#Segment Tree

Concepts & Qns...

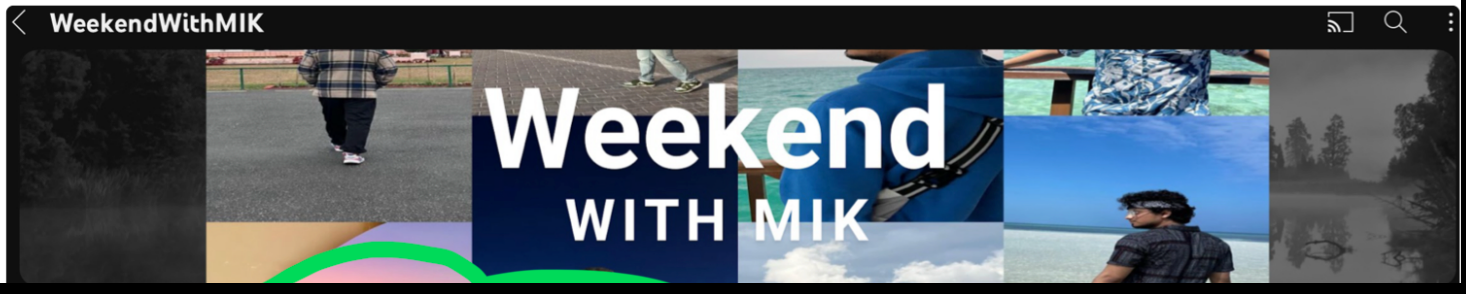
codestorywithmiK

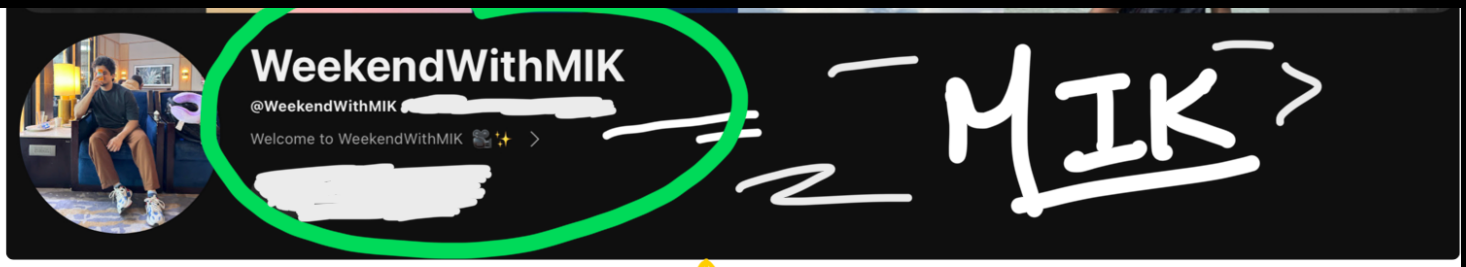
CSwithMIK

codestorywithMI

"No more fear of Segment Tree"

video - 14





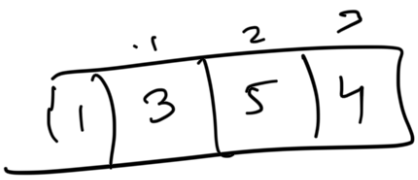
Try this channel to see my "Life behind the Scenes + Tech News"

Motivation :-

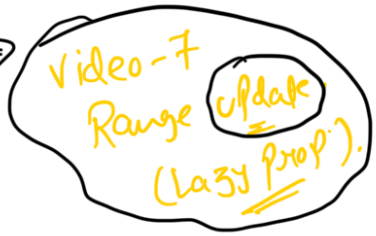
If you want to see a change in yourself, you need to act now.

It's never late to start and there is no place for those who easily quit.

