HW1-1 & 1-2 & 2:網格地圖開發

系級:資工碩一 學號:7113056041 姓名:蔡承晏

1. 目標：開發一個大小為 nxn 的網格地圖，允許用戶指定維度 'n'（範圍從 5 到 9），使用 Flask 網頁應用程式。

2.功能要求：用戶可以透過滑鼠點擊指定起始單元格（點擊後顯示為綠色）和結束單元格（點擊後顯示為紅色）。

用戶可以設定 n-2 個障礙物，透過滑鼠點擊將這些單元格變為灰色。

app.py

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

app = Flask(__name__)

# 固定參數
GAMMA = 0.9  # 折扣因子
theta = 0.01  # 收斂條件
actions = {'↑': (-1, 0), '↓': (1, 0), '←': (0, -1), '→': (0, 1)}

def value_iteration(grid_size, start, end, obstacles):
    V = np.zeros((grid_size, grid_size))
    policy = np.full((grid_size, grid_size), '', dtype=object)

    while True:
        delta = 0
        for x in range(grid_size):
            for y in range(grid_size):
                if (x, y) in obstacles or (x, y) == end:
                    continue  # 障礙物與終點不更新

                v_old = V[x, y]
                best_value = float('-inf')
                best_actions = []

                for action, (dx, dy) in actions.items():
                    nx, ny = x + dx, y + dy
                    if 0 <= nx < grid_size and 0 <= ny < grid_size and (nx, ny) not in obstacles:
                        reward = 1 if (nx, ny) == end else 0  # 終點獲得獎勵
                        new_value = reward + GAMMA * V[nx, ny]
                        if new_value > best_value:
                            best_value = new_value
                            best_actions = [action]  # 更新最佳動作
                        elif new_value == best_value:
                            best_actions.append(action)  # 記錄多個最優動作

                V[x, y] = best_value if best_value != float('-inf') else 0
                policy[x, y] = ''.join(best_actions) if best_actions else ' '
                delta = max(delta, abs(v_old - V[x, y]))

        if delta < theta:
            break

    return V.tolist(), policy.tolist()
```

```python
def traversal_path(grid_size, start, end, obstacles):
    visited = set()
    path = []

    def dfs(x, y):
        if (x, y) in visited or (x, y) in obstacles or x < 0 or x >= grid_size or y < 0 or y >= grid_size:
            return False
        visited.add((x, y))
        path.append((x, y))

        if (x, y) == end:
            return True

        for dx, dy in actions.values():
            if dfs(x + dx, y + dy):
                return True

        return False

    dfs(start[0], start[1])
    return path

@app.route('/', methods=['GET', 'POST'])
def index():
    n = 5
    if request.method == 'POST':
        n = int(request.form.get('n', 5))
        if n < 3 or n > 10:
            n = 5
    return render_template('index.html', n=n)

@app.route('/compute_policy', methods=['POST'])
def compute_policy():
    data = request.json
    grid_size = data['grid_size']
    start = tuple(data['start'])
    end = tuple(data['end'])
    obstacles = {tuple(obs) for obs in data['obstacles']}

    value_matrix, policy_matrix = value_iteration(grid_size, start, end, obstacles)
    return jsonify({'value_matrix': value_matrix, 'policy_matrix': policy_matrix})
```

```python
@app.route('/compute_traversal', methods=['POST'])
def compute_traversal():
    data = request.json
    grid_size = data['grid_size']
    start = tuple(data['start'])
    end = tuple(data['end'])
    obstacles = {tuple(obs) for obs in data['obstacles']}

    traversal_path_list = traversal_path(grid_size, start, end, obstacles)
    return jsonify({'traversal_path': traversal_path_list})

