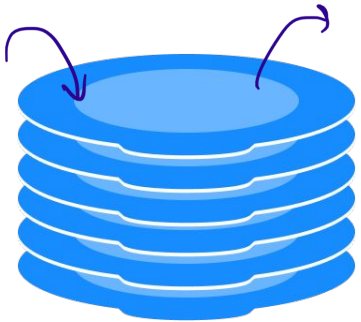
Stack Basics

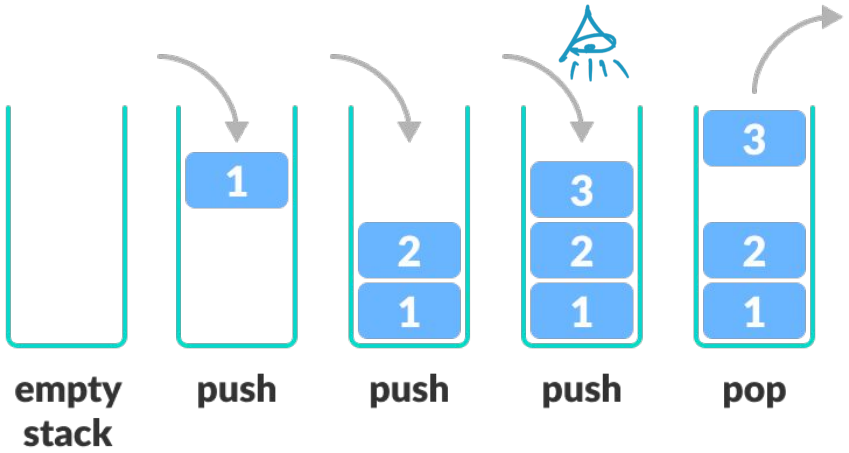
Stack Data Structure

Linear

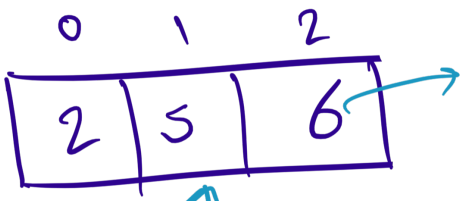
LIFO

Last In First Out





Implement Stack Using Array



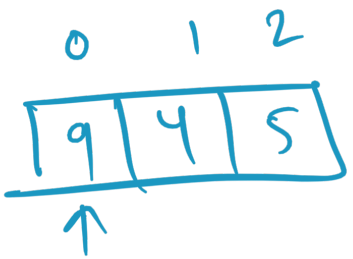
6

3 - size
↳ capacity.

top = -X → 0 → X → 2 → X → 2 → 1

push(2); ✓
push(5); ✓
push(4); ✓
push(1); X

pop() pop();
push(6);



top = 0

Capacity = 3

ans = 2

push(2)
push(4)
push(1)

== push(8)

pop()
push(5)

pop() pop()
push(9);

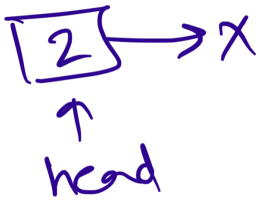
Peek()

Insertion at head ⟷ Stack push()

Implement Stack Using Linked List

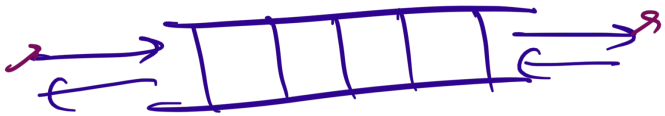
size

any = 5

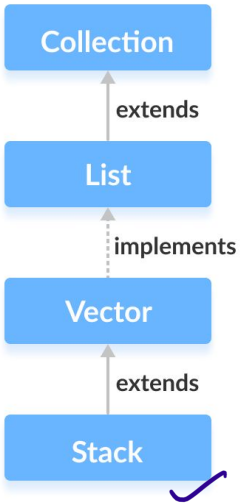


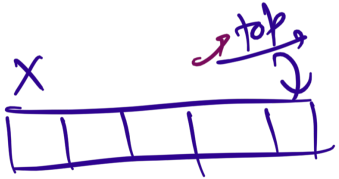
push(2)
push(5)
push(1)
pop();
pop();