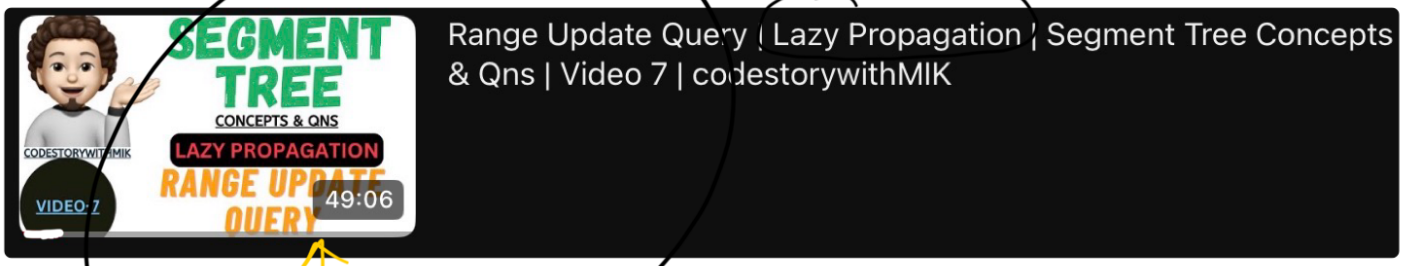
MIK...



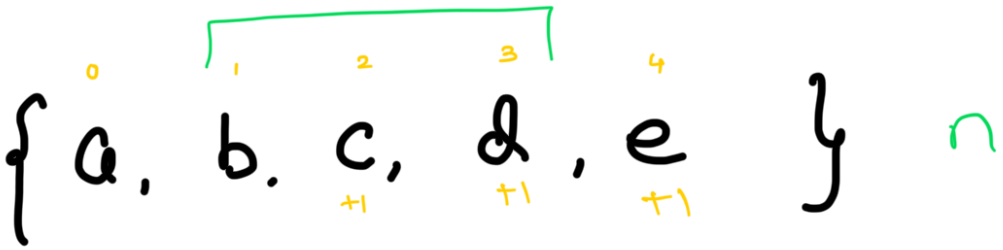
(1, 3) → min/max
(2, 4) → min/max



Mutable Range Min/Max

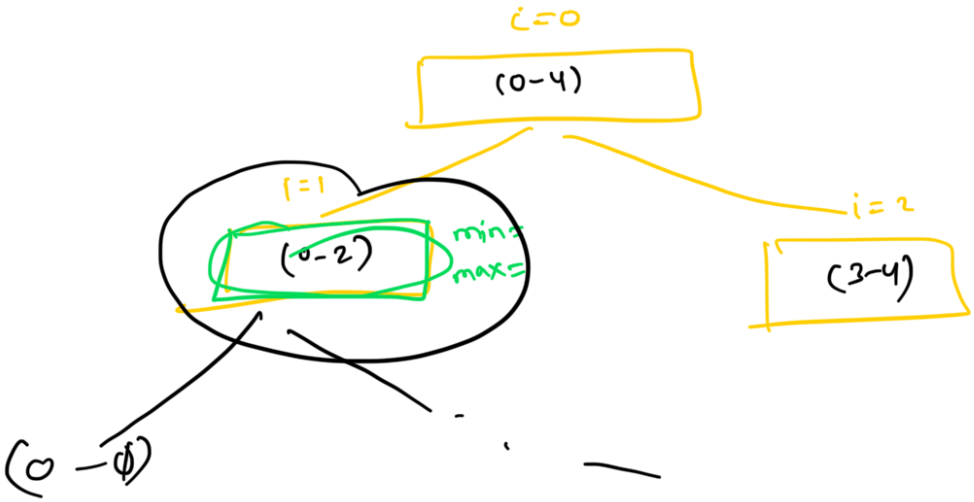


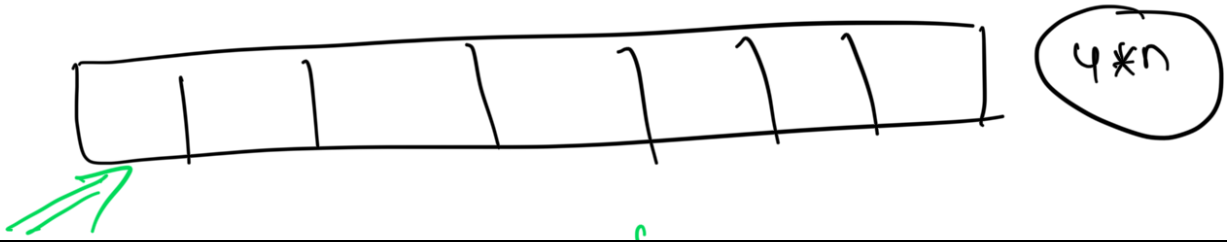
Must watch ...



