348. Design Tic-Tac-Toe Premium

Medium Topics Companies Hint

Assume the following rules are for the tic-tac-toe game on an `n x n` board between two players:

1. A move is guaranteed to be valid and is placed on an empty block.
2. Once a winning condition is reached, no more moves are allowed.
3. A player who succeeds in placing `n` of their marks in a horizontal, vertical, or diagonal row wins the game.

Implement the `TicTacToe` class:

- `TicTacToe(int n)` Initializes the object the size of the board `n`.
- `int move(int row, int col, int player)` Indicates that the player with id `player` plays at the cell `(row, col)` of the board. The mo to be a valid move, and the two players alternate in making moves. Return
  - `0` if there is **no winner** after the move,
  - `1` if **player 1** is the winner after the move, or
  - `2` if **player 2** is the winner after the move.

Example 1:

```
Input
["TicTacToe", "move", "move", "move", "move", "move", "move", "move"]
[[3], [0, 0, 1], [0, 2, 2], [2, 2, 1], [1, 1, 2], [2, 0, 1], [1, 0, 2], [2, 1, 1]]
Output
[null, 0, 0, 0, 0, 0, 0, 1]

Explanation
TicTacToe ticTacToe = new TicTacToe(3);
Assume that player 1 is "X" and player 2 is "O" in the board.
ticTacToe.move(0, 0, 1); // return 0 (no one wins)
|X| | |
| | | |    // Player 1 makes a move at (0, 0).
| | | |

ticTacToe.move(0, 2, 2); // return 0 (no one wins)
|X| |O|
| | | |    // Player 2 makes a move at (0, 2).
| | | |

ticTacToe.move(2, 2, 1); // return 0 (no one wins)
|X| |O|
| | | |    // Player 1 makes a move at (2, 2).
| | |X|

ticTacToe.move(1, 1, 2); // return 0 (no one wins)
```

Java O(1) solution, easy to understand

bdwalker
1413 72841 May 04, 2016

Initially, I had not read the Hint in the question and came up with an O(n) solution. After reading the extremely helpful hint; a much easier approach became apparent. The key observation is that in order to win Tic-Tac-Toe you must have the entire row or column. Thus, we don't need to keep track of an entire n^2 board. We only need to keep a count for each row and column. If at any time a row or column matches the size of the board then that player has won.

To keep track of which player, I add one for Player1 and -1 for Player2. There are two additional variables to keep track of the count of the diagonals. Each time a player places a piece we just need to check the count of that row, column, diagonal and anti-diagonal.

Also see a very similar answer that I believe had beaten me to the punch. We came up with our solutions independently but they are very similar in principle.
Aeonaxx's soln

```
public class TicTacToe {
private int[] rows;
private int[] cols;
private int diagonal;
private int antiDiagonal;

/** Initialize your data structure here. */
public TicTacToe(int n) {
    rows = new int[n];
    cols = new int[n];
}

/** Player {player} makes a move at ({row}, {col}).
    @param row The row of the board.
    @param col The column of the board.
    @param player The player, can be either 1 or 2.
    @return The current winning condition, can be either:
            0: No one wins.
            1: Player 1 wins.
            2: Player 2 wins. */
public int move(int row, int col, int player) {
    int toAdd = player == 1 ? 1 : -1;

    rows[row] += toAdd;
```

```java
public class TicTacToe {
private int[] rows;
private int[] cols;
private int diagonal;
private int antiDiagonal;

/** Initialize your data structure here. */
public TicTacToe(int n) {
    rows = new int[n];
    cols = new int[n];
}

/** Player {player} makes a move at ({row}, {col}).
    @param row The row of the board.
    @param col The column of the board.
    @param player The player, can be either 1 or 2.
    @return The current winning condition, can be either:
            0: No one wins.
            1: Player 1 wins.
            2: Player 2 wins. */
public int move(int row, int col, int player) {
    int toAdd = player == 1 ? 1 : -1;

    rows[row] += toAdd;
    cols[col] += toAdd;
    if (row == col)
    {
        diagonal += toAdd;
    }

    if (col == (cols.length - row - 1))
    {
        antiDiagonal += toAdd;
    }

    int size = rows.length;
    if (Math.abs(rows[row]) == size ||
        Math.abs(cols[col]) == size ||
        Math.abs(diagonal) == size  ||
        Math.abs(antiDiagonal) == size)
    {
```

```java
public TicTacToe(int n) {
    rows = new int[n];
    cols = new int[n];
}

/** Player {player} makes a move at ({row}, {col}).
    @param row The row of the board.
    @param col The column of the board.
    @param player The player, can be either 1 or 2.
    @return The current winning condition, can be either:
            0: No one wins.
            1: Player 1 wins.
            2: Player 2 wins. */
public int move(int row, int col, int player) {
    int toAdd = player == 1 ? 1 : -1;

    rows[row] += toAdd;
    cols[col] += toAdd;
    if (row == col)
    {
        diagonal += toAdd;
    }

    if (col == (cols.length - row - 1))
    {
        antiDiagonal += toAdd;
    }

    int size = rows.length;
    if (Math.abs(rows[row]) == size ||
        Math.abs(cols[col]) == size ||
        Math.abs(diagonal) == size  ||
        Math.abs(antiDiagonal) == size)
    {
        return player;
    }

    return 0;
}

}
```

339. Nested List Weight Sum Premium

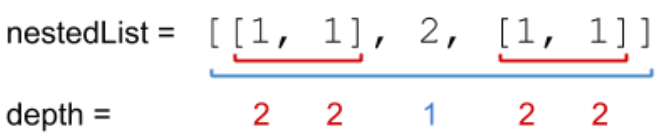
Medium Topics Companies

You are given a nested list of integers `nestedList`. Each element is either an integer or a list whose elements may also be integers or other

The **depth** of an integer is the number of lists that it is inside of. For example, the nested list `[1,[2,2],[[3],2],1]` has each integer's valu

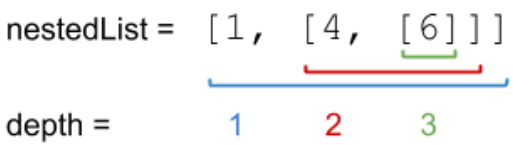
Return *the sum of each integer in* `nestedList` *multiplied by its* ***depth***.

Example 1:




```
Input: nestedList = [[1,1],2,[1,1]]
Output: 10
Explanation: Four 1's at depth 2, one 2 at depth 1. 1*2 + 1*2 + 2*1 + 1*2 + 1*2 = 10.
```

Example 2:




```
Input: nestedList = [1,[4,[6]]]
Output: 27
Explanation: One 1 at depth 1, one 4 at depth 2, and one 6 at depth 3. 1*1 + 4*2 + 6*3 = 27.
```

Example 3:

```
Input: nestedList = [0]
Output: 0
```

Constraints:

- `1 <= nestedList.length <= 50`
- The values of the integers in the nested list is in the range `[-100, 100]`.
- The maximum **depth** of any integer is less than or equal to `50`.

