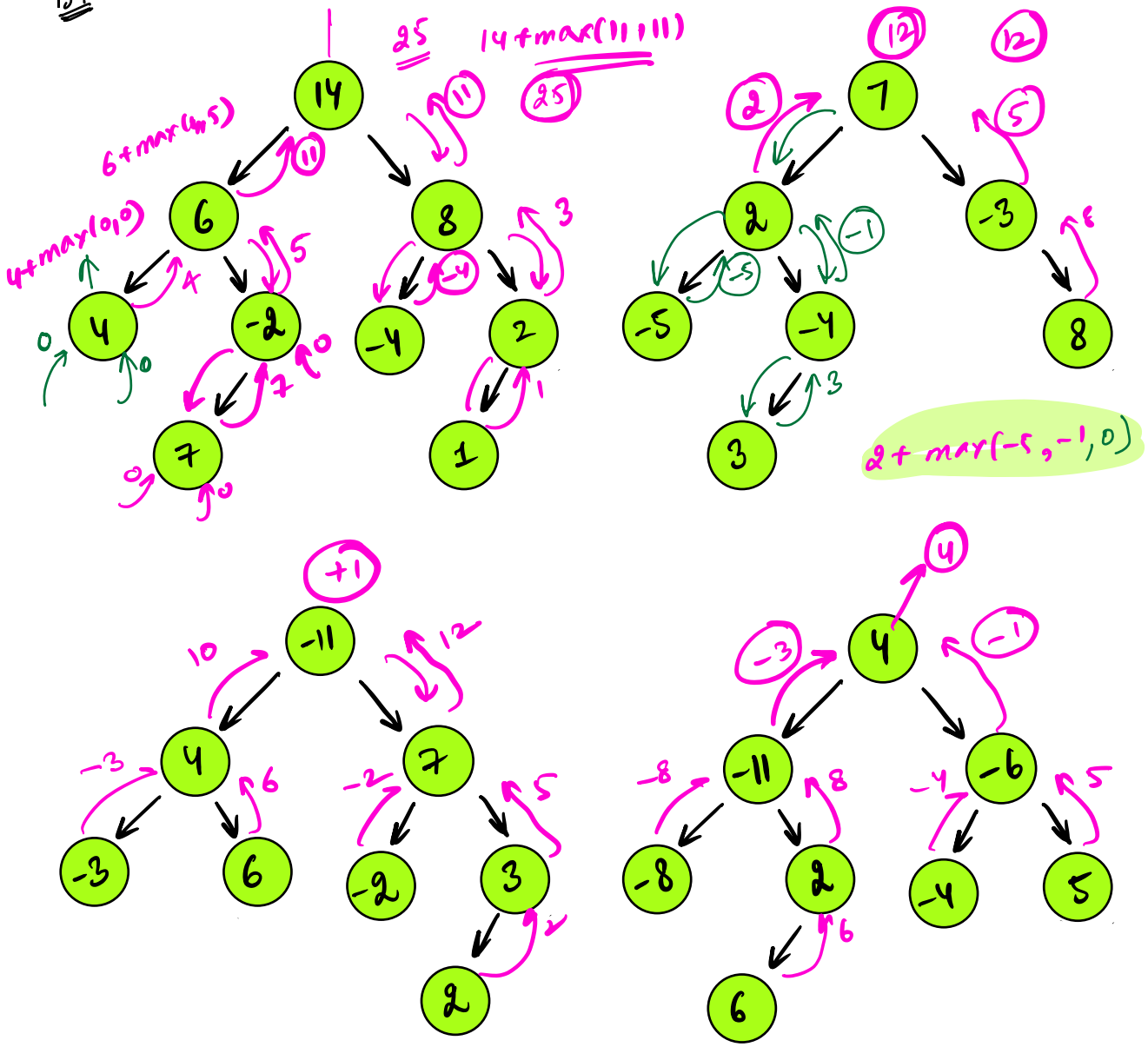
- Find maximum sum path starting from root (Return the max sum)

