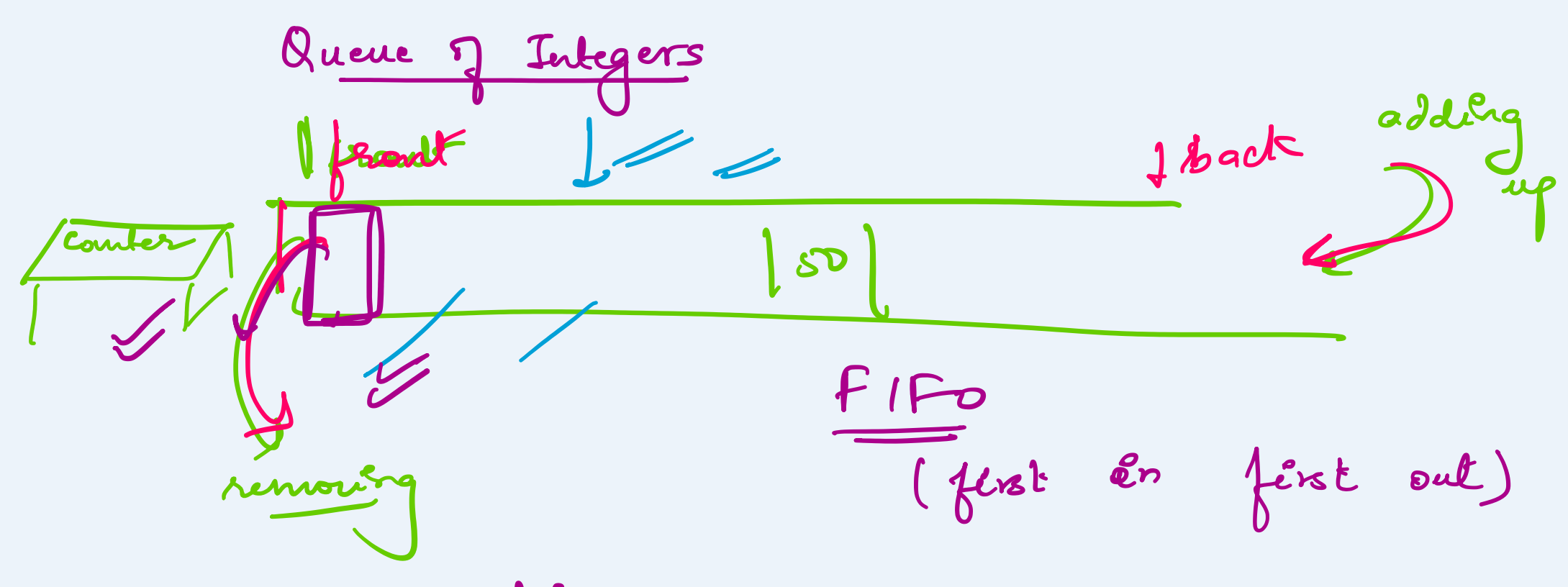
Wednesday, 31 March 2021 8:30 PM

Queues

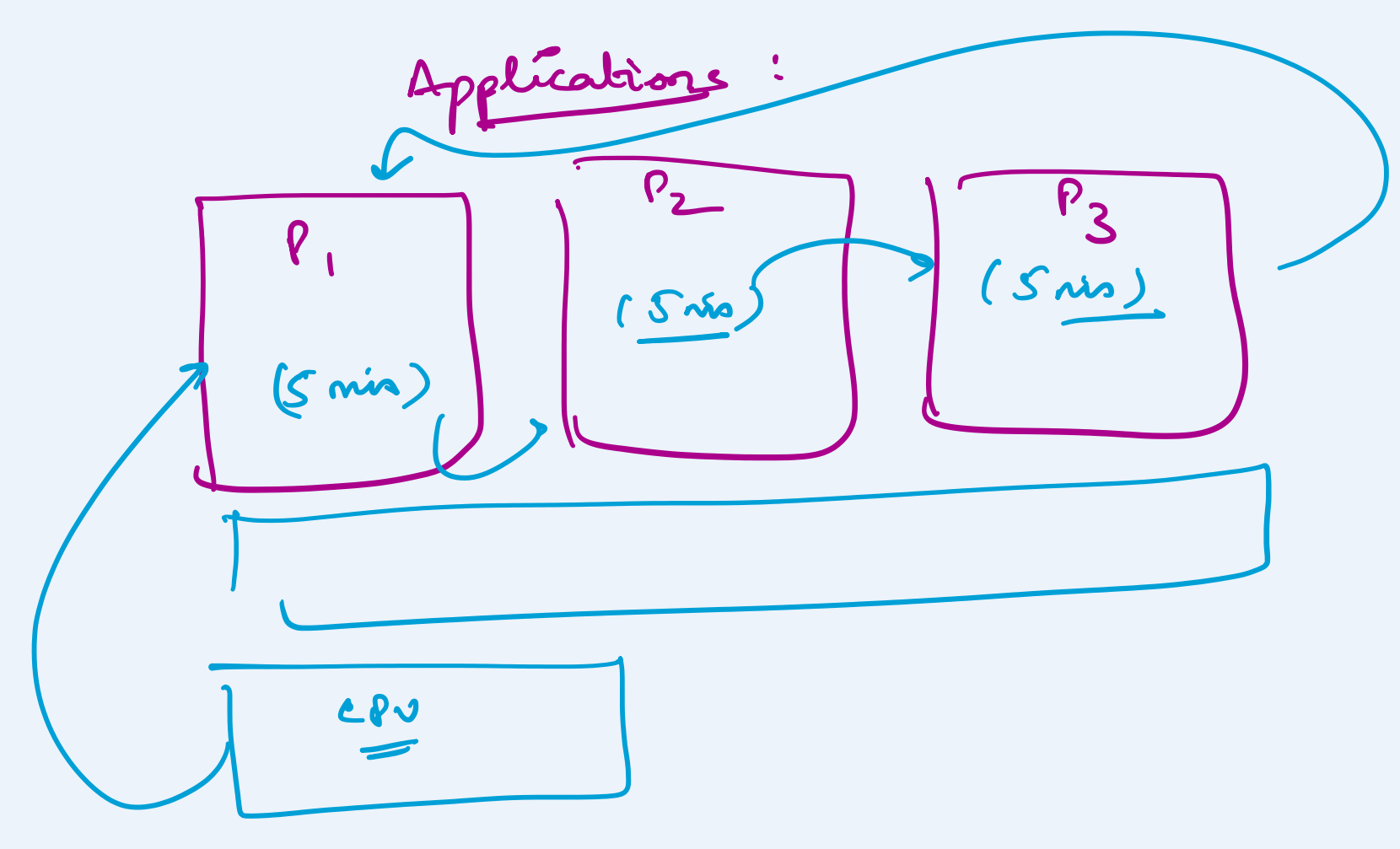
Queue of Integers

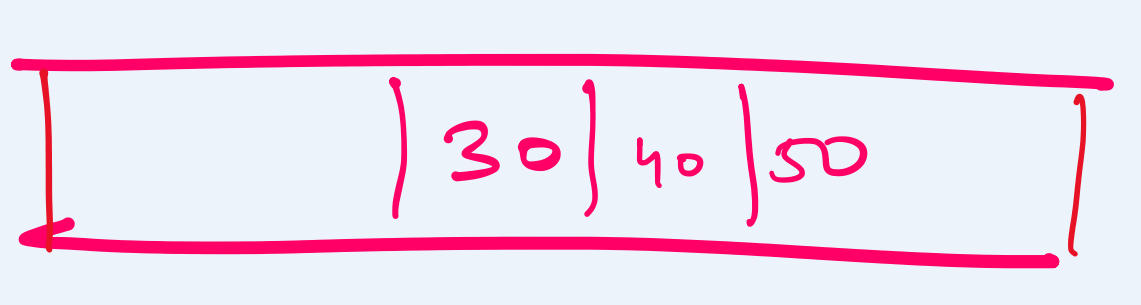


FIFO
(first in first out)

operations
1. enqueue [adding to queue]
2. dequeue [remove from queue]

