Validate Stack Sequences

Saturday, 4 December 2021 4:34 PM

push → {1 2 3 4 5}

popped → {4 5 3 2 1}

is it possible or not

push → [1, 2, 3, 4, 5] ↑

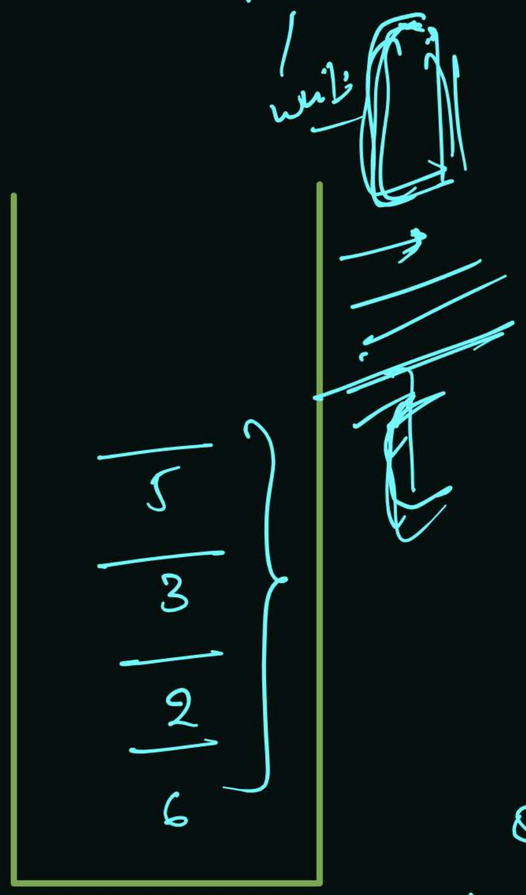
Pop → [4 3 5 1 2]

[4 3 5 2 1]

Not possible, we can't achieve this order.

push

for(i=1; i<=n; i++)



push → { 6 2 3 4 5 } [ ]

push

st.size() == 0

popped → { 4 6 3 2 1 }

st.size() == 0, || i == End || j == End

↳ Return true

pop

Space

O(n)

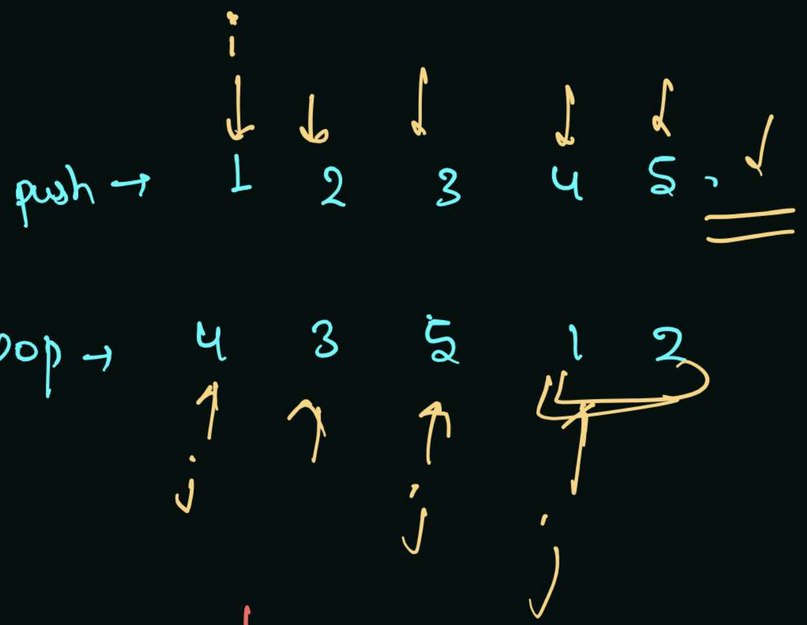
O(1)

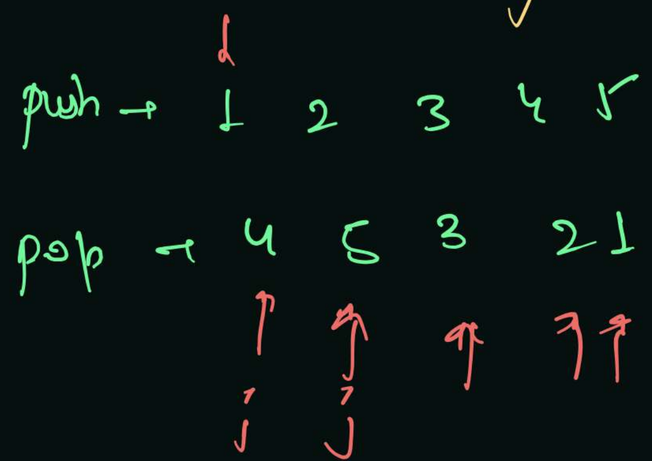
Amortized

O(1)

complexity

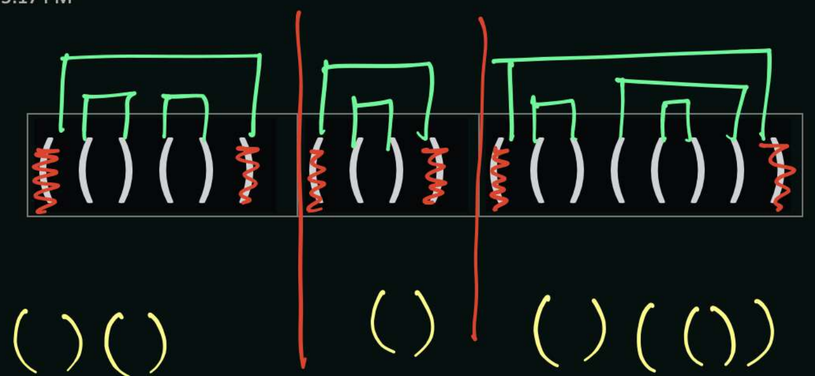
```java
public boolean validateStackSequences(int[] pushed, int[] popped) {
    int j = 0;
    Stack<Integer> st = new Stack<>();

    for(int i = 0; i < pushed.length; i++) {
        st.push(pushed[i]);
        while(st.size() > 0 && st.peek() == popped[j]) {
            st.pop();
            j++;
        }
    }
    return st.size() == 0;
}
```





Remove Outermost Parenthesis

Saturday, 4 December 2021 5:17 PM



op = ((( ) ( )) (( )))

cl = ((( ) ( )) (( )))

st index = 0 to

( ) ( ) ( ) ( ) (( ))

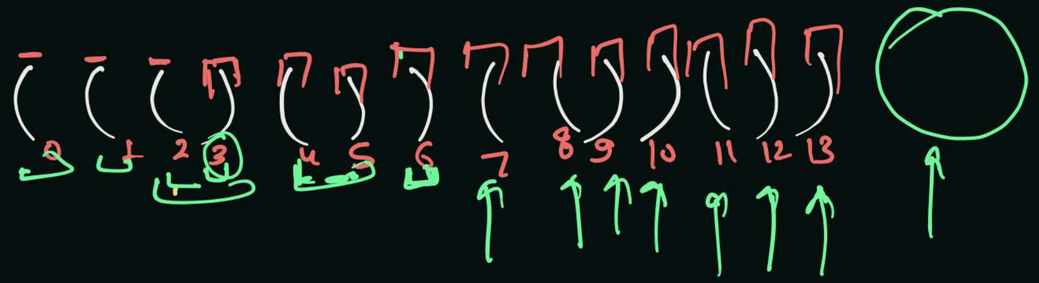
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

( ) ( ) ( ) ( ) (( ))

Score of Parenthesis

Saturday, 4 December 2021 6:28 PM

()()



