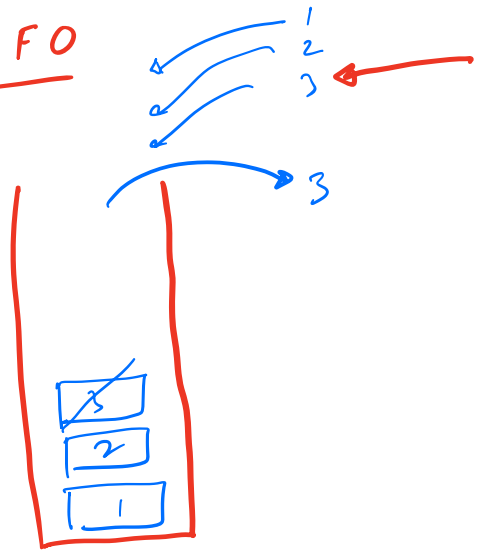
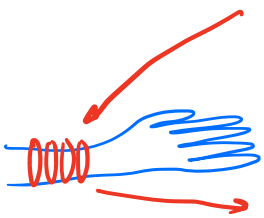
Stack

LIFO





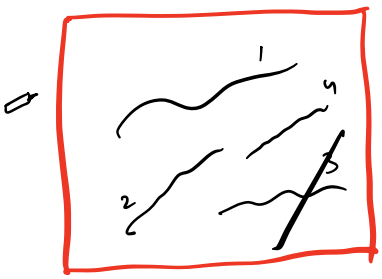
Ops of a stack

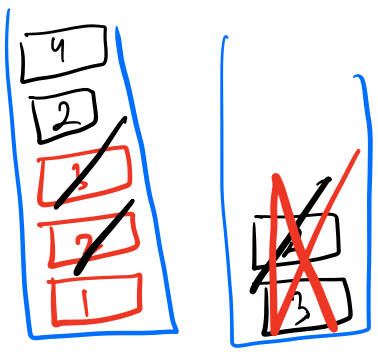
- push(x) → push x at the top of the stack
- pop() → remove the topmost —
- top(), peek() → get the topmost —
- size() → get the # of items —

TC = O(1)

