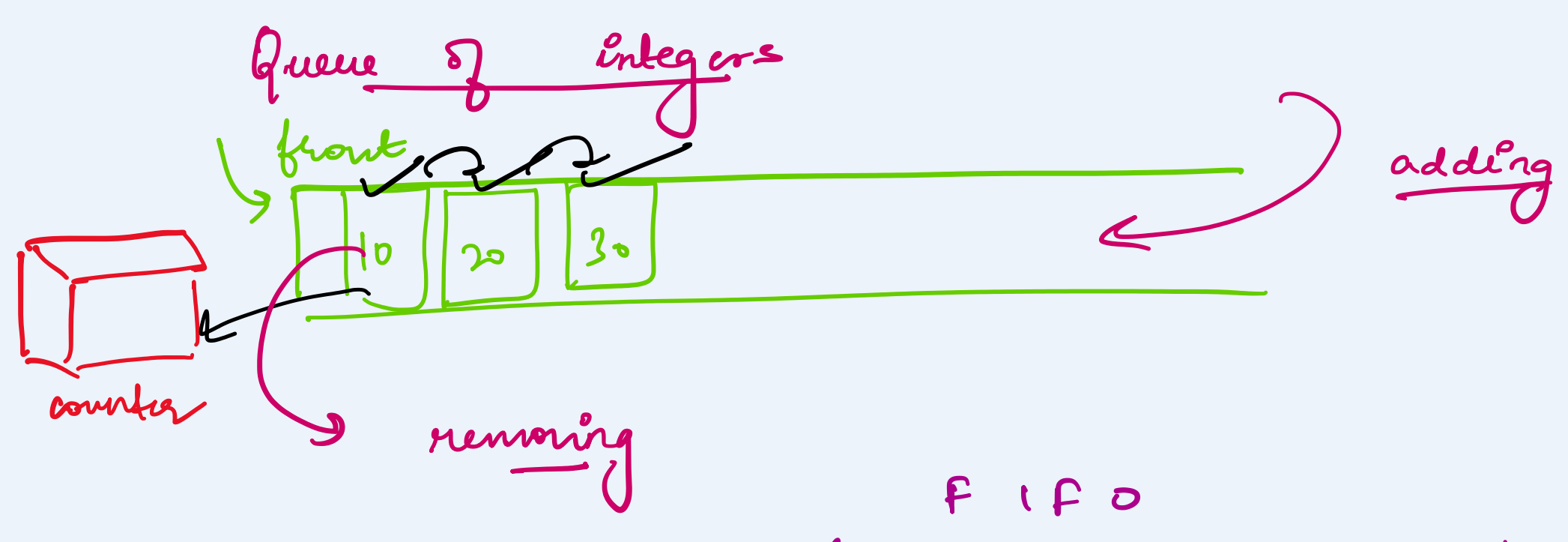
Queues

Queue of integers

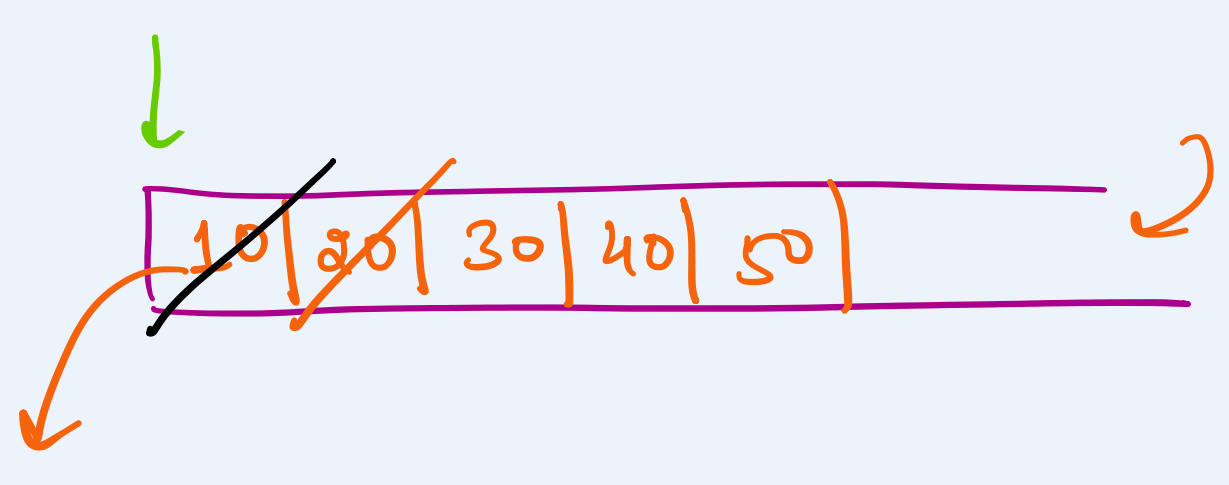


F I F O
(first in first out)