Code

Java
Auto

```java
/**
 * // This is the interface that allows for creating nested lists.
 * // You should not implement it, or speculate about its implementation
 * public interface NestedInteger {
 *     // Constructor initializes an empty nested list.
 *     public NestedInteger();
 *
 *     // Constructor initializes a single integer.
 *     public NestedInteger(int value);
 *
 *     // @return true if this NestedInteger holds a single integer, rather than a nested list.
 *     public boolean isInteger();
 *
 *     // @return the single integer that this NestedInteger holds, if it holds a single integer
 *     // Return null if this NestedInteger holds a nested list
 *     public Integer getInteger();
 *
 *     // Set this NestedInteger to hold a single integer.
 *     public void setInteger(int value);
 *
 *     // Set this NestedInteger to hold a nested list and adds a nested integer to it.
 *     public void add(NestedInteger ni);
 *
 *     // @return the nested list that this NestedInteger holds, if it holds a nested list
 *     // Return empty list if this NestedInteger holds a single integer
 *     public List<NestedInteger> getList();
 * }
 */
class Solution {
    public int depthSum(List<NestedInteger> nestedList) {
        if(nestedList == null){
            return 0;
        }

        int sum = 0;
        for (var v : nestedList ) {
          sum += getVal(v, 1);
        }

        return sum;
    }

    private int getVal(NestedInteger nestedInt, int level ) {
      if (nestedInt.isInteger()) {
        return nestedInt.getInteger() * level;
      }
      int ret = 0;
```

Saved to cloud

Testcase
Test Result

Case 1
Case 2
Case 3

nestedList =

[[1,1],2,[1,1]]

Code

Java
Auto

```java
        return sum;
    }

    private int getVal(NestedInteger nestedInt, int level ) {
      if (nestedInt.isInteger()) {
        return nestedInt.getInteger() * level;
      }
      int ret = 0;
      for (var v : nestedInt.getList()) {
        ret += getVal( v, level + 1);
      }
      return ret;


    }
}


//bfs
    // public int depthSum(List<NestedInteger> nestedList) {
    //     if(nestedList == null){
    //         return 0;
    //     }

    //     int sum = 0;
    //     int level = 1;

    //     Queue<NestedInteger> queue = new LinkedList<NestedInteger>(nestedList);
    //     while(queue.size() > 0){
    //         int size = queue.size();

    //         for(int i = 0; i < size; i++){
    //             NestedInteger ni = queue.poll();

    //             if(ni.isInteger()){
    //                 sum += ni.getInteger() * level;
    //             }else{
    //                 queue.addAll(ni.getList());
    //             }
    //         }

    //         level++;
    //     }

    //     return sum;
    // }
```

Saved to cloud

Testcase
Test Result

Case 1
Case 2
Case 3

nestedList =

[[1,1],2,[1,1]]

346. Moving Average from Data Stream Premium

Easy Topics Companies

Given a stream of integers and a window size, calculate the moving average of all integers in the sliding window.

Implement the `MovingAverage` class:

- `MovingAverage(int size)` Initializes the object with the size of the window `size`.
- `double next(int val)` Returns the moving average of the last `size` values of the stream.

Example 1:

```
Input
["MovingAverage", "next", "next", "next", "next"]
[[3], [1], [10], [3], [5]]
Output
[null, 1.0, 5.5, 4.66667, 6.0]

Explanation
MovingAverage movingAverage = new MovingAverage(3);
movingAverage.next(1); // return 1.0 = 1 / 1
movingAverage.next(10); // return 5.5 = (1 + 10) / 2
movingAverage.next(3); // return 4.66667 = (1 + 10 + 3) / 3
movingAverage.next(5); // return 6.0 = (10 + 3 + 5) / 3
```

Constraints:

- `1 <= size <= 1000`
- `-10^5 <= val <= 10^5`
- At most `10^4` calls will be made to `next`.

Seen this question in a real interview before? 1/4

Yes No

Accepted 360.3K Submissions 461.7K Acceptance Rate 78.0%

Topics

Companies

Similar Questions

Code

Java Auto

```java
class MovingAverage {


    private double previousSum = 0.0;
    private int maxSize;
    private Queue<Integer> currentWindow;

    public MovingAverage(int size) {
        currentWindow = new LinkedList<Integer>();
        maxSize = size;
    }

    public double next(int val) {
        if (currentWindow.size() == maxSize)
        {
            previousSum -= currentWindow.remove();
        }

        previousSum += val;
        currentWindow.add(val);
        return previousSum / currentWindow.size();
    }

}

/**
 * Your MovingAverage object will be instantiated and called as such:
 * MovingAverage obj = new MovingAverage(size);
 * double param_1 = obj.next(val);
 */
```

Saved to cloud

Testcase Test Result

Accepted Runtime: 2 ms

Case 1

Input

```
["MovingAverage","next","next","next","next"]
```

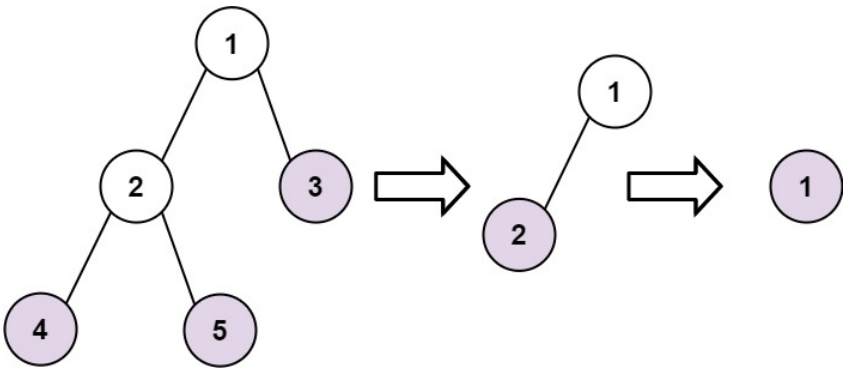
366. Find Leaves of Binary Tree Premium

Medium Topics Companies

Given the `root` of a binary tree, collect a tree's nodes as if you were doing this:

- Collect all the leaf nodes.
- Remove all the leaf nodes.
- Repeat until the tree is empty.

