```
public class Solution{
    public static int maxIndex(int[] row, int start, int end){
        int maxIndex = start;
        if(start == end){
            return start;
        }
        for(int i = start; i<=end; i++){
            if(row[i]>row[maxIndex]){
                maxIndex = i;
            }
        }
        return maxIndex;
    }
    public static int blockMaxValue(int[][] matrix, int startRow, int startCol, int
endRow, int endCol) {
        int middle = (int)Math.ceil(0.5*(endRow+startRow));
        int maxValue = matrix[startRow][startCol];
        int index = startCol;
        for(int i = startCol; i<=endCol; i++){ //finds the max value of the middle
row and the index
            if(matrix[middle][i] > maxValue) {
                maxValue =  matrix[middle][i];
                index = i;
            }
        }
        if(startRow == middle && endRow == middle){
            return maxValue;
        }

        if(startRow < middle){ //searching upper rows
            int recursiveCall = blockMaxValue(matrix, startRow, startCol, middle-1,
index);
            if (recursiveCall > maxValue) {
                return recursiveCall;
            }
        }

        if(endRow > middle){ //searching the lower rows
            int recursiveCall = blockMaxValue(matrix, middle+1, index, endRow,
endCol);
            if (recursiveCall > maxValue) {
                return recursiveCall;
            }
        }
        return maxValue;

    }

    public static int matrixMaxValue(int[][] matrix) {
        return blockMaxValue(matrix, 0, 0, matrix.length-1, matrix[0].length-1);
    }
}
```