PW SKILLS

Segment Trees

Lecture 1

Today's checklist

1. Requirement of Segment Trees
2. Maximum element in a given Range
3. Range Sum Query - Mutable
4. Range Frequency Queries

Prerequisites :

Recursion, Binary Trees, Implementation of MinHeap / MaxHeap

Segment trees → CP

→ Google

What and Why?

PW SKILLS

↓

User made Data Structure

→ Range Queries questions can be solved very efficiently using Segment trees

	0	1	2	3	4	5	6	7
arr = {	1,	4,	2,	8,	6,	4,	9,	3 }

2^{nd} – 6^{th} index tak ka maximum ⇒ 9

from l to r if I want to get max then **T.C. = O(r-l)**

= O(n)

What and Why?

PW SKILLS

What if we have an array & we run multiple queries on this array.

'q' queries

	0	1	2	3	4	5	6	7
arr =	1,	4,	2,	8,	6,	4,	9,	3

l1	r1
l2	r2
l3	r3
l4	r4

q queries

T.C. = O(qn)

10^5 10^5

TLE error

Build Segment Tree

Ques:

	0	1	2	3	4	5	6	7
arr = {	1,	4,	2,	8,	6,	4,	9,	3 }

PW SKILLS

Q1 : Find the Maximum element in a given Range. → [l, r]

(Multiple Queries)

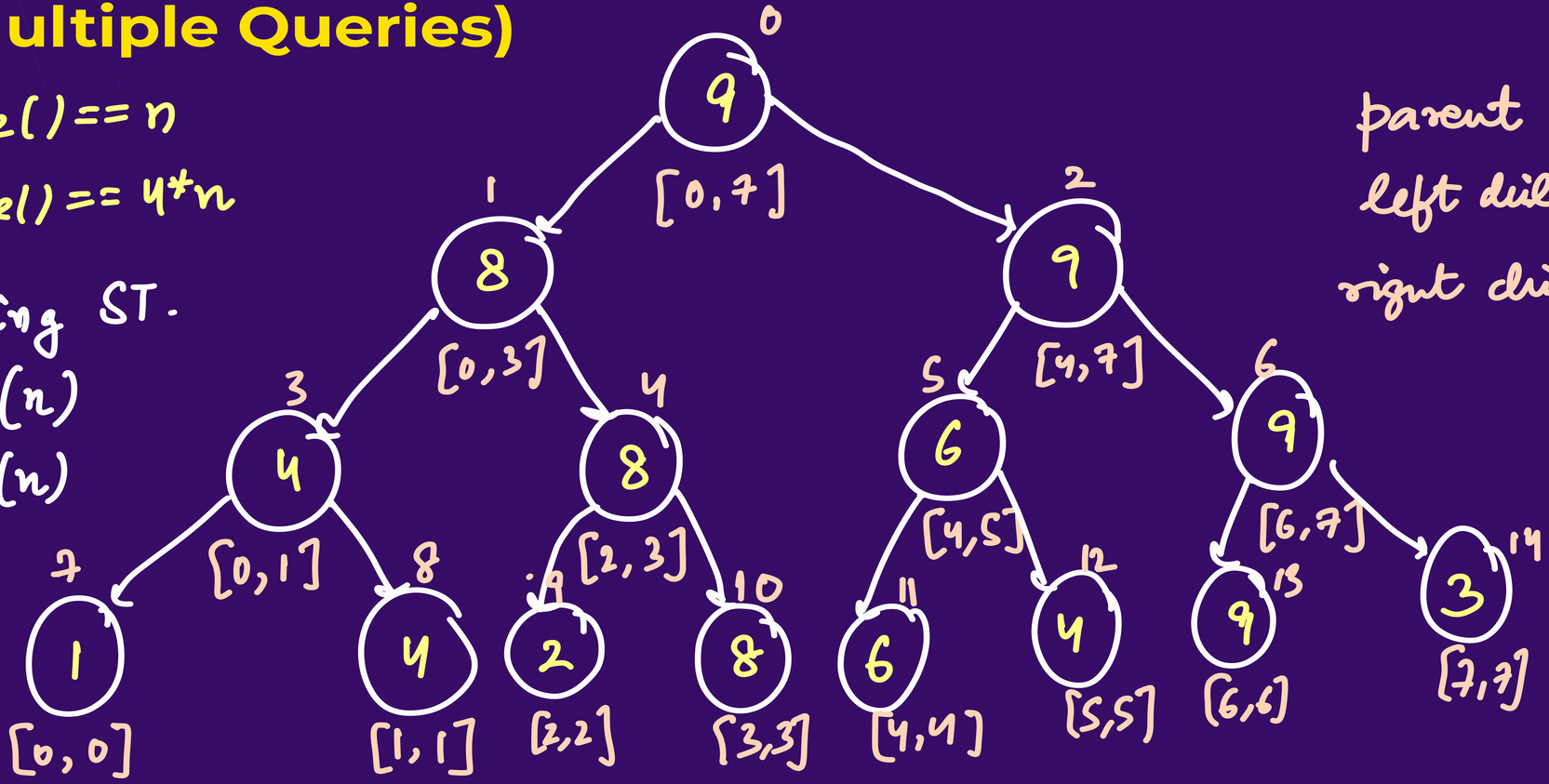
if arr.size() == n

st.size() == 4*n

For Building ST.