Example 1:




```
Input: root = [1,2,3,4,5]
Output: [[4,5,3],[2],[1]]
Explanation:
[[3,5,4],[2],[1]] and [[3,4,5],[2],[1]] are also considered correct answers since per each level it
matter the order on which elements are returned.
```

Example 2:

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

Constraints:

- The number of nodes in the tree is in the range `[1, 100]`.
- `-100 <= Node.val <= 100`

Seen this question in a real interview before? 1/4

Yes No

Accepted 252.1K Submissions 313.1K Acceptance Rate 80.5%

10 lines simple Java solution using recursion with explanation

sky-xu
1356 57394 Jun 24, 2016

Java

```java
public class Solution {
    public List<List<Integer>> findLeaves(TreeNode root) {
        List<List<Integer>> res = new ArrayList<>();
        height(root, res);
        return res;
    }
    private int height(TreeNode node, List<List<Integer>> res){
        if(null==node)  return -1;
        int level = 1 + Math.max(height(node.left, res), height(node.right, res));
        if(res.size()<level+1)  res.add(new ArrayList<>());
        res.get(level).add(node.val);
        return level;
    }
}
```

For this question we need to take bottom-up approach. The key is to find the height of each node.
Here the definition of height is:
The height of a node is the number of edges from the node to the deepest leaf. --CMU 15-121 Binary Trees

I used a helper function to return the height of current node. According to the definition, the height of leaf is 0. `h(node) = 1 + max(h(node.left), h(node.right))` .
The height of a node is also the its index in the result list (res). For example, leaves, whose heights are 0, are stored in res[0]. Once we find the height of a node, we can put it directly into the result.

UPDATE:
Thanks @adrianliu0729 for pointing out that my previous code does not actually remove leaves. I added one line `node.left = node.right = null;` to remove visited nodes

UPDATE:
There seems to be some debate over whether we need to actually "remove" leaves from the input tree. Anyway, it is just a matter of one line code. In the actual interview, just confirm with the interviewer whether removal is required.

Next
Java backtracking O(n) time O(n) space No hashing!

408. Valid Word Abbreviation Premium

Easy Topics Companies

A string can be **abbreviated** by replacing any number of **non-adjacent**, **non-empty** substrings with their lengths. The lengths **should not** l

For example, a string such as `"substitution"` could be abbreviated as (but not limited to):

- `"s10n"` (`"s ubstitutio n"`)
- `"sub4u4"` (`"sub stit u tion"`)
- `"12"` (`"substitution"`)
- `"su3i1u2on"` (`"su bst i t u ti on"`)
- `"substitution"` (no substrings replaced)

The following are **not valid** abbreviations:

- `"s55n"` (`"s ubsti tutio n"`, the replaced substrings are adjacent)
- `"s010n"` (has leading zeros)
- `"s0ubstitution"` (replaces an empty substring)

Given a string `word` and an abbreviation `abbr`, return *whether the string **matches** the given abbreviation*.

A **substring** is a contiguous **non-empty** sequence of characters within a string.

Example 1:

```
Input: word = "internationalization", abbr = "i12iz4n"
Output: true
Explanation: The word "internationalization" can be abbreviated as "i12iz4n" ("i nternational iz ati
```

Example 2:

```
Input: word = "apple", abbr = "a2e"
Output: false
Explanation: The word "apple" cannot be abbreviated as "a2e".
```

Constraints:

- `1 <= word.length <= 20`
- `word` consists of only lowercase English letters.
- `1 <= abbr.length <= 10`
- `abbr` consists of lowercase English letters and digits.
- All the integers in `abbr` will fit in a 32-bit integer

Code

Java
Auto

```java
class Solution {
    public boolean validWordAbbreviation(String word, String abbr) {
        int i = 0, j = 0;
        while (i < word.length() && j < abbr.length()) {
            if (word.charAt(i) == abbr.charAt(j)) {
                ++i;++j;
                continue;
            }
            if (abbr.charAt(j) <= '0' || abbr.charAt(j) > '9') {
                return false;
            }
            int start = j;
            while (j < abbr.length() && abbr.charAt(j) >= '0' && abbr.charAt(j) <= '9') {
                ++j;
            }
            int num = Integer.valueOf(abbr.substring(start, j));
            i += num;
        }
        return i == word.length() && j == abbr.length();
    }
}

```

Saved to cloud

Testcase
Test Result

Case 1
Case 2

word =

"internationalization"

abbr =

426. Convert Binary Search Tree to Sorted Doubly Linked List Premium

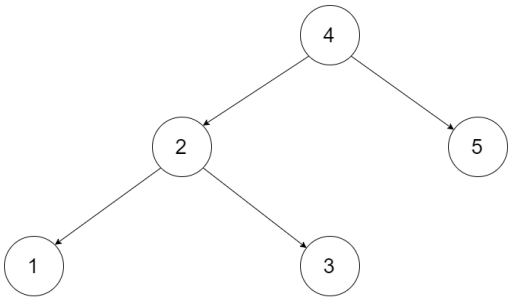
Medium Topics Companies

Convert a **Binary Search Tree** to a sorted **Circular Doubly-Linked List** in place.

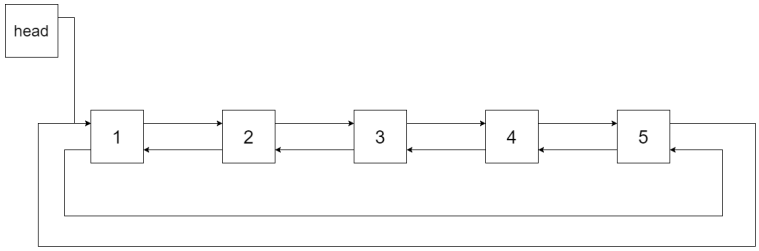
You can think of the left and right pointers as synonymous to the predecessor and successor pointers in a doubly-linked list. For a circular d predecessor of the first element is the last element, and the successor of the last element is the first element.

We want to do the transformation **in place**. After the transformation, the left pointer of the tree node should point to its predecessor, and t should point to its successor. You should return the pointer to the smallest element of the linked list.

Example 1:

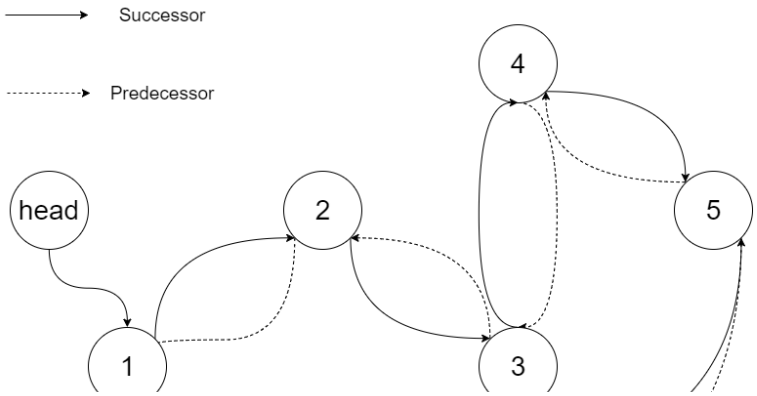


Input: root = [4,2,5,1,3]



Output: [1,2,3,4,5]

Explanation: The figure below shows the transformed BST. The solid line indicates the successor rela while the dashed line means the predecessor relationship.



