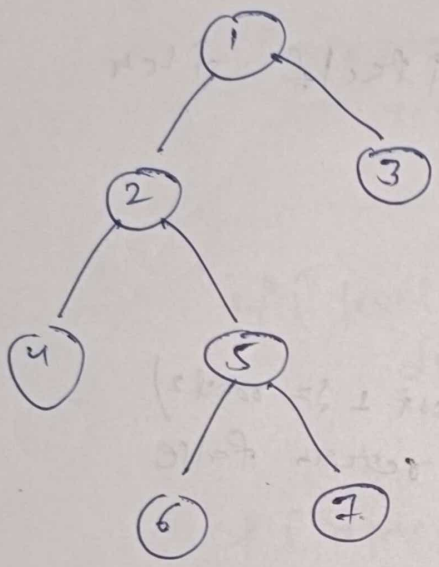
Root to Node Path

node = 7

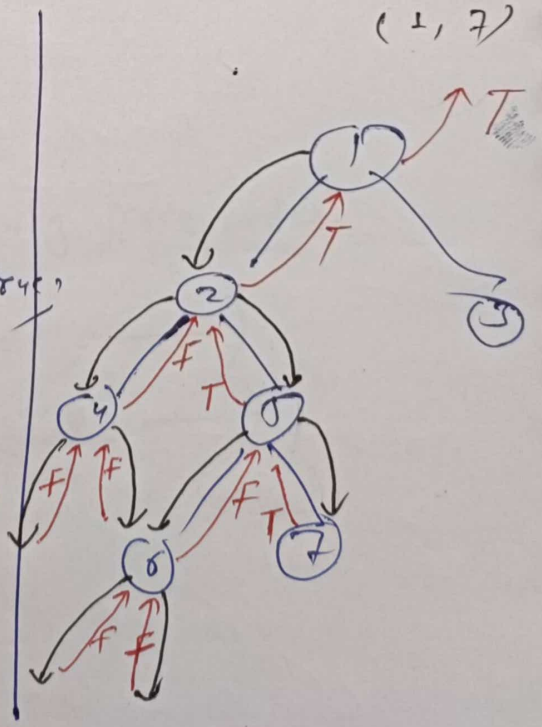
path = {1, 2, 5, 7}



Intuition :- compute root to node path
return ~~iso~~de & find node

To solve this problem we use problem like
isPresent that means is node / data present
in tree or not.

```
f(node, data)
{
  if(node == NULL) return false;
  if(node->val == data) return true;
  int l = f(node->left, data)
  if(l) return l;
  int r = f(node->right, data)
  return r;
}
```

