P01 Pacman Game

学号	姓名	专业(方向)
19335016	陈浩然	计算机科学与技术

1.Idea of A* Algorithm (Use a few sentences to describe your understanding of the algorithm)

- A^* 算法是一种在图形平面上，有多个节点的路径，求出最低通过成本的算法。该算法综合了最良优先搜索和 $Dijkstra$ 算法的优点：在进行启发式搜索提高算法效率的同时，可以保证找到一条最优路径（基于评估函数 $f(n)$ ）。$g(n)$ 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（根据所采用的评估函数的不同而变化）.$f(n) = g(n) + h(n)$

2. Idea of Min-Max and alpha-beta pruning algorithms

- The principle of Min-Max algorithm
- The principle of α-β pruning

3. Codes

Question 1

```
    pq = util.PriorityQueue()
    flag, path = [], []

    start = {
        'state' : problem.getStartState(),
        'cost' : 0,
        'parent' : None,
        'action' : None,
        'h' : heuristic(problem.getStartState(), problem)
    }

    pq.push(start, start['cost'] + start['h'])

    while (not pq.isEmpty()):
        top = pq.pop()
        if (top['state'] not in flag):
            flag.append(top['state'])

            if (problem.isGoalState(top['state'])):
                break
            arr = problem.getSuccessors(top['state'])

            for succ in list(arr):
                if (succ[0] not in flag):
                    child = {
                        'state' : succ[0],
                        'cost' : top['cost'] + succ[2],
                        'parent' : top,
                        'action' : succ[1],
```

```
                        'h' : heuristic(succ[0], problem)
                    }
                    pq.update(child, child['cost'] + child['h'])

    v = top
    while (v['action'] != None):
        path = [v['action']] + path
        v = v['parent']

    return path
    util.raiseNotDefined()
```

Question 2

```
def __init__(self, startingGameState):
    ...
    "*** YOUR CODE HERE ***"
    self.notVisitedCorners = []
    for _ in list(self.corners):
        if (self.startingPosition != _):
            self.notVisitedCorners.append(_)
    def getStartState(self):
        ...
        "*** YOUR CODE HERE ***"
        return (self.startingPosition, self.notVisitedCorners)
        util.raiseNotDefined()
    def isGoalState(self, state):
        ...
        "*** YOUR CODE HERE ***"
        if (len(state[1]) == 0):return True
        return False
        util.raiseNotDefined()
    def getSuccessors(self, state):
            ...
            "*** YOUR CODE HERE ***"
            x, y = state[0]
            dx, dy = Actions.directionToVector(action)
            nextx, nexty = int(x + dx), int(y + dy)
            hitsWall = self.walls[nextx][nexty]
            cost = 1

            if (not hitsWall):
                arr = state[1][:]
                if ((nextx, nexty) in state[1]):
                    arr.remove((nextx, nexty))
                    successors.append((((nextx, nexty), arr), action, cost))
                else:
                    successors.append((((nextx, nexty), state[1]), action, cost))
        self._expanded += 1
        return successors
def cornersHeuristic(state, problem):
    ...
    "*** YOUR CODE HERE ***"
    arr = state[1][:]
    place = state[0]
    h = 0
```

```python

    while arr != []:
        minn, i, j = 1919810, 0, 0
        for _ in arr:
            dis = abs(place[0] - _[0]) + abs(place[1] - _[1])
            if (dis < minn):
                minn = dis
                j = i
            i += 1
        h += minn
        place = arr[j]
        arr.remove(place)
    return h
```

Question 3

```python
def foodHeuristic(state, problem):
    ...
    """ YOUR CODE HERE """
    foods = foodGrid.asList()
    res = 0
    if (len(foods) == 0):
        return 0

    for food in foods:
        newProblem = PositionSearchProblem(problem.startingGameState,
                                           start = position,
                                           goal = food,
                                           warn=False,
                                           visualize=False)
        distance = len(search.bfs(newProblem))
        res = max(res, distance)
    return res
```

Question 4

```go
// 这里填写go代码
```

Question 5

```csharp
// 这里填写c#代码
```

4.结果展示

```
(py2) C:\Users\asd\P01\P01_Pacman\search>python pacman.py -l bigMaze -z .5 -p SearchAgent
-a fn=astar,heuristic=manhattanHeuristic
[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
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
```

Question 1

```
(py2) C:\Users\asd\P01\P01_Pacman\search>python pacman.py -l mediumCorners -p
AStarCornersAgent -z 0.5
Path found with total cost of 106 in 0.0 seconds
Search nodes expanded: 692
Pacman emerges victorious! Score: 434
Average Score: 434.0
Scores:        434.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Question 2

```
(py2) C:\Users\asd\P01\P01_Pacman\search>python pacman.py -l trickySearch -p
 AStarFoodSearchAgent
Path found with total cost of 60 in 19.6 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
```

Question 3

5.结果分析

1.Search in Pacman

- 在 $question1$ 中,其使用的 h 函数为 $h(n)=0$, 通过使用优先队列初步实现 A^* 算法; 在 $question2$ 中,通过 $Manhattan$ 距离实现 h 函数; 而在 $question3$ 中, 使用了 bfs 以实现.
- If u have innovation points, just write it down.

2.Multi-Agent Pacman

- Briefly analyze the complexity difference between α-β pruning and minmax algorithm（hints: search depth and time)

6.Experimental experience