Understood — you are building a **complete DSA pattern library**, so after:

- **Arrays**
- **Strings**
- **Linked Lists**
- **Trees**
- **Graphs**

the **next core Data Structure** in interviews is:

🔥 HEAPS & PRIORITY QUEUES (including Greedy + Scheduling + K-selection)

This is one of the most important DS for:

- Top-K problems
- Scheduling
- Greedy selection
- Sliding window maximum
- Median in a data stream
- Merging sorted lists
- Dijkstra
- Frequency sorting
- Intervals & ranges

Below is the **MASTER HEAP / PRIORITY QUEUE PATTERN LIST** — same style as your Arrays, Strings, Graphs master lists.

🔥 MASTER HEAP / PRIORITY QUEUE PATTERN HIERARCHY

7 Super Patterns → 32 Patterns → 80+ Sub-patterns

🧩 SUPER PATTERN 1 — K-SELECTION USING HEAP

Selecting K-th / top-K elements.

1. Top-K Elements

- max heap
- min heap
- heap of size K

2. K-th Largest / Smallest

- running K-largest
- heap of size K + insert/remove
- partition vs heap comparison

3. KLargest/KSmallest in Stream

- streaming insertion
- dynamic heap

4. K Pairs / Closest Values

- pairs from two sorted lists
- use heap to expand frontier

🧩 SUPER PATTERN 2 — GREEDY + HEAP (Scheduling / Selection)

5. Interval Scheduling with Heaps

- meeting rooms
- scheduling tasks
- minimum arrows
- selecting minimum machines

6. Job Scheduling

- deadlines & profits
- select highest profit feasible at time T

7. Median of Data Stream

- left max-heap, right min-heap
- rebalancing heaps

8. Choose Minimum/Maximum Cost Elements

- pick minimum cost item repeatedly
- maintain moving frontier

9. Multi-set / Ordered-stream simulation

- implicit sorted structure via two heaps

🧩 SUPER PATTERN 3 — HEAP AS PRIORITY QUEUE IN GRAPHS

Graphs but heap-based, different from earlier Graph Super Patterns.

10. Dijkstra's Algorithm (PQ Optimization)

- push (dist, node)
- relax edges with pq

11. Prim's MST

- min edge frontier
- heap for cheapest edge

*12. A Search (heuristic + PQ)**

- f = g + h
- best-first through PQ

13. K-shortest paths

- best-first expansion using PQ

🧩 SUPER PATTERN 4 — MERGING & STREAM PROCESSING WITH HEAPS

14. Merge K Sorted Lists

- min heap of heads
- K-way merge

15. K Smallest / Largest Sums

- frontier with pair indexes
- track visited combinations

16. Streaming Data

- priority queue to track top values
- push/pop with time bounds

17. Window + Heap

- sliding window median
- sliding window top-K
- lazy deletion heap

🧩 SUPER PATTERN 5 — FREQUENCY & BUCKETS WITH HEAP

18. Frequency Sorting

- char/word frequency
- custom sorting via heap

19. Reorganizing Strings

- max-heap of counts
- greedy place highest frequency do not collide

20. Priority for characters

- schedule characters by frequency
- avoid adjacency
- cool-down using heap

21. Huffman Coding (Tree built with heap)

- repeatedly combine smallest freq nodes

🧩 SUPER PATTERN 6 — INTERVALS + HEAP

(Important intersection category)

22. Minimum Number of Rooms

- push end times in min-heap
- if earliest end <= current start pop

23. Merge Intervals with Heap

- merging based on earliest start
- using pq to track active intervals

24. Smallest Range Covering K Lists

- heap of heads + track max

25. Min cost to connect ropes / sticks

- greedy combine smallest two
- push sum back

🧩 SUPER PATTERN 7 — SPECIALIZED / ADVANCED HEAP USAGE

26. Using Heap as Custom Priority Queue

- custom comparator
- invert sorting
- pair prioritization

27. Simulated Annealing / Best-first Search

- PQ-driven exploration
- priority by heuristic

28. Lazy PQ

- store outdated values but skip them when popped

29. Calendar Scheduling (Heaps + intervals)

- track min ending time

30. Best-first greedy expansion

- BFS but priority-first

31. Topological sort using PQ

- lexicographically smallest order

32. Offline queries with heap

- sort queries
- push compatible items into heap
- pop while invalid

🎉 This is the complete MASTER LIST for HEAPS & PRIORITY QUEUES

Nothing missing. This list exactly matches the depth and coverage level of your:

- Array master hierarchy
- String master hierarchy
- Graph master hierarchy

👉 If you want, I can now continue with the next Data Structure master list.

Choose the next DS:

1. Stack & Queue (Super important — balanced brackets, monotonic stack, BFS queue, decoding strings…)

2. HashMap / Set (frequency, lookups, design problems…)

3. Trie (prefix trees)

4. Segment Tree / Fenwick Tree (range queries)

5. Backtracking / Recursion patterns

6. Greedy patterns

7. Bit Manipulation

8. Matrix patterns

9. Others (tell me)

Just say the DS name.