

Stack – Stack Using Array

1. Introduction

A **Stack using Array** is an implementation of the stack data structure where elements are stored in a **linear array**.

All stack operations follow the **LIFO (Last In, First Out)** principle and are performed using an index variable called **Top**.

This is the **simplest and most commonly taught** stack implementation.

2. What is Stack Using Array?

In this implementation:

- A fixed-size array is used to store elements
 - An integer variable top keeps track of the **topmost element**
 - Initially, top = -1 indicating an empty stack
-

3. Stack Structure (Conceptual)

```
Array: [ 10 | 20 | 30 |   ]
```

```
Index:  0   1   2
```

```
Top → 2
```

Only the element at **top index** is accessible.

4. Components of Stack Using Array

- **Array** → Stores stack elements
 - **Top** → Points to the last inserted element
 - **Size** → Maximum capacity of the stack
-

5. Push Operation (Array Stack)

Logic (Plain English)

1. Check if $\text{top} == \text{size} - 1$
2. If true \rightarrow Stack Overflow
3. Otherwise:
 - Increment top
 - Insert element at $\text{array}[\text{top}]$

Example

```
Push 40
Top moves from 2 → 3
```

6. Pop Operation (Array Stack)

Logic (Plain English)

1. Check if $\text{top} == -1$
2. If true \rightarrow Stack Underflow
3. Otherwise:
 - Remove element at $\text{array}[\text{top}]$
 - Decrement top

Example

```
Pop → removes 30
Top moves from 2 → 1
```

7. Peek Operation

Logic (Plain English)

1. Check if stack is empty
 2. If not empty, return array[top]
 3. Stack remains unchanged
-

8. isEmpty Operation

- Stack is empty if:

```
top == -1
```

Used before pop or peek operations.

9. isFull Operation

- Stack is full if:

```
top == size - 1
```

Used before push operation.

10. Stack Overflow in Array Stack

Stack Overflow occurs when:

- Trying to push an element into a full array stack

Reason:

- Array has fixed size
-

11. Stack Underflow in Array Stack

Stack Underflow occurs when:

- Trying to pop an element from an empty stack
-

12. Time and Space Complexity

Operation	Time Complexity
Push	O(1)
Pop	O(1)
Peek	O(1)

- **Space Complexity:** $O(n)$

(where n is stack size)

13. Advantages of Stack Using Array

- Simple and easy to implement
 - Fast operations
 - Less memory overhead
 - Suitable for fixed-size problems
-

14. Limitations of Stack Using Array

- Fixed size (cannot grow dynamically)
 - Overflow occurs if size exceeded
 - Memory may be wasted if stack is underutilized
-

15. Real-World Applications

- Expression evaluation
 - Function call stack (conceptually)
 - Undo/Redo operations
 - Syntax checking
 - Educational demonstrations
-

16. Comparison with Stack Using Linked List

Feature	Array Stack	Linked List Stack
Size	Fixed	Dynamic
Memory	Contiguous	Non-contiguous
Overflow	Possible	Rare
Implementation	Simple	Moderate

17. Summary

- Stack using array stores elements in a fixed-size array
 - Uses top to track elements
 - Push, pop, and peek are $O(1)$
 - Easy to implement but size is limited
-