

Chapter 5

■ Dry Run

Walk through problems step by step — see every variable change

■ Dry Run 1: Valid Parentheses

Problem: Given string "`{ [ ] }`", check if all brackets are properly matched and closed.

Algorithm:

- 1. Traverse character by character
- 2. If opening bracket → push onto stack
- 3. If closing bracket → check if stack top is matching opener
- 4. If mismatch or stack empty → return false
- 5. At end, stack must be empty → return true

Char	Action	Stack (top→)	Valid?
(	push (	(	✓
{	push {	{ (	✓
[	push [	[{ (	✓
]	top=[, matches]	{ (	✓
}	top={, matches }	(	✓
)	top=(, matches)	(empty)	✓
END	stack empty	(empty)	TRUE ✓

■ Dry Run 2: Next Greater Element

Problem: For array [4, 5, 2, 10, 8], find the next greater element for each index.

Approach: Monotonic Stack — traverse right to left.

Index	Element	Stack (indices)	NGE
4	8	[4]	-1
3	10	[3]	-1 (stack has 8<10, pop; stack empty)
2	2	[3,2]	10 (top is index 3 → val 10 > 2)
1	5	[3,1]	10 (top index 3 → val 10 > 5)
0	4	[3,1,0]	5 (top index 1 → val 5 > 4)

Result:

NGE = [5, 10, 10, -1, -1] (4→5, 5→10, 2→10, 10→-1, 8→-1)

■ Dry Run 3: Queue using Two Stacks

Simulate enqueue(1), enqueue(2), enqueue(3), dequeue(), enqueue(4), dequeue()

Operation	Stack1 (in)	Stack2 (out)	Returned
enqueue(1)	[1]	[]	—
enqueue(2)	[1,2]	[]	—
enqueue(3)	[1,2,3]	[]	—
dequeue()	[] (poured)	[3,2,1]	1 (pop from S2)

enqueue(4)	[4]	[3,2]	—
dequeue()	[4]	[3,2]	2 (pop from S2)

Key Insight: Elements in Stack2 are in FIFO order because we reversed the reversal. Pouring S1 into S2 reverses order once → FIFO restored!