Given a *n* x *n* matrix where each of the rows and columns are sorted in ascending order, find 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:

```
matrix = [
   [ 1,  5,  9],
   [10, 11, 13],
   [12, 13, 15]
],
k = 8,

return 13.
```

```
class Solution {
    public int kthSmallest(int[][] matrix, int k) {
        PriorityQueue<int[]> q =
            new PriorityQueue<>(k, (a, b) -> matrix[a[0]][a[1]] - matrix[b[0]][b[1]]);
        q.offer(new int[]{0, 0});
        while (!q.isEmpty()) {
            int[] p = q.poll();
            k--;
            if (k == 0) {
                return matrix[p[0]][p[1]];
            }
            if (p[0] == 0 && p[1] + 1 < matrix[p[0]].length) {
                q.offer(new int[]{p[0], p[1]+1});
            }
            if (p[0] + 1 < matrix.length) {
                q.offer(new int[]{p[0]+1, p[1]});
            }
        }
        return -1;
    }
}
```

You are given two integer arrays **nums1** and **nums2** sorted in ascending order and an integer **k**.

Define a pair **(u,v)** which consists of one element from the first array and one element from the second array.

Find the k pairs **(u_1,v_1),(u_2,v_2) ...(u_k,v_k)** with the smallest sums.

Example 1:

```
Input: nums1 = [1,7,11], nums2 = [2,4,6], k = 3
Output: [[1,2],[1,4],[1,6]]
Explanation: The first 3 pairs are returned from the
sequence:
             [1,2],[1,4],[1,6],[7,2],[7,4],[11,2],[7,6],
[11,4],[11,6]
```

```
class Solution {
    public List<int[]> kSmallestPairs(int[] nums1, int[] nums2, int k) {
        PriorityQueue<int[]> que =
            new PriorityQueue<>((a,b) -> a[0] + a[1] - b[0] - b[1]);
        List<int[]> res = new ArrayList<>();
        if(nums1.length == 0 || nums2.length == 0 || k == 0)
            return res;
        for(int i = 0; i < nums1.length && i < k; i++)
            que.offer(new int[]{nums1[i], nums2[0], 0});
        while (k-- > 0 && !que.isEmpty()){
            int[] cur = que.poll();
            res.add(new int[]{cur[0], cur[1]});
            if (cur[2] == nums2.length-1)
                continue;
            que.offer(new int[]{cur[0], nums2[cur[2]+1], cur[2]+1});
        }
        return res;
    }
}
```