TJ Wiegman

ASM 591 AI

Lab 5

2024-10-23

Example 1: Tic-Tac-Toe

In [1]:
```python
# Tic-Tac-Toe Game and TD Learning Tutorial

# Step 1: Define the Tic-Tac-Toe Environment
# The game board is represented as a list with 9 positions.

import numpy as np
import random

class TicTacToe:
    def __init__(self):
        # Initializes a 3x3 board filled with empty spaces
        self.board = [' '] * 9  # A 3x3 board (represented as a list)
        self.current_winner = None  # To track the winner (X or O)

    def available_moves(self):
        # Returns a list of available (empty) positions on the board
        return [i for i, x in enumerate(self.board) if x == ' ']

    def make_move(self, square, letter):
        # Places the letter (X or O) on the board if the move is valid
        if self.board[square] == ' ':
            self.board[square] = letter
            if self.winner(square, letter):
                self.current_winner = letter
            return True
        return False

    def winner(self, square, letter):
        # Checks if the player with "letter" has won (3 in a row)
        row_ind = square // 3
        row = self.board[row_ind * 3:(row_ind + 1) * 3]
        if all([s == letter for s in row]):
            return True

        col_ind = square % 3
        column = [self.board[col_ind + i * 3] for i in range(3)]
        if all([s == letter for s in column]):
            return True

        if square % 2 == 0:  # Check diagonals
            diagonal1 = [self.board[i] for i in [0, 4, 8]]
            if all([s == letter for s in diagonal1]):
                return True
            diagonal2 = [self.board[i] for i in [2, 4, 6]]
```

```
            if all([s == letter for s in diagonal2]):
                return True
        return False

    def empty_squares(self):
        # Returns True if there are any empty squares left
        return ' ' in self.board

    def num_empty_squares(self):
        # Returns the number of empty squares
        return self.board.count(' ')

    def print_board(self):
        # Prints the Tic-Tac-Toe board
        for row in [self.board[i * 3:(i + 1) * 3] for i in range(3)]:
            print('| ' + ' | '.join(row) + ' |')

    def reset(self):
        # Resets the board for a new game
        self.board = [' '] * 9
        self.current_winner = None

# Step 2: Define the TD(0) Agent that learns using Temporal Difference Learning

class TDAgent:
    def __init__(self, symbol, alpha=0.1, gamma=0.9, epsilon=0.1):
        self.symbol = symbol  # 'X' or 'O'
        self.alpha = alpha  # Learning rate
        self.gamma = gamma  # Discount factor
        self.epsilon = epsilon  # Exploration rate
        self.V = {}  # Value function (state values)
        self.history = []  # History of states visited in an episode

    def get_board_state(self, board):
        # Converts the board (list) into a string for easy hashing
        return ''.join(board)

    def choose_action(self, game):
        # Chooses the next action (move) using epsilon-greedy strategy
        available_moves = game.available_moves()

        if np.random.rand() < self.epsilon:
            # Exploration: Random move
            return random.choice(available_moves)

        # Exploitation: Choose the move that leads to the best estimated value
        best_move = None
        best_value = -float('inf')
        for move in available_moves:
            next_board = game.board.copy()
            next_board[move] = self.symbol
            next_state = self.get_board_state(next_board)
            value = self.V.get(next_state, 0.5)  # Default value for unseen states
            if value > best_value:
                best_value = value
                best_move = move
```

```
        return best_move

    def update_value_function(self, reward):
        # Updates the value function V(s) using TD(0)
        for i in reversed(range(len(self.history))):
            state = self.history[i]
            if i == len(self.history) - 1:
                # Last state (terminal): Update based on reward
                self.V[state] = self.V.get(state, 0.5) + self.alpha * (reward - se
            else:
                # Non-terminal: Bootstrap from the next state
                next_state = self.history[i + 1]
                self.V[state] = self.V.get(state, 0.5) + self.alpha * (
                    self.gamma * self.V.get(next_state, 0.5) - self.V.get(state, 0
                )

    def add_state_to_history(self, state):
        # Adds the current state to the agent's history for updating later
        self.history.append(state)

    def reset_history(self):
        # Resets the agent's history after each game
        self.history = []

# Step 3: Define a Random Opponent (for training purposes)
# The RandomOpponent will take random moves to challenge the agent.

class RandomOpponent:
    def __init__(self, symbol):
        self.symbol = symbol

    def choose_action(self, game):
        # Randomly choose an available move
        return random.choice(game.available_moves())

# Step 4: Define the function to simulate a game between the TD agent and the oppo

def play_game(agent_x, agent_o, game, print_game=False):
    # Resets the game and the TD agent's history
    game.reset()
    agent_x.reset_history()  # Only TD agent needs to reset history

    players = [agent_x, agent_o]
    current_player = random.choice(players)  # Randomly pick who goes first

    while game.empty_squares():
        # Choose and make a move
        action = current_player.choose_action(game)
        game.make_move(action, current_player.symbol)

        # Track states for the TD agent
        if current_player == agent_x:
            agent_x.add_state_to_history(agent_x.get_board_state(game.board))

        # Print the board (optional)
        if print_game:
```

```
            game.print_board()
            print()

        if game.current_winner:
            # If there's a winner, update values
            if current_player == agent_x:
                agent_x.update_value_function(1)  # Agent X wins
            else:
                agent_x.update_value_function(-1)  # Agent X loses
            return current_player.symbol

        # Switch to the other player
        current_player = agent_x if current_player == agent_o else agent_o

    # It's a draw
    agent_x.update_value_function(0)  # Draw for TD agent
    return "Draw"


# Step 5: Train the TD Agent by playing multiple games

def train(agent_x, agent_o, game, episodes=10000, print_game=False):
    wins = {agent_x.symbol: 0, agent_o.symbol: 0, "Draw": 0}

    for episode in range(episodes):
        result = play_game(agent_x, agent_o, game, print_game=print_game)
        wins[result] += 1

        if episode % 1000 == 0:
            print(f"Episode {episode}: {wins}")

    print(f"Final results after {episodes} games: {wins}")

# Initialize the game and agents
game = TicTacToe()
agent_x = TDAgent(symbol='X', alpha=0.1, gamma=0.9, epsilon=0.1)
agent_o = RandomOpponent(symbol='O')

# Train the agent over 10,000 episodes
train(agent_x, agent_o, game, episodes=200)
```

```
Episode 0: {'X': 0, 'O': 1, 'Draw': 0}
Final results after 200 games: {'X': 95, 'O': 81, 'Draw': 24}
```