Operations :

1. Enqueue [adding to queue]
2. Dequeue [removing from queue]

Applications



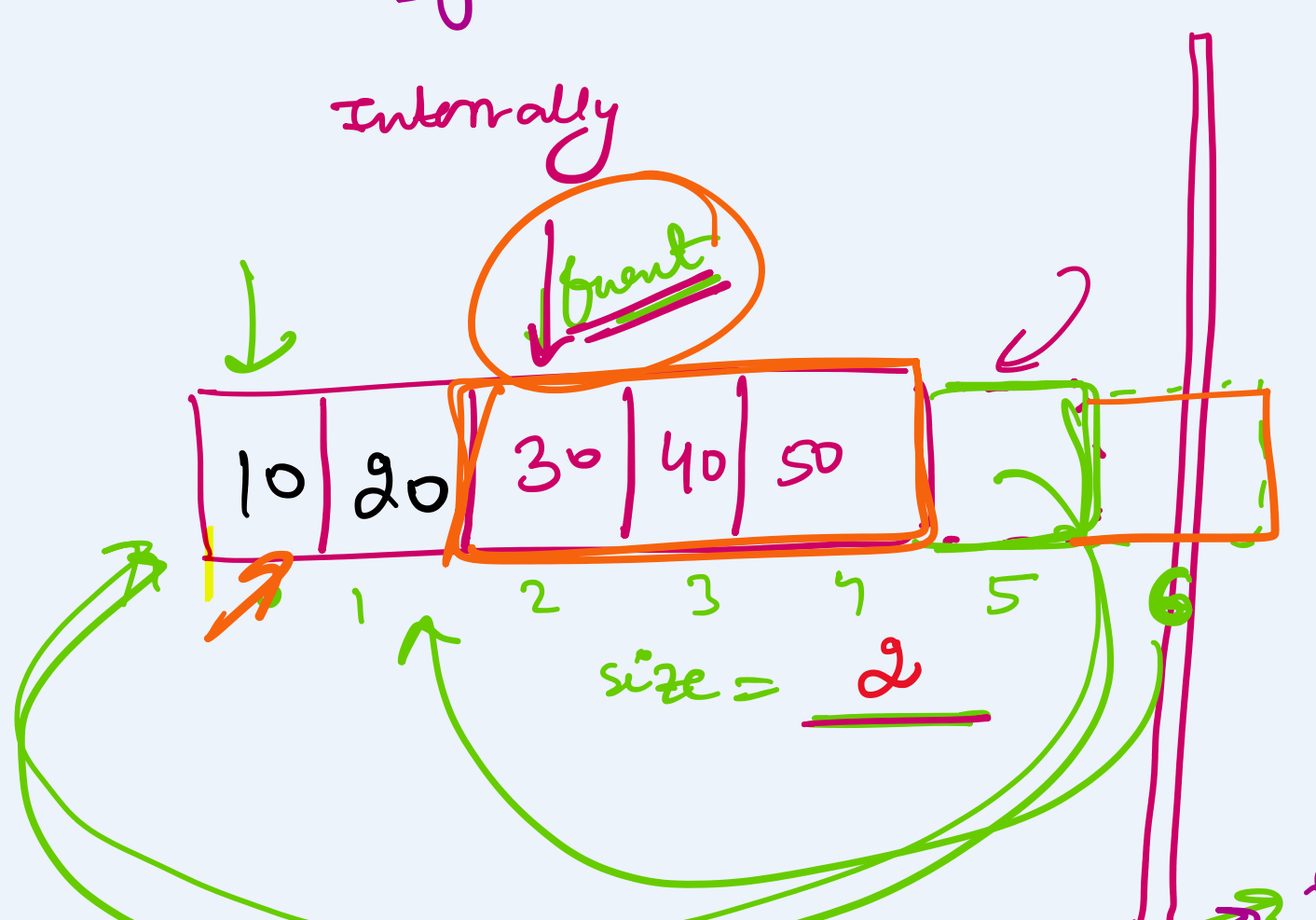
→ enqueue (10);
→ enqueue (20);
→ enqueue (30);
→ dequeue ();
→ enqueue (40);
→ enqueue (50);
dequeue ();

