1) Queues Basics.
2) Implementation.
3) Problems.

Queues :-

- Messaging Queues.
- CPU scheduling.
- Printers.

* FIFO :- First In, First Out.

7, 6, 12, 14,

~~7~~, 6, 12, 14,

f r

Queue fun :-

1) enqueue(x) : insert (x) into the queue.
2) dequeue() : Remove front element from the queue.
3) front() : returns the front element.
4) rear() : returns the rear element.
5) size() :
6) isEmpty() :

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

← ~~3~~, ~~7~~, 12, 8, 3 ←

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

~~4~~, ~~9~~, 3, 7, 11, 20

Implementation :-

↳ Arrays

→ Linked List.

* 1, 2, 3, 4, dq(), dq(), 5, 6, 7, dq(), dq(), 8

F ← R

0	1	2	3	4
~~1~~	~~2~~	3	4	5

f r

0	1	2	3	4
6	7	8	~~4~~	~~5~~

f r

$(r+1) \% N$

$5 \% 5 = 0$

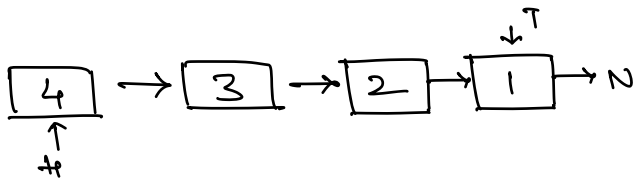
HW

* Implement Circular Queue

* Using Linked List

* Insert at head, delete at Tail

$O(1)$ $O(N)$



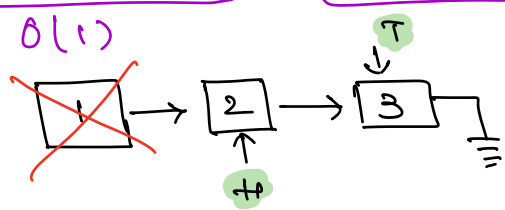
1, 2, 3, 4

enqueue() ⇒ $O(1)$

dequeue() ⇒ $O(N)$

* Insert at tail, delete at head.

$O(1)$ $O(1)$



1, 2, 3

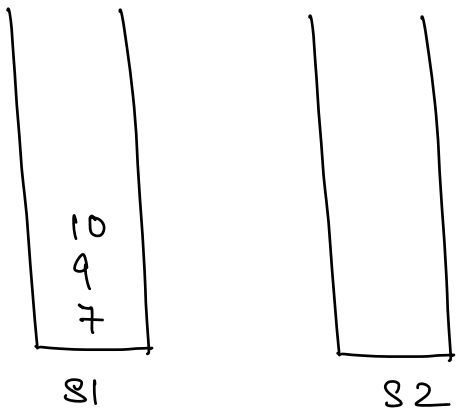
enqueue() ⇒ $O(1)$

dequeue() ⇒ $O(1)$

Q: Implement Queue using stacks

↳ push(x)
→ pop()
→ top()
→ size()
→ isEmpty().

eq(5), eq(4), eq(7), eq(9), dq(), eq(10), dq()



~~5~~ 4 7 9

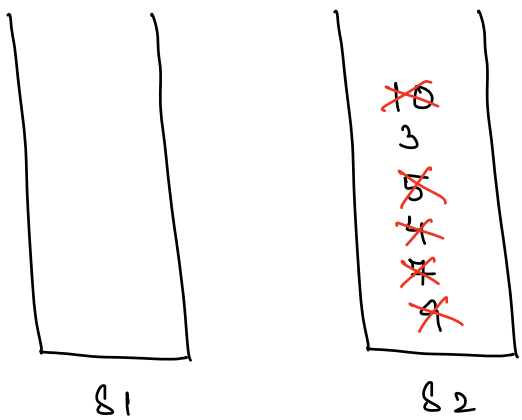
TC:

enqueue() : O(1)

dequeue() : O(N)

Approach 2 :-

eq(5), eq(4), eq(7), eq(9), dq(), eq(10), eq(3),
dq(), dq(), dq(), dq()



~~5~~ ~~4~~ ~~7~~ ~~9~~ ~~10~~ 3

Queue

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

⇒ TC: O(1)

```
void dequeue() {
    if ( S2.isEmpty() ) {
        while ( S1.size() > 0 ) {
            S2.push(S1.top())
            S1.pop()
        }
    }
    if ( !S2.isEmpty() ) {
        S2.pop();
    }
}
```

TC analysis :- N

1st dequeue() ⇒ N iterations.

for next (N-1) dequeue operations ⇒ 1 × (N-1) iterations

N dequeue() operations = 2N-1

≈ O(N)