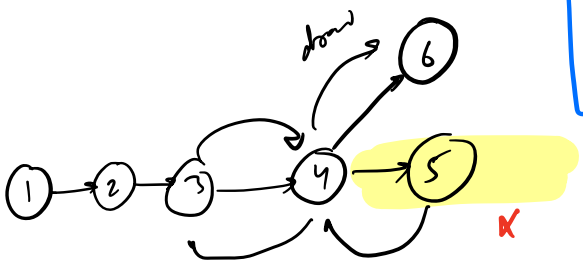
Application of Stack

- Recursion
- Undo / redo

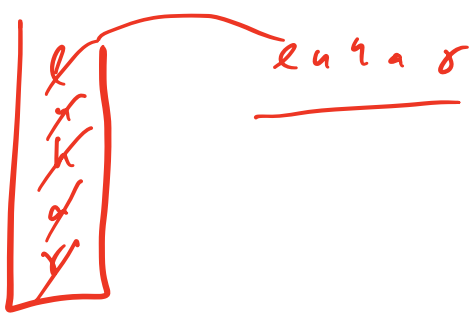




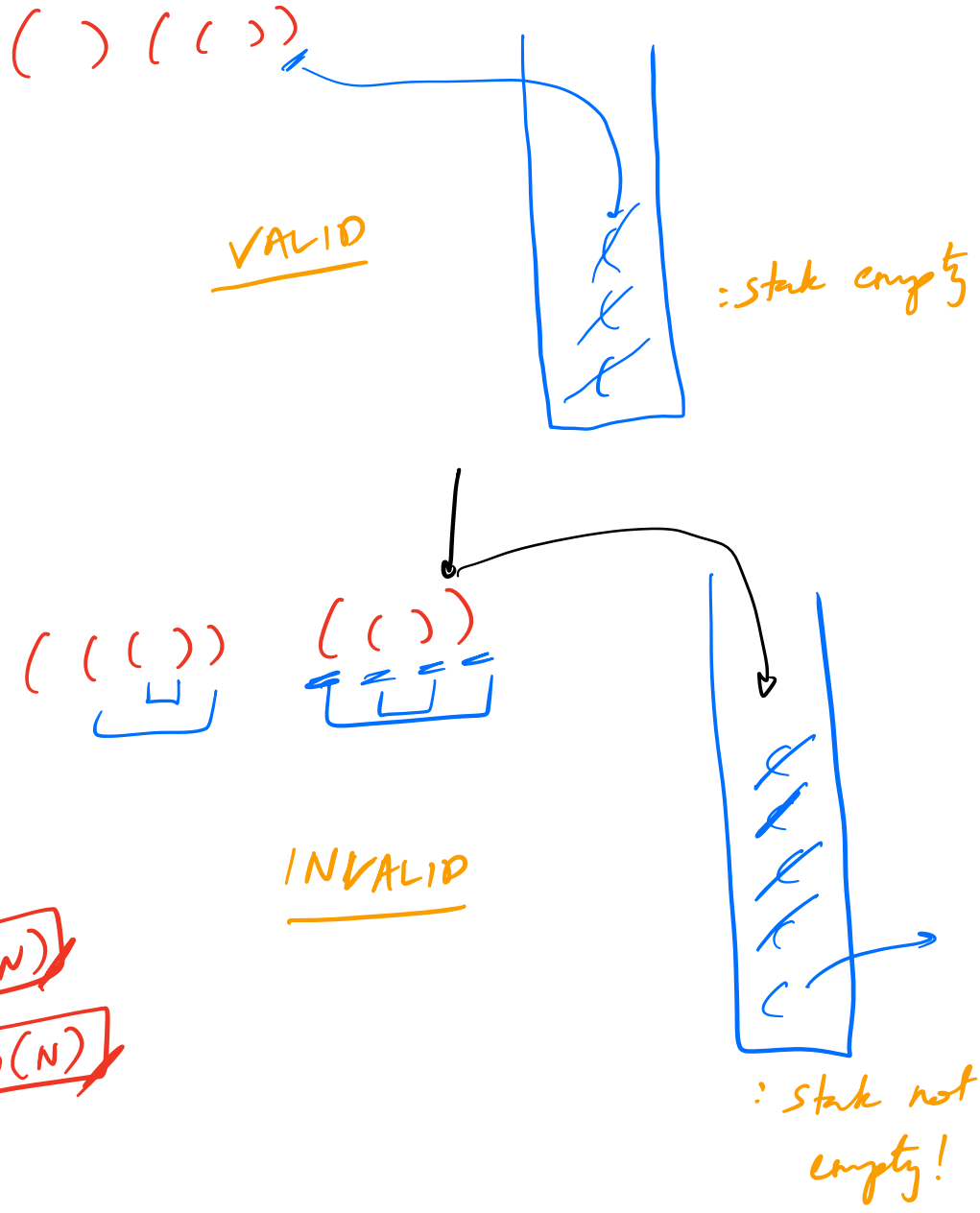
UNDO
UNDO
REDO
DRAW



- back / fwd button : browser
- Expression Evaluation
- parenthesis check
- Reverse anything !



Q Given an expression. Validate the parenthesis.



TC = O(N)

SC = O(N)

I Maintain an open variable

(( ) (( )))

op = 0 1 2 1 2 3 2 1 0 → op = 0 At end

↳ VALID

(( ))) ( )

op : 0 1 2 1 0

NOT VALID !

TC = O(N)

SC = O(1)

Q SAME QUES

( ) [ ] { }

({ ( ) } [ ]) { ( ) }

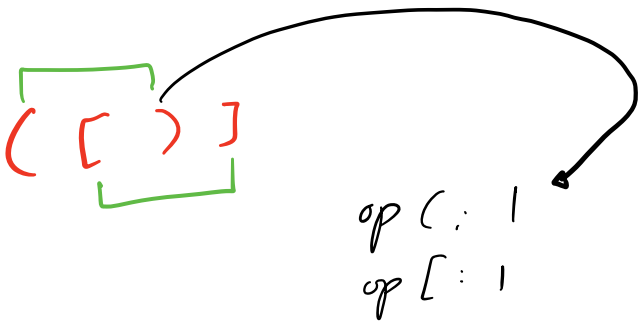
1) STACK

VALID !

