1) Queue Basics
2) Implementation of Queue
3) Problems
   - ① Nth no. using only 1, 2, 3
   - ② Find Nth perfect no.

Real world
- → ATM
- → Shaadi Ka Khana
- → Cafetaria

Computers
- Messaging Queue
- Playlist
- CPU Scheduling
- Printer

FIFO : First In First Out

Data: 7 12 6 14 ↗ 8 3 ↗ 12 ↗

logically ↓

coffee ~~7~~ ~~12~~ ~~6~~ 14 8 3 12

Functionalities of Queue

1) Enqueue(x) : an ele is insert at the back of queue
2) dequeue() : delete an ele from front of the queue
3) front() : return us the ele present at front
4) back()/rear() : returns ele present at back
5) size()
6) isEmpty()

Eg: 7 8 9 dq() front() 10 19 back()

(7 ↑ dq(), 8 ↑ front(), 19 ↑ back())

~~7~~ 8 9 10 19

Implementation

Linked List

Insert at head ~~Remove from tail~~ (Because we can't go back)

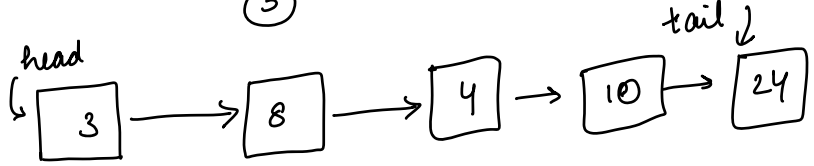
Dequeue will be O(N) solution

| Insert at Tail
| Remove from Head

tail.next = newNode
tail = newNode

3 8 4 10 front() 24 back()

front() → 3



```
int front() {
    if (head == NULL) return -1
    return head.data
}
```

```
int back() {
    if (head == NULL) return -1
    return tail.data
}
```

```
void enqueue (x) {
    Node newNode = new Node()
    if (head == NULL) {
        head = newNode
        tail = newNode
        return
    }
    tail.next = newNode
    tail = newNode
}
```

```
void dequeue () {
    if (head == NULL) return
    temp = head
    head = head.next
    temp.next = NULL
    free(temp)
}
```

* HW: If a single node is present set head & tail to NULL & free your temp

int size() : returns the size of the queue.

TC: O(1) for each function

```
queue<int> q;
q.enque(x)
q.dequeue()
q.size()
q.isEmpty()
q.front()
q.back()
```

* Find it out what library functions are there for queue in your language.

Que: Given K.
There's a series made of only 1, 2 & 3.
↓ increasing order.
Return Kth no. in this series.

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

K=7 : 21

K=9 : 23

K=4 : 11

K=1 : 1

Brute Force

1 .....

cnt=0
Iterate on nos. from [1 ..
check if all digits are 1, 2 & 3.
if yes cnt++
if (cnt == K) return that no. & break.

1 2 3 11 12 13 21 22 23 31 32 33 111 112 113 121

1 2 3 11 12 13 21 22 23 31 32 33 111 112 113 121 122 12

Queue

~~1~~ ~~2~~ 3 11 12 13 21 22 23

How to append 1, 2 & 3 behind a no.?

① String → str + '1' str + '2' str + '3'

② int x10 + 1
int x 10 + 2
int x10 + 3

K = 7

1	2	3

↓

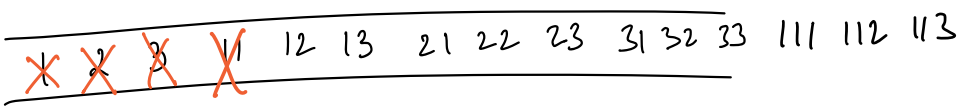
~~1~~ ~~2~~ ~~3~~ 11 12 13 21 22 23 31 32 33 ——

