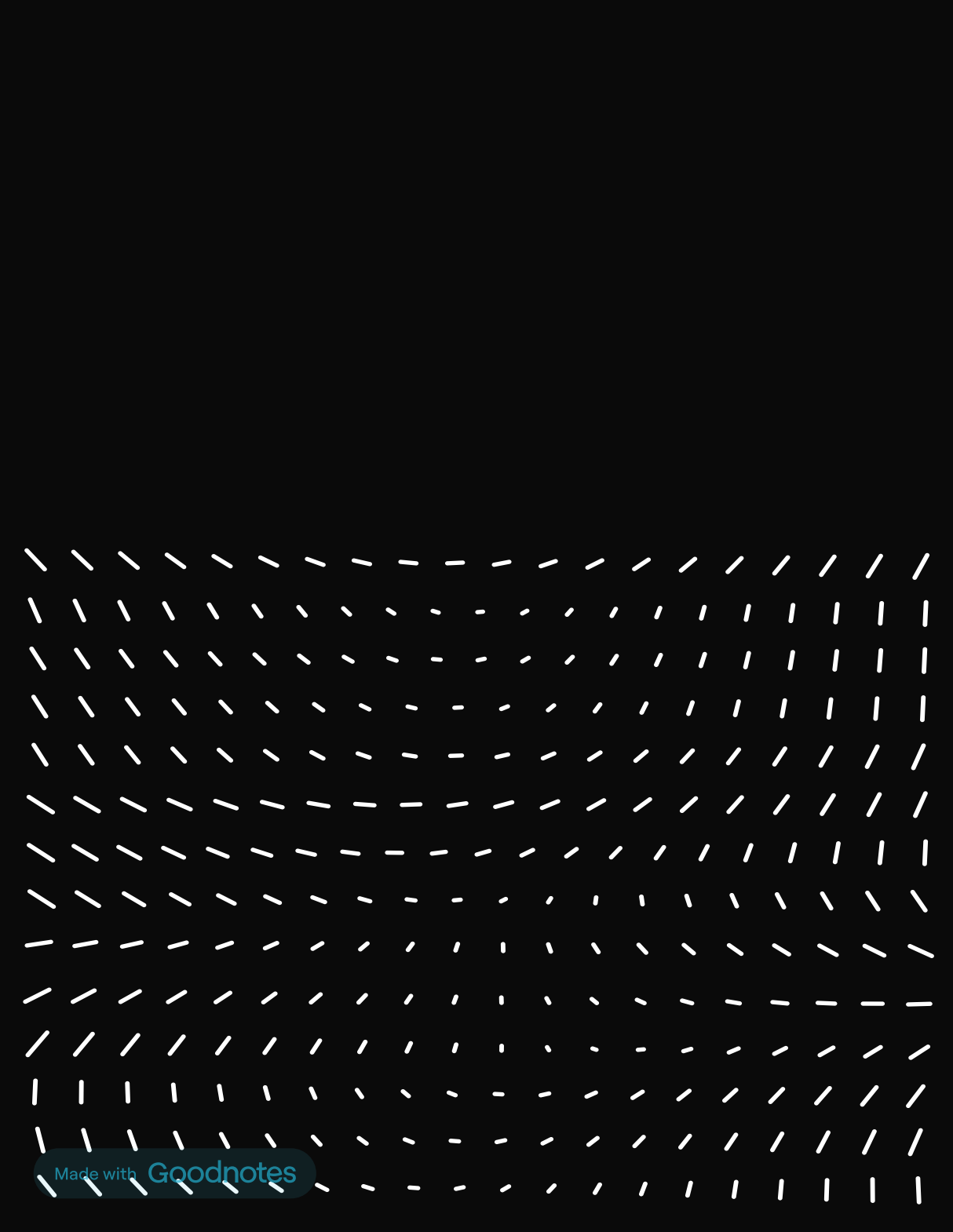
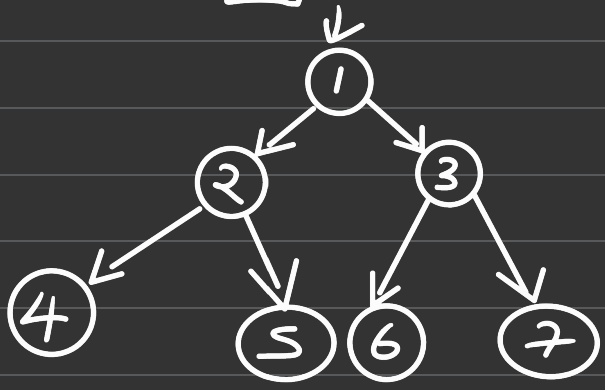
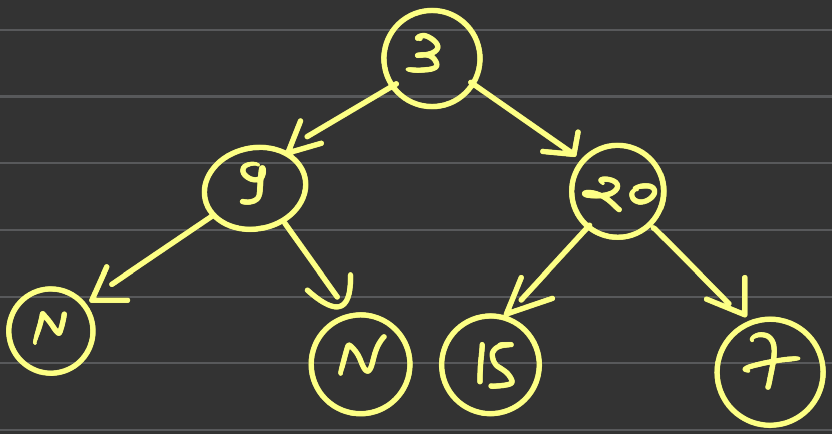


