BÁO CÁO KẾT QUẢ THUẬT TOÁN BỐN GỐC MÊ CUNG

1. Bài toán bốn góc mê cung

Trong phần bào toán này, mình sẽ thiết kế 1 **Class CornersProblem** trong file *searchAgent.py* để thể hiện bài toán tìm kiếm sau: tìm đường đi ngắn nhất cho Pacman trong mê cung sao cho Pacman đi qua bốn góc của mê cung.
-Hình ảnh minh hoạ cài Class mà cần làm:

```
#Đây là phần code cho câu 2.5 Tuần 4
class CornersProblem(search.SearchProblem):
    """
    This search problem finds paths through all four corners of a layout.
    You must select a suitable state space and successor function
    """

    def __init__(self, startingGameState):
        """
        Stores the walls, pacman's starting position and corners.
        """
        self.walls = startingGameState.getWalls()
        self.startingPosition = startingGameState.getPacmanPosition()
        top, right = self.walls.height-2, self.walls.width-2
        self.corners = ((1,1), (1,top), (right, 1), (right, top))
        for corner in self.corners:
            if not startingGameState.hasFood(*corner):
                print ('Warning: no food in corner ' + str(corner))
        self._expanded = 0  #Số lượng nút tìm kiếm được mở rộng


        #Bạn có thể thêm bất kỳ mã nào ở đây mà bạn muốn sử dụng

        # in initializing the problem
```

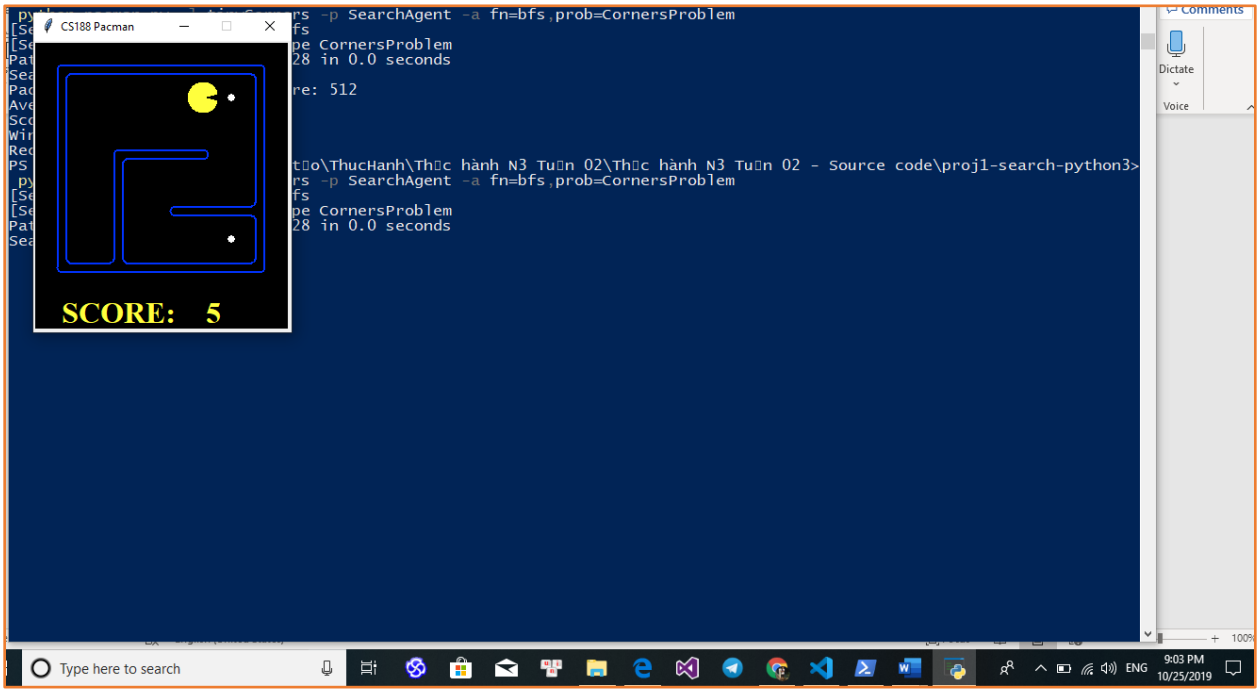
```
    #Bạn có thể thêm bất kỳ mã nào ở đây mà bạn muốn sử dụng

    # in initializing the problem
    "*** YOUR CODE HERE ***"
    self.right = right
    self.top = top

def getStartState(self):
    "Returns the start state (in your state space, not the full Pacman state space)"
    "*** YOUR CODE HERE ***"
    #khởi tạo các góc đều trả về false
    allCorners = (False, False, False, False)
    start = (self.startingPosition, allCorners)
    #trả về nơi bắt đầu
    return start
    util.raiseNotDefined()

def isGoalState(self, state):
    "Returns whether this search state is a goal state of the problem"
    "*** YOUR CODE HERE ***"
    #khi pacman xuất hiện thì ở gần 1 góc nào đó
    corners = state[1]
    boolean = corners[0] and corners[1] and corners[2] and corners[3]
    return boolean
    util.raiseNotDefined()

def getSuccessors(self, state):
    """
    Returns successor states, the actions they require, and a cost of 1.
     As noted in search.py:
```

```
successors = []
for action in [Directions.NORTH, Directions.SOUTH, Directions.EAST, Directions.WEST]:
    # Add a successor state to the successor list if the action is legal
    # Here's a code snippet for figuring out whether a new position hits a wall:
    #   x,y = currentPosition
    #   dx, dy = Actions.directionToVector(action)
    #   nextx, nexty = int(x + dx), int(y + dy)
    #   hitsWall = self.walls[nextx][nexty]

    "*** YOUR CODE HERE ***"
    x,y = state[0]
    holdCorners = state[1]
    dx, dy = Actions.directionToVector(action)
    nextx, nexty = int(x + dx), int(y + dy)
    hitsWall = self.walls[nextx][nexty]
    newCorners = ()
    nextState = (nextx, nexty)
    if not hitsWall:
        if nextState in self.corners:
            if nextState == (self.right, 1):
                newCorners = [True, holdCorners[1], holdCorners[2], holdCorners[3]]
            elif nextState == (self.right, self.top):
                newCorners = [holdCorners[0], True, holdCorners[2], holdCorners[3]]
            elif nextState == (1, self.top):
                newCorners = [holdCorners[0], holdCorners[1], True, holdCorners[3]]
            elif nextState == (1,1):
                newCorners = [holdCorners[0], holdCorners[1], holdCorners[2], True]
            successor = ((nextState, newCorners), action,  1)
        else:
            successor = ((nextState, holdCorners), action, 1)
```

