

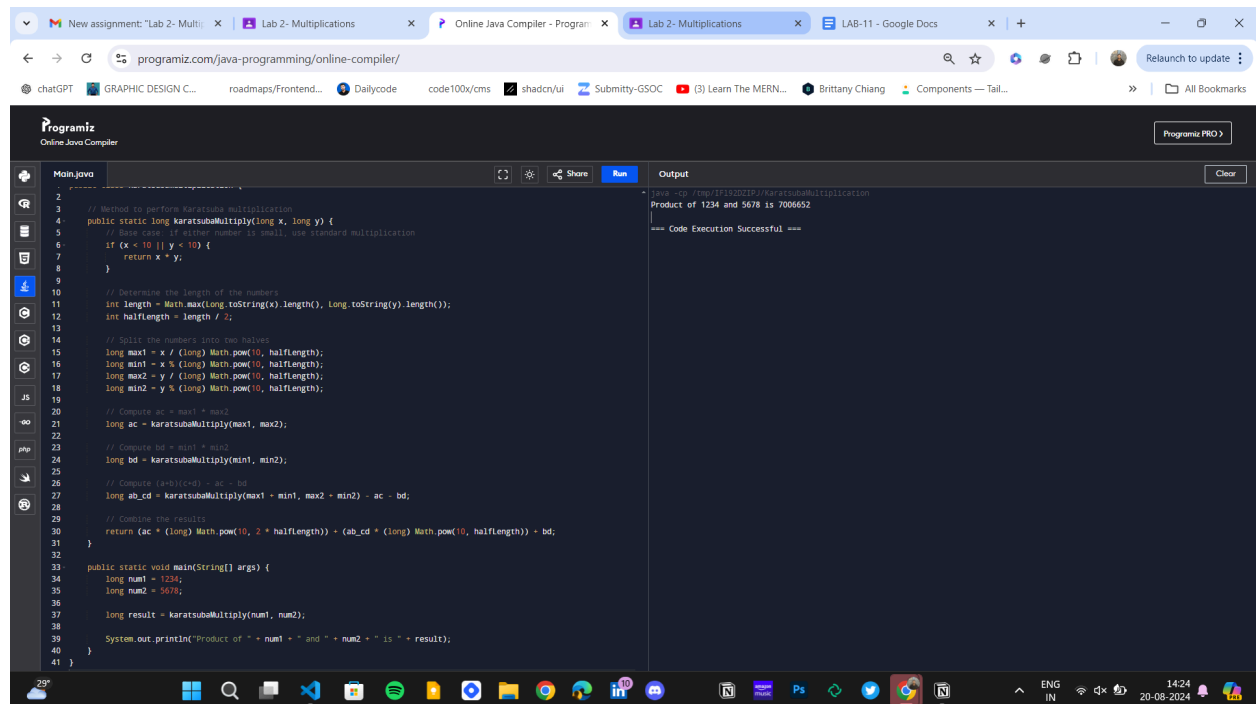
# **Advanced Algorithms**

## **Assignment**

**NAME : G.Sai Bharath Chandra Reddy**

**Roll.No : 22cs2013**

# (A).Karatsuba Multiplication Algorithm.



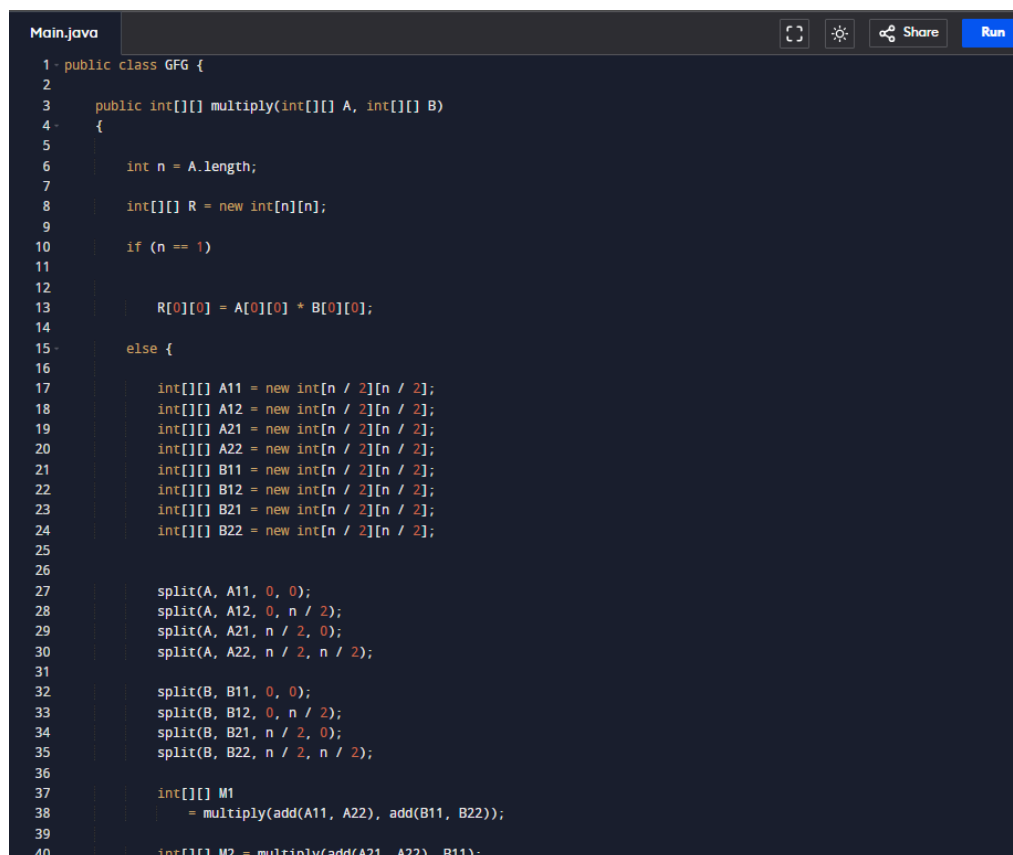
The screenshot shows a web browser with multiple tabs, including 'Lab 2 - Multiplications' and 'Online Java Compiler - Program'. The main content is the 'Programiz Online Java Compiler' interface. The code editor on the left contains a Java implementation of the Karatsuba Multiplication Algorithm. The output window on the right shows the execution result: 'Product of 1234 and 5678 is 7006652'.

```
1 // Method to perform Karatsuba multiplication
2
3 public static long karatsubaMultiply(long x, long y) {
4     // Base case: if either number is small, use standard multiplication
5     if (x < 10 || y < 10) {
6         return x * y;
7     }
8
9     // Determine the length of the numbers
10    int length = Math.max(Long.toString(x).length(), Long.toString(y).length());
11    int halflength = length / 2;
12
13    // Split the numbers into two halves
14    long max1 = x / (long) Math.pow(10, halflength);
15    long min1 = x % (long) Math.pow(10, halflength);
16    long max2 = y / (long) Math.pow(10, halflength);
17    long min2 = y % (long) Math.pow(10, halflength);
18
19    // Compute ac = max1 * max2
20    long ac = karatsubaMultiply(max1, max2);
21
22    // Compute bd = min1 * min2
23    long bd = karatsubaMultiply(min1, min2);
24
25    // Compute (a+b)(c+d) = ac + bd
26    long ab_cd = karatsubaMultiply(max1 + min1, max2 + min2) - ac - bd;
27
28    // Combine the results
29    return (ac * (long) Math.pow(10, 2 * halflength)) + (ab_cd * (long) Math.pow(10, halflength)) + bd;
30 }
31
32 public static void main(String[] args) {
33     long num1 = 1234;
34     long num2 = 5678;
35
36     long result = karatsubaMultiply(num1, num2);
37
38     System.out.println("Product of " + num1 + " and " + num2 + " is " + result);
39 }
40
41 }
```