Iterations for 1 operations = O(1)

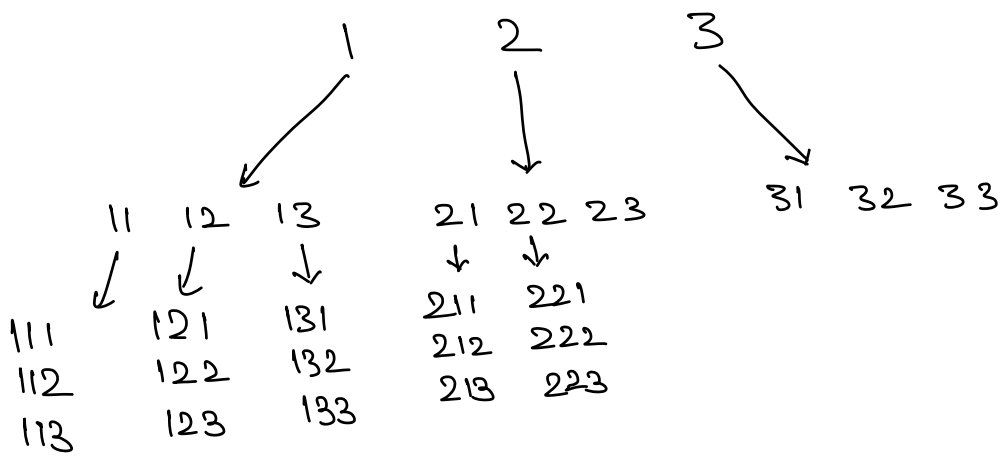
[Amortized TC]

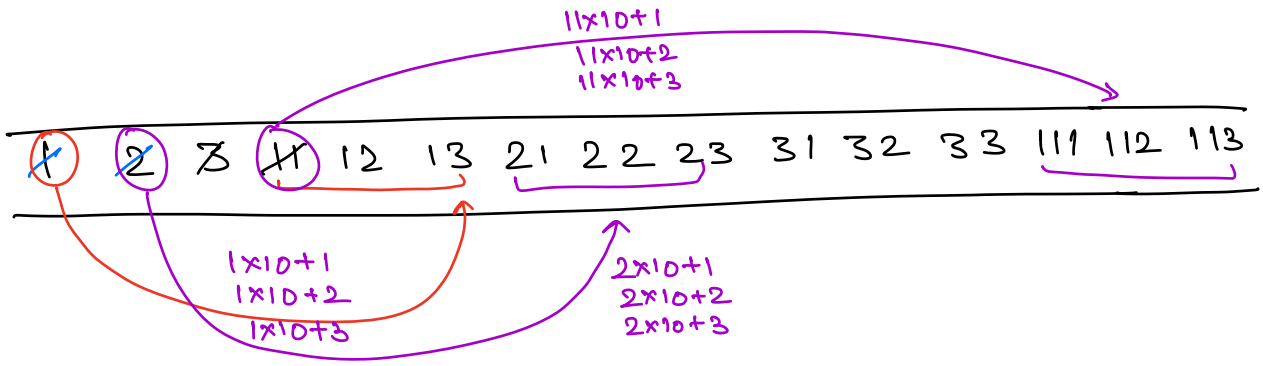
Q: N^{th} number using only digits 1, 2 or 3.

1
2
3
11
12
13
21
22
23
31
32
33
111
112
113
⋮

N = 5 ⇒ 12

N = 8 ⇒ 22





Q: Find N^{th} perfect number.

↳ Can only have 1 or 2 or both as digits.

→ Even length.

→ Palindrome.

11

22

1111

1221

2112

2222

111111

11·2211

⋮

No's using only digits 1, 2.

Q: Given a Queue, reverse it.

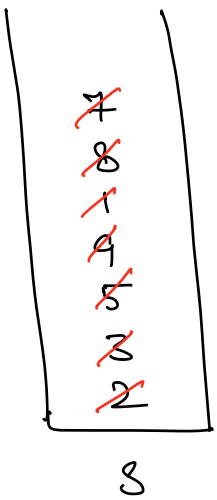
← 2, 3, 5, 9, 1, 8, 7 ←

↓

← 7, 8, 1, 9, 5, 3, 2 ←

Queue

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



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

```
void reverse (Queue q) {
    Stack<int> s;
    while ( q.size() > 0) {
        s.push(q.front())
        q.dequeue()
    }
    while ( s.size() > 0) {
        q.enqueue ( s.top());
        s.pop();
    }
}
```

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

* Explore how Queue is implemented in your language
  - ⇒ C++ STL
  - ⇒ Java Collections.

Doubts

← 7 2 3 5 9 1 8