() → 1

A + B → A+B

()() → 2

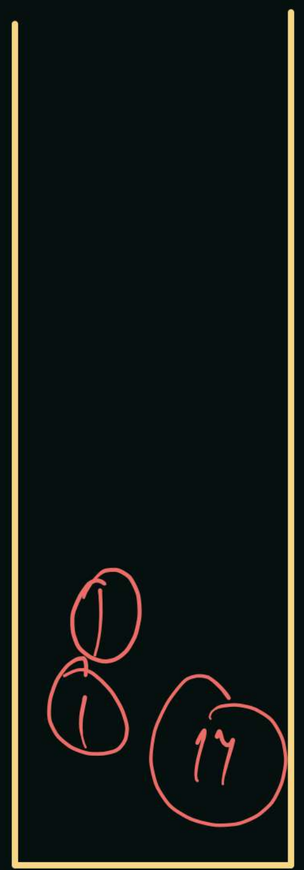
(A) → 2*A

(()) → (1) → 2

Marker → '(' → -1

closing - to solve

Score + Integer



→ Result

opening bracket

↳ push -1 marker for opening br

closing bracket

↳ top - opening - pop & push

make sum until opening is not encounter

at the end push 2* sum

Reverse String between parenthesis

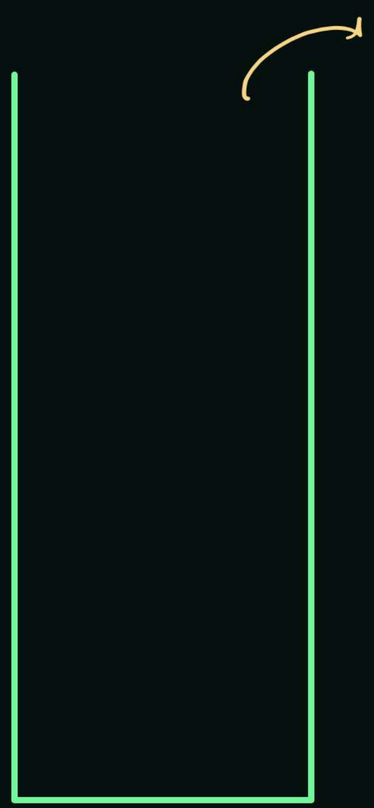
Saturday, 4 December 2021 7:06 PM

(ed(et(oadc))elf)(redoc)run

Opening bracket ||

↳ character ≡ String.
push in stack

Closing bracket



h:
u
r
coder
fleetcdaode

At the end. → concater with Reversed proper R
reverse once again.

nurredocedoadcteelf

fleetcdaodecoderrun

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
(	e	d	(	e	t	(	o	a	d	c	)	)	e	l	f	)	(	r	e	d	o	c	)	r	u	n

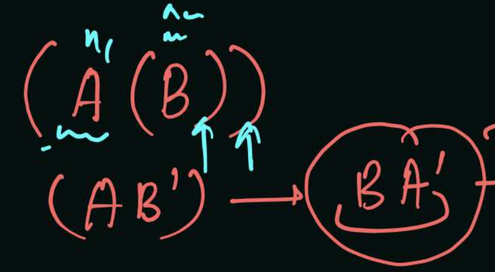
o d e r r u n

f l e e t c d a o d c c o d e r r u n

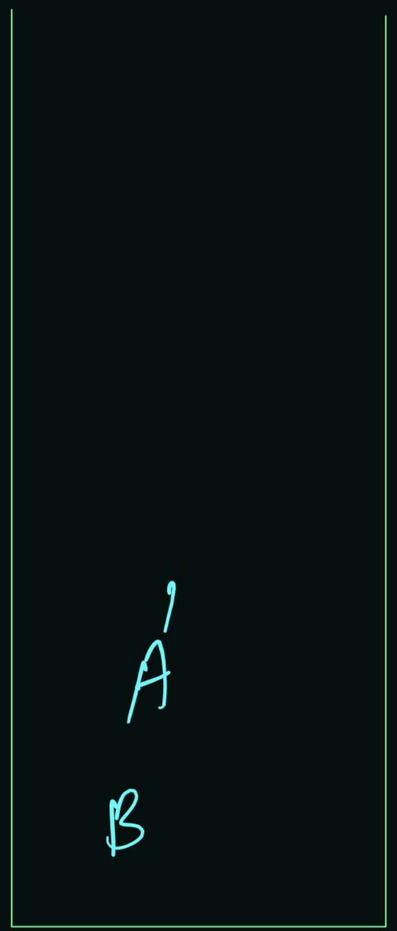
n u r r e d o c c d o a d c t e e l f

$(A(B)) \rightarrow (A\,B') \rightarrow BA'$

$(ab(cd)) \rightarrow (abdc) \rightarrow cd\,ba$



A & B are Strings.



$(AB')' \Rightarrow BA'$

④ At then end pop from Stack & concatenate in String.

⑤ Reverse String.

Steps:

① If opening (char. encounter then add it in stack.

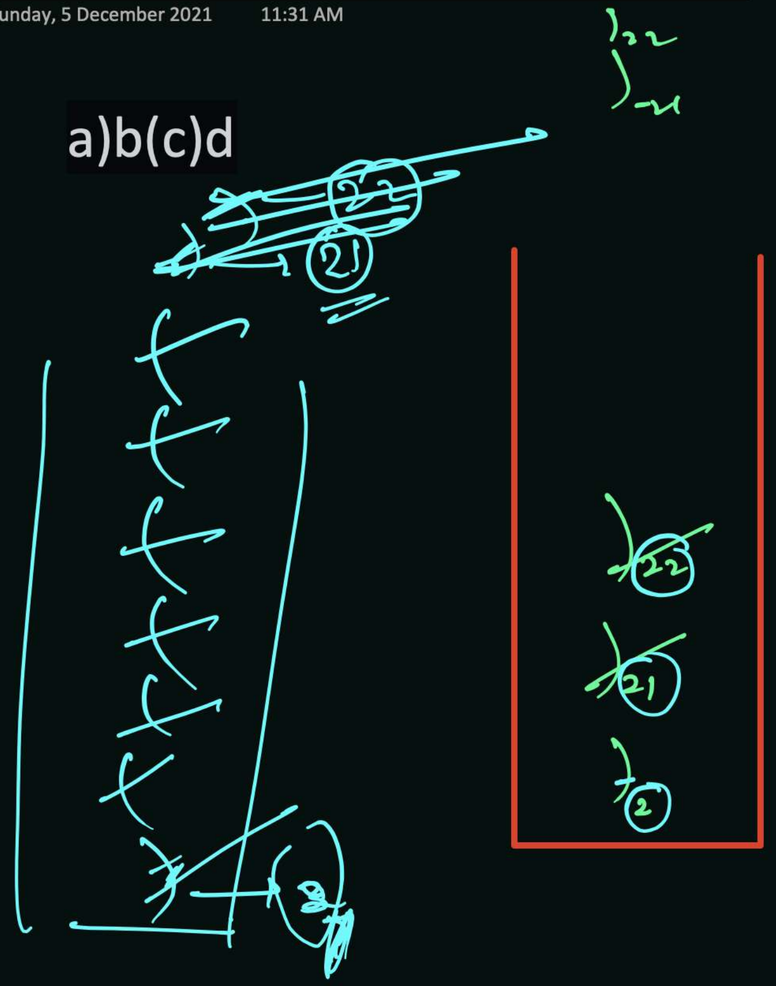
② If closing bracket encount, then add char in stringbuilder until opening is not encounter.

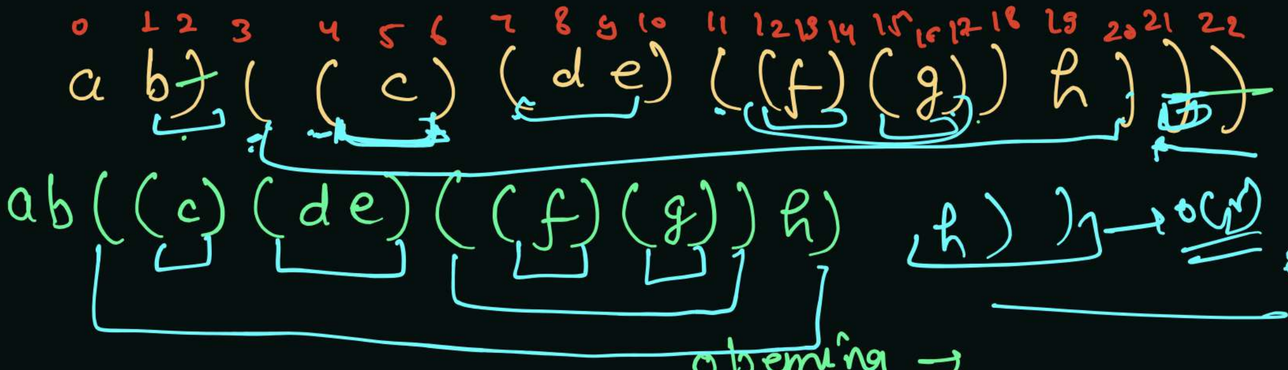
③ Push char in char from stringbuilder.