Implementations

1. Arrays / Circular Arrays
2. Linkedlist
3. Stack

Arrays

Internally



Users / programmers

q. enqueue (10);
q. enqueue (20);
q. enqueue (30);
q. dequeue ();
q. enqueue (); → error

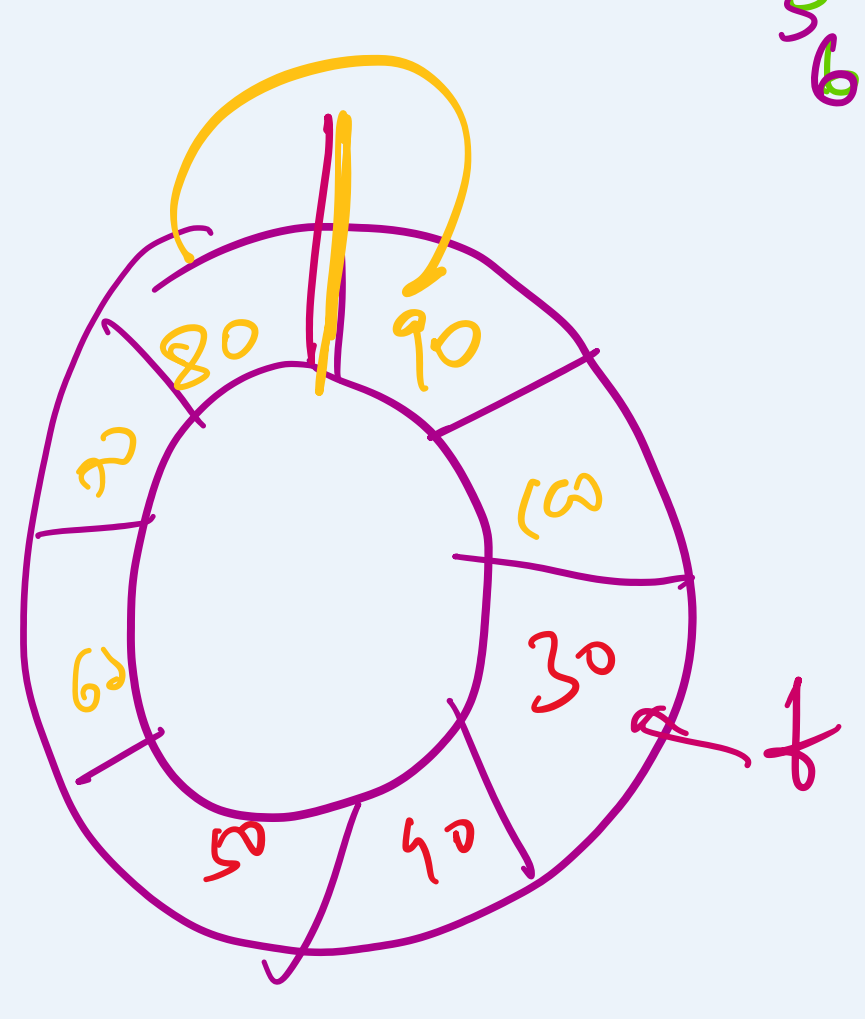
5 → 0
6 → 1

if (size == arr.length)

(front + size) % (size of arr)

5/5 → 0

(2 + 4)/5 → 6/5 = 1



Linkedlist

Enqueue → addAtHead ()
Dequeue → removeAtLast ()

h
40 → 30 → X

Users

q. enqueue (10);
q. enqueue (20);
q. enqueue (30);
q. dequeue ();
q. enqueue (40);
q. dequeue ()

