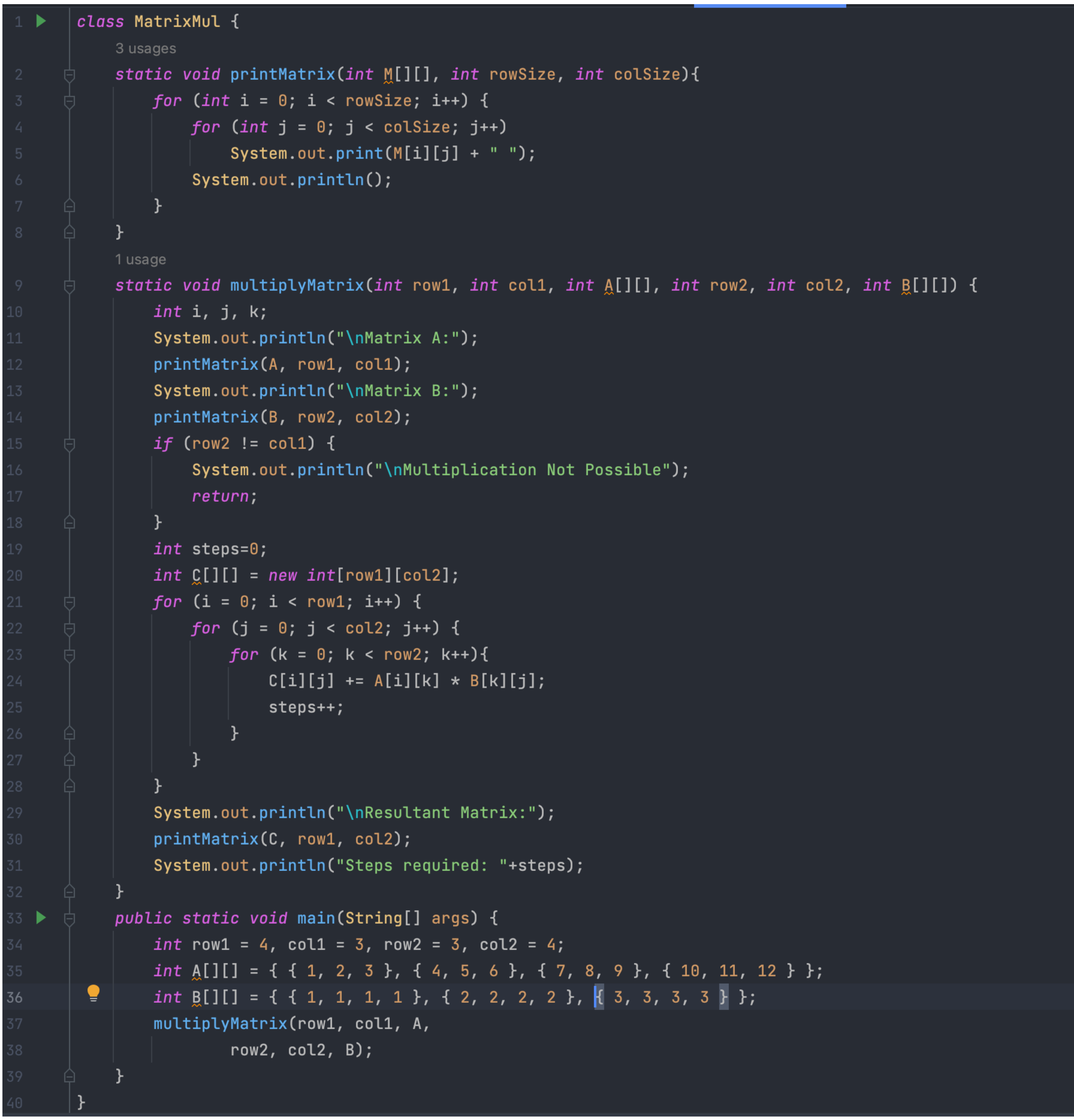
Q: Given two NxN matrices A and B, Find the time efficiency of the definition-based algorithm for computing their product C= AB. By definition, C is an NxN matrix whose elements are computed as the scalar (dot) products of the rows of matrix A and the columns of the matrix B.