T.C. = O(n)

S.C. = O(n)



parent → i

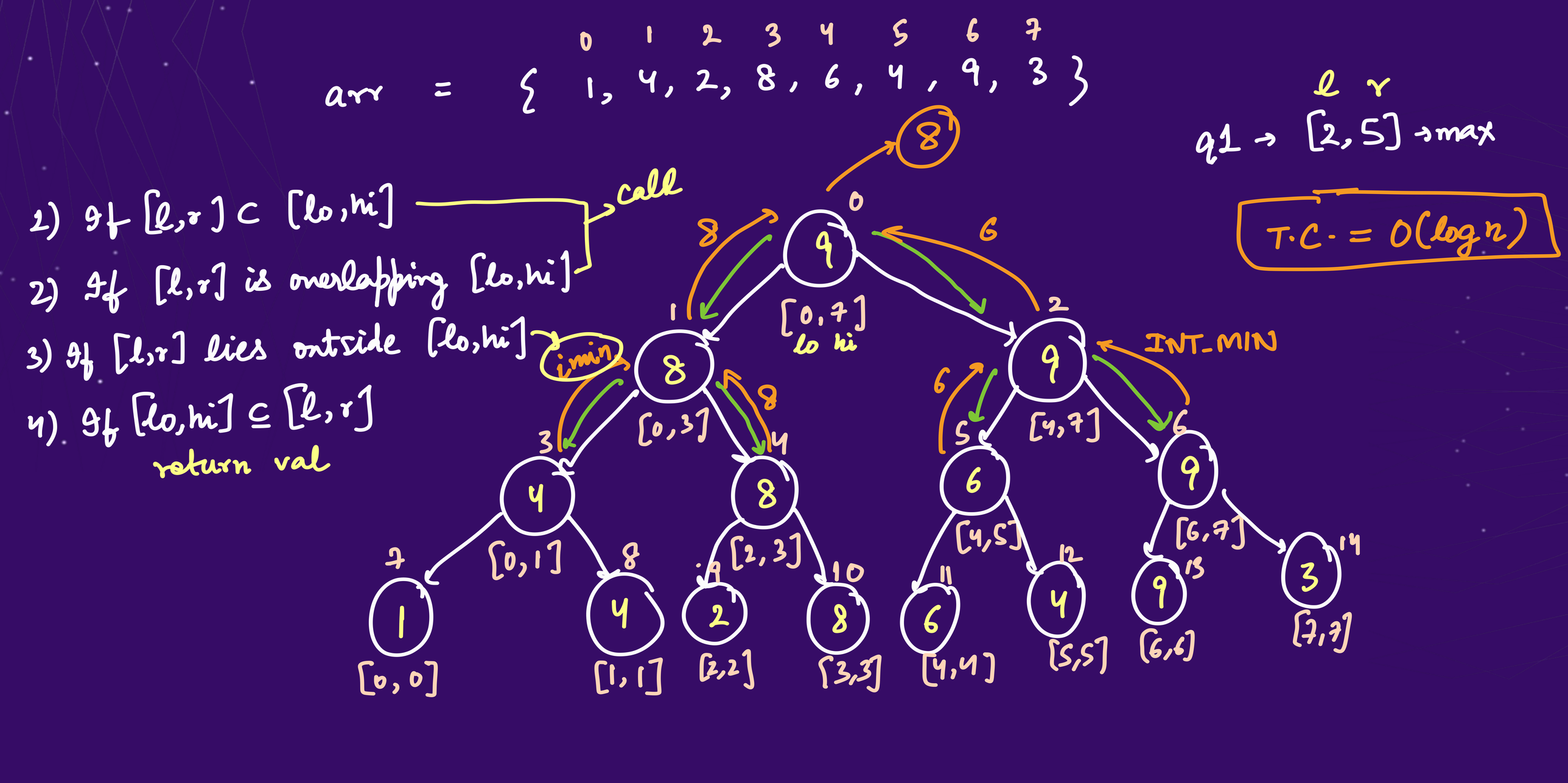
left child → 2i+1

right child → 2i+2

Ques:

Q1 : Find the Maximum element in a given Range.

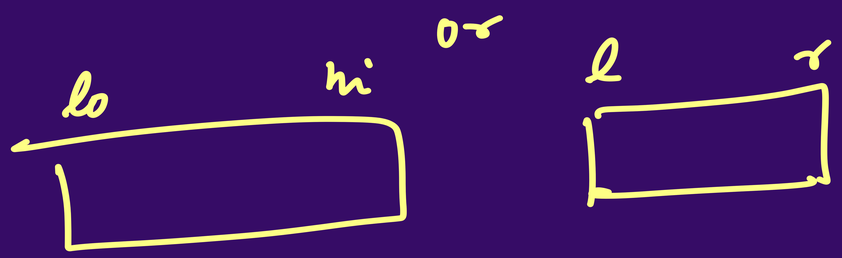
```cpp
vector<int> st;
void buildTree(int arr[], int i, int lo, int hi){
    if(lo==hi){ // base case
        st[i] = arr[lo];
        return;
    }
    int mid = lo + (hi-lo)/2; // (lo+hi)/2
    buildTree(arr,2*i+1,lo,mid); // left subtree
    buildTree(arr,2*i+2,mid+1,hi); // right subtree
    st[i] = max(st[2*i+1],st[2*i+2]);
}
int main(){
    int arr[] = {1,4,2,8,6,4,9,3}; // 0 to 7
    int n = sizeof(arr)/4;
    st.resize(4*n);
    buildTree(arr,0,0,n-1);
}
```


