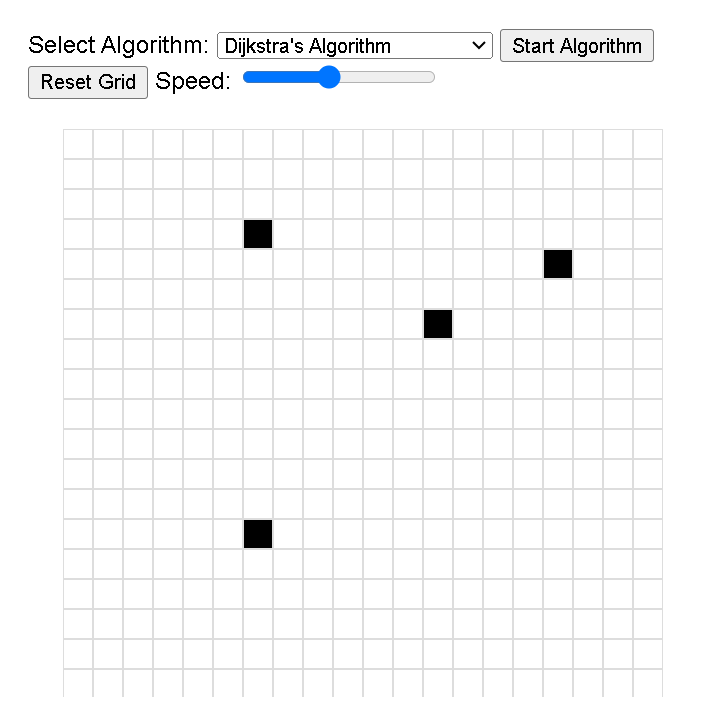
**Pathfinding Visualizer**



/\*index.html\*/

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Pathfinding Visualizer</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div id="controls">

<label for="algorithms">Select Algorithm:</label>

<select id="algorithms">

<option value="bfs">Breadth-First Search (BFS)</option>

<option value="dfs">Depth-First Search (DFS)</option>

<option value="dijkstra">Dijkstra's Algorithm</option>

<option value="aStar">A\* Algorithm</option>

<option value="greedy">Greedy Best-First Search</option>

<option value="biDir">Bidirectional Search</option>

</select>

<button id="start">Start Algorithm</button>

<button id="reset">Reset Grid</button>

<label for="speed">Speed:</label>

<input type="range" id="speed" min="1" max="10" value="5">

</div>

<div id="grid"></div>

<script src="script.js"></script>

</body>

</html>

/\* styles.css \*/

body {

font-family: Arial, sans-serif;

display: flex;

flex-direction: column;

align-items: center;

margin: 0;

padding: 0;

}

#controls {

margin: 20px;

}

#grid {

display: grid;

grid-template-columns: repeat(20, 20px); /\* Adjust as needed \*/

grid-template-rows: repeat(20, 20px); /\* Adjust as needed \*/

}

.cell {

width: 20px;

height: 20px;

border: 1px solid #ddd;

box-sizing: border-box;

}

.cell.obstacle {

background-color: black;

}

.cell.start {

background-color: green;

}

.cell.end {

background-color: red;

}

.cell.path {

background-color: blue;

}

.cell.visited {

background-color: lightblue;

}

// script.js

const gridSize = 20;

const grid = document.getElementById('grid');

const algorithms = document.getElementById('algorithms');

const startButton = document.getElementById('start');

const resetButton = document.getElementById('reset');

const speedControl = document.getElementById('speed');

let cells = [];

let startNode = null;

let endNode = null;

// Create the grid

function createGrid() {

grid.innerHTML = '';

cells = [];

for (let i = 0; i < gridSize; i++) {

const row = [];

for (let j = 0; j < gridSize; j++) {

const cell = document.createElement('div');

cell.className = 'cell';

cell.dataset.row = i;

cell.dataset.col = j;

cell.addEventListener('click', () => handleCellClick(i, j));

grid.appendChild(cell);

row.push(cell);

}

cells.push(row);

}

}

// Handle cell clicks

function handleCellClick(row, col) {

if (startNode) {

cells[startNode.row][startNode.col].classList.remove('start');

}

if (endNode) {

cells[endNode.row][endNode.col].classList.remove('end');

}

const cell = cells[row][col];

if (!startNode) {

startNode = { row, col };

cell.classList.add('start');

} else if (!endNode) {

endNode = { row, col };

cell.classList.add('end');

} else {

cell.classList.toggle('obstacle');

}

}

// Reset the grid

function resetGrid() {

startNode = null;

endNode = null;

cells.flat().forEach(cell => {

cell.classList.remove('start', 'end', 'obstacle', 'path', 'visited');

});

}

// Pathfinding algorithms

function runAlgorithm() {

const algorithm = algorithms.value;

switch (algorithm) {

case 'bfs':

bfs();

break;

case 'dfs':

dfs();

break;

case 'dijkstra':

dijkstra();

break;

case 'aStar':

aStar();

break;

case 'greedy':

greedy();

break;

case 'biDir':

bidirectional();

break;

}

}

function bfs() {

// Implement BFS algorithm

}

function dfs() {

// Implement DFS algorithm

}

function dijkstra() {

// Implement Dijkstra's algorithm

}

function aStar() {

// Implement A\* algorithm

}

function greedy() {

// Implement Greedy Best-First Search algorithm

}

function bidirectional() {

// Implement Bidirectional Search algorithm

}

// Initialize

createGrid();

startButton.addEventListener('click', runAlgorithm);

resetButton.addEventListener('click', resetGrid);