In [2]:

```
# Step 6: Evaluate the agent's performance after training

def evaluate_agent(agent_x, opponent, game, episodes=100, print_game=False):
    wins = {agent_x.symbol: 0, opponent.symbol: 0, "Draw": 0}

    for episode in range(episodes):
        result = play_game(agent_x, opponent, game, print_game=print_game)
        wins[result] += 1

    print(f"Evaluation results after {episodes} games: {wins}")
```

```
# Evaluate the agent by playing against a random opponent
evaluate_agent(agent_x, RandomOpponent('O'), game, episodes=100)
```

```
Evaluation results after 100 games: {'X': 61, 'O': 32, 'Draw': 7}
```

Problem 1: Grid World

In [3]:

```
import numpy as np

class Gridworld:
    def __init__(self, size=5, start=(0, 0), goal=(4, 4), penalty_cells=[]):
        self.size = size  # Grid is size x size
        self.start = start  # Starting position
        self.goal = goal  # Goal position
        self.penalty_cells = penalty_cells  # Cells with penalties

        self.grid = np.zeros((size, size))
        self.grid[goal] = 1  # Goal cell gives reward of +1
        for penalty in penalty_cells:
            self.grid[penalty] = -1  # Penalty cells give reward of -1

        self.reset()

    def reset(self):
        # Reset the environment, start from the initial state
        self.agent_position = self.start
        return self.agent_position

    def step(self, action):
        # Take a step in the environment
        # Actions: 0 = up, 1 = right, 2 = down, 3 = left
        moves = [(-1, 0), (0, 1), (1, 0), (0, -1)]  # (row, col) movement

        # Get current position
        old_row, old_col = self.agent_position
        new_row = old_row + moves[action][0]
        new_col = old_col + moves[action][1]

        # Boundary conditions
        bonk = 0
        if new_row < 0 or new_row >= self.size:
            new_row = max(new_row, 0)
            new_row = min(new_row, self.size-1)
            bonk += 1
        if new_col < 0 or new_col >= self.size:
            new_col = max(new_col, 0)
            new_col = min(new_col, self.size-1)
            bonk += 1

        # Save new position
        self.agent_position = (new_row, new_col)

        # Discourage running into walls
        reward = self.grid[self.agent_position] - bonk*0.1
```

```
        # Episode ends when goal is reached
        if (self.agent_position == self.goal): done = True
        else: done = False

        return self.agent_position, reward, done

    def available_actions(self):
        # There are always 4 possible actions (up, right, down, left)
        return [0, 1, 2, 3]

    def render(self):
        # Print the grid and agent's position
        grid_display = np.copy(self.grid)
        grid_display[self.agent_position] = 2  # Mark agent's position as 2
        print(grid_display)
```

