<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Bellman-Ford Visualization</title>

<style>

canvas {

background-color: #1e1e1e;

display: block;

margin: auto;

margin-top: 20px;

}

body {

color: white;

font-family: monospace;

text-align: center;

background-color: #121212;

}

</style>

</head>

<body>

<h1>Bellman-Ford Algorithm Visualization</h1>

<p id="status">Status: Initializing...</p>

<canvas id="canvas" width="600" height="700"></canvas>

<script>

const ROWS = 6, COLS = 6;

const TILE\_SIZE = 100;

const WIDTH = COLS \* TILE\_SIZE, HEIGHT = ROWS \* TILE\_SIZE + 100;

const canvas = document.getElementById("canvas");

const ctx = canvas.getContext("2d");

const statusText = document.getElementById("status");

const maze = [

[0, 0, 0, 1, 0, 0],

[1, 1, 0, 1, 0, 1],

[0, 0, 0, 0, 0, 0],

[0, 1, 1, 1, 1, 0],

[0, 0, 0, 0, 0, 0],

[1, 1, 0, 1, 0, 0],

];

const start = [0, 0];

const goal = [5, 5];

function getNeighbors(r, c) {

const deltas = [[-1,0], [1,0], [0,-1], [0,1]];

return deltas

.map(([dr, dc]) => [r + dr, c + dc])

.filter(([nr, nc]) =>

nr >= 0 && nr < ROWS && nc >= 0 && nc < COLS && maze[nr][nc] === 0

);

}

function buildGraph() {

const edges = [];

for (let r = 0; r < ROWS; r++) {

for (let c = 0; c < COLS; c++) {

if (maze[r][c] === 0) {

for (const [nr, nc] of getNeighbors(r, c)) {

edges.push({ from: [r, c], to: [nr, nc], weight: 1 });

}

}

}

}

return edges;

}

function drawGrid(dist, path = [], current = null, status = "") {

ctx.clearRect(0, 0, WIDTH, HEIGHT);

// Draw grid

for (let r = 0; r < ROWS; r++) {

for (let c = 0; c < COLS; c++) {

ctx.strokeStyle = "#333";

ctx.strokeRect(c \* TILE\_SIZE, r \* TILE\_SIZE, TILE\_SIZE, TILE\_SIZE);

if (maze[r][c] === 1) {

ctx.fillStyle = "#464646";

ctx.fillRect(c \* TILE\_SIZE, r \* TILE\_SIZE, TILE\_SIZE, TILE\_SIZE);

}

}

}

// Draw visited nodes

for (let key in dist) {

const [r, c] = key.split(",").map(Number);

ctx.fillStyle = "rgb(255,100,100)";

ctx.beginPath();

ctx.arc(c \* TILE\_SIZE + 50, r \* TILE\_SIZE + 50, 6, 0, Math.PI \* 2);

ctx.fill();

}

// Draw path

for (let [r, c] of path) {

ctx.fillStyle = "rgb(50,220,50)";

ctx.beginPath();

ctx.arc(c \* TILE\_SIZE + 50, r \* TILE\_SIZE + 50, 6, 0, Math.PI \* 2);

ctx.fill();

}

// Draw current

if (current) {

const [r, c] = current;

ctx.fillStyle = "yellow";

ctx.beginPath();

ctx.arc(c \* TILE\_SIZE + 50, r \* TILE\_SIZE + 50, 8, 0, Math.PI \* 2);

ctx.fill();

}

// Draw start and goal

const [sr, sc] = start;

const [gr, gc] = goal;

ctx.fillStyle = "rgb(0,170,255)";

ctx.beginPath();

ctx.arc(sc \* TILE\_SIZE + 50, sr \* TILE\_SIZE + 50, 10, 0, Math.PI \* 2);

ctx.fill();

ctx.fillStyle = "rgb(255,80,80)";

ctx.beginPath();

ctx.arc(gc \* TILE\_SIZE + 50, gr \* TILE\_SIZE + 50, 10, 0, Math.PI \* 2);

ctx.fill();

statusText.textContent = "Status: " + status;

}

async function bellmanFordVisual() {

const graph = buildGraph();

const dist = {};

const prev = {};

for (let r = 0; r < ROWS; r++) {

for (let c = 0; c < COLS; c++) {

if (maze[r][c] === 0) dist[[r, c]] = Infinity;

}

}

dist[start] = 0;

drawGrid(dist, [], null, "Starting...");

await sleep(1000);

for (let i = 0; i < ROWS \* COLS - 1; i++) {

let updated = false;

for (const { from, to, weight } of graph) {

if (dist[from] + weight < dist[to]) {

dist[to] = dist[from] + weight;

prev[to] = from;

updated = true;

drawGrid(dist, [], to, `Relaxing (Step ${i + 1})`);

await sleep(100);

}

}

if (!updated) break;

}

// Check for negative weight cycle

for (const { from, to, weight } of graph) {

if (dist[from] + weight < dist[to]) {

drawGrid(dist, [], null, "Negative cycle detected!");

return;

}

}

// Build path

const path = [];

let curr = goal;

while (prev[curr]) {

path.push(curr);

curr = prev[curr];

}

if (curr && curr.toString() === start.toString()) {

path.push(start);

path.reverse();

for (const node of path) {

drawGrid(dist, path, node, "Showing path...");

await sleep(100);

}

drawGrid(dist, path, null, "Done!");

} else {

drawGrid(dist, [], null, "No path found.");

}

}

function sleep(ms) {

return new Promise(resolve => setTimeout(resolve, ms));

}

bellmanFordVisual();

</script>

</body>

</html>