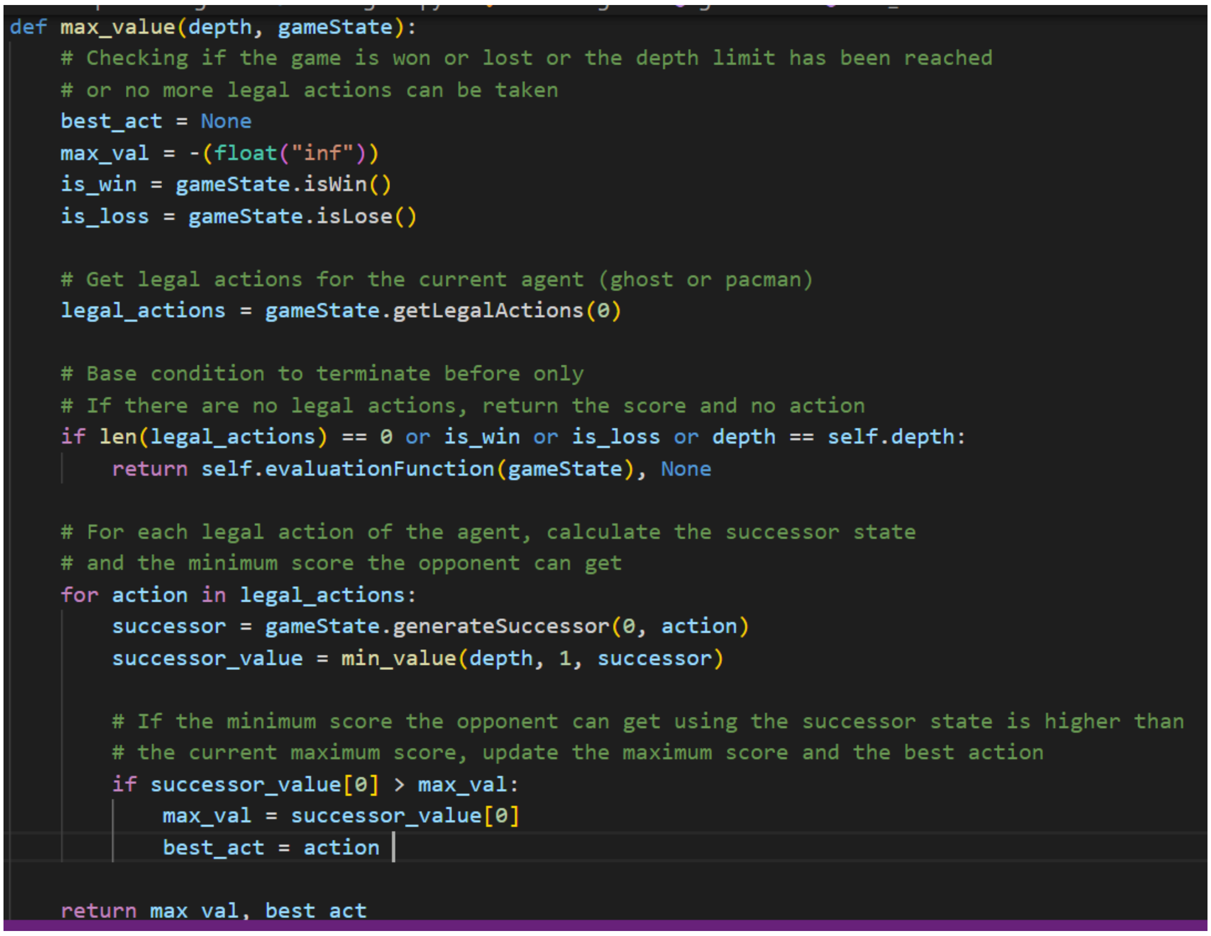
# COEN 266 Artificial Intelligence

**Group Homework 2**

Ruchi Dhore, Vaibhav Sachdeva, Sanskriti Patole

## 1. How does the "max" function work?

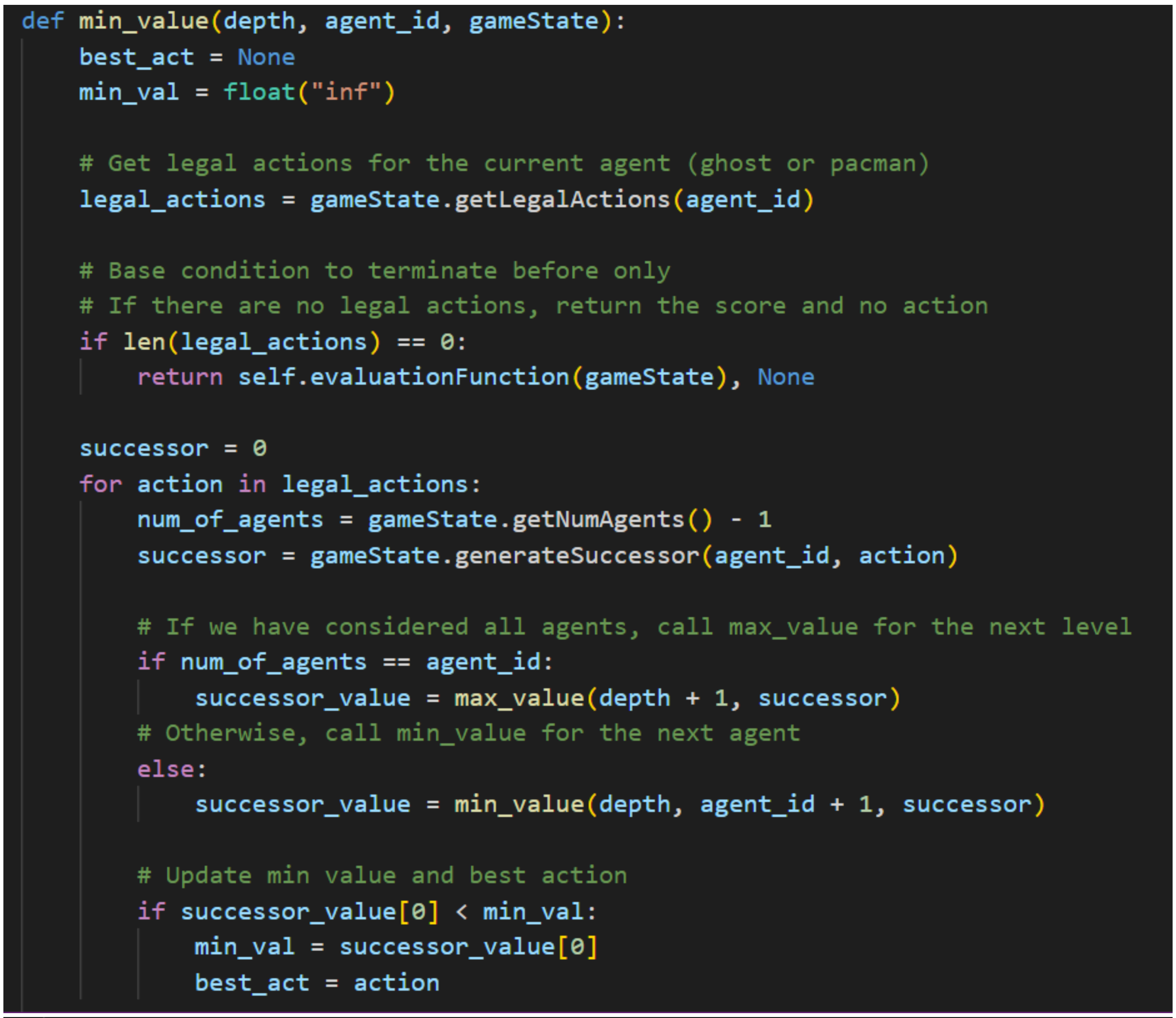


```python
def max_value(depth, gameState):
    # Checking if the game is won or lost or the depth limit has been reached
    # or no more legal actions can be taken
    best_act = None
    max_val = -(float("inf"))
    is_win = gameState.isWin()
    is_loss = gameState.isLose()

    # Get legal actions for the current agent (ghost or pacman)
    legal_actions = gameState.getLegalActions(0)

    # Base condition to terminate before only
    # If there are no legal actions, return the score and no action
    if len(legal_actions) == 0 or is_win or is_loss or depth == self.depth:
