Date: 19th Dec. 2021 ,

Max-stack

Check If Word Is Valid After Insertion

Design Hit Counter

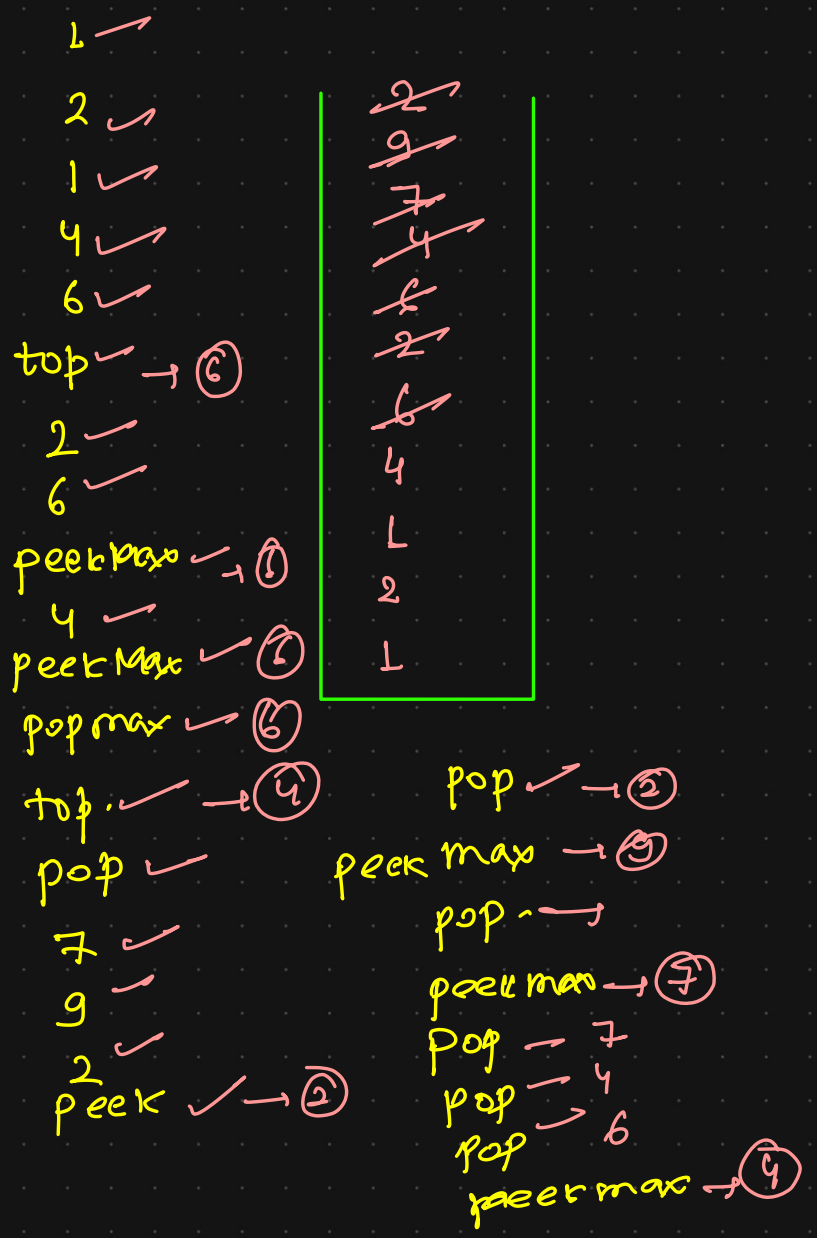
Number Of Recent Calls

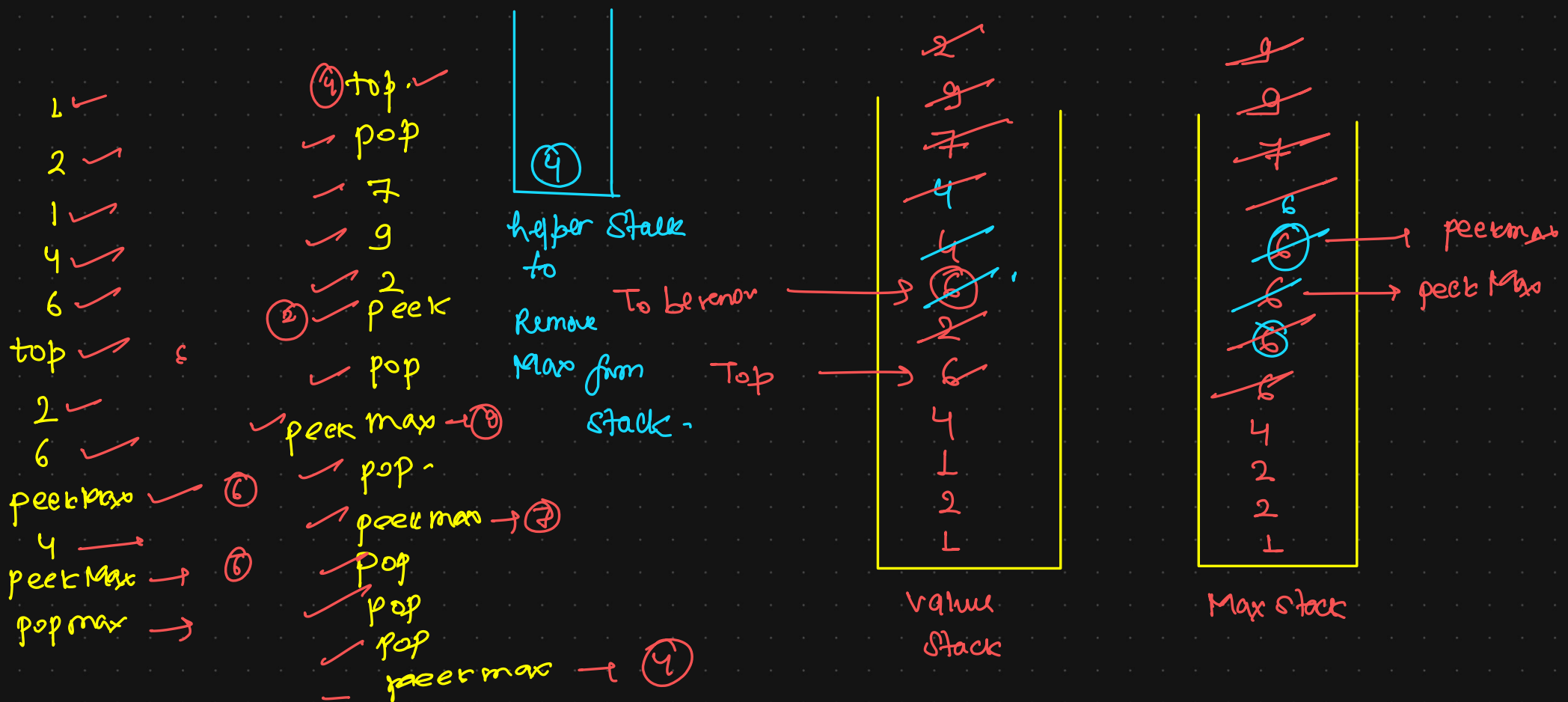
Moving Average From Data Stream

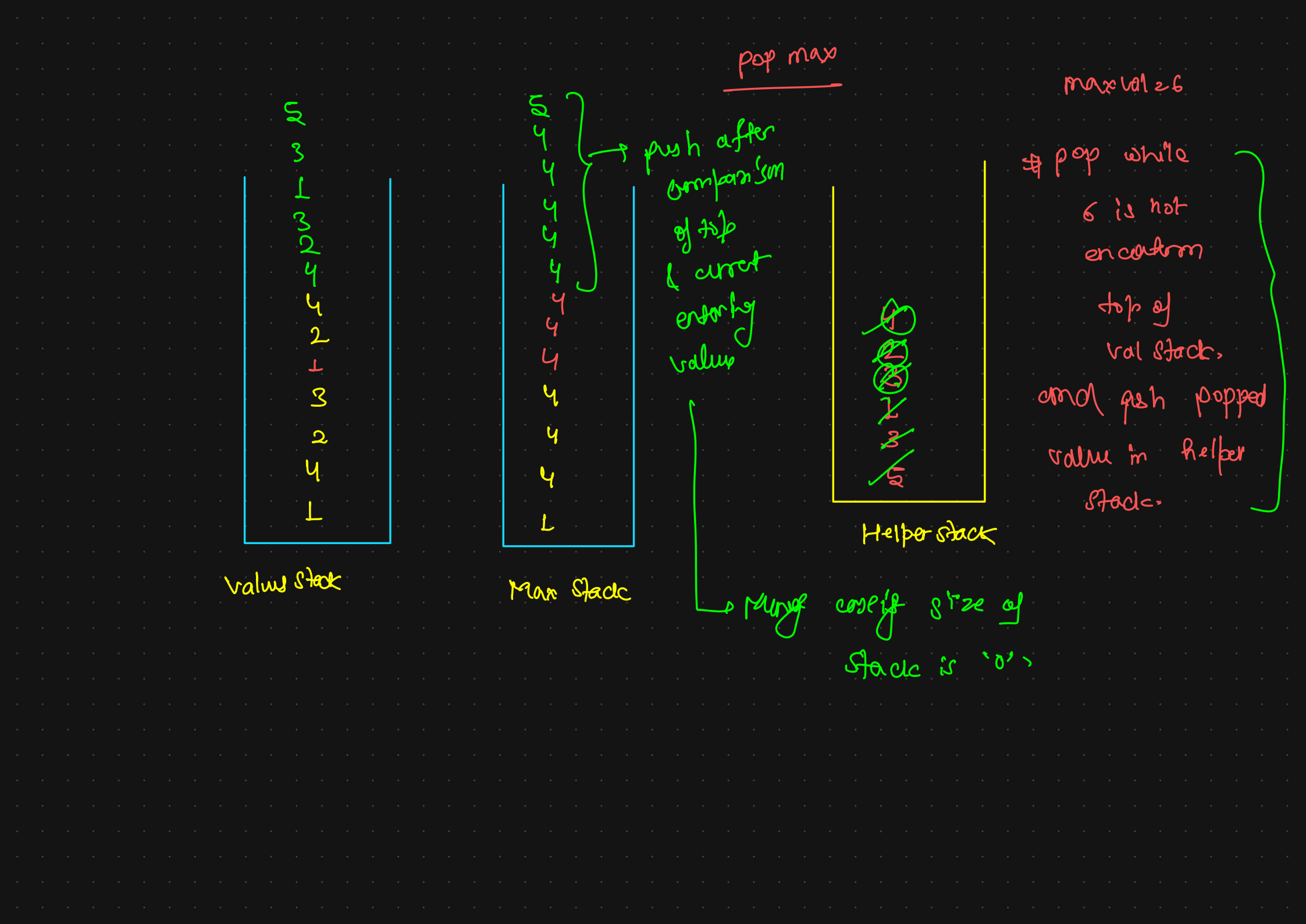
Max stack:

O(1)	push →	Normal stack push
O(1)	pop →	Normal stack pop
O(1)	top →	Normal stack top
O(1)	peekMax →	Max value in stack
O(n)	popMax →	Remove Max from stack

↳ If more than one value is available → Pop topmost max.




1
2
1
4
6
top
2
6
