**K-th type problems** from **simple linear to complex non-linear** based on their underlying data structures and algorithms:

**Linear Problems**

1. **Kth Largest Element in an Array** (LeetCode #215):
   * Find the k-th largest element in an unsorted array.
   * **Approach**: Sorting (O(n log n)) or Quickselect (O(n) average).
2. **Kth Smallest Element in a BST** (LeetCode #230):
   * Traverse a Binary Search Tree in-order to find the k-th smallest element.
   * **Approach**: In-order traversal (O(h + k)).
3. **Top K Frequent Elements** (LeetCode #347):
   * Find the k most frequent elements in an array.
   * **Approach**: Min-Heap or Bucket Sort (O(n log k) or O(n)).
4. **Kth Largest Element in a Stream** (LeetCode #703):
   * Maintain a running k-th largest element as new numbers are added.
   * **Approach**: Min-Heap (O(log k) per insertion).

**Moderately Complex Problems**

1. **Kth Smallest Element in a Sorted Matrix** (LeetCode #378):
   * Find the k-th smallest element in a row-and-column sorted matrix.
   * **Approach**: Min-Heap (O(k log n)) or Binary Search (O(n log(max-min))).
2. **K Closest Points to Origin** (LeetCode #973):
   * Find the k closest points to (0, 0).
   * **Approach**: Max-Heap or Quickselect (O(n log k)).
3. **Find Median from Data Stream** (LeetCode #295):
   * Dynamically track the median of a stream of numbers.
   * **Approach**: Two Heaps (O(log n) per operation).
4. **Find K Closest Elements** (LeetCode #658):
   * Find the k closest elements to a target in a sorted array.
   * **Approach**: Binary Search + Sliding Window (O(log n + k)).

**Complex Non-Linear Problems**

1. **Kth Ancestor of a Tree Node** (LeetCode #1483):
   * Find the k-th ancestor of a node in a tree.
   * **Approach**: Binary Lifting (O(log n) query after O(n log n) preprocessing).
2. **K-th Lexicographical String of All Happy Strings of Length n** (LeetCode #1415):
   * Generate the k-th lexicographical "happy string."
   * **Approach**: Backtracking (O(2^n)).
3. **Kth Smallest Number in Multiplication Table** (LeetCode #668):
   * Find the k-th smallest number in an m x n multiplication table.
   * **Approach**: Binary Search (O(m log(mn))).
4. **K-th Smallest in Lexicographical Order** (LeetCode #440):
   * Find the k-th smallest number in lexicographical order from 1 to n.
   * **Approach**: Trie-like DFS or Prefix Counting (O(log n)).
5. **Kth Symbol in Grammar** (LeetCode #779):
   * Find the k-th symbol in the n-th row of a grammar sequence.
   * **Approach**: Recursion or Iterative Simulation (O(n)).

This arrangement reflects the progression from **simple linear data structures** (arrays, heaps) to **complex non-linear problems** (trees, graphs, and backtracking). Let me know if you'd like to dive deeper into any of these problems!