

Nombre: Efrain Saul Mayta Yujra

Implementación de Buddy Trees

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

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Buddy Tree Simulation</title>
  <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.c
ss" rel="stylesheet">
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f5f5f5;
      padding: 20px;
    }

    canvas {
      display: block;
      margin: auto;
      background-color: #fff;
      border: 2px solid #555;
    }

    label {
      margin-right: 10px;
    }

    input {
      padding: 5px;
      margin-right: 10px;
    }

    button {
      padding: 8px 20px;
      background-color: #555;
      color: #fff;
      border: none;
      cursor: pointer;
    }
  </style>

```

```

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

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

        .container-fluid {
            max-width: 1200px;
        }

        #nodeValue {
            margin-top: 20px;
        }
    </style>
</head>

<body>
    <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>
                <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="nodeValue" 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:

The screenshot displays a web application titled "Buddy Tree Simulation" running in a browser. The application interface is split into two main sections: a code editor on the left and a simulation area on the right.

Code Editor (Left): Shows the HTML and JavaScript code for the application. The HTML includes a form for inserting and removing nodes. The JavaScript defines a `Node` class and a `BuddyTree` class with methods for inserting, removing, and drawing the tree.

Simulation Area (Right): Contains a visual representation of the binary tree and a control panel.

Tree Visualization: A binary tree with the following structure:

```

graph TD
    50((50)) --> 25((25))
    50 --> 66((66))
    25 --> 14((14))
    25 --> 28((28))
    14 --> 2((2))
    66 --> 70((70))
    70 --> 80((80))
    80 --> 75((75))

```

Control Panel: Includes two sections:

- Insertar Nodo:** A text input field for "Valor del nodo:" and a blue "Agregar Nodo" button.
- Eliminar Nodo:** A text input field for "Valor del nodo a eliminar:" and a red "Eliminar Nodo" button.
- Datos de los Nodos:** A list of the values stored in the tree nodes: 2, 14, 25, 28, 50, 66, 70, 75, 80.