FINDING THE MEDIAN OF THE ROW WISE SORTED MATRIX using binary search logic.

In the below program 'minimum' refers to the minimum value on the matrix which is found by comparing all the left most element values of the matrix.

and 'maximum' refers to the maximum value on the matrix which is found by comparing all the rightt most element values of the matrix.

Furtherly finding the all the numbers between the minimum and maximum and finding the median. Here an elemnt is finalized to be the median when it is greater than exactly half of the elemnts of the matrix.

In [5]:
```
# Python program to find median of matrix
# sorted row wise

from bisect import bisect_right as upper_bound
# For every number, to get the count of numbers less than or equal to that by using up

MAX = 2000; #specifying the constraint

# using the Function 'binaryMedian' to find median in the matrix.
def binaryMedian(matrix, row, column):
    minimum = matrix[0][0]
    maximum = 0
    for i in range(row):
        if matrix[i][0] < minimum:
            minimum = matrix[i][0]
        if matrix[i][column-1] > maximum :
            maximum = matrix[i][column-1]

    required = (row * column + 1) // 2  #here 'required'is the required index of the m

    while (minimum < maximum):
        mid = minimum + (maximum - minimum) // 2 #'mid' is the intented median.
        place = [0];

        # Finding the count of elements smaller than or equal to mid

        for i in range(row):
            j = upper_bound(matrix[i], mid)
            place[0] = place[0] + j
        if place[0] < required:
            minimum = mid + 1
        else:
            maximum = mid
    print ("Median is", minimum)
    return

# Driver code
row,column = 3, 3

matrix = [ [1, 3, 5], [2, 6, 9], [3, 6, 9]]
binaryMedian(matrix, row, column)
```

```
Median is 5
```

The given test case 1 passesd

similarly checking test case 2

In [8]:
```
# Python program to find median of matrix
# sorted row wise

from bisect import bisect_right as upper_bound
# For every number, to get the count of numbers less than or equal to that by using up

MAX = 2000; #specifying the constraint

# using the Function 'binaryMedian' to find median in the matrix.
def binaryMedian(matrix, row, column):
    minimum = matrix[0][0]
    maximum = 0
    for i in range(row):
        if matrix[i][0] < minimum:
            minimum = matrix[i][0]
        if matrix[i][column-1] > maximum :
            maximum = matrix[i][column-1]

    required = (row * column + 1) // 2  #here 'required'is the required index of the m

    while (minimum < maximum):
        mid = minimum + (maximum - minimum) // 2 #'mid' is the intented median.
        place = [0];

        # Finding the count of elements smaller than or equal to mid

        for i in range(row):
            j = upper_bound(matrix[i], mid)
            place[0] = place[0] + j
        if place[0] < required:
            minimum = mid + 1
        else:
            maximum = mid
    print ("Median is", minimum)
    return

# Driver code
row,column = 3, 1

matrix = [[1], [2], [3]]
binaryMedian(matrix, row, column)
```

```
Median is 2
```

The given test case 2 also passed

Hence claiming that the above written code is accurate.

In [ ]: