手写算法 JavaScript 版

内容提要

- 活用基本数据结构解题
- 妙用链式结构的案例
- 搞定二叉树相关问题

活用基本数据结构解题

问题：你还记得哪些基本数据结构?

必须要掌握这几种基本结构

四种基本数据结构

- 集合

JavaScript：`Set`，`WeakSet`（ES6+）

https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Set

- 线性结构

JavaScript内置对象：`Array`

MDN Docs：https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Array

- 树结构

JavaScript内置对象：无

- 图结构

JavaScript内置对象：无

衍生结构：栈、队列、哈希表

- 栈

JavaScript内置对象：`Array`， `push()` + `pop()`

- 队列

JavaScript内置对象：`Array`， `push()` + `shift()`

- 哈希表

JavaS vcript内置对象：`Map`，`WeakMap`（ES6+）

MDN Docs：https://developer.mozilla.org/zh-CN/docs/orphaned/Web/JavaScript/Reference/Global_Objects/Map

案例：队列用于广度优先搜索

#200. 岛屿数量

为了求出岛屿的数量，我们可以扫描整个二维网格。如果一个位置为 11，则将其加入队列，开始进行广度优先搜索。在广度优先搜索的过程中，每个搜索到的 11 都会被重新标记为 00。直到队列为空，搜索结束。

最终岛屿的数量就是我们进行广度优先搜索的次数。

- 代码

```
/**
 * @param {character[][]} grid
 * @return {number}
 */
var numIslands = function(grid) {
  if(grid.length < 1) return 0 //容错
  //初始化变量
  let m = grid.length
  let n = grid[0].length
  let islands = 0

  for(let i = 0; i < m; i++){
    for(let j = 0; j < n; j++){
      if(grid[i][j] == 1){
        islands++
        grid[i][j] = 0             // 把查找过的项变成0 防止重新查找
        let queue = []
        queue.push([i, j])         // 把当前点加入队列
        while(queue.length > 0){   // 当队列不为空时, 继续循环
          let cur = queue.shift()  // 拿出队列第一项
          let x = cur[0], y = cur[1]
          // 上下左右检查
          if(x - 1 >= 0 && grid[x-1][y] == 1){  // 上
            queue.push([x - 1, y])
            grid[x - 1][y] = 0
          }
          if(x + 1 < m && grid[x + 1][y] == 1){  // 下
            queue.push([x + 1, y])
            grid[x + 1][y] = 0
          }
          if(y - 1 >= 0 && grid[x][y - 1] == 1){  // 左
            queue.push([x, y - 1])
            grid[x][y - 1] = 0
```

```
            }
            if(y + 1 < n && grid[x][y + 1] == 1){  // 右
              queue.push([x, y + 1])
              grid[x][y + 1] = 0
            }
          }
        }
      }
    }
    return islands
};
```

插播广告：

优惠活动：前20位享受本期精英班最低价，但是听销售同事说只有7个名额了，有意向报名的同学，可以联系我们的老师进行入学考试

课程大纲：https://www.yidengfe.com/outline/

妙用链式结构

链式结构与内存

与数组相似，链表也是一种**线性**数据结构

- 单向链表
- 双向链表
- 双指针技巧

js中这样实现链式结构就错了

```
var link = {
    data : 1,
    next : {
        data:3,
        next : {
            data : 4,
            next : undefinde
        }
    }
}
```

leetcode案例

#19. 删除链表的倒数第 N 个结点

双指针的经典应用，如果要删除倒数第n个节点，让fast移动n步，然后让fast和slow同时移动，直到fast指向链表末尾。删掉slow所指向的节点就可以了。

```
/**
 * Definition for singly-linked list.
 * function ListNode(val, next) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.next = (next===undefined ? null : next)
 * }
 */
/**
 * @param {ListNode} head
 * @param {number} n
 * @return {ListNode}
 */
var removeNthFromEnd = function(head, n) {
    let ret = new ListNode(0, head),
        slow = fast = ret;
    while(n--) fast = fast.next;
    if(!fast) return ret.next;
    while (fast.next) {
        fast = fast.next;
        slow = slow.next
    };
    slow.next = slow.next.next;
    return ret.next;

};
```

搞定二叉树相关问题

树与二叉树

树是一种经常用到的数据结构，用来模拟具有树状结构性质的数据集合。

树里的每一个节点有一个值和一个包含所有子节点的列表。

二叉树是一种更为典型的树状结构，是每个节点最多有两个子树的树结构，通常子树被称作“左子树”和“右子树”。

搞懂二叉树遍历只需理解两点

- 递归
- 自相似

leetcode案例

#104. 二叉树的最大深度

深度优先

找出终止条件：当前节点为空
找出返回值：节点为空时说明高度为 0，所以返回 0；节点不为空时则分别求左右子树的高度的最大值，同时加1表示当前节点的高度，返回该数值
某层的执行过程：在返回值部分基本已经描述清楚
时间复杂度：O(n)O(n)

```
/**
 * Definition for a binary tree node.
 * function TreeNode(val) {
 *     this.val = val;
 *     this.left = this.right = null;
 * }
 */
/**
 * @param {TreeNode} root
 * @return {number}
 */
var maxDepth = function(root) {
    if(!root) {
        return 0;
    } else {
        const left = maxDepth(root.left);
        const right = maxDepth(root.right);
        return Math.max(left, right) + 1;
    }
};

```