📘 LeetCode 378 – Kth Smallest Element in a Sorted Matrix

🔧 Approach Used: Max-Heap of Size `k`

✅ Problem Statement (Simplified):

> Matrix ke har row aur column ascending sorted hai.
>
> Hume `k-th` smallest value nikaalni hai.
>
> **Tera code max-heap se kar raha hai**.

🔂 Code Used:

```
class Solution {
public:
    int kthSmallest(vector<vector<int>>& matrix, int k) {
        priority_queue<int> pq; // Max-heap

        for(int i = 0; i < matrix.size(); i++) {
            for(int j = 0; j < matrix.size(); j++) {
                pq.push(matrix[i][j]);        // Push element
                if(pq.size() > k) pq.pop();    // Pop max if size > k
            }
        }

        return pq.top(); // Return k-th smallest
    }
};
```

🧠 Logic Behind Each Step:

Step	Code Line	Explanation
1️⃣	`priority_queue<int> pq;`	Max-heap banayi jisme top pe **largest** value hoti hai.
2️⃣	Loop on `i` and `j`	Matrix ke har element ko access kar raha hai nested loop se.
3️⃣	`pq.push(matrix[i][j]);`	Har element ko heap me insert kar raha hai.
4️⃣	`if(pq.size() > k) pq.pop();`	Agar size `k` se bada ho jaye toh sabse bada element (top) hata deta hai.
5️⃣	`return pq.top();`	Heap me ab sirf `k` smallest elements hain. Top = `k-th` smallest.

🔄 Dry Run (Example):

Input:

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

Process:

- Har element heap me gaya
- Heap size `k` se zyada hone par sabse bada element pop hota raha
- End me heap me 8 chhote elements bache
- Top = 8th smallest = `13`

⏱️ Time & Space Complexity:

Type	Complexity	Reason
⏳ Time	$O(n^2 \log k)$	n^2 elements, each heap op is `log k`

🧠 Space	`O(k)`	Max heap size `k` hi rakha

🔚 Summary:

- ✅ Simple & intuitive
- ✅ Solves problem correctly
- ⚠️ Not optimal for very large `n`
- 🔥 **Perfect for first try / brute-force level**