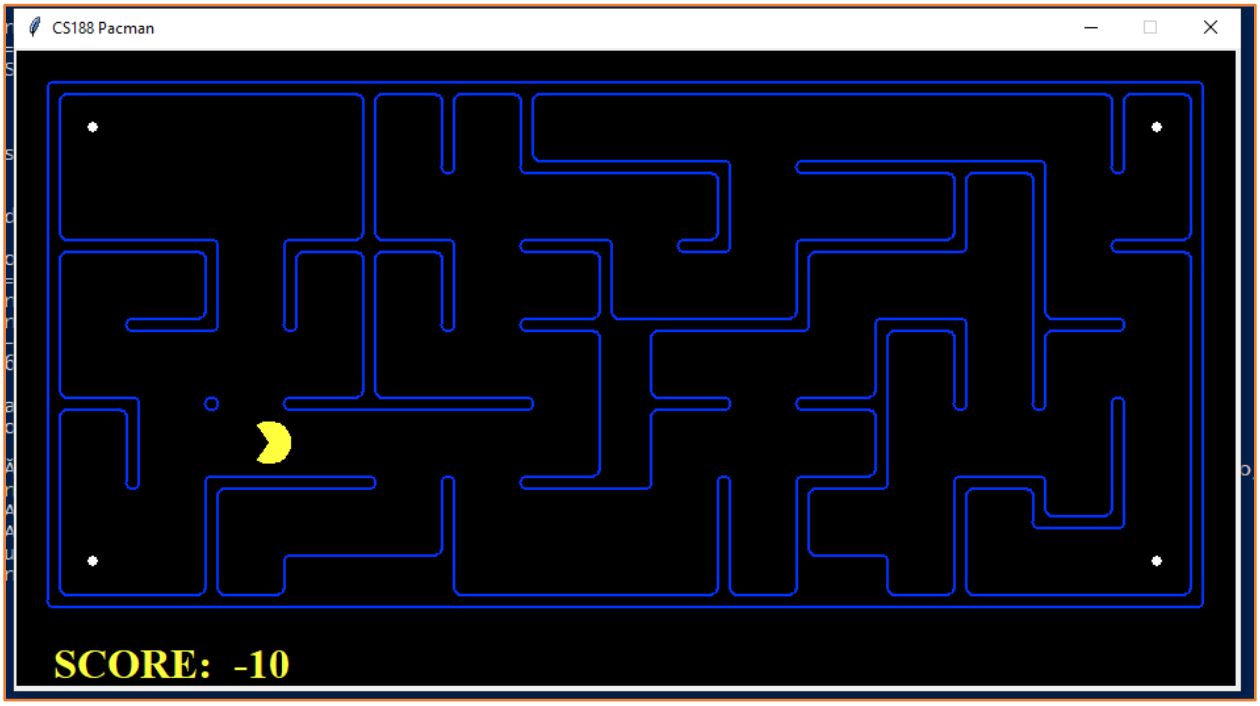
```
                    newCorners = [holdCorners[0], holdCorners[1], holdCorners[2], True]
                successor = ((nextState, newCorners), action,  1)
            else:
                successor = ((nextState, holdCorners), action, 1)
            successors.append(successor)

    self._expanded += 1
    return successors

def getCostOfActions(self, actions):
    """
    Returns the cost of a particular sequence of actions.  If those actions
    include an illegal move, return 999999.  This is implemented for you.
    """
    if actions == None: return 999999
    x,y= self.startingPosition
    for action in actions:
        dx, dy = Actions.directionToVector(action)
        x, y = int(x + dx), int(y + dy)
        if self.walls[x][y]: return 999999
    return len(actions)
```

