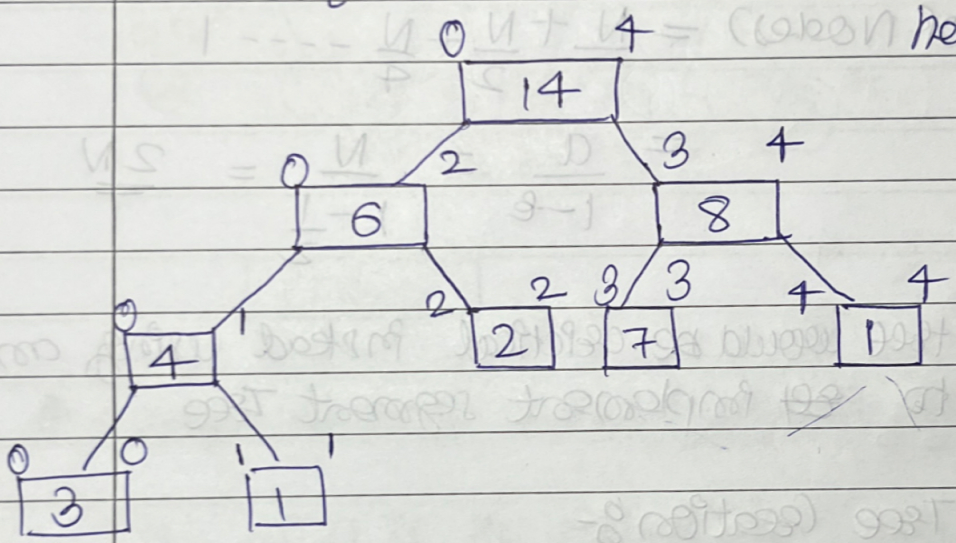
Segment Tree (Balanced Binary Tree)

__/__/____

- An efficient Data structure that allows
  - → Efficient Querying of Intervals / Range.
  - → Efficient Updating of Intervals / Range.

- Constructing Seg. Tree for an array,

0	1	2	3	4
3	1	2	7	1



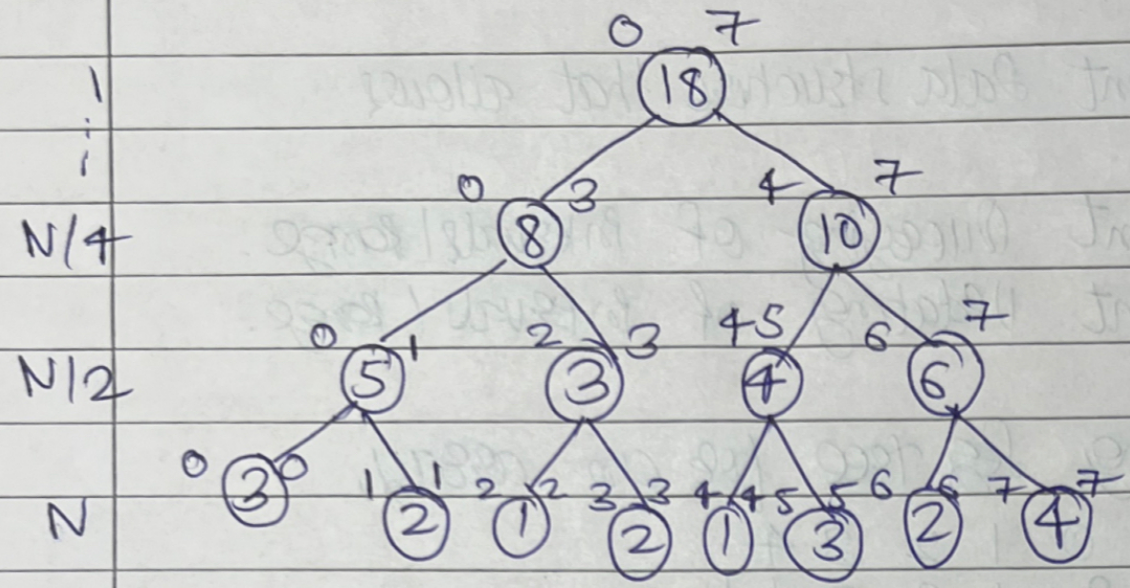
height = $\log_2 n$

- Every node in the segment tree is height-Balanced, called as Balanced Binary Tree.

