Stack

What is stack?

A Stack is a **linear data structure** that follows a particular order in which the operations are performed.
The order may be:

- **LIFO(Last In First Out)**
- **FILO(First In Last Out)**

Basic operations we can do on a stack

- **Push**: Adds a new element on the stack.
- **Pop**: Removes and returns the top element from the stack.
- **Peek**: Returns the top element on the stack.
- **isEmpty**: Checks if the stack is empty.
- **Size**: Finds the number of elements in the stack.

Here you can visualise these operations:
https://visualgo.net/en/list?mode=Stack

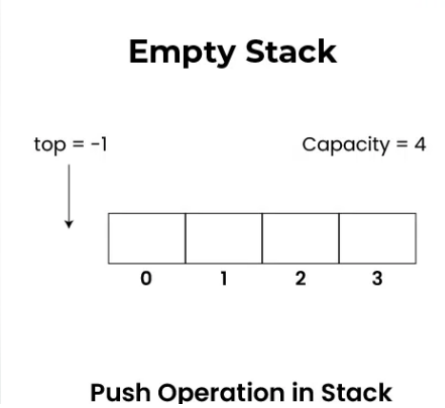
Types of stack

1. **Fixed size stack**- stack has a fixed size and cannot grow or shrink dynamically.
2. **Dynamic size stack**- stack can grow or shrink dynamically.

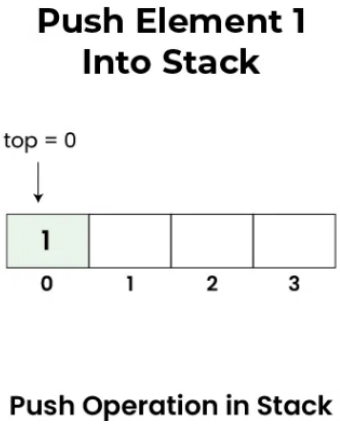
Implementation of stack

Using array

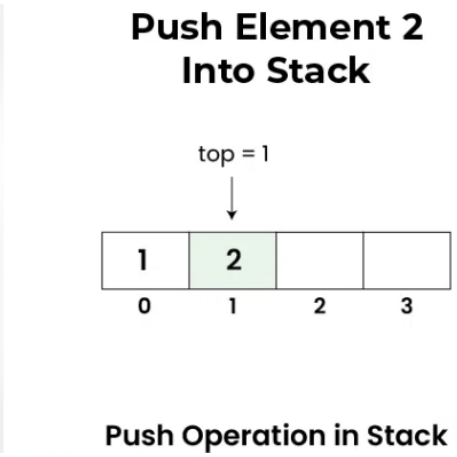
Step 1:



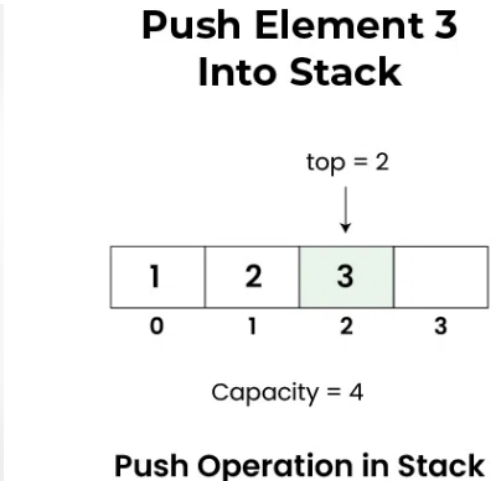
Step 2:



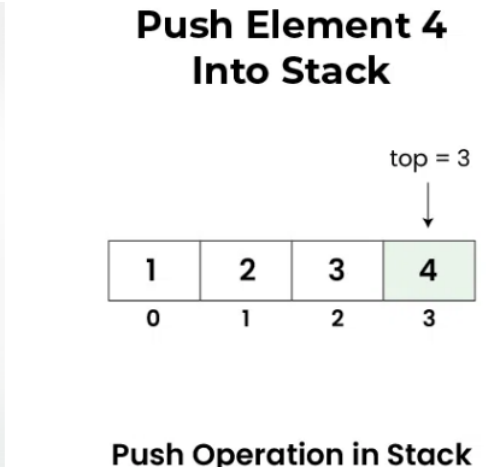
Step 3:



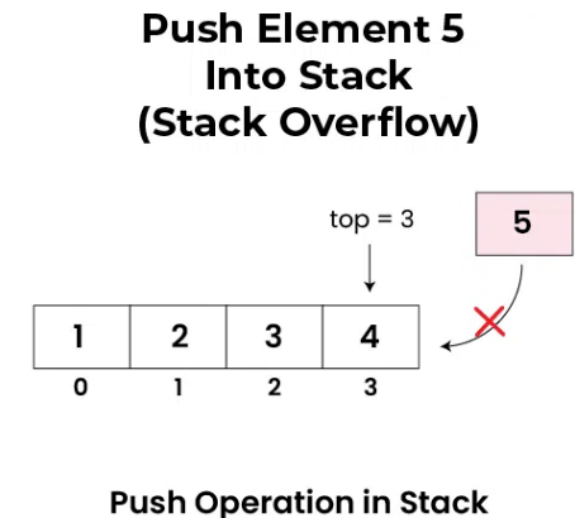
Step 4:



