

Chapter 9

■ Patterns & Connections

Stack and Queue patterns that appear across hundreds of interview problems

■ The Monotonic Stack Pattern

A Monotonic Stack maintains elements in increasing or decreasing order. It's used to find Next Greater/Smaller elements efficiently. Template: Iterate, pop elements that don't satisfy your monotonic property, the answer for those popped elements is the current element.

Problems using Monotonic Stack:

- Next Greater Element I & II (LeetCode 496, 503)
- Trapping Rain Water (LC 42) — classic hard
- Largest Rectangle in Histogram (LC 84)
- Daily Temperatures (LC 739)
- Remove K Digits (LC 402)

■ BFS using Queue

Breadth-First Search **REQUIRES** a queue. Every level-order traversal problem is BFS under the hood. The pattern: 1. Enqueue root 2. While queue not empty: dequeue node, process it, enqueue its children 3. Each 'round' of dequeuing = one level of the graph/tree

Problems using BFS Queue:

- Binary Tree Level Order Traversal (LC 102)

- Rotting Oranges (LC 994) — multi-source BFS
- Word Ladder (LC 127)
- Shortest Path in a Grid
- Clone Graph (LC 133)

■■ Pattern Recognition Map

Problem Pattern	Data Structure	Key Idea
Matching brackets/parentheses	Stack	Push openers, pop on closer
Undo/Redo functionality	Stack	Push actions, pop to undo
Next greater/smaller element	Monotonic Stack	Maintain decreasing/increasing order
Function call simulation	Stack	Simulate recursion iteratively
Level-order tree traversal	Queue	BFS — process level by level
Shortest path in unweighted graph	Queue (BFS)	BFS guarantees shortest path
Task scheduling / round-robin	Queue	FIFO ensures fair scheduling
Sliding window maximum	Deque	Monotonic deque for O(n) solution
Expression evaluation	Two Stacks	One for operands, one for operators
Design LRU Cache	Queue + HashMap	Queue for order, map for O(1) access