Deque 2 March

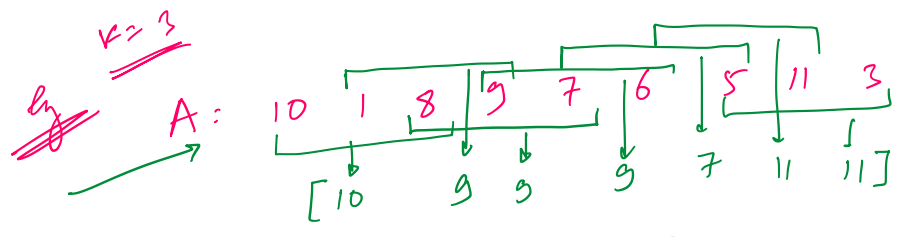
Saturday, 26 March 2022 9:03 PM

Sliding Window Max

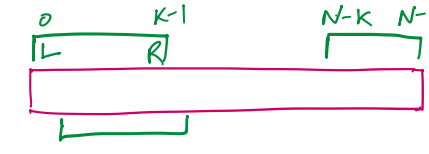
Q Given an integer arry A.
Find the max element ∀ subarrays of size K.

K = 3

Eg


A: 10 1 8 9 7 6 5 11 3
[10 9 9 9 7 11 11]

SAs of size K = N-K+1



BF :

```
L = 0, R = K-1
while( R < N) {
    // [L, R]
    maxi = -∞
    f( i = L; i<= R; i++) {
        maxi = max( maxi, A[i]);
    }
    print( maxi);
    L++, R++;
}
```

N-K+1

O(K)

TC: O(K (N-K+1))

~~TC : O(N K)~~

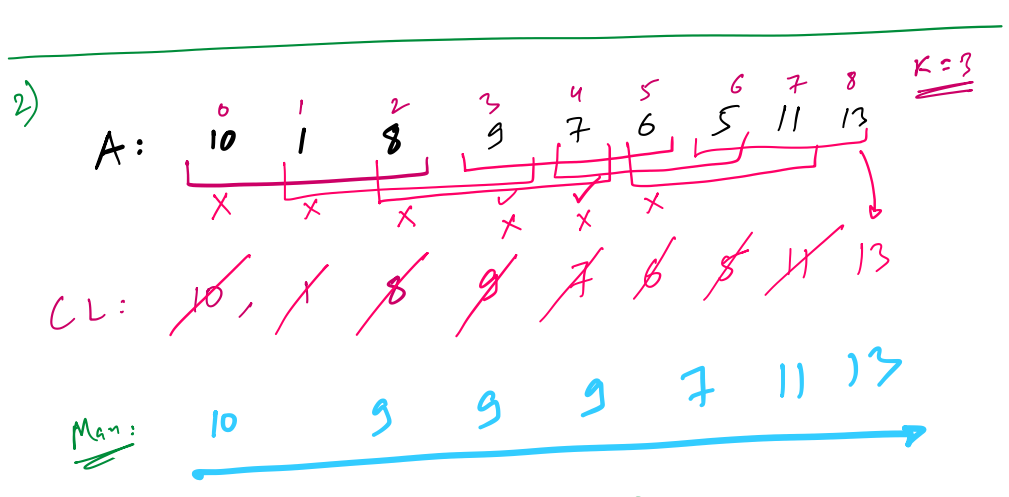
K = N/2

TC: O(N²) X

1 <= N <= 10⁵

---

2)


K=3
A: 10 1 8 9 7 6 5 11 13
0 1 2 3 4 5 6 7 8
CL: ~~10~~, ~~1~~ ~~8~~ ~~9~~ ~~7~~ ~~6~~ ~~5~~ ~~11~~ 13
Max: 10 9 9 9 7 11 13

9

CL: 12 9

⊛ NOTE: CL will always be sorted in decreasing order!