0 1 2 3 4 5 6 7
arr = { 1, 4, 2, 8, 6, 4, 9, 3 }
l r
q1 → [2, 5] → max
T.C. = O(log n)
1) If [l,r] ⊂ [lo,hi]
2) If [l,r] is overlapping [lo,hi]
3) If [l,r] lies outside [lo,hi]
4) If [lo,hi] ⊆ [l,r]
return val
call
imin
INT_MIN
lo hi
[0,7]
[0,3]
[4,7]
[0,1]
[2,3]
[4,5]
[6,7]
[0,0]
[1,1]
[2,2]
[3,3]
[4,4]
[5,5]
[6,6]
[7,7]

Outside lie → INT_MIN

if (r < lo)

or



if (l > hi)

Subset

if $[lo, hi] \subseteq [l, r]$

or

if(lo >= l && hi <= r) return val → st[i]

getMax

```
int getMax(int i, int lo, int hi, int& l, int& r){
    if(l>hi || r<lo) return INT_MIN;
    if(lo>=l && hi<=r) return st[i];
    int mid = lo + (hi-lo)/2; // (lo+hi)/2
    int leftMax = getMax(2*i+1,lo,mid,l,r);
    int rightMax = getMax(2*i+2,mid+1,hi,l,r);
    return max(leftMax,rightMax);
}
```

Homework:

PW SKILLS

Q : Find the Minimum element in a given Range. → of a given array (Mutable)

(Multiple Queries) (Almost Same)

What we did → same array → multiple queries

constructed a st

getMin(l, r)

update(idx, val)

Ques:

Prefix Sum : T.C. = O(n+q)
S.C. = O(n)

PW SKILLS

Q2 : Range Sum Query - Mutable

```
          0  1  2  3  4  5  6  7
arr  =  { 1, 4, 2, 8, 6, 4, 9, 3 }
```

sum from [2,5] = sum from [0,5] - sum from [0,1]

```
            l              r
pre  =  { 1, 5, 7, 15, 21, 25, 34, 37 }
```

[l,r] = pre[r] - pre[l-1]

pre[r] → 0,r pre[l-1] → 0,l-1

[Leetcode 307]

Ques:

Q2 : Range Sum Query - Mutable

	0	1	2	3	4	5	6	7
nums =	1	4	~~2~~ 10	8	6	4	9	3

O(1) ← Sum(2, 6)

O(1) ← Sum(1, 4)

O(1) ← Sum(0, 5)

O(n) ← update(2, 10)

