Given an n x n matrix where each of the rows and columns is 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.

You must find a solution with a memory complexity better than O(n2).

**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 == 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

**Follow up:**

* Could you solve the problem with a constant memory (i.e., O(1) memory complexity)?
* Could you solve the problem in O(n) time complexity? The solution may be too advanced for an interview but you may find reading [this paper](http://www.cse.yorku.ca/~andy/pubs/X+Y.pdf) fun.