Output:

```
java -cp /tmp/SP192DZP/ KaratsubaMultiplication
Product of 1234 and 5678 is 7006652
=== Code Execution Successful ===
```

# (B). Strassen Matrix Algorithm



The screenshot shows a web browser with multiple tabs, including 'Lab 2 - Multiplications' and 'Online Java Compiler - Program'. The main content is the 'Programiz Online Java Compiler' interface. The code editor on the left contains a Java implementation of the Strassen Matrix Algorithm. The output window on the right is empty.

```
1 public class GFG {
2
3     public int[][] multiply(int[][] A, int[][] B)
4     {
5
6         int n = A.length;
7
8         int[][] R = new int[n][n];
9
10        if (n == 1)
11        {
12            R[0][0] = A[0][0] * B[0][0];
13        }
14
15        else {
16
17            int[][] A11 = new int[n / 2][n / 2];
18            int[][] A12 = new int[n / 2][n / 2];
19            int[][] A21 = new int[n / 2][n / 2];
20            int[][] A22 = new int[n / 2][n / 2];
21            int[][] B11 = new int[n / 2][n / 2];
22            int[][] B12 = new int[n / 2][n / 2];
23            int[][] B21 = new int[n / 2][n / 2];
24            int[][] B22 = new int[n / 2][n / 2];
25
26
27            split(A, A11, 0, 0);
28            split(A, A12, 0, n / 2);
29            split(A, A21, n / 2, 0);
30            split(A, A22, n / 2, n / 2);
31
32            split(B, B11, 0, 0);
33            split(B, B12, 0, n / 2);
34            split(B, B21, n / 2, 0);
35            split(B, B22, n / 2, n / 2);
36
37            int[][] M1
38            = multiply(add(A11, A22), add(B11, B22));
39
40            int[][] M2 = multiply(add(A21, A22), B11);
```

```

        int[][] M3 = multiply(A11, sub(B12, B22));

        int[][] M4 = multiply(A22, sub(B21, B11));

        int[][] M5 = multiply(add(A11, A12), B22);

        int[][] M6
            = multiply(sub(A21, A11), add(B11, B12));

        int[][] M7
            = multiply(sub(A12, A22), add(B21, B22));

        int[][] C11 = add(sub(add(M1, M4), M5), M7);

        int[][] C12 = add(M3, M5);

        int[][] C21 = add(M2, M4);

        int[][] C22 = add(sub(add(M1, M3), M2), M6);

        join(C11, R, 0, 0);
        join(C12, R, 0, n / 2);
        join(C21, R, n / 2, 0);
        join(C22, R, n / 2, n / 2);
    }

    return R;
}

public int[][] sub(int[][] A, int[][] B)
{
    //
    int n = A.length;

    int[][] C = new int[n][n];

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

```

```

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

                C[i][j] = A[i][j] - B[i][j];

    return C;
}

public int[][] add(int[][] A, int[][] B)
{
    //
    int n = A.length;

    int[][] C = new int[n][n];

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

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

            C[i][j] = A[i][j] + B[i][j];

    return C;
}

public void split(int[][] P, int[][] C, int iB, int jB)
{
    for (int i1 = 0, i2 = iB; i1 < C.length; i1++, i2++)

        for (int j1 = 0, j2 = jB; j1 < C.length;
            j1++, j2++)

            C[i1][j1] = P[i2][j2];
}

public void join(int[][] C, int[][] P, int iB, int jB)
{
    //
}

```

(C)

$$x = 1234$$

$$y = 5678$$

$$a = 12, b = 34$$

$$c = 56, d = 78$$

$$a * c = 12 * 56 = 272$$

$$b * d = 34 * 78 = 2652$$

q-Mul

q-Mul

$$a + b = 46$$

$$c + d = 134$$

$$(a + b) * (c + d) = 6164$$

$$(a * c) * 10000 + (b * d) + (a + b) * (c + d)$$

$$T(n) = 3T(n/2) + O(n)$$

$$O(n^{1.59}) \text{ (Master Theorem)}$$