In [18]:

```
class TDAgent:
    def __init__(self, gridworld, alpha=0.1, gamma=0.9, epsilon=0.1):
        self.gridworld = gridworld
        self.alpha = alpha  # Learning rate
        self.gamma = gamma  # Discount factor
        self.epsilon = epsilon  # Exploration rate
        self.V = np.zeros((gridworld.size, gridworld.size))  # Value function (sta
        self.history = []  # History of states visited in an episode

    def choose_action(self):
        # Epsilon-greedy action selection
        if np.random.rand() < self.epsilon:
            return  random.choice(self.gridworld.available_actions())
        else:
            best_reward = -float("inf")
            for move in self.gridworld.available_actions():
                _, reward, done = self.simulate_action(move)
                if done:
                    return move
                elif reward > best_reward:
                    best_reward = reward
                    best_action = move
            return best_action

    def simulate_action(self, action):
        # Simulate taking an action (without changing the environment)
        current_pos = self.gridworld.agent_position
        next_pos, reward, done = self.gridworld.step(action)
        self.gridworld.agent_position = current_pos  # Reset the position back
        return next_pos, reward, done

    def update_value_function(self, reward, next_state):
        # TD(0) update rule
        current_state = self.gridworld.agent_position
        self.V[current_state] += self.alpha * (reward + self.gamma * self.V[next_s

    def reset(self):
        self.gridworld.reset()  # Reset the gridworld
        self.history = []  # Reset agent history
```

```
    def learn(self, episodes=1000):
        for episode in range(episodes):
            print(f"Training {episode}/{episodes}...", end="")
            state = self.gridworld.reset()
            done = False
            while not done:
                action = self.choose_action()
                next_state, reward, done = self.gridworld.step(action)
                self.update_value_function(reward, next_state)

                # Optionally: Display the grid during learning
                # self.gridworld.render()
                if done:
                    break
            print("                        ", end="\r")
```

In [19]:
```
# Create a Gridworld environment
gridworld = Gridworld(size=5, start=(0, 0), goal=(4, 4), penalty_cells=[(1, 1), (3

# Initialize the TD Agent
agent = TDAgent(gridworld, alpha=0.1, gamma=0.9, epsilon=0.1)

# Train the agent for 1000 episodes
agent.learn(episodes=1000)

# After learning, let's visualize the learned value function
print("Learned Value Function (State Values):")
print(agent.V)
```

```
Learned Value Function (State Values):
[[-0.38249657 -0.02307558 -0.03797931 -0.01624941 -0.04608191]
 [ 0.         -4.69094457  0.          0.         -0.01574417]
 [ 0.          0.          0.          0.         -0.02282826]
 [ 0.          0.          0.         -6.18952882 -0.04988883]
 [ 0.          0.          0.          0.          9.99956829]]
```

In [33]:
```
# Function to evaluate and visualize the agent's performance after training

def evaluate_agent(agent, gridworld):
    gridworld.reset()
    done = False
    total_reward = 0
    n_step = 0
    gridworld.render()  # Visualize starting point
    while not done:
        action = agent.choose_action()
        next_state, reward, done = gridworld.step(action)
        gridworld.render()  # Visualize each step
        n_step += 1
        total_reward += reward
    print(f"Total reward: {total_reward}, gathered in {n_step} steps")

# Evaluate and visualize agent's performance
evaluate_agent(agent, gridworld)
```

```
[[ 2.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  2.  0.  0.  0.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  2.  0.  0.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  2.  0.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  2.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  2.  0.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
```

```
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  2.  0.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  0.]
```

```
 [ 0.  0.  0.  0.  2.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
```

```
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
```

```
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  2.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  2.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  2.  0.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
```

```
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
```

```
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
```

```
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  2.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
```

```
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
```

```
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  2.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  2.  0.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
```

```
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
```

```
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
```

```
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
```

```
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
```

```
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  2.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  2.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  1.]]
```

```
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  2.]
 [ 0.  0.  0.  0.  1.]]
[[ 0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  2.]]
Total reward: 0.7, gathered in 169 steps
```