AGENDA

- → Stacks Basics
- → Remove duplicates
- → Sort stack
- → Infix to Postfix
- → Postfix Evaluation

Stacks

LIFO :

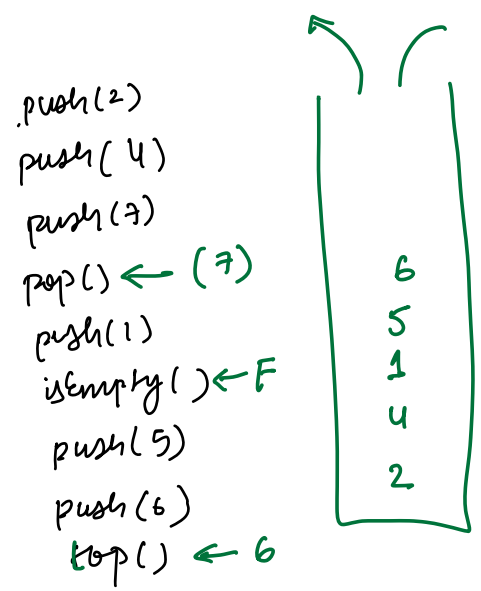
A data structure in which element inserted at last is the first one to come out

Applications

1) UNDO / REDO
2) Browser tabs
3) Recursion
4) Expression Evaluation
5) Balanced Parenthesis

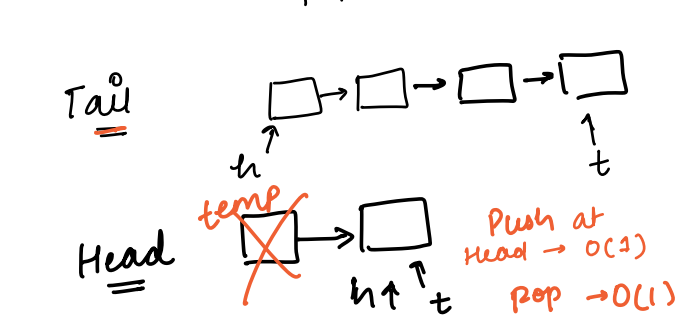
Functionalities

1) push(x) : pushes an ele on top of stack
2) pop() : deletes the ele from top of stack
3) peek()/top() : returns the ele present on top of stack.
4) isEmpty() : returns T/F depending on stack is empty / full.



Implementation → Arrays.
→ Linked List

push(x) // O(1)
pop() // O(1)



Node newNode = new Node(x)
newNode.next = head
head = newNode

Q. Walmart | Google | Bikaji

Given a string, remove every pair of consecutive duplicates until there are no consecutive duplicates.

s: a c b b c K

a c c K

[a K] ans

s: a a a b

[a b]

s: a b c K K c b a m

a b c c b a m

a b b a m

a a m

m.

s: a b a b a b

↓

a b a b a b ≡

s: a c r b b c K

a c r c K

Palindrome

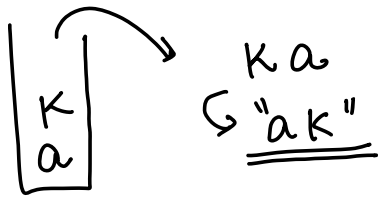
a a r b b x x r a c a

Try to implement

a c b b c K

TC : O(N)

SC : O(N)

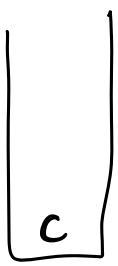


Follow up questions

Remove all duplicated characters.

