DRL HW1 說明報告

1. 作業概述

本作業的目標是開發一個 Flask 網頁應用程式，實作一個網格地圖（Gridworld），允許使用者指定維度 （範圍從 5 到 9），並透過互動設定起始格、終點格及障礙物。此外，還需顯示策略矩陣（Policy Matrix）及價值矩陣（Value Matrix），並利用策略評估推導各狀態的價值。

本作業分為兩個部分：

- HW1-1: 網格地圖開發（60%）
- HW1-2: 策略顯示與價值評估（40%）

2. HW1-1 網格地圖開發

2.1 功能實作

(1) 網格生成與動態設定

- 使用者可以輸入數值（介於 5 到 9），生成對應大小的網格。
- 使用 AJAX 進行非同步請求，後端 Flask 會根據使用者輸入更新網格大小。

(2) 起始點、終點與障礙物設置

- 點擊一個單元格可將其標記為起始格（綠色）。
- 再次點擊另一單元格可將其標記為終點格（紅色）。
- 使用者最多可標記 個障礙物（灰色）。
- 單元格可以重複點擊來取消設定。

2.2 程式碼設計

後端 (Flask) 設計

- `set_size()` 方法處理網格大小變更。
- 變數 `GRID_SIZE`、`START_CELL`、`END_CELL`、`OBSTACLES` 負責存儲當前的網格設定。

前端 (HTML + JavaScript) 設計

- `generateGrid()` 負責創建網格。
- `handleCellClick()` 處理單元格點擊事件，根據不同條件更新格子顏色。
- `startGame()` 發送請求到 Flask，取得策略與價值矩陣。

3. HW1-2 策略顯示與價值評估

3.1 功能實作

(1) 策略矩陣 (Policy Matrix) 生成

- 針對非終點與障礙物的單元格，隨機選擇一組合法動作（↑ ↓ ← →）。
- 確保選擇的動作不會超出網格範圍或進入障礙物。

(2) 價值矩陣 (Value Matrix) 計算

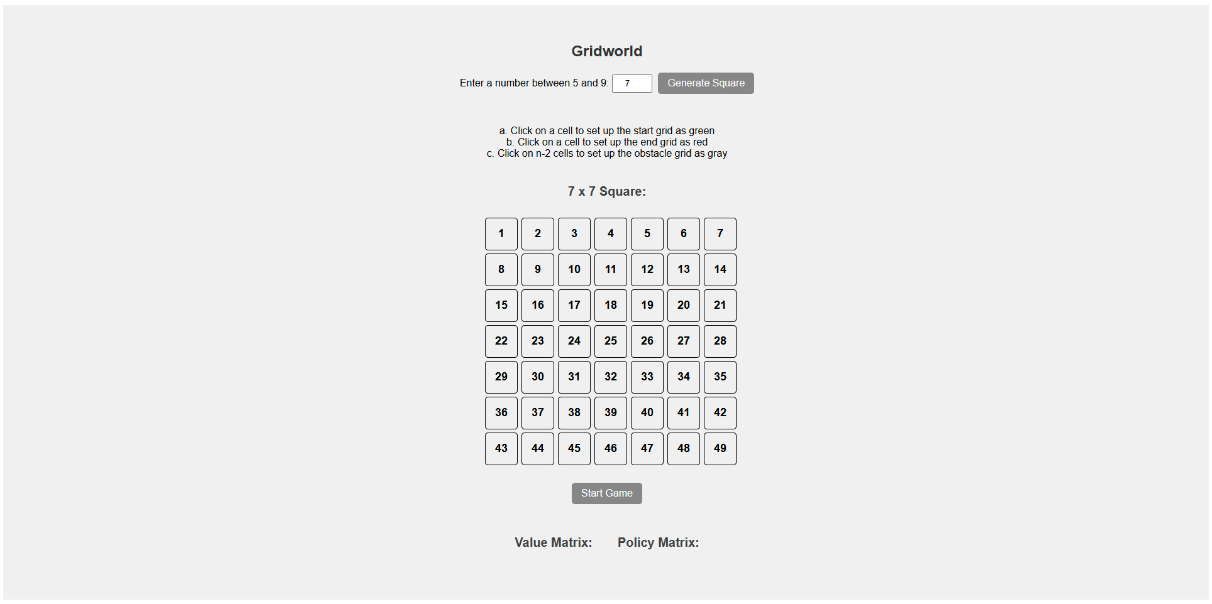
- 終點格的獎勵設為 `1.0`，障礙物的價值設為 `-1.0`。
- 其他格子的初始值在 `[-0.5, 0.5]` 之間隨機分配。

