Stacks

Last In First Out
(LIFO)

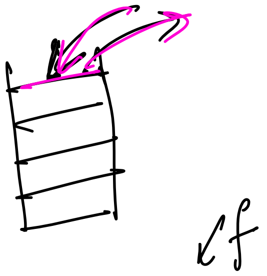
push(x)
pop()
size()
peek() / top()
→ O(1)

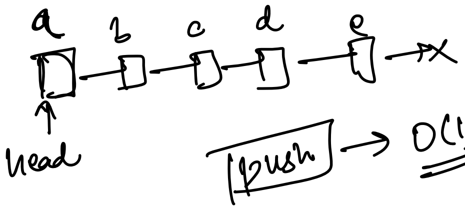
Applications :- undo / redo
recursion
back button

Implementation

arrays

linked list





push → O(1).

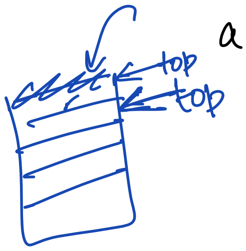
f → a → b → c → d → e → x

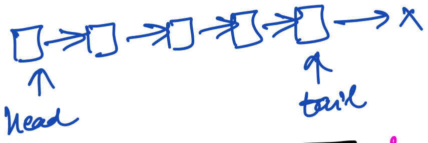
head

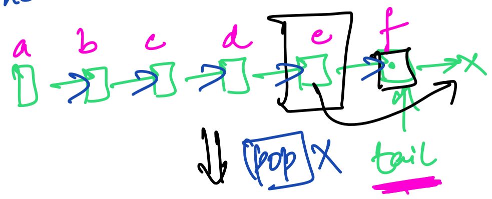
⇓ O(1).

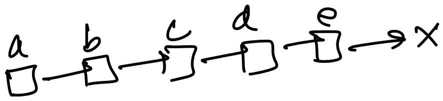
a → b → c → d → e → NULL

head = head.next









Q. Given a string, remove every consecutive duplicates 'until' there are no consecutive duplicates

~~aa~~.

abax

I/P:- a c ~~bb~~ c k

a ~~cc~~ k

ak ✓

(II) ~~aa~~ab → ab ✓

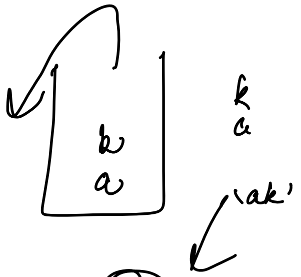
(III) abc~~kk~~cbam

ab~~cc~~bam

a~~bb~~am

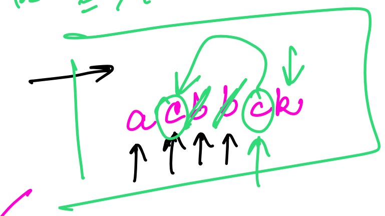
~~aa~~m

m ✓

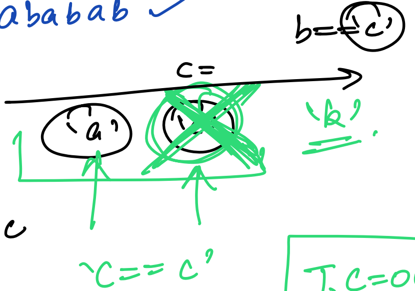


(IV) ababab → ababab ✓

Var lastremembered = ~~a~~ ~~c~~ 'b'



ac



Stack ✓

T.C = O(N)
S.C = O(N)

