Description | Editorial | Solutions (9.4K) | Submissions

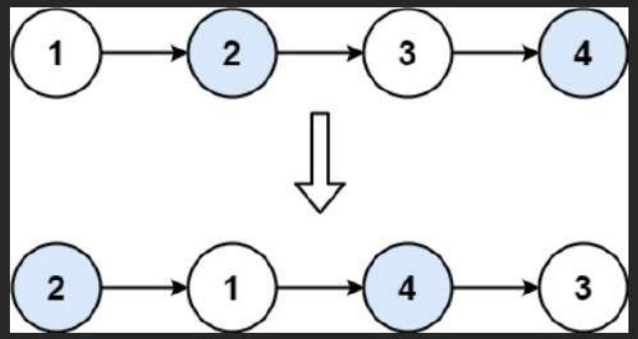
24. Swap Nodes in Pairs

Medium | 10.7K | 389

Companies

Given a linked list, swap every two adjacent nodes and return its head. You must solve the problem without modifying the values in the list's nodes (i.e., only nodes themselves may be changed.)

Example 1:




```
Input: head = [1,2,3,4]
Output: [2,1,4,3]
```

Example 2:

```
Input: head = []
Output: []
```

Example 3:

```
Input: head = [1]
```

Java | Auto

```
7   *     ListNode(int val) { this.val = val; }
8   *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
9   * }
10  */
11  class Solution {
12      public ListNode swapPairs(ListNode head) {
13          if(head == null || head.next == null)return head;
14          ListNode d = new ListNode(0), a;
15          d.next = head;
16          a=d;
17          while(head!=null&&head.next!=null) {
18              a.next = head.next;
19              head.next = head.next.next;
20              a.next.next = head;
21              a = a.next.next;
22              head =  head.next;
23          }
24          return d.next;
25      }
26  }
```

Testcase | Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

head =
[1,2,3,4]

Output

[2,1,4,3]

Expected

Console | Run | Submit

Description | Editorial | Solutions (5K) | Submissions

111. Minimum Depth of Binary Tree

Easy | 6.4K | 1.2K

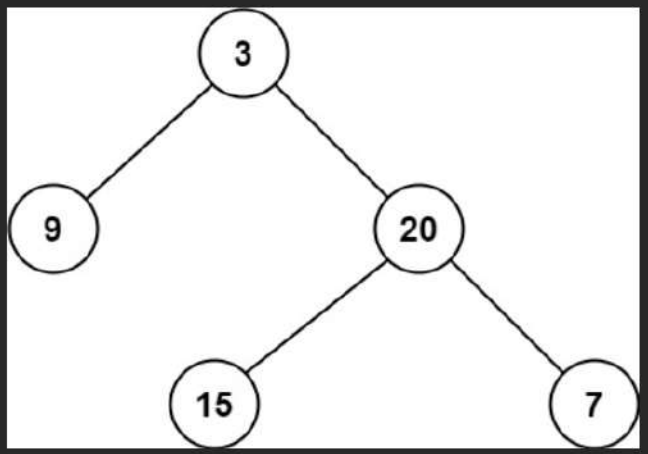
Companies

Given a binary tree, find its minimum depth.

The minimum depth is the number of nodes along the shortest path from the root node down to the nearest leaf node.

Note: A leaf is a node with no children.

Example 1:




```
Input: root = [3,9,20,null,null,15,7]
Output: 2
```

Example 2:

```
Input: root = [2,null,3,null,4,null,5,null,6]
```

Java | Auto

```java
*     }
* }
*/
class Solution {
    public int minDepth(TreeNode root)
    {
        if(root == null) return 0;

        int leftDepth = minDepth(root.left);
        int rightDepth = minDepth(root.right);

        if(root.left == null && root.right == null ) return 1;

        if(root.left == null) return 1 + rightDepth;

        if(root.right == null) return 1 + leftDepth;

        return Math.min(leftDepth,rightDepth) + 1;
    }
}
```

Testcase | Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

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

Output

2

Expected

Console | Run | Submit