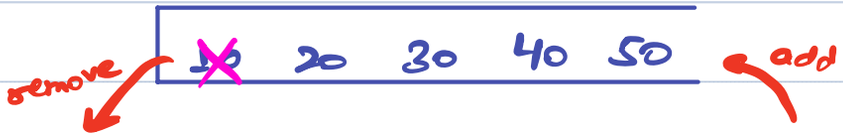
Today's agenda

- Queue basics
- Reverse first K ele in Queue
- implement queue using Stack
- Kth number using only 1 & 2

Stacks → LIFO

Queue



↳ First in first out (FIFO)

real life ex:

① line

② task scheduling

Syntax:

↳ Queue < Integer > q = new LinkedList <>();

① q.add(x) → insert x at the end of queue.

② q.remove() → delete element from front.

③ q.size() → no. of element in queue.

④ q.peek() → return the front element.

Q) Reverse first K elements

↳ Given a Queue, Reverse it's first K elements.

Ex: K=4

3 10 2 12 19 6 8 10 14

↓

12 2 10 3 19 6 8 10 14

Idea

↳ Put first K elements in stack.

K=4

Q: ~~3~~ ~~10~~ ~~2~~ ~~12~~ 19 6 8 10 14

Q: ~~19~~ ~~6~~ ~~8~~ ~~10~~ ~~14~~ 12 2 10 3

19 6 8 10 14

Q: 12 2 10 3 19 6 8 10 14

~~12~~
~~2~~
~~10~~
~~3~~

//Pseudo code

```
Queue<> ReverseKelements (Queue<> q, int k){
    Stack<Integer> s = new Stack<>();

    for(int i:0; i<k; i++){
        s.push(q.remove());
    }

    for(int i:0; i<k; i++){
        q.add(s.pop());
    }

    for(int i:0; i<n-k; i++){
        q.add(q.remove())
    }
    return q;
}
```

T.C: $O(k+k+n-k)$
$= O(n+k) = O(n)$

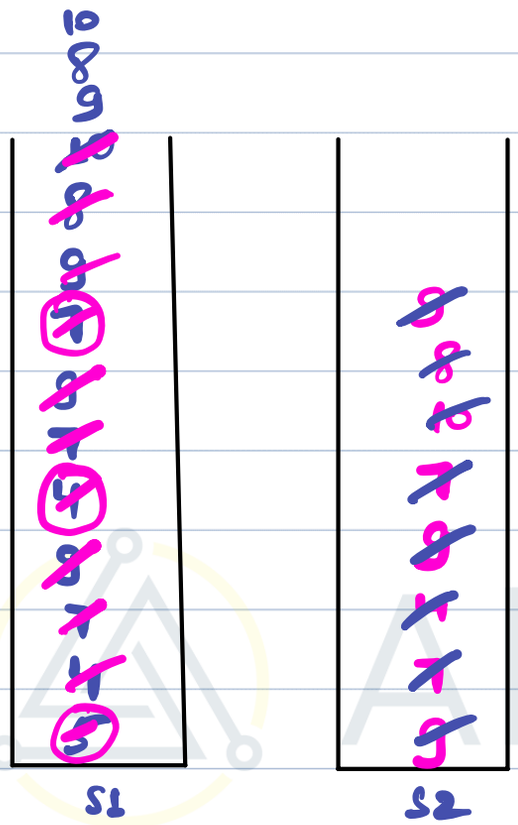
S.C: $O(1)$

Q) Implement Queue using Stacks

FIFO → LIFO

op: 5 4 7 9 rem rem() 8 10 rem rem 14 rem

rem → 5, rem() → 4, rem → 7



Idea1 → add efficient

add(n): add n in S1.