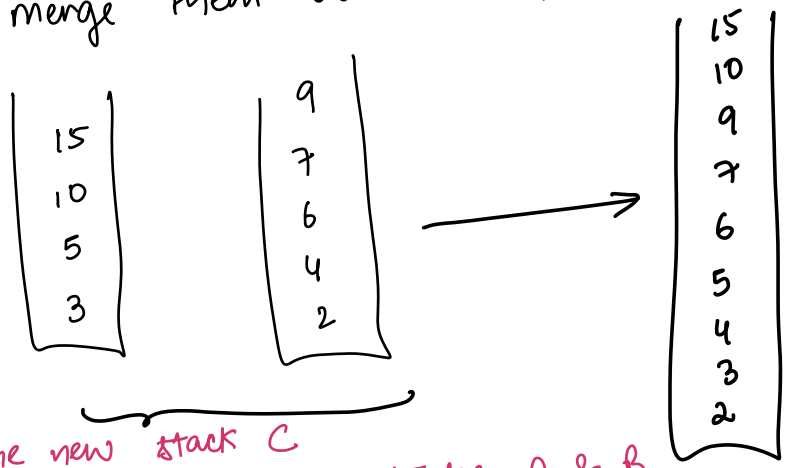
b a a a b b c

flag = ~~F~~ ~~T~~ ~~F~~ T F

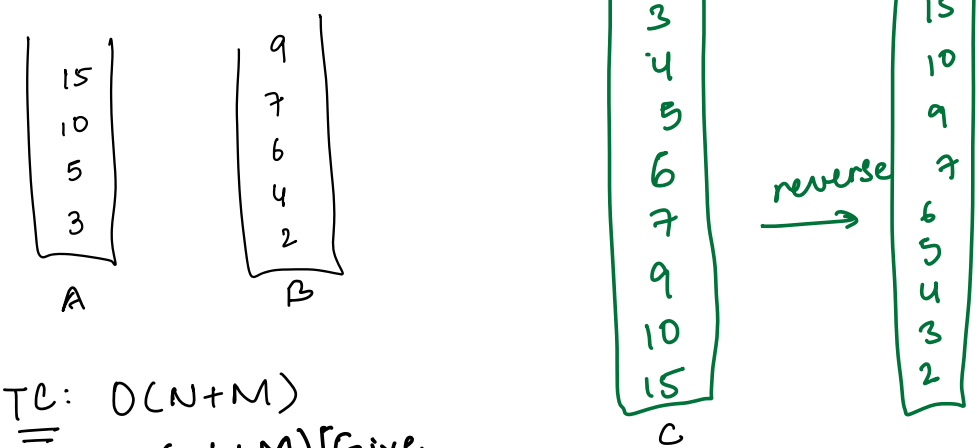


a a r r r b b x x r a c a

Que. Given 2 sorted stacks.
merge them into one sorted stack



Take the new stack C
Compare top ele on both stacks A & B
Push the bigger ele on top of stack.



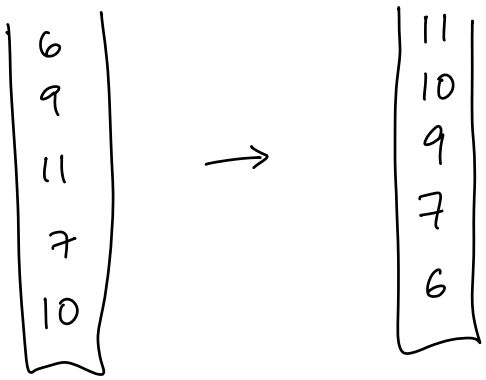
TC: O(N+M)
SC: O(N+M) [Give good explanation for it]

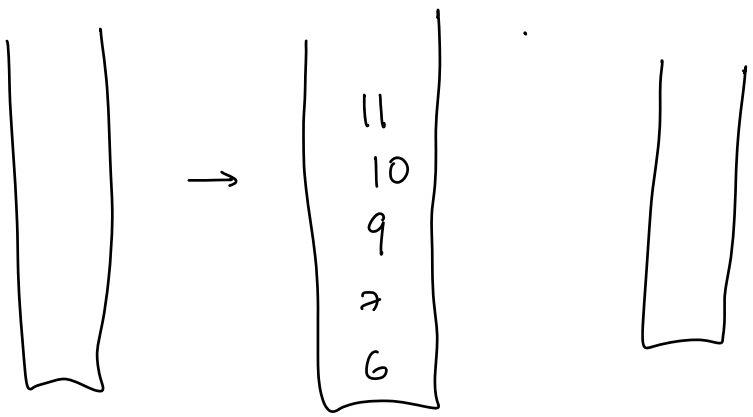
If LL: O(1)
If arrays: O(N+M)