Description
Solutions
Submissions
Editorial

Code

Java
Auto

```java
/*
// Definition for a Node.
class Node {
    public int val;
    public Node left;
    public Node right;

    public Node() {}

    public Node(int _val) {
        val = _val;
    }

    public Node(int _val,Node _left,Node _right) {
        val = _val;
        left = _left;
        right = _right;
    }
};
*/

class Solution {
    Node prev = null;
    public Node treeToDoublyList(Node root) {
      if (root == null) return null;
      Node dummy = new Node(0, null, null);
      prev = dummy;
      helper(root);
      //connect head and tail
      prev.right = dummy.right;
      dummy.right.left = prev;
      return dummy.right;
    }

    private void helper (Node cur) {
      if (cur == null) return;
      helper(cur.left);
      prev.right = cur;
      cur.left = prev;
      prev = cur;
      helper(cur.right);
    }

}
```

Saved to cloud

Testcase
Test Result

Case 1
Case 2

[4,2,5,1,3]

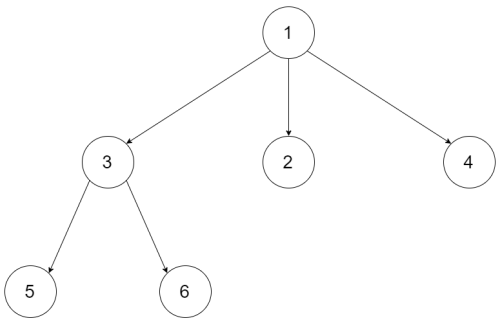
428. Serialize and Deserialize N-ary Tree Premium

Hard Topics Companies

Serialization is the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer across a network connection link to be reconstructed later in the same or another computer environment.

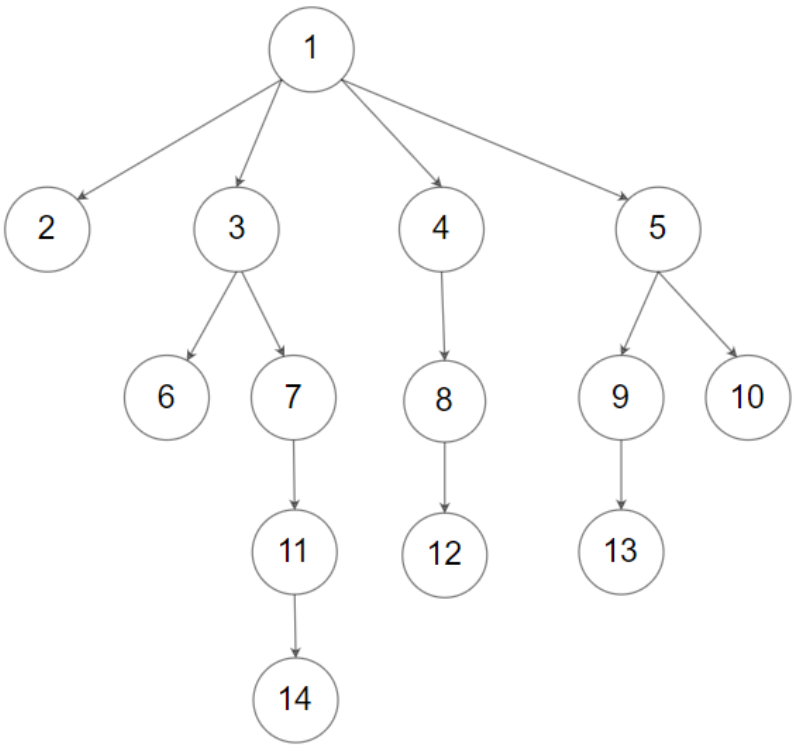
Design an algorithm to serialize and deserialize an N-ary tree. An N-ary tree is a rooted tree in which each node has no more than N childre restriction on how your serialization/deserialization algorithm should work. You just need to ensure that an N-ary tree can be serialized to a string can be deserialized to the original tree structure.

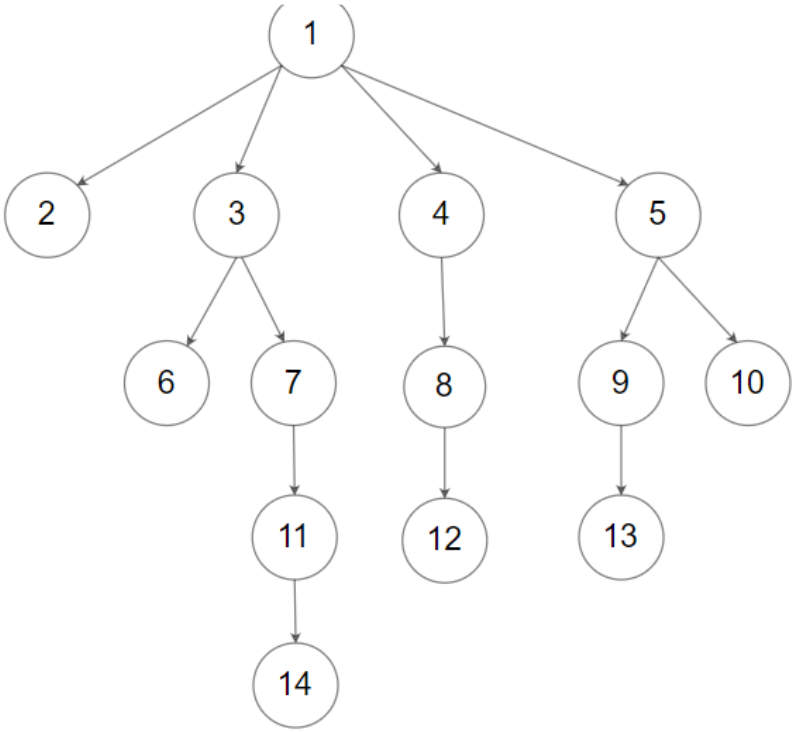
For example, you may serialize the following `3-ary` tree



as `[1 [3[5 6] 2 4]]`. Note that this is just an example, you do not necessarily need to follow this format.

Or you can follow LeetCode's level order traversal serialization format, where each group of children is separated by the null value.





For example, the above tree may be serialized as `[1,null,2,3,4,5,null,null,6,7,null,8,null,9,10,null,null,11,null,12,null,13,`

You do not necessarily need to follow the above-suggested formats, there are many more different formats that work so please be creative different approaches yourself.

