7. Problem Statement: Matrix Multiplication

**Problem Analysis:**

Let A and B be two n × n matrices to be multiplied, and let C be their product. To keep things simple, when we use a divide and conquer algorithm to compute the matrix product C = A . B, we assume that n is an exact power of 2 in each of the n × n matrices. We make this assumption because in each divide step, we will divide n × n matrices into four n/2× n/2 matrices, and by assuming that n is an exact power of 2, we are guaranteed that as long as n ≥ 2, the dimension n/2 is an integer.

Suppose that we partition each of A, B and C into four n/2 × n/2 matrices





**Algorithm:**

multiply(int A[][N], int B[][N], int C[][N])

{

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

{

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

{

C[i][j] = 0;

for (int k = 0; k < N; k++)

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

}

}

}

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

int a[10][10], b[10][10], mul[10][10], r1, c1, r2, c2, i, j, k;

cout << "Enter rows and columns for first matrix: ";

cin >> r1 >> c1;

cout << "Enter rows and columns for second matrix: ";

cin >> r2 >> c2;

if (c1!=r2)

{

cout << "Error! Column of first matrix is not equal to row of second matrix.\n";

cout << "Enter rows and columns for first matrix: ";

cin >> r1 >> c1;

cout << "Enter rows and columns for second matrix: ";

cin >> r2 >> c2;

}

cout <<"\nEnter elements of matrix 1:" << endl;

for(i = 0; i < r1; ++i)

for(j = 0; j < c1; ++j)

{

cout << "Enter element a" << i + 1 << j + 1 << " : ";

cin >> a[i][j];

}

cout << endl <<"Enter elements of matrix 2:" << endl;

for(i = 0; i < r2; ++i)

for(j = 0; j < c2; ++j)

{

cout << "Enter element b" << i + 1 << j + 1 << " : ";

cin >> b[i][j];

}

for(i = 0; i < r1; ++i)

for(j = 0; j < c2; ++j)

mul[i][j]=0;

for(i = 0; i < r1; ++i)

for(j = 0; j < c2; ++j)

for(k = 0; k < c1; ++k)

mul[i][j] += a[i][k] \* b[k][j];

cout << endl << "Output Matrix: " << endl;

for(i = 0; i < r1; ++i)

{

for(j = 0; j < c2; ++j)

cout << " " << mul[i][j];

cout << endl;

}

return 0;

}

**Sample Input:**

Enter rows and columns for first matrix: 1 2

Enter rows and columns for second matrix: 1 2

Error! Column of first matrix is not equal to row of second matrix.

Enter rows and columns for first matrix: 3 2

Enter rows and columns for second matrix: 2 3

Enter elements of matrix 1:

Enter element a11 : 1

Enter element a12 : 2

Enter element a21 : 3

Enter element a22 : 4

Enter element a31 : 5

Enter element a32 : 6

Enter elements of matrix 2:

Enter element b11 : 9

Enter element b12 : 8

Enter element b13 : 7

Enter element b21 : 6

Enter element b22 : 5

Enter element b23 : 4

**Sample Output:**

Output Matrix:

21 18 15

51 44 37

81 70 59