

Our Solution(s)

Run Code

Solution 1

Solution 2

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 // O(n) time | O(n) space
4 function invertBinaryTree(tree) {
5   const queue = [tree];
6   while (queue.length) {
7     const current = queue.shift();
8     if (current === null) continue;
9     swapLeftAndRight(current);
10    queue.push(current.left);
11    queue.push(current.right);
12  }
13 }
14
15 function swapLeftAndRight(tree) {
16   const left = tree.left;
17   tree.left = tree.right;
18   tree.right = left;
19 }
20
21 exports.invertBinaryTree = invertBinaryTree;
22
```

Your Solutions

Run Code

Solution 1

Solution 2

Solution 3

```
1 function invertBinaryTree(tree) {
2   // Write your code here.
3 }
4
5 // Do not edit the line below.
6 exports.invertBinaryTree = invertBinaryTree;
7
```

Our Tests

Custom Output

Submit Code

```

1  class Solution:
2      def merge(self, nums1: List[int], m: int, nums2: List[int], n: int) -> None:
3          """
4          Do not return anything, modify nums1 in-place instead.
5          """
6          # Create a temporary array to hold the merged elements
7          temp = [0] * (m + n)
8          # Merge the two arrays into the temporary array
9          i = 0
10         j = 0
11         k = 0
12         while i < m and j < n:
13             if nums1[i] <= nums2[j]:
14                 temp[k] = nums1[i]
15                 i += 1
16             else:
17                 temp[k] = nums2[j]
18                 j += 1
19             k += 1
20         # Copy the remaining elements from nums1
21         while i < m:
22             temp[k] = nums1[i]
23             i += 1
24             k += 1
25         # Copy the remaining elements from nums2
26         while j < n:
27             temp[k] = nums2[j]
28             j += 1
29             k += 1
30         # Copy the merged array back to nums1
31         for i in range(m + n):
32             nums1[i] = temp[i]

```

Run or submit code when you're ready.