Stack — LIFO

- push(x)
- Pop()
- Size()
- Peek() / top()

Applications

- Undo / redo
- rec call stack
- back button
- Calculations / Expression eval
- Memory

Implement → Array, Linked Lists

Amazon

Q Given a string. Remove every consicutive duplicate (pair) then untill there are no consecutive duplicate.

s : a c b b c k

↓

a c c k

↓

a k

s : a a a b

↓

a b

s : a b c k k c b a m

↓

m

s : a b a b a b

↓

a b a b a b

a c b b c k

a c b k ↓ ↓ => a k => Stack

Follow-up Questions

- Remove all duplicates

  b a a a b b c ⟶ c

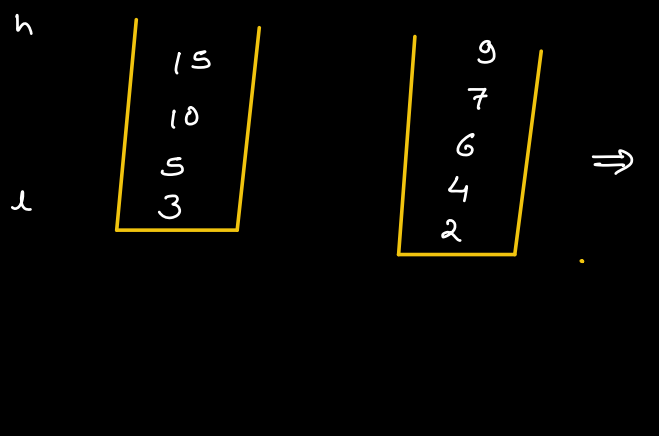
- Remove k-duplicate

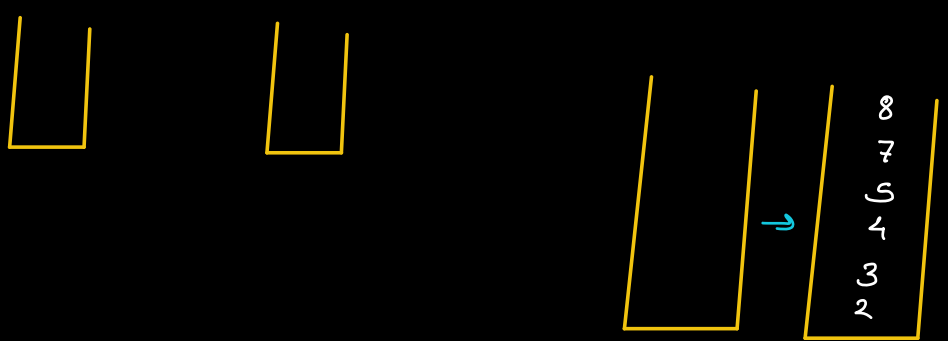
  b a a a a k = 3

  ⟶ b a

Q. Given 2 sorted stacks. Merge them in sorted orch.

Amazon





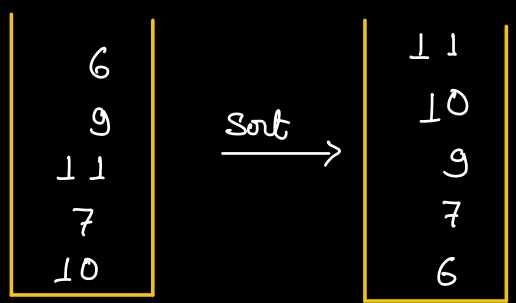
HW: Reverse a stack using recursion.

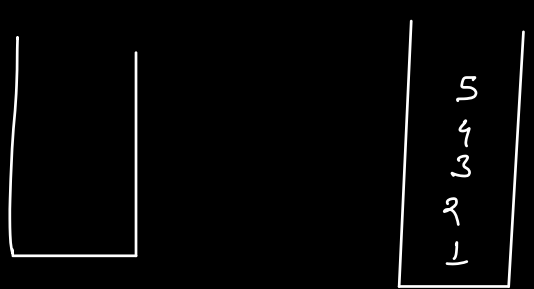
```
Stack<Integer> merge(Stack<Int> S1,
                     Stack<Int> S2){

    Stack<Integer> S = new Stack<Integer>();

    while (S1.size() > 0 && S2.size() > 0){

        if(S1.peek() > S2.peek()){
            S.push(S1.pop());
        } else {
            S.push(S2.pop());
        }
    }

    if(S1.size() == 0){
        while (S2.size() > 0){
            S.push(S2.pop());
        }
    }

    if(S2.size() == 0){
        while (S1.size() > 0){
            S.push(S1.pop());
        }
    }

    S = reverse(S);
    ret S;
}
```

```
Stack<Integer> reverse(St s){
    Stack<Integer> revSt = new Stack<Integer>();
    while(S.size() > 0){
        revSt.push(S.pop());
    }
    ret revSt;
}
```

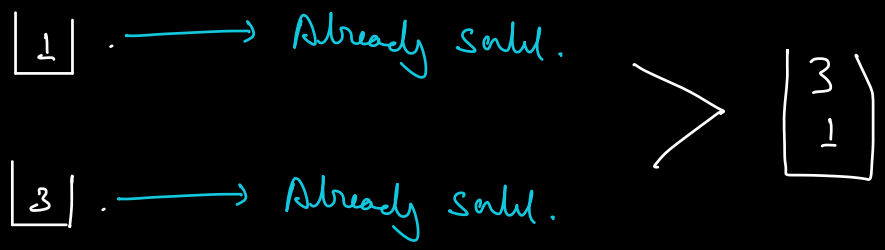
Q Given a stack. Sort it in desc Order. → By using only stacks.

