PEAK ELEMENT IN 2D

☆ In this problem, we are given a 2D matrix of integers and we are supposed to return the row, column of an element which is greater than all its adjacent elements

Brute force solution is to obviously conduct a m x n search and have four checks as any element will have 4 adjacents. Time complexi-ty here is O(4mn). This can be improved by conducting a O(MN) search and returning the largest element as the Largest element will always be one of the peaks.

Optimal solution involves the use of Binary Search. We use a similar approach as we did in Peak Element in a 1D array.

We start by running a linear search in the first row to find the largest element as it will have the highest probability of being a peak. Then we find the largest element in its column and if it is greater than our current,

we run a check for the peak. If it passes, we return else we repeat the process

Pseudocode :

```
peakElement2D(arr, M, N) {
    low = 0, high = N - 1
    while (low <= high) {
        mid = (low + high) / 2
        row = maxElement(mat, M, N, mid)
        left = mid - 1 >= 0 ? arr[row][mid-1]
                            : -1
        right = mid - 1 >= 0 ? arr[row][mid+1]
                             : -1
        if (arr[row][mid] > left &&
                arr[row][mid] > right){
            return (row, mid)
        } else if (arr[row][mid] < left){
            high = mid - 1
        } else {
            low = mid + 1
        }
    }
}

maxElement(arr, M, N, col) {
    maxV, index = -1, -1
    for(i = 0 → M-1) {
        if(arr[i][col] > maxV) {
            maxV = arr[i][col]
            index = i
        }
    }
}
```

```
    return i
}
```