Segment Trees

Q Given an array, & given Q queries.

1) i, val → make a[i] = val
2) l, r → Find minimum in index range [l:r]

0	1	2	3	4	5	6	7
10	2	~~1~~ -7	-3	5	8	1	15

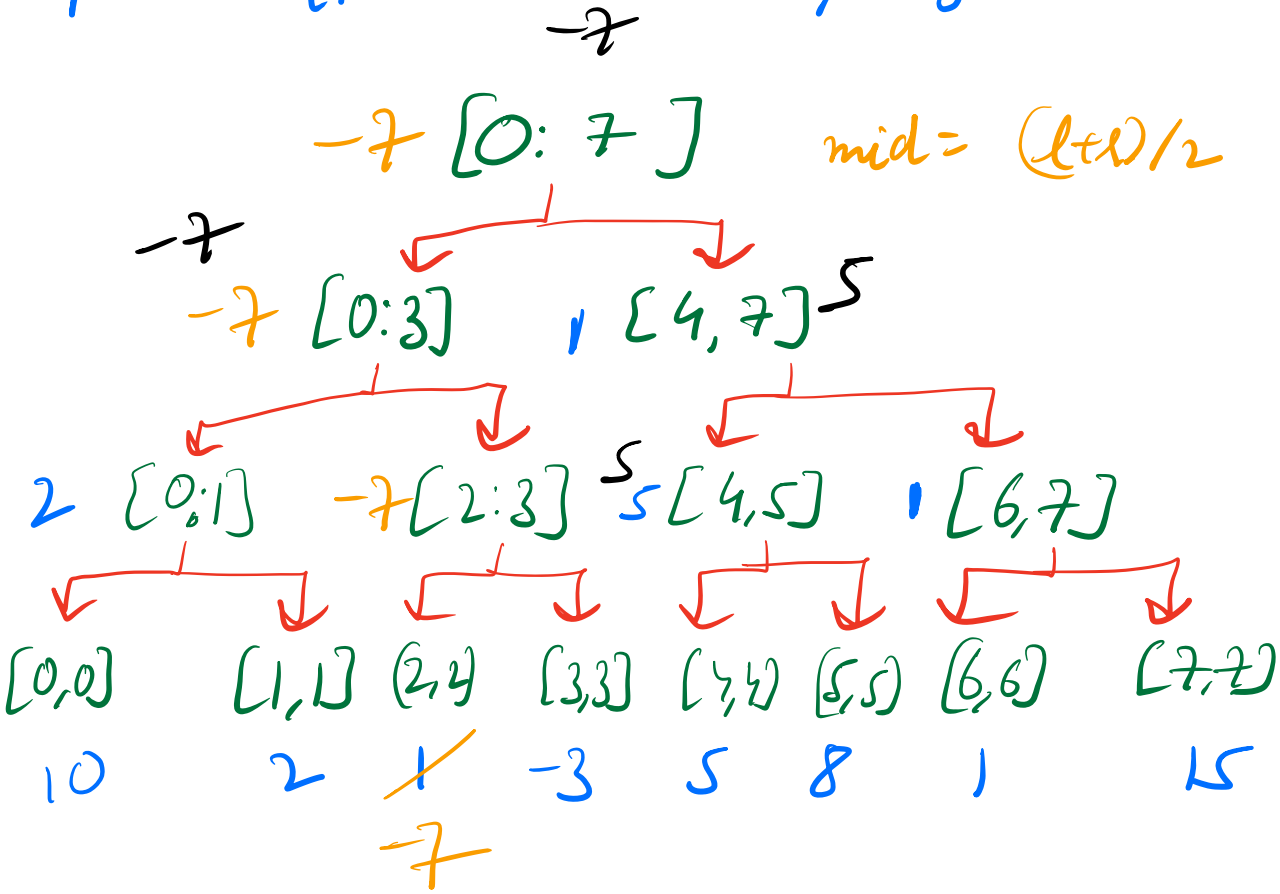
2) L=1 R=4 -3
2) L=4 R=6 1
1) i=2 val=-7
2) L=0 R=5 -7

