<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Travelling Salesman Problem Visualizer</title>

<style>

canvas {

border: 1px solid black;

}

body {

font-family: Arial, sans-serif;

text-align: center;

}

button {

margin-top: 10px;

padding: 10px 20px;

}

</style>

</head>

<body>

<h2>Travelling Salesman Problem Simulation (Step-by-Step Visualization)</h2>

<canvas id="tspCanvas" width="800" height="600"></canvas>

<br>

<button onclick="solveTSP()">Solve TSP</button>

<button onclick="clearCanvas()">Clear</button>

<script>

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

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

const cities = [];

canvas.addEventListener("click", function(event) {

const rect = canvas.getBoundingClientRect();

const x = event.clientX - rect.left;

const y = event.clientY - rect.top;

cities.push({ x, y });

drawCities();

});

function drawCities(path = []) {

ctx.clearRect(0, 0, canvas.width, canvas.height);

ctx.fillStyle = "blue";

cities.forEach(city => {

ctx.beginPath();

ctx.arc(city.x, city.y, 5, 0, Math.PI \* 2);

ctx.fill();

});

if (path.length > 1) {

ctx.strokeStyle = "red";

ctx.beginPath();

ctx.moveTo(path[0].x, path[0].y);

for (let i = 1; i < path.length; i++) {

ctx.lineTo(path[i].x, path[i].y);

}

ctx.stroke();

}

}

function distance(a, b) {

return Math.hypot(a.x - b.x, a.y - b.y);

}

function sleep(ms) {

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

}

async function solveTSP() {

if (cities.length < 2) {

alert("Add at least 2 cities.");

return;

}

const unvisited = [...cities];

const path = [];

let current = unvisited.shift(); // start with the first city

path.push(current);

while (unvisited.length > 0) {

let nearest = null;

let minDist = Infinity;

let index = -1;

unvisited.forEach((city, i) => {

const d = distance(current, city);

if (d < minDist) {

minDist = d;

nearest = city;

index = i;

}

});

current = nearest;

unvisited.splice(index, 1); // remove from unvisited

path.push(current);

drawCities(path); // draw after each step

await sleep(50); // delay for visualization

}

// Draw final connection back to start

path.push(path[0]);

drawCities(path);

const totalDist = path.reduce((sum, city, i) => {

if (i === path.length - 1) return sum;

return sum + distance(city, path[i + 1]);

}, 0);

alert(`(Approximate) Shortest path length: ${totalDist.toFixed(2)} pixels`);

}

function clearCanvas() {

cities.length = 0;

ctx.clearRect(0, 0, canvas.width, canvas.height);

}

</script>

</body>

</html>