A: 1 2 3 13 18 171 2 46 6 8 21

CL ~~1~~ ~~2~~ ~~3~~ ~~13~~ ~~18~~ ~~171~~ ~~2~~ ~~46~~ ~~6~~ ~~8~~ | 21

Max: 3 13 13 171 171 171 46 46 21

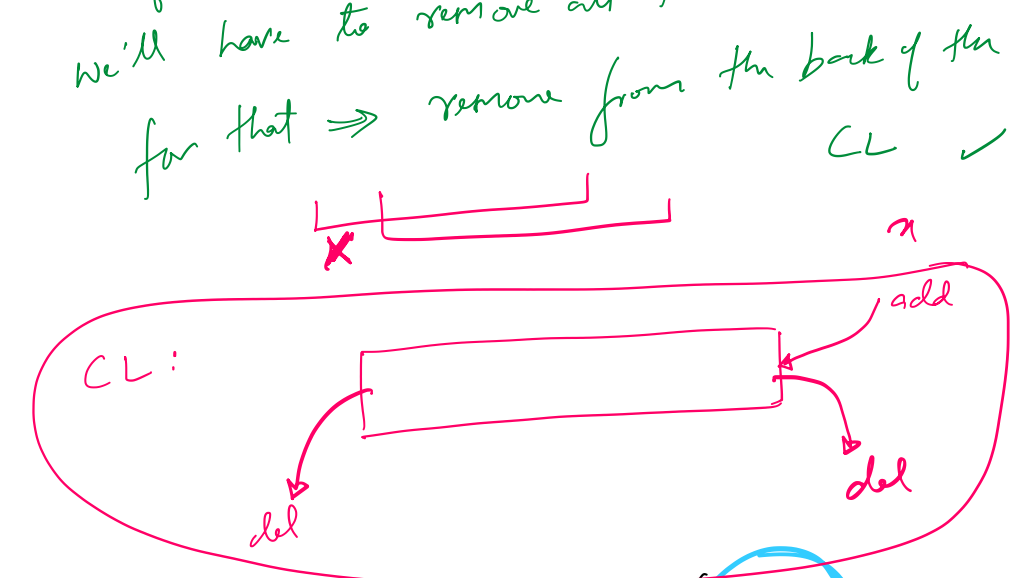
⊛ If an element is going out of the window, if it is present, it would be at the frontmost pos. of CL.

⊛ Max element of CL would be the frontmost element of the CL

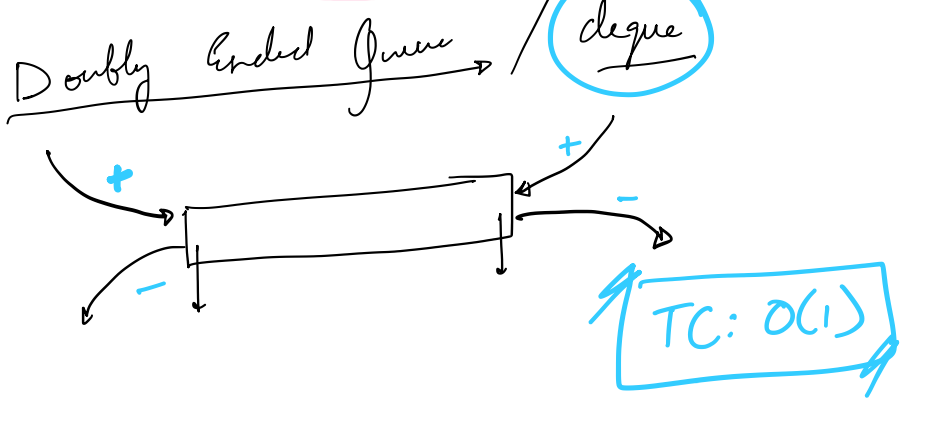
⊛ CL: 10 9 ~~7~~ ~~6~~ ~~5~~

x = 8

If a new element enters, before inserting it in the CL, we'll have to remove all the elements <= it.
for that ⇒ remove from the back of the CL ✓