BT



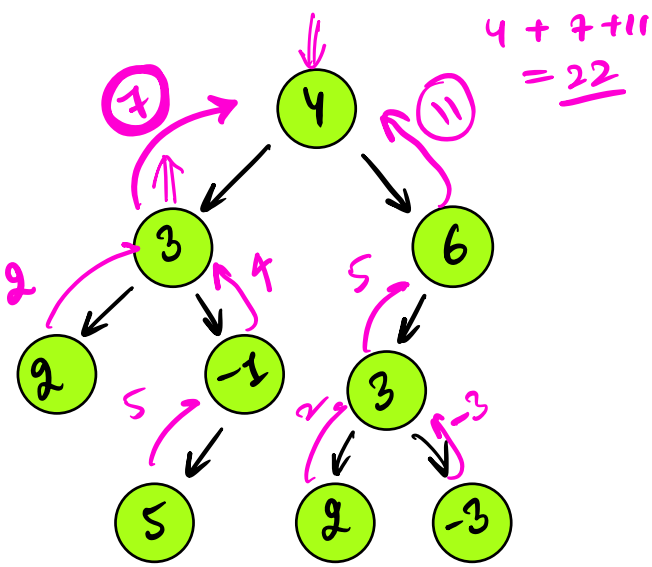

```
root.data + max(left, right, 0);
```

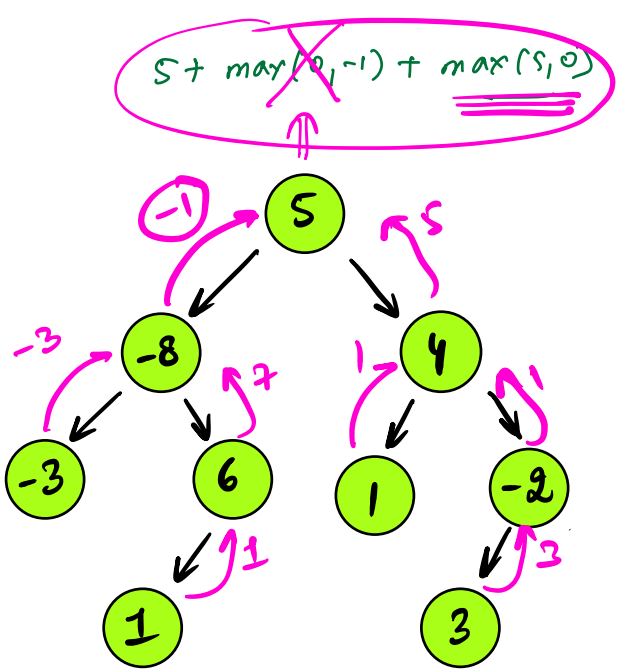
```
int maxpath( Node root)
{
    if( root ==null) return 0;

    int l = maxpath( root.left);
    int r = maxpath (root.right);

    return root.data + max( l, r, 0);
}
```

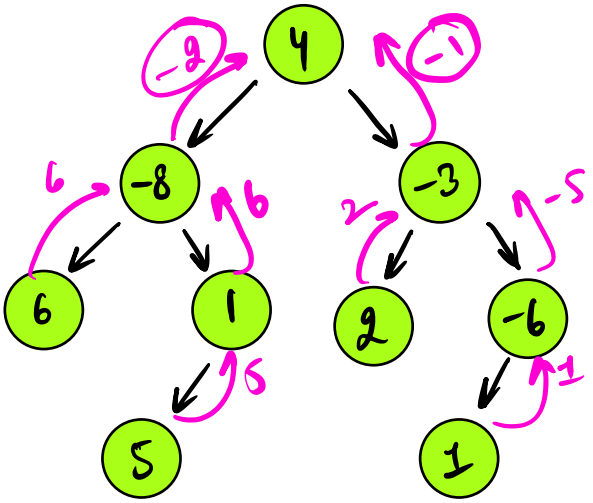
T.C: O(n)

S.C: O(H)