4. 成果截圖與程式碼

4.1 成果截圖

以下為 Gridworld 介面執行結果的截圖：

HW1-1: Result



Gridworld

Enter a number between 5 and 9: 7

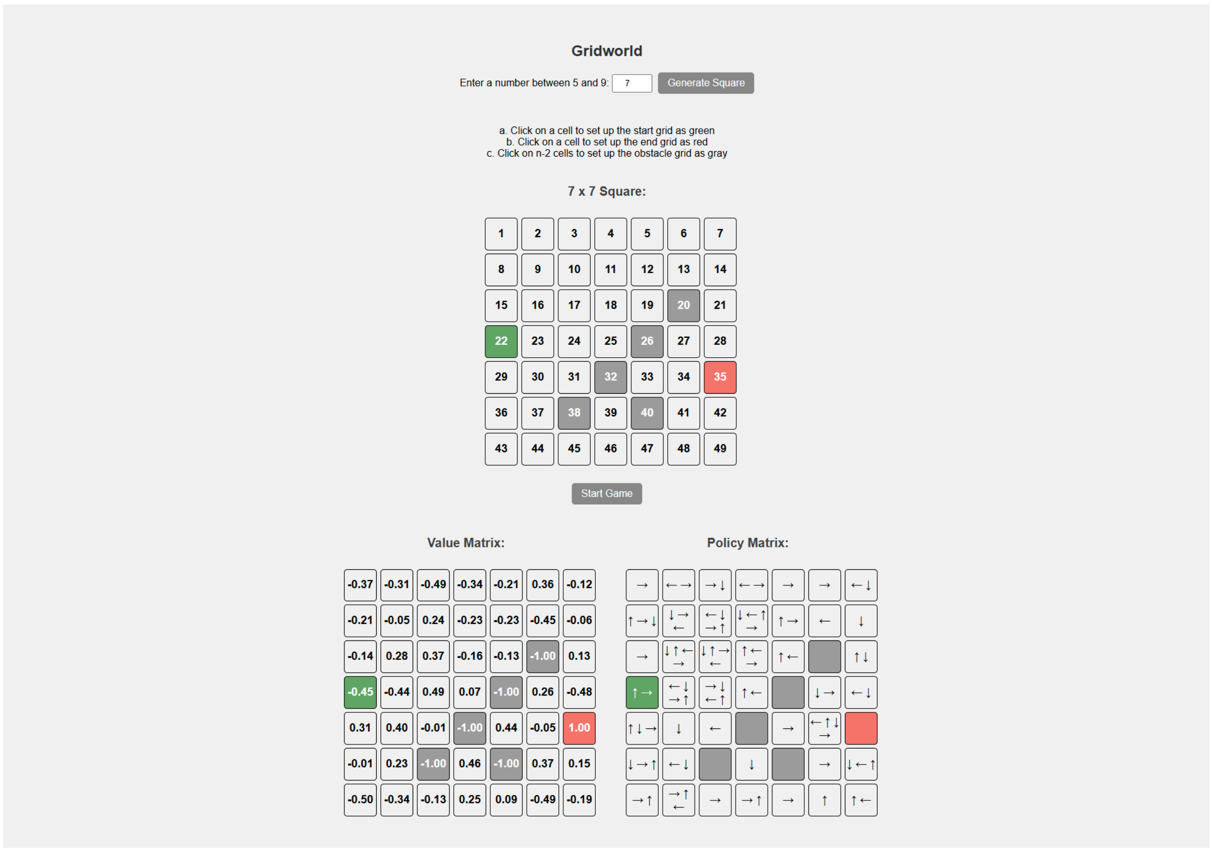
a. Click on a cell to set up the start grid as green
b. Click on a cell to set up the end grid as red
c. Click on n-2 cells to set up the obstacle grid as gray