1 x 10 + 1 = 11
1 x 10 + 2 = 12
1 x 10 + 3 = 13

2 x 10 + 1 = 21
2 x 10 + 2 = 22
2 x 10 + 3 = 23

K = 5

cnt = 0




```
queue<int> q
q.enqueue(1)
q.enqueue(2)
q.enqueue(3)
cnt = 0
while (cnt != K-1) {
    int ele = q.front()
    q.dequeue()
    cnt ++
    q.enqueue(ele x 10 + 1)
    q.enqueue(ele x 10 + 2)
    q.enqueue(ele x 10 + 3)
}
return q.front()
```

cnt	K-1
1	!= 4
2	!= 4
3	!= 4
4	== 4

TC : O(K * 3)

O(K * d)

↳ only 3 digits Hence d is constant

O(K)

SC :

O(K x 3)

O(K x d)

O(K)

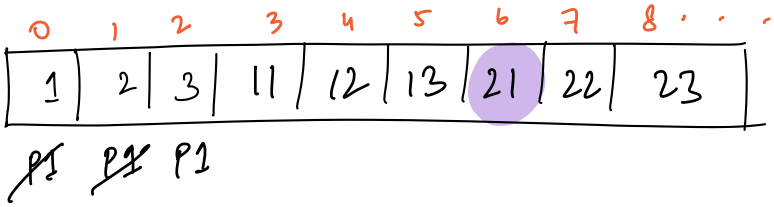
```
List<int> l ;
l.add(1)
l.add(2)
l.add(3)
cnt = 3
while (cnt < K) {
    ele = l[P1]
    l.add(ele x 10 + 1)
    l.add(ele x 10 + 2)
    l.add(ele x 10 + 3)
    P1 ++
    cnt += 3
}
return l[K-1]
```

TC : O(K)

SC : O(K)

K=7 → 6th index

cnt = ~~3~~ ~~6~~ 9



Break : 10 : 50

Q. Nth perfect No.

- ↳ even length
- ↳ palindrome
- ↳ digit (1, 2)

Eg: 11 22 121 343 1221

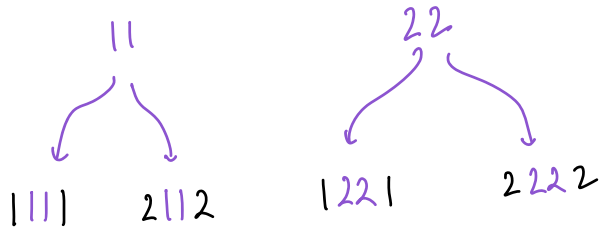
(121 ✗, 343 ✗)

11 22 1111 1221 2112 2222 111111 112211

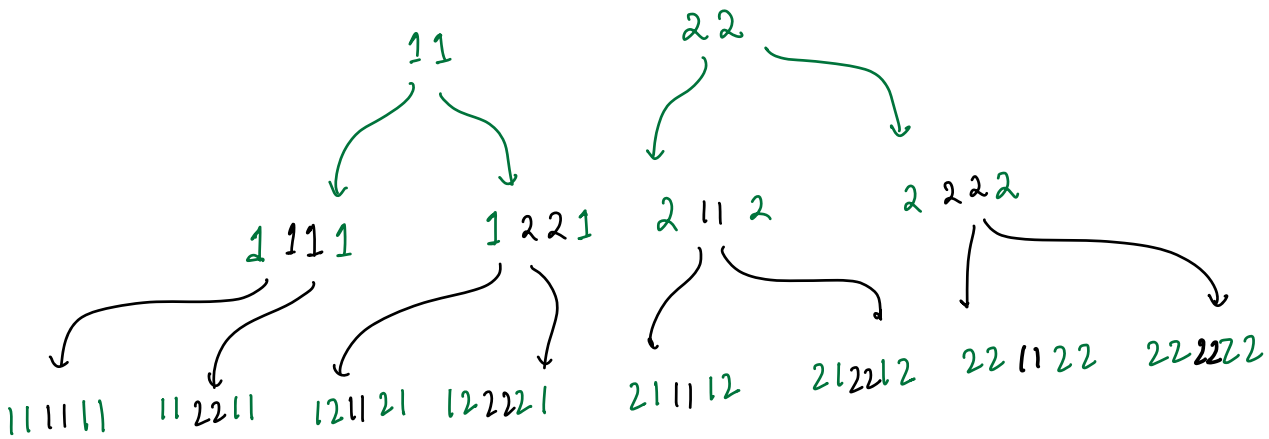
N = 5 2112

When can we say something is palindrome?