Min. Remove to make parenthesis Valid

Sunday, 5 December 2021 11:31 AM

a)b(c)d





ab((c)(de)((f)(g))h)) → O(n)

opening →

st.push.

reverse O(n)

O(n)

lee(t(c)o)de)

StringBuilder → O(1)

ArrayList str = str + something

str = something + str

O(n)

closing →

st.size == 0 || st.top == ')'

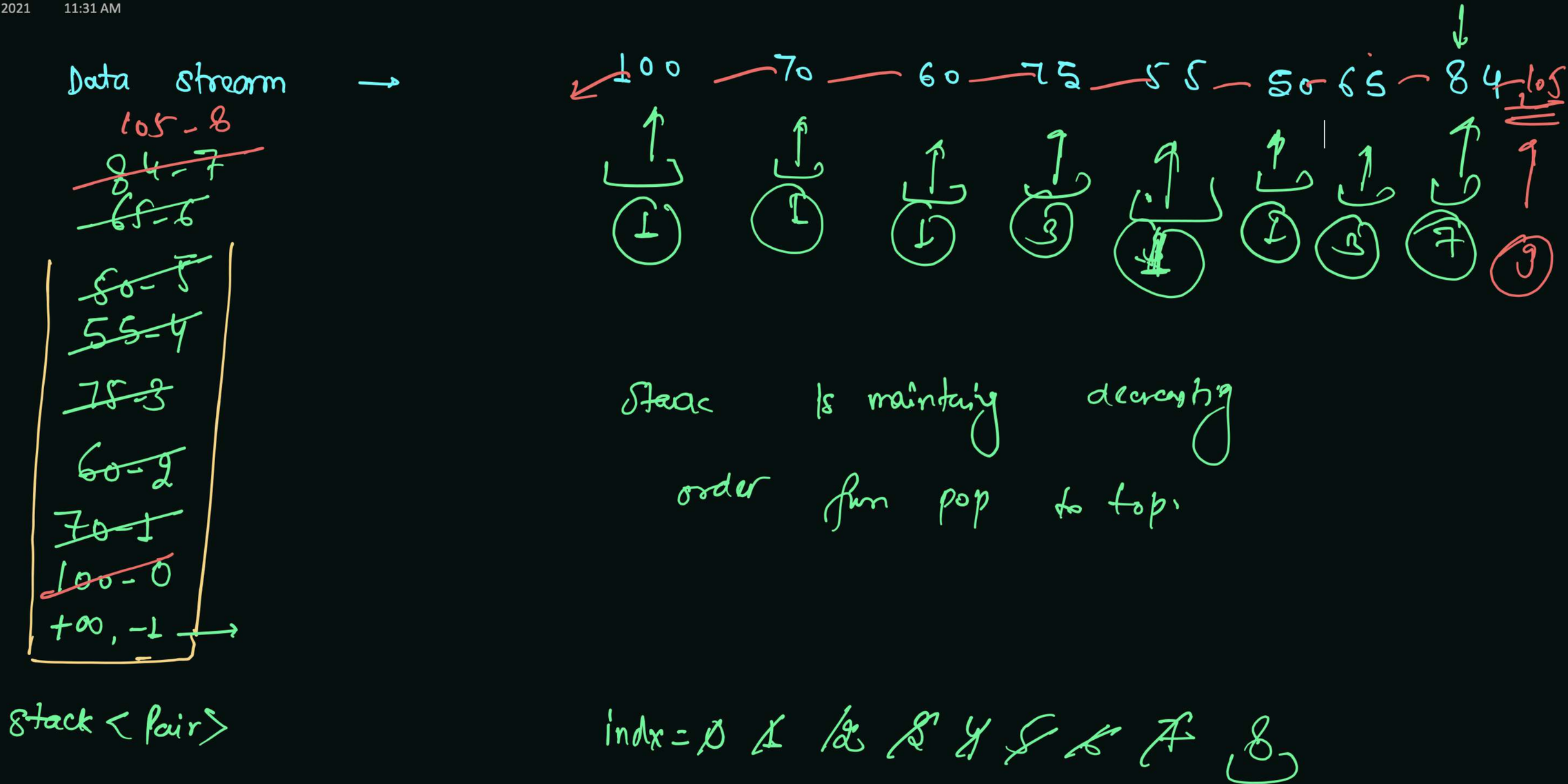
st.push(y);

if st.top == '('

pop

Online stock span

Sunday, 5 December 2021 11:31 AM



Daily Temperature

Sunday, 5 December 2021 11:31 AM

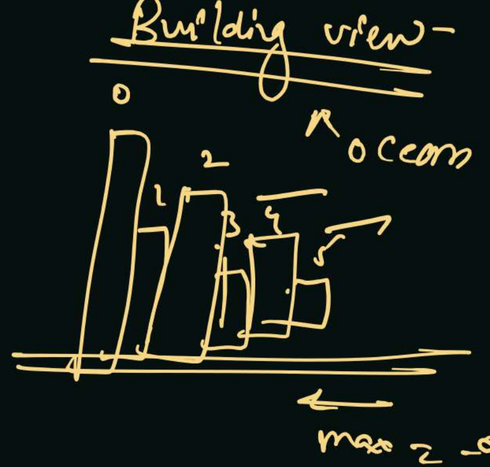
temperatures = [73,74,75,71,69,72,76,73]

[1, 1, 4, 2, 1, 1, 0, 0]

Next greater Index

→ 1 2 6 5 5 6 8 8

[1 1 4 2 1 1 0 0]



[0, 2, 4, 5]

from rear

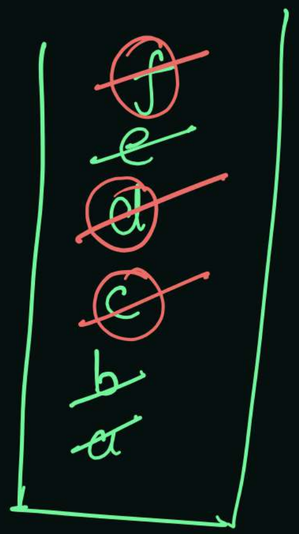
[5, 4, 2, 0]

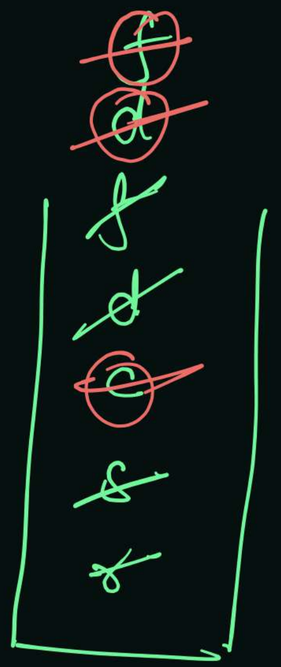
Backspace String Compare

Sunday, 5 December 2021 11:32 AM

String s1 = a b # # c d e # f

String s2 = r s # # c d f # # d f





https://leetcode.com/problems/simplify-path/

https://leetcode.com/problems/minimum-insertions-to-balance-a-parentheses-string/

leet code 301 - Recursion

Evaluation

Tuesday, 7 December 2021 9:11 PM

1. Infix Evaluation — Basic Calc - I
2. Prefix Evaluation — Basic Calc - II
3. Postfix Evaluation — Basic Calc - III

operator operand

operation

arithmetic operation

$[A + B] = C$

A & B are operand

&

'+' is an operator

BDMAS

Brackets Divide Multiplication Addition Subtraction

① $2+3+\underbrace{7*4}_{28} = 5+28 = 33$

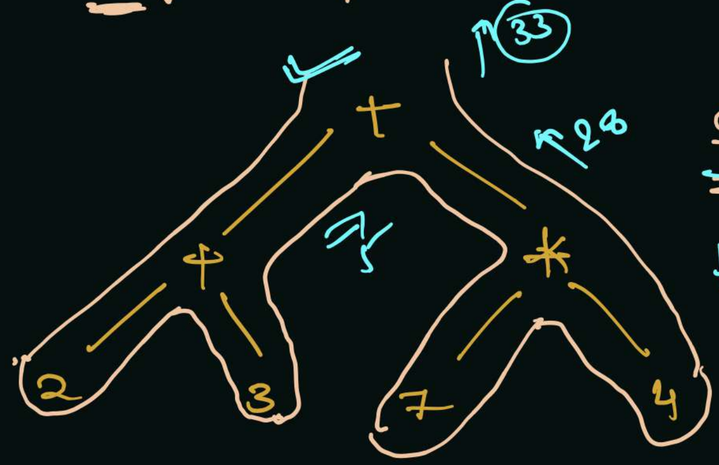
② $2+3+7*4$

$5+7*4$

$12*4 = 48$

Expression have some set of rules

Infix Expression



Infix Exp.

In order → $[2+3+7*4]$

preorder → $++23*74$ → prefix Exp.

postorder → $23+74*+$ → postfix Exp.