- Segment Tree Creation :-
  - TC → $O(N)$
  - SC → $O(2N) + O(\log_2 N) \approx O(N)$

__/__/____

0 1 2 3 4 5 6 7
[3, 2, 1, 2, 1, 3, 2, 4]



Sum(No. of Nodes) $= \frac{N}{} + \frac{N}{2} + \frac{N}{4}$ ---- 1

$= \frac{a}{1-e} = \frac{N}{1-\frac{1}{2}} = \underline{\underline{2N}}$

- To create tree would be cedrical Instead using an array to ~~see~~ Implement segment Tree

- Segment Tree Creation :-

SEGMENT TREE

OUTLINE

TIMELINE

```
91  ll segmentTree[10000000001]={0};
92  void createSegmentTree(ll index,ll low,ll high,vd<ll>&arr){
93      if(low == high){
94          segmentTree[index]=arr[low];
95          return ;
96      }
97      ll mid = (low + high) >> 1;
98      createSegmentTree( (index << 1 ) + 1 , low,mid,arr);
99      createSegmentTree( (index << 1 ) + 2 , mid+1,high,arr);
100     segmentTree[index]=segmentTree[(index << 1 ) + 1]+segmentTree[(index << 1 ) + 2];
101 }
102 void itachi_1609(){
103     vd<ll>arr={3,1,2,7,1};
104     ll index=0,low=0,high=len(arr) - 1;
105     createSegmentTree(index,low,high,arr);
106     FOR( (len(arr) << 1 ))print(segmentTree[_]," ");
107     printn();
108     return;
109 }
```

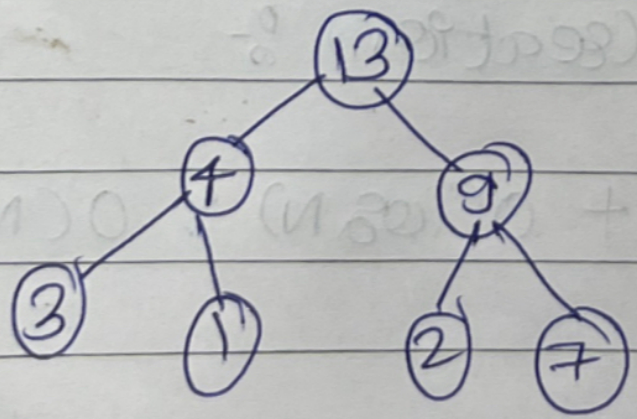
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

```
cd "/Users/bhushanthombare/Desktop/Segment Tree/" && g++ -std=c++17 tempCodeRunnerFile.cpp -o tempC
tempCodeRunnerFile
Segment Tree > cd "/Users/bhushanthombare/Desktop/Segment Tree/" && g++ -std=c++17 tempCodeRunnerFi
bare/Desktop/Segment Tree/"tempCodeRunnerFile
14   6   8   4   2   7   1   3   1   0
```

Segment Tree Updation :-

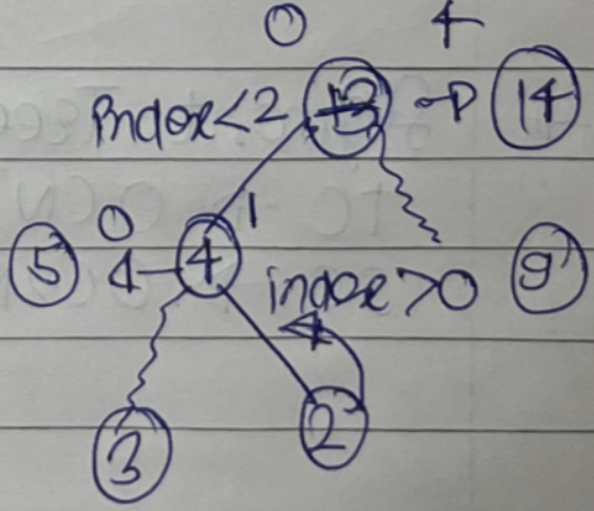
[3, 1, 2, 7]

