Lesson:

Java

2D Array Problems 3

PW SKILLS

Pre-Requisites

- 2D Arrays

Today's Checklist

- **Problems based on Prefix Sums in 2D Arrays**

Problem 1: Given a matrix 'a' of dimension n x m and 2 coordinates (l1, r1) and (l2, r2). Return the sum of the rectangle from (l1,r1) to (l2, r2).

Input:

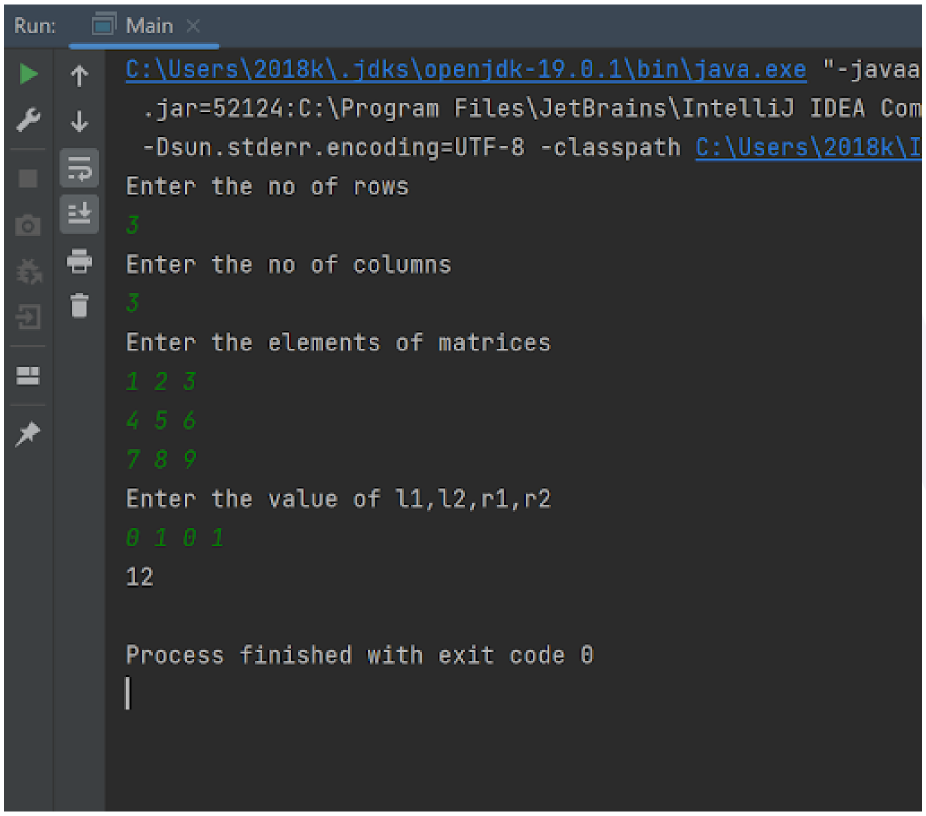
```
Enter the no of rows
3
Enter the no of columns
3
Enter the elements of matrices
1 2 3
4 5 6
7 8 9
Enter the value of l1,l2,r1,r2
0 1 0 1
```

Output

```
12
```

Method 1: Brute Force

```java
import java.io.*;
import java.util.*;
public class Main
{
    public static int solve(int[][] a, int l1, int l2, int r1, int r2){
        int ans = 0;
        for(int i = l1; i <=  l2; i++) {
            for(int j = r1; j <= r2; j++) {
                ans += a[i][j];
            }
        }
        return ans;
    }

public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       System.out.println("Enter the no of rows");
       int row = sc.nextInt();
        System.out.println("Enter the no of columns");
       int col = sc.nextInt();
        int[][] matrix = new int[row][col];
        System.out.println("Enter the elements of matrices");
        for(int i=0;i<row;i++){
            for(int j=0;j<col;j++){
                matrix[i][j]=sc.nextInt();
            }
        }
        System.out.println("Enter the value of l1,l2,r1,r2");
        int l1 = sc.nextInt();
        int l2 = sc.nextInt();
        int r1 = sc.nextInt();
        int r2 = sc.nextInt();

        System.out.println(solve(matrix,l1,l2,r1,r2));
    }
}
```

