Artificial Intelligence Course

Project 1: Search in Pacman

First name	Last name	Student number
Maureen	Boudart	2591892
Charton	Clément	2592257
Lemercier	Nelson	2593887

Comments about the assignment (if you have)

Question 1: Finding a Fixed Food Dot using Depth First Search (3 points)

The principle of the Depth first search is to visit the nodes until the leaf before exploring a new branch. The use of the stack permit to visit the successor of a node by popping up the last nodes pushed before visiting the node at the same layer.

To perform the Depth First Search algorithm, we choose to use a stack as a fringe to store the successors of the visited node so the first successors in the fringe are the last popped. To keep a track of each node, we decided to push a tuple in the stack: the state of the node and the path from the start state to this node state. Once the goal state is found, we just have to return the path from the start state to the goal state.

Question 2: Breadth First Search (3 points)

Contrary to the Depth first Search, Breadth First Search visit in priority the nodes in same layer, and explore the tree, layer by layer. The use of the queue permit to pop up the older elements in the queue. Thus, when we are searching the successor of each nodes and adding them to the queue, the node on the same layer will be explore before the successors.

To perform the Breadth First Search algorithm, we choose to use a queue as a fringe. In this queue, we store the path from the start node to the node we are visiting. Each time we take a list in the queue, we search the successors of the last element of the list, we add the successors in the list and we push it in the queue. Once the goal is found, we take all the direction of the list we just popped and return it.

This function was modified to fit the Q5 requirements.

Question 3: Varying the Cost Function (3 points)

To perform the uniform cost search, we choose a priority queue as a fringe. We push the successor nodes and the cost. In a second priority queue, we push the path and the cost of this path. Using a priority queue allows to take always the path with the less cost. We are working on the two priority queue simultaneously. Once the goal is found, we return the path taken from the priority queue.

Question 4: A* search (3 points)

We take back the code from the Q3 and added the heuristic function.

Question 5: Finding All the Corners (3 points)

We implemented all the functions of the class corners problem in the file searchAgent.py.

In the function getStartState, we return the starting position and the coordinates of the corners.

In the function GoalState, we return a boolean value if all the corners have been visited or not.

In the function getSuccessor, we verify if the successors are one of the corner. If it is the case, we remove it from the corner list. Finally we return the successors and the corners left we have to visit.