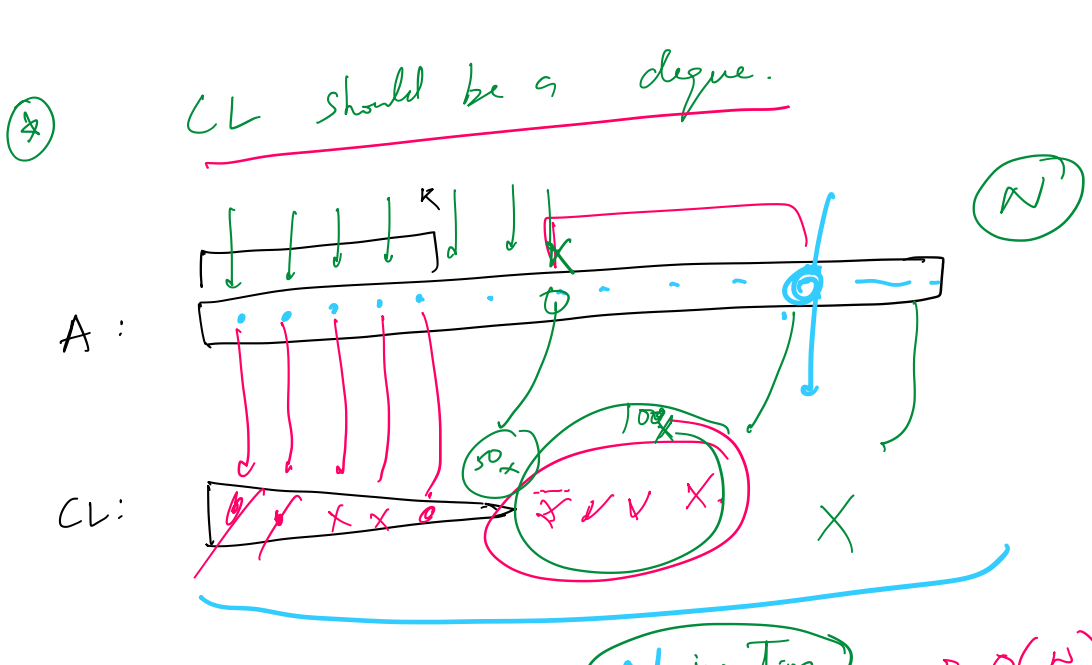

CL:
x
add
del
del

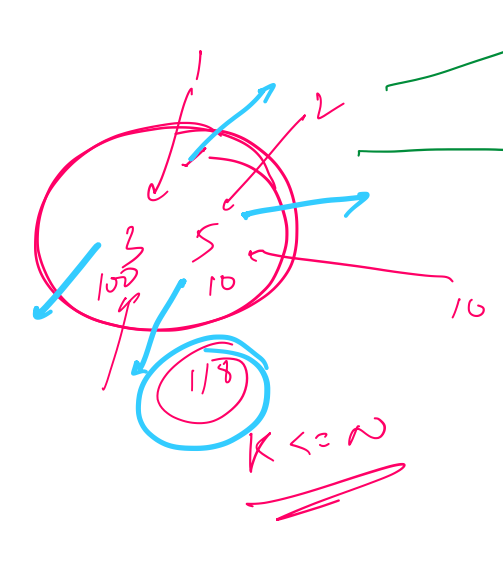
⊙ Doubly Ended Queue / deque



TC: O(1)

⊛ CL should be a deque.


A:
CL:
K
N
50
100
X


1
2
3
5
10
100
10
1/8
K <= N

N insertion → O(N)

N del → O(N)

N ite → O(N)

N acc from front → O(N)

TC: O(N)

SC: O(K)

```
deque <int> CL;
fL i=0; i< K; i++) {
    while( !CL.empty() && CL.back() <= A[i]) {
        CL.pop_back();
    }
    CL.push_back( A[i]);
}
print ( CL.front());
L=1, R=K;
while( R<N) {
    // [L R]
    if(!CL.empty() && CL.front() == A[L-1])
        CL.pop_front();
    }
    while( ! CL.isempty() && CL.back() <= A[R]) {
        CL.pop_back();
    }
    CL.push_back( A[R]);
    print( CL.front());
    L++, R++;
}
```

TC : O(N)

SC : O(K)

Q Given an infix expression, you have to convert it to postfix.