arr = {1, 5, 7, 15, 21, 25, 34, 37}

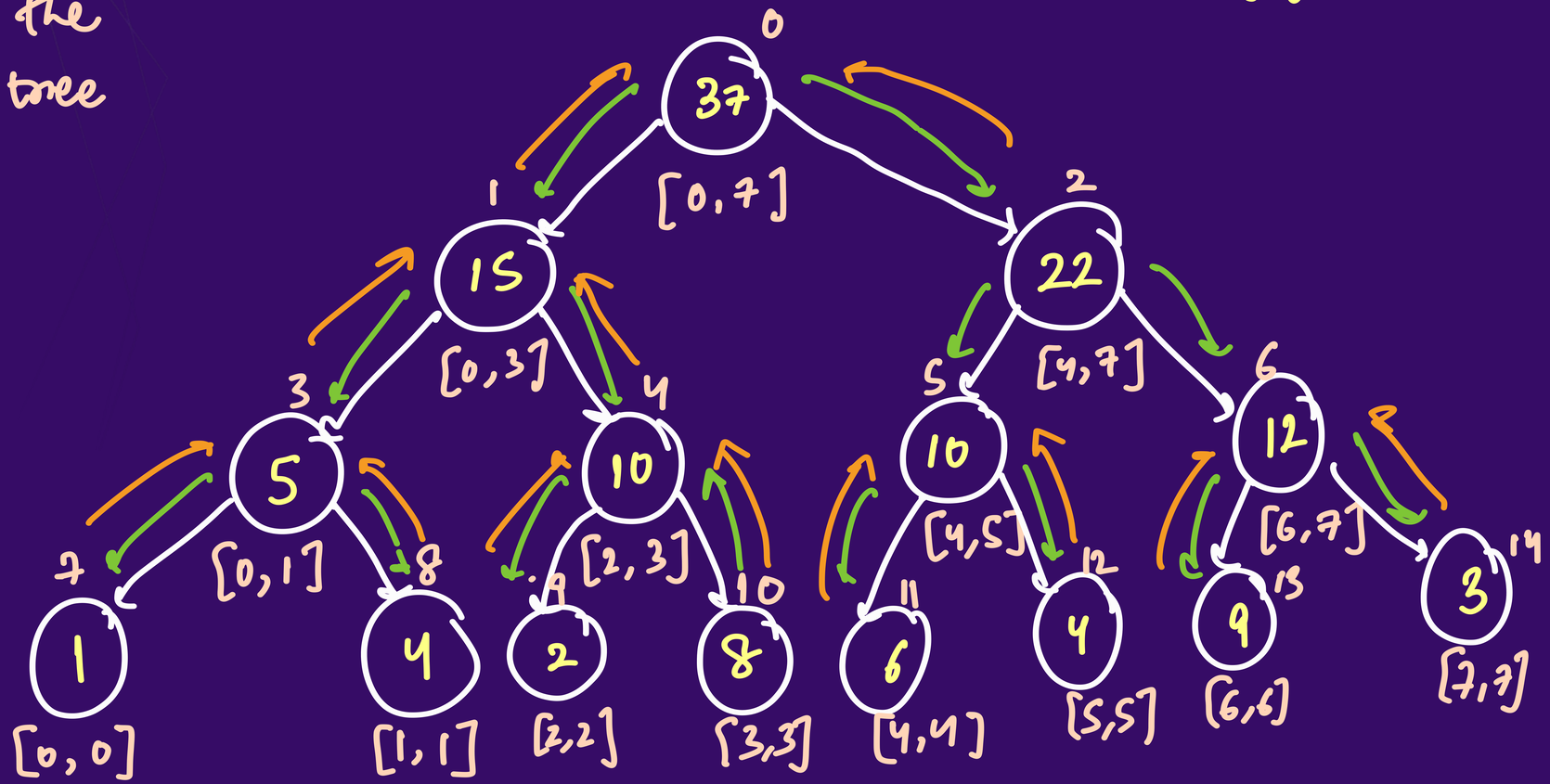
If we use prefix sum array, then in each 'update' query, we need to change our prefix sum array

[Leetcode 307]

	0	1	2	3	4	5	6	7
arr = {	1,	4,	2,	8,	6,	4,	9,	3 }

$st[i] = st[2i+1] + st[2i+2];$

1) Building the segment tree



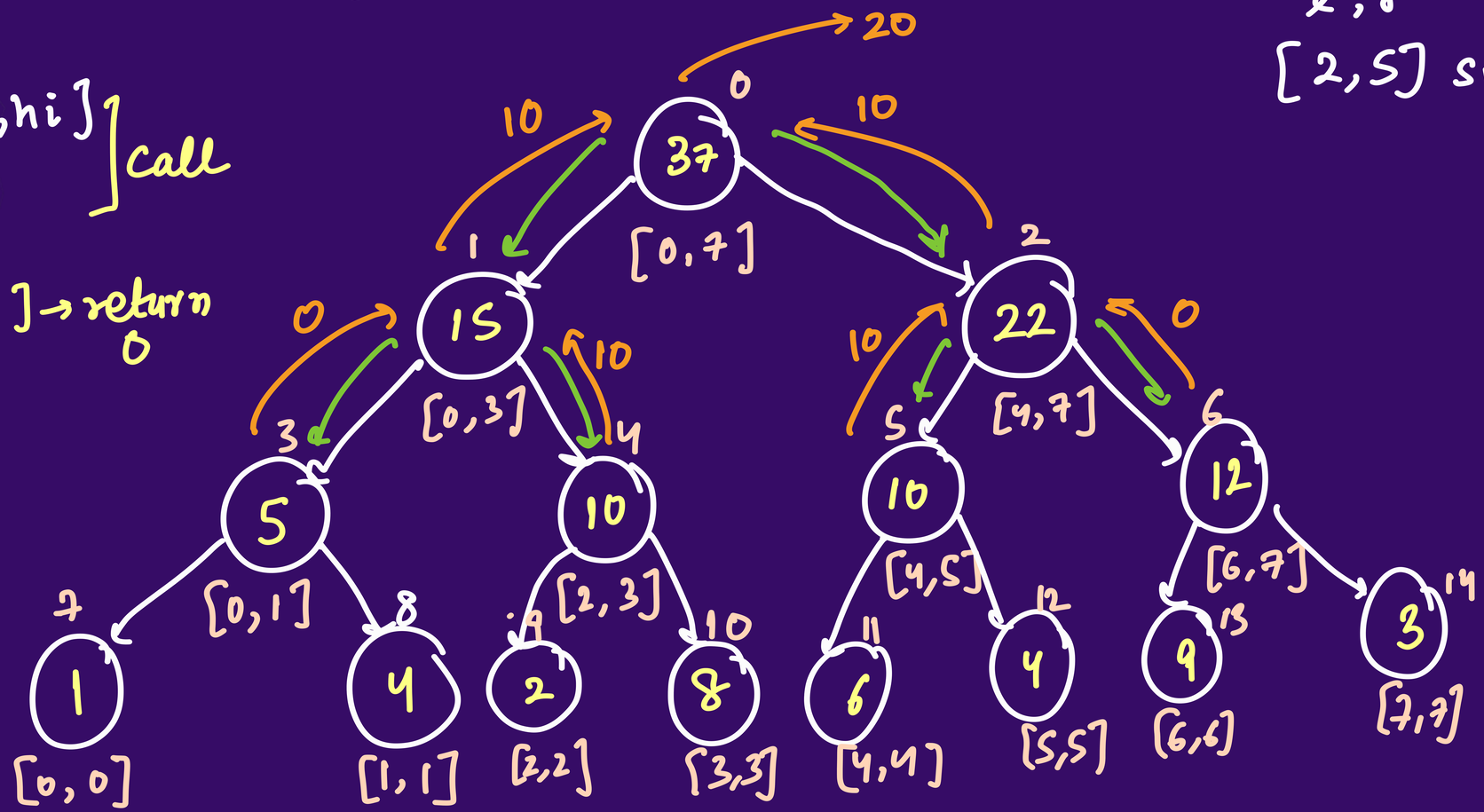
getSum(l, r)