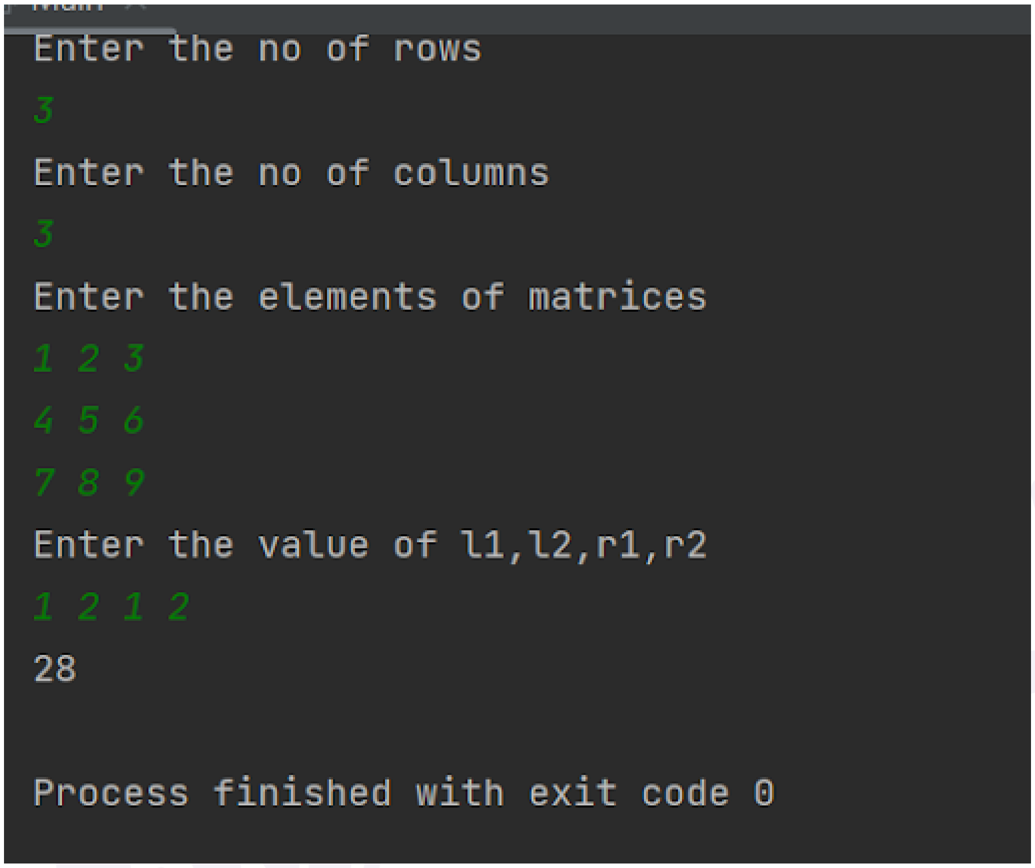


```
C:\Users\2018k\.jdks\openjdk-19.0.1\bin\java.exe "-javaa
 .jar=52124:C:\Program Files\JetBrains\IntelliJ IDEA Com
 -Dsun.stderr.encoding=UTF-8 -classpath C:\Users\2018k\I
Enter the no of rows
3
Enter the no of columns
3
Enter the elements of matrices
1 2 3
4 5 6
7 8 9
Enter the value of l1,l2,r1,r2
0 1 0 1
12

Process finished with exit code 0
```

Explanation Method 1: We add the value of each element in the given range from l1 to l2 and from r1 to r2. This can be done using a brute force approach where we use two nested for loops and keep adding the value of the elements to the answer.