INFIX : 3 + 5

3 + 5 * 4

(3 + 5) * 4 / 2

IN POST

POST FIX : 3 + 4 → 3 4 +

3 * 4 + 2

(3 4 *) + 2

3 4 * 2 +

IN 4 / 7 + 2 - 5 * 2

4 7 / + 2 - 5 2 *

4 7 / 2 + - 5 2 *

POST 4 7 / 2 + 5 2 * -

5 + 6 - 10

5 6 + 10 -

⊙ 3 + 5 ⇒ 3 5 +

+

⊙ 10 + 3 * 4 ⇒ 10 3 4 * + ✓

~~+~~ ~~*~~

⊙ 10 * 3 + 4 ⇒ 10 3 * 4 +

~~*~~ ~~+~~

⊙ 5 + 6 - 10 ⇒ 5 6 + 10 -

~~+~~ -

⊙ 5 + 6 * 10 - 4 ⇒ 5 6 10 * + 4 -

~~+~~ ~~*~~ ~~-~~

⊙ (3 + 5) * 10 ⇒ 3 5 + 10 *

~~(~~ ~~+~~ ~~*~~

3 10 3 4 2 / - * + 3 +

⊙ 3 + 10 * (3 - 4/2) + 3

~~+~~ ~~*~~ ~~(~~ ~~-~~ ~~/~~ ~~+~~

3 10 3 4 2 / - * + 3 +

Stack

traverse the INFIX

- Operand: put in postfix exp.
- "(": push in the stack
- ")": pop all operators until we get "(" AND add all those popped op in the postfix exp.
- Operator: pre (inc) <= pre(st.top())
  - T: pop from stack AND add to postfix exp
  - F: push the op in the stack

+

HM :