min - max (1, 3)
min - max (2, 4)
:
} q * n ✗

Segment

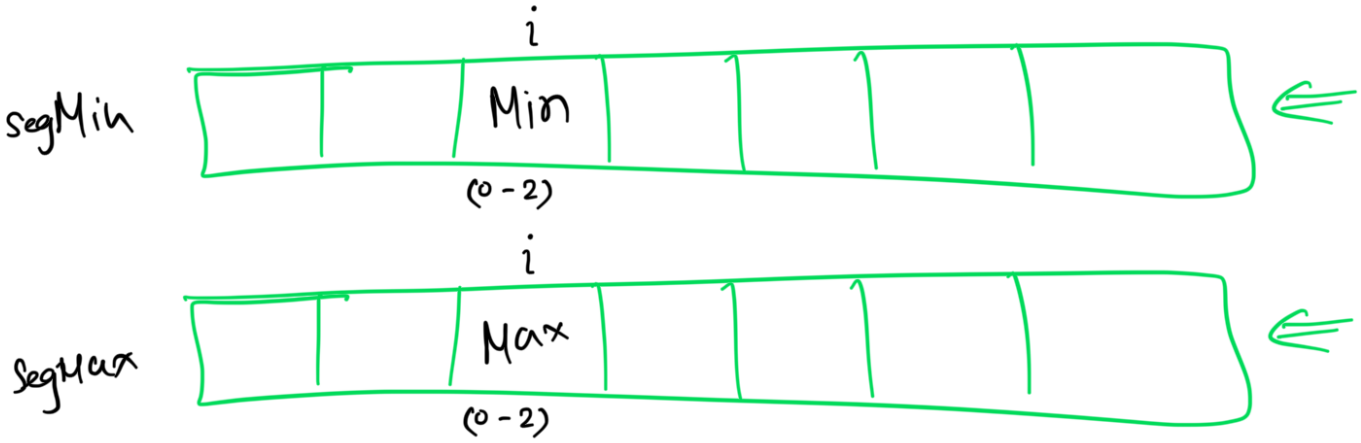


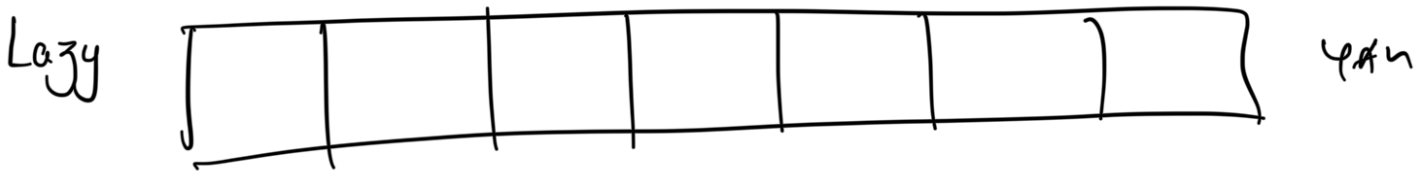