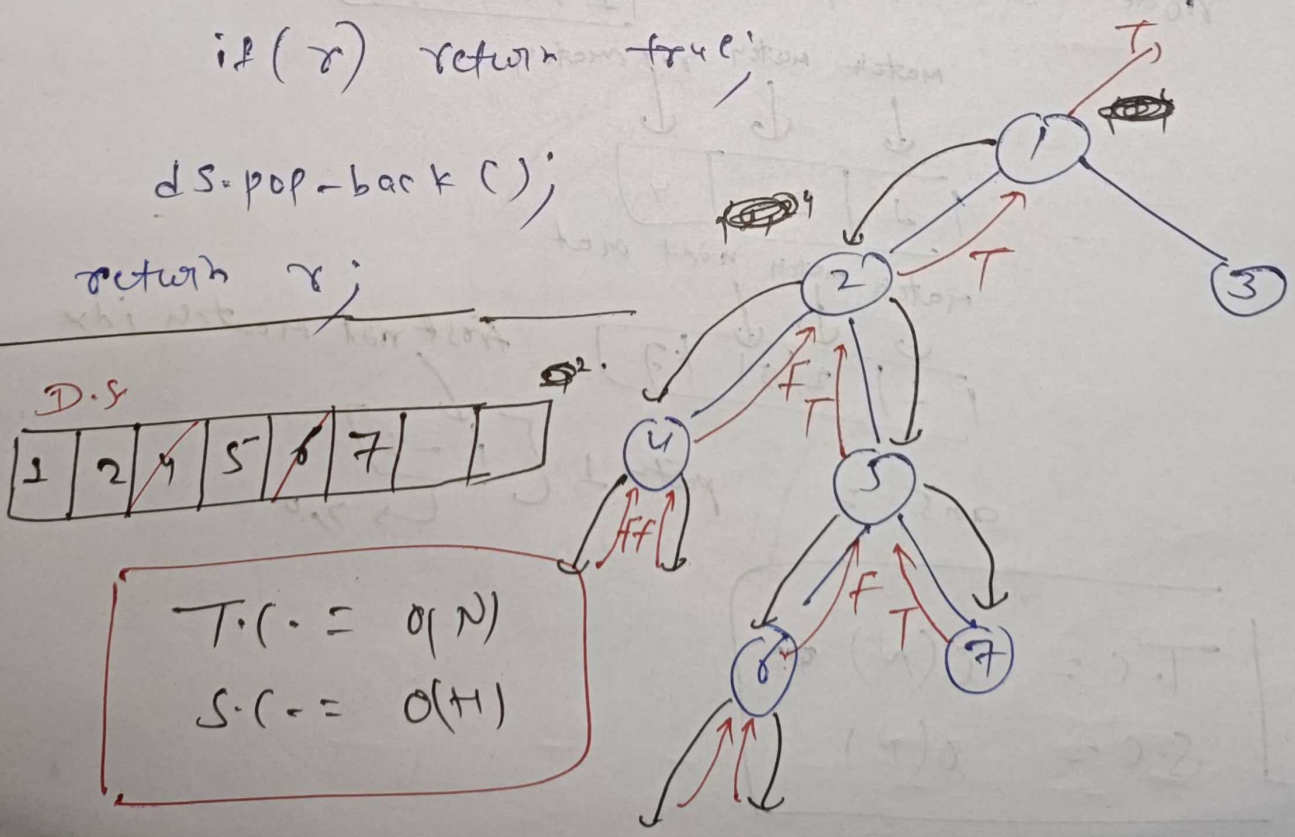


So in previos code we maintain a Data structure which will store Node to root path.

```
f(node, data, ds)
{
    if(node == NULL)
        return false;
    ds.push_back(node->val);
    if(node->val == data)
        return true;
    int l = f(node->left, data, ds)
        if(l) return true;
    int r = f(node->right, data, ds)
        if(r) return true;
    ds.pop_back();
    return r;
}
```

D.S.

1	2	5	7



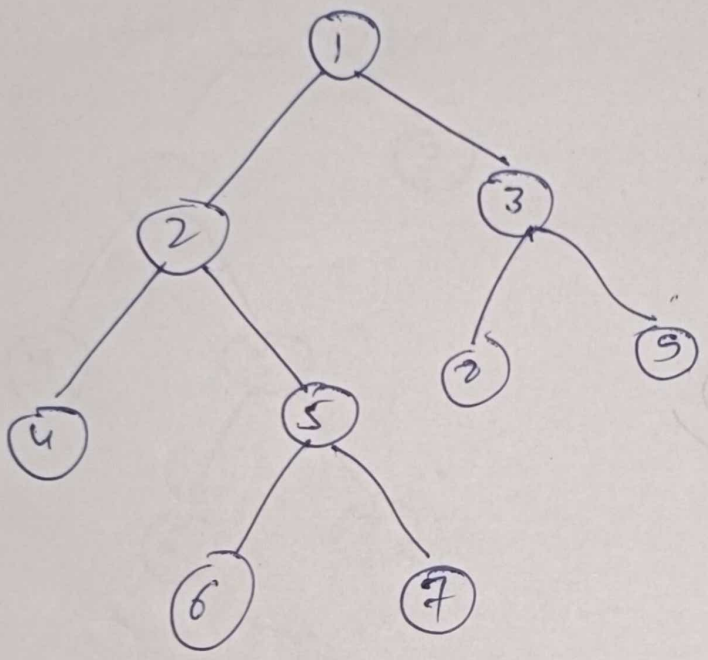
D.S

1	2	4	5	6	7		

$T.C. = O(N)$

$S.C. = O(H)$

Lowest Common Ancestor (Binary Tree)



LCA(4,7) = 2

Brute force approch :- we can find the root to node path for both nodes.