TC : O(N)

SC : O(N)

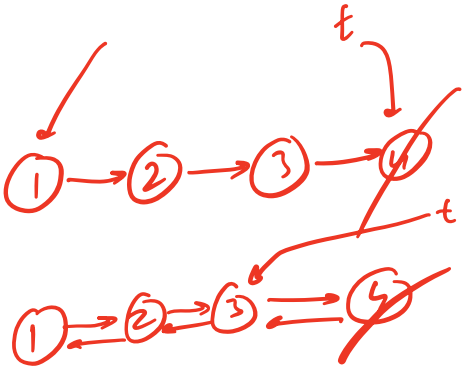
2) 3 Variables → X



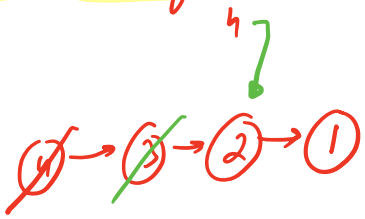
⊙ Implementation of Stack →

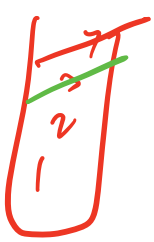
⊙ Array → Static ✗
⊙ Array → dynamic ✓

⊙ LL → SLL → ✓✓
⊙ LL → DLL → ✓



SLL: Adding a node at head.





Q Given a string. Remove every consecutive duplicate pair of chars until there are none left.

