Stacks & Queues in DSA: Beginner to Advanced

Subtitle: Learn how to use stacks and queues effectively, recognize patterns, and solve common coding interview problems.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Stacks and queues are fundamental data structures that help you manage data in **specific orders**. Understanding them is crucial for solving real-world problems like undo/redo functionality, task scheduling, and breadth/depth traversal in algorithms. This guide will teach you everything from basics to practical problems.

Step 1: What is a Stack?

A **stack** is a **LIFO (Last In First Out)** data structure.

- The last element added is the first to be removed
- Operations:
  - push() → Add element
  - pop() → Remove top element
  - peek() → View top element

Example:

```
let stack = [];

stack.push(10);

stack.push(20);

console.log(stack.pop()); // 20

console.log(stack[stack.length - 1]); // peek → 10
```

Step 2: What is a Queue?

A **queue** is a **FIFO (First In First Out)** data structure.

- The first element added is the first to be removed
- Operations:
  - enqueue() → Add element
  - dequeue() → Remove element

Example:

```
let queue = [];

queue.push(10); // enqueue

queue.push(20);

console.log(queue.shift()); // dequeue → 10
```

Step 3: Stack Problems & Patterns

1. Balanced Parentheses

```
function isBalanced(str) {
  let stack = [];
  let map = { '(': ')', '{': '}', '[': ']' };

  for (let char of str) {
    if (map[char]) {
      stack.push(char);
    } else {
      if (char !== map[stack.pop()]) return false;
    }
  }
  return stack.length === 0;
}
```

2. Reverse a String Using Stack

```
let stack = [];

for (let char of str) stack.push(char);

let reversed = '';

while (stack.length) reversed += stack.pop();
```

Step 4: Queue Problems & Patterns

1. Sliding Window Maximum

- Useful for **fixed size subarray problems**

2. BFS (Breadth First Search)

- Queue is used to traverse level by level in **trees/graphs**

Example:

```
let queue = [root];

while (queue.length) {

  let node = queue.shift();

  console.log(node.val);

  if (node.left) queue.push(node.left);

  if (node.right) queue.push(node.right);

}
```

Step 5: Common Mistakes

- Using the wrong data structure for a problem
- Not understanding LIFO/FIFO
- Ignoring edge cases (empty stack/queue)
- Using inefficient operations (shift() in large arrays can be slow in JS)

Step 6: Practice Plan

Stacks

- Day 1: Push, pop, peek → 5 problems
- Day 2: Balanced parentheses + string reversal → 5–7 problems

Queues

- Day 3: Basic enqueue/dequeue → 5 problems
- Day 4: Sliding window + BFS → 5–7 problems
- Day 5: Mixed stack + queue problems → 10 problems

Step 7: Mini Exercises

1. Implement stack using **linked list**
2. Implement queue using **stack**
3. Evaluate postfix expression using stack
4. First-in-first-out task scheduler using queue

Key Takeaways

- **Stacks = LIFO**, **Queues = FIFO**
- Identify problems that match these patterns
- Mastering these makes **trees, graphs, and algorithm questions easier**
- Practice consistently with **edge cases and variations**

Visit haas.dev for more DSA practice problems, coding guides, and interview preparation material.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app