**Practical -3**

**Aim:** Write program to perform Strassen’s Matrix multiplication using Divide and Conquer

techniques.

**Theory:**

**Strassen's Matrix Multiplication:**

Strassen's algorithm is a divide-and-conquer method for matrix multiplication that was designed to be more efficient than the standard matrix multiplication algorithm, especially for large matrices. The key idea is to decompose the matrix multiplication into a set of subproblems, reducing the number of multiplications required compared to the traditional algorithm.

The algorithm works by dividing the input matrices into smaller submatrices and recursively computing products of these submatrices using a set of seven multiplicative operations. The resulting products are then combined to obtain the final result.

**Acceptance Criteria:**

The acceptance criteria for this code would involve ensuring that the Strassen's matrix multiplication algorithm is correctly implemented and that the program successfully takes input from the user, performs the multiplication, and outputs the correct result. Specifically:

* The program should prompt the user to input the size of matrices.
* The program should then prompt the user to enter the elements of the two matrices.
* The Strassen's matrix multiplication algorithm should be implemented correctly.
* The program should output the correct result of the matrix multiplication.

**Code:**

def strassen\_matrix\_multiply(A, B):

    n = len(A)

    if n == 1:

        return [[A[0][0] \* B[0][0]]]

    mid = n // 2

    A11 = [row[:mid] for row in A[:mid]]

    A12 = [row[mid:] for row in A[:mid]]

    A21 = [row[:mid] for row in A[mid:]]

    A22 = [row[mid:] for row in A[mid:]]

    B11 = [row[:mid] for row in B[:mid]]

    B12 = [row[mid:] for row in B[:mid]]

    B21 = [row[:mid] for row in B[mid:]]

    B22 = [row[mid:] for row in B[mid:]]

    S1 = [[B12[i][j] - B22[i][j] for j in range(mid)] for i in range(mid)]

    S2 = [[A11[i][j] + A12[i][j] for j in range(mid)] for i in range(mid)]

    S3 = [[A21[i][j] + A22[i][j] for j in range(mid)] for i in range(mid)]

    S4 = [[B21[i][j] - B11[i][j] for j in range(mid)] for i in range(mid)]

    S5 = [[A11[i][j] + A22[i][j] for j in range(mid)] for i in range(mid)]

    S6 = [[B11[i][j] + B22[i][j] for j in range(mid)] for i in range(mid)]

    S7 = [[A12[i][j] - A22[i][j] for j in range(mid)] for i in range(mid)]

    S8 = [[B21[i][j] + B22[i][j] for j in range(mid)] for i in range(mid)]

    S9 = [[A11[i][j] - A21[i][j] for j in range(mid)] for i in range(mid)]

    S10 = [[B11[i][j] + B12[i][j] for j in range(mid)] for i in range(mid)]

    P1 = strassen\_matrix\_multiply(A11, S1)

    P2 = strassen\_matrix\_multiply(S2, B22)

    P3 = strassen\_matrix\_multiply(S3, B11)

    P4 = strassen\_matrix\_multiply(A22, S4)

    P5 = strassen\_matrix\_multiply(S5, S6)

    P6 = strassen\_matrix\_multiply(S7, S8)

    P7 = strassen\_matrix\_multiply(S9, S10)

    C11 = [[P5[i][j] + P4[i][j] - P2[i][j] + P6[i][j] for j in range(mid)] for i in range(mid)]

    C12 = [[P1[i][j] + P2[i][j] for j in range(mid)] for i in range(mid)]

    C21 = [[P3[i][j] + P4[i][j] for j in range(mid)] for i in range(mid)]

    C22 = [[P5[i][j] + P1[i][j] - P3[i][j] - P7[i][j] for j in range(mid)] for i in range(mid)]

    result = []

    for i in range(mid):

        result.append(C11[i] + C12[i])

    for i in range(mid):

        result.append(C21[i] + C22[i])

    return result

def input\_matrix(n):

    matrix = []

    print(f"Enter {n}x{n} matrix elements:")

    for i in range(n):

        row = list(map(int, input().split()))

        matrix.append(row)

    return matrix

size = int(input("Enter size of matrices: "))

print("Enter Matrix A:")

matrix\_A = input\_matrix(size)

print("Enter Matrix B:")

matrix\_B = input\_matrix(size)

result\_matrix = strassen\_matrix\_multiply(matrix\_A, matrix\_B)

print("Resultant Matrix:")

for row in result\_matrix:

    print(' '.join(map(str, row)))

**Input:**

Enter size of matrices: 2

Enter Matrix A:

Enter 2x2 matrix elements:

1 2 3

4 5 6

Enter Matrix B:

Enter 2x2 matrix elements:

4 5 6

7 8 9

**Output:**

Resultant Matrix:

18 21

51 60

**Conclusion:**

The provided Python code successfully implements Strassen's matrix multiplication algorithm. It demonstrates how to divide matrices into submatrices, recursively compute products using the Strassen's algorithm, and combine the results to obtain the final matrix multiplication.

The acceptance criteria can be validated by running the program with different matrix inputs and verifying that the output matches the expected results of Strassen's matrix multiplication.

In conclusion, the code provides a functional implementation of Strassen's matrix multiplication, showcasing the divide-and-conquer technique for more efficient matrix operations.