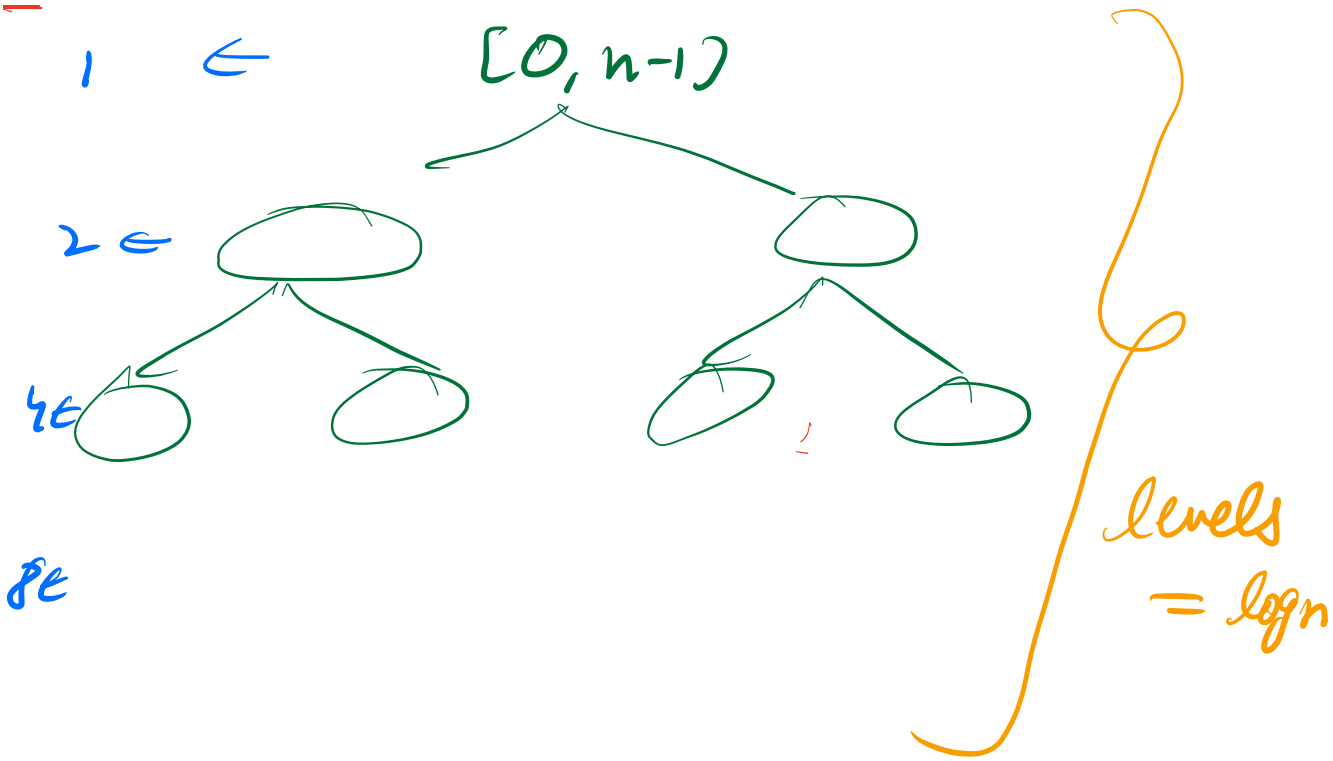
Brute force :

1) Make a[i] = val
2) Iterate on [l:r].

TC: O(Q*N)

Segment tree ⇒ Maintain answers for different blocks/segments





level $i \Rightarrow 2^i$ nodes

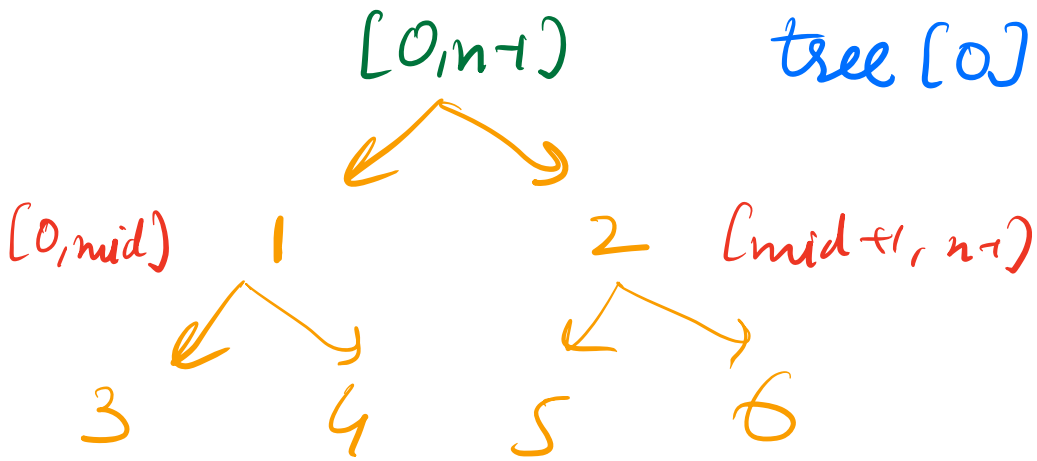
Total $\Rightarrow$ Level 0 + level 1 + level 2 ... $2^{\log n}$