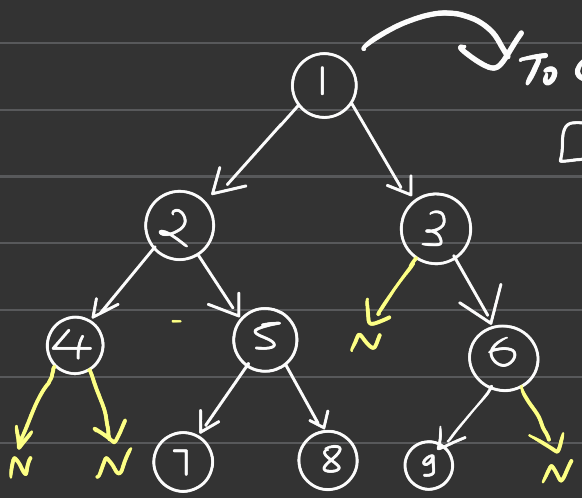
Construct tree from level order traversal..

Given an array, for Ex → 1 2 3 4 5 6 7



Input root = [3, 9, 20, null, null, 15, 7]





To write in Array form

[1,2,3,4,5,N,6,N,7,8,9,N]

[1,2,3,4,5,N,6,N,7,8,9,N]
i j

Queue<node*> q

1

Algorithm:-

⇒ Node*root = New node(arr[0]); ⇒ q.Push(1); ⇒ while q.size()>0

⇒ Node* l = new node(arr[i]);

⇒ Node* r = new node(arr[j]);

Now

Node* temp = q.front(); q.Pop();

temp -> left = l;

temp -> right = r;

Also push l and r to the Queue:-

i = i+2;

j = j+2;

Now Step by step breakdown :-

[1, 2, 3, 4, 5, N, 6, N, 7, 8, 9, N]
 i j

queue<node*> q

3 4 5

Step 1) root = new node(1);

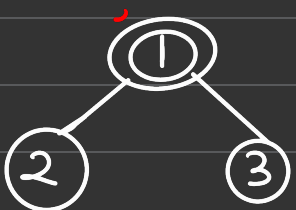
1 → And pushed in queue...

Btree Status:

Now we would create A new node

l = 2

r = 3



Step 2) Next iteration:

l = 4

r = 5

Btree STATUS:-

[1,2,3,4,5,N,6,N,7,8,9,N]
i j

queue<node*> q

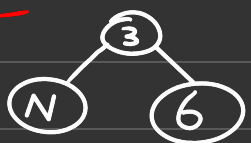
4 5 6

Step 3) Next iteration:

l = N

r = 6

Btree STATUS:-



[1,2,3,4,5,N,6,N,7,8,9,N]
i j

queue<node*> q

4 5 6

[1,2,3,4,5,N,6,N,7,8,9,N]
i j

queue<node*> q

5 6 7

Step4) Next iteration:

l = N

r = 7

Btree STATUS:-

[1,2,3,4,5,N,6,N,7,8,9,N]
i j

Queue<node*> q
5 6 7

[1,2,3,4,5,N,6,N,7,8,9,N]
i j

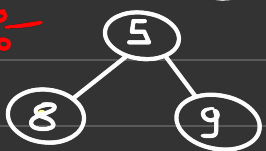
Queue<node*> q
6 7 8 9

Steps) Next iteration:

l = 8

r = 9

Btree STATUS:-



[1,2,3,4,5,N,6,N,7,8,9,N]
i j

Queue<node*> q
6 7 8 9

[1,2,3,4,5,N,6,N,7,8,9,N]
i j

Queue<node*> q
7 8 9

Steps) Next iteration:

l =

r =

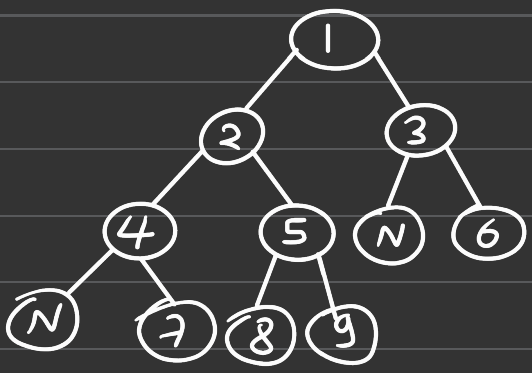
Btree STATUS:-

[1,2,3,4,5,N,6,N,7,8,9,N]
i j

queue<node*> q

7 8 9

All in All Working mechanism:—



l =

r =

temp = 6

The tree is Successfully Constructed...

Coding Implementation:-

```
node* binary_tree(int arr[],int n){
  node* root = new node(arr[0]);
  int i = 1;
  int j = 2;
  queue<node*>q;
  q.push(root);
  while(q.size()>0 && i<n){
    node* temp = q.front();
    q.pop();
    node* l;
    node* r;
    if(arr[i] != INT_MIN) l = new node(arr[i]);
    else  l = NULL;
    if(j<n && arr[j] != INT_MIN) r = new node(arr[j]);
    else  r = NULL;
    temp -> left = l;
    temp -> right = r;
    if(l)q.push(l);
    if(r)q.push(r);
    i += 2;
    j += 2;
  }
  return root;
}
```

The code works on exactly the same logic... Check out other document for code workflow...