Linear DS Continued!!

Suppose → MAX_SIZE = 100

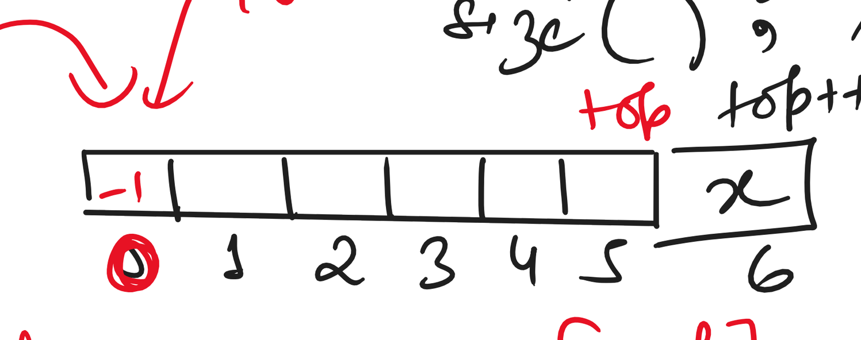
Stack → Last In First Out

(Representation using an Array) (OOP)

top: -1 … 99

Key points :→ top; push(); pop(); size(); isEmpty(); peek()

arr[top]



top = -1

top++ (Addition)

arr[top] = element;

top = Max - 1 (0 — 99)

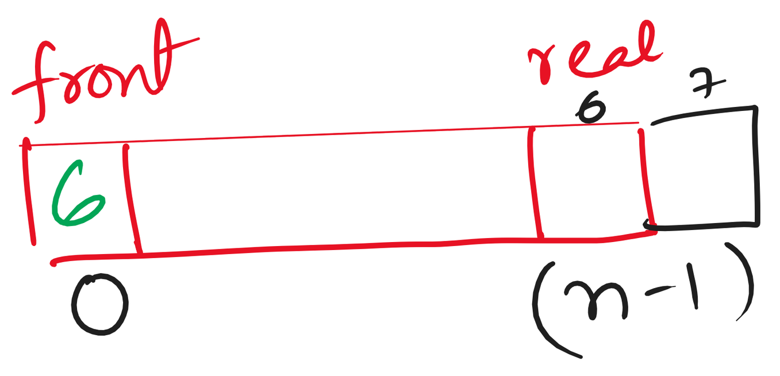
top--; → removal

Queue Data Structure :→

First In First Out (BFS | Level Order)

Using array :

S<=e
S>e → front, rear, enqueue, dequeue, size, isEmpty,



removal of elements ← front … rear → Addition of elements at the rear end

f++ rear++

f == r

Generate Binary 1 to N :→

(n=5) a[i] + '0' + '1'

Initial Queue : → ["1"]

Step 1: Pop "1" Print → Push "10", "11" → Q["10", "11"]

Step 2: Pop "10" Print → Push "100", "101" → Q["11", "100", "101"]

Step 3: Pop "11" Print → Push "110", "111" → Q['100', '101', '110', '111']

Step 4: Pop '100' Print → Push 1000, 1001

Next Greater Element :→ (Stack)

Input → [4, 5, 2, 25]
(indices 0 1 2 3)

i = 3 → 25 res[3] = -1

i = 2 → 2

25 <= 2 res[2] = 25

i = 1 → 5

2 <= 5

2 > 25

4 → 5
5 → 25
2 → 25
25 → -1

res [-1, 25, 25, -1]

I/p : [4, 5, 2, 25] [-1 -1 -1 (25) -1]

Processing the elements right to left using stack

i = 3 → 25 : Stack Empty → res[3] = -1 → Push 25

i = 2 → 2 : Stack top 25 > 2 → res[2] = 25 → Push 2

i = 1 → 5 : 2 < 5 → Pop 2, 25 > 5 res[1] = 25 → Push 5

i = 0 → 4 : 5 > 4 → res[0] → 5 push 4

Standard Template Library :→

Hash Table → Java

* stack → LIFO
* queue → FIFO
* list → forward_list → SLL; list → DLL
* set → ordered_set → Ascending Alphabetical Order; unordered_set → Hashing Order
* map → ordered_map → Alphabetical Order (Ascending); unordered_map → Hashing Order
* vector → dynamic array

b p a v g a
1 2 3 4 5