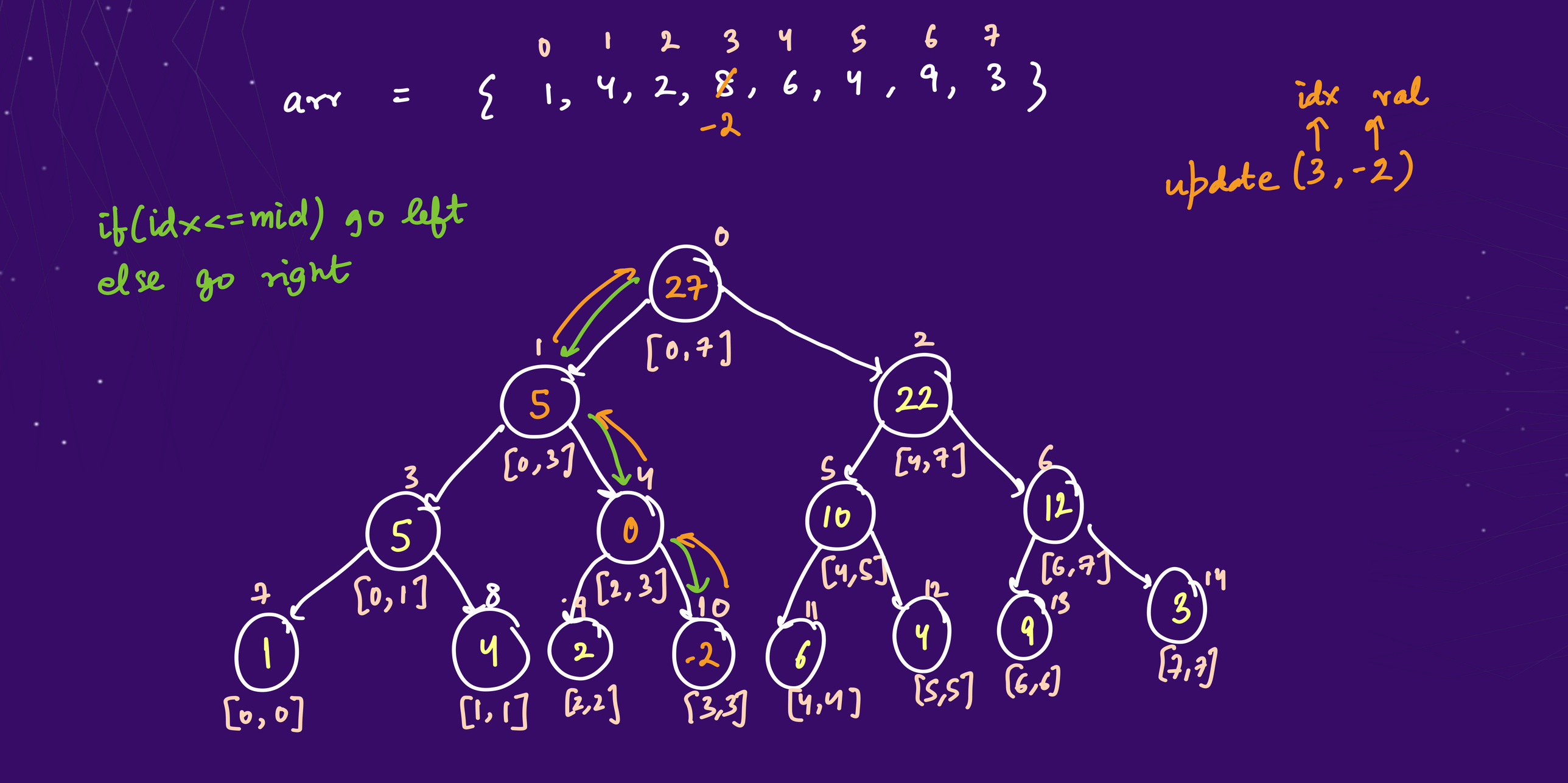
arr = { 1, 4, 2, 8, 6, 4, 9, 3 }

0	1	2	3	4	5	6	7
1	4	2	8	6	4	9	3

l, r
[2,5] sum

1) [l,r] ⊂ [lo,hi]] call
2) overlapping
3) lie outside] → return 0
4) [lo,hi] ⊆ [l,r]
↓
return st[i]