Example 1:

```
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,
Output: [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]
```

Example 2:

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

Example 3:

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

Constraints:

- The number of nodes in the tree is in the range `[0, 10^4]`.
- `0 <= Node.val <= 10^4`
- The height of the n-ary tree is less than or equal to `1000`

Java preorder recursive solution using queue

xzh98
298 24239 Jul 19, 2018

Idea: preorder recursive traversal; add number of children after root val, in order to know when to terminate.
Example: The example in description is serialized as: "1,3,3,2,5,0,6,0,2,0,4,0"

```
class Codec {

    // Encodes a tree to a single string.
    public String serialize(Node root) {
        List<String> list=new LinkedList<>();
        serializeHelper(root,list);
        return String.join(",",list);
    }

    private void serializeHelper(Node root, List<String> list){
        if(root==null){
            return;
        }else{
            list.add(String.valueOf(root.val));
            list.add(String.valueOf(root.children.size()));
            for(Node child:root.children){
                serializeHelper(child,list);
            }
        }
    }

    // Decodes your encoded data to tree.
    public Node deserialize(String data) {
        if(data.isEmpty())
            return null;

        String[] ss=data.split(",");
        Queue<String> q=new LinkedList<>(Arrays.asList(ss));
        return deserializeHelper(q);
    }

    private Node deserializeHelper(Queue<String> q){
        Node root=new Node();
        root.val=Integer.parseInt(q.poll());
```

```
                serializeHelper(child,list);
            }
        }
    }

    // Decodes your encoded data to tree.
    public Node deserialize(String data) {
        if(data.isEmpty())
            return null;

        String[] ss=data.split(",");
        Queue<String> q=new LinkedList<>(Arrays.asList(ss));
        return deserializeHelper(q);
    }

    private Node deserializeHelper(Queue<String> q){
        Node root=new Node();
        root.val=Integer.parseInt(q.poll());
        int size=Integer.parseInt(q.poll());
        root.children=new ArrayList<Node>(size);
        for(int i=0;i<size;i++){
            root.children.add(deserializeHelper(q));
        }
        return root;
    }
}
```

Next
Serialize and Deserialize Binary and N-ary Tree Summary

Comments (25)

Sort by: Best

Type comment here... (Markdown supported)

Preview Comment

mmao3 Jan 05, 2019

489. Robot Room Cleaner Premium

Hard Topics Companies

You are controlling a robot that is located somewhere in a room. The room is modeled as an `m x n` binary grid where `0` represents a wall a empty slot.

The robot starts at an unknown location in the room that is guaranteed to be empty, and you do not have access to the grid, but you can m the given API `Robot`.

You are tasked to use the robot to clean the entire room (i.e., clean every empty cell in the room). The robot with the four given APIs can mo left, or turn right. Each turn is `90` degrees.

When the robot tries to move into a wall cell, its bumper sensor detects the obstacle, and it stays on the current cell.

Design an algorithm to clean the entire room using the following APIs:

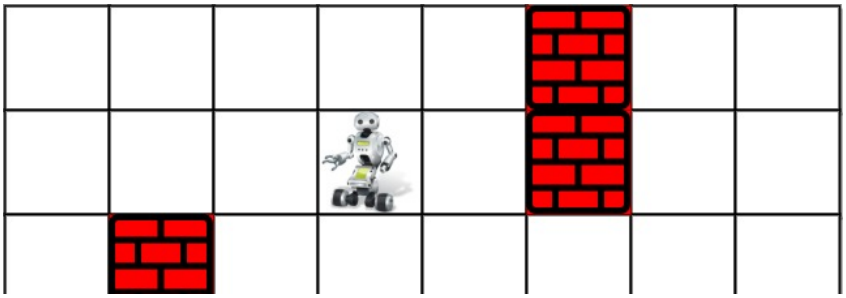
```
interface Robot {
  // returns true if next cell is open and robot moves into the cell.
  // returns false if next cell is obstacle and robot stays on the current cell.
  boolean move();

  // Robot will stay on the same cell after calling turnLeft/turnRight.
  // Each turn will be 90 degrees.
  void turnLeft();
  void turnRight();

  // Clean the current cell.
  void clean();
}
```

Note that the initial direction of the robot will be facing up. You can assume all four edges of the grid are all surrounded by a wall.

Custom testing:

The input is only given to initialize the room and the robot's position internally. You must solve this problem "blindfolded". In other words, y robot using only the four mentioned APIs without knowing the room layout and the initial robot's position.

Example 1:



Description Solutions Submissions Editorial

Code

Java Auto

```java
/**
 * // This is the robot's control interface.
 * // You should not implement it, or speculate about its implementation
 * interface Robot {
 *     // Returns true if the cell in front is open and robot moves into the cell.
 *     // Returns false if the cell in front is blocked and robot stays in the current cell.
 *     public boolean move();
 *
 *     // Robot will stay in the same cell after calling turnLeft/turnRight.
 *     // Each turn will be 90 degrees.
 *     public void turnLeft();
 *     public void turnRight();
 *
 *     // Clean the current cell.
 *     public void clean();
 * }
 */

class Solution {
    private static final int[][] directions = {{-1, 0}, {0, 1}, {1, 0}, {0, -1}};
    public void cleanRoom(Robot robot) {
      clean(robot, 0, 0, 0, new HashSet<>());
    }
   private void clean(Robot robot, int x, int y, int curDirection, Set<String> visited) {
        // Cleans current cell.
        robot.clean();
        visited.add(x + "-" + y);

        for (int i=0; i<4; ++i ){
            int nextDir = (curDirection+i) % 4;
            int nx = directions[nextDir][0] + x;
            int ny = directions[nextDir][1] + y;
            if (!visited.contains(nx + "-" + ny) && robot.move()) {
                clean(robot, nx, ny, nextDir, visited);
              // Moves backward one step while maintaining the orientation.
              robot.turnRight();
              robot.turnRight();
              robot.move();
              robot.turnRight();
              robot.turnRight();

            }
            // Changed orientation.
            robot.turnRight();
        }

    }
}
```

Saved to cloud

Testcase Test Result

Case 1 Case 2

```
[[1,1,1,1,1,0,1,1],[1,1,1,1,1,0,1,1],[1,0,1,1,1,1,1,1],[0,0,0,1,0,0,0,0],[1,1,1,1,1,1,1,1]]
```

```
1
```