peekMax
4
peekMax
popMax
4 top
pop
7
9
2
2 peek
pop
peek max → 9
pop
peek max → 7
pop
pop
pop
peek max → 4
helper Stack to Remove max from stack.
To be removed
Top
value Stack
Max Stack
peek max
peek max


pop max
maxVal = 6
5 3 1 3 2 4 4 2 1 3 2 4 1
Value Stack
5 4 4 4 4 4 4 4 4 4 4 4 1
Max Stack
push after comparison of top & current entering value
Helper Stack
pop while 6 is not encountered top of val Stack, and push popped value in helper Stack.
merge condition size of Stack is '0'.

Check if word is valid after Insertion:

String S = "aabcbababcccabc"

"" → "abc" → "aabcbc" → "aabcbabcc" ↓

"aabcbababccc" ↓

"aabcbababcccabc"

Return true.

Because string 'S' is achievable from empty string.

* we can insert "abc" at any index and we can do it infinite time to match the given string.
* If it is possible return true, otherwise return false.

String S = "ababcababcccababcc"

"" → "abc" → "ababcc" → "ababcabcc" → "ababcababccc" ↓

ababcababccccabc

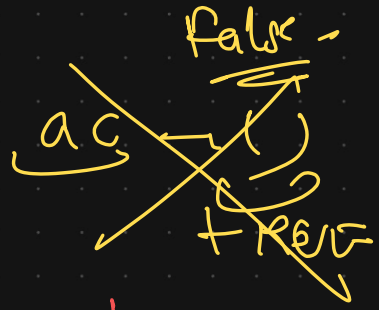
"ababcababcccababcc" ←

Return "TRUE"