Method 2: Pre-Calculating the horizontal sum for each row in the Matrix

```
import java.io.*;
import java.util.*;
public class Main
{
    public static int solve(int[][] a, int l1, int l2, int r1, int r2){
        int ans = 0;
        int n = a.length, m = a[0].length;

        // calculating the horizontal sum for each row in the Matrix

        for(int i = 0; i < n; i++) {       // 'i' is row
            for(int j = 1; j < m; j++) {     // 'j' is column
                a[i][j] += a[i][j-1];
            }
        }

        // now only traversing over rows through below for loop
        // as we have precalculated prefix matrix

        for(int i = l1; i <= l2; i++) {
            if(r1 >= 1)ans += a[i][r2] - a[i][r1-1];
            else ans += a[i][r2];
        }

        return ans;
    }

 public static void main(String[] args) {
   Scanner sc = new Scanner(System.in);
   System.out.println("Enter the no of rows");
   int row = sc.nextInt();
   System.out.println("Enter the no of columns");
   int col = sc.nextInt();
   int[][] matrix = new int[row][col];
   System.out.println("Enter the elements of matrices");
   for(int i=0;i<row;i++){
       for(int j=0;j<col;j++){
           matrix[i][j]=sc.nextInt();
     }
       }
       System.out.println("Enter the value of l1,l2,r1,r2");
       int l1 = sc.nextInt();
       int l2 = sc.nextInt();
       int r1 = sc.nextInt();
       int r2 = sc.nextInt();

       System.out.println(solve(matrix,l1,l2,r1,r2));
    }
}
```



```
Enter the no of rows
3
Enter the no of columns
3
Enter the elements of matrices
1 2 3
4 5 6
7 8 9
Enter the value of l1,l2,r1,r2
1 2 1 2
28

Process finished with exit code 0
```

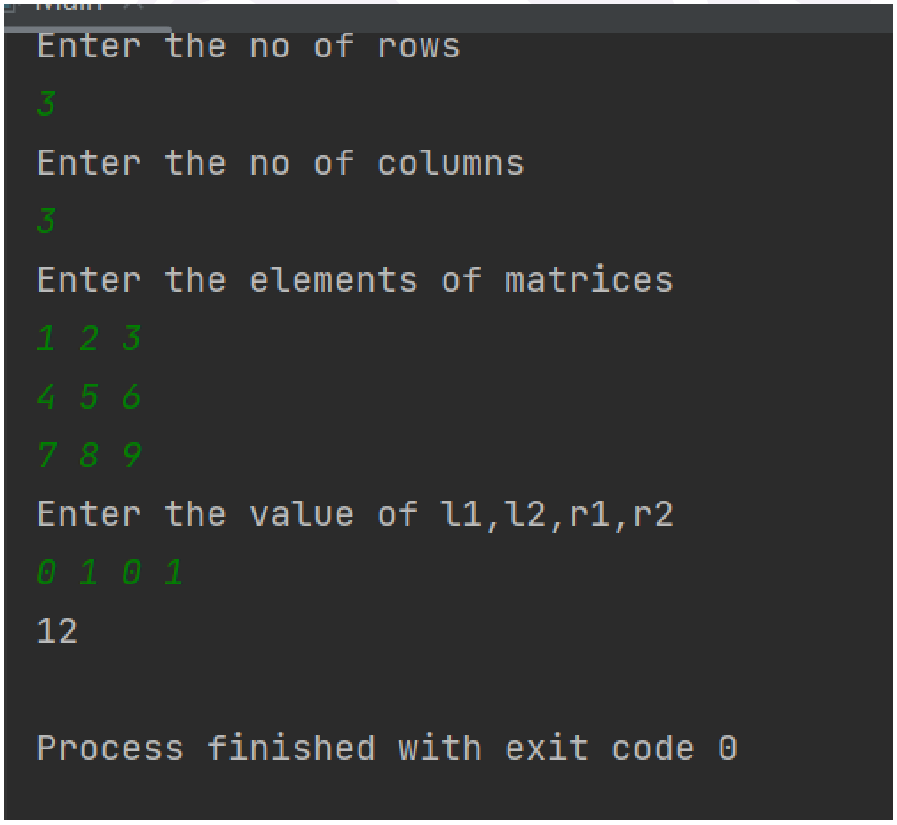
Explanation Method 2 : We add the value of each element in the given range from l1 to l2 and from r1 to r2. Rather than using the standard brute force approach we try to optimize it and pre-calculate the horizontal sum for each row in the matrix. This can be stored in the same matrix. Now we can traverse only the rows through a single for loop rather than using a nested for loop as we have a precalculated prefix matrix and calculate the answer using prefix sum technique.