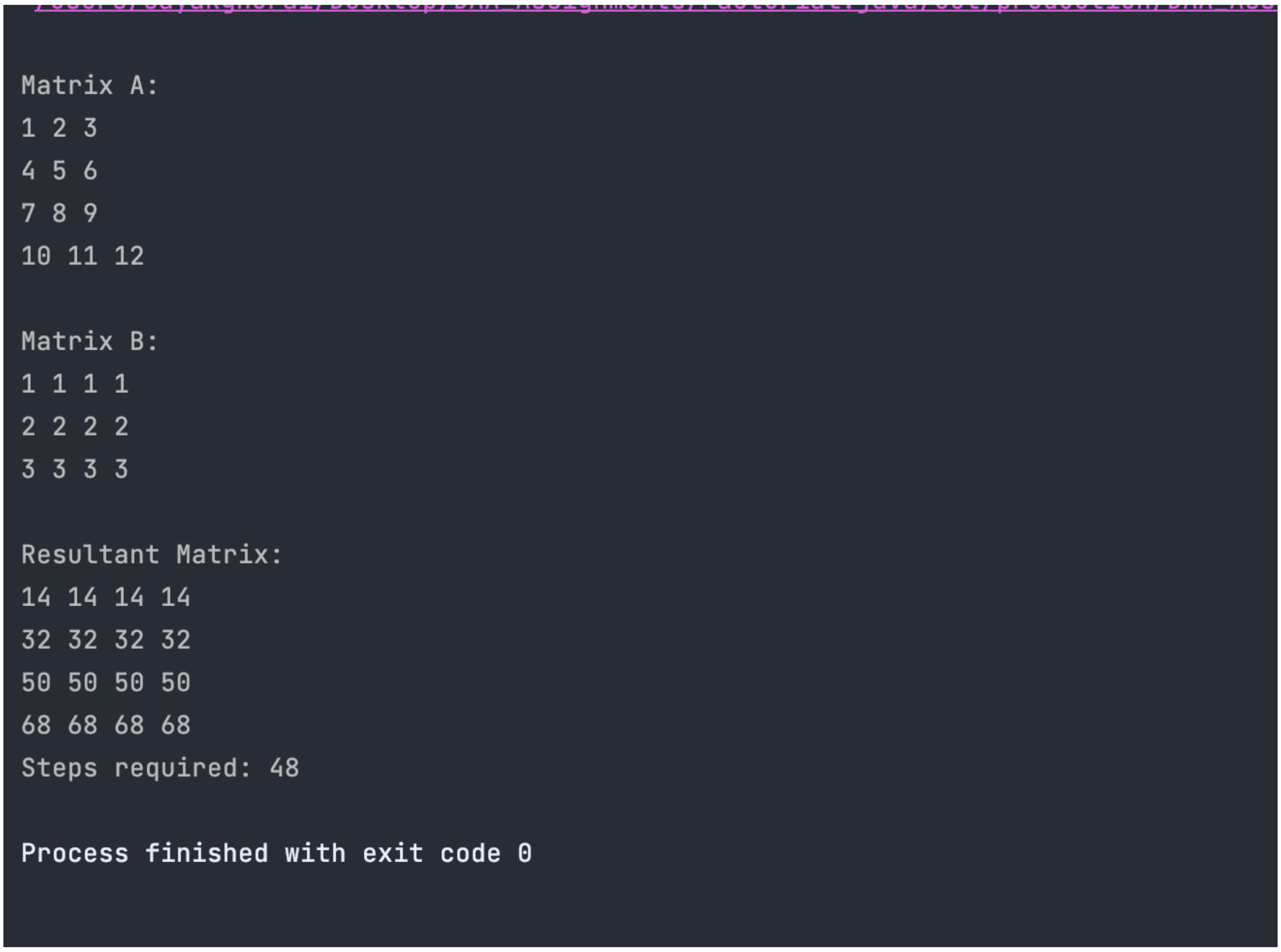
Analyze the time complexity.

Code:



```java
class MatrixMul {
    3 usages
    static void printMatrix(int M[][], int rowSize, int colSize){
        for (int i = 0; i < rowSize; i++) {
            for (int j = 0; j < colSize; j++)
                System.out.print(M[i][j] + " ");
            System.out.println();
        }
    }
    1 usage
    static void multiplyMatrix(int row1, int col1, int A[][], int row2, int col2, int B[][]) {
        int i, j, k;
        System.out.println("\nMatrix A:");
        printMatrix(A, row1, col1);
        System.out.println("\nMatrix B:");
        printMatrix(B, row2, col2);
        if (row2 != col1) {
            System.out.println("\nMultiplication Not Possible");
            return;
        }
        int steps=0;
        int C[][] = new int[row1][col2];
        for (i = 0; i < row1; i++) {
            for (j = 0; j < col2; j++) {
                for (k = 0; k < row2; k++){
                    C[i][j] += A[i][k] * B[k][j];
                    steps++;
                }
            }
        }
        System.out.println("\nResultant Matrix:");
        printMatrix(C, row1, col2);
        System.out.println("Steps required: "+steps);
    }
    public static void main(String[] args) {
        int row1 = 4, col1 = 3, row2 = 3, col2 = 4;
        int A[][] = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 }, { 10, 11, 12 } };
        int B[][] = { { 1, 1, 1, 1 }, { 2, 2, 2, 2 }, { 3, 3, 3, 3 } };
        multiplyMatrix(row1, col1, A,
                row2, col2, B);
    }
}
```

Output:



```
[illegible]

Matrix A:
1 2 3
4 5 6
7 8 9
10 11 12

Matrix B:
1 1 1 1
2 2 2 2
3 3 3 3

Resultant Matrix:
14 14 14 14
32 32 32 32
50 50 50 50
68 68 68 68
Steps required: 48

Process finished with exit code 0
```

TimeComplexity:
As we can see in the code, there is three nested for loops and each for loop runs in order of N.
So the total complexity will be NxNxN=N^3