. . . . . .



Not in correct order.




```
queue<string> q;
q.enqueue("11")
q.enqueue("22")
```

---

11 22

How to insert in b/w ?

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

str(0 - 3) + "11" + str(4 - 7)

Check the String operation in your language

Que. Implement Queues using stacks (only)

enqueue(x)
dequeue()

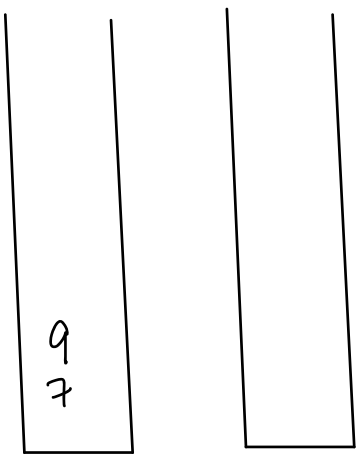
- push(x)
- pop()
- top()
- size()
- isEmpty()

Imaginary

~~9~~ ~~7~~ 7 9

One way

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



TC:

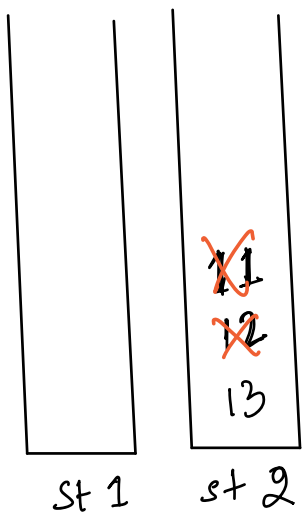
Enqueue (x) $\Rightarrow O(1)$

dequeue() $\Rightarrow O(N)$

Another way.

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

~~5~~ ~~4~~ ~~7~~ ~~9~~ ~~10~~ 11 12 13



eq(7) eq(9) eq(11) eq(14) eq(17) eq(19)

eq(1) eq(7) dq() dq() dq() dq()

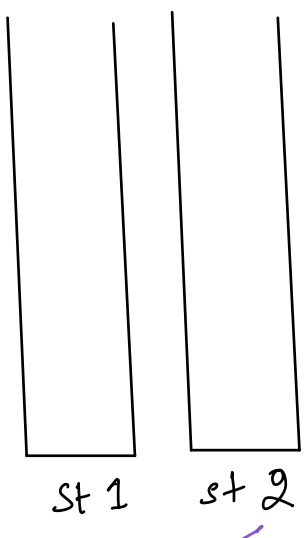
7 9 11 14

eq(7) dq() dq()

17 19

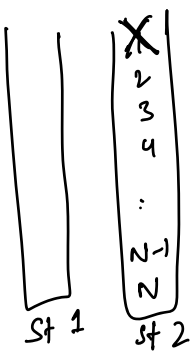
dq() dq() dq() dq()

1 ≠ 7



```
void equeue(x){
    s1.push(x)
}
```

```
void dequeue(){
    if(s2.size==0){
        while(s1.size()>0){
            s2.push(s1.top())
            s1.pop()
        }
    }
    if(s2.size()>0){
        s2.pop()
    }
}
```