B → Brackets

D → Divide
M → Multiplication
} → Equal

A → Addition
S → Subtraction
} → Equal

Agenda:

Set-I.
- *Infix Evaluation.
- Infix to prefix
- Infix to postfix

Set-II
- *prefix Evaluation
- prefix to Infix
- prefix to postfix

Set-III
- postfix Evaluation
- postfix to prefix
- postfix to Infix

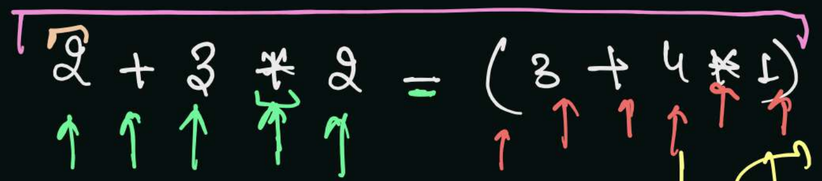
How to Manage priority of operation. →

HashMap:

B →	3 * we deal with brackets in diff way
/ →	2
* →	2
+ →	1
− →	1

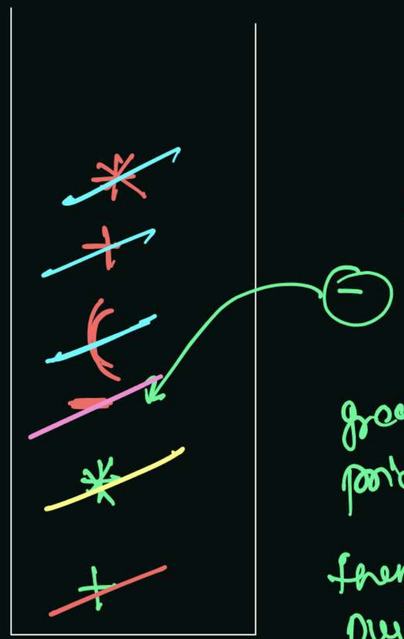
Infix Evaluation:

(Single Digit Number)



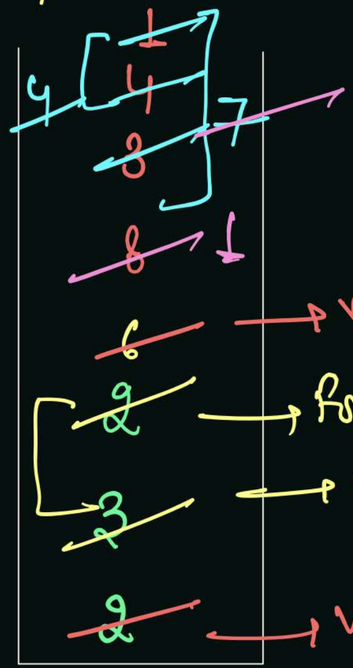
Result = 1

val1 operator val2 → val2 val1 (val2 is at top than val1)



if current operator's priority is greater than top's then push

smaller than or equal to already filled operator



Final Result

operator = * +

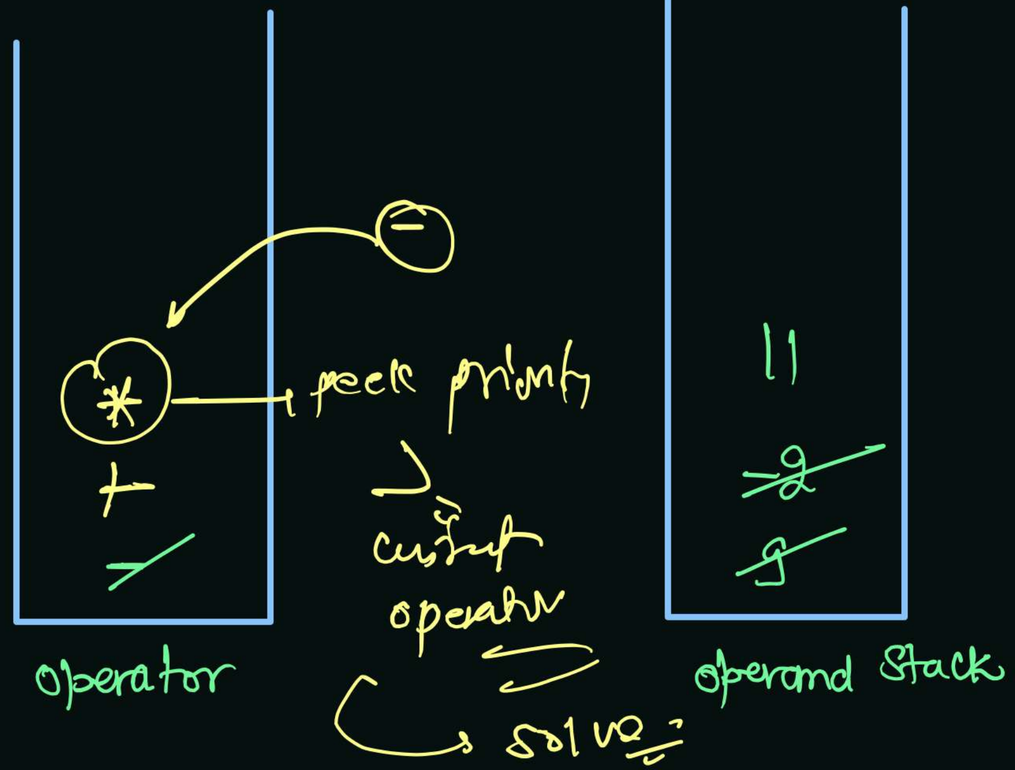
val2 = 2 6

val1 = 2 2

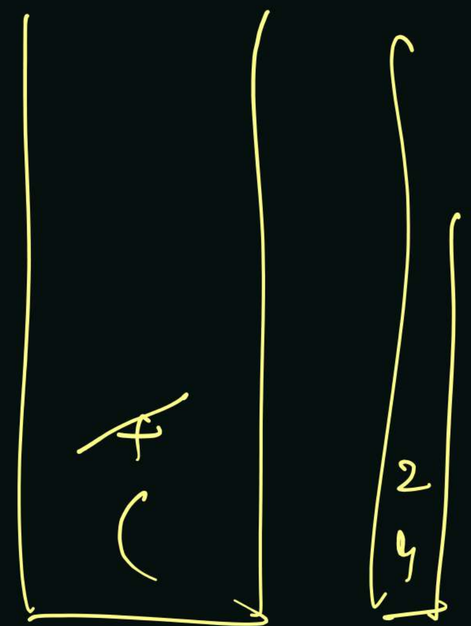
result = 6 8

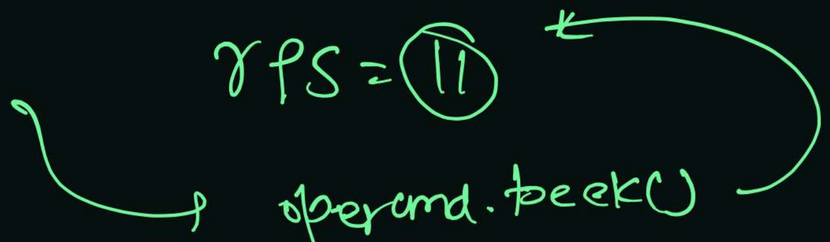
operator =	*	+	—
val2 =	4	4	7
val1 =	3	3	8
res =	4	7	1

Infix Expression → 7 = 3 * 2 + 8 - (4 + 2 + 1 * 6 - (7 * 2))



op =	~~*~~	/	-
v2 =	~~2~~	14	-2
v1 =	~~7~~	12	9
res =	14	~~-2~~	11





i=0 ton ⇒ operand ⟶

operandStack.push(val);

operator ⟶

op == '(' ⟶ push

stack.size() == 0 ⟶ push.

priority (stack.peek()) < priority (op) ⟶ push.

op.

vice versa pop & solve until priority (stack.peek()) ≥ priority (op)

↳ operand ⟶ val2, val1

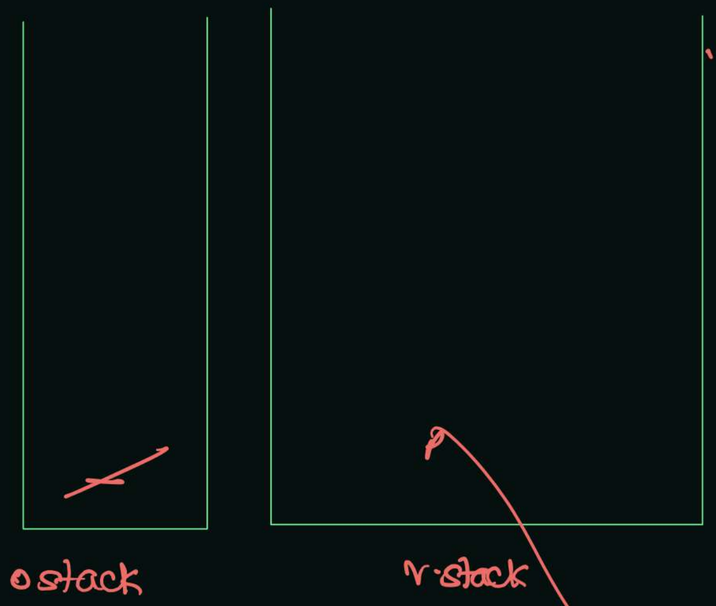
operator ⟵ op.