ac~~bb~~cka

a~~cc~~ka

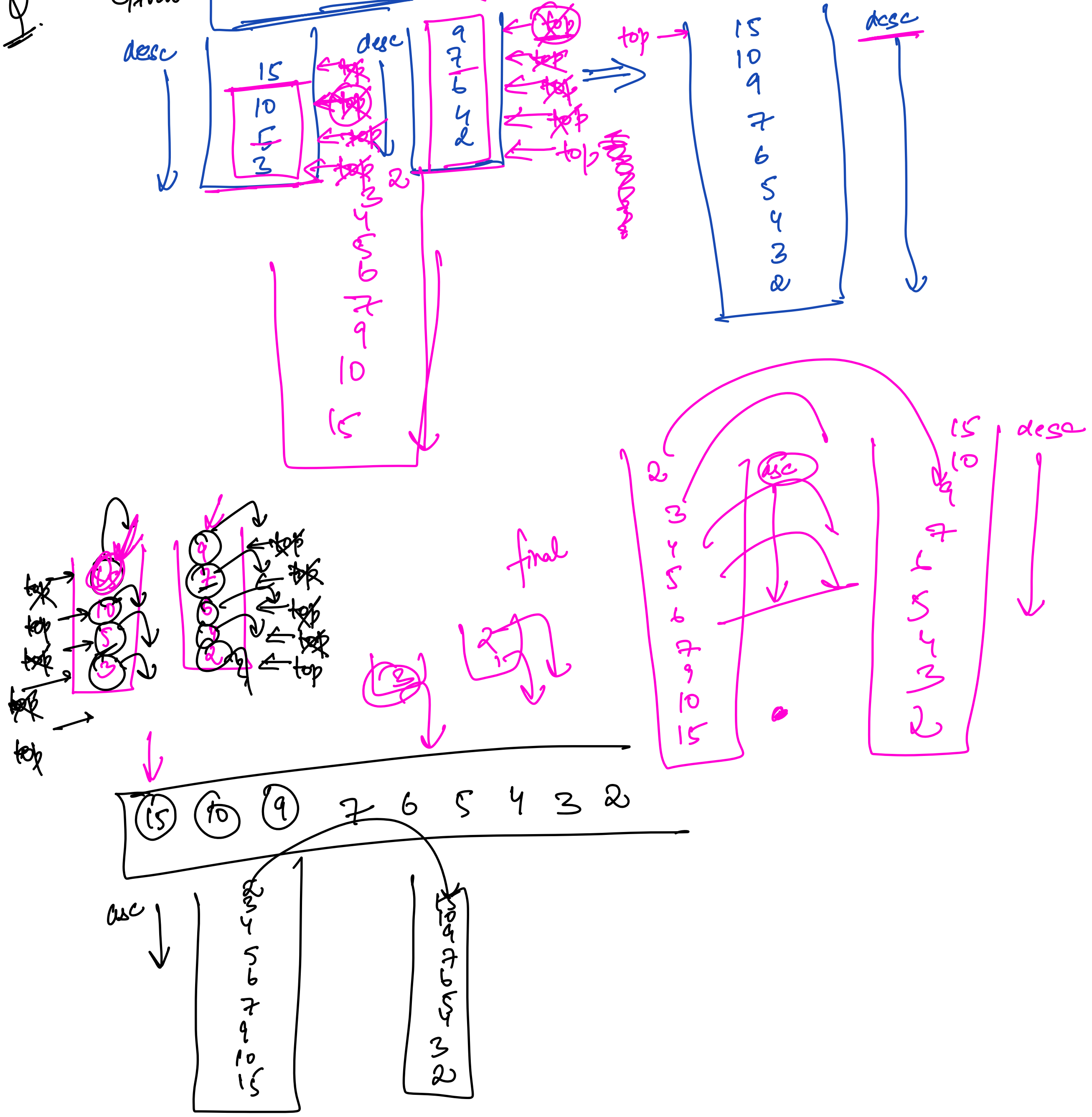
aka ✓

* Remove all consecutive duplicates.

b~~aaa~~bbc → c

~~bbb~~c

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



```
stack< int>   merge( stack<int> s1,   stack<int> s2) {

    stack<int> finalstack = new stack<int>();
    while( s1.size() >0  && s2. size >0) {
        if( s1. top() > s2. top()) {
            finalStack .push( s1. top())
              s1. pop() ;
        }
        else {
            final stack. push( s2. top())
              s2. pop();

        }
    }
    if( s1. size() ==0) {

        while( s2. size !=0) {
            final stack. push(s2. top())
              s2. pop();
        }
    }
    if( s2 size() ==0) {

        while( s1. size !=0) {
            final stack. push( s1. top())
              s1. pop();
        }

    }
}
```

this will have elements in sorted order but asc'

stack.pop()

H.W.