        return self.evaluationFunction(gameState), None

    # For each legal action of the agent, calculate the successor state
    # and the minimum score the opponent can get
    for action in legal_actions:
        successor = gameState.generateSuccessor(0, action)
        successor_value = min_value(depth, 1, successor)

        # If the minimum score the opponent can get using the successor state is higher than
        # the current maximum score, update the maximum score and the best action
        if successor_value[0] > max_val:
            max_val = successor_value[0]
            best_act = action

    return max_val, best_act
```

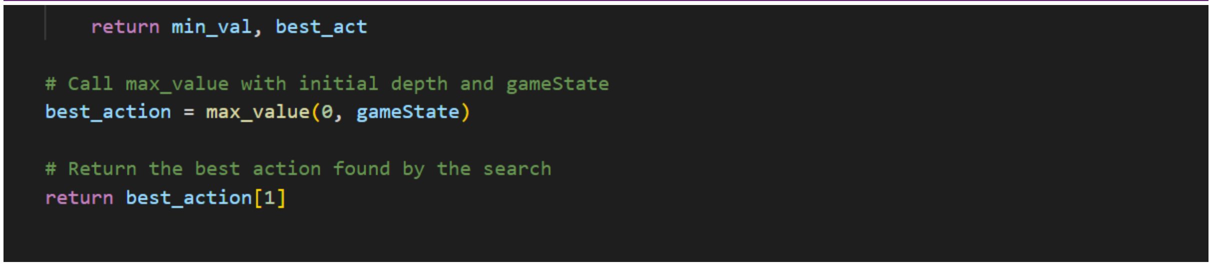
The purpose of this function is to determine the maximum value that the current agent (Pacman) can get in the current state, given a certain depth limit.

