UNIVERSIDAD NACIONAL DEL ALTIPLANO

Facultad Ingeniería Mecánica, Eléctrica, Electrónica y Sistemas

Escuela Profesional de Ingeniería de Sistemas

Trabajo 2 :

Buddy Trees

DOCENTE:

ING. COLLANQUI MARTINEZ FREDY

PRESENTADO POR:

CHOQUE CHURA OLIVER FRAN

SEXTO SEMESTRE 2024-I

PUNO - PERÚ

2024

CÓDIGO EN JAVASCRIPT

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Buddy Tree Simulaci�n</title>
                                                                    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #010313; /* Fondo negro */
            color: #fff; /* Texto blanco para mejor contraste */
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            flex-direction: column;
        }

        .container-fluid {
            max-width: 1200px;
            text-align: center;
            border: 2px solid #555; /* A�ade un borde a todo el contenedor */
            border-radius: 10px; /* Redondea las esquinas del borde */
               padding: 20px; /* A�ade un padding para evitar que el contenido toque el
borde */
            background-color: #222; /* Fondo oscuro para el contenedor */
        }

        canvas {
            display: block;
            margin: auto;
            background-color: #fff;
            border: 2px solid #555;
            margin-left: -15px; /* Desplaza el canvas a la izquierda */
        }

        label {
            margin-right: 10px;
        }

        input {
            padding: 5px;
            margin-right: 10px;
        }
        button {
            padding: 8px 20px;
            background-color: #555;
            color: #fff;
            border: none;
            cursor: pointer;
        }

        button:hover {
            background-color: #333;
        }

        .form-group {
            margin-bottom: 15px;
        }

        #nodeValues {
            margin-top: 20px;
        }

        .title {
            text-align: center;
            width: 100%;
            margin-bottom: 20px;
        }

        .title h1 {
            font-size: 36px;
            font-weight: bold;
            text-shadow: 2px 2px 4px rgba(187, 90, 90, 0.5);
        }

        .function-box {
            background-color: #f8f9fa;
            padding: 0px;
            border-radius: 10px;
        }
    </style>
</head>

<body>
    <div class="title">
        <h1>Buddy Tree Simulaci�n</h1>
    </div>
    <div class="container-fluid">
        <div class="row">
            <div class="col-md-8 mx-auto">
```

```
                <canvas id="treeCanvas" width="800" height="600"></canvas>
            </div>
            <div class="col-md-4">
                <div class="row mt-3">
                    <div class="col-md-12">
                        <div class="form-group text-center">
                            <h3>Insertar Nodo</h3>
                            <label for="nodeValue">Valor del nodo:</label>
                            <input type="number" class="form-control" id="nodeValue">
                            <button class="btn btn-primary mt-2" onclick="addNode()">Agregar Nodo</button>
                        </div>
                    </div>
                </div>
                <div class="row mt-3">
                    <div class="col-md-12">
                        <div class="form-group text-center">
                            <h3>Eliminar Nodo</h3>
                            <label for="nodeToRemove">Valor del nodo a eliminar:</label>
                            <input type="number" class="form-control" id="nodeToRemove">
                            <button class="btn btn-danger mt-2" onclick="removeNode()">Eliminar Nodo</button>
                        </div>
                    </div>
                </div>
                <div class="row mt-3">
                    <div class="col-md-12">
                        <div id="nodeValues" class="text-center">
                            <h3>Datos de los Nodos</h3>
                            <ul id="nodeList"></ul>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <script>
        class Node {
            constructor(value) {
                this.value = value;
                this.left = null;
                this.right = null;
            }
        }

        class BuddyTree {
            constructor(canvas) {
                this.root = null;
                this.canvas = canvas;
                this.ctx = canvas.getContext('2d');
                this.nodeRadius = 20;
                this.levelGap = 80;
                this.verticalGap = 60;
            }

            insert(value) {
                if (!this.root) {
                    this.root = new Node(value);
                } else {
                    this.insertNode(this.root, value);
                }
                this.drawTree();
                this.updateNodeList();
            }

            insertNode(node, value) {
                if (value < node.value) {
                    if (!node.left) {
                        node.left = new Node(value);
                    } else {
                        this.insertNode(node.left, value);
                    }
                } else {
                    if (!node.right) {
                        node.right = new Node(value);
                    } else {
                        this.insertNode(node.right, value);
                    }
                }
            }

            remove(value) {
                this.root = this.removeNode(this.root, value);
                this.drawTree();
                this.updateNodeList();
            }

            removeNode(node, value) {
                if (!node) {
                    return null;
                }

                if (value < node.value) {
                    node.left = this.removeNode(node.left, value);
                    return node;
                } else if (value > node.value) {
                    node.right = this.removeNode(node.right, value);
                    return node;
                } else {
                    if (!node.left && !node.right) {
                        return null;
                    }

                    if (!node.left) {
                        return node.right;
                    }

                    if (!node.right) {
                        return node.left;
                    }

                    const minRight = this.findMinNode(node.right);
                    node.value = minRight.value;
                    node.right = this.removeNode(node.right, minRight.value);
                    return node;
                }
            }

            findMinNode(node) {
                if (!node.left) {
                    return node;
                }
                return this.findMinNode(node.left);
            }

            drawTree() {
                this.ctx.clearRect(0, 0, this.canvas.width, this.canvas.height);
                if (this.root) {
                    this.drawNode(this.root, this.canvas.width / 2, 50, 0);
                }
            }

            drawNode(node, x, y, level) {
                this.ctx.beginPath();
                this.ctx.arc(x, y, this.nodeRadius, 0, Math.PI * 2);
                this.ctx.fillStyle = '#fff';
                this.ctx.strokeStyle = '#555';
                this.ctx.lineWidth = 2;
                this.ctx.fill();
                this.ctx.stroke();
                this.ctx.closePath();