Q Max sum containing root node





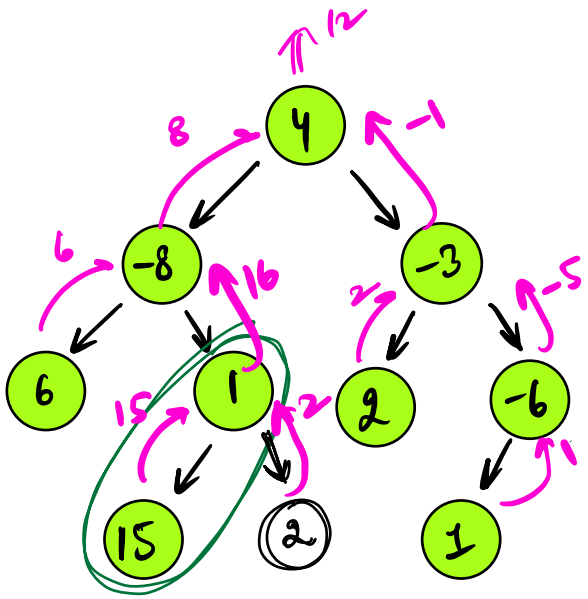


4 + max(0,-2) + max(0,-1)

root.data + max(l,0) + max(r,0)

```
main()
{
    int left = maxpath(root.left);
    int right = maxpath(root.right);

    return  root.data + max(left,0) + max(right,0)
}
```

- Maximum path sum

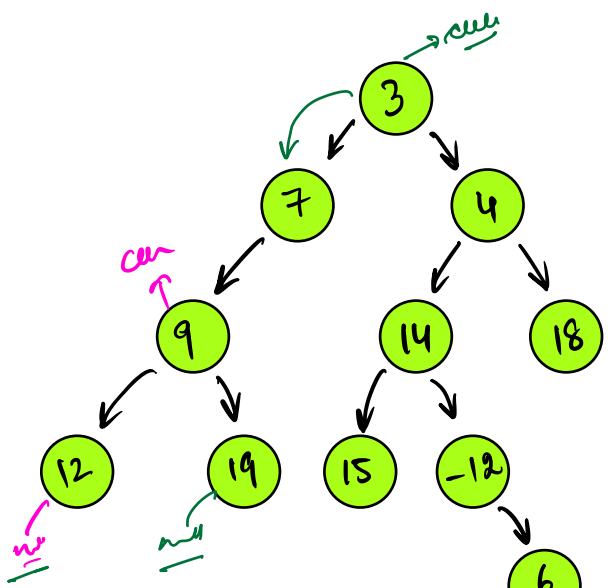


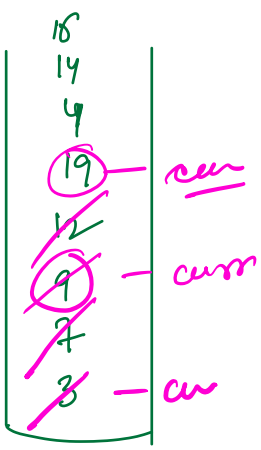
maxpath (node)
↓
max sum
starting from node

maxpathsum = INT_MIN;

```
int maxpath( Node root)
{
    if( root==null) return 0;

    int l = maxpath( root.left);
    int r = maxpath (root.right);
    maxpathsum = max( maxpathsum, root.data + max(l,0)
                                            + max(r,0))
    return root.data + max( l, r, 0);
}
```

- traversals (iterative)





inorder

```
curr = root;
stack<int> s;

while (!st.empty || curr != null)
{   while (curr != null)
    {  st.push(curr);
       curr = curr.left;
    }

    curr = st.top();
    print (curr.data);
       st.pop();
    curr = curr.right;

}
```