Reverse a stack using Recursion

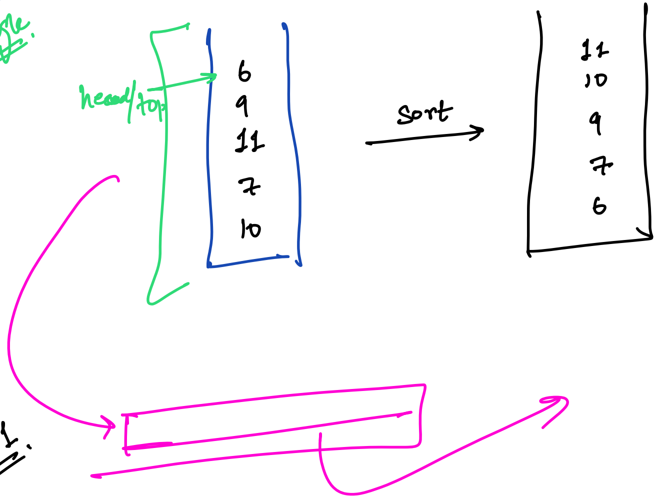
```
stack< int>  reverse( stack<int>  stack) {
    stack<int>  reversedStack = new stack<int>();
    while( stack. size() !=0) {
            reverseStack. push( stack. top());
            stack . pop();
    }
}
```

Q. Given a stack. Sort it in desc order.

Google.

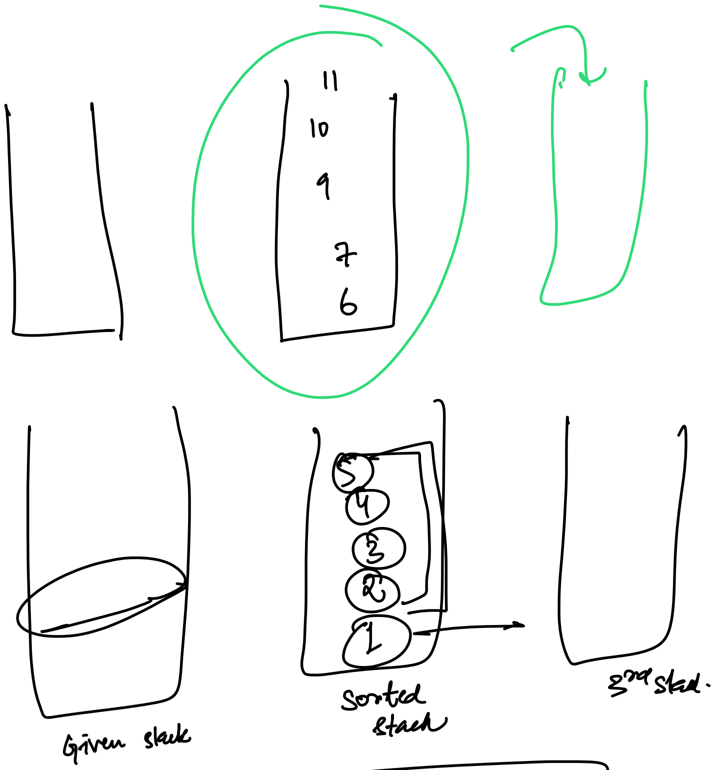
you're only allowed to use stacks

desc



A1.

