210. Course Schedule II

There are a total of `numCourses` courses you have to take, labeled from `0` to `numCourses - 1`. You are given an array `prerequisites` where `prerequisites[i] = [$a_i$, $b_i$]` indicates that you **must** take course b_i first if you want to take course a_i.

- For example, the pair `[0, 1]`, indicates that to take course `0` you have to first take course `1`.

Return *the ordering of courses you should take to finish all courses*. If there are many valid answers, return **any** of them. If it is impossible to finish all courses, return **an empty array**.

Example 1:

```
Input: numCourses = 2, prerequisites = [[1,0]]
Output: [0,1]
Explanation: There are a total of 2 courses to take. To take course 1 you should have finished course 0. So the correct course order is [0,1].
```

Example 2:

```
Input: numCourses = 4, prerequisites = [[1,0],[2,0],[3,1],[3,2]]
Output: [0,2,1,3]
Explanation: There are a total of 4 courses to take. To take course 3 you should have finished both courses 1 and 2. Both courses 1 and 2 should be taken after you finished course 0.
So one correct course order is [0,1,2,3]. Another correct ordering is [0,2,1,3].
```

Example 3:

```
Input: numCourses = 1, prerequisites = []
Output: [0]
```

Constraints:

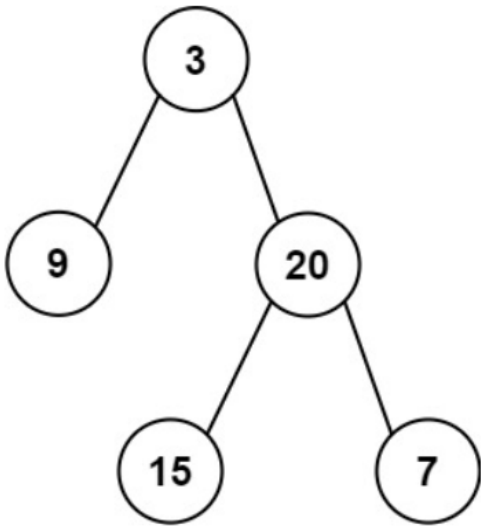
- `1 <= numCourses <= 2000`
- `0 <= prerequisites.length <= numCourses * (numCourses - 1)`
- `prerequisites[i].length == 2`
- `0 <= $a_i$, $b_i$ < numCourses`
- `$a_i$ != $b_i$`
- All the pairs `[$a_i$, $b_i$]` are **distinct**.

404. Sum of Left Leaves

Given the `root` of a binary tree, return *the sum of all left leaves.*

A **leaf** is a node with no children. A **left leaf** is a leaf that is the left child of another node.

Example 1:




```
Input: root = [3,9,20,null,null,15,7]
Output: 24
Explanation: There are two left leaves in the binary tree, with values 9 and 15
respectively.
```

Example 2:

```
Input: root = [1]
Output: 0
```

Constraints:

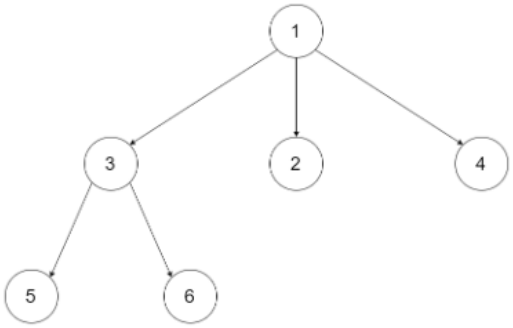
- The number of nodes in the tree is in the range `[1, 1000]`.
- `-1000 <= Node.val <= 1000`

429. N-ary Tree Level Order Traversal

Given an n-ary tree, return the *level order* traversal of its nodes' values.

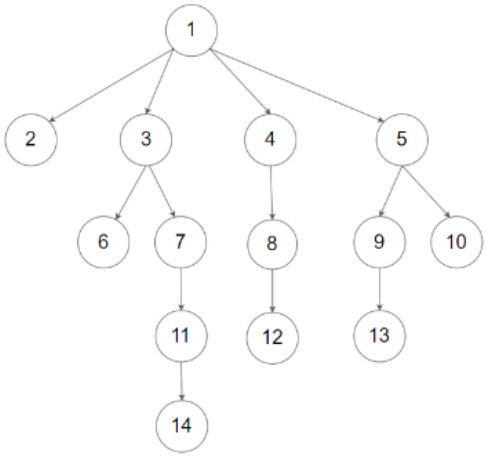
Nary-Tree input serialization is represented in their level order traversal, each group of children is separated by the null value (See examples).

Example 1:




```
Input: root = [1,null,3,2,4,null,5,6]
Output: [[1],[3,2,4],[5,6]]
```

Example 2:




```
Input: root =
[1,null,2,3,4,5,null,null,6,7,null,8,null,9,10,null,null,11,null,12,null,13,null
,null,14]
Output: [[1],[2,3,4,5],[6,7,8,9,10],[11,12,13],[14]]
```

Constraints:

- The height of the n-ary tree is less than or equal to `1000`
- The total number of nodes is between `[0, 10^4]`