UK BRIGHT NETWORK INTERNSHIP ON DEMAND

NAME: SYED MUHAMMAD SHOAIB

amazon Coding Challenge

Bright Network Internship, June 2022

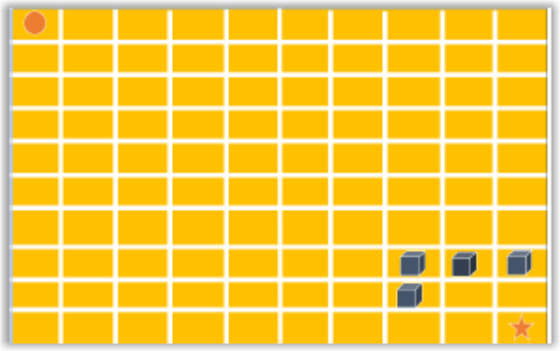
Overview:

Thank you for choosing to take part in Amazon's Coding Challenge for Bright Network. We are very happy to have you here. We hope you have as much fun implementing this challenge as we had creating it!

In this challenge, you are going to implement Amazon's pathfinding algorithm for Amazon's self-driving delivery vehicles. The self-driving vehicle will need to create a path on a 2D-grid that contains a starting point *(x,y)*, a delivery point *(x,y)* and a number of obstacles. Your vehicle can navigate to any of the adjacent squares (even diagonally), as long as the squares are inbound and do not contain an obstacle.

General notes:

You can use any language and ideally the output is to a command line.



Phase 1:

Implement a 10x10 grid that contains a starting point on (0,0), the delivery point on (9,9) and the following obstacles on locations (9,7) (8,7) (6,7) (6,8).

Your algorithm should calculate a valid path avoiding the obstacles and reaching the delivery point.

Your solution should print the path in the format of
[(x1,y2), (x2,y2)….] and also the number of steps.

Phase 2:

Add an additional 20 randomly placed obstacles and print their location using the format [(x1,y1),(x2,y2)…].

The obstacles should not overlap existing ones and should not be places at the start and delivery points.

Your algorithm should calculate a valid path avoiding the obstacles and reaching the delivery point.

Your solution should print the path in the format of
[(x1,y2), (x2,y2)….]

Bonus:

In the event that your vehicle is unable to reach its destination, your algorithm should print "Unable to reach delivery point" and identify which obstacles to be removed in order for the vehicle to reach its destination.

Your algorithm should suggest the least amount of obstacles using the format [(x1,y1,(x2,y2)…)] in order for your vehicle to reach its destination.

CODE:

```
# Setting up my environment and importing relevant libraries
from pathfinding.core.grid import Grid; # to create a grid
from pathfinding.core.diagonal_movement import DiagonalMovement    # to allow diagonal movement
from pathfinding.finder.a_star import AStarFinder    # importing A* pathfinding algorithm (this takes
                    # weights into account so in this case all walkable nodes will be given the same weight)
import random    # This will use to generate random locations for the additional obstacles
# Phase 1
# Setting up a 10x10 grid with starting point, delivery point and 4 obstacles at
# (0,0), (9,9), (9,7), (8,7), (7,7) and (6,8) respectively:
print("------Amazon Path Finding Algorithm 2D Grid------\n")
matrix = []
for i in range(10):
   matrix.append([1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
matrix[7][7] = 0
matrix[6][8] = 0    #nodes with obstacles are given weight 0 as they are not walkable
matrix[8][7] = 0
matrix[9][7] = 0
# Calculating a valid path to the delivery point, avoiding obstacles:
def find_path(matrix):
   grid = Grid(matrix=matrix)
   start = grid.node(0, 0)
   end = grid.node(9, 9)
   finder = AStarFinder(diagonal_movement = DiagonalMovement.always)
   path, runs = finder.find_path(start, end, grid)
   if path != []:
      print('\nNumber of Steps Required to Reach Destination: ', runs)
      print('Total Nodes Visited: ', path)
```

```
    else:
        print("\nUnable to reach delivery point")
```

#Print path and number of steps required:

#solution format: [(x1, y1), (x2, y2), ...]

```
find_path(matrix) #already have print statement inside function to make code neater
```

#Phase 2

#Adding 20 randomly placed obstacles, ensuring they do not overlap existing ones:

```
obstacle_coords = []
while len(obstacle_coords) < 21:
    coord = (random.randint(0, 9), random.randint(0, 9))
    if coord != (6,8) and (7,7) and (8,7) and (9,7) and (0,0) and (9, 9):
        obstacle_coords.append(coord)
print('\n20 additional obstacles Placed at: ',obstacle_coords)
```

 # printing locations of new obstacles:

 # Change 1s in these locations in matrix to 0s to represent obstacles:

```
for coord in obstacle_coords:
    matrix[coord[0]][coord[1]] = 0
```

#Calculate valid path to delivery point avoiding obstacles and print coordinates:

```
find_path(matrix)
```

OUTPUT:

Phase 1

```
------Amazon Path Finding Algorithm 2D Grid------


Number of Steps Required to Reach Destination:  24
Total Nodes Visited:  [(0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 6), (8, 7), (8, 8), (9, 9)]
```

Phase 2

```
20 additional obstacles Placed at:  [(8, 7), (6, 3), (5, 1), (1, 8), (0, 3), (4, 8), (2, 7), (2, 9), (5, 5), (0, 9), (4, 3),
(0, 1), (1, 2), (1, 2), (5, 1), (0, 8), (2, 1), (2, 1), (1, 0), (9, 6), (1, 5)]

Number of Steps Required to Reach Destination:  18
Total Nodes Visited:  [(0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (5, 4), (6, 5), (7, 6), (8, 7), (8, 8), (9, 9)]
```