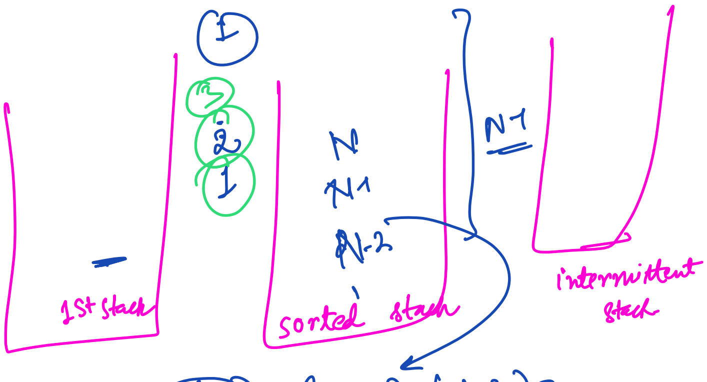
A2



$$(N-1) + (N-2) + (N-3) + \dots 1$$

$$\frac{(N)(N+1)}{2} = \frac{(N-1)\,(N)}{2}$$

$$O(N^2)$$

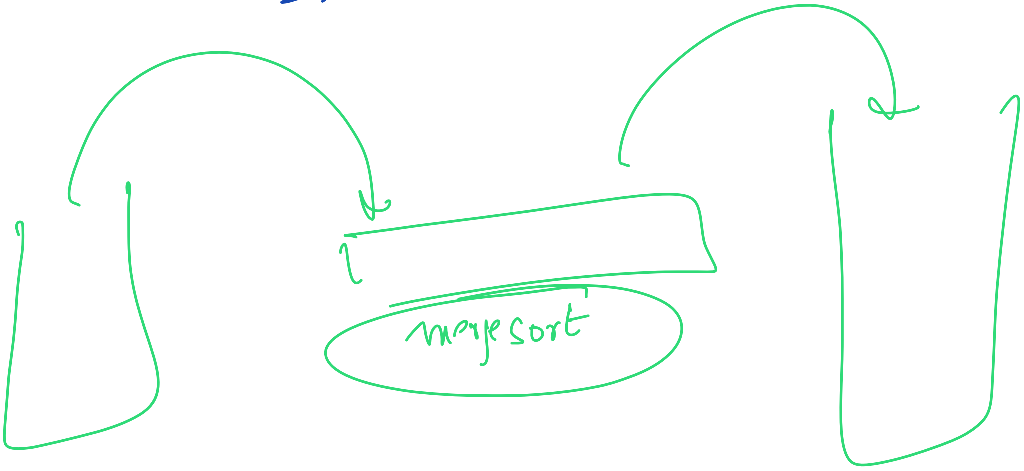


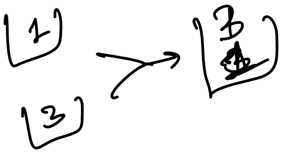
$\alpha((N-1))\quad (N-2)\alpha(N-3)2$

$\alpha[N^2]$