S: a c b b c k

a c c k

a k

ak

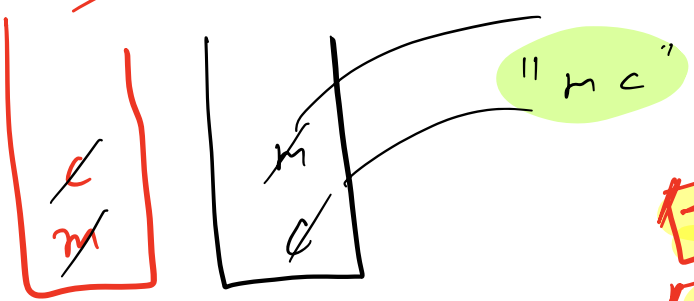
S: a b c k k c b a m d d c

a b c c b a m c

a b b a m c

a a m c

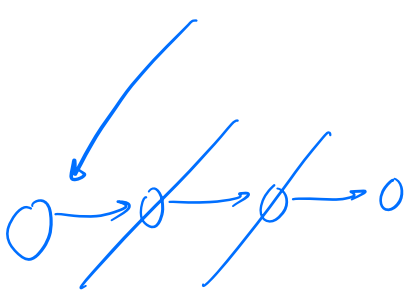
m c



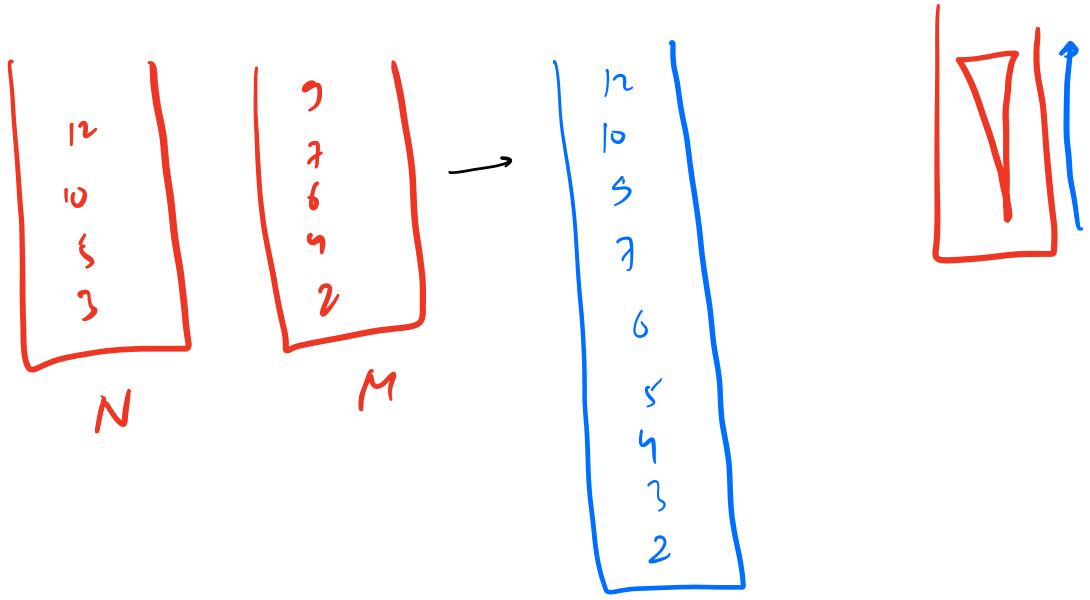
"m c"

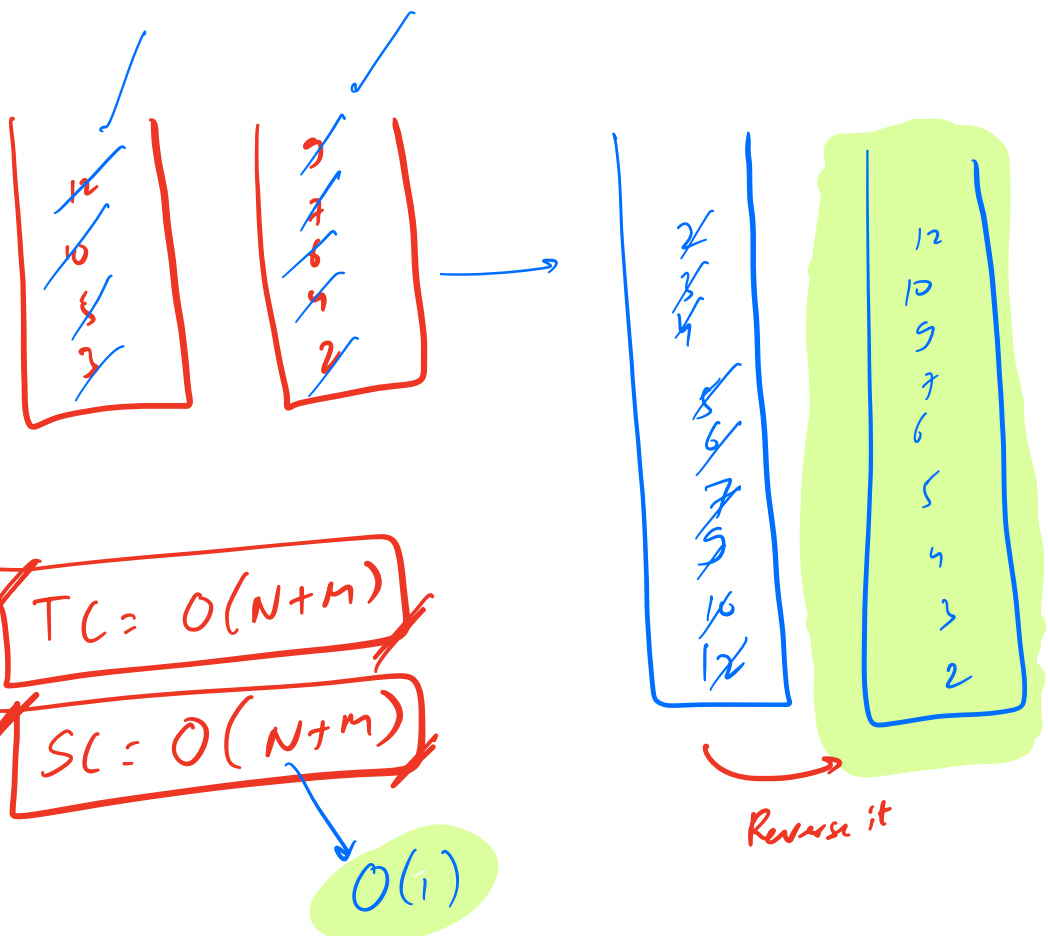
TC = O(N)

