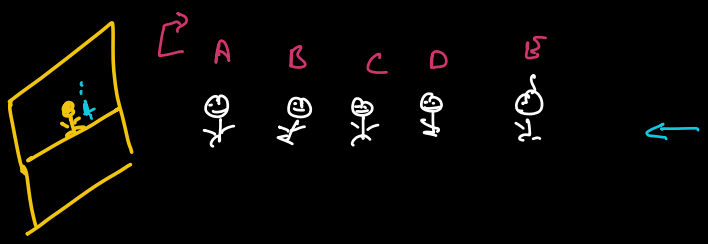


enque(x) ⟶ Insert x in the queue

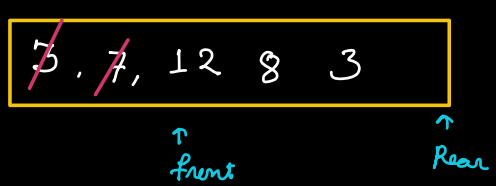
dequeue() ⟶ Remove one element from the queue.
/poll()

front() ⟶ Return the front of queue

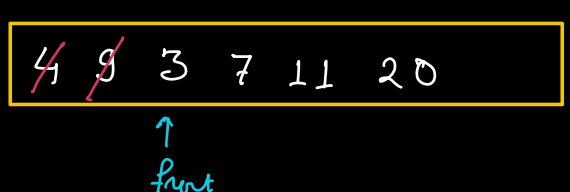
isEmpty() ⟶ Return true if queue is empty.

Implementation

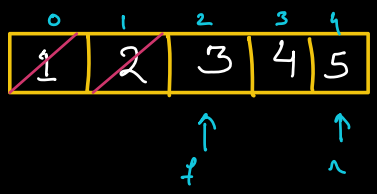
eq(3) eq(7) eq(12) dq() dq() eq(8) eq(3)



eq(4), dq(), eq(9), eq(3), eq(7), eq(11), eq(20), dq()



Using Arrays



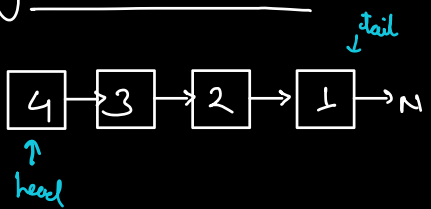
6

(r+1)

(r+1) % N

HW : Implement a Circular Queue using an array

Using Linked → lists



eq(1) eq(2) eq(3) eq(4) → O(1)

dq() → O(N)

Add at tail ⇒ O(1)

Remove from head ⇒ O(1)

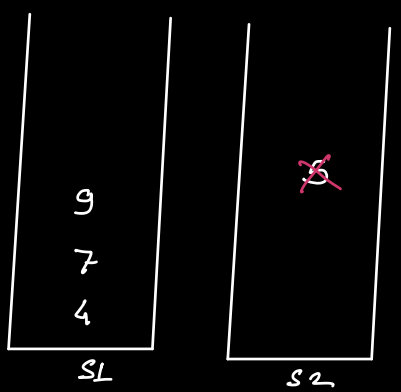
Q Implement a queue using stacks (only)

enqueue(x)

dequeue()

→ Push(x)
→ Pop()
→ top()
→ isEmpty()
→ size()

eq(5) eq(4) eq(7) eq(9) dq()

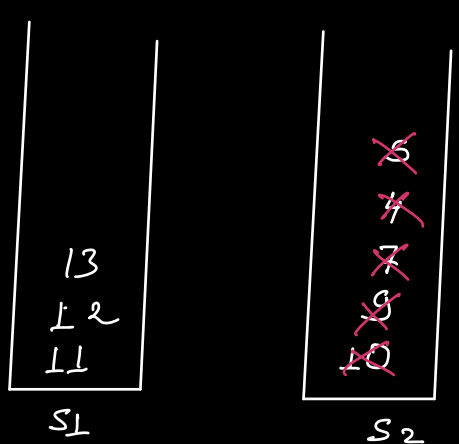


eq(x) ⇒ O(1)
dq() ⇒ O(N)

5, 4, 7, 9

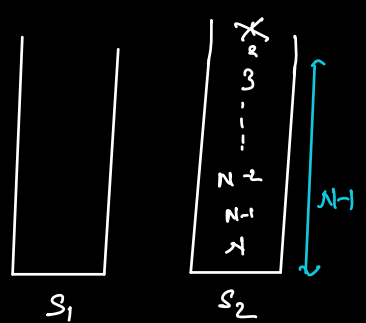
5, 4, 7, 9, 10, dq(), 11, 12, 13 dq() dq() dq()

