378. Kth Smallest Element in a Sorted Matrix

Given an `n x n` `matrix` where each of the rows and columns are sorted in ascending order, return *the* `kth` *smallest element in the matrix*.

Note that it is the `kth` smallest element **in the sorted order**, not the `kth` **distinct** element.

Example 1:

Input: matrix = [[1,5,9],[10,11,13],[12,13,15]], k = 8
Output: 13
Explanation: The elements in the matrix are [1,5,9,10,11,12,13,**13**,15],
and the 8th smallest number is 13

Example 2:

Input: matrix = [[-5]], k = 1
Output: -5

Constraints:

- `n == matrix.length`
- `n == matrix[i].length`
- `1 <= n <= 300`
- `-109 <= matrix[i][j] <= 109`
- All the rows and columns of `matrix` are **guaranteed** to be sorted in **non-decreasing order**.
- `` `1 <= k <= n2` `````Java
  class Solution {
  // O(n^2 * log(max - min)), where n^2 comes from function smallerOrEqual()
  // assume k is valid, meaning n*n >= k
  public int kthSmallest(int[][] matrix, int k) {
  if(matrix == null || matrix[0].length == 0 ||
  matrix[0] == null || matrix[0].length == 0)
  return 0;

```
int n = matrix.length;
int l = matrix[0][0], r = matrix[n-1][n-1];

while(l < r){
    int m = l + (r - l) / 2;
    // find the number of elements <= m in matrix
```

```
            if(smallerOrEqual(matrix, m, n) < k) // Note don't do <= k
                l = m + 1;
            else
                r = m;
        }

        return l; // or return r
    }

    // start from top-right. Does this ring any bell?
    // It is the problem 240. Search a 2D Matrix II
    // https://leetcode.com/problems/search-a-2d-matrix-ii/
    // We search for the value but do not return the boolean,
    // what we do is to find num of elements that are
    // smaller or equal to the target number: m
    public int smallerOrEqual(int[][] matrix, int m, int n){
        int row = 0, col = n - 1, count = 0;
        while(row < n && col >= 0){
            if(matrix[row][col] <= m){ // then everything to matrix[row]
[col]'s left is <= m
                row++;
                count += (col + 1); // note index start at 0, so col+1
            }else{
                col--;
            }
        }
        return count; // number of elements smaller or equal than m
    }

}
```