Miguel Angel Nivia Ortega **10/09/2024**
Willian Chapid Tobar
Daniel Andrez Vasquez Murillo

NOTA: Para la realización del trabajo se usó Claude(IA), e hicimos los siguientes prompts para llegar a las respuestas:

PROMPT REALIZADO

Conoces el proyecto berkeley sobre la implementación de algoritmos de búsqueda para el juego de pacman, construido en Python 2.7.0. A continuación, se detallan las instrucciones para resolver el problema Q5 que realiza el proceso pero con A* sin embargo para este caso cambia en base a lo siguiente:

Instrucciones

Debes implementar la función geneticSearch. La implementación debe cumplir con los siguientes requisitos:

1. Implementar Algoritmo de búsqueda genetico:

- ○ Implementa el algoritmo de búsqueda genética.
- ○ El algoritmo debe retornar una lista de acciones que lleven al agente desde el estado inicial hasta el objetivo. Asegúrate de que todas las acciones sean movimientos legales (no pasar a través de paredes).

2. Uso de SearchAgent:

- ○ Hay se encuentra el CornersProblem que necesitamos para poder construir el laberinto con la comida de pacman a las esquinas y como debe comportarse este en el mapa.

Consideraciones:

- ○ **Solo puedes modificar en searchAgent el CornersProblem nada más.**
- ○ **La funcion de Search de genética debe usar el CornersProblem para su movimiento.**

Código de Inicio

Ten presente este código para el searcAgent:
”””

```
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 # DO NOT CHANGE; Number of search nodes
expanded
        # Please add any code here which you would like to use
```

```
        # in initializing the problem
        "*** YOUR CODE HERE ***"

    def getStartState(self):
        """
        Returns the start state (in your state space, not the full
Pacman state
        space)
        """
        "*** YOUR CODE HERE ***"
        util.raiseNotDefined()

    def isGoalState(self, state):
        """
        Returns whether this search state is a goal state of the
problem.
        """
        "*** YOUR CODE HERE ***"
        util.raiseNotDefined()

    def getSuccessors(self, state):
        """
        Returns successor states, the actions they require, and a cost
of 1.

         As noted in search.py:
            For a given state, this should return a list of triples,
(successor,
            action, stepCost), where 'successor' is a successor to the
current
            state, 'action' is the action required to get there, and
'stepCost'
            is the incremental cost of expanding to that successor
        """

        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 ***"

        self._expanded += 1 # DO NOT CHANGE
        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
```

Miguel Angel Nivia Ortega **10/09/2024**
Willian Chapid Tobar
Daniel Andrez Vasquez Murillo

```
        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)
```

“””

Resultados Autograder:

```
Question q5
===========

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

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

```
Finished at 15:36:43

Provisional grades
==================
Question q1: 3/3
Question q2: 3/3
Question q3: 3/3
Question q4: 3/3
Question q5: 3/3
Question q6: 0/3
Question q7: 0/4
Question q8: 0/3
------------------
Total: 15/25

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.

FAMILIA (main *) search
```