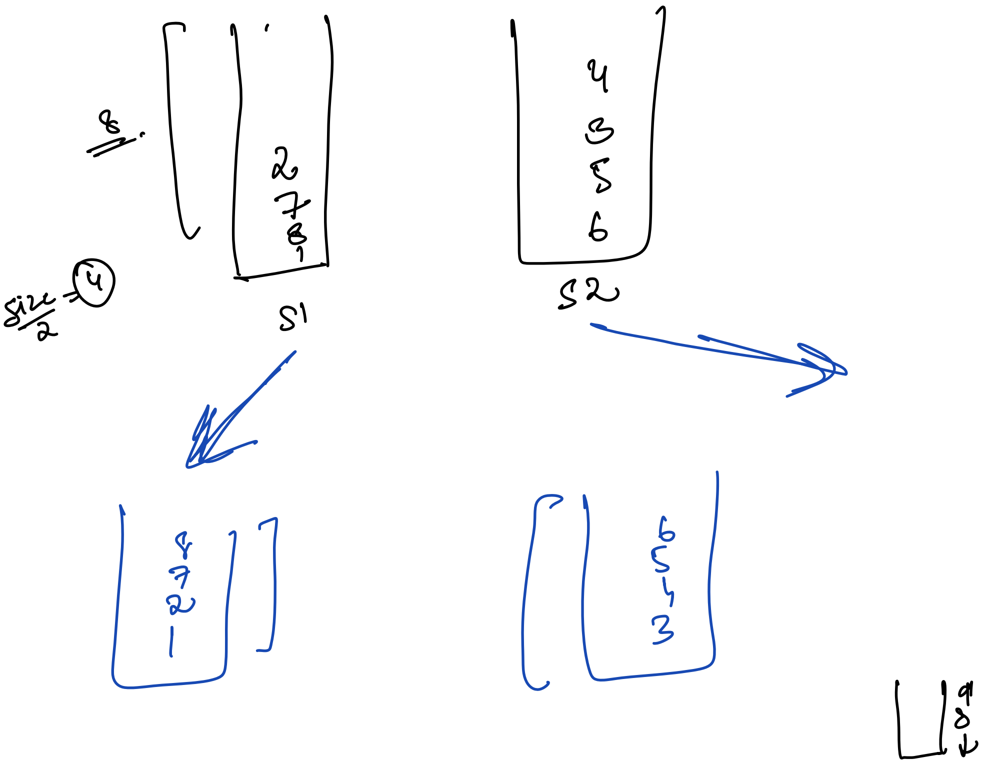
→ $O(N^2)$

$0+1+2+\cdots\ N-1$





```
mergeSort(arrn) {
    p1 -> first half
    p2 -> second half
 -> p1 = mergeSort(p1)
 -> p2 = mergeSort(p2)
         merge(p1, p2)
}
```

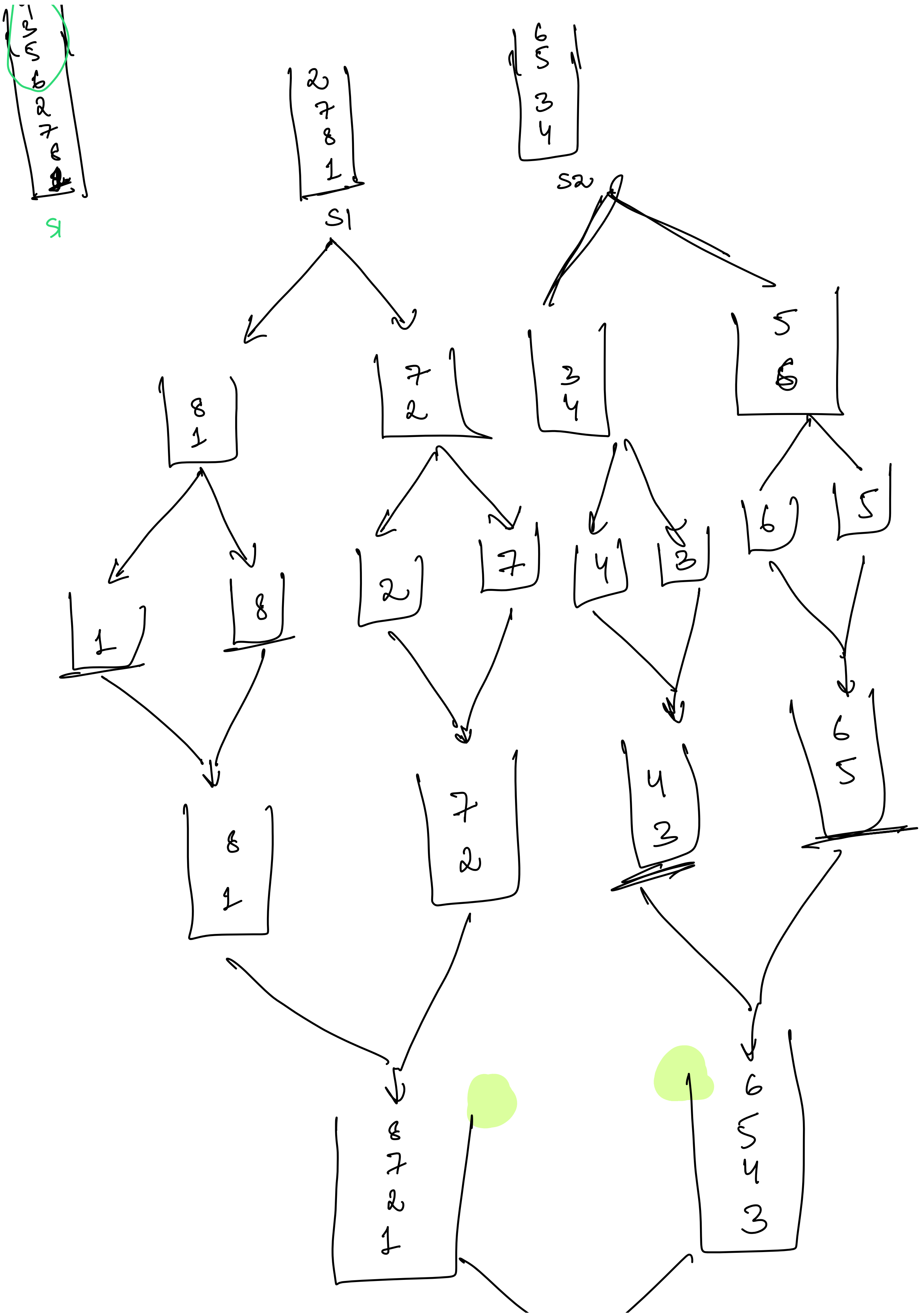


```
stack<int> mergeSort( stack<int> s1){
    if( s1. size() == 1) return s1;
    stack<int> s2= new stack<int> ();
    int n= s1.size()/2 ;   4
    for( int i=0; i < n; i++) {
        s2.push( s1.top());
        s1.pop();
    }
    s1= mergeSort(s1);
    s2= mergeSort(s2);
    return merge( s1,s2)
}
```