Res = val1 operator val2 ⟶ push in operand.

↳ closing bracket ↳ coming until opening.

Infix to prefix ⟶

2 + 3 * 2 = (3 + 4 * 1)



op = ~~*~~ ~~+~~ −

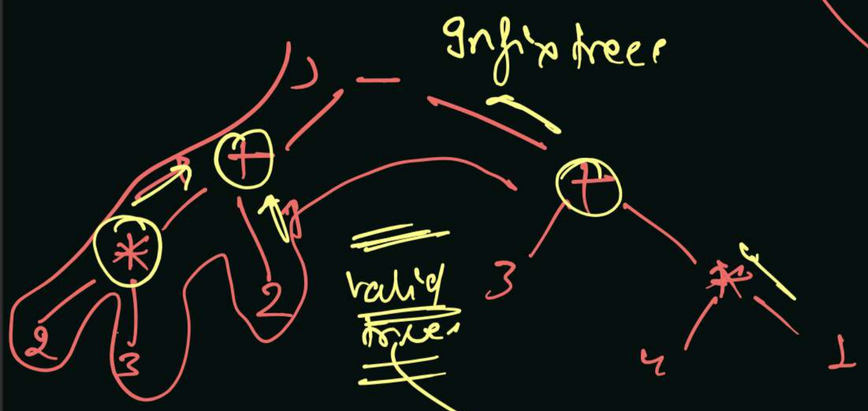
v2 = ~~1~~ * 41 + 3 * 41

v1 = ~~4~~ 3 - + * 2 3 2

prefix = op val1 val2

postfix = val1 val2 op

~~* 4 1~~ + 3 * 41

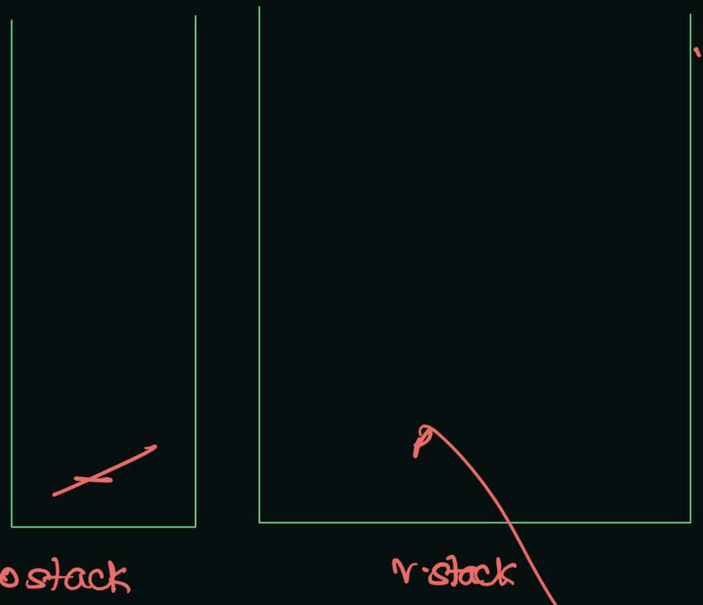


− + * 2 3 2 + 3 * 4 1] prefix pair

[prefix pair − + * 2 3 2 + 3 * 4 1

postfix 2 3 * 2 + 3 4 1 * + −

Infix to prefix ⟶ 2 + 3 * 2 = (3 + 4 * 1)