7 x 7 Square:

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49

Value Matrix: Policy Matrix:

HW1-2: Result


Gridworld

Enter a number between 5 and 9: 7 Generate Square

a. Click on a cell to set up the start grid as green
b. Click on a cell to set up the end grid as red
c. Click on n-2 cells to set up the obstacle grid as gray

7 x 7 Square:

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49

Start Game

Value Matrix:

-0.37	-0.31	-0.49	-0.34	-0.21	0.36	-0.12
-0.21	-0.05	0.24	-0.23	-0.23	-0.45	-0.06
-0.14	0.28	0.37	-0.16	-0.13	-1.00	0.13
-0.45	-0.44	0.49	0.07	-1.00	0.26	-0.48
0.31	0.40	-0.01	-1.00	0.44	-0.05	1.00
-0.01	0.23	-1.00	0.46	-1.00	0.37	0.15
-0.50	-0.34	-0.13	0.25	0.09	-0.49	-0.19

Policy Matrix:

→	←→	→↓	←→	→	→	←↓
↑→↓	↓→←	←↓→↑	↓←↑→	↑→	←	↓
→	↓↑←→	↓↑→←	↑←→	↑←	▨	↑↓
↑→	←↓→↑	→↓←↑	↑←	▨	↓→	←↓
↑↓→	↓	←	▨	→	←↑↓→	🟥
↓→↑	←↓	▨	↓	▨	→	↓←↑
→↑	→↑←	→	→↑	→	↑	↑←