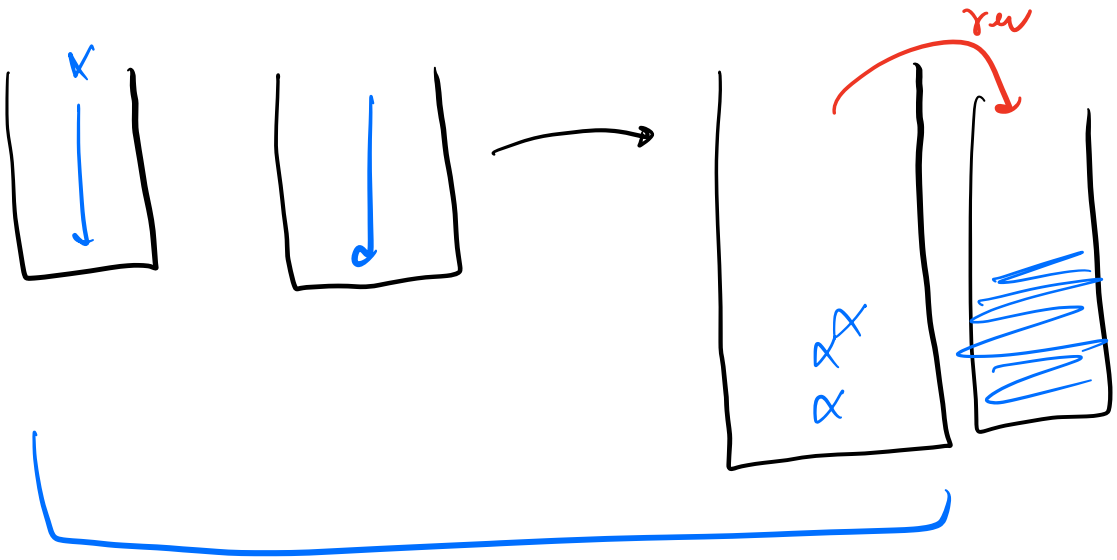
SC = O(N)



Q Given 2 sorted stacks. Merge them into 1 sorted stack!