Applications :





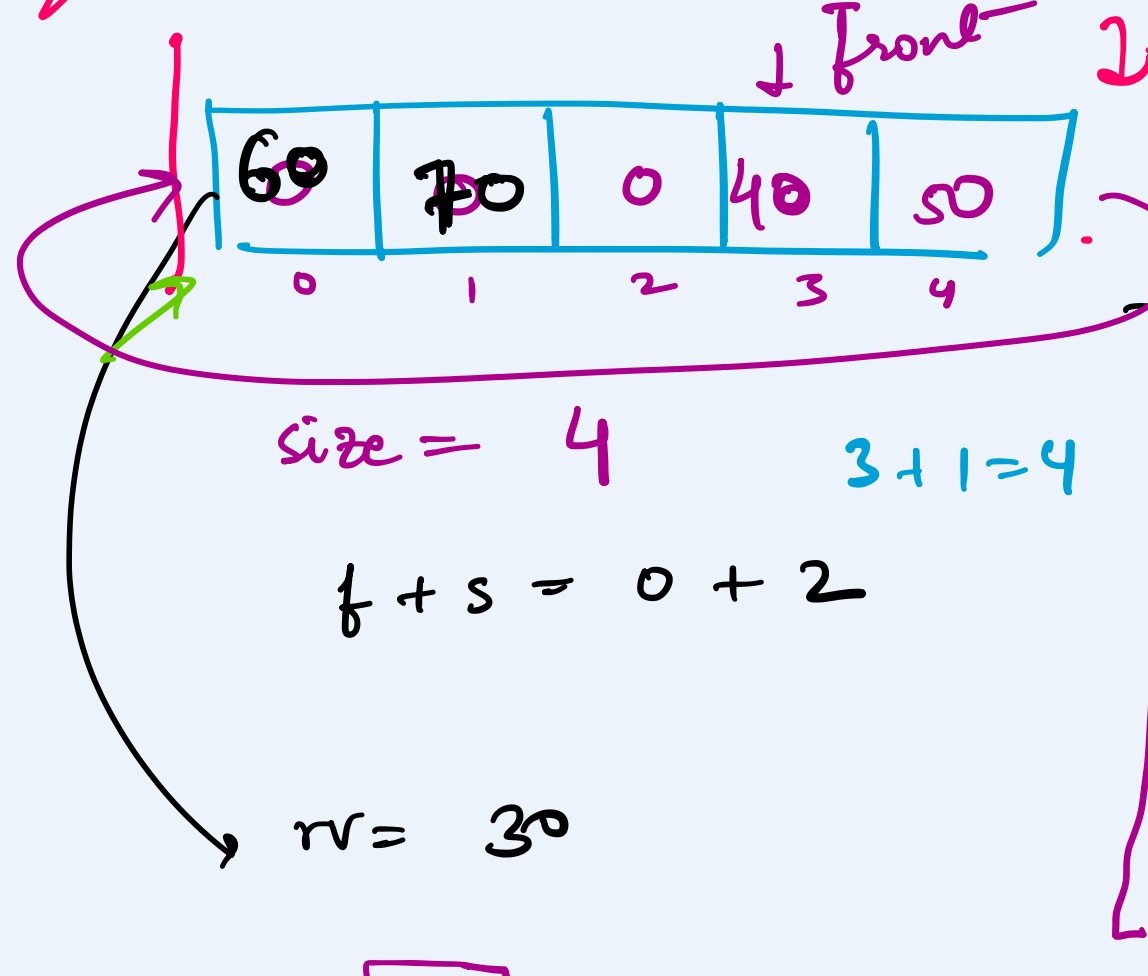
enqueue (10);
enqueue (20);
enqueue (30);
dequeue()
dequeue()
enqueue (40);
enqueue (50)

Queue Implementations :

1. Arrays / Circular arrays O(1) , O(1)
2. Linkedlist
3. stack

5 Internally
f + s
2 + 3

f + s
1 + 2 = 3



size = 4 3+1=4
f + s = 0 + 2

rv = 30

6

Users / programmers
q. enqueue (10);
q. enqueue (20);
q. enqueue (30);
q. dequeue();
q. enqueue (40);
q. enqueue (50);
→ q. dequeue ();
→ q. enqueue (60);
→ q. enqueue (70);
q. dequeue ();

