30 April 2023 19:33

Math.pow (_/_)

double

Sc n. nextInt()

int

.nextLong()

long

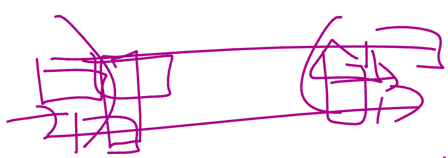
Implement Stack → Q
- → LL
- → A.L
- → Arr

Impl Q → S
- → LL
- → AL
- → arr

10 → 20 → 30 → 40

H T

	LL head	LL head tail	double LL head tail
Add First	O(1)	O(1)	O(1)
RemoveF	O(1)	O(1)	O(1)
add Last	O(n)	O(1)	O(1)
Rem Last	O(n)	O(n)	O(1)



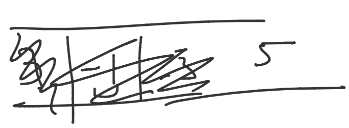
add ... adL

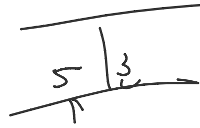
remove — dequeue [illegible]

5, 4, 3, 2, 1 k = ?

Window position	Max
[1 3 -1] -3 5 3 6 7	**3**
1 [3 -1 -3] 5 3 6 7	**3**
1 3 [-1 -3 5] 3 6 7	**5**
1 3 -1 [-3 5 3] 6 7	**5**
1 3 -1 -3 [5 3 6] 7	**6**
1 3 -1 -3 5 [3 6 7]	**7**

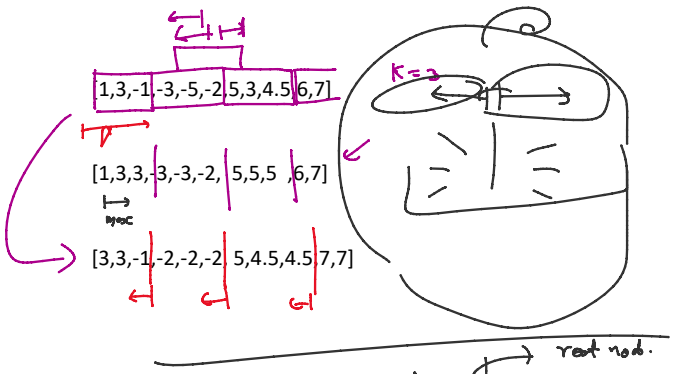
From <https://leetcode.com/problems/sliding-window-maximum/>

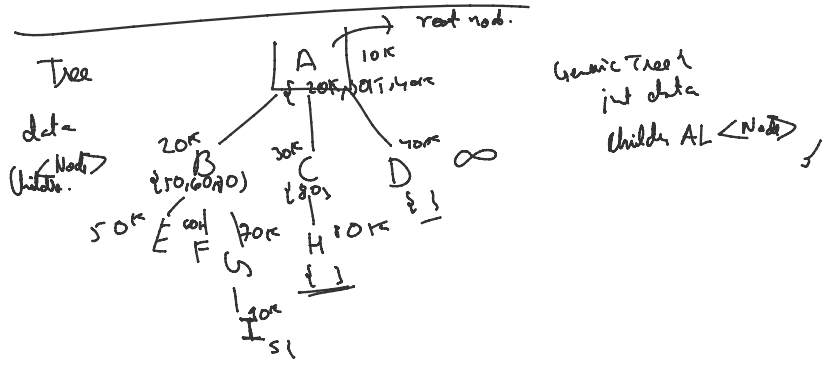


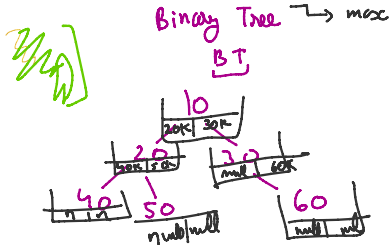


[1,3,-1,-3,-5,-2,5,3,4.5,6,7]

From <https://leetcode.com/problems/sliding-window-maximum/>







Node {
data
Node left
Node right
}

10 K

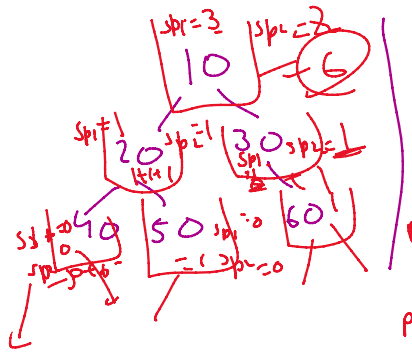
S.L.R → Pre

L.S.R → In

L.R.S → Post

```
private void print(Node nn) {
    if(nn==null) {
        return;
    }
    System.out.println(nn.data);
    print(nn.left);
    print(nn.right);
}
```