'+' : 1

'-' : 1

'*' : 2

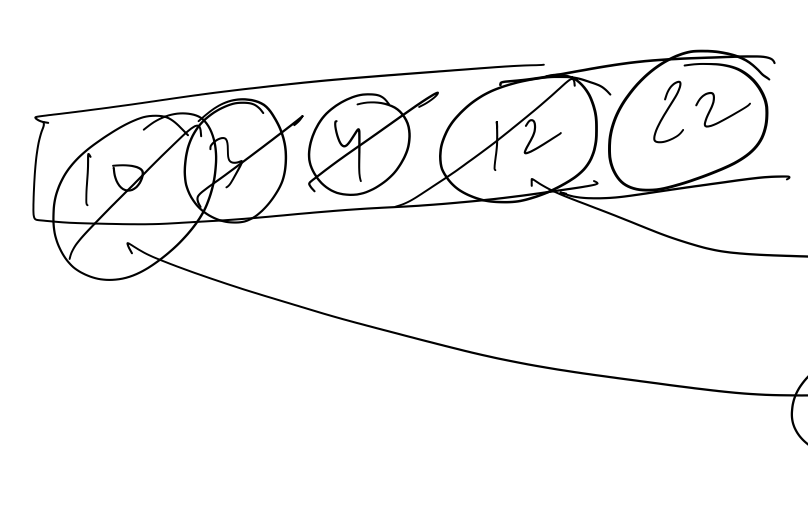
'/' : 2

'^' : 3

Q Evaluation of postfix

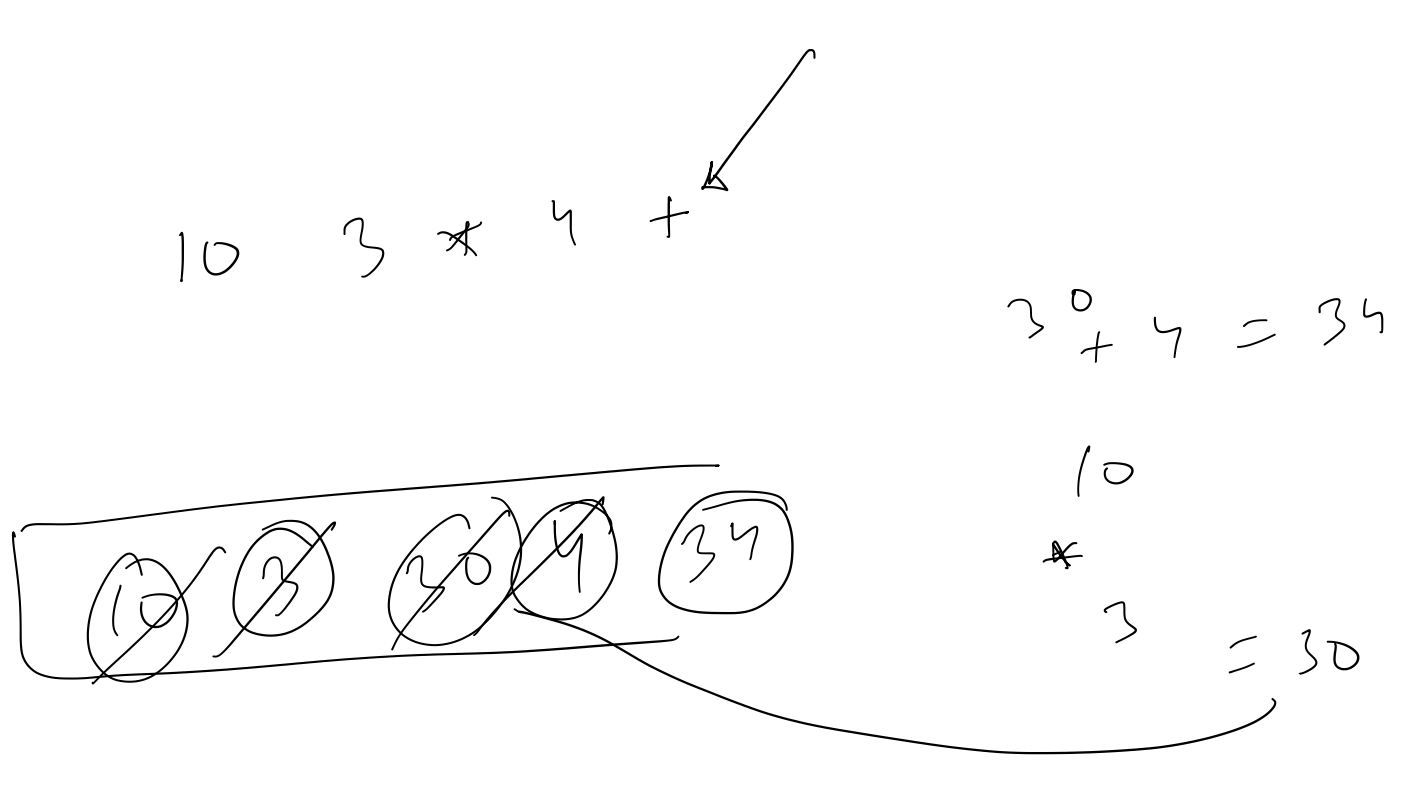
⊙ 10 3 +

operand operand operator



⊙ 10 3 4 * +

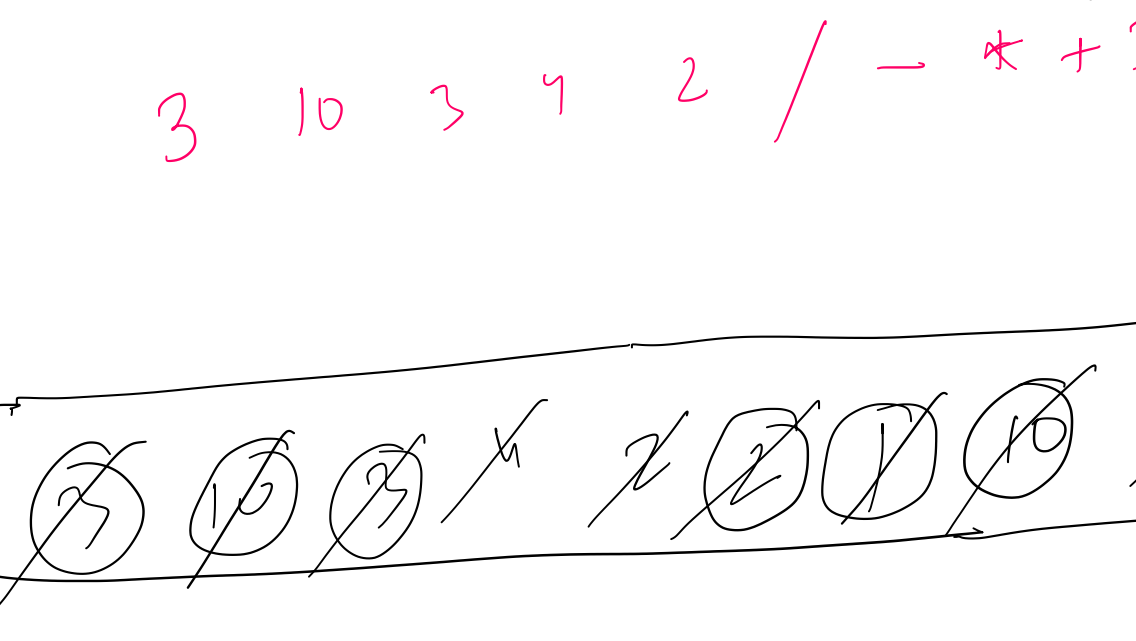
10 + (3*4)