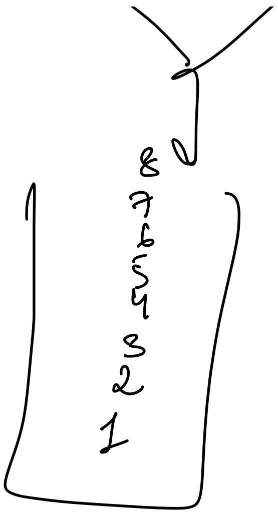
→ assume that these fn calls get the work done.

$T(N) = 2T(N/2) + O(N)$

$TC = n\log n$


3 5 6 2 7 8 1
2 7 8 1
S1
6 5 3 4
S2
8 1
7 2
3 4
5 6
1
8
2
7
4
3
6
5
8 1
7 2
4 3
6 5
8 7 2 1
6 5 4 3

$T.C = O(n\log n)$



$7 \times 1 + 2 - 8 \times 3 + 10/5 = \boxed{2}$

-13

$7 \times 1 = \boxed{7}$ $7 + 2 = 9 - 8$

$1 \times 3 = 3 + 10 = 13/5$

( )

/

*

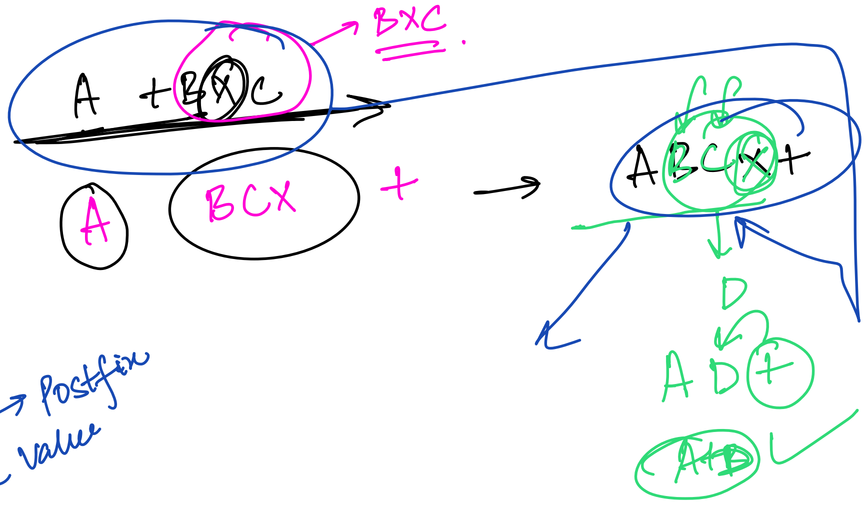
+ −

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

$7 + 2 - 24 + 2$

$11 - 24 = \boxed{-13}$

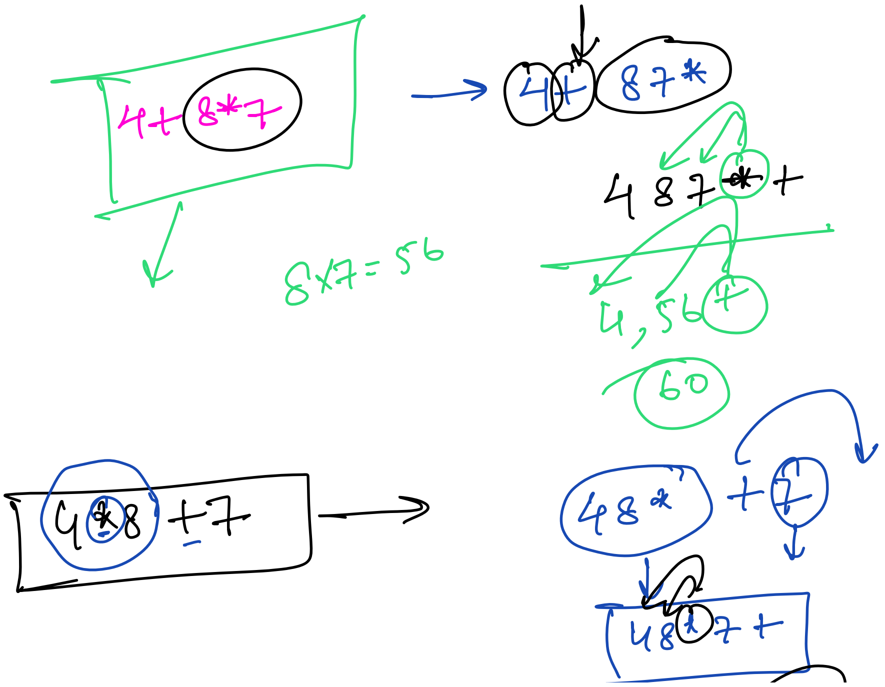
Infix Notations	Postfix Notations
A × B	AB*
A ÷ B	AB÷
A + B	AB+
A − B	AB−



* ~~Pos~~ Infix → Postfix

* Calculate value

(I)