5 → 0
6 → 1
7 → 2

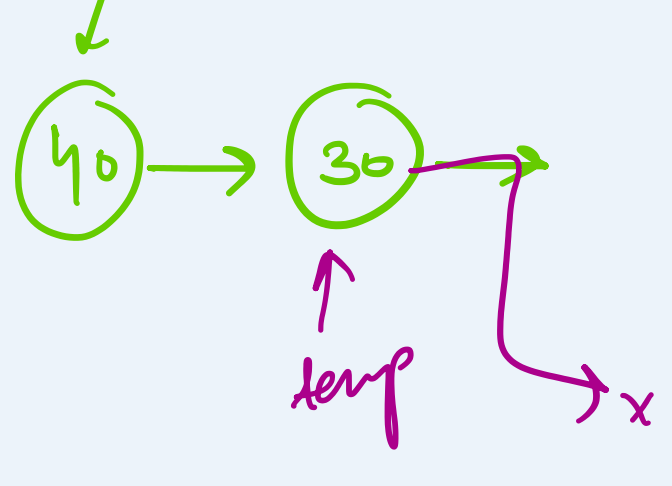
5 % size of array → 0
6 % 5 size → 1
7 % 5 → 2

if (size == arr. length)

Linked List

✓ Enqueue → add at head() [ ]
✓ dequeue → remove at last(); [ ] × tail

Internally



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

	Arrays	Linkedlist	Queues
enqueue	O(1)	O(1)	→ O(n)
dequeue	O(1)	O(n) × tail O(1) ✓ tail	→ O(1)

Pros:
- → easy
- → operations optimised
- → It takes less space

Cons
- → not flexible

Linked list

Pro :
- → flexible / dynamic

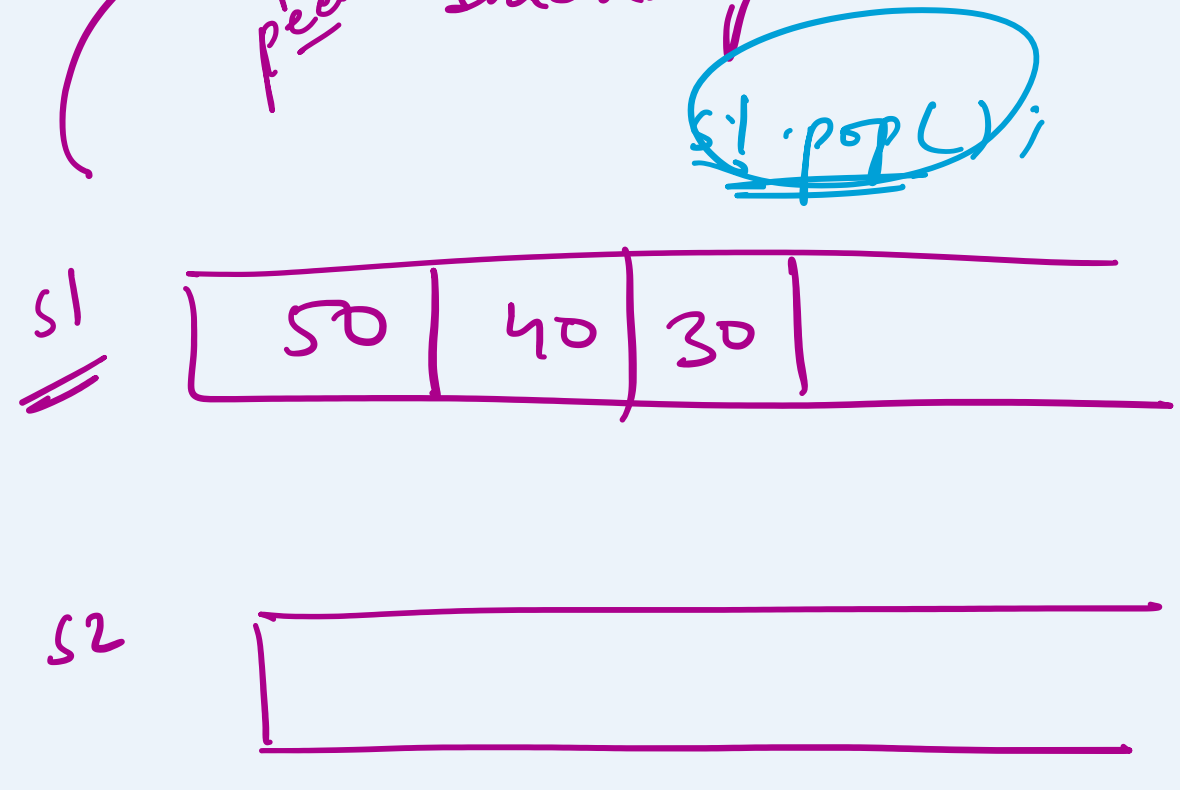
Cons :
- → added complexity of operations
- → extra memory

