**Day 5: Divide and Conquer**

**📊 1. Merge Sort**

**Problem Statement:** Sort large datasets (e.g. customer orders or log entries) efficiently using a divide-and-conquer approach.  
**Sample Input:** [64,34,25,12,22,11,90]  
**Expected Output:** [11,12,22,25,34,64,90]

**⚡ 2. Quick Sort**

**Problem Statement:** Rearrange product prices for display by recursively partitioning around pivots — ideal for quick bulk sorting.  
**Sample Input:** [64,34,25,12,22,11,90]  
**Expected Output:** [11,12,22,25,34,64,90]

**🔍 3. Binary Search**

**Problem Statement:** Search for an item ID in a sorted inventory list quickly using binary partitioning.  
**Sample Input:** arr = [1,2,3,4,5,6,7,8,9], target = 5  
**Expected Output:** 4

**🔄 4. Search in Rotated Sorted Array**

**Problem Statement:** Locate a product in a rotated shipment manifest using modified binary search.  
**Sample Input:** nums = [4,5,6,7,0,1,2], target = 0  
**Expected Output:** 4

**📈 5. Median of Two Sorted Arrays**

**Problem Statement:** Find the median salary between two sorted employee datasets.  
**Sample Input:** nums1 = [1,3], nums2 = [2]  
**Expected Output:** 2.0

**🔢 6. Kth Largest Element**

**Problem Statement:** Identify the second-highest bid in an auction without fully sorting the bids.  
**Sample Input:** nums = [3,2,1,5,6,4], k = 2  
**Expected Output:** 5

**🔁 7. Count Inversions**

**Problem Statement:** Detect how out-of-order a list of task priorities is — useful for measuring deviation from optimal order.  
**Sample Input:** [2,3,8,6,1]  
**Expected Output:** 5

**🗳️ 8. Majority Element (Divide & Conquer)**

**Problem Statement:** Determine the most frequent response in a feedback dataset using recursive frequency checks.  
**Sample Input:** [2,2,1,1,1,2,2]  
**Expected Output:** 2

**📍 9. Closest Pair of Points**

**Problem Statement:** In a drone fleet, find the closest pair of drones based on GPS coordinates.  
**Sample Input:** [[2,3],[12,30],[40,50],[5,1],[12,10],[3,4]]  
**Expected Output:** 1.414...

**🧬 10. Merge K Sorted Lists**

**Problem Statement:** Combine multiple sorted appointment queues into a single streamlined schedule.  
**Sample Input:** [[1,4,5],[1,3,4],[2,6]]  
**Expected Output:** Linked list of [1,1,2,3,4,4,5,6]

**⛰️ 11. Find Peak Element**

**Problem Statement:** Locate the highest point in a fluctuating sensor reading using minimal checks.  
**Sample Input:** [1,2,3,1]  
**Expected Output:** 2

**🔋 12. Power(x, n)**

**Problem Statement:** Calculate exponential growth or decay, such as interest rate modeling, using optimized recursion.  
**Sample Input:** x = 2.0, n = 10  
**Expected Output:** 1024.0

**📉 13. Find Minimum in Rotated Array**

**Problem Statement:** Find the earliest shipment arrival in a rotated delivery schedule.  
**Sample Input:** [3,4,5,1,2]  
**Expected Output:** 1

**📈 14. Maximum Subarray (Divide & Conquer)**

**Problem Statement:** Identify the most profitable span in sales data using recursive comparison.  
**Sample Input:** [-2,1,-3,4,-1,2,1,-5,4]  
**Expected Output:** 6

**🌳 15. Convert Sorted Array to BST**

**Problem Statement:** Build a balanced search tree from a sorted customer ID list for fast lookup.  
**Sample Input:** [-10,-3,0,5,9]  
**Expected Output:** BST root pointing to a balanced tree.

**🌱 16. Construct Tree from Preorder & Inorder**

**Problem Statement:** Reconstruct a dependency tree from its traversal logs — useful in recovery or parsing.  
**Sample Input:** preorder = [3,9,20,15,7], inorder = [9,3,15,20,7]  
**Expected Output:** Root node of the constructed binary tree.

**🔡 17. Longest Common Prefix (Divide & Conquer)**

**Problem Statement:** Extract common prefixes across code snippets for auto-grouping or tagging.  
**Sample Input:** ["flower", "flow", "flight"]  
**Expected Output:** "fl"

**🔄 18. Number of Reverse Pairs**

**Problem Statement:** Identify pairs of sales where earlier entries are significantly larger — used in trend reversal detection.  
**Sample Input:** [1,3,2,3,1]  
**Expected Output:** 2

**📏 19. Smallest Range from K Lists**

**Problem Statement:** Find the tightest range that includes at least one price from each supplier — ideal for price comparison engines.  
**Sample Input:** [[4,10,15,24,26],[0,9,12,20],[5,18,22,30]]  
**Expected Output:** [20,24]

**🏙️ 20. Skyline Problem**

**Problem Statement:** Generate the city skyline silhouette based on building dimensions — used in simulation and rendering.  
**Sample Input:** [[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]  
**Expected Output:** [[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]