4.2 程式碼

後端程式碼 (Flask - `app.py`)

```
from flask import Flask, render_template, request, jsonify
import numpy as np
import random

app = Flask(__name__)

GRID_SIZE = 5
START_CELL = None
END_CELL = None
OBSTACLES = set()

ACTIONS = {
    "↑": (-1, 0),
    "↓": (1, 0),
    "←": (0, -1),
    "→": (0, 1)
}

@app.route('/')
```

```
def index():
    return render_template('index.html', grid_size=GRID_SIZE)

@app.route('/set_size', methods=['POST'])
def set_size():
    global GRID_SIZE, START_CELL, END_CELL, OBSTACLES
    try:
        data = request.get_json()
        size = data.get("size")

        if 5 <= size <= 9:
            GRID_SIZE = size
            START_CELL = None
            END_CELL = None
            OBSTACLES = set()
            return jsonify({"status": "success", "grid_size": GRID_SIZE})

        return jsonify({"status": "error", "message": "Grid size must be between 5 and 9"}),
400

    except Exception as e:
        return jsonify({"status": "error", "message": str(e)}), 500

@app.route('/find_best_path', methods=['POST'])
def find_best_path():
    """隨機產生 Value Matrix 和 Policy Matrix，並確保不會超出範圍"""
    global GRID_SIZE, START_CELL, END_CELL, OBSTACLES

    try:
        data = request.get_json()
        if not data or 'start' not in data or 'end' not in data or 'obstacles' not in data or 'grid_s
ize' not in data:
            return jsonify({"status": "error", "message": "Missing required parameters"}), 400

        start = tuple(data['start'])
        end = tuple(data['end'])
        obstacles = set(tuple(obs) for obs in data['obstacles'])
        GRID_SIZE = data['grid_size']  # 讀取前端傳來的 grid_size

        # 產生隨機 Value Matrix
        values = np.random.uniform(-0.5, 0.5, (GRID_SIZE, GRID_SIZE))
        values[end] = 1.0  # 設定終點獎勵
        for obs in obstacles:
            values[obs] = -1.0  # 設定障礙物
        # 產生隨機 Policy Matrix，確保不會超出範圍
        policy = np.full((GRID_SIZE, GRID_SIZE), " ", dtype=object)
        for r in range(GRID_SIZE):
            for c in range(GRID_SIZE):
                if (r, c) == end or (r, c) in obstacles:
                    continue

                valid_actions = []
                for action, (dr, dc) in ACTIONS.items():
                    nr, nc = r + dr, c + dc
                    if 0 <= nr < GRID_SIZE and 0 <= nc < GRID_SIZE and (nr, nc) not in obstacles:
                        valid_actions.append(action)

                if valid_actions:
                    num_actions = random.randint(1, len(valid_actions))  # 隨機選擇 1~所有合法動
作
                    policy[r, c] = "".join(random.sample(valid_actions, num_actions))

        return jsonify({
            "status": "success",
            "value_matrix": values.tolist(),
            "policy_matrix": policy.tolist()
        })

    except Exception as e:
        return jsonify({"status": "error", "message": str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```

前端程式碼 (`index.html`)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Gridworld</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            text-align: center;
            margin: 40px;
            background-color: #f4f4f4;
```

```css
}

h2 {
    font-size: 24px;
    font-weight: bold;
    color: #333;
}

h3 {
    font-size: 20px;
    font-weight: bold;
    color: #444;
}

input[type="number"] {
    width: 50px;
    text-align: center;
    padding: 5px;
    margin-right: 5px;
}

button {
    font-size: 16px;
    background-color: #888888;
    color: white;
    border: none;
    padding: 8px 15px;
    cursor: pointer;
    border-radius: 5px;
    transition: background 0.3s ease;
}

button:hover {
    background-color: #5a5a5a;
}

.grid-container {
    display: grid;
    column-gap: 18px; /* 只調整水平間距 */
    row-gap: 5px;    /* 可選：控制垂直間距 */
    margin-top: 10px;
    padding: 10px;
    border-collapse: collapse;
}
```

```css
.grid-item {
    width: 50px;
    height: 50px;
    display: flex;
    justify-content: center;
    align-items: center;
    border: 1px solid black;
    font-size: 18px;
    cursor: pointer;
    font-weight: bold;
    transition: all 0.2s ease;
    border-radius: 5px; /* 設置圓角，數值越大圓角越明顯 */
}
.grid-item:hover {
    transform: scale(1.1);
}

#grid-container {
    display: flex;
    justify-content: center;
    margin-top: 20px;
}

#matrix-container {
    display: flex;
    justify-content: center;
    align-items: flex-start;
    gap: 40px;
    margin-top: 30px;
}

@media (max-width: 768px) {
    #matrix-container {
        flex-direction: column;
        align-items: center;
    }
}

.start {
    background-color: #64a866;
    color: white;
}

.end {
    background-color: #f5736a;
```

```
        color: white;
      }

      .obstacle {
        background-color: #9E9E9E;
        color: white;
      }

    </style>
</head>
<body>

<h2>Gridworld</h2>
<p>Enter a number between 5 and 9:
    <input type="number" id="grid-size" min="5" max="9">
    <button onclick="generateGrid()">Generate Square</button>
</p><br>
<p>
    a. Click on a cell to set up the start grid as green <br>
    b. Click on a cell to set up the end grid as red <br>
    c. Click on n-2 cells to set up the obstacle grid as gray
</p>

<!-- Gridworld 區域 -->
<div id="grid-container">
    <div>
      <h3><span id="grid-title">5 × 5 Square:</span></h3>
      <div id="grid" class="grid-container"></div><br>