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

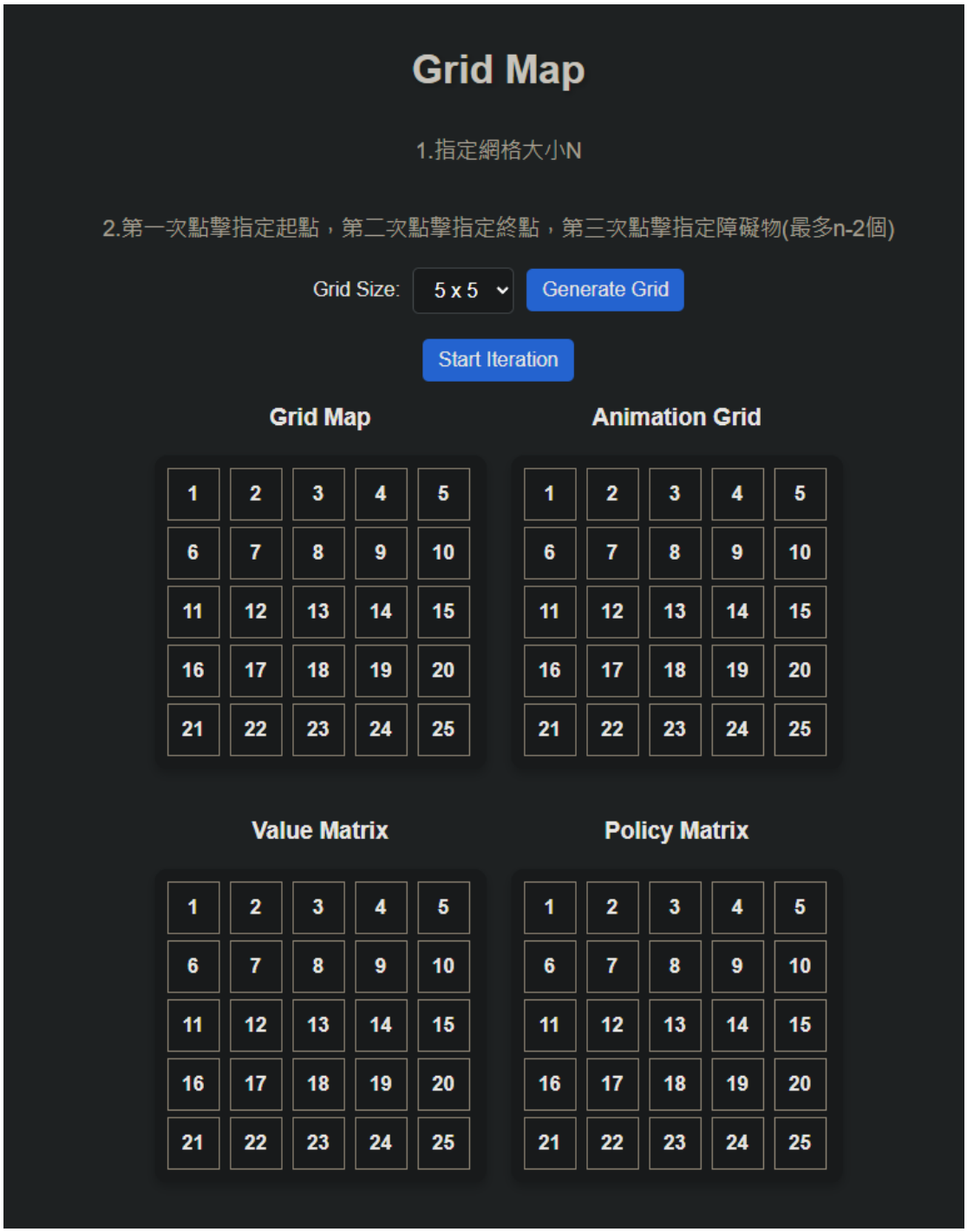
index.html

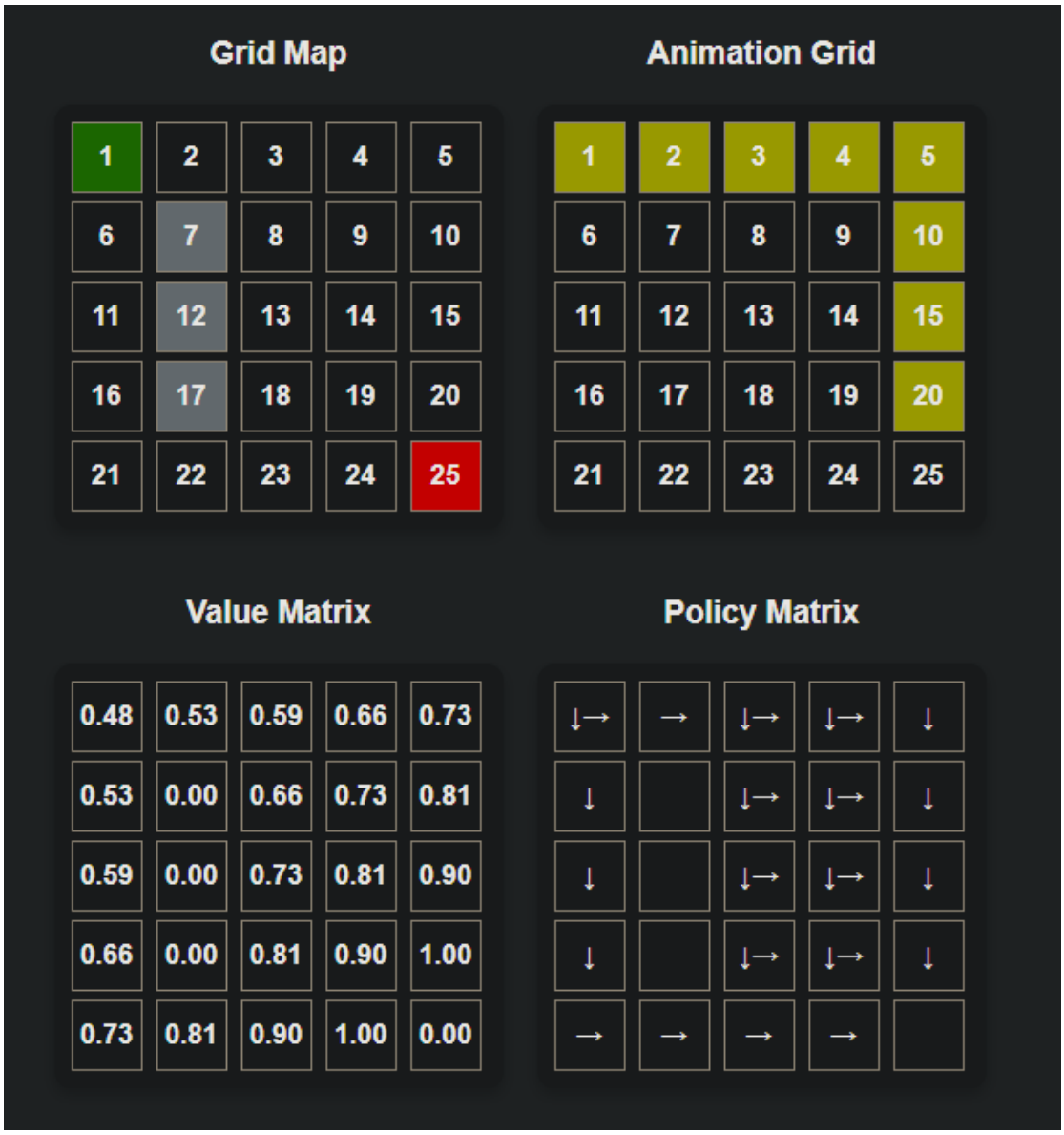
```html
<body>
    <h1>Grid Map</h1>
    <p>
        1.指定網格大小N
    </p>
    <p>
        2.第一次點擊指定起點，第二次點擊指定終點，第三次點擊指定障礙物(最多n-2個)
    </p>

    <form method="post" id="gridForm">
        <label for="n">Grid Size:</label>
        <select name="n" id="gridSize">
            <option value="5">5 x 5</option>
            <option value="6">6 x 6</option>
            <option value="7">7 x 7</option>
            <option value="8">8 x 8</option>
            <option value="9">9 x 9</option>
        </select>
        <button type="submit">Generate Grid</button>
    </form>

    <button id="computeButton">Start Iteration</button>

    <div class="grid-wrapper">
        <!-- 第一排：User Input Grid 和 Animation Grid -->
        <div class="grid-section">
            <h3>Grid Map</h3>
            <div id="grid" class="grid-container"></div>
        </div>
        <div class="grid-section">
            <h3>Animation Grid</h3>
            <div id="animationGrid" class="grid-container"></div>
        </div>

        <!-- 第二排：Value Matrix 和 Policy Matrix -->
        <div class="grid-section">
            <h3>Value Matrix</h3>
            <div id="valueGrid" class="grid-container"></div>
        </div>
        <div class="grid-section">
            <h3>Policy Matrix</h3>
            <div id="policyGrid" class="grid-container"></div>