Google

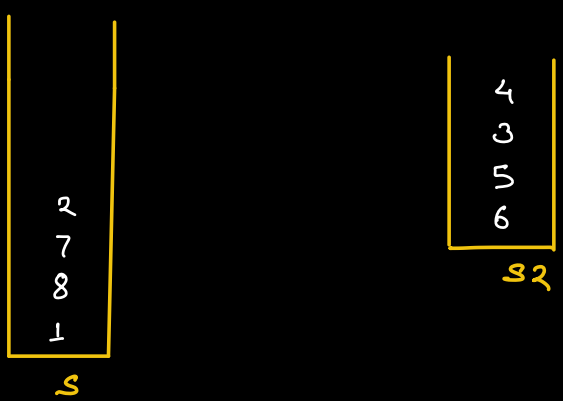




$\Rightarrow O(N^2)$



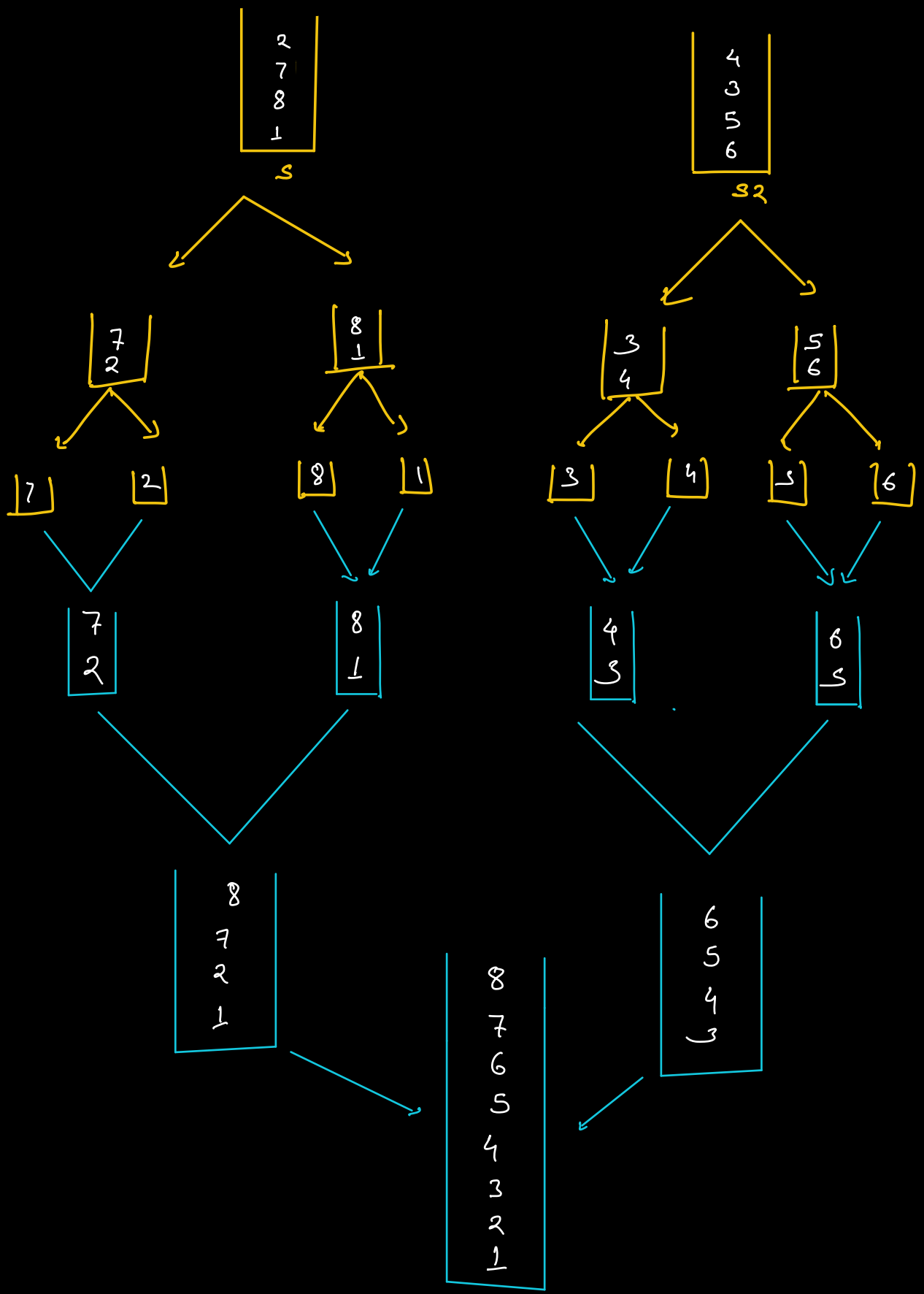

```
mergeSort(Data) {
    D1 → first half of Data
    D2 → 2nd half of Data
    D1 = mergeSort(D1);
    D2 = mergeSort(D2);
    ret merge(D1, D2);
}
```



```
Stack<Int> mergeSort(Stack<Int> S) {
    if (S.size() <= 1)
        ret S;

    Stack<Int> S2 = new Stack<Int>();
    size = S.size();
    for(i=0; i< size/2 ; i++){
        S2.push(S.pop());
    }
    S = mergeSort(S);
    S2 = mergeSort(S2);
    ret merge(S, S2);
}
```