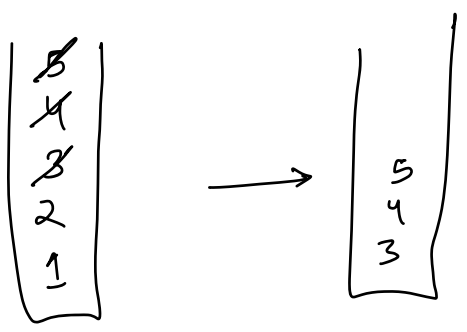
Merge the two stacks into one.

```
stack<int> merge(stack<int> A, stack<int> B) {
    stack<int> C;
    while (A.size()>0 && B.size()>0) {
        if (A.top() > B.top()) {
            C.push(A.pop())
        }
        else {
            C.push(B.pop())
        }
    }
    while (A.size() > 0) {
        C.push(A.pop())
    }
    while (B.size() > 0) {
        C.push(B.pop())
    }
    C = reverse(C);
    return C;
}
```

Q. Given an unsorted stack, sort it.

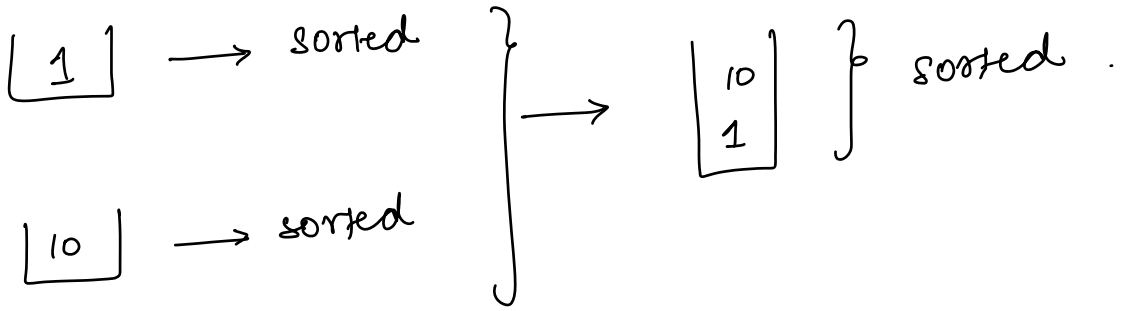






TC: $O(N^2)$

desc

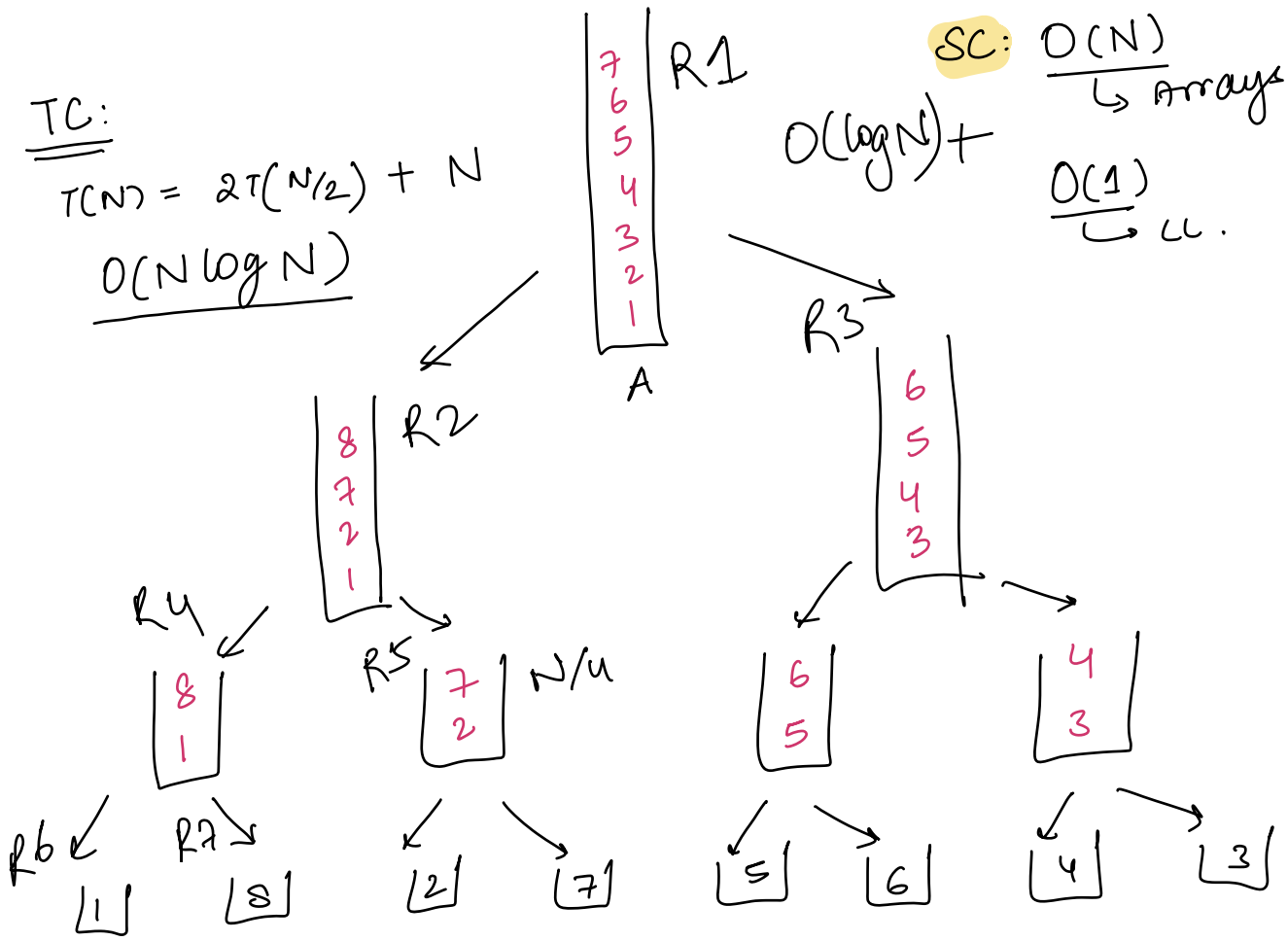


Merge Sort

```
mergeSort (data) {
    d1 = first half
    d2 = second half
    mergeSort(d1)
    mergsort (d2)
    retun merge (d1 , d2)
}
```

```
stack<int> mergeSort ( stack<int> A) {
    if ( A.size() <= 1) retun A
    stack<int> B
    s = A.size()
    for(i=0; i< (s/2) ; i++){
        B.push( A. pop()) ;
    }
    A = mergeSort(A)
    B = mergeSort (B)
    retun merge( A, B)
}
```

i	A.size()/2		
0	8/2	4	(1)
1	/2	3	(2)
2	/2	3	(3)
3	/2	2	



Break

11:00 PM

Expression Evaluation

$7 \times 1 + 2 - 8 \times 3 + 10 / 5$

$7 + 2 - 24 + 2$

$9 - 24 + 2$

$-15 + 2$

-13

$/ \ *$ — L to R

$+ \ -$ — L to R

Infix Expressions

A + B

A − B

A / B

A * B

Postfix Expression

A B +

A B −

A B /

A B *

A + B x C

A + BC *

A BC * +

Infix → Postfix

Postfix → Ans.