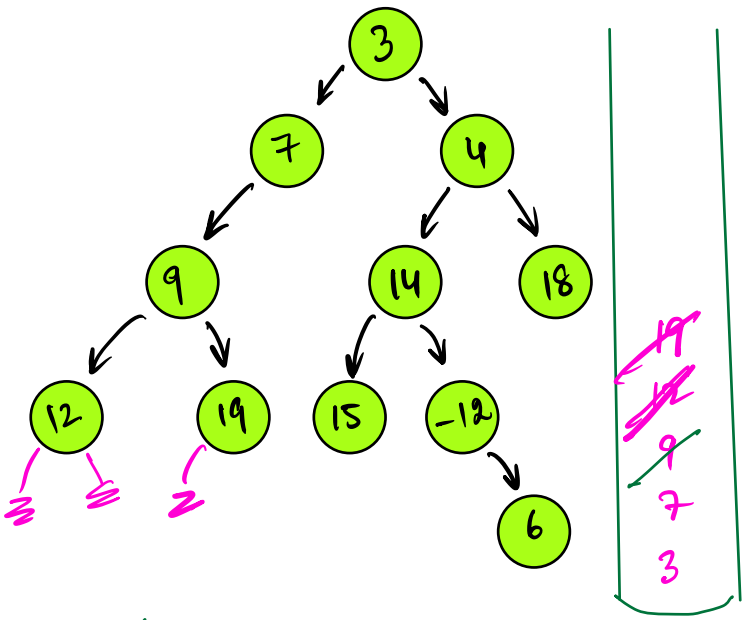
12 9 19 7 3

preorder ⇒

```
while( !st.empty || curr != null)
{
    while(curr != null)
    {
        print (curr.data);
        st.push( curr.right);
        curr = curr.left();
    }

    curr = st.top();
       st.pop().
```

if(curr.right != null) ←

}



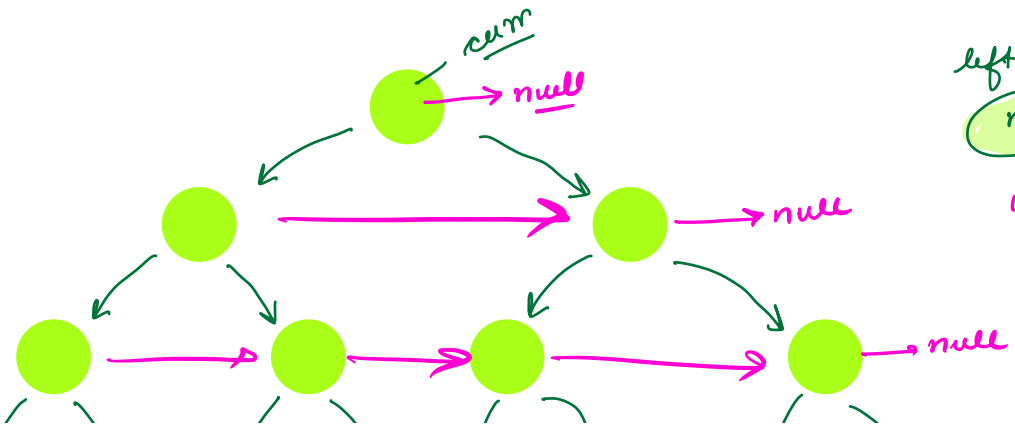
prev=~~19~~ 19

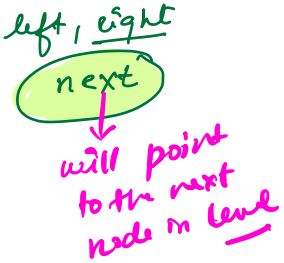
12 19 9

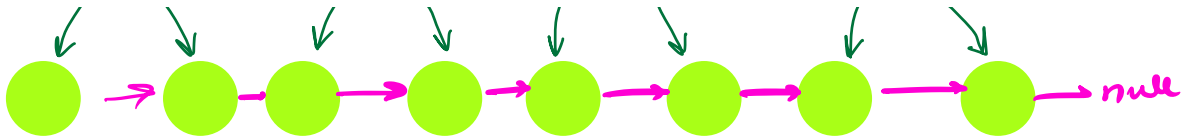
```
prev = null;
while ( !st.empty || curr != null )
{
    while ( curr != null)
    {
        st.push(curr);
        curr = curr.left;
    }
    curr = st.top();
    if(right==null || prev == curr.right)
    {
        print(curr.data);
        st.pop();
        prev = curr;
        curr = null;
    }
    else
    {
        curr = curr.right;
    }
}
```

* populate Right next pointer

perfect tree







S.C : O(1)

```
curr = root;
while (curr != null)
{
    Node temp = curr;

    while (temp.left != null)
    { temp.left.next = temp.right;
      temp.right.next = temp.next.left;
        temp = temp.next;
    }
    curr = curr.left;
}
```