Queue using stack

isEmpty
pop
push
peek

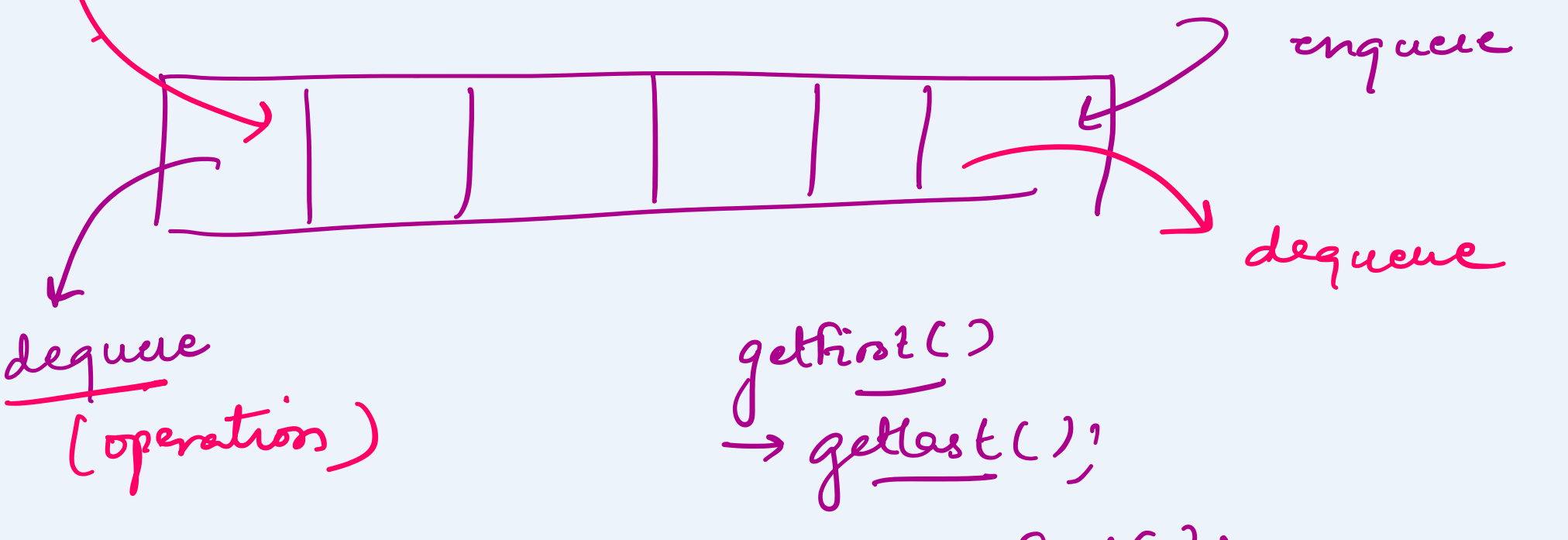
Internally

s1.pop();