490. The Maze Premium

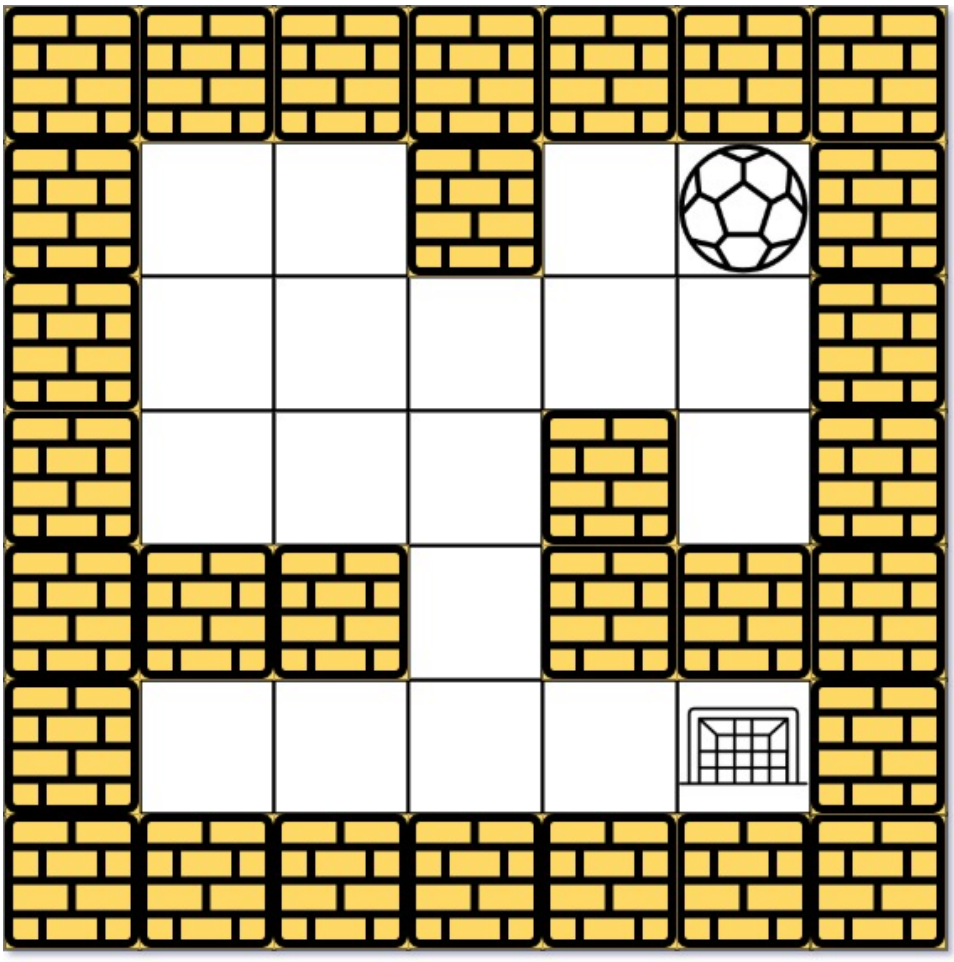
Medium Topics Companies

There is a ball in a `maze` with empty spaces (represented as `0`) and walls (represented as `1`). The ball can go through the empty spaces by **left or right**, but it won't stop rolling until hitting a wall. When the ball stops, it could choose the next direction.

Given the `m x n` `maze`, the ball's `start` position and the `destination`, where `start = [start_row, start_col]` and `destination = [destination_row, destination_col]`, return `true` if the ball can stop at the destination, otherwise return `false`.

You may assume that **the borders of the maze are all walls** (see examples).

Example 1:



```
Input: maze = [[0,0,1,0,0],[0,0,0,0,0],[0,0,0,1,0],[1,1,0,1,1],[0,0,0,0,0]], start = [0,4], destinat
Output: true
Explanation: One possible way is : left -> down -> left -> down -> right -> down -> right.
```

Example 2:

Code

Java
Auto

```java
class Solution {
    class Point {
        int x,y;
        public Point(int _x, int _y) {x=_x;y=_y;}
    }
    public boolean hasPath(int[][] maze, int[] start, int[] destination) {
        int m=maze.length, n=maze[0].length;
        if (start[0]==destination[0] && start[1]==destination[1]) return true;
        int[][] dir=new int[][] {{-1,0},{0,1},{1,0},{0,-1}};
        boolean[][] visited=new boolean[m][n];
        LinkedList<Point> list=new LinkedList<>();
        visited[start[0]][start[1]]=true;
        list.offer(new Point(start[0], start[1]));
        while (!list.isEmpty()) {
            Point p=list.poll();
            int x=p.x, y=p.y;
            for (int i=0;i<4;i++) {
                int xx=x, yy=y;
                while (xx>=0 && xx<m && yy>=0 && yy<n && maze[xx][yy]==0) {
                    xx+=dir[i][0];
                    yy+=dir[i][1];
                }
                xx-=dir[i][0];
                yy-=dir[i][1];
                if (visited[xx][yy]) continue;
                visited[xx][yy]=true;
                if (xx==destination[0] && yy==destination[1]) return true;
                list.offer(new Point(xx, yy));
            }
        }
        return false;

    }
}
```

Saved to cloud

Testcase
Test Result

Case 1
Case 2
Case 3

maze =

[[0,0,1,0,0],[0,0,0,0,0],[0,0,0,1,0],[1,1,0,1,1],[0,0,0,0,0]]

start =

536. Construct Binary Tree from String Premium

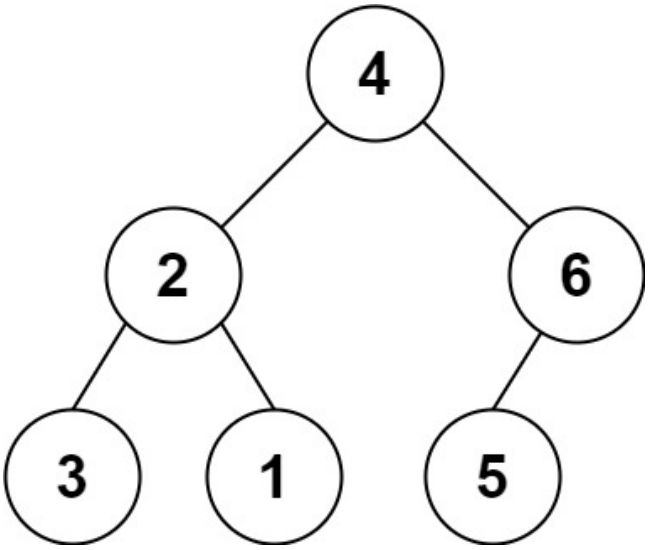
Medium Topics Companies

You need to construct a binary tree from a string consisting of parenthesis and integers.

The whole input represents a binary tree. It contains an integer followed by zero, one or two pairs of parenthesis. The integer represents the pair of parenthesis contains a child binary tree with the same structure.

You always start to construct the **left** child node of the parent first if it exists.

Example 1:




```
Input: s = "4(2(3)(1))(6(5))"
Output: [4,2,6,3,1,5]
```

Example 2:

```
Input: s = "4(2(3)(1))(6(5)(7))"
Output: [4,2,6,3,1,5,7]
```

Example 3:

```
Input: s = "-4(2(3)(1))(6(5)(7))"
Output: [-4,2,6,3,1,5,7]
```

Constraints:

- `0 <= s.length <= 3 * 10^4`
- `s` consists of digits, `'('`, `')'`, and `'-'` only.

Java stack solution

fallcreek
337 19564 Mar 12, 2017

```
ic class Solution {
ublic TreeNode str2tree(String s) {
    Stack<TreeNode> stack = new Stack<>();
    for(int i = 0, j = i; i < s.length(); i++, j = i){
        char c = s.charAt(i);
        if(c == ')')    stack.pop();
        else if(c >= '0' && c <= '9' || c == '-'){
            while(i + 1 < s.length() && s.charAt(i + 1) >= '0' && s.charAt(i + 1) <= '9') i++;
            TreeNode currentNode = new TreeNode(Integer.valueOf(s.substring(j, i + 1)));
            if(!stack.isEmpty()){
                TreeNode parent = stack.peek();
                if(parent.left != null)    parent.right = currentNode;
                else parent.left = currentNode;
            }
            stack.push(currentNode);
        }
    }
    return stack.isEmpty() ? null : stack.peek();
}
```

Next
Logical Thinking with Java Code Beats 95.72%

Comments (25)

Sort by: Best

Type comment here... (Markdown supported)

Preview Comment

wjixiaopeng Mar 12, 2017

Its reverse version of preorder traversal, so nice code and beautiful thought.

