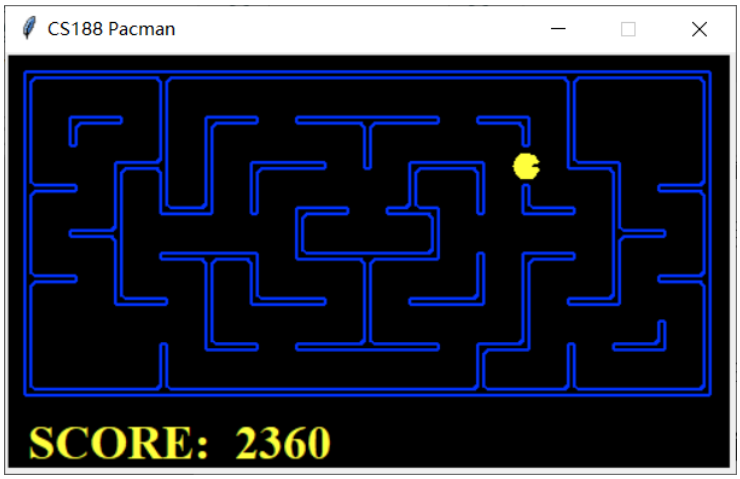
图 3.3.4 代价一致搜索结果截图

	DFS	BFS	UCS	A*
扩展结点数	391	620	620	549
耗时	0.0	0.0	0.0	0.0
Cost	212	210	210	210

可以看到，A*扩展结点数和 BFS、代价一致搜索的 cost 相同，但由于 A*算法采用每次扩展最小的 f(n)=g(n)+h(n)，因为比 BFS 和代价一致搜索扩展的结点总数都少，DFS 虽然结点数扩展的比 A*要少，但由 DFS 算法可知，DFS 寻找到的路径不一定是最优的，从结果看 DFS 的 cost 比 A*算法要高。

（2）问题八

运行 python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5 命令，可得




```
C:\Users\86133\Downloads\QQ\人工智能\LAB\search-p3>python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with cost 350.
Pacman emerges victorious! Score: 2360
Average Score: 2360.0
Scores:        2360.0
Win Rate:      1/1 (1.00)
Record:        Win
```

图 3.3.5 问题八结果截图

与问题七使用 A*算法对比：

运行 python pacman.py -l trickySearch -p ClosestDotSearchAgent

```
C:\Users\86133\Downloads\QQ\人工智能\LAB\search-p3>python pacman.py -l trickySearch -p ClosestDotSearchAgent
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with cost 68.
Pacman emerges victorious! Score: 562
Average Score: 562.0
Scores:        562.0
Win Rate:      1/1 (1.00)
Record:        Win
```

图 3.3.6 次优先搜索算法截图

运行 python pacman.py -l trickySearch -p AStarFoodSearchAgent

```
C:\Users\86133\Downloads\QQ\人工智能\LAB\search-p3>python pacman.py -l trickySearch -p AStarFoodSearchAgent
Path found with total cost of 60 in 20.1 seconds
Search nodes expanded: 4137
Pacman emerges victorious! Score: 570
Average Score: 570.0
Scores:        570.0
Win Rate:      1/1 (1.00)
Record:        Win
```

图 3.3.7 A*算法截图

对比可知，相同的问题下，A*算法的代价更低，但消耗的时间远大于此优先搜索。

运行 python autograder.py 命令，可以看到 q4,q8 均通过。

```
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
```

图 3.3.8 脚本测试截图

四、总结

4.1 碰到的问题及解决方法

碰到的问题：在搜索算法中，仅仅考虑了避免对同一结点的访问，疏忽了对可能存在更新结点的父指针指向问题。

解决方法:

对于满足下界条件和一致性的启发函数，那些父指针需要改变的结点仅存在于 open 表中（若存在 close 表中，则该结点的 f(n)一定小于等于正在扩展的结点，由单调性知，通过正在扩展的结点到达该结点的 f(n)一定比之前的 f(n)要大，无需改变），因此增加判断，对到达同一结点的两条路径，选择代价更小的路径作为到达该结点的路径，并更新在优先队列里的排队优先级。

4.2 实验的启发、总结、建议

通过本次实验，我实现了部分搜索算法，如 A*算法和次优先搜索算法，加深了对理论课上所学搜索知识的掌握，同时在小组合作中，增强了我的沟通能力和合作的意识。在实验过程中，我也意识到对于 A*算法，设计一个好的启发函数对于搜索效率的影响。