-Sau khi xây dựng code xong, mình sẽ chạy thực thi đoạn code qua lệnh:

python pacman.py -l tinyCorners -p SearchAgent -a fn=bfs,prob=CornersProblem



python pacman.py -l mediumCorners -p SearchAgent -a fn=bfs,prob=CornersProblem



-Và bạn có thể bạn gõ ***python autograder.py -q q5*** để kiểm tra phần cài đặt của bạn với các bộ test khác nhau.

```
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 ###


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

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


Finished at 21:05:06

Provisional grades
==================
Question q2: 3/3
Question q5: 3/3
------------------
Total: 6/6
```

2. Heuristic cho bài toán bốn góc mê cung

Sau khi hoàn thành câu 2.5 thì mình tiếp tục đến với thuật toán tìm đường ăn thức ăn bốn gốc mê cung bằng **Heuristic .**Mình sẽ viết hoàn chỉnh hàm **cornersHeuristic** trong file *searchAgent.py*. Hàm này nhận hai tham số đầu vào là trạng thái state và bài toán problem, và trả về chi phí ước lượng từ state đến đích.

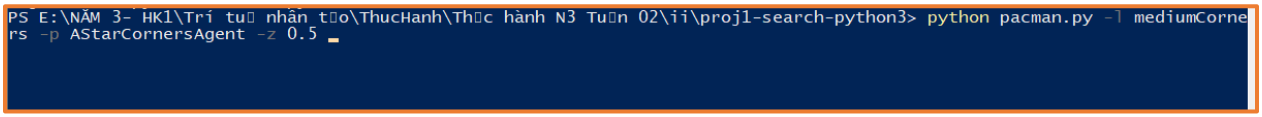
-Hình ảnh minh hoạ code và note lại những ghi chú trong khi code:

```python
def cornersHeuristic(state, problem):
    """
    A heuristic for the CornersProblem that you defined.
      state:   The current search state
               (a data structure you chose in your search problem)
      problem: The CornersProblem instance for this layout.
    This function should always return a number that is a lower bound
    on the shortest path from the state to a goal of the problem; i.e.
    it should be admissible (as well as consistent).
    """
    corners = problem.corners # These are the corner coordinates
    walls = problem.walls # These are the walls of the maze, as a Grid (game.py)

    "*** YOUR CODE HERE ***"
    #thuật toán tìm kiếm mê cung 4 góc
    H = 0   #Chi phí khi khởi tạo bằng 0
    position = state[0][:]  # trạng thái là 0 khi ở vị trí hiện tại
    visitedlist = state[1][:]  # danh sách mà pacman đã đi qua các trạng thái
    for i in range(4):  # vẫn còn nhiều nhất là 4 góc không mong muốn, vì vậy chúng ta cần phải đi trong bốn lần
        cornersDistance = [0, 0, 0, 0]  # 4 góc
        cornerIndex = 0
        # tính khoảng cách giữa vị trí hiện tại và từng góc
        for corner in visitedlist:
            #gọi lại hàm manhattanDistance để tính khoảng cách pacman đi tìm trong file Util.py, trả về 1 bộ
            cornersDistance[cornerIndex] = util.manhattanDistance(
                position, corners[cornerIndex])
            cornerIndex += 1
```