        </div>
    </div>
```

```
<script>
    // 儲存使用者選擇的 n
    const gridSizeSelect = document.getElementById("gridSize");
    if (localStorage.getItem("selectedGridSize")) {
        gridSizeSelect.value = localStorage.getItem("selectedGridSize");
    }

    gridSizeSelect.addEventListener("change", function () {
        localStorage.setItem("selectedGridSize", this.value);
    });

    // 取得 n 的數值
    const n = parseInt(gridSizeSelect.value);
    const grid = document.getElementById("grid");
    const valueGrid = document.getElementById("valueGrid");
    const policyGrid = document.getElementById("policyGrid");
    const animationGrid = document.getElementById("animationGrid");

    function createGrid(container) {
        container.innerHTML = ""; // 清除舊的 grid
        container.style.gridTemplateColumns = `repeat(${n}, 45px)`;
        let count = 1; // 初始數字
        for (let row = 0; row < n; row++) {
            for (let col = 0; col < n; col++) {
                const cell = document.createElement("div");
                cell.classList.add("cell");
                cell.dataset.row = row;
                cell.dataset.col = col;

                // 用 <span> 來放數字，避免影響點擊
                const number = document.createElement("span");
                number.textContent = count++;
                cell.appendChild(number);

                container.appendChild(cell);
            }
        }
    }