0 1 2 3 4 5 6 7
arr = { 1, 4, 2, 8, 6, 4, 9, 3 }
-2
idx val
update (3, -2)
if(idx<=mid) go left
else go right
0
27
[0,7]
1
5
[0,3]
2
22
[4,7]
3
5
[0,1]
4
0
[2,3]
5
10
[4,5]
6
12
[6,7]
7
1
[0,0]
8
4
[1,1]
9
2
[2,2]
10
-2
[3,3]
11
6
[4,4]
12
4
[5,5]
13
9
[6,6]
14
3
[7,7]

Build Tree

```
vector<int> st;
int n;
NumArray(vector<int>& nums) {
    n = nums.size();
    st.resize(4*n);
    buildTree(nums,0,0,n-1);
}

void buildTree(vector<int>& nums, int i, int lo, int hi){
    if(lo==hi){
        st[i] = nums[hi];
        return;
    }
    int mid = lo + (hi-lo)/2;
    buildTree(nums,2*i+1,lo,mid);
    buildTree(nums,2*i+2,mid+1,hi);
    st[i] = st[2*i+1] + st[2*i+2];
}
```

```
void updateVal(int i, int lo, int hi, int& index, int& val){
    if(lo==hi){
        st[i] = val;
        return;
    }
    int mid = lo + (hi-lo)/2;
    if(index<=mid) updateVal(2*i+1,lo,mid,index,val);
    else updateVal(2*i+2,mid+1,hi,index,val);
    st[i] = st[2*i+1] + st[2*i+2];
}

void update(int index, int val) { // extra
    updateVal(0,0,n-1,index,val);
}
```

```
int getSum(int i, int lo, int hi, int& l, int& r){
    if(l>hi || r<lo) return 0;
    if(lo>=l && hi<=r) return st[i];
    int mid = lo + (hi-lo)/2; // (lo+hi)/2
    int leftSum = getSum(2*i+1,lo,mid,l,r);
    int rightSum = getSum(2*i+2,mid+1,hi,l,r);
    return leftSum + rightSum;
}

int sumRange(int left, int right) {
    return getSum(0,0,n-1,left,right);
}
```

Ques: Each node of st will be a map.