545. Boundary of Binary Tree Premium

Medium Topics Companies

The **boundary** of a binary tree is the concatenation of the **root**, the **left boundary**, the **leaves** ordered from left-to-right, and the **reverse o boundary**.

The **left boundary** is the set of nodes defined by the following:

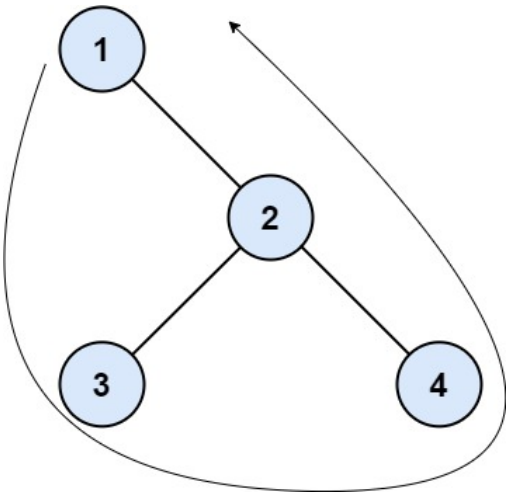
- The root node's left child is in the left boundary. If the root does not have a left child, then the left boundary is **empty**.
- If a node in the left boundary and has a left child, then the left child is in the left boundary.
- If a node is in the left boundary, has **no** left child, but has a right child, then the right child is in the left boundary.
- The leftmost leaf is **not** in the left boundary.

The **right boundary** is similar to the **left boundary**, except it is the right side of the root's right subtree. Again, the leaf is **not** part of the **ri** the **right boundary** is empty if the root does not have a right child.

The **leaves** are nodes that do not have any children. For this problem, the root is **not** a leaf.

Given the `root` of a binary tree, return *the values of its* ***boundary***.

Example 1:




```
Input: root = [1,null,2,3,4]
Output: [1,3,4,2]
Explanation:
- The left boundary is empty because the root does not have a left child.
- The right boundary follows the path starting from the root's right child 2 -> 4.
  4 is a leaf, so the right boundary is [2].
- The leaves from left to right are [3,4].
Concatenating everything results in [1] + [] + [3,4] + [2] = [1,3,4,2].
```

Example 2:

Java(12ms) - left boundary, left leaves, right leaves, right boundary

earlme
9511 37177 Mar 25, 2017

```
List<Integer> nodes = new ArrayList<>(1000);
public List<Integer> boundaryOfBinaryTree(TreeNode root) {

    if(root == null) return nodes;

    nodes.add(root.val);
    leftBoundary(root.left);
    leaves(root.left);
    leaves(root.right);
    rightBoundary(root.right);

    return nodes;
}
public void leftBoundary(TreeNode root) {
    if(root == null || (root.left == null && root.right == null)) return;
    nodes.add(root.val);
    if(root.left == null) leftBoundary(root.right);
    else leftBoundary(root.left);
}
public void rightBoundary(TreeNode root) {
    if(root == null || (root.right == null && root.left == null)) return;
    if(root.right == null)rightBoundary(root.left);
    else rightBoundary(root.right);
    nodes.add(root.val); // add after child visit(reverse)
}
public void leaves(TreeNode root) {
    if(root == null) return;
    if(root.left == null && root.right == null) {
        nodes.add(root.val);
        return;
    }
    leaves(root.left);
    leaves(root.right);
}
```

Next
python dfs solution

616. Add Bold Tag in String Premium

Medium Topics Companies

You are given a string `s` and an array of strings `words`.

You should add a closed pair of bold tag `<b>` and `</b>` to wrap the substrings in `s` that exist in `words`.

- If two such substrings overlap, you should wrap them together with only one pair of closed bold-tag.
- If two substrings wrapped by bold tags are consecutive, you should combine them.

Return `s` *after adding the bold tags*.

Example 1:

```
Input: s = "abcxyz123", words = ["abc","123"]
Output: "<b>abc</b>xyz<b>123</b>"
Explanation: The two strings of words are substrings of s as following: "abcxyz123".
We add <b> before each substring and </b> after each substring.
```

Example 2:

```
Input: s = "aaabbb", words = ["aa","b"]
Output: "<b>aaabbb</b>"
Explanation:
"aa" appears as a substring two times: "aaabbb" and "aaabbb".
"b" appears as a substring three times: "aaabbb", "aaabbb", and "aaabbb".
We add <b> before each substring and </b> after each substring: "<b>a<b>a</b>a</b><b>b</b><b>b</b><b
Since the first two <b>'s overlap, we merge them: "<b>aaa</b><b>b</b><b>b</b><b>b</b>".
Since now the four <b>'s are consecutive, we merge them: "<b>aaabbb</b>".
```

Constraints:

- `1 <= s.length <= 1000`
- `0 <= words.length <= 100`
- `1 <= words[i].length <= 1000`
- `s` and `words[i]` consist of English letters and digits.
- All the values of `words` are **unique**.

Note: This question is the same as 758: https://leetcode.com/problems/bold-words-in-string/

Java Solution, boolean array

shawngao
17101 32932 Jun 10, 2017

Use a boolean array to mark if character at each position is bold or not. After that, things will become simple.

```
public class Solution {
    public String addBoldTag(String s, String[] dict) {
        boolean[] bold = new boolean[s.length()];
        for (int i = 0, end = 0; i < s.length(); i++) {
            for (String word : dict) {
                if (s.startsWith(word, i)) {
                    end = Math.max(end, i + word.length());
                }
            }
            bold[i] = end > i;
        }

        StringBuilder result = new StringBuilder();
        for (int i = 0; i < s.length(); i++) {
            if (!bold[i]) {
                result.append(s.charAt(i));
                continue;
            }
            int j = i;
            while (j < s.length() && bold[j]) j++;
            result.append("<b>" + s.substring(i, j) + "</b>");
            i = j - 1;
        }

        return result.toString();
    }
}
```

Next
Java solution, Same as Merge Interval.

Comments (48)

Sort by: Best

Type comment here. (Markdown supported)

642. Design Search Autocomplete System Premium

Hard Topics Companies

Design a search autocomplete system for a search engine. Users may input a sentence (at least one word and end with a special character

You are given a string array `sentences` and an integer array `times` both of length `n` where `sentences[i]` is a previously typed sentence corresponding number of times the sentence was typed. For each input character except `'#'`, return the top `3` historical hot sentences tha prefix as the part of the sentence already typed.

Here are the specific rules:

- The hot degree for a sentence is defined as the number of times a user typed the exactly same sentence before.
- The returned top `3` hot sentences should be sorted by hot degree (The first is the hottest one). If several sentences have the same hot de code order (smaller one appears first).
- If less than `3` hot sentences exist, return as many as you can.
- When the input is a special character, it means the sentence ends, and in this case, you need to return an empty list.