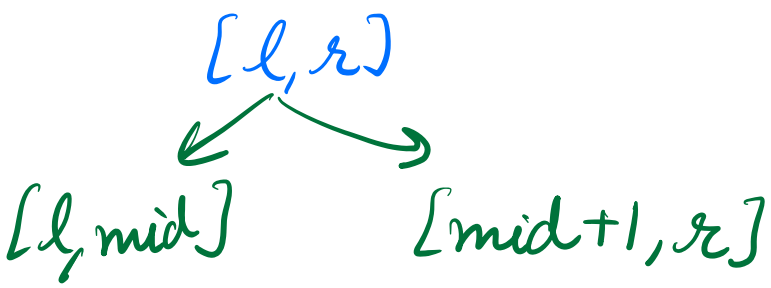
$$2^0 + 2^1 + 2^2 + \cdots + 2^{\log n}$$

$$\frac{a(r^n - 1)}{r - 1} = \frac{1(2^{\log n + 1} - 1)}{2 - 1}$$

$$= 2N - 1 = O(N)$$

- TC of query : $O(\log n)$
- TC of update : $O(\log n)$





Implementation using arrays

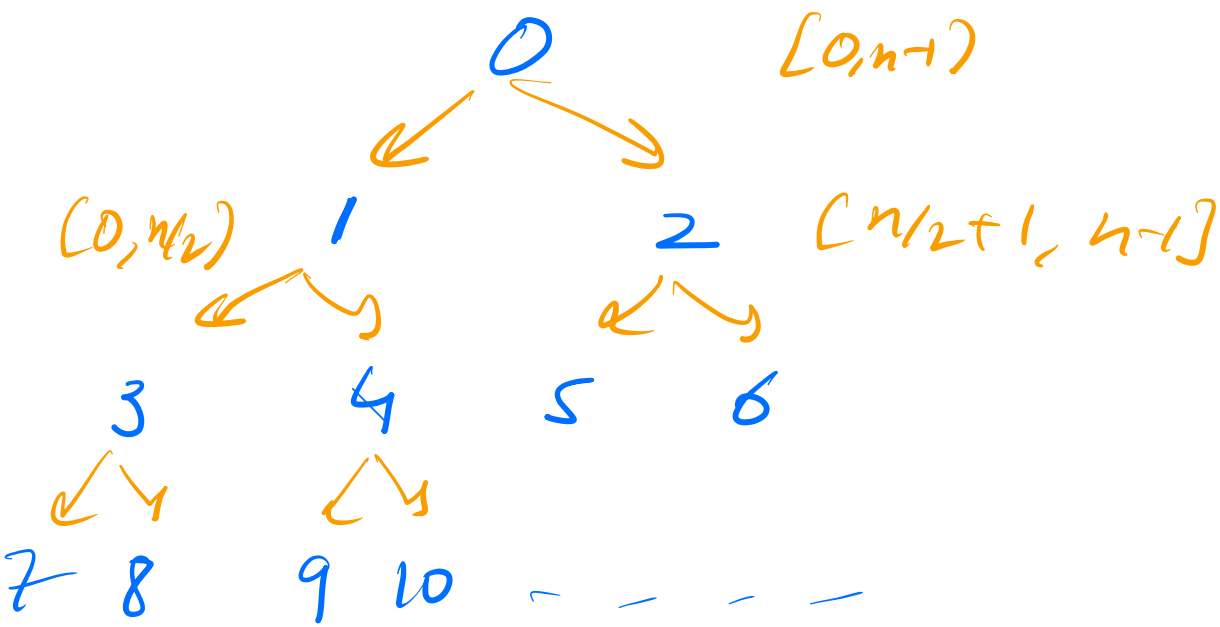
1) tree [4 N]

root → 0, n-1 → 0

i → left child → 2i + 1

i → right child → 2i + 2

i → parent → (i-1)/2



```
void build ( idx , start, end ) {
    if (start == end) {
        tree[idx] = a[start]
        return;
    }
    else {
        mid = (start+end)/2
        lc = 2*idx+1     rc = 2*idx+2
        build (lc, start, mid)
        build (rc, mid+1, end)
        tree[idx] = min(tree[lc],
                        tree[rc])
    }
}
```

current range in consideration

```
int query ( int idx, int x, int y,
            int l, int r ) {
    if ( x >= l && y <= r )
        return tree[idx]
    if ( x > r || y < l )
        return INT_MAX
    mid = (x+y)/2
    return min ( query ( 2idx+1, x, mid, l, r),
                 query (2idx+2, mid+1, y, l, r))
}
```

query range

```
void update( int idx, int i, int val,
             int l, int r ){
                 query range
  if ( l==r) {
     a[i] = val
     tree[idx] = val
  }
  else {
     mid= (l+r)/2
     lc = 2idx +1      rc= 2idx +2
     if( x≤ i     && ( i ≤ mid)
       update( lc , i , val , x, mid)
     else
       update ( rc , i , val, mid+1 , y )
  tree [idx] = min (tree[lc],
                    tree [rc])
}
```

0, 7

0, 3

2, 3

2, 2

done