5, 4, 7, 9, 10, 11, 12



```
void enqueue(x) {
    S1.push(x);
}
```

```
void dequeue() {
    if (S2.isEmpty()) {
        while (!S1.isEmpty()) {
            S2.push(S1.top());
            S1.pop();
        }
    }
    if (!S2.isEmpty()) {
        S2.pop();
    }
}
```



for 1st dq() ⟶ N iterations
for rest (N-1) dq() ⟶ 1 iteration × (N-1)

Total iteration ⟶ N + (N-1)

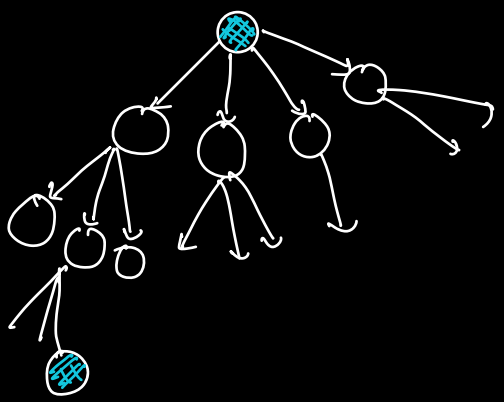
⇒ 2N-1

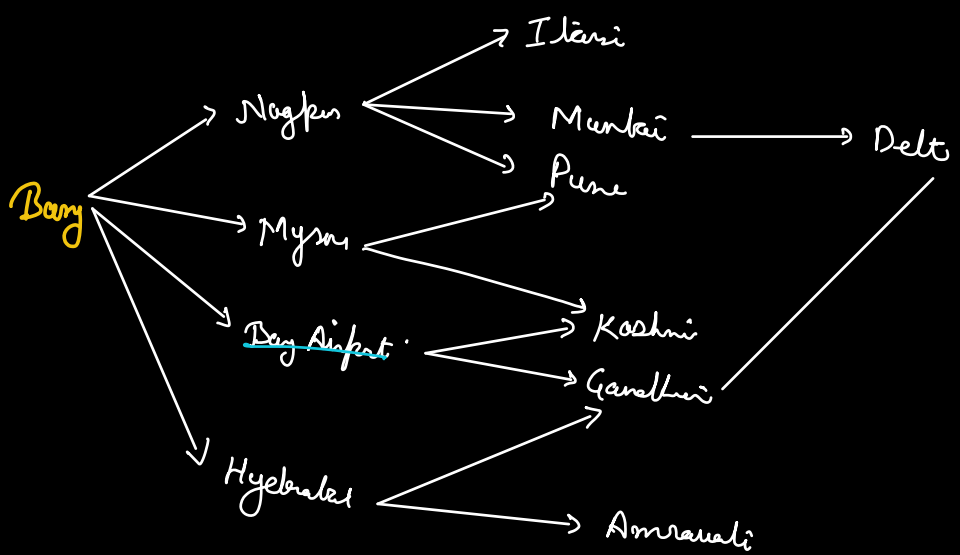
N dq() ⟶ 2N-1 iterations ≈ O(N)

1 dq() ⟶ O(1)

Amortized

GOI-





Bang 1 4
2 7
3 8

Itāri Mumbai Pune Kochi Gandhi
Amravati

Bang Nag Mysu Hyd

Q N^{th} number using only 1, 2, 03 as digits.

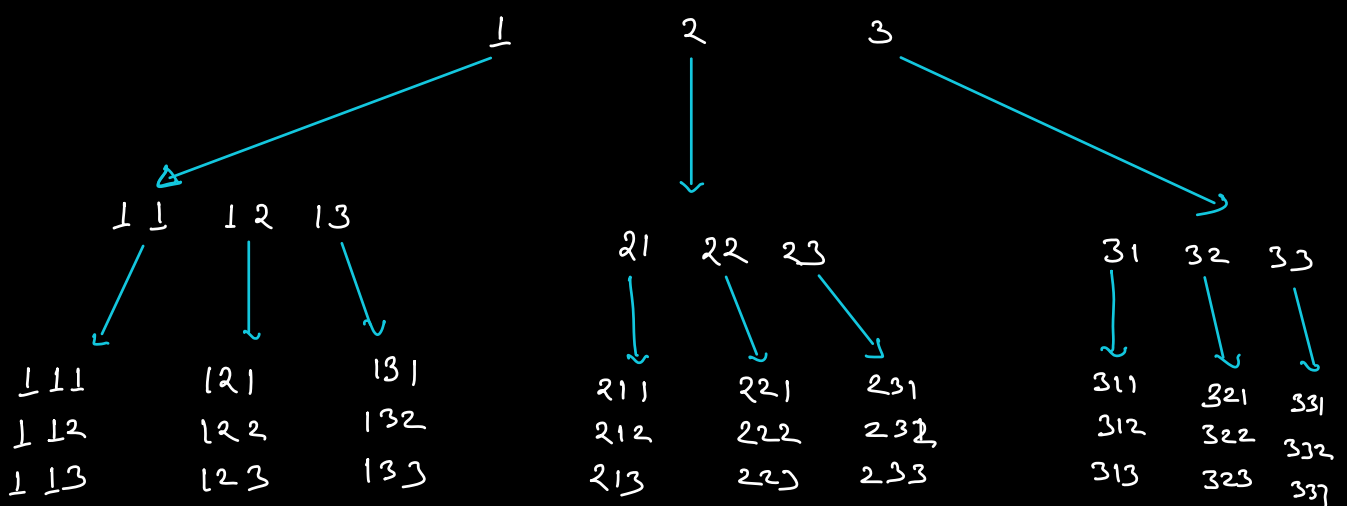
↳ (digits apart from 1, 2, 3 must not be 0th)