10
3
4
12
22
3 * 4 = 12
10 + 12
22

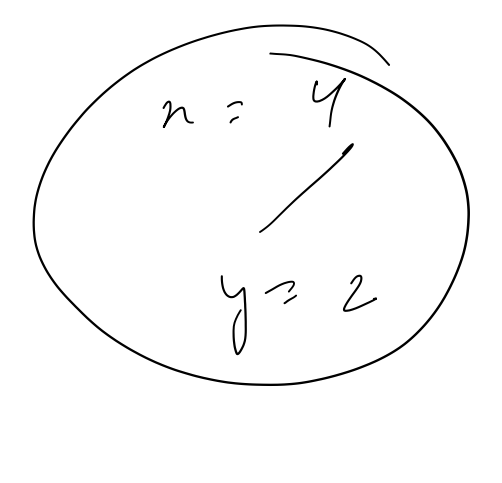
⊙ 10 3 * 4 +

30 + 4 = 34

10 * 3 = 30

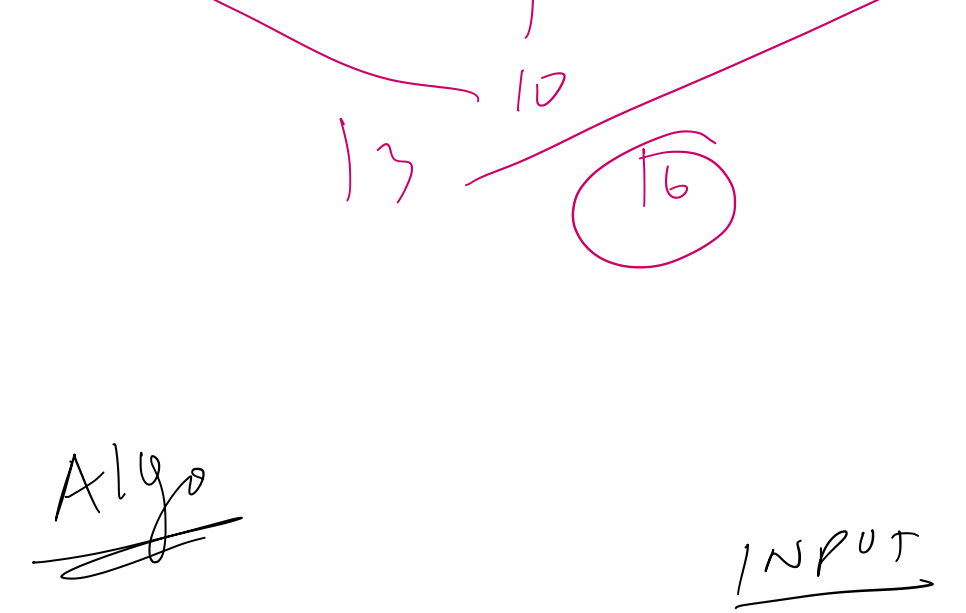


⊙ 3 10 3 4 2 / - * + 3 +


3
10
3
4
2
2
1
10
13
3
16

x = 4

y = 2


3 + 10 * (3 - 4/2) + 3
1
10
13
16

Algo

INPUT

- operand: push it in stack
- operator:
  - y = st.pop();
  - x = st.pop();
  - push in stack → x operator y

final ans : st.top()

O(N)

---

a ^ b ^ c

a^{b^c}

~~^~~ ~~^~~

a b ^ c ^