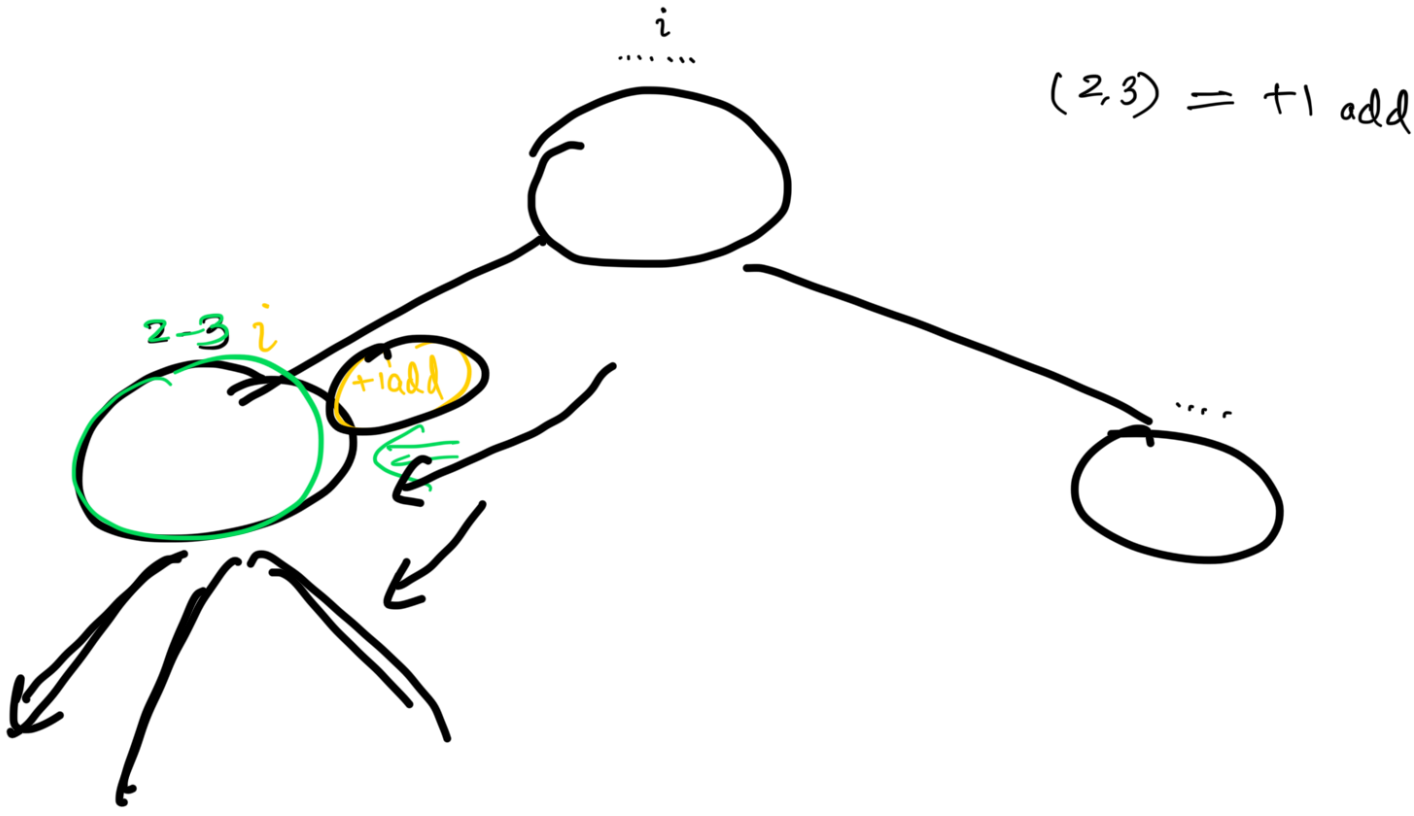
Node {
min =
max =

}

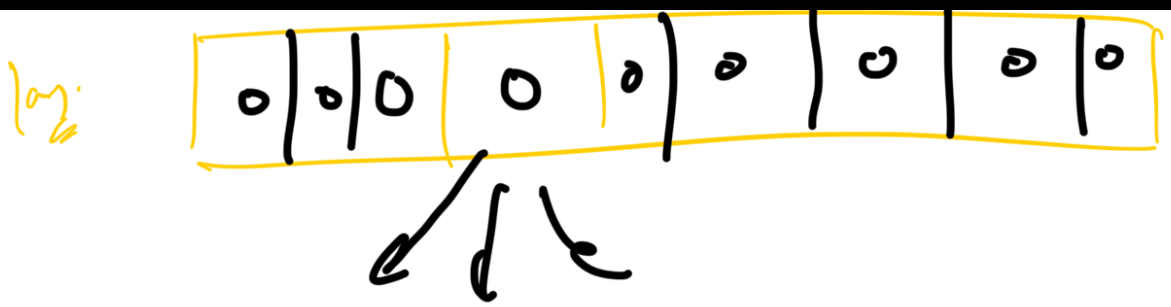
segTree<Node> (4*n).

