**15.Write a program to perform Strassen’s Matrix Multiplication.**

**Aim**

To implement **Strassen’s Matrix Multiplication** algorithm to multiply two square matrices in a more efficient way than conventional matrix multiplication.

**Algorithm**

**Strassen’s Algorithm Steps:**

1. **Input:** Two square matrices A and B of size n × n, where n is a power of 2.
2. If n == 1, return A[0][0] \* B[0][0].
3. Otherwise:
   1. Divide A and B into four n/2 × n/2 submatrices:

A11, A12, A21, A22

B11, B12, B21, B22

* 1. Compute the following seven products recursively:

M1 = (A11 + A22) × (B11 + B22)

M2 = (A21 + A22) × B11

M3 = A11 × (B12 - B22)

M4 = A22 × (B21 - B11)

M5 = (A11 + A12) × B22

M6 = (A21 - A11) × (B11 + B12)

M7 = (A12 - A22) × (B21 + B22)

* 1. Compute result submatrices:

ini

CopyEdit

C11 = M1 + M4 - M5 + M7

C12 = M3 + M5

C21 = M2 + M4

C22 = M1 - M2 + M3 + M6

1. Combine the submatrices into the final matrix C.
2. **Output:** Product matrix C.

**C Program**

#include <stdio.h>

#include <stdlib.h>

#define MAX 4 // size of matrix (must be power of 2)

void add(int a[MAX][MAX], int b[MAX][MAX], int c[MAX][MAX], int n)

{

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

c[i][j] = a[i][j] + b[i][j];

}

void subtract(int a[MAX][MAX], int b[MAX][MAX], int c[MAX][MAX], int n) {

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

c[i][j] = a[i][j] - b[i][j];

}

void strassen(int A[MAX][MAX], int B[MAX][MAX], int C[MAX][MAX], int n) {

if (n == 1) {

C[0][0] = A[0][0] \* B[0][0];

return;

}

int k = n / 2;

int A11[MAX][MAX], A12[MAX][MAX], A21[MAX][MAX], A22[MAX][MAX];

int B11[MAX][MAX], B12[MAX][MAX], B21[MAX][MAX], B22[MAX][MAX];

int M1[MAX][MAX], M2[MAX][MAX], M3[MAX][MAX], M4[MAX][MAX], M5[MAX][MAX], M6[MAX][MAX], M7[MAX][MAX];

int temp1[MAX][MAX], temp2[MAX][MAX];

// Divide matrices

for (int i = 0; i < k; i++) {

for (int j = 0; j < k; j++) {

A11[i][j] = A[i][j];

A12[i][j] = A[i][j + k];

A21[i][j] = A[i + k][j];

A22[i][j] = A[i + k][j + k];

B11[i][j] = B[i][j];

B12[i][j] = B[i][j + k];

B21[i][j] = B[i + k][j];

B22[i][j] = B[i + k][j + k];

}

}

// M1 = (A11 + A22) × (B11 + B22)

add(A11, A22, temp1, k);

add(B11, B22, temp2, k);

strassen(temp1, temp2, M1, k);

// M2 = (A21 + A22) × B11

add(A21, A22, temp1, k);

strassen(temp1, B11, M2, k);

// M3 = A11 × (B12 - B22)

subtract(B12, B22, temp2, k);

strassen(A11, temp2, M3, k);

// M4 = A22 × (B21 - B11)

subtract(B21, B11, temp2, k);

strassen(A22, temp2, M4, k);

// M5 = (A11 + A12) × B22

add(A11, A12, temp1, k);

strassen(temp1, B22, M5, k);

// M6 = (A21 - A11) × (B11 + B12)

subtract(A21, A11, temp1, k);

add(B11, B12, temp2, k);

strassen(temp1, temp2, M6, k);

// M7 = (A12 - A22) × (B21 + B22)

subtract(A12, A22, temp1, k);

add(B21, B22, temp2, k);

strassen(temp1, temp2, M7, k);

// C11 = M1 + M4 - M5 + M7

add(M1, M4, temp1, k);

subtract(temp1, M5, temp2, k);

add(temp2, M7, C, k);

// C12 = M3 + M5

add(M3, M5, C[0] + k, k);

// C21 = M2 + M4

add(M2, M4, C[k], k);

// C22 = M1 - M2 + M3 + M6

subtract(M1, M2, temp1, k);

add(temp1, M3, temp2, k);

add(temp2, M6, C[k] + k, k);

}

int main() {

int A[MAX][MAX], B[MAX][MAX], C[MAX][MAX];

int n = 4;

printf("Enter elements of matrix A (%dx%d):\n", n, n);

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

scanf("%d", &A[i][j]);

printf("Enter elements of matrix B (%dx%d):\n", n, n);

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

scanf("%d", &B[i][j]);

strassen(A, B, C, n);

printf("Resultant Matrix C:\n");

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

for (int j = 0; j < n; j++)

printf("%d ", C[i][j]);

printf("\n");

}

return 0;

}

**Sample Input**

java

CopyEdit

Enter elements of matrix A (4x4):

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

Enter elements of matrix B (4x4):

1 0 0 1

0 1 1 0

1 1 0 0

0 0 1 1

**Sample Output**

mathematica

CopyEdit

Resultant Matrix C:

4 5 5 4

12 13 13 12

20 21 21 20

28 29 29 28