0 1 2 3



Updation (index)

[3, 2, 2, 7]



O(log N) For Updation.

```cpp
void update(ll index,ll low,ll high,ll &indexChange,ll &changeVal){
    if(low == high){
        if(low == indexChange)segmentTree[index]=changeVal;
        return ;
    }
    ll mid = (low + high) >> 1;
    update( (index << 1 ) + 1 , low,mid,indexChange,changeVal);
    update( (index << 1 ) + 2 , mid+1,high,indexChange,changeVal);
    segmentTree[index]=segmentTree[(index << 1 ) + 1]+segmentTree[(index << 1 ) + 2];
}
void itachi_1609(){
    vd<ll>arr={3,1,2,7};
    ll index=0,low=0,high=len(arr) - 1;
    createSegmentTree(index,low,high,arr);
    FOR( (len(arr) << 1 ))print(segmentTree[_]," ");
    index=0;
    low=0;
    high=len(arr) - 1;
    ll indexChange=1,changeVal=2;
    update(index,low,high,indexChange,changeVal);
    printn();
    FOR( (len(arr) << 1 ))print(segmentTree[_]," ");
    printn();
    return;
}
```

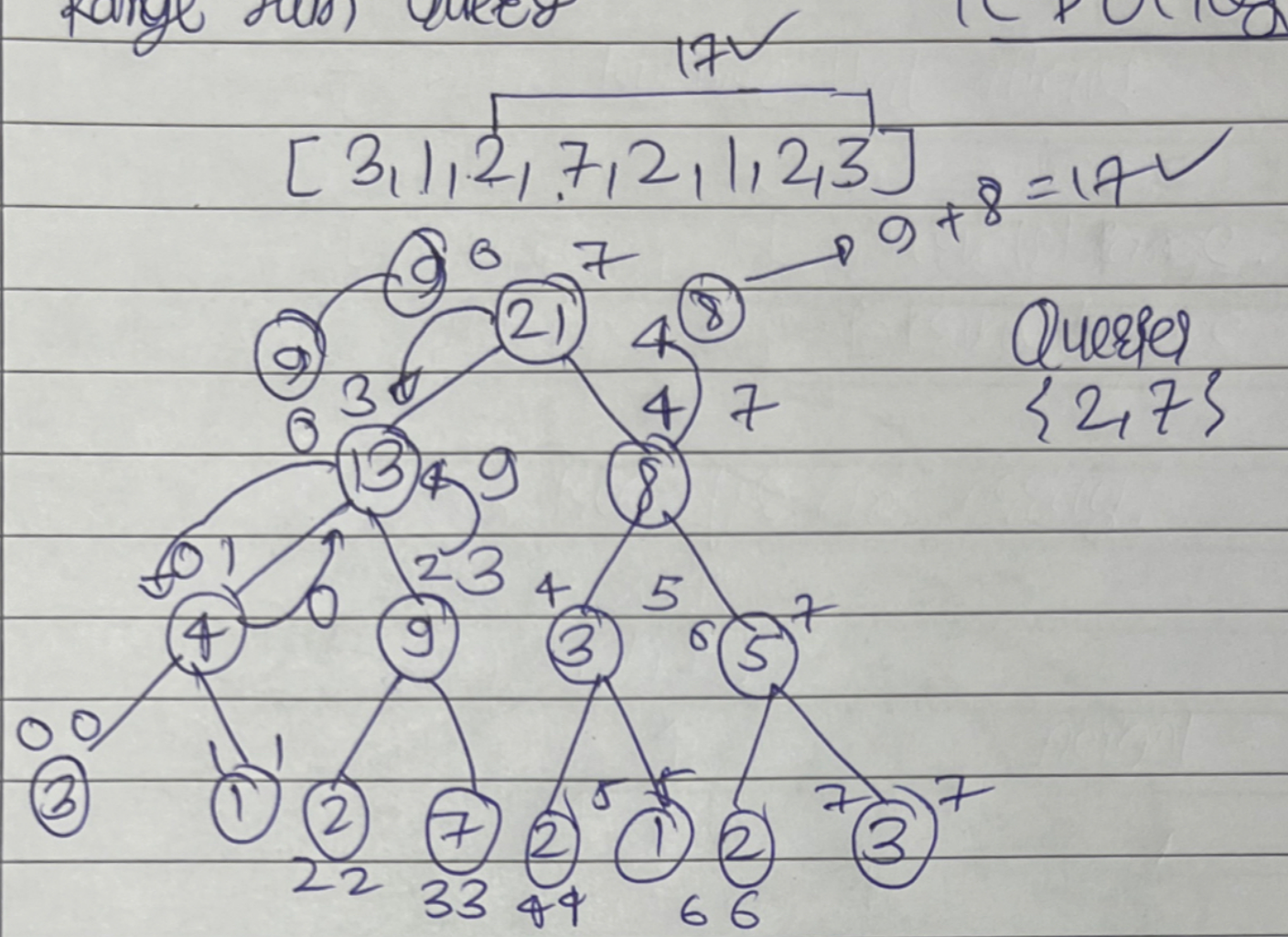
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

```
cd "/Users/bhushanthombare/Desktop/Segment Tree/" && g++ -std=c++17 tempCodeRunnerFil
gment Tree/"tempCodeRunnerFile
Segment Tree > cd "/Users/bhushanthombare/Desktop/Segment Tree/" && g++ -std=c++17 te
bare/Desktop/Segment Tree/"tempCodeRunnerFile
13   4   9   3   1   2   7   0
14   5   9   3   2   2   7   0
Segment Tree >
```

