Strivers-A2Z-DSA-Sheet-main\02.Binary Search\2D Arrays\4.Peak_element_in_matrix.cpp

```
/*
QUESTION:
A peak element in a 2D grid is an element that is strictly greater than all of its adjacent neighbors to the left, right, top, and bottom.

Given a 0-indexed m x n matrix mat where no two adjacent cells are equal, find any peak element mat[i][j] and return the length 2 array [i,j].

You may assume that the entire matrix is surrounded by an outer perimeter with the value -1 in each cell.

You must write an algorithm that runs in O(m log(n)) or O(n log(m)) time.

Example 1:
Input: mat = [[1,4],[3,2]]
Output: [0,1]
Explanation: Both 3 and 4 are peak elements so [1,0] and [0,1] are both acceptable answers.

Example 2:
Input: mat = [[10,20,15],[21,30,14],[7,16,32]]
Output: [1,1]
Explanation: Both 30 and 32 are peak elements so [1,1] and [2,2] are both acceptable answers.

APPROACH:
- Perform a binary search on the columns of the matrix.
- Find the maximum element in each column and check if it is a peak element by comparing it with its adjacent elements.
- If it is a peak element, return its position [i, j].

TIME COMPLEXITY: O(m log(n)) or O(n log(m)) - Binary search is performed on the columns of the matrix.
SPACE COMPLEXITY: O(1) - Constant space is used.

CODE:
*/

int max_finder(vector<int>& row) {
    int maxi = INT_MIN;
    int ans = -1;
    for (int i = 0; i < row.size(); i++) {
        if (row[i] > maxi) {
            maxi = row[i];
            ans = i;
        }
    }
    return ans;
}

vector<int> findPeakGrid(vector<vector<int>>& mat) {
    int n = mat.size();
    int low = 0, high = n - 1;
    while (low <= high) {
```

```cpp
        int mid = low + (high - low) / 2;
        int col = max_finder(mat[mid]);
        if ((mid == 0 || mat[mid][col] > mat[mid - 1][col]) &&
            (mid == n - 1 || mat[mid][col] > mat[mid + 1][col]))
            return {mid, col};
        else {
            if (mid != 0 && mat[mid][col] < mat[mid - 1][col])
                high = mid - 1;
            else
                low = mid + 1;
        }
    }
    return {-1, -1};
}

/*
TIME COMPLEXITY: O(m log(n)) or O(n log(m)) - Binary search is performed on the columns of the matrix.
SPACE COMPLEXITY: O(1) - Constant space is used.
*/
```