Pascals travels

CONCEPT

Pascal's travels is a way to move from top-left corner to the bottom-right corner obeying rules of traveling within cells in the matrix.

RULES

1. We start in top-left corner
2. We move only in right and bottom directions.
3. Cell has an value by which you move either bottom or right
4. If move exceed position+matrix size in given direction we change direction if it is not possible we finish the move
5. When we find 0 we have found the path (right-bottom corner)

GOAL

Is to count all paths which are possible to move from top-left to right-bottom corner.

TECHNIQUE

In order to solve pascal travels we should use dynamic programming. Dynamic programming is idea of storing result of subproblems of bigger problem and then to solve bigger problems we use result of subproblems.

Solution

We implement two matrices one, which represents Pascals Travels matrix and second as a counting paths. Counting paths stores in cell how many paths went through cell. We use the result to sum up paths. We iterate over matrix from left-top corner to right-bottom starting from columns, then rows. We take value from counting_path_matrix[i][k] and move it by value from matrix[i][k] in bottom or right direction when its possible:
Conditions:
val_bottom = matrix[i][k] + I < size go bottom
Val_right = Matrix[k][i] + j < size go right

Then we sum values in counting_path_matrix using dynamic programming (using previous solved paths)

counting_path_matrix[val_bottom + I][k] += counting_path_matrix[i][k]

Or

counting_path_matrix[i][j+val_right] += counting_path_matrix[i][k]

After iterating loop result is printed counting_path_matrix[size][size]

ALGORITHM IN JAVA.

```
int sizeN;
int[][] matrix;
int[][] savedPaths;
int findPaths(){
    setMatrix();
    for(int i=0; i<sizeN;i++){
        for(int k=0; k<sizeN;k++){
            if(matrix[i][k]==0)
                return savedPaths[sizeN-1][sizeN-1];
            int value = matrix[i][k];
            if(value+k<sizeN){
                savedPaths[i][value+k] += savedPaths[i][k];
            }
            if(value+i<sizeN){
                savedPaths[value+i][k] += savedPaths[i][k];
            }
        }
    }
    return savedPaths[sizeN-1][sizeN-1];
}
```

EXAMPLE

iterating over columns and rows. Example is shorten due to i*k steps It is shown row by row.

We take value from saving paths matrix[i][j] and move and add to field by value inside cell of pascal travels matrix[i][j]] when we reach matrix[size][size] we finish the program and get value from matrix[size][size] its the number of paths .

Pascals travels matrix

2	3	1	1
2	2	1	2
1	2	1	1
2	2	2	0

1) saving paths matrix

1	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

2) saving paths matrix

1	0	1	1
0	0	1	1
1	0	0	0
0	0	0	0

3) saving paths matrix

1	0	1	1
0	0	1	2
1	0	1	0
0	0	0	2

4) saving paths matrix

1	0	1	1
0	0	1	2
1	1	1	2
1	0	1	4

5) saving paths matrix

1	0	1	1
0	0	1	2
1	1	1	2
1	0	2	4