Method 3: Prefix sum Over columns and Rows Both

Code:

```
import java.io.*;
import java.util.*;
public class Main
{
    public static int solve(int[][] a, int l1, int l2, int r1, int r2){
        int n = a.length, m = a[0].length;

        // making prefix in same matrix without constructing a new prefix matrix
        // prefix sum of both row and columns


        // horizontal prefix sum of every row is being calculated
        // i is row and j is column in below for loops
        for(int i = 0; i < n; i++) {
            for(int j = 1; j < m; j++) {
                a[i][j] += a[i][j-1];
            }
        }

        // vertical prefix sum
        for(int j = 0; j < m; j++) {
            for(int i = 1; i < n; i++) {
                a[i][j] += a[i-1][j];
            }
        }
 int left = 0, up = 0, leftup = 0;

        if(l1 >= 1) up = a[l1-1][r2];
        if(r1 >= 1) left = a[l2][r1-1];
        if(l1 >= 1 && r1 >= 1) leftup = a[l1-1][r1-1];


        int ans = a[l2][r2] - left - up + leftup;
        return ans;
    }
    public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       System.out.println("Enter the no of rows");
       int row = sc.nextInt();
        System.out.println("Enter the no of columns");
       int col = sc.nextInt();
        int[][] matrix = new int[row][col];
        System.out.println("Enter the elements of matrices");
        for(int i=0;i<row;i++){
            for(int j=0;j<col;j++){
                matrix[i][j]=sc.nextInt();
            }
        }
        System.out.println("Enter the value of l1,l2,r1,r2");
        int l1 = sc.nextInt();
        int l2 = sc.nextInt();
        int r1 = sc.nextInt();
        int r2 = sc.nextInt();

        System.out.println(solve(matrix,l1,l2,r1,r2));
    }
}
```

OUTPUT :



```
Enter the no of rows
3
Enter the no of columns
3
Enter the elements of matrices
1 2 3
4 5 6
7 8 9
Enter the value of l1,l2,r1,r2
0 1 0 1
12

Process finished with exit code 0
```

```
    public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       System.out.println("Enter the no of rows");
       int row = sc.nextInt();
        System.out.println("Enter the no of columns");
       int col = sc.nextInt();
        int[][] matrix = new int[row][col];
        System.out.println("Enter the elements of matrices");
        for(int i=0;i<row;i++){
            for(int j=0;j<col;j++){
                matrix[i][j]=sc.nextInt();
            }
        }
        System.out.println("Enter the value of l1,l2,r1,r2");
        int l1 = sc.nextInt();
        int l2 = sc.nextInt();
        int r1 = sc.nextInt();
        int r2 = sc.nextInt();

        System.out.println(solve(matrix,l1,l2,r1,r2));
    }
}
```

Explanation Method 3 : We add the value of each element in the given range from l1 to l2 and from r1 to r2. Now we even try to optimize the previous method by creating a prefix sum of both row and column in the same matrix. This can be precalculated as horizontal prefix sum for every particular row and vertical prefix sum for columns. Now we can calculate the final answer without using any for loop as we have pre calculated the prefix matrix.

Upcoming Class Teasers

- Arraylist in Java Collection Framework