


# # Segment Tree

## Concepts & Qns... #



Facebook  
Instagram } → code story with MIK  
(Twitter) → CS with MIK  
code story with MIK → 

"No more fear of Segment Tree"

"If you are afraid of something,  
go for it" 

video - ③

## Recap :-

- we understood about segment Tree ? what ? why ? when ?
- buildSegmentTree
- Example - Range Sum in an array
- Update Query

## Query in Segment Tree

# Example:- Range Sum Query #

↓ ↓  
nums = { 3, 1, 2, 7, 2, 1, 2, 3 }

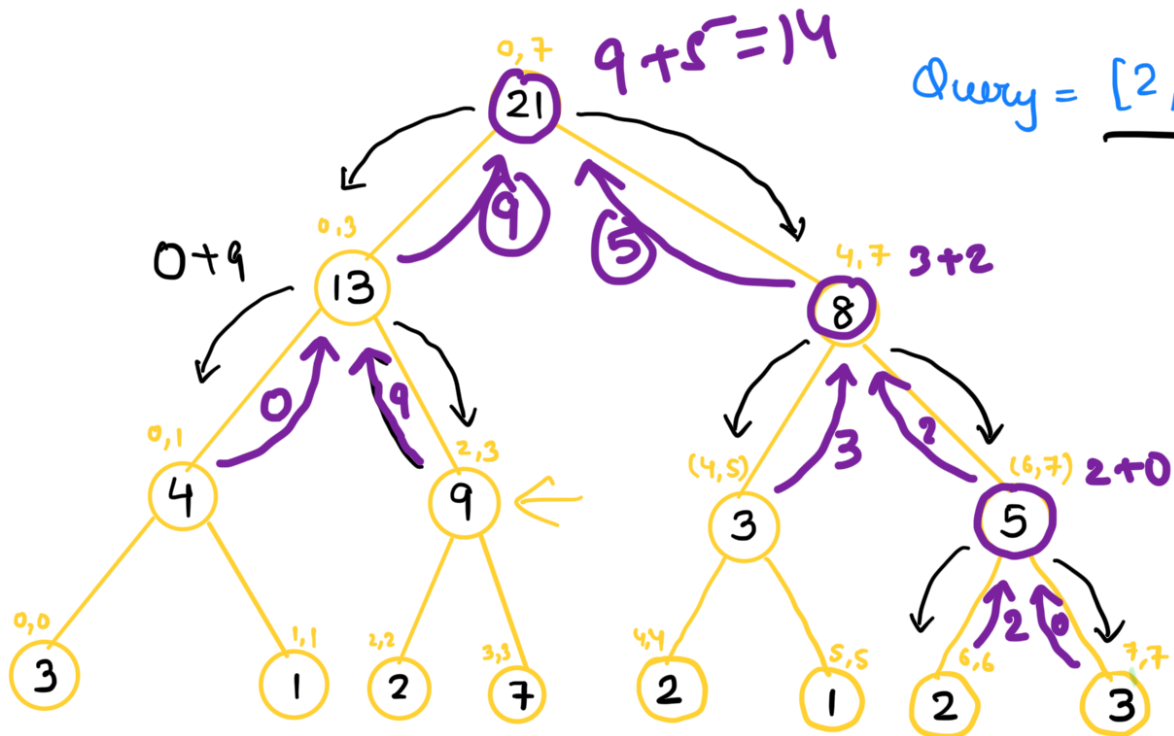
Find Sum → [2, 4] = 2 + 7 + 2

Find Sum → [1, 5] = 1 + 2 + 7 + 2 + 1

Find Sum → [2, 6] = . . . .

etc...

buildSegmentTree (0, 0, 7)



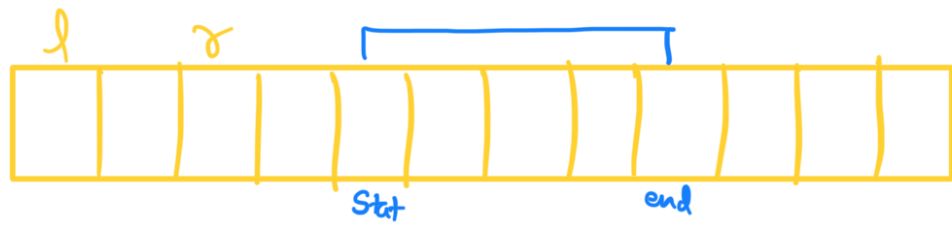
{ 3, 1, 2, 7, 2, 1, 2, 3 }

14

[l, r]