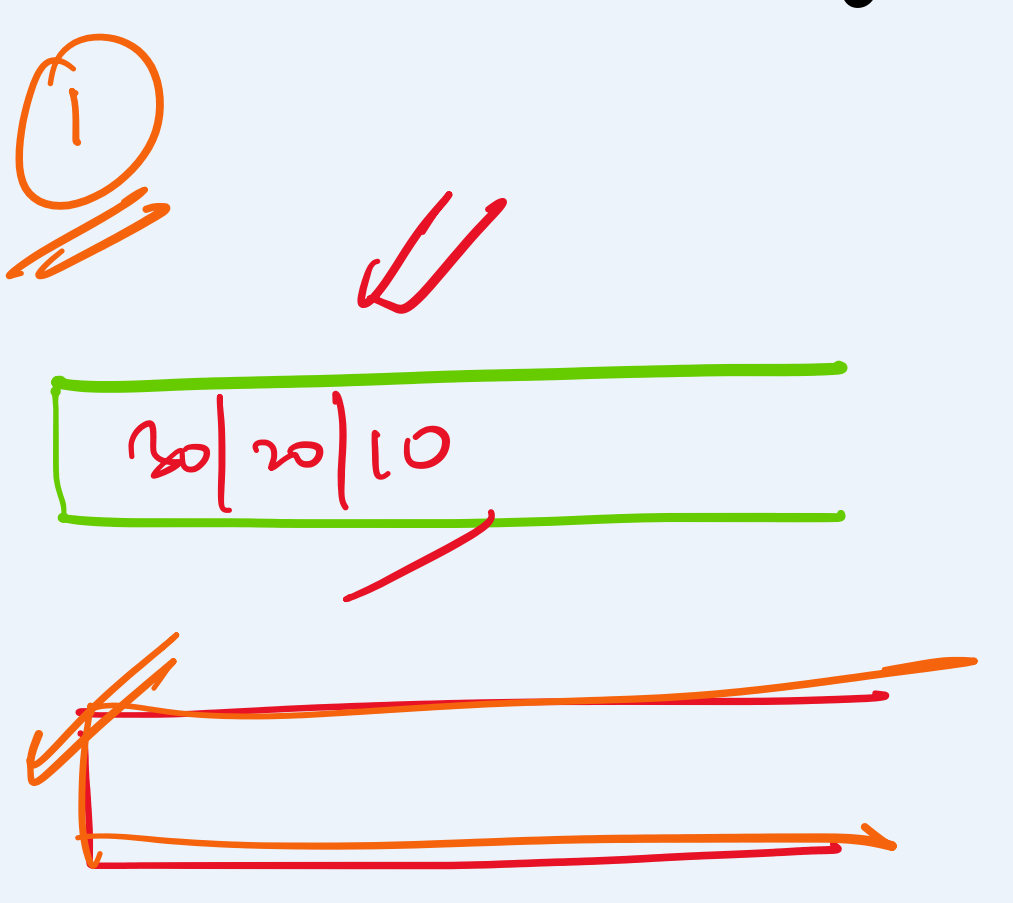
	Arrays	Linkedlist
enqueue	O(1)	O(1)
dequeue	O(1)	O(n)

Arrays
Pros
→ easy
→ operations optimised
→ Arrays takes less space
Cons
→ not flexible

Linkedlist
Pros
→ flexible / dynamic
Cons
→ added complexity of operations
→ extra memory