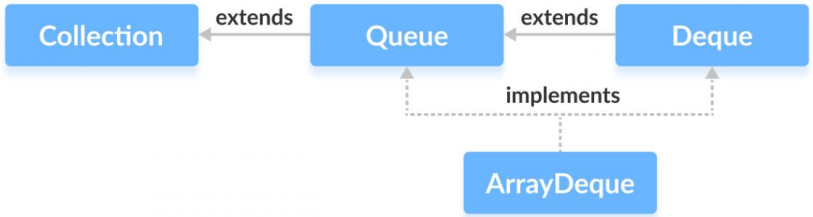
Deletion at head ⟷ stack pop()



Stack & ArrayDeque in Collection Framework







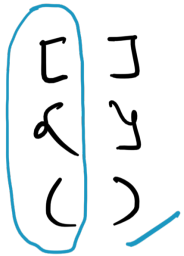
- `push()` - adds an element to the top of the stack
- `peek()` - returns an element from the top of the stack
- `pop()` - returns and removes an element from the top of the stack

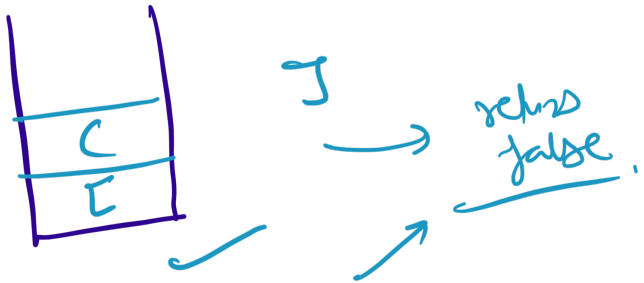
Parenthesis Matching Problem

"[a()b()]"

"[()(a]"

Using stack → O(N).





"(( ))[ ]"

top = '['

Previous Smaller Element

LIFO.

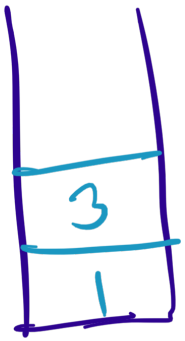
a[] = { 5, 1, 2, 3, 4, 6, 10, 4, 3 }

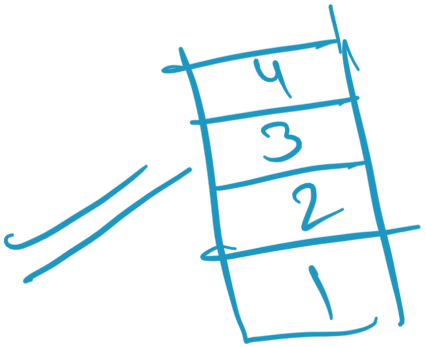
ans => { -1, -1, 1, 2, 3, 3, -1, 1, 1 }

Stack ⟶ O(N)!

{ 1, 2, 3, 4 }
{ -1, 1, 2, 3 }

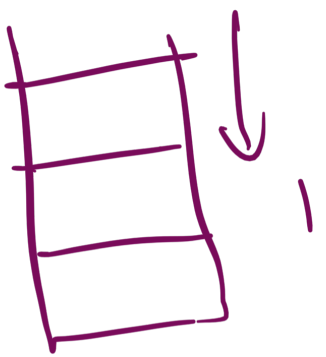
increasing order





a[] = { 4, 1, 3, 5, 2, 1 }

ans[] = { -1, -1, 1, 3, 1, -1 }



1

O(N).

Practice Problems

1. Previous / Next Greater Element
2. Previous / Next Smaller Element
3. Reverse words in a given Sentence.
4. Reverse a stack using Recursion
5. Reverse a stack without using extra space
6. Delete the mid element from a given Stack