Implement the `AutocompleteSystem` class:

- `AutocompleteSystem(String[] sentences, int[] times)` Initializes the object with the `sentences` and `times` arrays.
- `List<String> input(char c)` This indicates that the user typed the character `c`.
  - Returns an empty array `[]` if `c == '#'` and stores the inputted sentence in the system.
  - Returns the top `3` historical hot sentences that have the same prefix as the part of the sentence already typed. If there are fewer than return them all.

Example 1:

```
Input
["AutocompleteSystem", "input", "input", "input", "input"]
[[["i love you", "island", "iroman", "i love leetcode"], [5, 3, 2, 2]], ["i"], [" "], ["a"], ["#"]]
Output
[null, ["i love you", "island", "i love leetcode"], ["i love you", "i love leetcode"], [], []]

Explanation
AutocompleteSystem obj = new AutocompleteSystem(["i love you", "island", "iroman", "i love leetcode"
2]);
obj.input("i"); // return ["i love you", "island", "i love leetcode"]. There are four sentences that
"i". Among them, "ironman" and "i love leetcode" have same hot degree. Since ' ' has ASCII code 32 a
ASCII code 114, "i love leetcode" should be in front of "ironman". Also we only need to output top 3
sentences, so "ironman" will be ignored.
obj.input(" "); // return ["i love you", "i love leetcode"]. There are only two sentences that have
obj.input("a"); // return []. There are no sentences that have prefix "i a".
obj.input("#"); // return []. The user finished the input, the sentence "i a" should be saved as a h
sentence in system. And the following input will be counted as a new search.
```

Only thing more than a normal `Trie` is added a map of `sentence` to `count` in each of the `Trie` node to facilitate process of getting top 3 results.

```
public class AutocompleteSystem {
    class TrieNode {
        Map<Character, TrieNode> children;
        Map<String, Integer> counts;
        boolean isWord;
        public TrieNode() {
            children = new HashMap<Character, TrieNode>();
            counts = new HashMap<String, Integer>();
            isWord = false;
        }
    }

    class Pair {
        String s;
        int c;
        public Pair(String s, int c) {
            this.s = s; this.c = c;
        }
    }

    TrieNode root;
    String prefix;


    public AutocompleteSystem(String[] sentences, int[] times) {
        root = new TrieNode();
        prefix = "";

        for (int i = 0; i < sentences.length; i++) {
            add(sentences[i], times[i]);
        }
    }

    private void add(String s, int count) {
        TrieNode curr = root;
        for (char c : s.toCharArray()) {
            TrieNode next = curr.children.get(c);
            if (next == null) {
                next = new TrieNode();
```

```
private void add(String s, int count) {
    TrieNode curr = root;
    for (char c : s.toCharArray()) {
        TrieNode next = curr.children.get(c);
        if (next == null) {
            next = new TrieNode();
            curr.children.put(c, next);
        }
        curr = next;
        curr.counts.put(s, curr.counts.getOrDefault(s, 0) + count);
    }
    curr.isWord = true;
}

public List<String> input(char c) {
    if (c == '#') {
        add(prefix, 1);
        prefix = "";
        return new ArrayList<String>();
    }

    prefix = prefix + c;
    TrieNode curr = root;
    for (char cc : prefix.toCharArray()) {
        TrieNode next = curr.children.get(cc);
        if (next == null) {
            return new ArrayList<String>();
        }
        curr = next;
    }

    PriorityQueue<Pair> pq = new PriorityQueue<>((a, b) -> (a.c == b.c ? a.s.compareTo(b.s
    for (String s : curr.counts.keySet()) {
        pq.add(new Pair(s, curr.counts.get(s)));
    }

    List<String> res = new ArrayList<String>();
    for (int i = 0; i < 3 && !pq.isEmpty(); i++) {
        res.add(pq.poll().s);
    }
    return res;
}
```

708. Insert into a Sorted Circular Linked List Premium

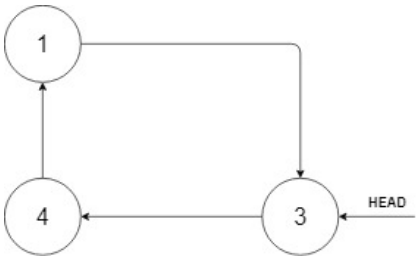
Medium Topics Companies

Given a Circular Linked List node, which is sorted in non-descending order, write a function to insert a value `insertVal` into the list such th
sorted circular list. The given node can be a reference to any single node in the list and may not necessarily be the smallest value in the circ

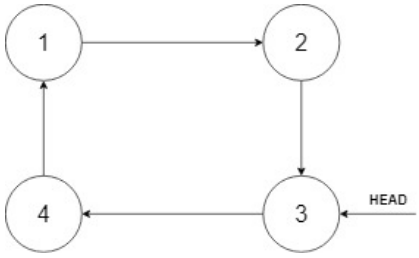
If there are multiple suitable places for insertion, you may choose any place to insert the new value. After the insertion, the circular list shou

If the list is empty (i.e., the given node is `null`), you should create a new single circular list and return the reference to that single node. Oth
return the originally given node.

Example 1:




```
Input: head = [3,4,1], insertVal = 2
Output: [3,4,1,2]
Explanation: In the figure above, there is a sorted circular list of three elements. You are given a
the node with value 3, and we need to insert 2 into the list. The new node should be inserted betwee
node 3. After the insertion, the list should look like this, and we should still return node 3.
```



Example 2:

```
Input: head = [], insertVal = 1
Output: [1]
Explanation: The list is empty (given head is null). We create a new single circular list and return
to that single node.
```

Example 3:

```
Input: head = [1], insertVal = 0
```

Code

Java
Auto

```java
class Node {
    public int val;
    public Node next;

    public Node() {}

    public Node(int _val) {
        val = _val;
    }

    public Node(int _val, Node _next) {
        val = _val;
        next = _next;
    }
};
*/

class Solution {

    public Node insert(Node start, int x) {
        // if start is null, create a node pointing to itself and return
        if (start == null) {
            Node node = new Node(x, null);
            node.next = node;
            return node;
        }
        // is start is NOT null, try to insert it into correct position
        Node cur = start;
        while (true) {
            // case 1A: has a tipping point, still climbing
            if (cur.val < cur.next.val) {
                if (cur.val <= x && x <= cur.next.val) { // x in between cur and next
                    insertAfter(cur, x);
                    break;
                }
            // case 1B: has a tipping point, about to return back to min node
            } else if (cur.val > cur.next.val) {
                if (cur.val <= x || x <= cur.next.val) { // cur is the tipping point, x is max or min val
                    insertAfter(cur, x);
                    break;
                }
            // case 2: NO tipping point, all flat
            } else {
                if (cur.next == start) {  // insert x before we traverse all nodes back to start
                    insertAfter(cur, x);
                    break;
                }
            }
            // None of the above three cases met, go to next node
            cur = cur.next;
        }
        return start;
    }

    // insert value x after Node cur
    private void insertAfter(Node cur, int x) {
        cur.next = new Node(x, cur.next);
    }
```

Saved to cloud

Testcase
Test Result