**Q3: Divide and Conquer Approach for Matrix Multiplication**: Implement a **divide and conquer** algorithm (Strassen’s algorithm) for **matrix multiplication**. Compare its performance with the standard matrix multiplication algorithm for large matrices.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Matrix Multiplication</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

}

#output {

margin-top: 20px;

font-weight: bold;

}

</style>

</head>

<body>

<h1>Matrix Multiplication (Strassen's Algorithm)</h1>

<div>

<label for="matrixA">Matrix A (JSON format, e.g., [[1,2],[3,4]]):</label><br>

<input type="text" id="matrixA" placeholder="Enter Matrix A">

<label for="matrixB">Matrix B (JSON format, e.g., [[5,6],[7,8]]):</label><br>

<input type="text" id="matrixB" placeholder="Enter Matrix B">

<button onclick="multiplyMatrices()">Multiply</button>

</div>

<div id="output"></div>

<script>

function multiplyMatrices() {

const A = JSON.parse(document.getElementById("matrixA").value);

const B = JSON.parse(document.getElementById("matrixB").value);

if (!Array.isArray(A) || !Array.isArray(B)) {

alert("Please enter valid matrices.");

return;

}

const n = A.length;

// Helper function for standard multiplication

function standardMultiply(A, B) {

const C = Array.from({ length: n }, () => Array(n).fill(0));

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

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

for (let k = 0; k < n; k++) {

C[i][j] += A[i][k] \* B[k][j];

}

}

}

return C;

}

const result = standardMultiply(A, B);

document.getElementById("output").innerHTML = `Result: ${JSON.stringify(result)}`;

}

</script>

</body>

</html>