op = * +

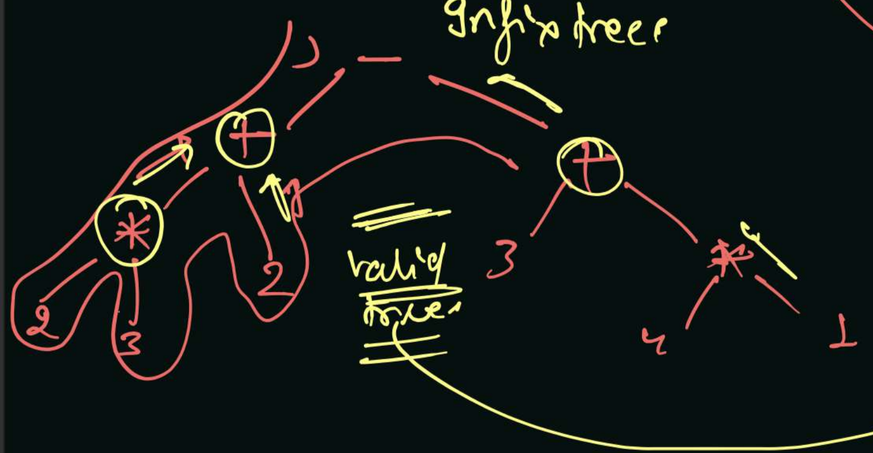
v2 = 2 * 41 + 3 * 41

v1 = 3 4 1 + * 232

prefix = op val1 val2

postfix = val1 val2 op

* 41 + 3 * 41



- + * 23 2 + 3 * 41 ← prefix pool

prefix post - + * 23 2 + 3 * 41

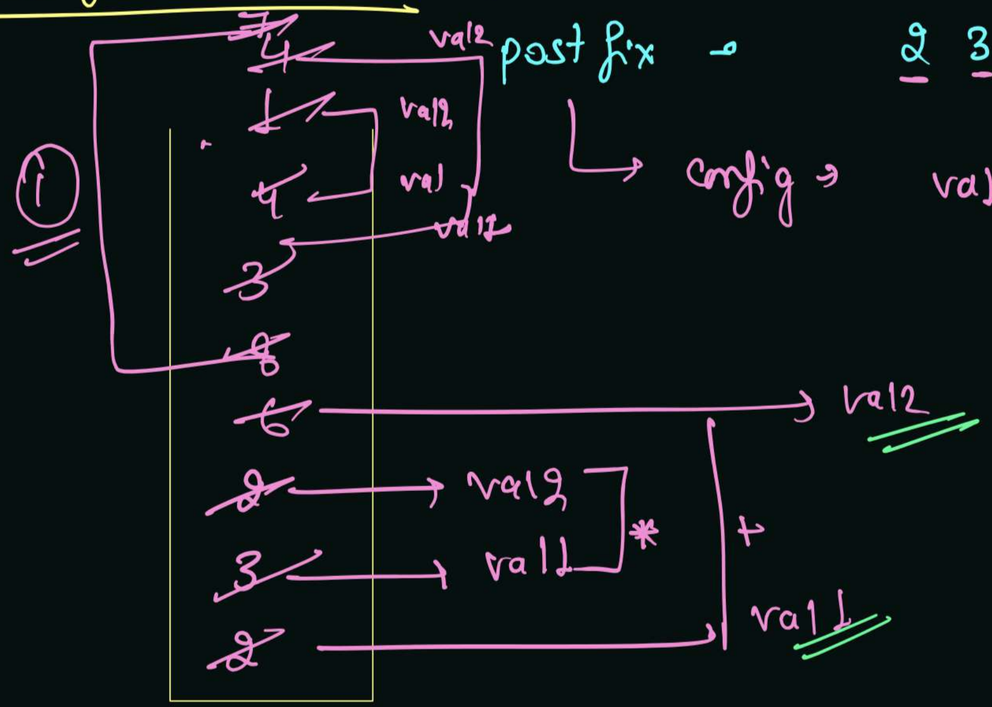
postfix 2 3 * 2 + 3 4 1 * + -

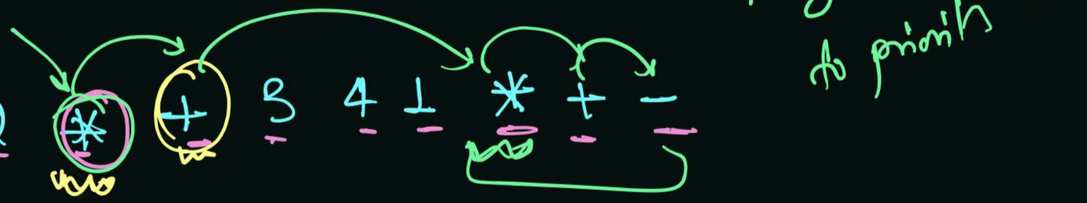
Diff. b/w infix prefix and postfix

operators are arranged Random

operators are arrange in priority order

Post fix Evaluation →



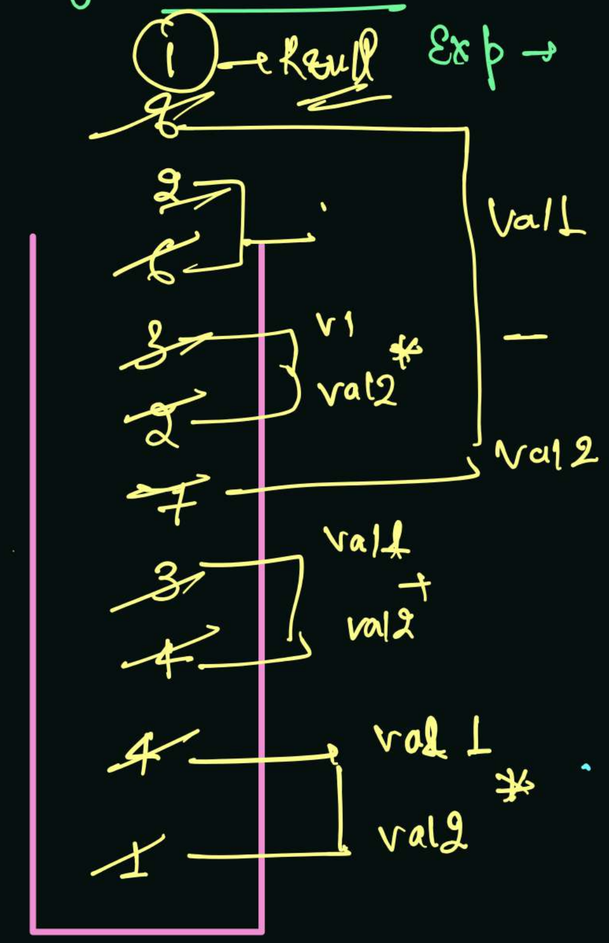


Evaluation

for infix → (val1 op val2)

for prefix → (op val1 val2)

Prefix Evaluation :→



op val1 val2

− + 2 * 3 2 + 3 * 4 1

Travel from n to 0

Prefix Generic → operator val1 val2

prefix to postfix

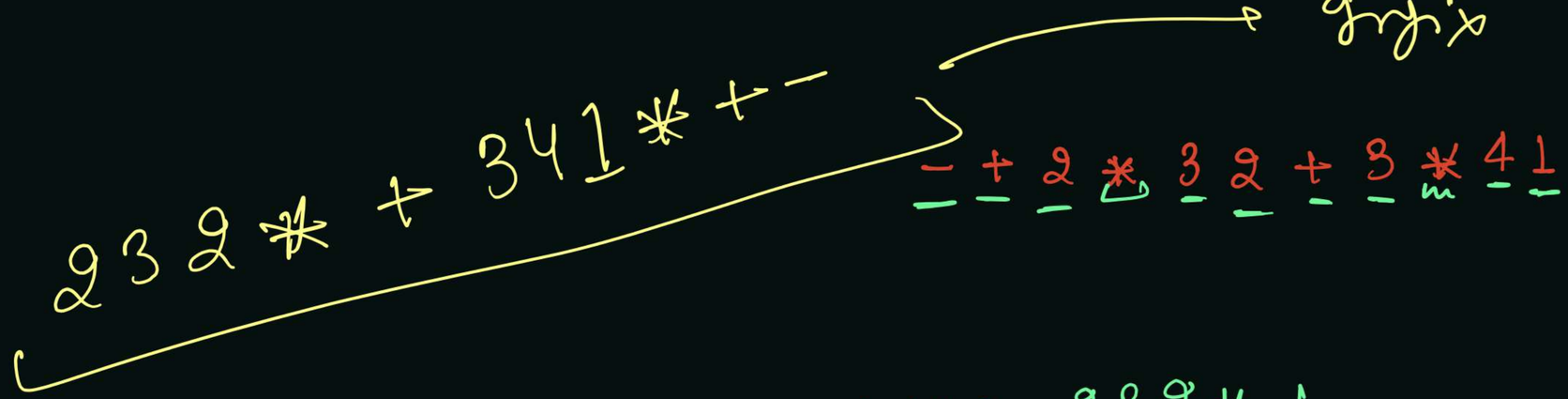
val1
val2
op

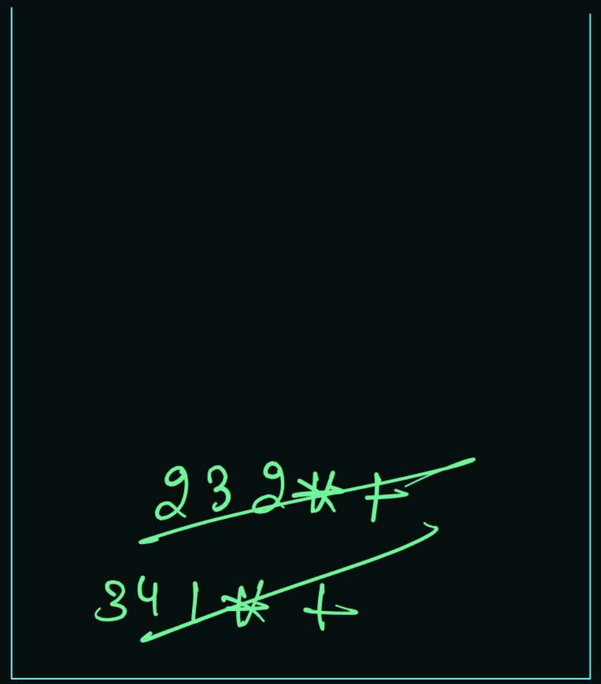
→ val1 val2 op

push in stack

prefix to infix →

→ (val1 op val2)





val1 = 232*+

val2 = 341*+

op = -

res = 232*+, 341*+, -

val1 val2 op

= + 2 * 3 2 + 3 * 4 1

$$\frac{((2+(3*2))-(3+(4*1)))}{(2+(3*2))-(3+(4*1))}$$

$$(-(3-4) = 7$$

Based on Evaluation

- Basic Calculator - I
- Basic calculator - II
- Basic calculator - III

} tough. Important question

↳ Excel Clone

formulate → stuck ↳ Excel

Exclusive Time of Function

Thursday, 9 December 2021 9:15 PM

logs

ID : status : timestamp

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Execution time.

0 → 2 sec.

1 → 4 sec.

n → no. of function (7)

0 → 6 - ID

logs →

ID: Start/End : timestamp

[0 : Start : 0
1 : Start : 2
1 : End : 5
0 : End : 6]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

0 → 6

1 → 4

n = 2 [i.e. 0 , 1]

0 : Start : 0

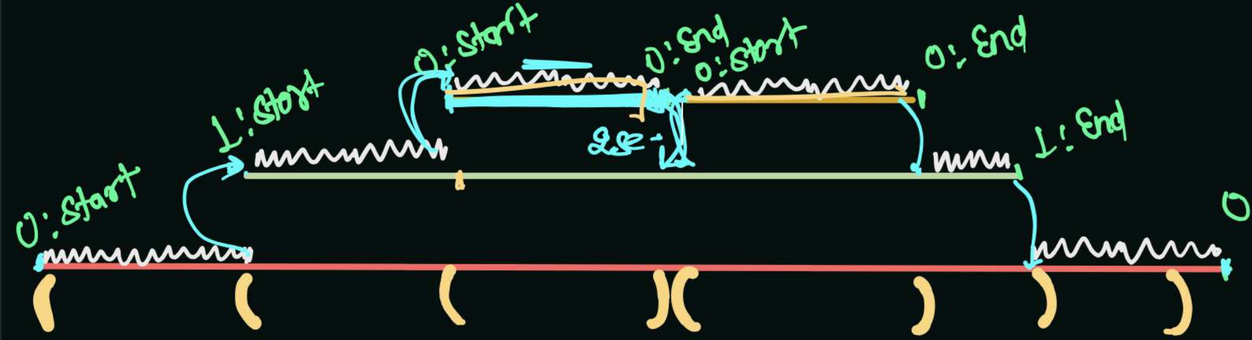
1 : Start : 3

1 : End : 4

1 : Start : 6

1 : End : 7

0 : End : 9



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

fn. diff. time
8 - 2 + 1 = 7

fn. Ex.
= 7 - 4 = 3

10 - 0 + 1 = 11sec.
14 - 7 = 4sec.