User
→ q. enqueue (10);
→ q. enqueue (20);
→ q. dequeue ();
→ q. enqueue (30);
→ q. enqueue (40);
→ q. enqueue (50);
→ q. dequeue ();

Queue
↓
Deque [DS]



getFirst()
→ getLast();
→ remove First();
→ remove Last();
→ addFirst ();
→ add Last ();

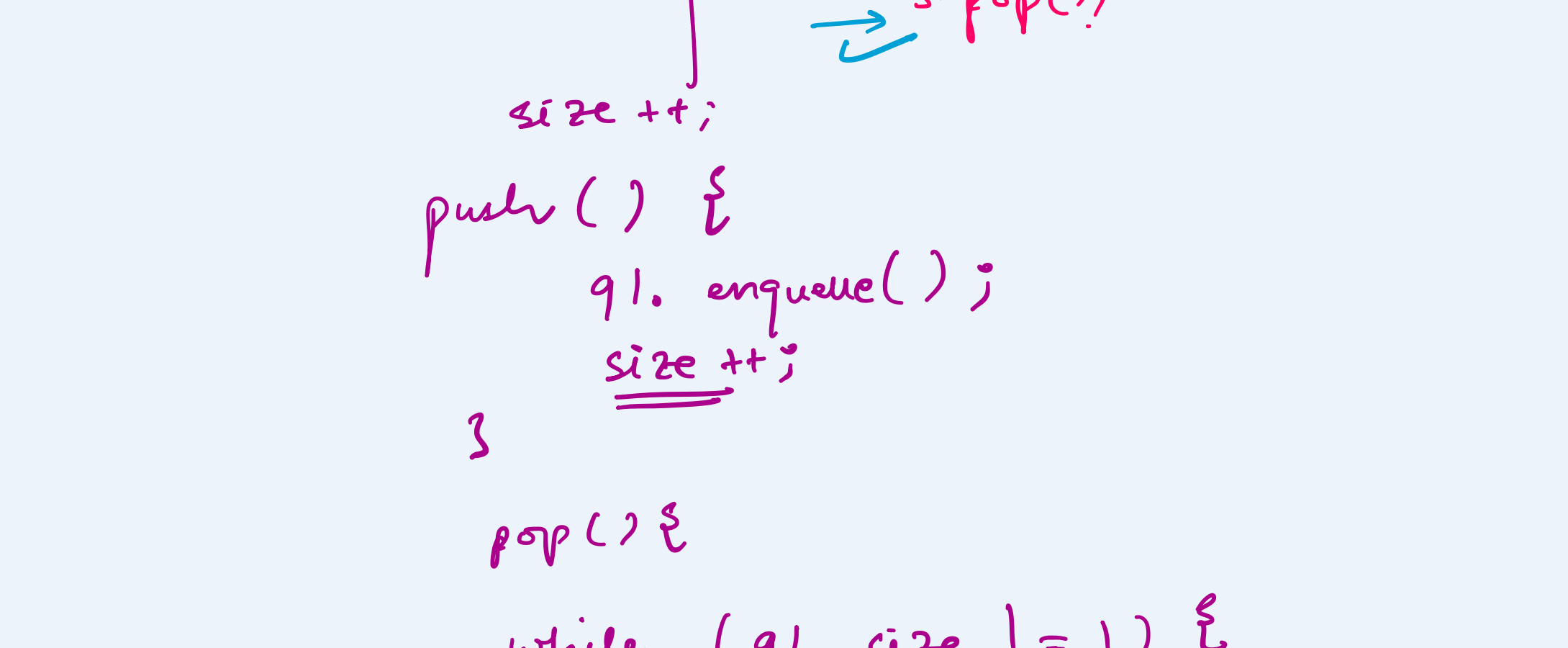
Implementations

1. arrays
2. doubly linked list

Stacks
- → arrays
- → Linkedlists
- → Queue

1. dequeue()
2. enqueue()

Q stack using queue



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

size ++;

```
push() {
    q1. enqueue();
    size ++;
}
pop() {
    while (q1. size != 1) {
    q2. enque(q1. dequeue());

}
    int rv = q1. dequeue();
    q1 = q2
    q2 = new Queue();
}
```