📘 LeetCode 973 – K Closest Points to Origin

✅ Problem Statement:

You are given an array of `points` where `points[i] = [xi, yi]` represents a point on the X-Y plane, and an integer `k`.

Return the `k` closest points to the origin `(0, 0)` using **Euclidean distance**.

> Distance formula:
>
> $\text{Distance} = \sqrt{x^2 + y^2}$
>
> Since square root preserves order, we can compare just x^2 + y^2

🧠 Thought Process:

1. **Goal**: Find the `k` points closest to the origin.
2. We don't need actual distance (no `sqrt`), only **relative comparison**, so we use `x² + y²`.
3. For every point, calculate its distance from the origin.
4. Use a **max-heap** to keep the `k` smallest distances.
5. If heap size > `k`, pop the point with the **largest distance**.
6. In the end, extract all points from the heap and return them.

✅ Code):

```
class Solution {
public:
    typedef pair<int,vector<int>> pi;
```

```
    vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {
        priority_queue<pi> pq;
        for(vector<int> v:points){
            int x = v[0];
            int y = v[1];
            int distance = x*x + y*y;
            pq.push({distance,v});
            if(pq.size()>k){
                pq.pop();
            }
        }
        vector<vector<int>> ans;
        while(pq.size()>0){
            ans.push_back(pq.top().second);
            pq.pop();
        }
        return ans;
    }
};
```

🧪 Dry Run:

Input: `points = [[1,3], [-2,2]]`, `k = 1`

1. **Point [1, 3]:**
   - Distance = $1^2 + 3^2 = 10$
   - Heap = `[{10, [1,3]}]`
2. **Point [-2, 2]:**
   - Distance = $(-2)^2 + 2^2 = 8$
   - Heap = `[{10, [1,3]}, {8, [-2,2]}]`
   - Heap size > k → pop max → remove `[1,3]`

Final heap = `[{8, [-2,2]}]`

Return `[[-2,2]]` ✅

📈 Time Complexity:

- **Heap operations**: For `n` points, each `push` / `pop` in heap of size `k` takes `O(log k)`
- **Overall traversal:** `O(n * log k)`
- **Result collection (k elements):** `O(k)`

Total: `O(n * log k)`

📦 Space Complexity:

- Heap stores at most `k` elements → `O(k)`
- Answer vector stores `k` elements → `O(k)`
- **Total:** `O(k)`