↳ O(1)

remove(): → O(N)

(i) move n-1 elements S1 → S2

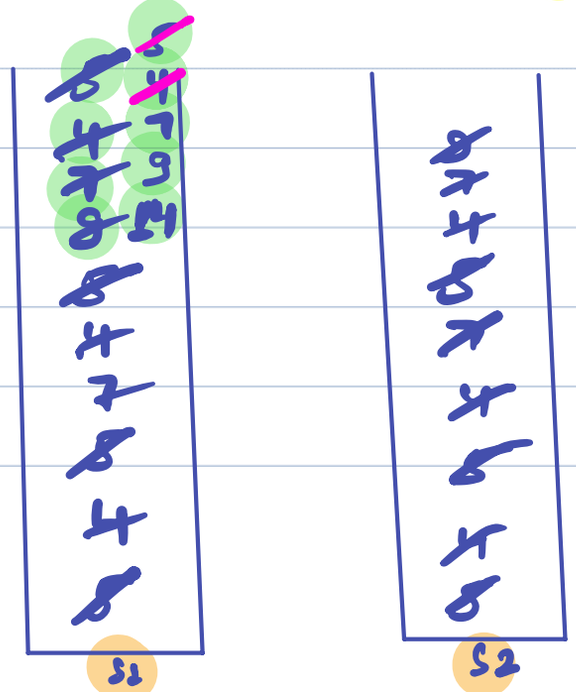
(ii) remove the one from S1

(iii) move all elements S2 → S1

Idea2 → remove efficient

op: 5 4 7 9 14 rem rem() 8 10 rem rem

rem → 5, rem() → 4



add(n): → O(N)

(i) move all elements S1 → S2

(ii) add n to S1

(iii) Put back all elements S2 → S1

remove() → O(1)

↳ remove from S1.

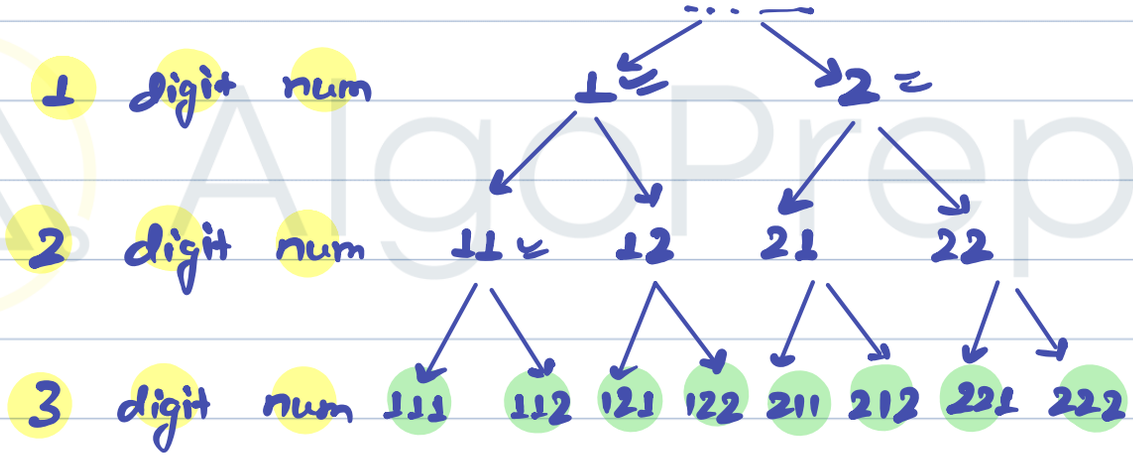
Q) Kth Number

↳ Generate **Kth number** in series using digits **1 and 2 only.**

K=5 1 2 11 12 21

K=7 1 2 11 12 21 22 111

Idea



Algo

K=5

Count = 0 1 2 3 4 5

"1" "2" "11" "12" "21" "22" "111" "112" "121" "122"

ans = "21"

//Pseudo code

```
String kthnumber(int k){
    Queue<String> q = ....;

    q.add("1");
    q.add("2");

    String ans = "";

    for(int i=1; i<=k; i++){
        String t = q.remove();
        if(i==k){ans = t;}
        q.add(t + "1");
        q.add(t + "2");

    }

    return ans;

}
```

Avg. digit count

T.C: O(k*n)

S.C: O(k)

Break till 10:50 Pm

K:5

ans: 21

```
q.add("1");
q.add("2");

String ans = "";

for(int i=1; i<=k; i++){
    String t = q.remove();
    if(i==k){ans = t;}
    q.add(t + "1");
    q.add(t + "2");
}

return ans;
```

t: 21

t=2

Q)