PW SKILLS

Q3 : Range Frequency Queries

	0	1	2	3	4	5	6	7
arr = {	2,	4,	2,	4,	6,	4,	2,	6 }

query(0, 2, 10) → 0

query(1, 5, 4) → 3 ans

l, r, ele

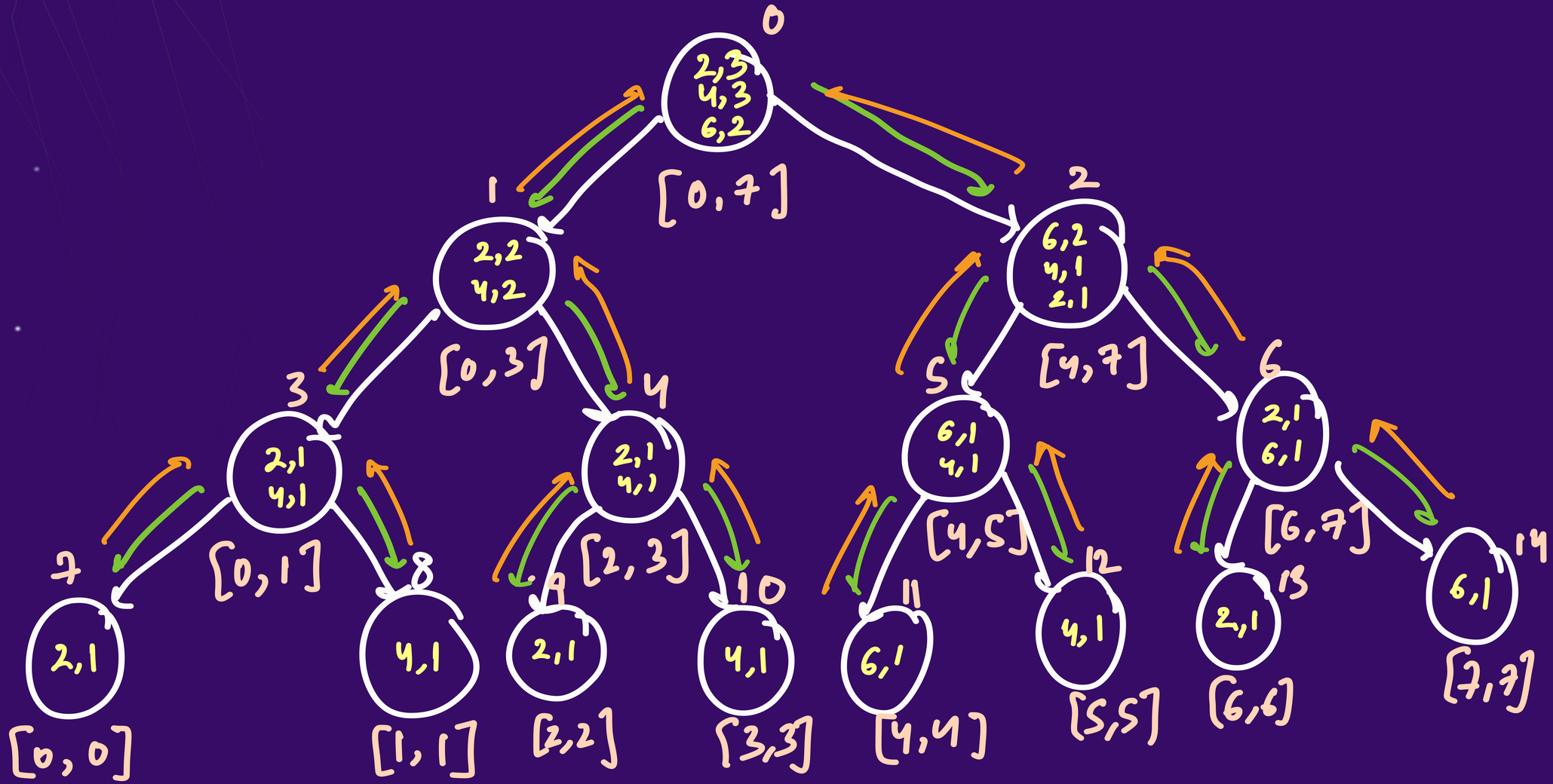
Hashmap

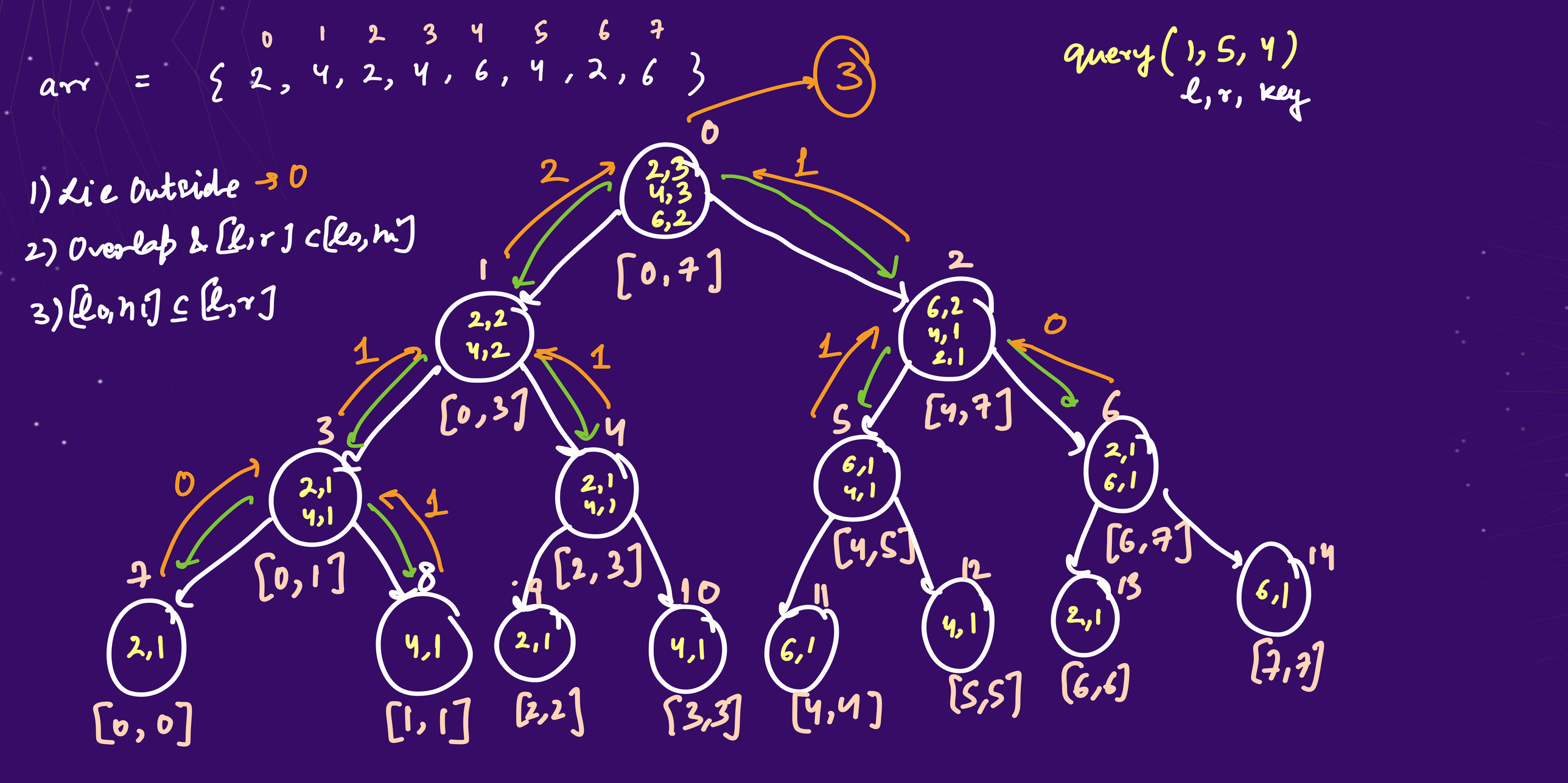
2	3
4	3
6	2

[Leetcode 2080]