QUIZ 1 4 + 8 * 7

4 + 8 7 *

4 8 7 * +

QUIZ 2 $10 + 3 * 4 - 7$

$10 + 34* - 7$

$10\ 3\ 4\ *\ +\ -\ 7$

$10\ 3\ 4\ *\ +\ 7\ -$

QUIZ 3 $10 / (4-2) * 6 + 9$

$10 / 42- * 6 + 9$

$10\ 4\ 2\ -\ /\ * 6 + 9$

$10\ 4\ 2\ -\ /\ 6\ *\ 9\ +$

QUIZ 4 : $(10+3) * 2 - (7-6) * (4+8)$

$10\ 3\ +\ *2-\ 7\ 6\ -\ *\ 4\ 8\ +$

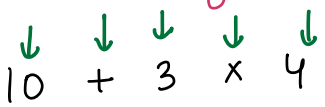
$10\ 3\ +\ 2\ *\ -\ 7\ 6\ -\ 4\ 8\ +\ *$

$10\ 3\ +\ 2\ *\ 7\ 6\ -\ 4\ 8\ +\ *\ -$

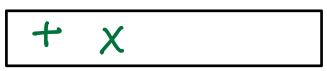
$10 + 3 \implies 10\ 3\ +$

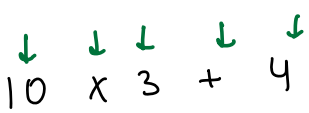
$10 + 3 \times 4 \implies 10\ 3\ 4\ \times\ +$

* Relative ordering of operands is maintained.

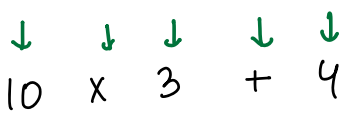


$10\ 3\ 4\ \times$

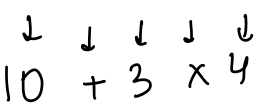




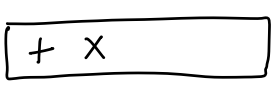
$10\ 3\ 4$



$10\ 3\ \times\ 4\ +$



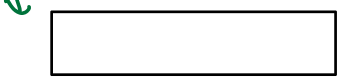
$10\ 3\ 4\ \times\ +$



10 + 3 X 4 - 7

10 3 4 * + 7 -

10 3 4 X + 7 -

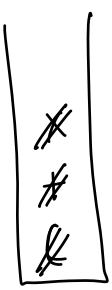


10 X (3 + 4)

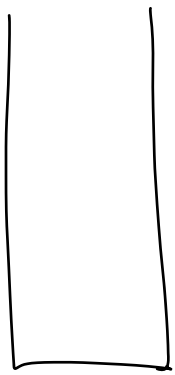
10 3 4 + X

(10 + 8) X 5

10 8 + 5 X

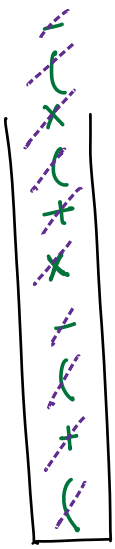


$3 + 10 \times (3 - 4/2) + 3$



3 10 3 4 2 / - x + 3 +

$(2 + (4 - 1) \times 5) + (6 \times (5 - 1))$



2 4 1 - 5 * + 6 5 1 - x +

Traversing the string

s[i]

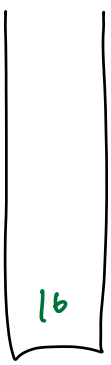
- operand → directly add to the ans
- '(' → push on stack
- ')' → keep popping ele from stack & adding to the answer
- operator → while(prec(top) >= prec(curr))
  - True → keep popping & adding to the postfix answer
  - False → push operator on stack.

TC: O(N)

SC: O(N)

Evaluation

3 10 3 4 2 / − x + 3 +



(16) =