__/__/____

Range Sum Query

TC → O(log N)



OUTLINE

TIMELINE

```
112  ll rangeSumQuery(ll index,ll low,ll high,ll start,ll end){
113      if(low>end or high<start) return 0;
114      if(low>=start and high<=end)return segmentTree[index];
115      ll mid = (low + high) >> 1;
116      return rangeSumQuery((index << 1 ) + 1 ,low,mid,start,end) +
117      rangeSumQuery((index << 1 ) + 2 ,mid+1,high,start,end);
118  }
119  void itachi_1609(){
120      vd<ll>arr={3,1,2,7,2,1,2,3};
121      ll index=0,low=0,high=len(arr) - 1;
122      createSegmentTree(index,low,high,arr);
123      index=0;
124      low=0;
125      high=len(arr) - 1;
126      print("Sum is : ",rangeSumQuery(0,low,high,2,7));
127      printn();
128      return;
129  }
```

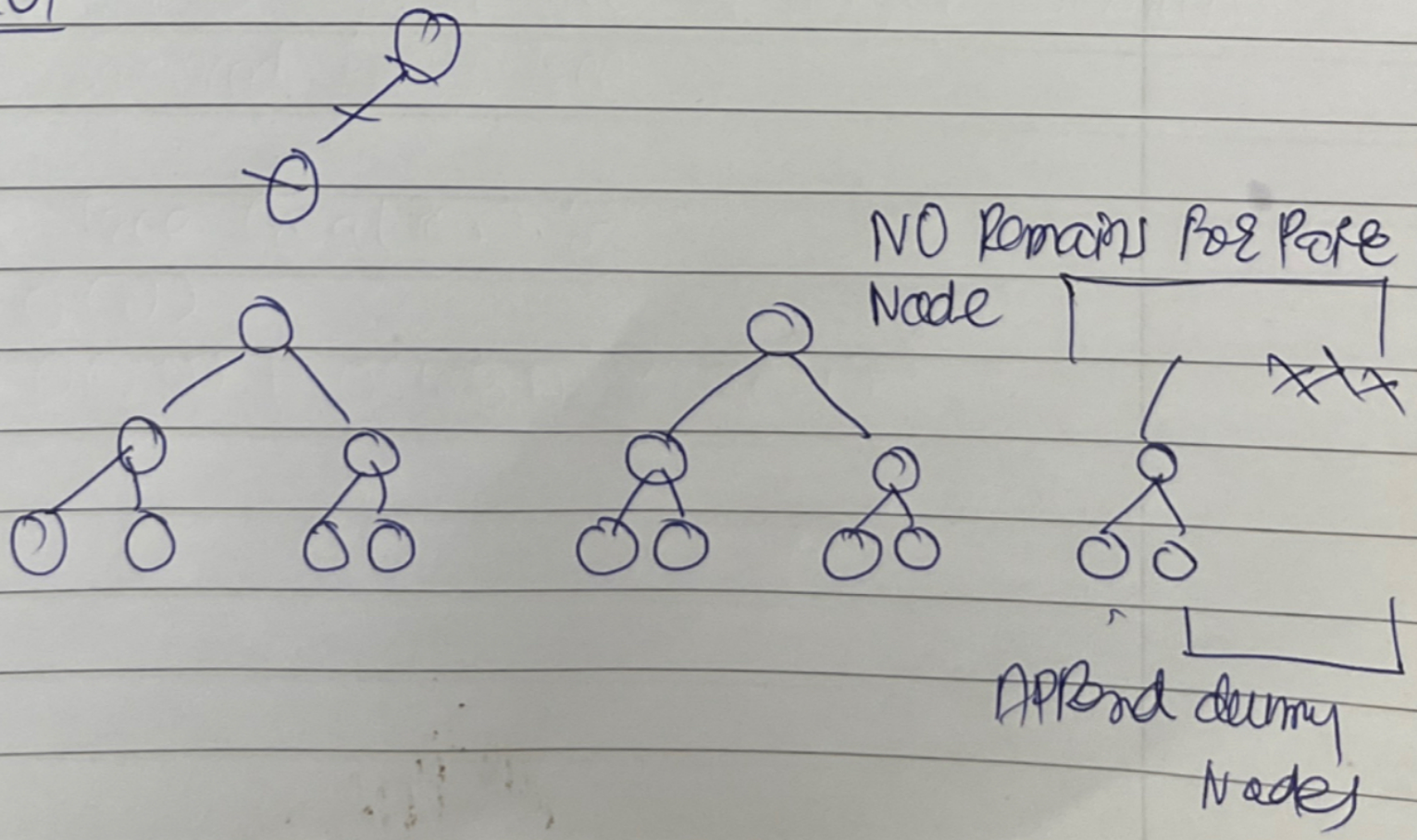
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

```
cd "/Users/bhushanthombare/Desktop/Segment Tree/" && g++ -std=c++17 tempCodeRunnerFile.cpp -o
gment Tree/"tempCodeRunnerFile
Segment Tree > cd "/Users/bhushanthombare/Desktop/Segment Tree/" && g++ -std=c++17 tempCodeRun
bare/Desktop/Segment Tree/"tempCodeRunnerFile
Sum is :  17
Segment Tree >
```

Why we have 4*n size array for segment Tree?

If the n is the power of 2, it will then all the nodes have value in between

n=10



4N

__/__/____

Range Update Query :- { using Lazy Propagation }

0 1 2 3 4 5 6 7

[3, 4, 2, 5, 3, 6, 3, 4]