arr = { 2, 4, 2, 4, 6, 4, 2, 6 }
(0 1 2 3 4 5 6 7)

st[i] is a map




arr = { 2, 4, 2, 4, 6, 4, 2, 6 }
0 1 2 3 4 5 6 7
query (1, 5, 4)
l, r, key
1) Lie Outside → 0
2) Overlap & [l,r] ⊂ [lo,hi]
3) [lo,hi] ⊆ [l,r]
3
0
2,3
4,3
6,2
[0,7]
2
1
1
2,2
4,2
[0,3]
2
6,2
4,1
2,1
[4,7]
1
1
1
0
3
4
5
6
2,1
4,1
[0,1]
2,1
4,1
[2,3]
6,1
4,1
[4,5]
2,1
6,1
[6,7]
0
1
7
8
9
10
11
12
13
14
2,1
[0,0]
4,1
[1,1]
2,1
[2,2]
4,1
[3,3]
6,1
[4,4]
4,1
[5,5]
2,1
[6,6]
6,1
[7,7]

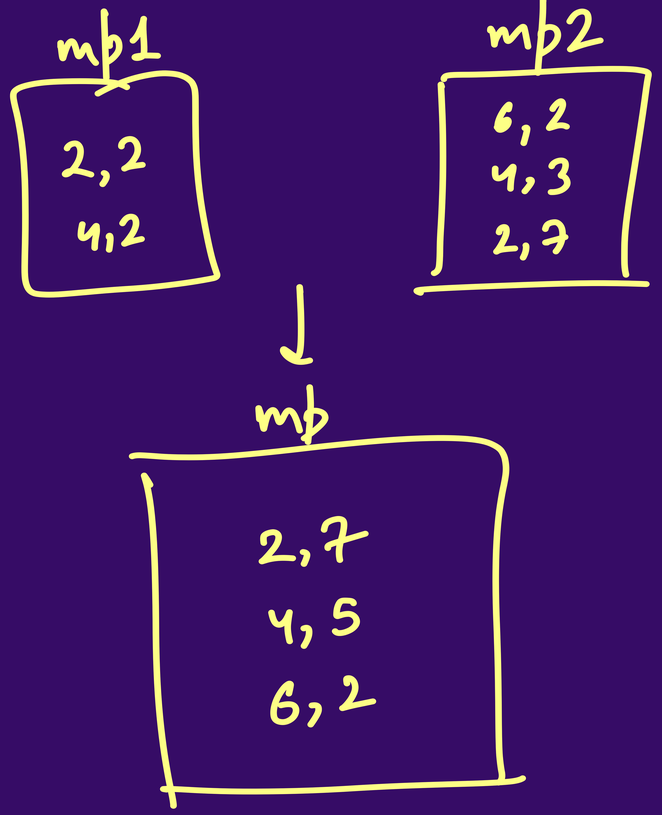
2, 5

↓

mp.insert({2,5});

or

mp[2] = 5



```cpp
vector< unordered_map<int,int> > st;
int n;

unordered_map<int,int> addMaps(unordered_map<int,int>& mp1, unordered_map<int,int>& mp2){
    unordered_map<int,int> mp;
    for(auto x : mp1){
        mp.insert(x);
    }
    for(auto x : mp2){
        int key = x.first, freq = x.second;
        if(mp.find(key)==mp.end()) // key not found in mp
            mp.insert(x);
        else mp[key] += freq;
    }
    return mp;
}
```

```
void buildTree(vector<int>& arr, int i, int lo, int hi){
    if(lo==hi){
        st[i][arr[lo]] = 1;
        return;
    }
    int mid = lo + (hi-lo)/2;
    buildTree(arr,2*i+1,lo,mid);
    buildTree(arr,2*i+2,mid+1,hi);
    st[i] = addMaps(st[2*i+1],st[2*i+2]);
}

RangeFreqQuery(vector<int>& arr) {
    n = arr.size();
    st.resize(4*n);
    buildTree(arr,0,0,n-1);
}
```

```
int getFreq(int i, int lo, int hi, int& l, int& r, int& key){
    if(l>hi || r<lo) return 0;
    if(lo>=l && hi<=r){
        if(st[i].find(key)==st[i].end()) return 0;
        else return st[i][key];
    }
    int mid = lo + (hi-lo)/2; // (lo+hi)/2
    int leftAns = getFreq(2*i+1,lo,mid,l,r,key);
    int rightAns = getFreq(2*i+2,mid+1,hi,l,r,key);
    return leftAns + rightAns;
}
int query(int left, int right, int value) {
    return getFreq(0,0,n-1,left,right,value);
}
```

PW SKILLS

THANK YOU