      <button id="start-game" onclick="startGame()">Start Game</button>
    </div>
</div>

<!-- Value Matrix 和 Policy Matrix -->
<div id="matrix-container">
    <div>
      <h3>Value Matrix:</h3>
      <div id="value-matrix" class="grid-container"></div>
    </div>
    <div>
      <h3>Policy Matrix:</h3>
      <div id="policy-matrix" class="grid-container"></div>
    </div>
</div>

<script>
```

```
let gridSize = 5;
let startCell = null;
let endCell = null;
let obstacles = new Set();

function generateGrid() {
    gridSize = parseInt(document.getElementById("grid-size").value);
    if (gridSize < 5 || gridSize > 9 || isNaN(gridSize)) {
        alert("Please enter a valid number between 5 and 9.");
        return;
    }

    $.ajax({
        url: "/set_size",
        type: "POST",
        contentType: "application/json",
        data: JSON.stringify({ size: gridSize }),
        success: function(response) {
            if (response.status === "success") {
                startCell = null;
                endCell = null;
                obstacles.clear();
                renderGrid();
            } else {
                alert(response.message);
            }
        },
        error: function(xhr) {
            console.log("AJAX Error:", xhr.responseText);
            alert("Error setting grid size.");
        }
    });
}

function renderGrid() {
    let grid = document.getElementById("grid");
    grid.innerHTML = "";
    grid.style.gridTemplateColumns = `repeat(${gridSize}, 40px)`;

    for (let row = 0; row < gridSize; row++) {
        for (let col = 0; col < gridSize; col++) {
            let cell = document.createElement("div");
            cell.classList.add("grid-item");
            cell.textContent = row * gridSize + col + 1;
            cell.dataset.row = row;
            cell.dataset.col = col;
            cell.addEventListener("click", () => handleCellClick(cell));
            grid.appendChild(cell);
        }
    }
    document.getElementById("grid-title").textContent = `${gridSize} x ${gridSize} Square:`;
}

function handleCellClick(cell) {
    let row = parseInt(cell.dataset.row);
    let col = parseInt(cell.dataset.col);
    let cellPos = [row, col];

    if (startCell && startCell[0] === row && startCell[1] === col) {
        startCell = null;
        cell.className = "grid-item";
    } else if (endCell && endCell[0] === row && endCell[1] === col) {
        endCell = null;
        cell.className = "grid-item";
    } else if (obstacles.has(JSON.stringify(cellPos))) {
        obstacles.delete(JSON.stringify(cellPos));
        cell.className = "grid-item";
    } else if (!startCell) {
        startCell = cellPos;
        cell.className = "grid-item start";
    } else if (!endCell) {
        endCell = cellPos;
        cell.className = "grid-item end";
    } else if (obstacles.size < gridSize - 2) {
        obstacles.add(JSON.stringify(cellPos));
        cell.className = "grid-item obstacle";
    } else {
        alert(`You can only place up to ${gridSize - 2} obstacles.`);
    }
}

function startGame() {
    if (!startCell || !endCell) {
        alert("Please select both a start and end cell.");
        return;
    }

    let obstacleList = Array.from(obstacles).map(JSON.parse);

    $.ajax({
        url: "/find_best_path",
        type: "POST",
        contentType: "application/json",
        data: JSON.stringify({
            start: startCell,
            end: endCell,
            obstacles: obstacleList,
            grid_size: gridSize  // ✅ 傳遞最新 gridSize
        }),
        success: function(response) {
            if (response.status === "success") {
                renderMatrix("value-matrix", response.value_matrix, false);
                renderMatrix("policy-matrix", response.policy_matrix, true);
            } else {
                alert(response.message);
            }
        },
        error: function(xhr) {
            alert("Error calculating matrices.");
        }
    });
}

function renderMatrix(id, matrix, isPolicy) {
    let container = document.getElementById(id);
    container.innerHTML = "";
    container.style.gridTemplateColumns = `repeat(${gridSize}, 40px)`;

    for (let row = 0; row < gridSize; row++) {
        for (let col = 0; col < gridSize; col++) {
            let cell = document.createElement("div");
            cell.classList.add("grid-item");

            // ✅ 保留格子顏色 (start, end, obstacle)
            let cellPos = JSON.stringify([row, col]);
            if (startCell && startCell[0] === row && startCell[1] === col) {
                cell.classList.add("start");
            } else if (endCell && endCell[0] === row && endCell[1] === col) {
                cell.classList.add("end");
            } else if (obstacles.has(cellPos)) {
                cell.classList.add("obstacle");
            }

            // ✅ 設定數值或箭頭
            if (isPolicy) {
                let actions = matrix[row][col];
                cell.innerHTML = actions.replace(/↑/g, "↑ ")
                                        .replace(/↓/g, "↓ ")
                                        .replace(/←/g, "← ")
                                        .replace(/→/g, "→ ");
            } else {
                cell.textContent = matrix[row][col].toFixed(2);
            }

            container.appendChild(cell);
        }
    }
}

window.onload = renderGrid;
</script>

</body>
</html>
```