The function starts by checking if the game is won or lost or if the depth limit has been reached, in which case it returns the score of the current state and no action. Otherwise, it gets the legal actions for the current agent and loops through each action. For each action, it generates a successor state and calculates the minimum score the opponent (ghost) can get from that state by calling the 'min_value' function. It then updates the maximum score and the best action if the minimum score is higher than the current maximum score. Finally, it returns the maximum score and the best action.

## 2. How does the "min" function work?



```python
def min_value(depth, agent_id, gameState):
    best_act = None
    min_val = float("inf")

    # Get legal actions for the current agent (ghost or pacman)
    legal_actions = gameState.getLegalActions(agent_id)

    # Base condition to terminate before only
    # If there are no legal actions, return the score and no action
    if len(legal_actions) == 0:
        return self.evaluationFunction(gameState), None

    successor = 0
    for action in legal_actions:
        num_of_agents = gameState.getNumAgents() - 1
        successor = gameState.generateSuccessor(agent_id, action)

        # If we have considered all agents, call max_value for the next level
        if num_of_agents == agent_id:
            successor_value = max_value(depth + 1, successor)
        # Otherwise, call min_value for the next agent
        else:
            successor_value = min_value(depth, agent_id + 1, successor)

        # Update min value and best action
        if successor_value[0] < min_val:
            min_val = successor_value[0]
            best_act = action
```



```python
    return min_val, best_act

# Call max_value with initial depth and gameState
best_action = max_value(0, gameState)

# Return the best action found by the search
return best_action[1]
```

The ‘min_value’ function computes the minimum value and corresponding best action that the current ghost player can take at a certain depth in the game tree. It does this by first getting all the legal actions available to the current ghost player using the ‘getLegalActions’ method of the ‘gameState’ object.

The function then checks if there are any legal actions available. If not, it returns the score and no action. Otherwise, it generates a successor game state for each legal action and calls either ‘max_value’ or ‘min_value’ recursively, depending on whether it has considered all the ghost players or not.

If ‘min_value’ has considered all the ghost players, it calls ‘max_value’ for the next level of the tree. If not, it calls ‘min_value’ for the next ghost player in turn.

This process continues until the function has reached the maximum depth of the game tree or there are no legal actions available for the current ghost player.

After considering all the possible actions and their corresponding successor game states, ‘min_value’ returns the minimum value and corresponding best action. This value represents the best possible outcome for the current ghost player if all players play optimally.

The ‘max_value’ function works in a similar way, but instead computes the maximum value and corresponding best action that Pacman can take at a certain depth in the game tree.

Finally, the algorithm returns the best action found by calling max_value with initial depth and game state. This is the optimal move that Pacman should make at the current state of the game.

**3. How to recursively call the “max” function and “min” function?**

An overview of how the recursive calls work in the minimax algorithm:

a. The algorithm begins at the top of the game tree, with Pacman as the maximizing player.
b. Pacman's max function is called, which searches through all possible moves Pacman can make.
c. For each of Pacman's possible moves, the min function is called for the opposing player, who is trying to minimize Pacman's score.
d. The min function recursively calls the max function for Pacman, and the process repeats itself until a terminal state is reached (i.e., the game is over).
e. At each level of the game tree, the max function selects the move with the highest score, while the min function selects the move with the lowest score.

**4. How is depth=4 reached?**

The depth refers to the number of moves that Pacman and the ghosts are allowed to make ahead in the game tree while searching for the best possible move.
So whenever iteratively Pacman maximizes the algorithm and Ghosts minimizes the algorithm, the depth is increased by 1.
And hence after 4 such iterations, depth is reached at 4.

**5. How does the root node return the best action?**

In the minimax algorithm, the root node assumes that the opponent will always choose the action that is most detrimental to Pacman, and Pacman chooses the action that maximizes his own score. The algorithm recursively explores the tree until the terminating conditions are found, i.e., depth = 4, isWin(), isLose(), or when no further legal actions are found.
After that, the child(score) returns the best possible action to the parent. In such a way, the scores are then propagated back up the tree, and the best action is chosen at the root node based on the highest score.

6. **How do you set the PlayerIndex for Pacman and for three ghosts?**

   In our case, there are three ghosts, and each ghost is assigned a value of 1, 2 and 3 respectively at each iteration. In each 'min_value()' iteration, the ghost is assigned a value 1 / 2 / 3. And the Pacman is assigned the value zero in the 'max_value()' iteration.

7. **When depth, how to go from the 3rd ghost to the next player-MAX, and at the same time increase the depth by 1?**

   After each iteration of three ghosts, the depth value is increased by 1. So in each iteration, we check if the playerIndex of ghost is achieved as 3, if so we then increase the value of depth by 1, and in the next iteration we maximize Pacman.