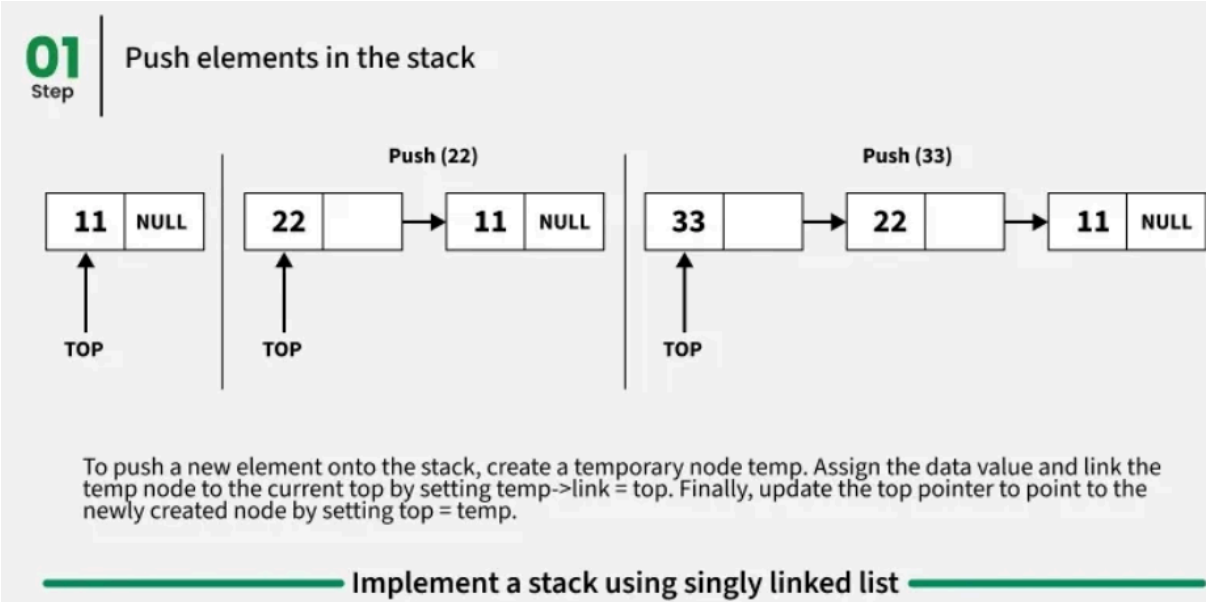
Step 5:

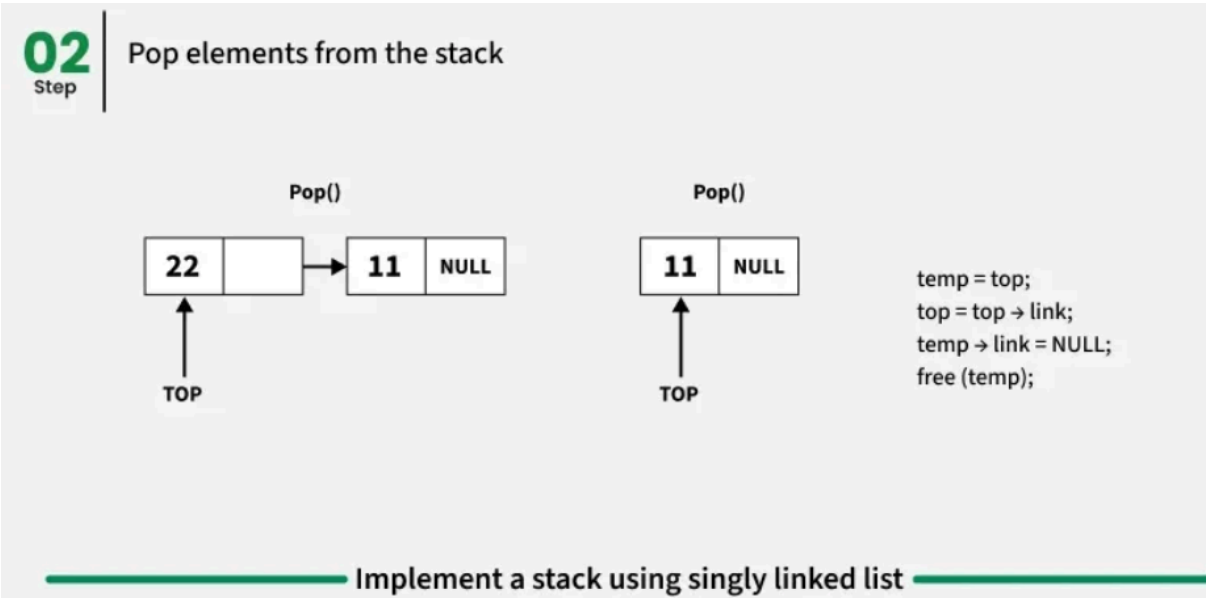


Step 6:



Using Linked list

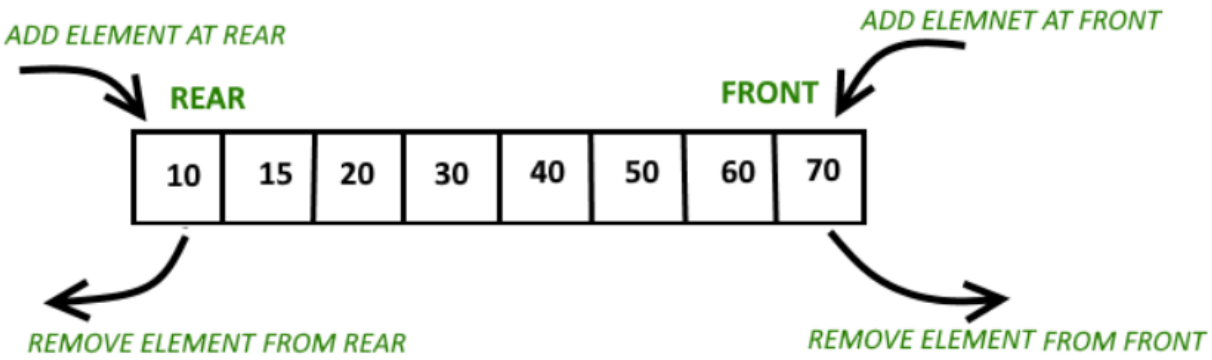




Here is the code in python for reference:
https://www.geeksforgeeks.org/stack-in-python/
Here is the code in C++ for reference:
https://www.geeksforgeeks.org/stack-in-cpp-stl/
Here is the code in javascript for reference:
https://www.geeksforgeeks.org/implementation-stack-javascript/

Using deque

A doubly ended queue or deque allows insertion and deletion at both ends. In a stack, we need to do insertions and deletions at one end only. We can use either end of deque (front or back) to implement a stack.



Declaration of stack

In C++

```
std::stack<int> myStack;
```

You can replace int with other types, like float, char, double, etc.

In python

```
stack = []
```

In javascript

```
let stack = [];
```

Basic operation on stack

1. In C++

Inserting element

```
myStack.push(10);
```

Accessing element

```
myStack.top();
```

Deleting element

```
myStack.pop();
```

Pseudo traversal

```
// Create a copy
stack<int> temp(myStack);

while(!temp.empty()) {
    cout << temp.top() << " ";
    temp.pop();
}
```

Operation	Time Complexity
Insert an element (push)	**O(1)**

Delete an element (pop)	**O(1)**
Access top element (peek)	**O(1)**
Traverse the stack	**O(n)**

To learn in more depth refer to:
https://www.geeksforgeeks.org/stack-in-cpp-stl/

2. In python

Inserting element

```
myStack.append(10)
```

Accessing element

```
myStack[-1]  # Peeks at the top element without removing it
```

Deleting element

```
myStack.pop()
```

Size of stack

```
len(self.myStack)
```

To learn in more depth refer to:
https://www.geeksforgeeks.org/stack-in-python/

3. In javascript

Push operation

```
myStack.push(10);
```

Pop operation

```
myStack.pop();
```

Peek

```
myStack[myStack.length - 1];  // Peek at top element
```

Size

```
myStack.length
```

To learn more in depth refer to:
https://www.geeksforgeeks.org/implementation-stack-javascript/

Resources

https://www.geeksforgeeks.org/stack-data-structure/
https://www.w3schools.com/dsa/dsa_data_stacks.php

Practice problems

Easy

1. Valid Parentheses - LeetCode | solution
2. https://leetcode.com/problems/next-greater-element-i/description/ | solution
3. Remove All Adjacent Duplicates In String - LeetCode | solution
4. Maximum Nesting Depth of the Parentheses - LeetCode | solution
5. Minimum String Length After Removing Substrings - LeetCode | solution
6. Implement Stack using Queues - LeetCode | solution
7. Binary Tree Inorder Traversal - LeetCode | solution

Medium

1. Reorder List - LeetCode | solution
2. Min Stack - LeetCode | solution
3. Remove Duplicate Letters - LeetCode | solution
4. Decode String - LeetCode | solution
5. 132 Pattern - LeetCode | solution

Hard

1. Largest Rectangle in Histogram - LeetCode | solution
2. Maximal Rectangle - LeetCode | solution
3. Trapping Rain Water - LeetCode | solution