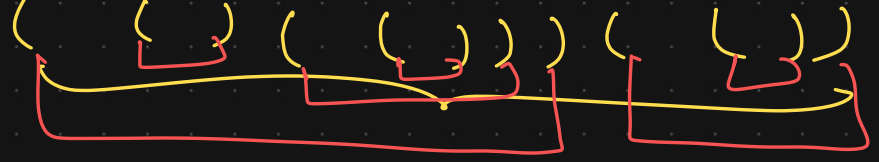
String S = "ab abc c cab" abc

" → "abc" → "ababc" Return false

How we decide is it possible or not → Similar to valid parenthesis



"a b abc ab abc c c a babcc"



true

"a abc b a b abcc c abc" Arrangement of 'b' is also important

a b a b c c cab

Return false

↳ It is similar but we to arrange b also

a → opening

c → closing

b → arrange.

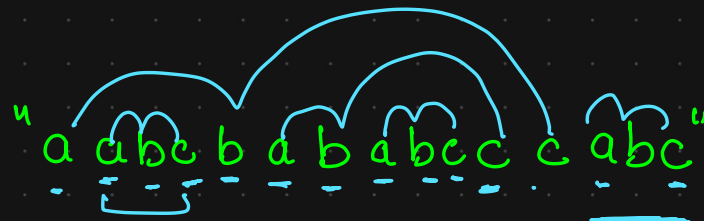
↳ opening → push

closing → pop & check the Result

Discussion → Empty to 'S'

Solve / Implement / Solution → Try to Convert S into empty string.

because problem is similar to valid parenthesis



b a c

looping character traversal

Stack.size == 0 → Return true.

a → push in stack

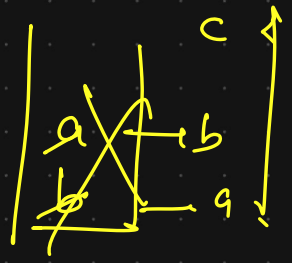
b → push in stack

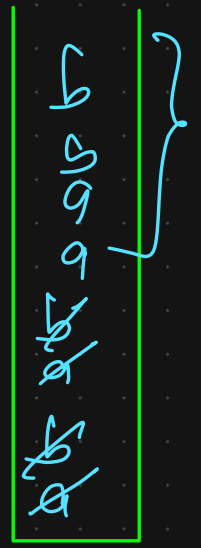
c → if(size >=2, top → b, again top → a)

pop → 2 time

} else {

return false.





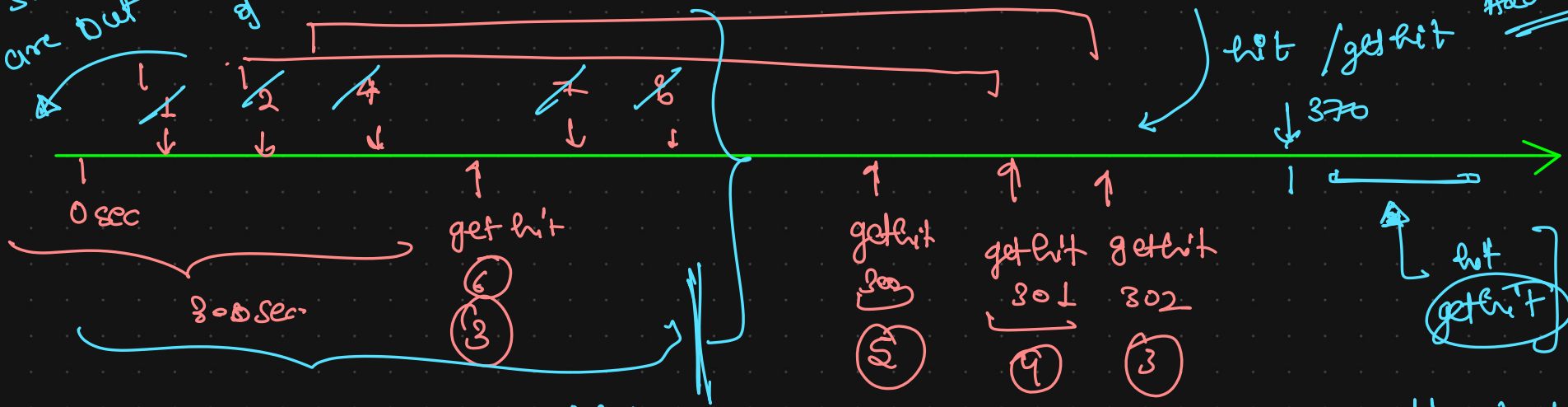
abc abc aabb → conclusion → At the end, if Stack.size > 0 Return False

At last → Return true.

