LeetCode Easy Summary

461. Hamming Distance: bit operation, xor

339. Nested List Weight Sum: simple recursion

500. Keyboard Row: HashMap<character, integer>存的是键盘上的字母和对应的行数;
    LinkedsList<String> res
    corner case: String[] words == null || length == 0;

259. Logger Rate Limiter(design a logger limiter, if within the timer, return false)
    HashMap<String, Integer> 存的是输入内容和时间

344.Reverse String: two pointers: 0; length - 1; while(i < j) swap;

346. Moving Average from data stream:Queue<Integer> = new LinkedList<>();

496. Next Greater Element 1: [2, 4], [1, 2, 3, 4] ->[3, -1]
    HashMap<Integer, Integer>; Stack<Integer> stack;
    HashMap<stack.pop(), num>
    while(!stack.isEmpty() && stack.peek() < num)

463. Island Perimeter: int islands, int neighbors;
    for(int i = 0; i < grid.length; i ++) for (int j = 0; j < grid[0].length; j ++)
    if(i < grid.length - 1 && grid[i + 1][j] == 1) neighbors ++
    return islands * 4 - neighbors * 2;

266. Palindrome Permutation: HashSet<>();

292. Nim Game: trick: move stones from 1 to 3, you are the first, the one who removes the last stone will be the winner. check whether n % 4 == 0;

136. Single Number: bit_operation. return and ^ nums[i];

448. Find All numbers Disappeared in an Array: O(n) required
    因为数组是［1，n］，所以这里面也可以表示index，我把每一个数－1算成是index，令这个index得到的数取负数，如果一个数是负数，就说明这个index出现了1次，如果一个数是正数，那么就说明没有对应他的index，也就是缺少的index。
104. Maximum Depth of Binary Tree: recursion; corner case: root == null -> return 0;
    return Math.max(MaxDepth(root.left), MaxDepth(root.right)) + 1;

243. Shortest Word Distance: two pointers. one pointer points to first word's index. the other points to second word's index.

371. Sum of Two Integers: return b == 0 ? a : getSum(a ^ b, (a & b) << 1); recursion; bit operation

226. Invert Binary Tree: Queue<TreeNode> queue; corner case: root == null -> return null;

258. Add Digits: O(1) time required; trick; return num == 0 ? 0 : (num % 9 == 0 ? 9 : num % 9);

283. Move Zeros: Without making a copy of the array required. Pointer: points to the position.
int interposition = 0; for(int num : nums): if(num != 0) nums[interposition ++] = num;
add zeros at the end;

530. Minimum Absolute Difference in BST: recursion; TreeSet<Integer>; corner case: root == null -> return min;
if(!set.isEmpty()): if(set.floor(root.val) != null) min = math.min(min, Math.abs(root.val - set.floor(root.val); set.ceiling;
set.add(root.val);
getMinimumdifference(root.left) getMinimumdifference(root.right);
return min;

506. Relative Ranks: int[][] pair; Arrays.sort(pair, (a, b) -> (b[0] - a[0]));

404. Sum of Left Leaves: recursion; corner case: if(root == null) -> return 0;
if(root.left != null): if(root.left.left == null && root.left.right == null) -> sum += root.left.val;
else{ sum += sumOfLeftLeaves(root.left)) final: sum += sumOfLeftLeaves(root.right); return sum;

350. Intersection of Two Arrays II: Queue<Integer> q; Arrays.sort(nums2); Arrays.sort(nums1);

543. Diameter of Binary Tree: recursion; corner case: if(root == null) -> return 0; int left = maxDepth(root.left); int right = maxDepth(root.right); max = Math.max(max, left + right); return Math.max(left, right) + 1;

108. Convert Sorted Array to Binary Search Tree: helper(int[]nums, int low, int high)
    corner case: if(low > high) -> return null; int mid = low + (high - low) / 2; node = new Treenode(nums[mid]); node.left = helper(nums, low, mid - 1); node.right = helper(nums, mid + 1, high); return node;

437. Path Sum III: (find the number of paths that sum to a given value)
backtrack(Treenode root, int curSum, int target, HashMap<Integer, Integer> map)

501. Find Mode in Binary Search Tree(find the most frequently occurred element in a given BST)
void traverse(TreeNode root, List<Integer> list): corner case: if(root == null) -> return;
traverse(root.left, list); if(prev != null){if(prev.val == root.val){count++;}else: count = 1; pre = root.
traverse(root.right, list);

107. Binary Tree Level Order Traversal II(List<List<Integer>>): Queue; no recursion!!!!
corner case: if(root == null) -> return res;
queue.offer(root);
while(!queue.isEmpty())把每个定点放在queue里面，然后查left，right，再放进去，最后把queue.poll.val放到sublist里面。（BFS)

235. Lowest Common Ancestor of a Binary search tree: recursion;
corner case: if(root == null || root == p || root == q): return root;
TreeNode left = lowestCommonAncester(root.left, p, q);
TreeNode right = lowestCommonAncester(root.right, p, q);
if(left != null && right != null): return root;
if(left != null): return left;
if(right != null): return right;
return null;

459. Repeated Substring Pattern(return true or false if it can be separated by repeated items)
从str的一半来算，for(int i = len / 2; i >= 1; i - -) i: length of repeated part.
if(len % i == 0): m = len / i; String subS = s.substring(0,i);int j, for(j = 1; i < m; j ++); if(!
subS.equals(s.substring(j * i, i + j * i)): break; if(j == m): return true; return false;

342. Power of four: return (Math.log10(num) / Math.log10(4)) % 1 == 0;

345. Reverse Vowels of A String: two pointers.

118. Pascal’s Triangle
List<List<Integer>> res = new ArrayList<List<Integer>>();
ArrayList<Integer> in = new ArrayList<>();
for(int i = 0; i < numRows; i ++): in.add(0, 1);for(int j = 1; j < in.size - 1; j ++): in.set(j, in.get(j) +
in.get(j + 1);;res.add(new ArrayList<Integer>(in));

257. Binary Tree Paths: recursion
private void getPath(TreeNode root, String path, List<String> res):
```
corner case: if root.left == null && root.right == null: res.add(path + root.val)
             if root.left != null: getPath(root.left, path + root.val + “->”, res);
             if root.right != null: getPath(root.right, path + root.val + “->”, res):
```

141. Linked list Cycle
corner case: head == null || head.next == null: return false;
fast = head.next; slow = head;
while(fast != slow): if(fast == null || fast.next == null): return false;
fast = fast.next.next;
slow = slow.next;