TC : $O(N \log N)$

SC : $O(N)$


2 7 8 1
S
4 3 5 6
S2
7 2
8 1
3 4
5 6
7
2
8
1
3
4
5
6
7 2
8 1
4 3
6 5
8 7 2 1
6 5 4 3
8 7 6 5 4 3 2 1

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

$\Rightarrow -13$

/

*

+ −

Infix Notation

Postfix Notation

$A \times B \longrightarrow A\ B\ \times$

$A \div B \longrightarrow A\ B\ \div$

$A + B \longrightarrow A\ B\ +$

$A - B \longrightarrow A\ B\ -$

$A + (B \times C)$

$A + (B\ C\ \times)$

$A\ B C \times\ +$

* Infix → Postfix ✓
* Postfix evaluation ✓

Infix to postfix

$4 + (8 \times 7) \Rightarrow (4) + (8\ 7\times) \Rightarrow 4\ 8\ 7\times +$

$10 + (3 \times 4) - 7 \Rightarrow (10 + (3\ 4\times) - 7 \Rightarrow (10\ 3\ 4\times +) - (7)$

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

$10 / (4-2) \times 6 + 9 \Rightarrow (10 / (4\ 2-)) \times 6 + 9$

$(10\ 4\ 2-/) \times 6 + 9$

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

$(10+3) \times 2 - (7-6) \times (4+8) \Rightarrow (10,3+) \times 2 - (7,6-) \times (4,8+)$

$10,3+\ 2\times\ -\ 7,6-\ 4,8+\ \times$

$10,3.+\ 2\times\ 7.6-\ 4\ 8+\ \times\ -$

$10 + 3 \longrightarrow 10\ 3\ +$

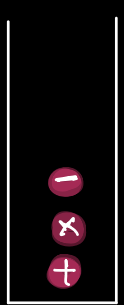
$10 + 3 \times 4 \longrightarrow 10\ 3\ 4\ \times\ +$

The operands follow the same relative order as in the infix story.

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

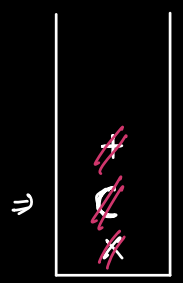
$10 \times 3 + 4 \longrightarrow 10\ 3\ \times\ 4\ +$

$10 + 3 \times 4 - 7$



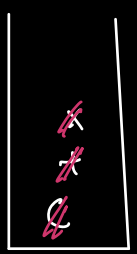
$10\ 3\ 4\ \times\ +\ 7\ -$

$10 \wedge (3+4)$



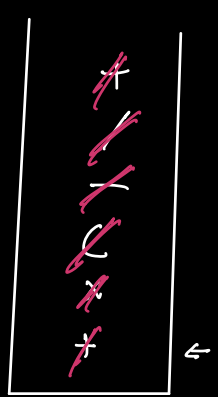
10 3 4 + ^

$(10+8) \times 5$



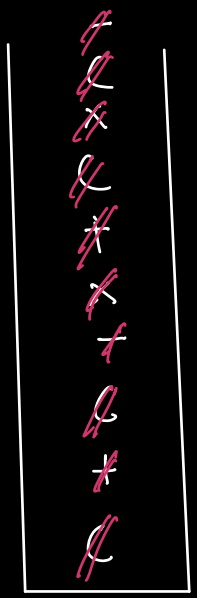
10 8 + 5 x

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

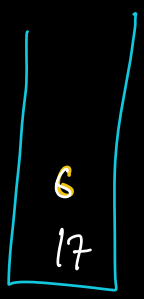


3 10 3 4 2 / - ^ + 3 +

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



2 4 1 − 5 × + 6 5 1 − × +



− 1 5

Traverse the string

S[i]

Operand

add it to Postfix string

"("

Push to stack

")"

Keep popping & adding in postfix string till you get "("

Operator (+, /, x, −)

if(Prec(cur) <= Prec(top)){

True

Keep popping from stack & keep adding in postfix Untill Prec(top) < Prec(cur) // stack is emp

false

Push

Keep popping & adding in postfix Untill stack is empty.

$4 - 1 \times 5 \longrightarrow 4\ 1\ 5\ \times\ -$

$(4 - 1) \times 5 \longrightarrow 4\ 1\ -\ 5\ \times$