10, 20, 40, 50, 70, 30, 60



Pre → S L R → 10, 20, 40, 50, 30, 60

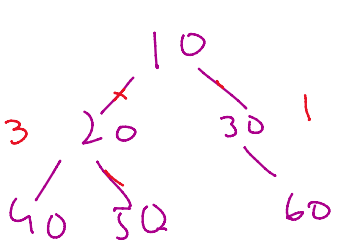
In → L S R → 40, 20, 50, 10, 30, 60

Post → L R S → 40, 50, 20, 60, 30, 10

Post S R L →

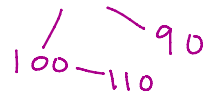
In R S L →

Pre R L S →

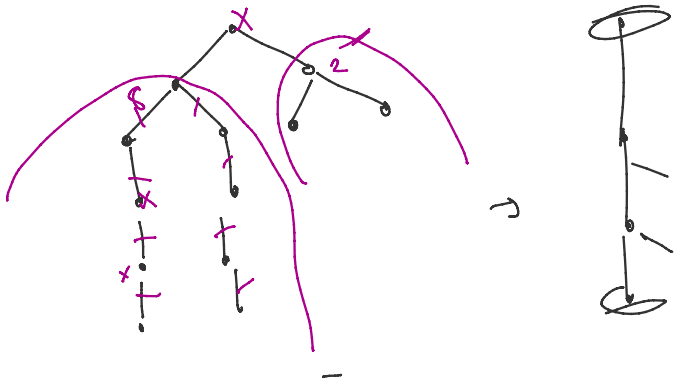


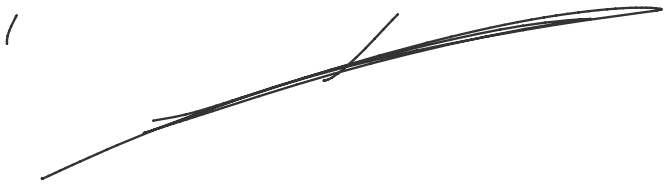
10

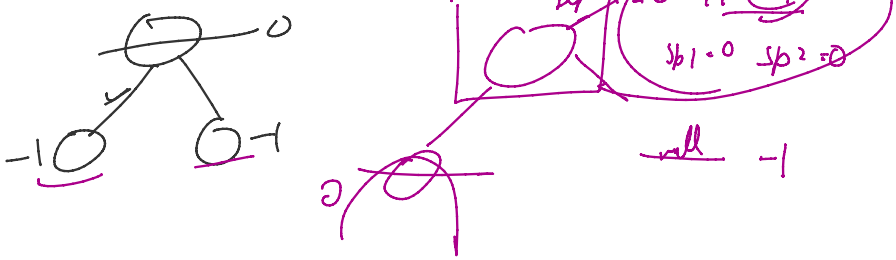
lt

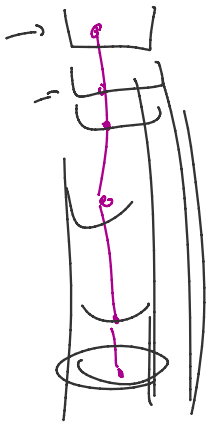


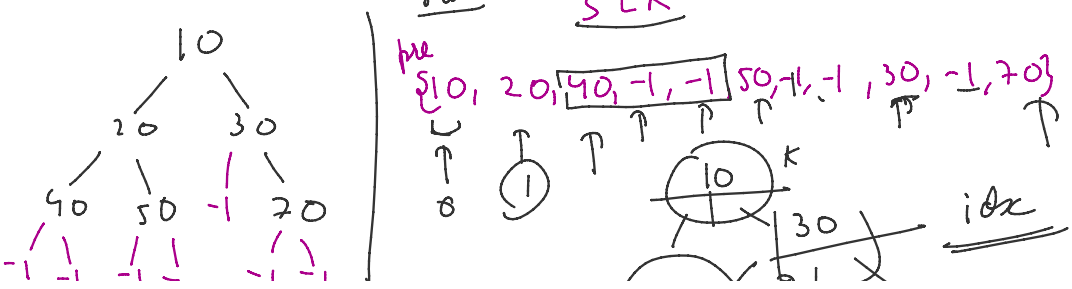
Dia → max dist any 2 node.

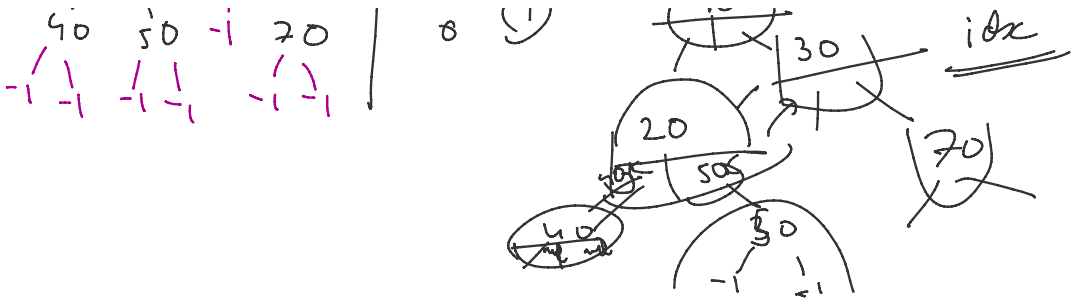











40
50
-1
70
-1
-1
-1
-1
-1
-1
0
1
30
idx
20
70
40
50
30
-1
-1