# Queue 1

| |
|---|
| Madhan Kumar M S |
| Abhishek Sharma |
| Akansh Nirmal |
| amit khandelwal |
| Balaji S K |
| Bhaveshkumar |
| Burhan |
| Gagan Kumar S |
| Murali krishna Talluri |
| Naval Oli |
| Nikhil Pandey |
| Pankaj Bhanu |
| Prajwal Khobragade |
| PREETHAM |
| Purusharth A |
| Rajat Sharma |
| Rajendra |
| Sanket Agarwal |
| Sanket Giri |
| Saurabh Ruikar |
| Shani Jaiswal |
| sharath r |
| Subhashini |
| Subhranil Kundu |
| Sumit Adwani |
| Suyash Gupta |
| Vasanth |
| Vimal Kumar |
| Vishal Mosa |
| Yugesh v |

AGENDA:

- Introduction
- Implementation
- Implement Queue via stack
- Perfect Numbers
- Sliding window Maximum.

Current PSP

63 % → 70%.



SBI

First In First Out

FIFO

## Operations on Queue

TC of operations = O(1)

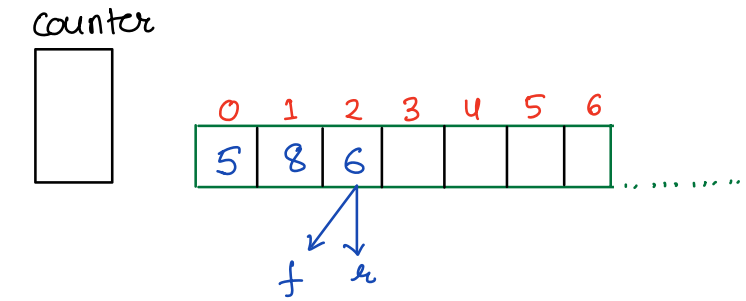
- enqueue() ⟶ add at rear end
- dequeue() ⟶ remove from front
- isEmpty() ⟶ check if queue is empty

## Optional functions

- front() ⟶ element at front
- rear() ⟶ element at rear.

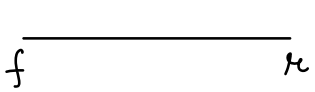
# Implement Queue using Dynamic Arrays

enqueue ( 5 ) ✓
enqueue ( 8 ) ✓
enqueue ( 6 ) ✓
dequeue ( )
dequeue ( )
isEmpty ( )




f = 0
r = ~~-1~~ 0

[f ..... r]



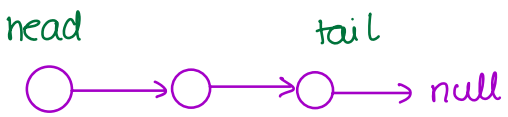

```
void enqueue ( int val ) {  // Overflow
    r++
    A[r] = val
}
```

Dynamic array.

```
bool isEmpty () {
    return f > r
}
```

```
int dequeue ( ) {
    // check underflow
    if ( isEmpty() ) return -1
    rval = A[f]
    f++
    return rval
}
```

# Implement Queue via LL




- enqueue() ⟶ add new elements at tail
- dequeue() ⟶ remove from head
- isEmpty() ⟶ head == null

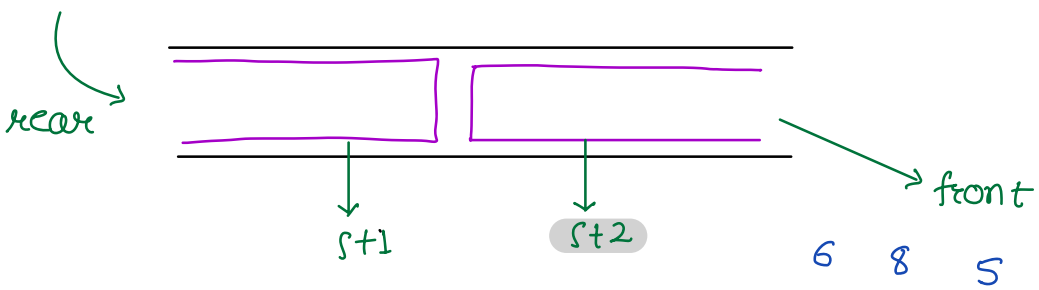
| | head | tail |
|---|---|---|
| add | O(1) | O(1) |
| remove | O(1) | O(n) |