4 7 5 8

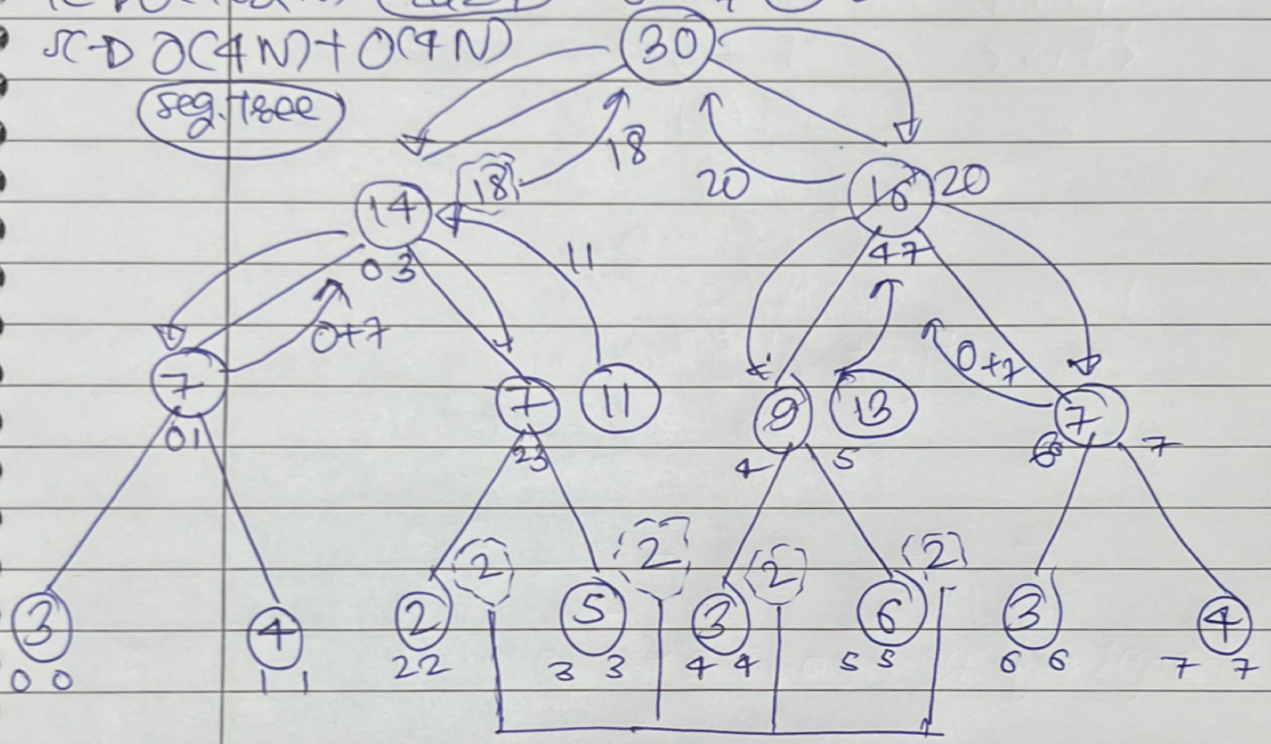
[2, 5] ~+2

start ↑ ↑ end

TC → O(log N) (Lazy) 0 7 38 ✓

SC → O(4N) + O(4N)

(Seg. Tree)



Lazy Tree Node

```
void updateRangeQuery(ll index,ll low,ll high,ll &start,ll &end,ll &val){
    if(lazy[index] != 0){
        segmentTree[index] += (high-low+1) * lazy[index];
        if(low != high){
            lazy[(index << 1) + 1] += lazy[index];
            lazy[(index << 1) + 2] += lazy[index];
        }
        lazy[index]=0;
    }
    if(start>high or end<low or low>high)return;
    if(low>=start and high<=end){
        segmentTree[index] += (high-low+1) * val;
        if(low != high){
            lazy[(index << 1) + 1] += val;
            lazy[(index << 1) + 2] += val;
        }
        return ;
    }
    ll mid=(low+high) >> 1;
    updateRangeQuery((index << 1) + 1,low,mid,start,end,val);
    updateRangeQuery((index << 1) + 2,mid+1,high,start,end,val);
    segmentTree[index]=segmentTree[(index << 1) + 1]+segmentTree[(index << 1) + 2];
}
ll rangeSumLazy(ll index,ll low,ll high,ll &start,ll &end){
    if(lazy[index] != 0){
        segmentTree[index]=(high-low+1) * lazy[index];
        if(low != high){
            lazy[(index << 1) + 1] += lazy[index];
            lazy[(index << 1) + 2] += lazy[index];
        }
        lazy[index]=0;
    }
    if(start>high or end<low or low>high)return 0;
    if(low>=start and high<=end) return segmentTree[index];
    ll mid=(low+high) >> 1;
    return rangeSumLazy((index << 1) + 1,low,mid,start,end)+
    rangeSumLazy((index << 1) + 2,mid+1,high,start,end);
}
```