for 1 dq → N iterations → 1 X N

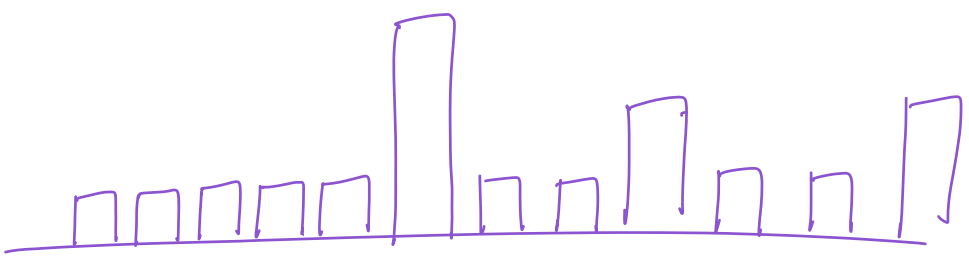
for N-1 dq → TC: O(1) → (N-1) x 1

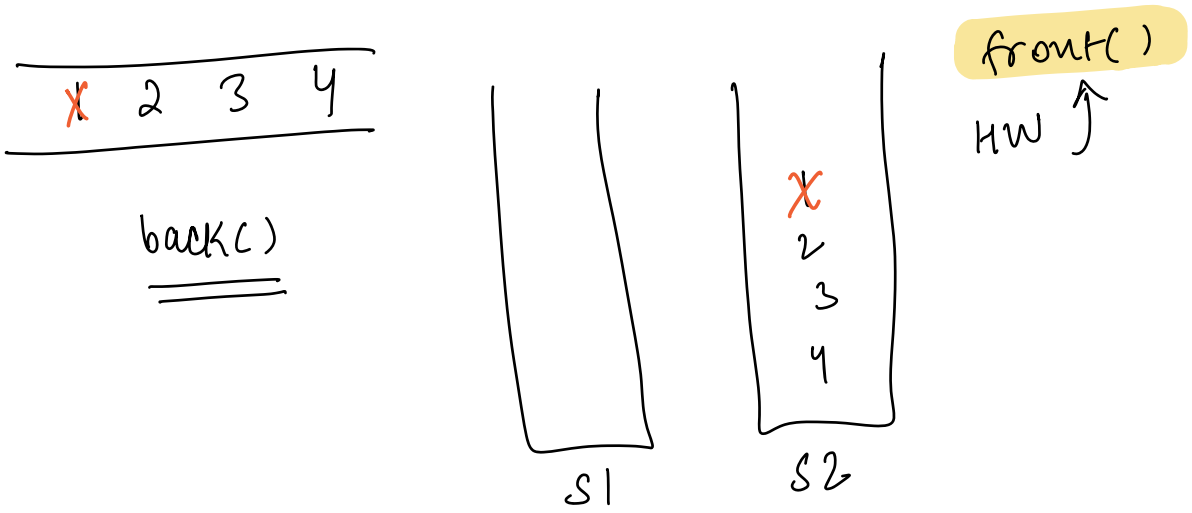
Total iterations:

N dq → N + N - 1 = 2N - 1

≈ O(N)

1 dq → O(1) Amortized

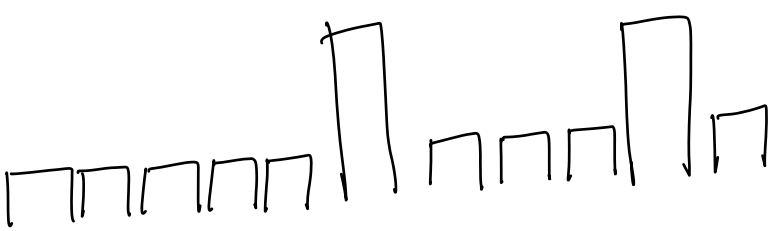




Doubts

O(N) + O(Q)

queries.



Q → O(1) amortized

```
queue<int> q
q.enqueue(1)
q.enqueue(2)
q.enqueue(3)
cnt = 3
while ( cnt < K ) {
    int ele = q.front()
    q.dequeue()
    if (cnt < K) {
        q.enqueue( ele x10 + 1)
        cnt ++
    }
    if (cnt < K) {
        q.enqueue( ele x 10 +2)
        cnt ++
    }
    if (cnt < K) {
        q.enqueue( ele x 10 +3)
        cnt ++
    }
}
return q.back()
```

K = 7

~~1~~ 2 3 11 12 13 21

cnt		K
3	<	7
4	<	7
5	<	7
6	<	7