Hit Counter:→

Data Structure → Deque

Some hits are Out after passing of 5 min. — Removal



hit (timestamp)

getHit(timestamp)

timestamp ≥ 370

0 to 370-300 → cycle

Return count of hits before & min. of current time stamp

300 sec. of current stamp.

hit → 1

hit → 2

hit → 4

getHit(6) → 3

hit → 7

hit → 8

getHit → 300 i.e. 300 sec. of current stamp.

getHit → 301

getHit → 302

hit → 370

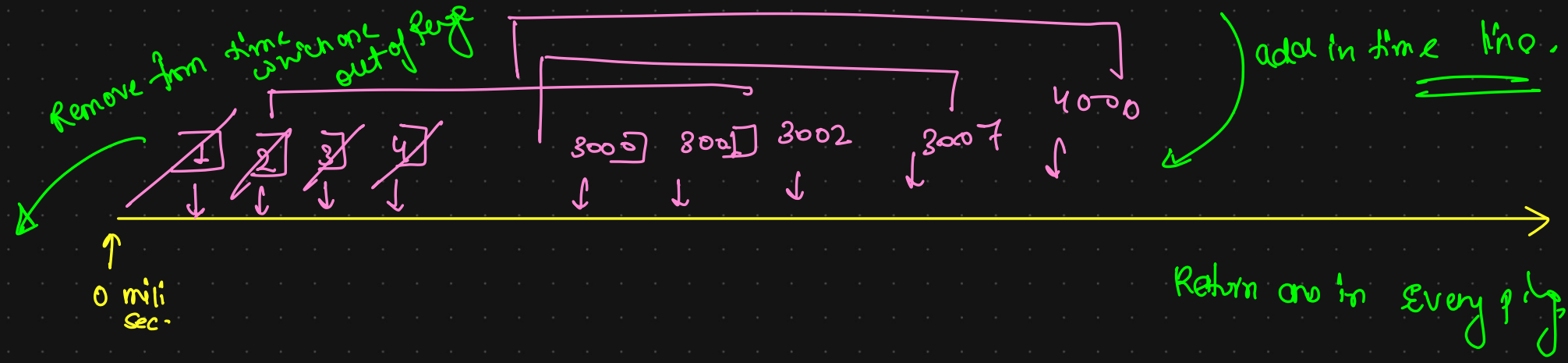
$timestamp_{i-1} <= timestamp_i$

hit → Range — Range element in Deque.

How → Remove Element from start which are out of Range.

getHit → Size of Deque

Number of Recent Calls :→



ping(1) → ①
ping(2) → ②
ping(3) → ③
ping(4) → ④
ping(3000) → ⑤ [1-3000, 1]
ping(3001) → ⑥ [0-3001]
ping(3002) → ④ [3001-3000, 3001] [1, 3001]

ping(3007) → ④
ping(4000) → ⑤

3002-3000, 3002
[2, 3002]

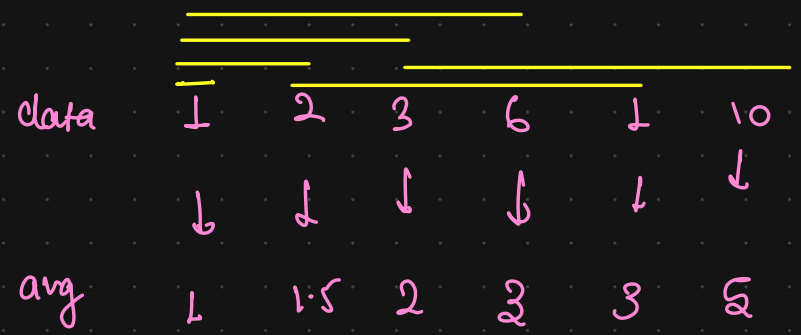
3007-3000, 3007
[7, 3007]

4000-3000, 4000
[1000, 4000]

Moving Average from Data Stream

k=4 → window -

Stream of Data → Average for every data.



if enough count of data is available then

→ sum/k] Avg

add in queue and remove.] Sum

otherwise

→ sum/size of Deque -] add in queue] Sum

k=4

data. 1 2 3 6 1 10 - - - -

avg. 1 1.5 2 2 3 5

if size > k

sum = sum + n.data - old data.

12+10-2 = 20

sum = 0 1 3 6 12+1 size is smaller than k
=13-1=12