32, 7 +

39 ✓

10 + 12 - 7 = 22-7 = 15 ✓

(II) 10 + 3 * 4 - 7

10 + 3,4* - 7

10, 3,4*+ - 7

10,3,4*+7 -

3x4 = 12

10, 12 + 7 -

22, 7 - = 22-7 = 15

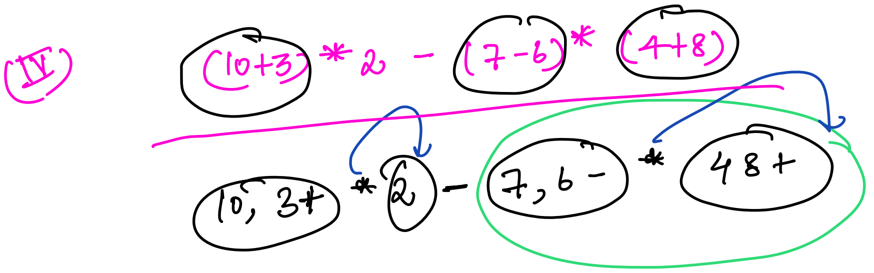
(III) 10/(4-2) * 6 + 9

10/42- * 6 + 9

10, 42- / * 6 + 9

10, 42-/6* + 9

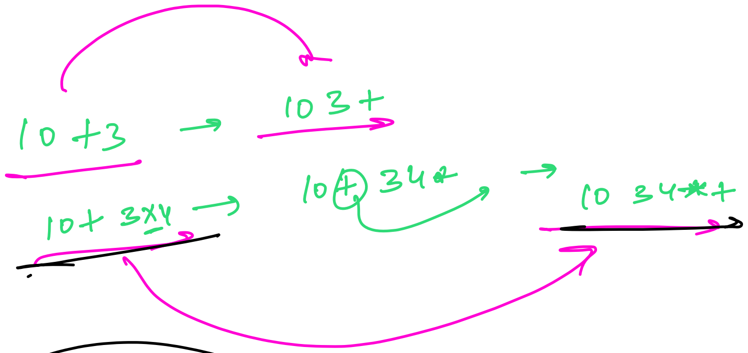
10, 4,2 -/ 6* 9 +

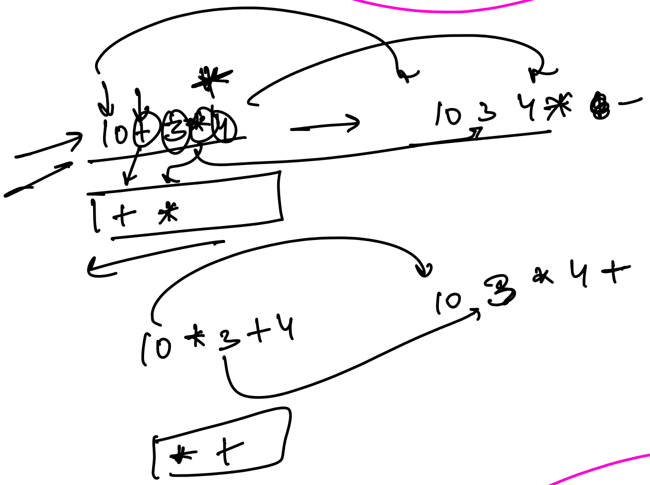


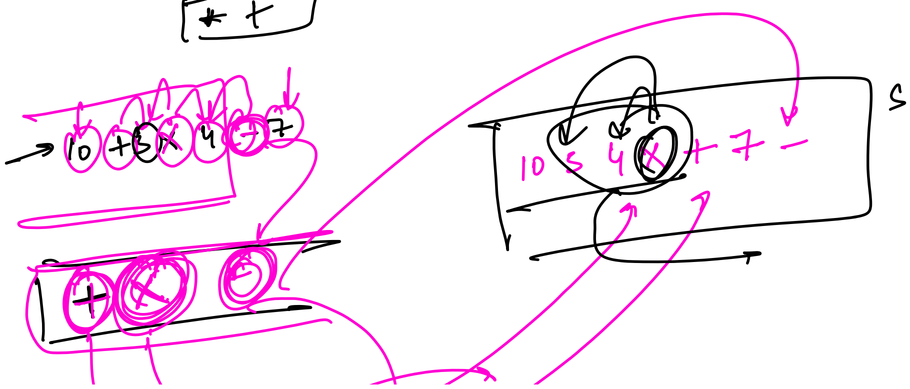
7,6-4,8+*

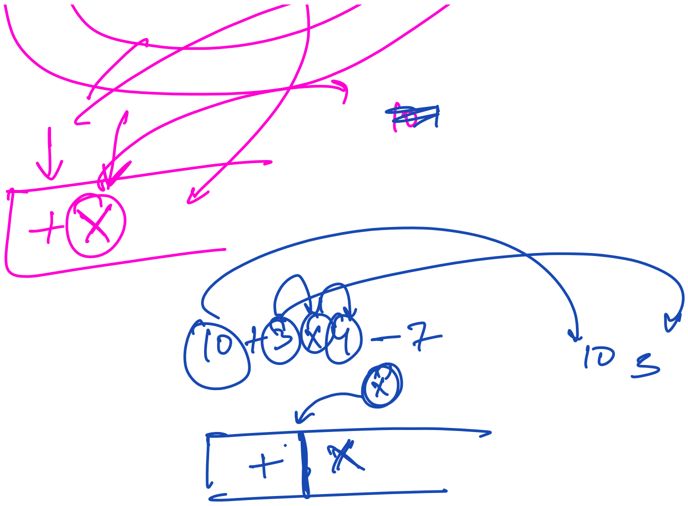
10,3+2*

-









while