# Implement Queue using Two stacks ****.

enqueue
dequeue
isEmpty

push
pop
isEmpty
peek

enqueue (5) ✓
enqueue (8) ✓
enqueue (6) ✓
dequeue () ✓
dequeue () ✓
isEmpty ()




```
void enqueue (int val) {
    st1.push (val)
}
```

```
int dequeue () {
    if (isEmpty ()) {
        return -1
    }
    if (st2.isEmpty ()) {
        while (!st1.isEmpty ()) {
            st2.push (st1.pop ())
        }
    }
    st2.pop()
}
```

```
bool isEmpty () {
    return st1.isEmpty()
    && st2.isEmpty()
}
```

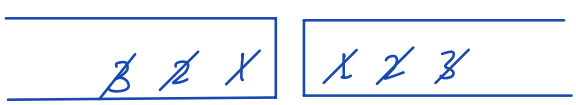
| | iterations |
|---|---|
| E(1) ✓ | 1 |
| E(2) ✓ | 1 |
| E(3) ✓ | 1 |
| E(4) ✓ | 1 |
| D() ✓ | 5 |
| E(6) ✓ | 1 |
| E(7) ✓ | 1 |
| D() ✓ | 1 |
| D() ✓ | 1 |
| D() ✓ | 1 |


4 3 2 1

On average TC per operation O(1)

Break 22:20

| | |
|---|---|
| E(1) ✓ | 1 |
| D() ✓ | 3 |
| E(2) | 1 |
| D() | 3 |
| E(3) | 1 |
| D() | 3 |



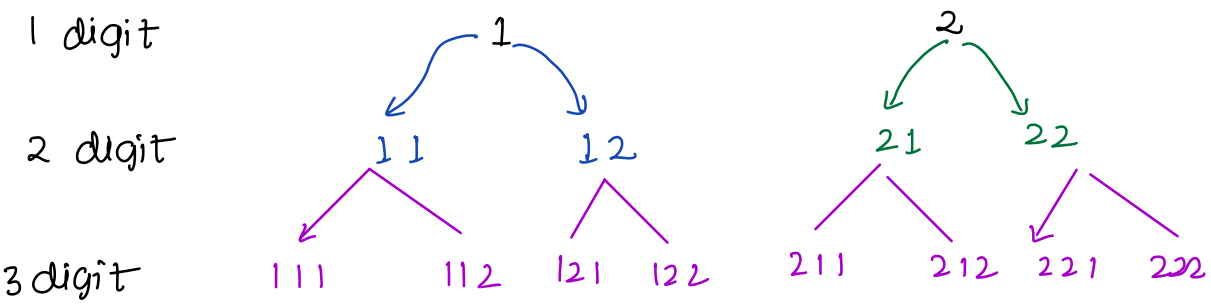
## $N^{th}$ Perfect Number

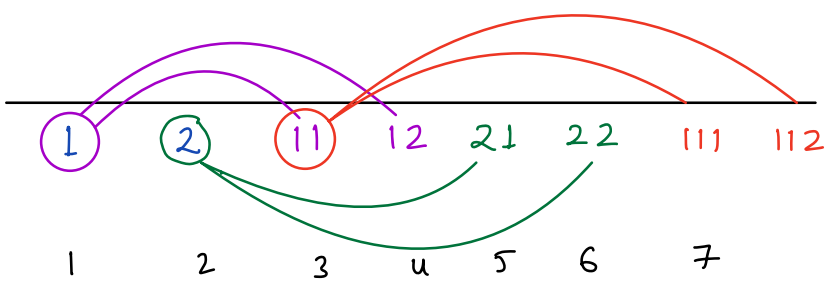
Find $N^{th}$ perfect number ie number formed by only digits 1 or 2

Input — integer N

Output — integer representing $N^{th}$ perfect no.

| N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | ..... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 11 | 12 | 21 | 22 | 111 | 112 | 121 | 122 | ..... |




$7^{th}$ no.