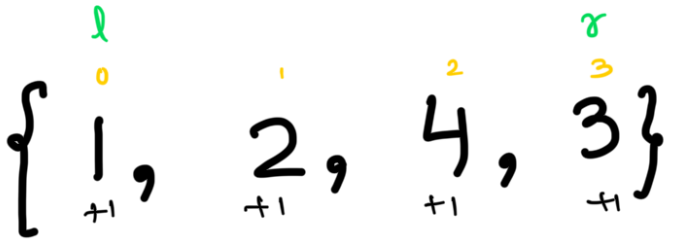






i



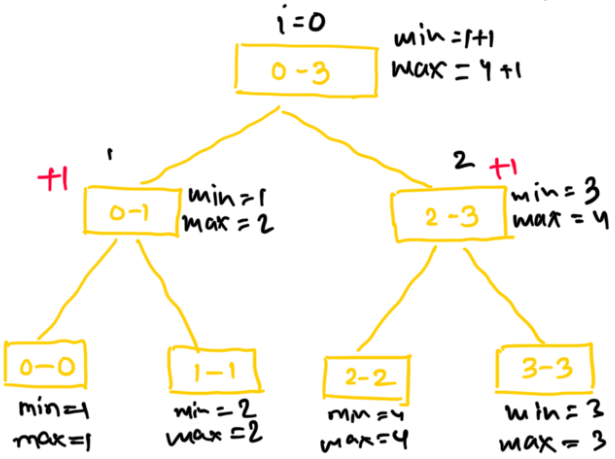


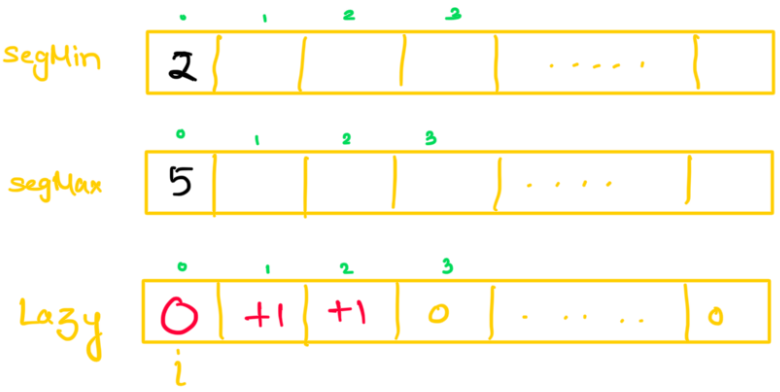
start = 0 ←
end = 3 ←
val = +1

updateRange (0, 3, +1, i=0, l, r)

l = 0, r = 3

val = 1





if (l >= start && r <= end) {

}

segMin[i] = min value in the range represented by i^{th} node.

segMax[i] = max value in the range represented by i^{th} node.

laz[i] = value held at i^{th} node to be propagated

```
if ( l >= start  &&  r <= end ) {
    lazy[i] += val;
    // 2i+1  +val (lazy)
    // 2i+1  +val (lazy)
    // ith node  min += val
    //           max += val
    // lazy[i] = 0;
}
```

root node

```
void updateRange ( start, end, i , l, r, val) {
```

query: start, end — i=0 — 0 - n-1: l, r — update: val

```
    propagate ( i, l, r ) ;   → check lazy[i] != 0

    if ( start > r  ||  end < l )
        return ;

    if ( l >= start  &&  r <= end ) {
        laz[i] += val;
        Propagate (i, l, r);
        return;   // optimisation.
    }

    int mid = (l+r)/2;
    updateRange (start, end, 2i+1, l, mid, val);
```

```
        updateRange (start, end, 2i+2, mid+1, r, val);

        segMin[i] = min (segMin[2i+1] , segMin[2i+2]);
        segMax[i] = min (segMax[2i+1] , segMax[2i+2]);
    }
```

```
void propagate (i, l, r) {
    if (lazy[i] != 0) {
        segMin[i] += val;
        segMax[i] += val;
        if (l != r) {
            lazy[2i+1] += lazy[i];
            lazy[2i+2] += lazy[i];
        }
        lazy[i] = 0;
    }
```

i l, r

[start, end]

log (n)


Q * log(n).
S.C = O(4*n).
P.O.T.D.