Queue using stack

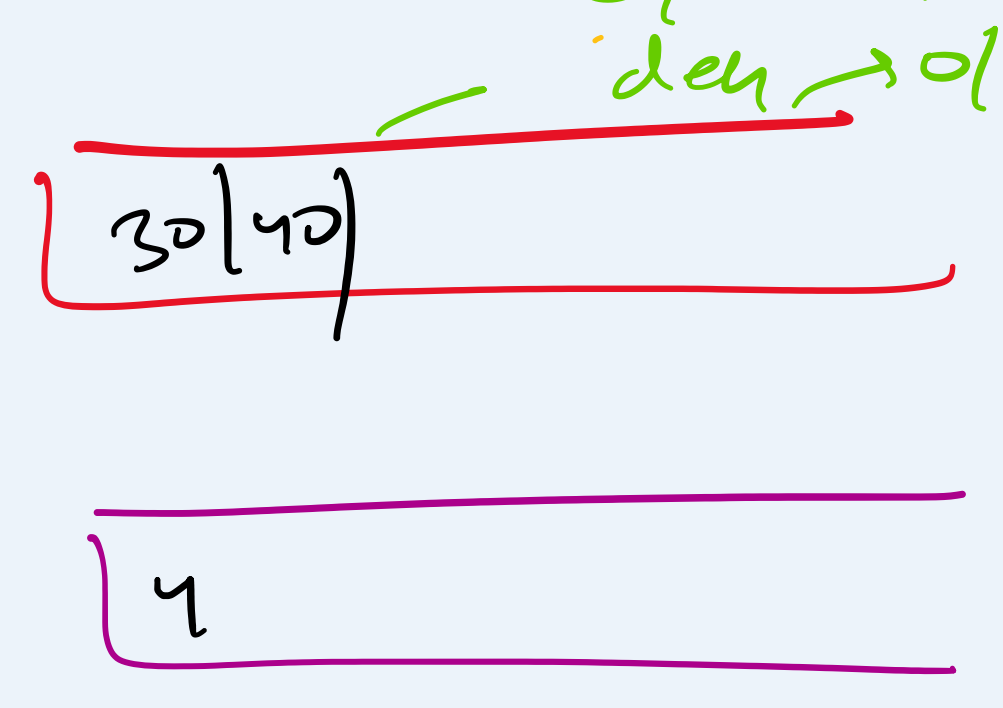
1



Users

q. enqueue (10); → O(n)
q. enqueue (20);
q. enqueue (30);
q. dequeue (); → s.pop(); → O(1)
q. enqueue (40);
q. enqueue (50);
q. dequeue();

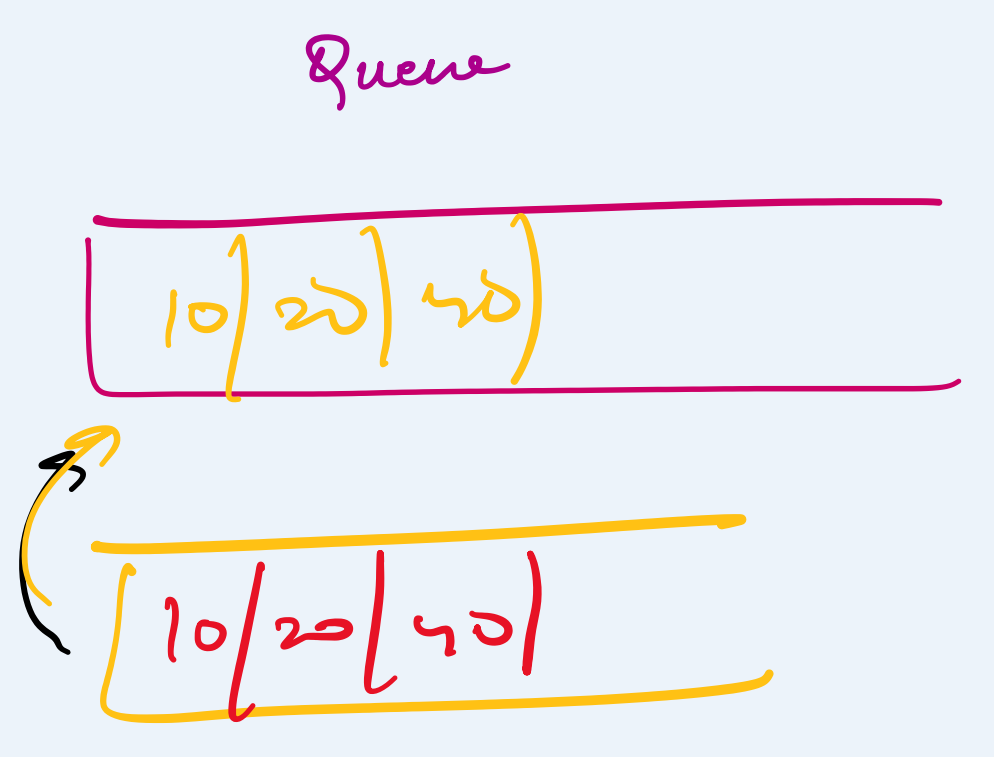
enq → O(1)
deq → O(n)