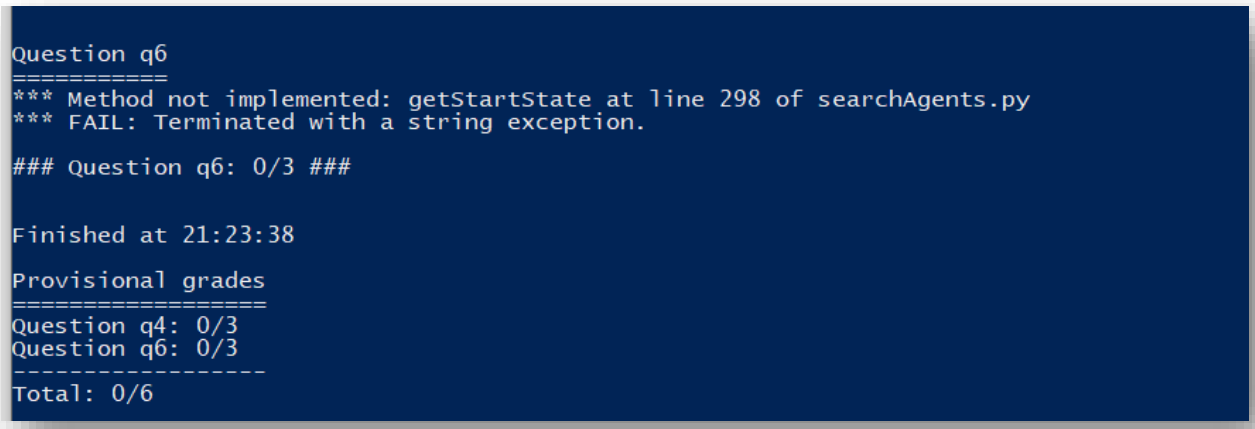
```python
        closestIndex = 0
        cornerIndex = 0  # reset to 0
        for cornerDiatance in cornersDistance:
            if (visitedlist[closestIndex]):  # Trường hợp góc thứ 1 đã ghé thăm  thì ta sẽ
                closestIndex = cornerIndex

            #  kiểm tra từng góc:
            #  1. không được truy cập đến  hay ghé thăm
            #  2. khoảng cách nhỏ hơn khoảng cách gần nhất hiện tại = > được đặt thành góc gần nhất mới
            if (not visitedlist[cornerIndex]) and (cornerDiatance < cornersDistance[closestIndex]):
                closestIndex = cornerIndex
            cornerIndex += 1

        if (not visitedlist[closestIndex]):
            H += cornersDistance[closestIndex]
            position = corners[closestIndex][:]
            visitedlist[closestIndex] = True
        else:
            break

    return H  #trả về chi phí
```

-Sau khi đã cài đặt xong, bạn có thể kiểm tra một cách trực quan trên game Pacman bằng cách gõ câu lệnh:

python pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5 với AStarCornersAgent là "shortcut" của: -p SearchAgent -a fn=aStarSearch,prob=CornersProblem,heuristic=cornersHeuristic



-Sau khi chạy lệnh , bạn có thể gõ ***python autograder.py -q q6*** để kiểm tra phần cài đặt của bạn với các bộ test khác nhau:



--→ Kết quả này bị Fail rồi, mình sẽ cố gắng sửa lại thuật toán sớm nhất có thể !

Kế đến, tùy vào số node mà A* phải mở (phụ thuộc vào chất lượng của heuristic) mà bạn sẽ nhận được các điểm số khác nhau:

- Nếu số node mở > 2000 thì bạn sẽ được 0/3 điểm.
- Nếu 1600 < số node mở ≤ 2000 thì bạn sẽ được 1/3 điểm.
- Nếu 1200 < số node mở ≤ 1600 thì bạn sẽ được 2/3 điểm.
- Nếu số node mở ≤ 1200 thì bạn sẽ được 3/3 điểm.

THANK YOU