↳ Generate Kth Palindrome number in series using digits 1 and 2 only

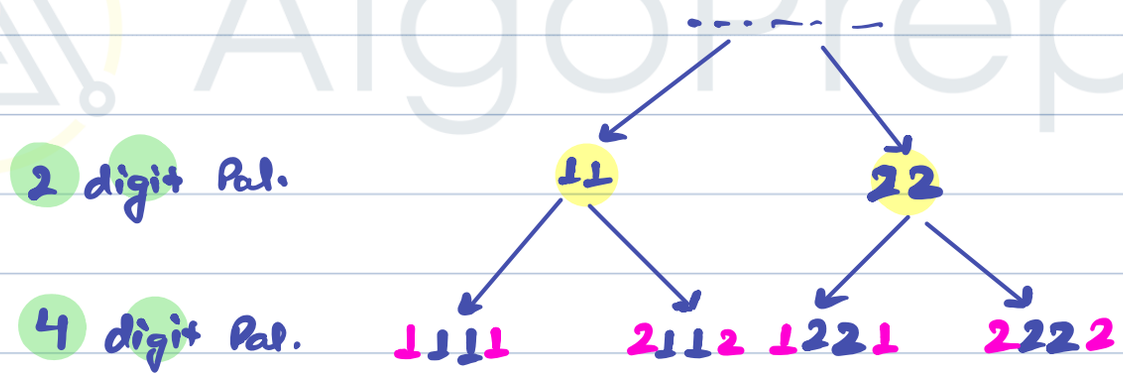
Note: only consider even digit numbers.

K=5 : 11 22 1111 1221 2112

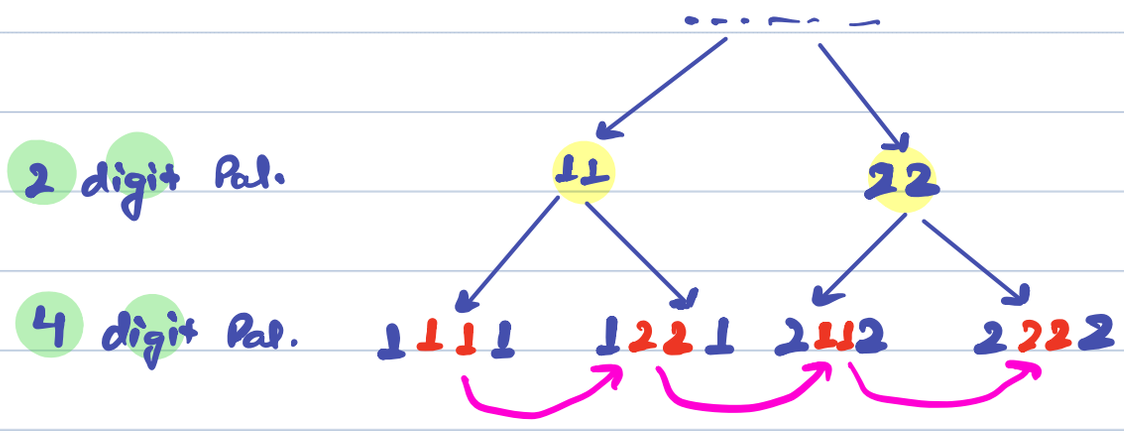
Idea 1

↳ Keep generating numbers using 1 and 2, check even digit Palindrome count then only. Return the kth one.

Idea 2



→ insert 11 and 22 in the middle



IIPSuedo code

```
String KthPalindrom(int k){
    Queue<String> q = ....;

    q.add("11");
    q.add("22");
    String ans = "";

    for(int i=1; i<=k; i++){
        String temp = q.remove();
        if(i==k){ans=temp;}

        String left = temp.subString(0, temp.length()/2);

        String right = temp.subString(temp.length()/2, temp.length());

        q.add(left + "11" + right);
        q.add(left + "22" + right);
    }

    return ans;
}
```

Avg. digit count

T.C: O(k*n)

S.C: O(k)