```

```
        function animatePath(container, policyMatrix) {
            let [row, col] = start;

            function step() {
                if (row === end[0] && col === end[1]) return;

                const direction = policyMatrix[row][col];
                if (!direction) return;

                container.querySelector(`.cell[data-row='${row}'][data-col='${col}']`).style.backgroundColor = 'yellow';

                const [dx, dy] = direction === '↑' ? [-1, 0] : direction === '↓' ? [1, 0] : direction === '←' ? [0, -1] : [0, 1];
                row += dx;
                col += dy;

                setTimeout(step, 500);
            }
            step();
        }

        function displayResults(container, matrix, isPolicy = false) {
            container.querySelectorAll(".cell").forEach((cell) => {
                const row = parseInt(cell.dataset.row);
                const col = parseInt(cell.dataset.col);
                cell.textContent = isPolicy
                    ? matrix[row][col]
                    : matrix[row][col].toFixed(2);
            });
        }
    </script>
</body>
```

網頁截圖


Grid Map
1.指定網格大小N
2.第一次點擊指定起點，第二次點擊指定終點，第三次點擊指定障礙物(最多n-2個)
Grid Size:
5 x 5
Generate Grid
Start Iteration
Grid Map
Animation Grid
Value Matrix
Policy Matrix
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


Grid Map
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
Animation Grid
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
Value Matrix
0.48 0.53 0.59 0.66 0.73
0.53 0.00 0.66 0.73 0.81
0.59 0.00 0.73 0.81 0.90
0.66 0.00 0.81 0.90 1.00
0.73 0.81 0.90 1.00 0.00
Policy Matrix
↓→ → ↓→ ↓→ ↓
↓ ↓→ ↓→ ↓
↓ ↓→ ↓→ ↓
↓ ↓→ ↓→ ↓
→ → → →