Brute:

def rowWithMax1s(matrix: [[int]], n: int, m: int) -> int:

    #    maxCount stores the maximum number of 1s found till now and ans is the index of that particular row.

    maxCount = 0

    ans = -1

    for i in range(n):

        #    Count for number of ones for the current row.

        curr = 0

        for j in range(m):

            #   Increment count if the value is 1.

            if matrix[i][j] == 1:

                curr += 1

        #    Update curr row and maximum count if the current row has the maximum number of ones.

        if curr > maxCount:

            ans = i

            maxCount = curr

    #    Return the row with maximum number of ones.

    return ans

    pass

Optimal:

def binarysearch(matrix,i,j):

    s=0

    e=j-1

    ans1=0

    ans2=0

    while(s<=e):

        mid=(s+e)//2

        if matrix[i][mid]==1:

            ans1=mid

            e=mid-1

        else:

            s=mid+1

    s=ans1

    e=j-1

    while(s<=e):

        mid=(s+e)//2

        if matrix[i][mid]==1:

            ans2=mid

            s=mid+1

        else:

            s=mid+1

    if ans2<ans1:

        return -1

    else:

        return (ans2-ans1+1)

def rowWithMax1s(matrix: [[int]], n: int, m: int) -> int:

    #    maxCount stores the maximum number of 1s found till now and ans is the index of that particular row.

    maxCount = 0

    ans = -1

    for i in range(n):

        #    Count for number of ones for the current row.

        curr = binarysearch(matrix,i,m)

        #    Update curr row and maximum count if the current row has the maximum number of ones.

        if curr > maxCount:

            ans =i

            maxCount = curr

    #    Return the row with maximum number of ones.

    return ans

    pass