

33. Leftmost Column with at Least a One A row-sorted binary matrix means that all elements are 0 or 1 and each row of the matrix is sorted in non-decreasing order. Given a row-sorted binary matrix `binaryMatrix`, return the index (0-indexed) of the leftmost column with a 1 in it. If such an index does not exist, return -1. You can't access the Binary Matrix directly. You may only access the matrix using a `BinaryMatrix` interface: • `BinaryMatrix.get(row, col)` returns the element of the matrix at index (row, col) (0-indexed). • `BinaryMatrix.dimensions()` returns the dimensions of the matrix as a list of 2 elements [rows, cols], which means the matrix is rows x cols. Submissions making more than 1000 calls to `BinaryMatrix.get` will be judged Wrong Answer. Also, any solutions that attempt to circumvent the judge will result in disqualification. For custom testing purposes, the input will be the entire binary matrix `mat`. You will not have access to the binary matrix directly. Example 1: Input: `mat = [[0,0],[1,1]]` Output: 0 Example 2: Input: `mat = [[0,0],[0,1]]` Output: 1 Example 3: Input: `mat = [[0,0],[0,0]]` Output: -1

PROGRAM:

```
class BinaryMatrix:
```

```
    def __init__(self, mat):
```

```
        self.mat = mat
```

```
    def get(self, row, col):
```

```
        return self.mat[row][col]
```

```
    def dimensions(self):
```

```
        return [len(self.mat), len(self.mat[0])]
```

```
def leftMostColumnWithOne(binaryMatrix):
```

```
    rows, cols = binaryMatrix.dimensions()
```

```
    row, col = 0, cols - 1
```

```
    leftmost_col = -1
```

```
    while row < rows and col >= 0:
```

```
        if binaryMatrix.get(row, col) == 1:
```

```
            leftmost_col = col
```


```
            col -= 1
```

```
        else:
```

```
            row += 1
```

```
    return leftmost_col  
mat1 = [[0, 0], [1, 1]]  
mat2 = [[0, 0], [0, 1]]  
mat3 = [[0, 0], [0, 0]]  
  
binaryMatrix1 = BinaryMatrix(mat1)  
binaryMatrix2 = BinaryMatrix(mat2)  
binaryMatrix3 = BinaryMatrix(mat3)  
  
print(leftMostColumnWithOne(binaryMatrix1))  
print(leftMostColumnWithOne(binaryMatrix2))  
print(leftMostColumnWithOne(binaryMatrix3))
```

OUTPUT:



TIME COMPLEXITY:  $O(\text{rows} + \text{columns})$