0: 0: 7

ID: Start : child Execution time.

function time diff = End time - Start time + 1
= 5 - 4 + 1 = 2 sec.

fn. Execution time = fn. time diff - child Exe. time.
= 2 - 0 = 2 sec.

fn. Ex. th = 2 - 0 = 2sec.

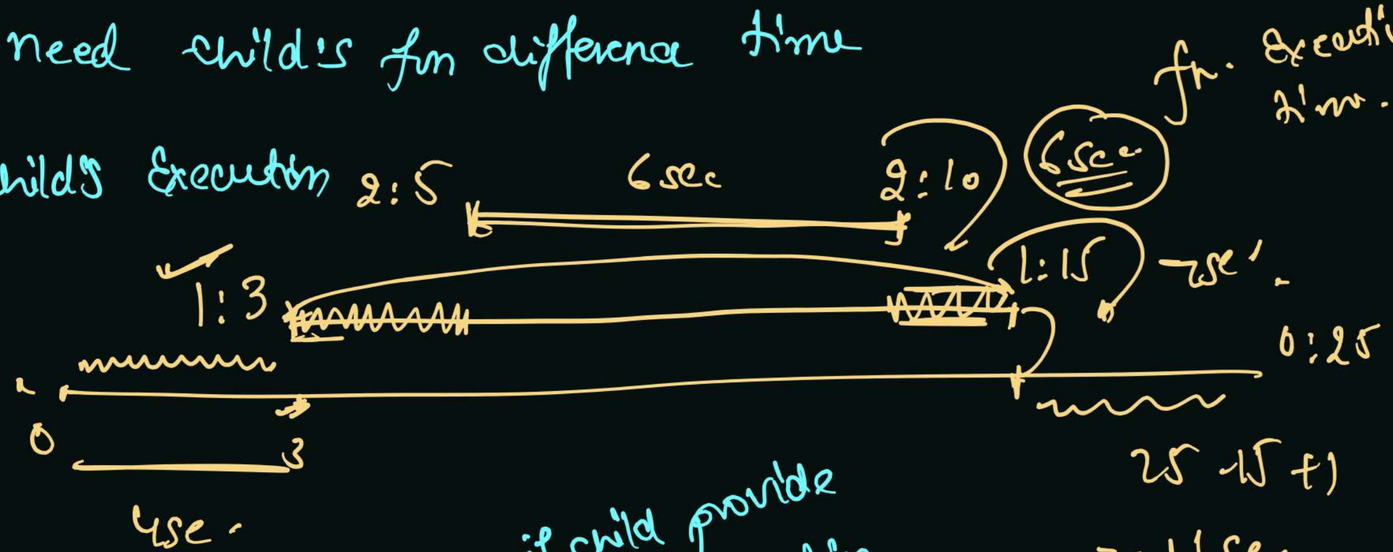
Hmap.

0 → 2 + 2 + 4 = 8
1 → 3 = 3

fn. time diff = 8 - 2 + 1 = 7sec.

0: Start : 0
1: start : 2
0: start : 4
0: End : 5
0: start :
0: End :
1 : End : 8
0 : End : 10

Why parent need child's fn difference time rather than child's Execution time.



0 → $\underbrace{25-0+1}_{\text{fn. diff}} - 7\text{se.} =$ (19sec) if child provide Execution time.

1 → $\underbrace{15-3+1}_{\text{fn. diff}} - 6\text{se} = 7\text{ sec.}$

2 → $\underbrace{10-5+1}_{\text{fn. diff}} - \underbrace{0}_{\text{child Ex. time.}} = 6$

Total = 15+4 = (19sec) if child provide fn. diff. time.

132 Pattern

Thursday, 9 December 2021 10:42 PM

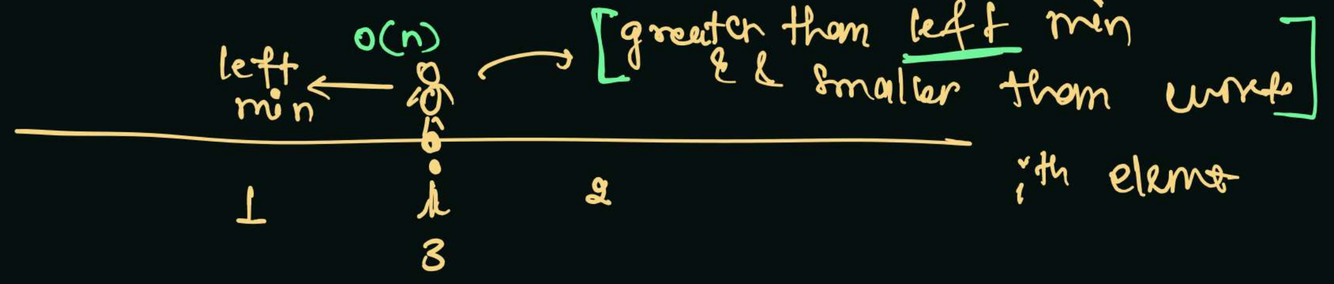
we have three Elements, arr[i], arr[j], and arr[k], $i < j < k$

1 3 2

and arr[i] < arr[k] < arr[j]

1 2 3

Brute force →



Exist → true

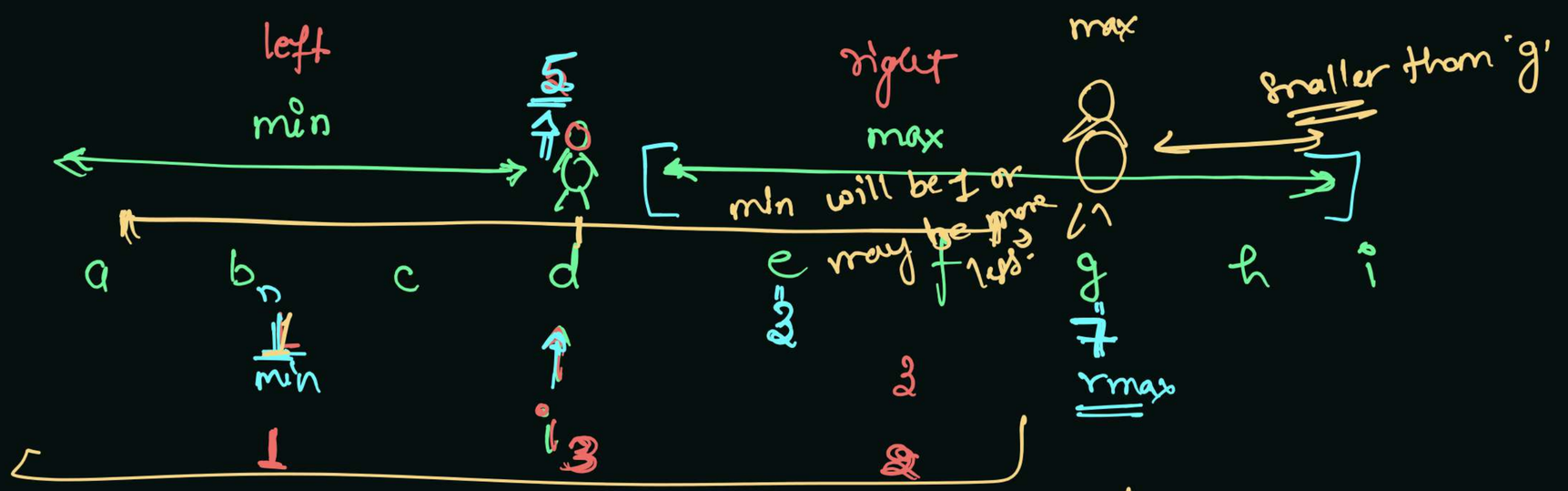
otherwise → false.

$O(n^2)$ → Time

$O(1)$ - Space

optimise - $O(n)$ → Time.

