Check if Number Is Perfect Square

Given an integer n, return whether n = k * k for some integer k. This should be done without using built-in square root function.

Constraints
0 ≤ n < 2 ** 31

Example 1
Input
n = 25
Output
true
Explanation
25 = 5 * 5

Can prime factors help in some way ? Or maybe Binary Search ?

Fixed Point

Given a list of unique integers nums sorted in ascending order, return the minimum i such that nums[i] == i. If there's no solution, return -1.

This should be done in \mathcal{O}(log(n))O(log(n)) time.

Constraints
n ≤ 100,000 where n is the length of nums

Example 1
Input
nums = [-5, -2, 0, 3, 4]
Output
3
Explanation
Both nums[3] == 3 and nums[4] == 4 but 3 is smaller.

Example 2
Input
nums = [-5, -4, 0]
Output
-1
Explanation
There's no i such that nums[i] = i.

For sorted array, it hints for binary search

Level Order Traversal

Given a binary tree root return a level order traversal of the node values.

Constraints
n ≤ 100,000 where n is the number of nodes in root

Example 1
Input
Visualize
root = `[0, [5, null, null], [9, [1, [4, null, null], [2, null, null]], [3, null, null]]]`
Output
[0, 5, 9, 1, 3, 4, 2]

Example 2
Input
Visualize
root = `[0, [1, [2, [3, null, null], null], null], null]`
Output
[0, 1, 2, 3]

Example 3
Input
Visualize
root = `[0, null, [1, null, [2, null, [3, null, null]]]]`
Output
[0, 1, 2, 3]

Invert Tree

Given a binary tree root, invert it so that its left subtree and right subtree are
swapped and the children are recursively inverted.

Constraints
n ≤ 100,000 where n is the number of nodes in root

Example 1
Input
Visualize
root = `[0, [2, null, null], [9, [7, null, null], [12, null, null]]]`
Output
Visualize

```
[0, [9, [12, null, null], [7, null, null]], [2, null, null]]
```

Go through every node using DFS and swap their child