node - 4 → path = [1 | 2 | 4]

node - 7 → path = [1 | 2 | 5 | 7]

match match not match

[1 | 1 | 2 | 4]

match match not mat

[1 | 2 | 5 | 7]

first not match idx

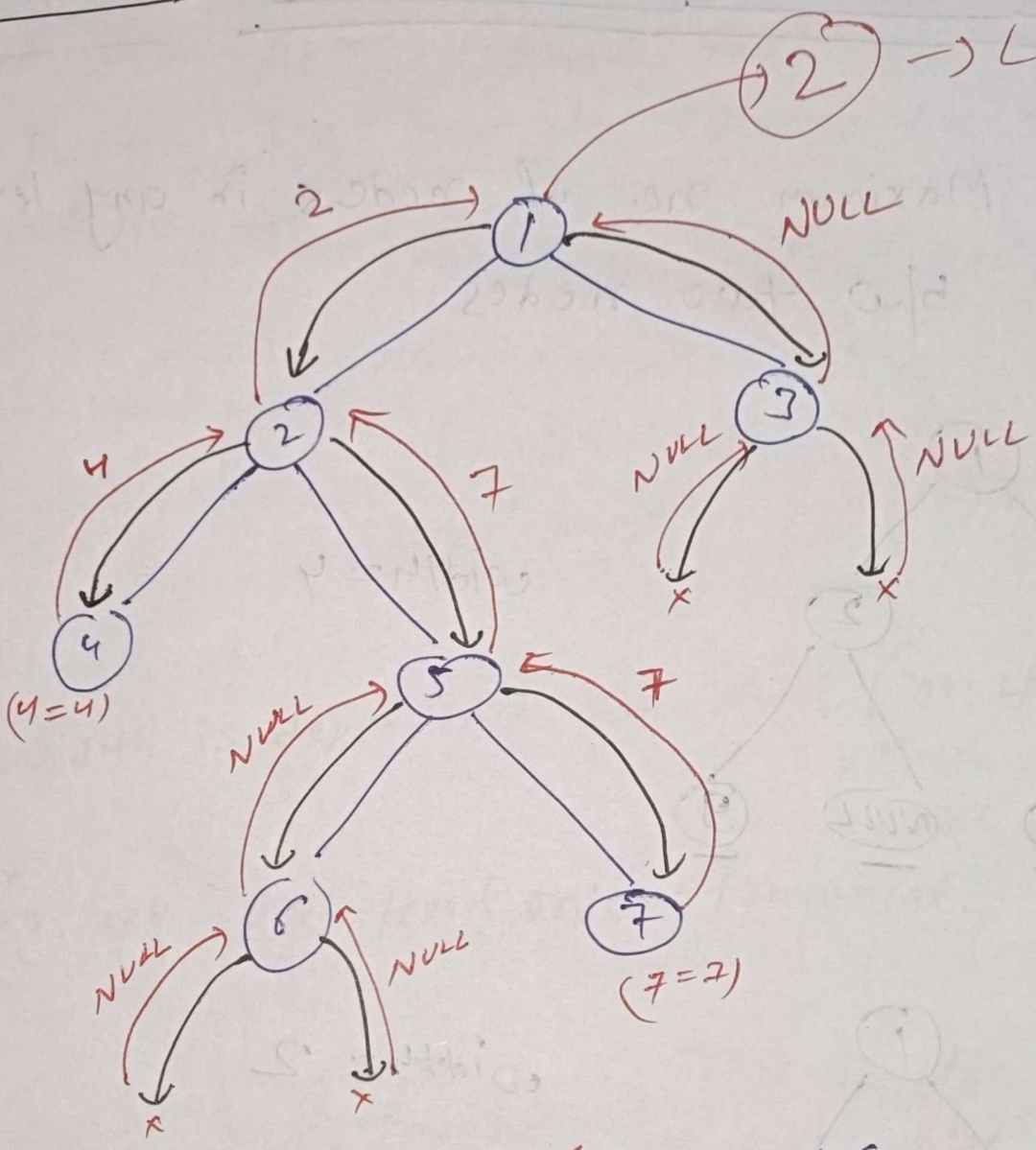
ans = path 1 [1 | -1]

↳ 1, 0

T.C = O(N)
S.C = O(N)

Approach - 2

LCA(4,7) = 2



ex-2

LCA(5,4) = 5