Pseudocode

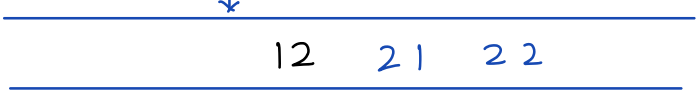
```
if (N <= 2) { return N }
i = 3
queue.enqueue(1)
queue.enqueue(2)

while ( True ) {
    x = queue.dequeue()
    a = 10*x + 1
    b = 10*x + 2
    if (i == N) return a
    if (i+1 == N) return b
    queue.enqueue(a)
    queue.enqueue(b)
    i+=2
}
```

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

N = 7

i = ~~3~~ ~~5~~ 7



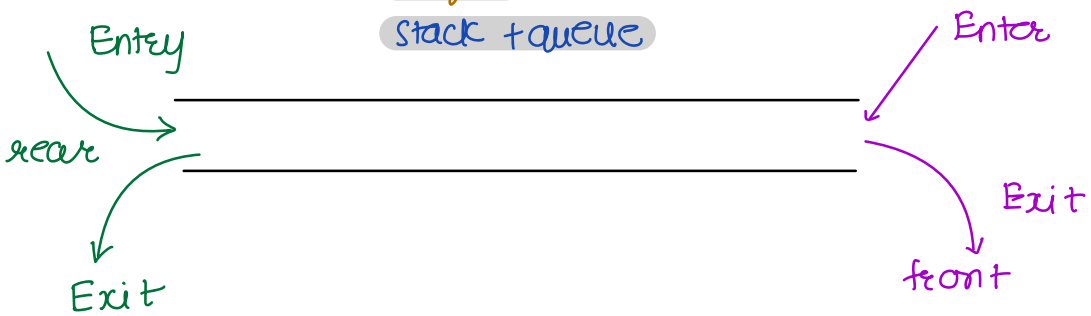

x = ~~1~~ ~~2~~ 11

a = 111
b = 112

Doubly Ended Queue { implemented by DLL }

deque

stack + queue



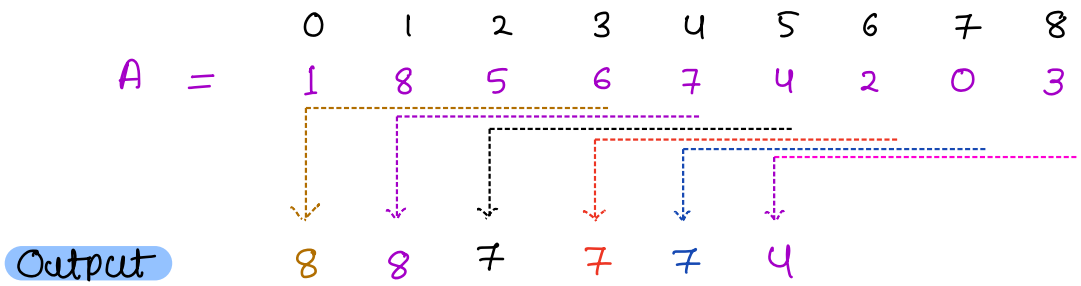

- enqueue() → front / rear
- dequeue() → front / rear
- isEmpty()

learn about deque in your language ***

## Sliding window Maximum ******

Given an integer array A,
∀ window of size K find the max element.

K = 4



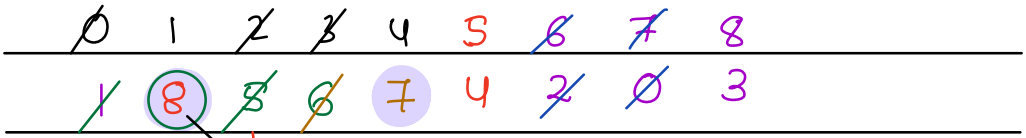

### Bruteforce

For every window of size == k
Find max element.

TC : O(N * K)

$k = 4$

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| A = | 1 | 8 | 5 | 6 | 7 | 4 | 2 | 0 | 3 |



due to 8
1 can't be
my answer for
future windows.
LIFO

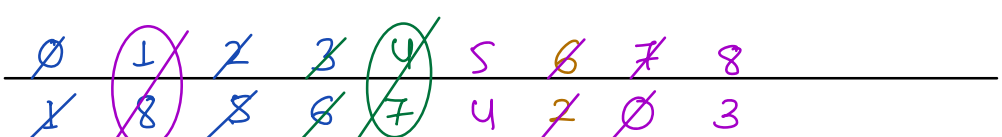
After inserting 4 at index 5
8 at index 1 is out of my
window
FIFO

due to 6
5 can't be
my answer for
future windows
LIFO

$\forall i >= k-1$ → store front of deque

$k = 4$

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| A = | 1 | 8 | 5 | 6 | 7 | 4 | 2 | 0 | 3 |



if $i - k == q.front$

arr 8 8 7 7 7 4

By storing index ⟶ access value

By storing value ⟶ (crossed out) access index