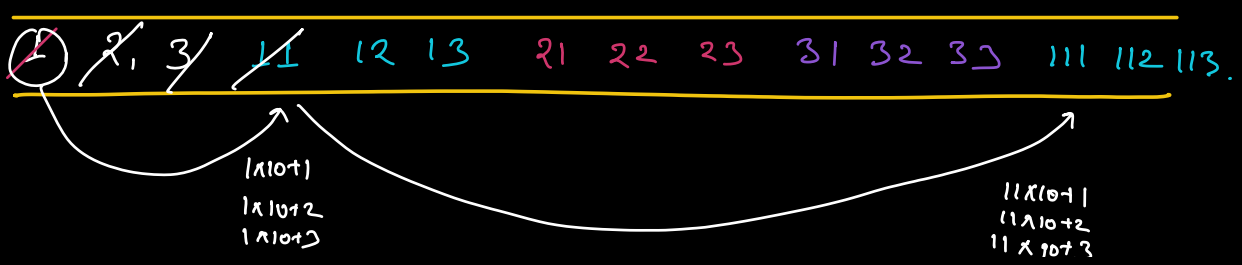
1 → 1st
2 → 2nd
3 → 3rd
11 → 4th
12 → 5th
13 → 6th
21 → 7th
22 → 8th
23 → 9th
31
32
33
⋮

N = 3 → 3

N = 8 → 22

N = 15 ↓





1 , 2, 3

11 12 13 21 22 23

Q Find Nth perfect no.

- Can only have 1 or 2 or both as digits
- Even length
- Palindrome

11
22
1111
1221
2112
2222
11111111
11112

137731

1, 2

11, 12, 21, 22

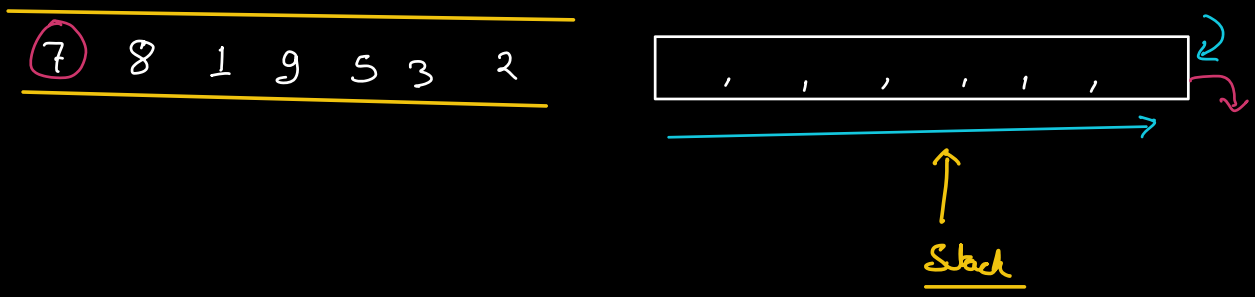
111, 112, 121, 122

Q Given a queue. Reverse it.

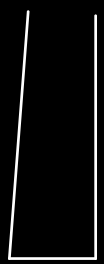
2, 3, 5, 9, 1, 8, 7

front

7, 8, 1, 9, 5, 3, 2



TC : O(N)
SC : O(N)



```
void reverse (Q q) {
    Stack st;
    while (q.size() > 0) {
        st.push (q.poll());
    }
    while (st.size() > 0) {
        q.eq (st.pop);
    }
}
```