$O(n)$ → Space



min = ① = may be more smaller & ①

if (min < arr[i] > right max right max > min)

return true;

→ kithat

→ kahy

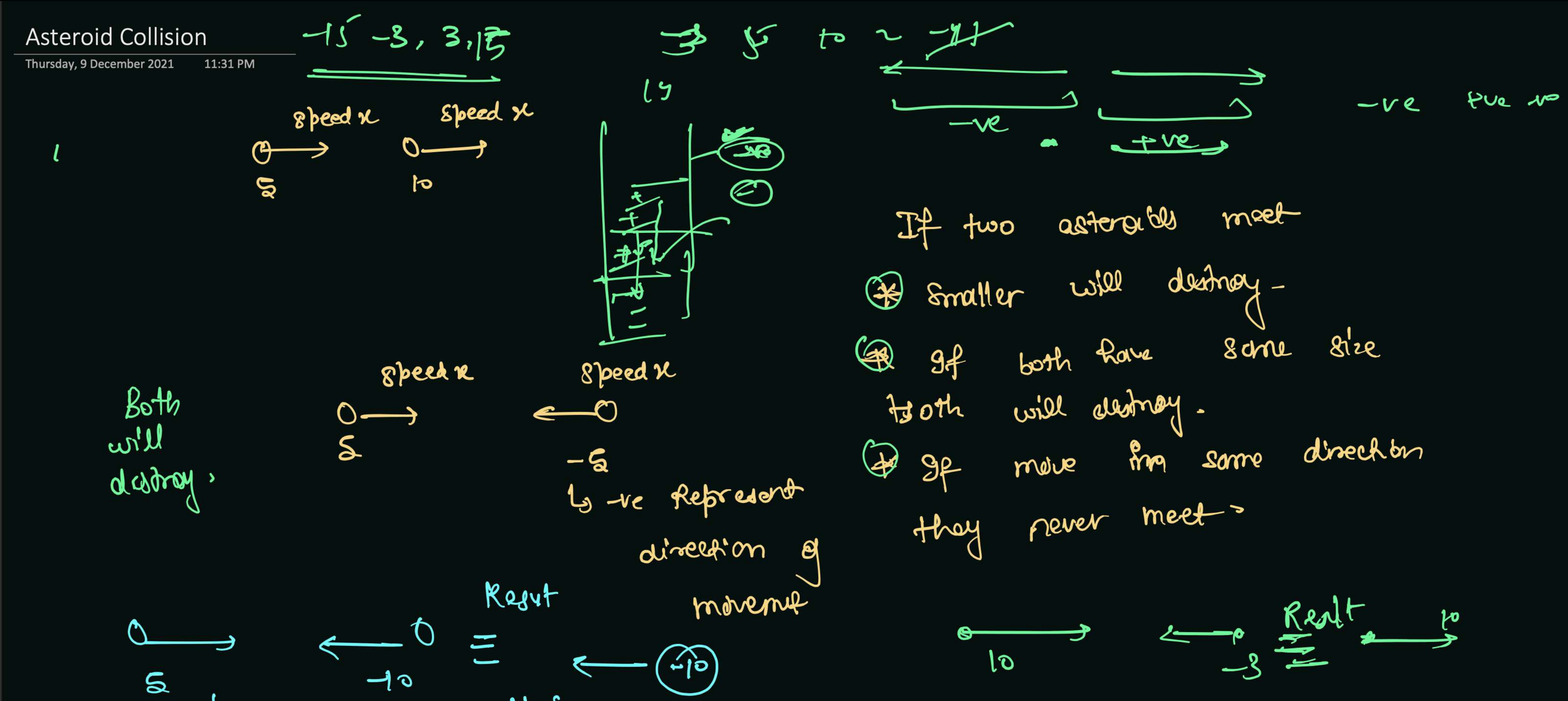
→ How

left min → array & left min

right max → travel and manage;

Asteroid Collision

Thursday, 9 December 2021 11:31 PM



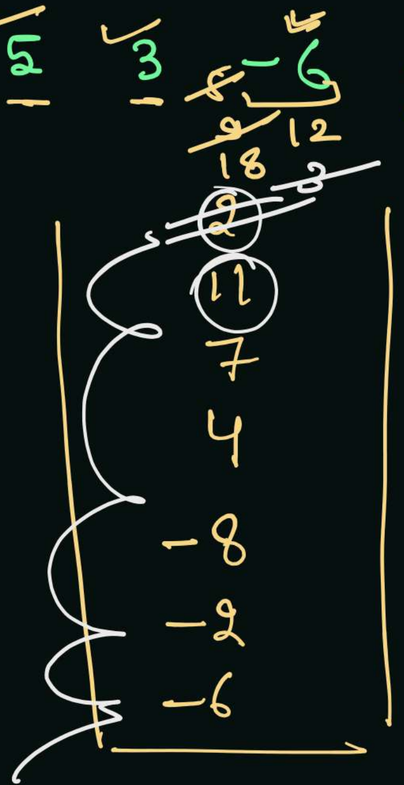
If +ve encounter → push in Stack

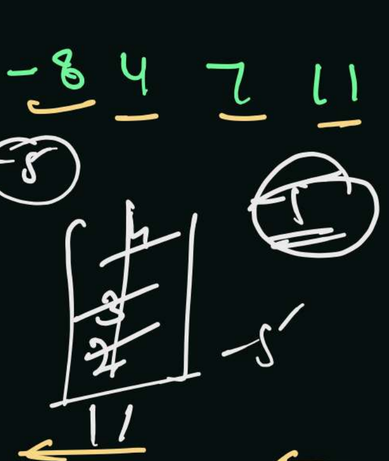
↳ -ve encounter → pop +ve until it can

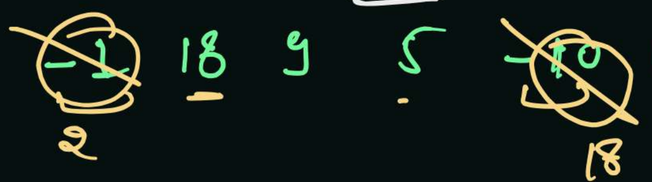
if +ve is completely out put -ve

otherwise +ve will destroy impact of -ve

5 3 -6 -2 -8 4 7 11 2 -1 18 9 5 -10 12







Speed constant they are stable & never meet

-6 -2 -8 4 7 11 2 18 12] Size

Remove K Digits

Thursday, 9 December 2021 11:41 PM

String num = 1 4 3 2 2 5 6 1

1 6 5 3 2 2 1

[1 2 5 6 1]

k = 3

smaller

4 digit

[1, 2, 4, 6]

smallest possible number

1 2 4 6

largest possible

6 4 2 1

NOTE:

k = 3 & k > 0

1 2 1 9

single digit

String num = 1 4 3 2 2 1 9

9 1 2 2 4 1

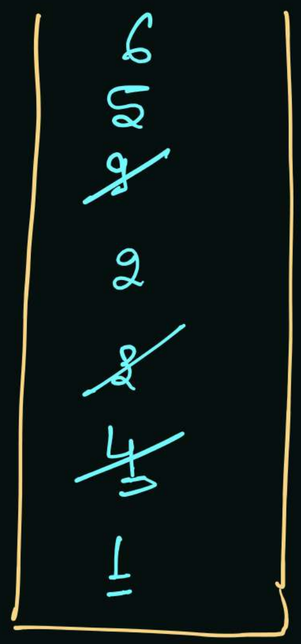
k = 3 & k > 0

[1 2 1 9] smaller

2

1

String = 1 4 3 2 9 5 6 7 → k = 2



If index is Ended and k is remain,

→ If k is remain ther Stack is certainly Increasing in order form Bottom to top.

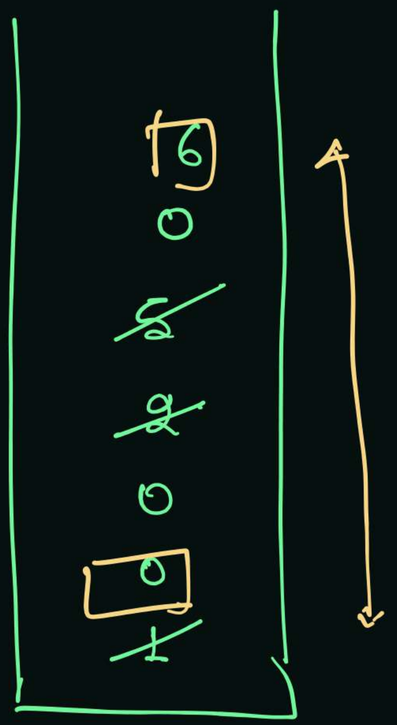
1 2 5 6 7

Remove Remaining 'k' from last i.e. top of Stack.

String num = "1 0 0 25 06"

Recursion & Backtrack → StringBuilder

Stack → Adapters



k = 2 2 1 0

Res = 0 0 0 6

Remove leading zero

2) "C" Deque

Stack Ques.

LinkedList

Array & String

trees.

Implement of Heap

" " HashHeap

Concept

graph

Level -1

Parallely often

L2

Question Banks

Mon →

Tusday. —

Wed →

Thursday →

Frid →

Sat → — —

Sun → →

Job.

Challenge + Benefit

Pen

Notes

L1 → Range

L2 → graph → move ahead

L2 - array & String → move ahead

Lil Trees → move ahead