- ## 865. Smallest Subtree with all the Deepest Nodes

```
class Solution {
    int maxdepth = 0;
    Map<TreeNode, Integer> map;
    public TreeNode subtreeWithAllDeepest(TreeNode root) {
        if(root == null){
            return null;
        }
        map = new HashMap<>();
        dfs(root, 0);
        return findNode(root);

    }

    private void dfs(TreeNode root, int depth){
        if(root.left != null){
            dfs(root.left, depth + 1);
        }
        if(root.right != null){
            dfs(root.right, depth + 1);
        }
        map.put(root, depth);
        if(depth > maxdepth){
            maxdepth = depth;
        }
    }

    private TreeNode findNode(TreeNode node){
        if(node == null || map.get(node) == maxdepth){
            return node;
        }
        TreeNode left = findNode(node.left);
        TreeNode right = findNode(node.right);
        if(left != null && right != null){
            return node;
        }
        if(left != null){
            return left;
        }
        if(right != null){
            return right;
        }
        return null;
    }
}
```

- ## 547. Friend Circles

```
Hint:
  1.  设置visited array记录访问过的点
  2.  Dfs记录朋友圈
class Solution {
    public int findCircleNum(int[][] M) {
        boolean[] visited = new boolean [M.length];
        int num = 0;
        for(int row = 0; row < M.length; row++){
            if(visited[row] == false){
                visited[row] = true;
                num++;
                for(int col = 0; col < M[0].length; col++ ){
                    if(row == col){
                        continue;
                    }
                    if(M[row][col] == 1){
                        if(visited[col]){
                            dfs(M, col, visited);
                        }
                    }
                }
            }
        }
        return num;
    }

    private boolean inbound(int[][] M, int row, int col){
        return row >= 0 && col >= 0 && row < M.length && col < M[0].length;
    }

    private void dfs(int[][]M, int col, boolean[] visited){
        visited[col] = true;
        for(int i = 0; i < M[0].length; i++){
            if(i == col){
                continue;
            }
            if(M[col][i] == 1 && visited[i] == false){
                dfs(M, i, visited);
            }
        }
    }
}
```

- ## 695. Max Area of Island

```
Hint:
  1. Dfs
  2. 访问过的点要更改为0
class Solution {
    public int maxAreaOfIsland(int[][] grid) {
        int maxnum = 0;
        for(int row = 0; row < grid.length; row++){
            for(int col = 0; col < grid[0].length; col++){
                if(grid[row][col] == 1){
                    int area = 1;
                    grid[row][col] = 0;
                    area += dfs(grid, row + 1, col);
                    area += dfs(grid, row - 1, col);
                    area += dfs(grid, row, col + 1);
                    area += dfs(grid, row, col - 1);
                    if(area > maxnum){
                        maxnum = area;
                    }
                }
            }
        }
        return maxnum;
    }
    private int dfs(int[][] grid, int row, int col){
        int num = 0;
        if(row < 0 || col < 0 || row >= grid.length || col >= grid[0].length){
            return 0;
        }
        if(grid[row][col] == 1){
            num = 1;
            grid[row][col] = 0;
            num += dfs(grid, row + 1, col);
            num += dfs(grid, row - 1, col);
            num += dfs(grid, row, col + 1);
            num += dfs(grid, row, col - 1);
        }
        return num ;
    }
}
```

- ## 394. Decode String

```
Hint:
  1.用一个function实现dfs
  2.数字有可能大于9
class Solution {
    int pos = 0;
    public String decodeString(String s) {
        StringBuilder ans = new StringBuilder();
        String num = "";
        for(int i = pos; i < s.length(); i++){
            if (s.charAt(i) != '[' && s.charAt(i) != ']' && !Character.isDigit(s.charAt(i))) {
                ans.append(s.charAt(i));
            }
            else if(Character.isDigit(s.charAt(i))){
                num += s.charAt(i);
            }
            else if(s.charAt(i) == '['){
                pos = i + 1;
                String temp = decodeString(s);
                for(int j = 0; j < Integer.parseInt(num); j++){
                    ans.append(temp);
                }
                num = "";
                i = pos;
            }
            else if(s.charAt(i) == ']'){
                pos = i;
                return ans.toString();
            }
        }
        return ans.toString();
    }
}
```