                this.ctx.font = '14px Arial';
                this.ctx.fillStyle = '#555';
                this.ctx.textAlign = 'center';
                this.ctx.textBaseline = 'middle';
                this.ctx.fillText(node.value, x, y);

                if (node.left) {
                    const childX = x - this.levelGap / Math.pow(2, level + 1);
                    const childY = y + this.verticalGap;
                    this.drawNode(node.left, childX, childY, level + 1);
                    this.drawLine(x, y, childX, childY);
                }

                if (node.right) {
                    const childX = x + this.levelGap / Math.pow(2, level + 1);
                    const childY = y + this.verticalGap;
                    this.drawNode(node.right, childX, childY, level + 1);
                    this.drawLine(x, y, childX, childY);
                }
            }

            drawLine(x1, y1, x2, y2) {
                this.ctx.beginPath();
                this.ctx.moveTo(x1, y1 + this.nodeRadius);
                this.ctx.lineTo(x2, y2 - this.nodeRadius);
                this.ctx.strokeStyle = '#555';
                this.ctx.lineWidth = 2;
                this.ctx.stroke();
                this.ctx.closePath();
            }

            updateNodeList() {
                const nodeList = document.getElementById('nodeList');
                nodeList.innerHTML = '';
                this.traverseInOrder(this.root, nodeList);
            }

            traverseInOrder(node, list) {
                if (node) {
                    this.traverseInOrder(node.left, list);
                    const listItem = document.createElement('li');
                    listItem.textContent = node.value;
                    list.appendChild(listItem);
                    this.traverseInOrder(node.right, list);
                }
            }
        }

        const canvas = document.getElementById('treeCanvas');
        const tree = new BuddyTree(canvas);
```

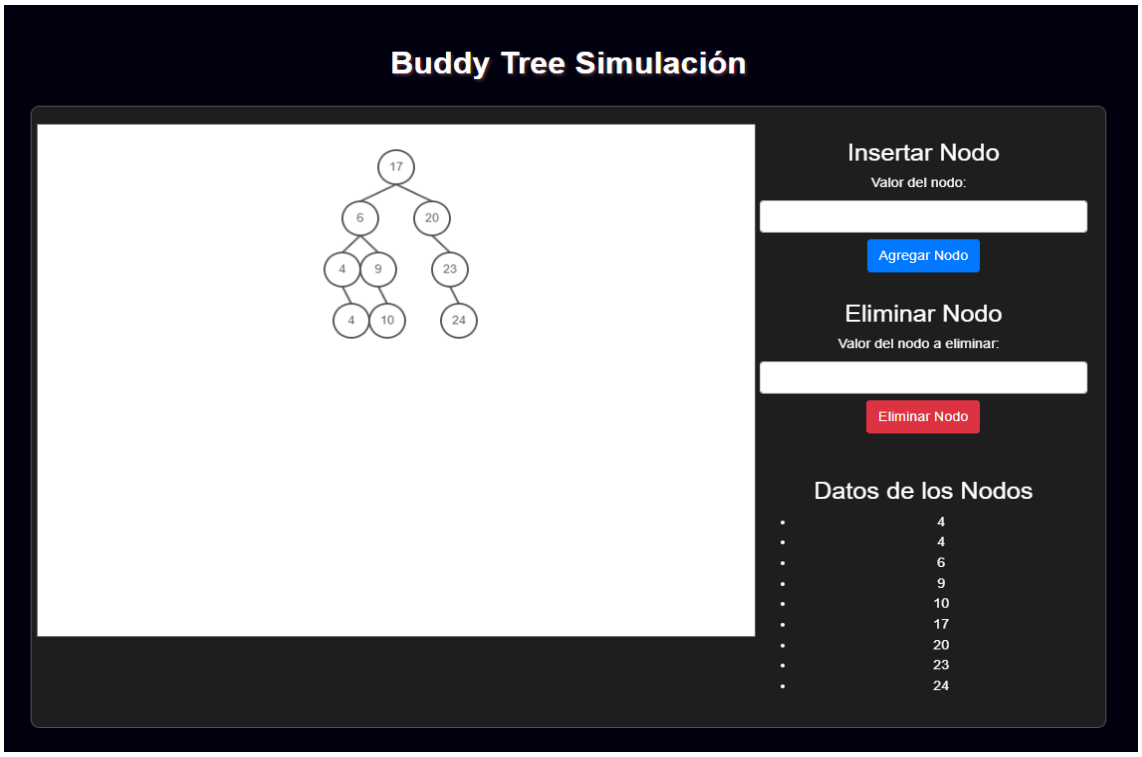
```
        function addNode() {
            const valueInput = document.getElementById('nodeValue');
            const value = parseInt(valueInput.value);
            if (!isNaN(value)) {
                tree.insert(value);
                valueInput.value = '';
            }
        }

        function removeNode() {
            const valueInput = document.getElementById('nodeToRemove');
            const value = parseInt(valueInput.value);
            if (!isNaN(value)) {
                tree.remove(value);
                valueInput.value = '';
            }
        }

        tree.drawTree();
    </script>
</body>
</html>
```

RESULTADO

https://github.com/OliverChoque/Buddy-Trees.git



Buddy Tree Simulación

Insertar Nodo

Valor del nodo:

Agregar Nodo

Eliminar Nodo

Valor del nodo a eliminar:

Eliminar Nodo

Datos de los Nodos

- 4
- 4
- 6
- 9
- 10
- 17
- 20
- 23
- 24