1221. Split a String in Balanced Strings

```
class Solution:
    def balancedStringSplit(self, s: str) -> int:
        count = 0
        res = 0
        for i in s:
            if i == 'L':
                count += 1
            else:
                count -= 1
            if count == 0:
                res += 1
        return res
```

1222. Queens That Can Attack the King

```
class Solution:
    def queensAttacktheKing(self, queens: List[List[int]],
king: List[int]) -> List[List[int]]:
        res = []
        memo = set(map(lambda a: tuple(a), queens))
        k_x, k_y = king
        for i in range(k_x + 1, 8):
            if (i, k_y) in memo:
                res.append([i, k_y])
                break
```

```
for i in range(k_x - 1, -1, -1):
    if (i, k_y) in memo:
        res.append([i, k_y])
        break

for j in range(k_y + 1, 8):
    if (k_x, j) in memo:
        res.append([k_x, j])
        break

for j in range(k_y, -1, -1):
    if (k_x, j) in memo:
        res.append([k_x, j])
        break
for d_i, d_j in [[1,1], [1,-1], [-1, 1], [-1, -1]]:
    for count in range(1, 8):
        new_x = d_i * count + k_x
        new_y = d_j * count + k_y
        if 0 <= new_x < 8 and 0 <= new_y < 8:
            if (new_x, new_y) in memo:
                res.append([new_x, new_y])
                break
        else:
            break
return res
```

968. Binary Tree Cameras
We will put camera on parent nodes. TC is O(n)

```
class Solution:
    def minCameraCover(self, root: TreeNode) -> int:
        self.res = 0
        def dfs(node):
          if not node:
            return 2
          left, right = dfs(node.left), dfs(node.right)
          if left == 0 or right == 0:
            self.res += 1
            return 1
          if left == 1 or right == 1:
            return 2
          else:
            return 0

        return self.res + 1 if dfs(root) == 0 else self.res
```

337. House Robber III

```
class Solution:
    def rob(self, root: TreeNode) -> int:
        def dfs(node):
          if not node:
            return (0, 0)
          left, right = dfs(node.left), dfs(node.right)
```

```
        return (left[1] + right[1] + node.val, max(left[0],
left[1]) + max(right[0],right[1]))
      return max(list(dfs(root)))
```

979. Distribute Coins in Binary Tree

```
class Solution:
    def distributeCoins(self, root: TreeNode) -> int:
        self.move = 0
        def traverse(node):
          if not node:
            return 0
          left, right = traverse(node.left), traverse(node.right)
          val = node.val + left + right - 1
          self.move += abs(val)
          return val
        traverse(root)
        return self.move
```