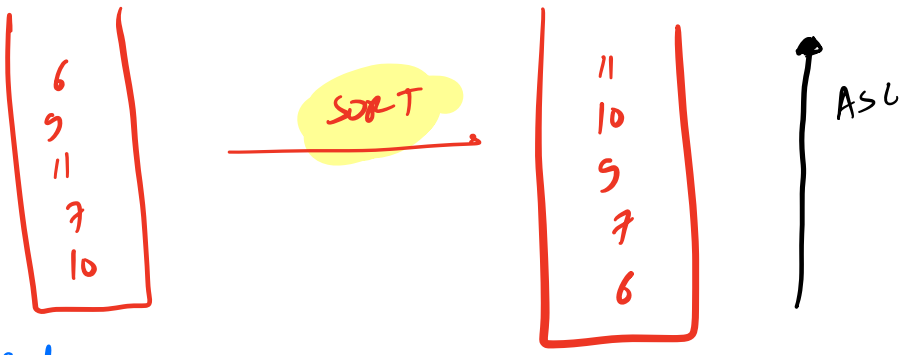




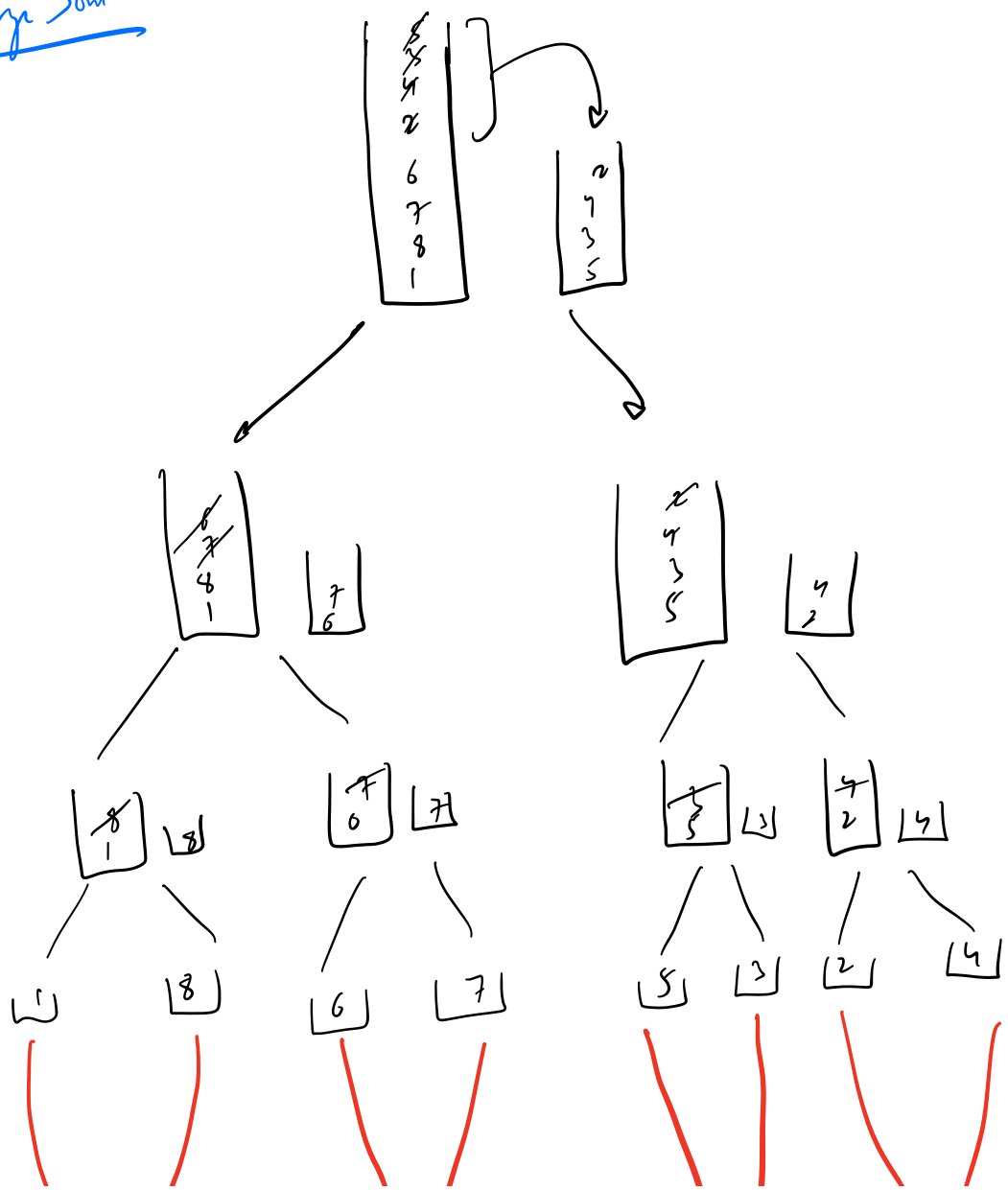

Nth ?

```
stack<int> merge(stack<int> s1, stack<int> s2) {
    stack<int> s;
    while (!s1.isEmpty() && !s2.isEmpty()) {
        if (s1.top() > s2.top()) {
            s.push(s1.top());
            s1.pop();
        }
        else {
            s.push(s2.top());
            s2.pop();
        }
    }
    while (!s1.isEmpty()) {
        s.push(s1.top());
        s1.pop();
    }
    while (!s2.isEmpty()) {
        s.push(s2.top());
        s2.pop();
    }
    while (!s.isEmpty()) {
        s1.push(s.top());
        s.pop();
    }
    return s1;
}
```

Given a stack! Sort it!

