【第十四课】哈希表与布隆过滤器

318. 最大单词长度乘积

思路：1.要求是不含相同字母的两个单词，重点是如何判断不含相同字母。

2.将单词全部转换成二进制数字，然后进行与运算，结果为零时没有相同字母，进行运算；

```
  /**
 * @param {string[]} words
 * @return {number}
 */

var maxProduct = function(words) {

    var getCharCodeDiff = function(char) {
        return char.charCodeAt(0) - 'a'.charCodeAt(0);
    };

    var n = words.length,
        lens = [],
        masks = [];
    for (var i = 0; i < n; i++) {

        var len = words[i].length;
        lens.push(len);

        var mask = 0;
        for (var j = 0; j < len; j++) {
            mask |= 1 << getCharCodeDiff(words[i][j]);
        }
        masks.push(mask);
    }

    var max = 0;
    for (var i = 0; i < n; i++) {
        for (var j = i+1; j < n; j++) {
            if ((masks[i] & masks[j]) == 0) {
                max = Math.max(max, lens[i] * lens[j]);
            }
        }
    }

    return max;
};
```

240. 搜索二维矩阵 II

思路： 1.这个矩阵特性就是横行有序，纵行有序。并且在这个n*m的矩阵中，最多走n+m步就可以找到x的值。

2.最特殊的两个值，左上角15，右下角18。如果X大于15，因为第一行的最大值就是15，证明，x一定不在第一行，所以我们当前位置往下移动找；如果小于15，15是最后一列的最小值，证明x不在最后一列上，x就往前走。

3.最后发现我们通过比较，当前元素的值和x的值大小关系，就能确定x往前走还是往下走。同样右下角的18也是发挥一样的作用。

```
/**
 * @param {number[][]} matrix
 * @param {number} target
 * @return {boolean}
 */
var searchMatrix = function(matrix, target) {
    let i = 0, j = matrix[0].length -1;
    while(i < matrix.length && j >= 0){
        // i, j 等于待查找值
        if(matrix[i][j] == target) return true;
        if(matrix[i][j] < target) i += 1;
        else j -= 1;
    }
    return false;
};
```

979. 在二叉树中分配硬币

思路：1.二叉树有n个节点，也有n个硬币，问我们如果让一节点只有一个硬币，移动最少的次数。

2.依次统计每一个子树的节点数量和硬币数量，节点数量 减去金币数量就是，会经过它脑瓜顶上的这条边的硬币数量。

3。所以我们最后统计的差，就是我们最后求的数量。

4.这道题本身是在考结构化思想，整体思考 硬币的最少移动次数，等价地把原问题拆解成，每条边上到底经过了多少硬币。

```
//  思想：移动硬币的次数 转化成 每条边经过的硬币数量 统计出来 就是我们最后要求
var distributeCoins = function(root) {
    function dfs(root){
          if(root == null) return [0,0];//移动的步数，一个是需要的金币的数量
          let left = dfs(root.left),
            right = dfs(root.right),
            read = (right[1] + left[1] + root.val - 1);
            return [Math.abs(read) + left[0] + right[0],read ]
    }
    return dfs(root)[0];

};
```

430. 扁平化多级双向链表

思路： 1.拿第一个例子举例，就是把多级地双向链表变成统一地单向链表。这个过程就是一个递归地过程。

2.如果多级链表中有一个特殊地指针域指向了孩子节点，就是例子中的3指向了7。我们将倒数第一行和倒数第二行，框起来，变成一个单向链表。

3.就是把11 12 插入到 8和9中间，这样就变成了，7-8-11-12-9-10。

4.接着再把这条链表插入到3和4的中间，最后变成：1-2-3-7-8-11-12-9-10-4-5-6地一条链表结构。

```
var flatten = function(head) {
    if(head == null) return null;
    // 指针p
    let p = head,q,k;
    // 当p不指向空地时候，每次都指向当前地这个节点
    while(p){
        k = null;
        if(p.child){
            //如果当前节点要是有孩子节点，就把孩子节点进行扁平化处理，存到k 里面
            //k是扁平化之后的链表， 将p跟k进行连接，再移动到k链表的最后一位，k和q进行连接，q是p的下一位那么就跟q连接
            k = flatten(p.child);
            p.child = null;
            q = p.next;
            p.next = k;
            k.prev = p;
            // 再让p顺着k链表走到最后一位
            while(p.next) p = p.next;
            p.next = q;
            if(q) q.prev = p;//要判断q是否为空，如果q为空，那么这样访问 q.prev是非法的
        }
        p = p.next;
    }
    return head;
};
```

863. 二叉树中所有距离为 K 的结点

思路： 1.拿第一个例子说，距离元素5，路程为2的节点，元素5往上走是节点1；往下走是节点7和节点4；主要是考递归。

2.在递归过程中，先找到目标节点，把目标节点当作根节点,接着往上找距离为k的节点;找完再往下找距离为k的节点

```
var distanceK = function(root, target, k) {
    if(!root) return [];
    let targetNode = null;
    let res = [];
    let paths = [];

    // 找到target节点，存储到targetNode中
    dfs(root, target);
    // 从当前节点向下寻找
    getdownDis(targetNode, k);
    // 从当前节点向上寻找
    while(targetNode.parent && k>0){
        targetNode = targetNode.parent;
        getdownDis(targetNode, --k);
    }
```

```
    // 辅助函数
    function dfs(root, target){
        if(!root || targetNode) return;
        if(root.val === target.val){
            targetNode = root;
        }
        if(root.left){
            root.left.parent = root;
            dfs(root.left, target);
        }
        if(root.right){
            root.right.parent = root;
            dfs(root.right, target);
        }
    }

    // 辅助函数
    function getdownDis(node, k){
        if(node === null || paths.indexOf(node) !== -1) return;
        paths.push(node);
        if(k>0){
            getdownDis(node.left, k-1);
            getdownDis(node.right, k-1);
        }else if(k === 0){
            res.push(node.val);
        }
    }

    return res;

};
```

kaikeba 开课吧