Search In Sorted 2D Matrix

Problem statement Send feedback

You are given a 2D matrix ***'MATRIX'*** of ***'N'*'M'*** dimension. You must check whether a given number ***'target'*** is present in the matrix.

The following properties apply to the given matrix:

1. In each row, integers are sorted from left to right.
2. Each row's first integer is greater than the previous row's last integer.

Example:

Input:
'MATRIX' = [[1, 3, 5, 7], [10, 11, 16, 20], [23, 30, 34, 60]], 'TARGET' = 3
Output:1
Explanation: Since the given number 'TARGET' is present in the matrix, we return true.

Detailed explanation (Input/output format, Notes, Images)

Sample Input 1:

3 3
1 3 7
10 12 15
19 20 21
12

Sample Output 1:

1

Explanation Of Sample Input 1:

Input:
'MATRIX' = [[1, 3, 7], [10, 12, 15], [19, 20, 21]], 'TARGET' = 12
Output: 1

Explanation: Since the given number 'TARGET' is present in the matrix, we return true.

→ Brute force:

↳ Go to each and every cell and verify if the element is present in the matrix or not.

→ Optimal Approach:-

↳ Map the 2D array to 1D array.

↳ Then use binary search to get to the required result.

```
bool searchElement(vector<vector<int>> &MATRIX, int target) {
    int n = MATRIX.size();
    int m = MATRIX[0].size();
    int start = 0, end = n * m - 1;

    while(start <= end) {
        int mid = (start + end) >> 1;
        int i = mid / m;
        int j = mid % m;

        if(MATRIX[i][j] == target) return true;
        else if (MATRIX[i][j] > target) end = mid - 1;
        else start = mid + 1;
    }

    return false;
}
```