enq (10)
enq (20)
enq (30)
enq (40)
deq
deq

Stack using queue

Queue



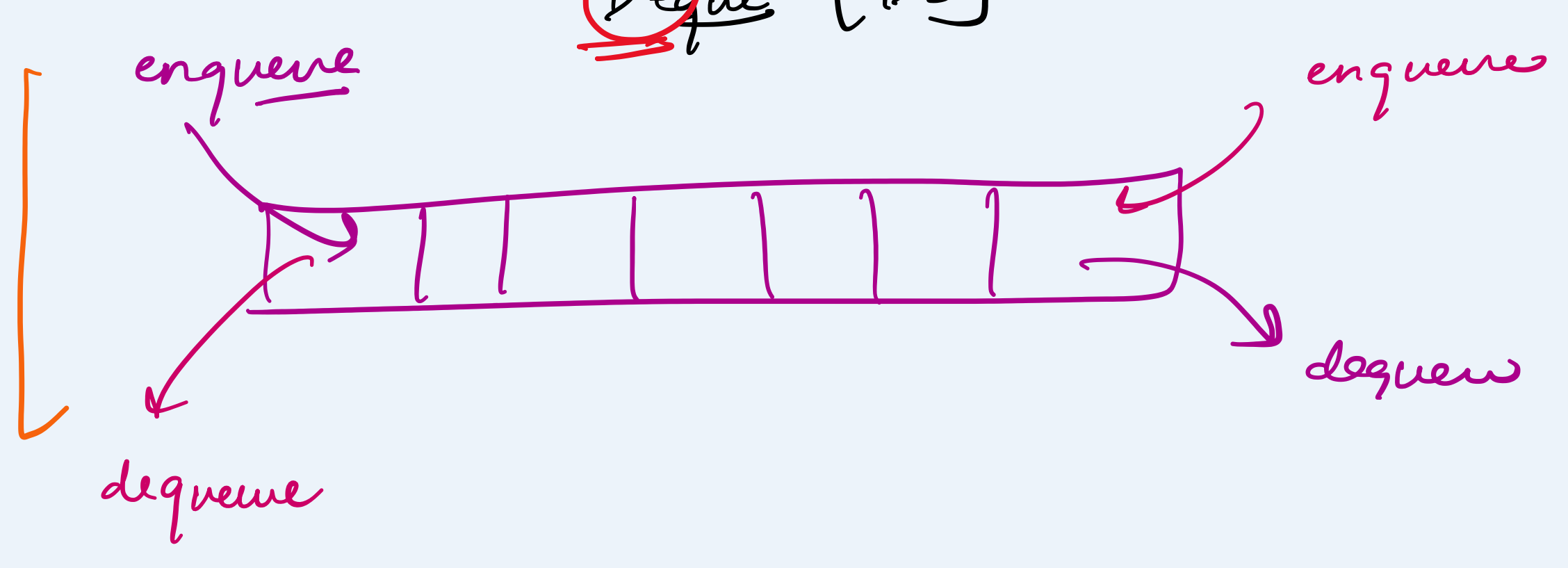
enque

Users

s. push(10);
s. push (20);
s. push (30);
s. pop ();
s. push(40);
s. push(50);
s. pop()

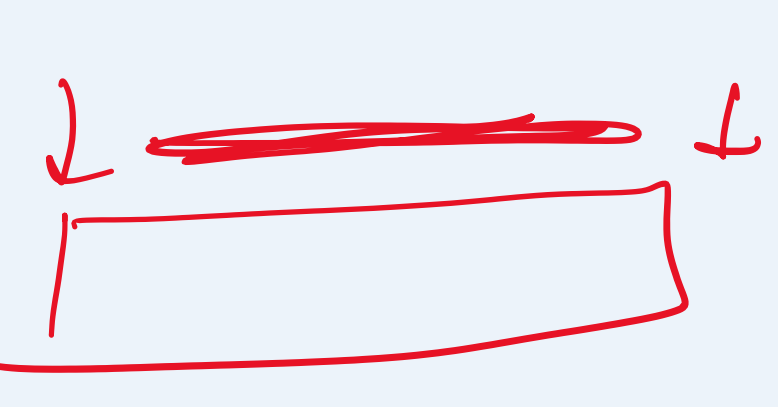
Deque [DS]

Deque → Double ended queue



Implementation

1. Arrays



2. doubly linkedlist