Query = [start, end]



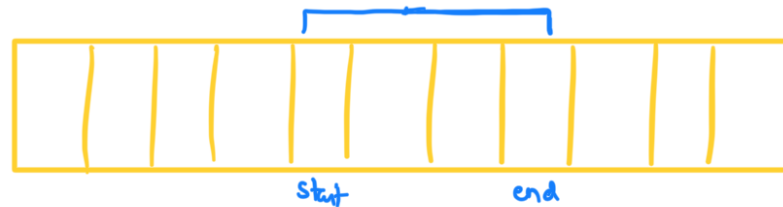


① Out of bound  $\rightarrow$  return 0 ;

```

if ( l > end || r < start ) {
    return 0 ;
}

```



②  $[l, r]$  is entirely inside  $[start, end]$

```

return segTree[i] ;

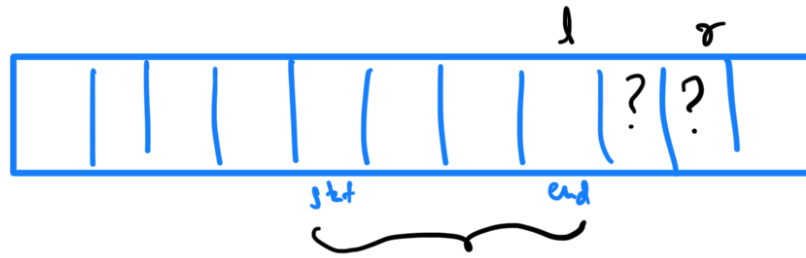
```

```

if ( l >= start && r <= end ) {
    return segTree[i] ;
}

```

③ else:



Overlapping :-

return Query(left value) + Query(right value);

Tree Diagram (STORY)  
to  
Code ...

Query(start, end, i, l, r);

int Query(start, end, i, l, r) {

if (l > end || r < start) return 0;

if (l > end || r < start) // out of bound.

return 0;

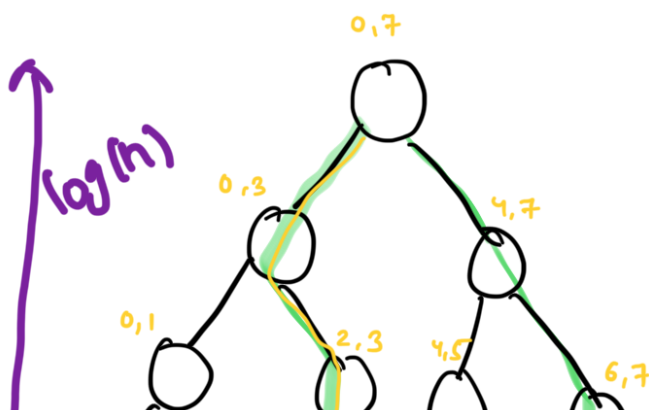
if (l >= start && r <= end) {  
    return segtree[i];  
}

return Query(start, end, 2\*i+1, l, mid)

Query(start, end, 2\*i+2, mid+1, r);

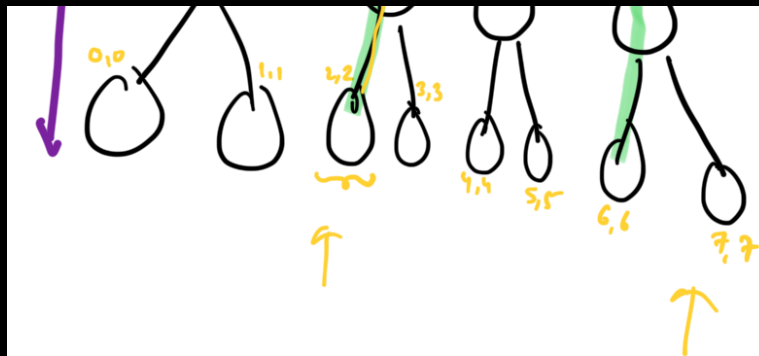
}

# Time Complexity :-



Query [start, end]

[2, 6]



2, 3, 4, 5, 6

$\log(n)$