Problem 1: Heapsort

1. Convert to Max Heap:

Given array: [9, 5, 2, 11, 7, 6, 14, 1, 3]

١. Heapify index 2: [9, 5, 14, 11, 7, 6, 2, 1, 3]

٢. Heapify index 1: [9, 11, 14, 5, 7, 6, 2, 1, 3]

٣. Heapify index 0: [14, 11, 9, 5, 7, 6, 2, 1, 3]

Max Heap: [14, 11, 9, 5, 7, 6, 2, 1, 3]

2. Heapsort Steps:

١. Swap 14 ↔ 3, Heapify → [11, 7, 9, 5, 3, 6, 2, 1, 14]

٢. Swap 11 ↔ 1, Heapify → [9, 7, 6, 5, 3, 1, 2, 11, 14]

٣. Swap 9 ↔ 2, Heapify → [7, 5, 6, 2, 3, 1, 9, 11, 14]

٤. Swap 7 ↔ 1, Heapify → [6, 5, 1, 2, 3, 7, 9, 11, 14]

٥. Swap 6 ↔ 3, Heapify → [5, 3, 1, 2, 6, 7, 9, 11, 14]

٦. Swap 5 ↔ 2, Heapify → [3, 2, 1, 5, 6, 7, 9, 11, 14]

٧. Swap 3 ↔ 1, Heapify → [2, 1, 3, 5, 6, 7, 9, 11, 14]

٨. Swap 2 ↔ 1, Heapify → [1, 2, 3, 5, 6, 7, 9, 11, 14]

Sorted Array: [1, 2, 3, 5, 6, 7, 9, 11, 14]

3. Worst-case Complexity:

O(nlog⁡n)O(n \log n)

4. When is Heapsort Preferred Over Quicksort?

Stable worst-case O(nlog⁡n)O(n \log n)
Uses less memory (in-place)
Better for real-time systems

Problem 2: Counting Sort with Negative Numbers

1. Modify Counting Sort for Negatives:

١. Find min = -10, max = 10, shift values by +10

٢. Apply Counting Sort as usual

٣. Shift values back

Sorted Array: [-10, -5, -3, -1, 0, 5, 8, 10]

2. Why is Counting Sort Inefficient for Large Ranges?

Consumes too much memory if range is large

3. Is it suitable for 1M numbers from -100,000 to 100,000?

No, requires 200,001 extra space, inefficient.

Problem 3: Radix Sort vs. Merge Sort

1. Why is Radix Sort good for 1M 9-digit integers?

Linear time O(d(n+k))O(d(n + k)), better for fixed-length numbers

2. Number of Passes (Base 10 vs. Base 256)?

Base 10 → 9 passes, Base 256 → 3-4 passes

3. Radix Sort on [234, 455, 224, 323, 123] (Base 10):

١. **Sort by 1s digit:** [220, 221, 222, 223, 224]

٢. **Sort by 10s digit:** [220, 221, 222, 223, 224]

٣. **Sort by 100s digit:** [220, 221, 222, 223, 224]

Sorted Array: [123, 224, 234, 323, 455]

4. Radix Sort vs. Merge Sort for 100M Integers?

Radix Sort is faster for fixed-length numbers
Merge Sort is better for large arbitrary data