# 题目

给定一个 n x n 矩阵，其中每行和每列元素均按升序排序，找到矩阵中第k小的元素。

请注意，它是排序后的第k小元素，而不是第k个不同的元素。

**示例：**

matrix = [

[ 1, 5, 9],

[10, 11, 13],

[12, 13, 15]

],

k = 8,

返回 13。

**提示：**

你可以假设 k 的值永远是有效的，1≤k≤n2 。

# 分析

要解决“n x n升序矩阵中找第k小元素”的问题，核心思路是二分查找。利用矩阵“每行每列均升序”的特性，通过二分枚举可能的数值，统计矩阵中小于等于该数值的元素个数，逐步缩小范围，最终定位到第k小元素。

解题思路

1、二分查找的边界：

- 下界（left）：矩阵的最小元素（左上角matrix[0][0]，因矩阵升序）。

- 上界（right）：矩阵的最大元素（右下角matrix[n-1][n-1]，同理）。

2、关键：统计小于等于mid的元素个数：

对于二分枚举的中间值mid，需高效统计矩阵中≤mid的元素总数（记为count）：

- 利用矩阵“每行升序、每列升序”的特性，从左下角（i = n-1, j = 0）开始遍历：

若matrix[i][j] ≤ mid：当前列（j）所有元素（从i=0到i）均≤mid，count += i+1，并向右移动列（j++）。

若matrix[i][j] > mid：当前行（i）元素均> `mid`，向上移动行（i--）。

- 此遍历方式时间复杂度为O(n)（每行每列仅遍历一次），远快于暴力统计。

3、二分调整逻辑：

- 若count < k：第k小元素在mid右侧，更新left = mid + 1。

- 若count ≥ k：第k小元素在mid左侧或等于mid，更新right = mid。

- 当left == right时，left即为第k小元素。

代码：

class Solution {

public:

int kthSmallest(vector<vector<int>>& matrix, int k) {

int n = matrix.size();

int left = matrix[0][0]; // 二分下界：矩阵最小值

int right = matrix[n-1][n-1]; // 二分上界：矩阵最大值

while (left < right) {

int mid = left + (right - left) / 2; // 中间值（避免溢出）

// 统计矩阵中≤mid的元素个数

int count = countLessOrEqual(matrix, mid, n);

if (count < k) {

// 第k小元素在mid右侧，调整左边界

left = mid + 1;

} else {

// 第k小元素在mid左侧或等于mid，调整右边界

right = mid;

}

}

return left; // 最终left==right，即为第k小元素

}

private:

// 统计矩阵中≤target的元素总数（O(n)时间）

int countLessOrEqual(vector<vector<int>>& matrix, int target, int n) {

int i = n - 1; // 从左下角行开始（最后一行）

int j = 0; // 从左下角列开始（第一列）

int count = 0;

while (i >= 0 && j < n) {

if (matrix[i][j] <= target) {

// 当前列j的前i+1个元素（0~i行）均≤target，累加计数

count += i + 1;

j++; // 向右移动列，检查下一列

} else {

// 当前行i的元素均>target，向上移动行，检查上一行

i--;

}

}

return count;

}

};

代码解释

1、二分初始化：

left和right分别取矩阵的最小和最大值，确定二分查找的初始范围。

2、中间值计算：

用left + (right - left)/2而非(left+right)/2，避免left+right溢出（虽题目中矩阵元素范围未明确，但属于通用优化）。

3、计数函数（countLessOrEqual）：

- 从左下角（i=n-1, j=0）出发，利用矩阵升序特性快速统计：

当matrix[i][j] ≤ target时，当前列上方所有元素均更小，直接累加i+1个元素（行号从0到i），并右移列（j++）。

当matrix[i][j] > target时，当前行右侧元素均更大，上移行（i--）。

- 遍历终止条件：i < 0（行越界）或j >= n（列越界），确保覆盖所有可能元素。

4、二分调整：

通过count与k的比较，不断缩小left和right的范围，直到两者相等，此时的值即为第k小元素。

复杂度分析

- 时间复杂度：O(n log(max - min))，其中n是矩阵边长，max/min是矩阵最大/最小元素。

二分查找的次数：log(max - min)（矩阵元素范围通常较小，如int类型最多32次）。

每次二分的计数操作：O(n)（遍历一行一列）。

- 空间复杂度：O(1)，仅使用固定变量，无额外动态空间开销。