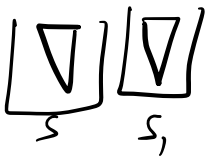


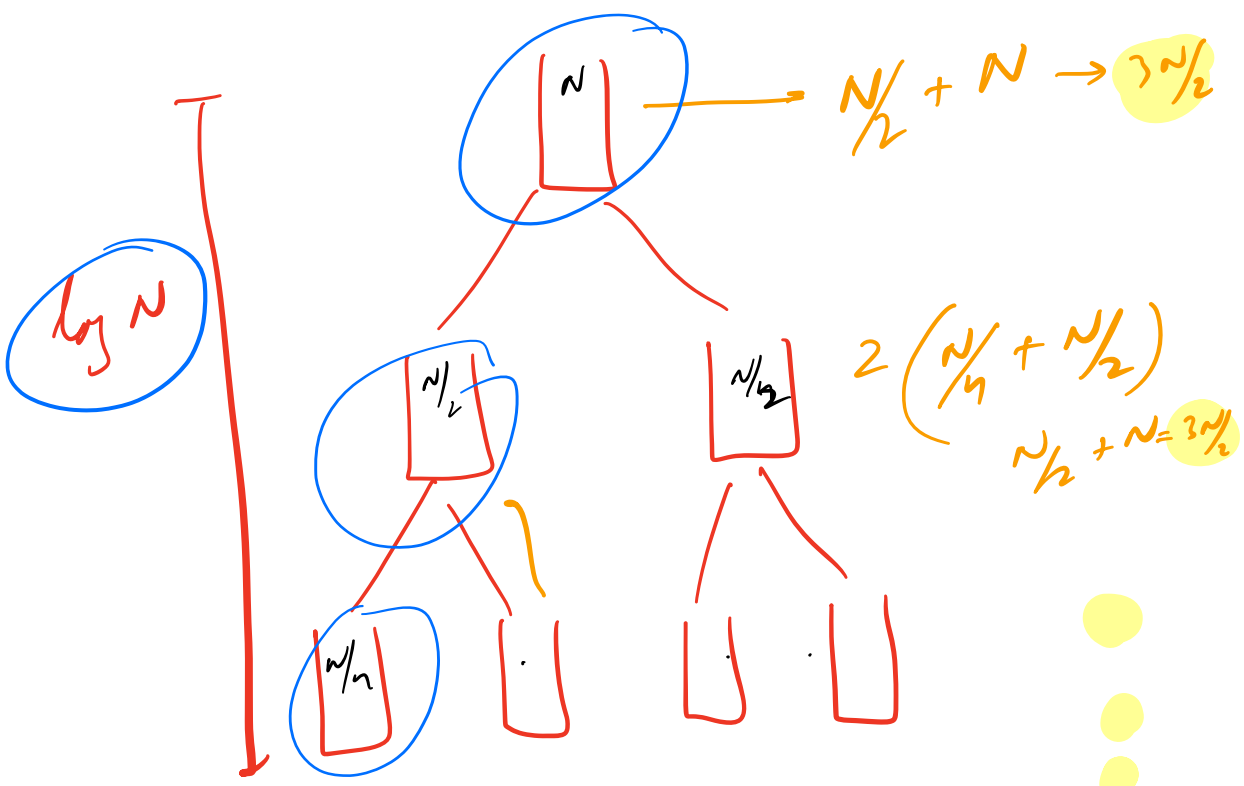
Merge Sort



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

```
stack<int> mergeSort (stack<int> s) {
    if(s.size() <= 1) ret s;
    stack<int> s1;        s2 = s.size()
    for(i=0; i< s2/2 ; i++) {
        s1.push(s.top());
        s.pop();
    }
    s = mergeSort(s);
    s1 = mergeSort(s1);
    ret merge(s, s1);
}
```





$TC : O(N \lg(N))$

$SC : O(\lg N)$