## 本周作业

### 简单：

* 写一个关于 HashMap 的小总结。  
  说明：对于不熟悉 Java 语言的同学，此项作业可选做。
* [有效的字母异位词](https://leetcode-cn.com/problems/valid-anagram/description/" \t "https://u.geekbang.org/lesson/_blank)（亚马逊、Facebook、谷歌在半年内面试中考过）
* [两数之和](https://leetcode-cn.com/problems/two-sum/description/" \t "https://u.geekbang.org/lesson/_blank)（近半年内，亚马逊考查此题达到 216 次、字节跳动 147 次、谷歌 104 次，Facebook、苹果、微软、腾讯也在近半年内面试常考）
* [N 叉树的前序遍历](https://leetcode-cn.com/problems/n-ary-tree-preorder-traversal/description/" \t "https://u.geekbang.org/lesson/_blank)（亚马逊在半年内面试中考过）
* HeapSort ：自学 [https://www.geeksforgeeks.org/heap-sort/](https://www.geeksforgeeks.org/heap-sort/" \t "https://u.geekbang.org/lesson/_blank)

### 中等：

* [字母异位词分组](https://leetcode-cn.com/problems/group-anagrams/" \t "https://u.geekbang.org/lesson/_blank)（亚马逊在半年内面试中常考）
* [二叉树的中序遍历](https://leetcode-cn.com/problems/binary-tree-inorder-traversal/" \t "https://u.geekbang.org/lesson/_blank)（亚马逊、字节跳动、微软在半年内面试中考过）
* [二叉树的前序遍历](https://leetcode-cn.com/problems/binary-tree-preorder-traversal/" \t "https://u.geekbang.org/lesson/_blank)（字节跳动、谷歌、腾讯在半年内面试中考过）
* [N 叉树的层序遍历](https://leetcode-cn.com/problems/n-ary-tree-level-order-traversal/" \t "https://u.geekbang.org/lesson/_blank)（亚马逊在半年内面试中考过）
* [丑数](https://leetcode-cn.com/problems/chou-shu-lcof/" \t "https://u.geekbang.org/lesson/_blank)（字节跳动在半年内面试中考过）
* [前 K 个高频元素](https://leetcode-cn.com/problems/top-k-frequent-elements/" \t "https://u.geekbang.org/lesson/_blank)（亚马逊在半年内面试中常考）

#### [242. 有效的字母异位词](https://leetcode-cn.com/problems/valid-anagram/)

class Solution {

    public boolean isAnagram(String s, String t) {

        if (s.length() != t.length())

            return false;

        char[] cs = s.toCharArray();

        char[] ct = t.toCharArray();

        int[] map = new int[26];

        int count = 0;

        for (int i = 0; i < cs.length; i++) {

            if (++map[cs[i] - 'a'] == 1) {

                count++;

            }

            if (--map[ct[i] - 'a'] == 0) {

                count--;

            }

        }

        return count == 0;

    }

}

#### [1. 两数之和](https://leetcode-cn.com/problems/two-sum/)

class Solution {

    public int[] twoSum(int[] nums, int target) {

        Map<Integer, Integer> cache = new HashMap<>();

        for (int i = 0; i < nums.length; ++i) {

            int cur = target - nums[i];

            if (cache.containsKey(cur)) return new int[]{cache.get(cur), i};

            cache.put(nums[i], i);

        }

        return new int[0];

    }

}

#### [589. N叉树的前序遍历](https://leetcode-cn.com/problems/n-ary-tree-preorder-traversal/)

class Solution {

    private List<Integer> res;

    public List<Integer> preorder(Node root) {

        res = new LinkedList<>();

        dfs(root);

        return res;

    }

    private void dfs(Node root) {

        if(root == null)    return;

        res.add(root.val);

        for(var child : root.children)

            dfs(child);

    }

}

#### [49. 字母异位词分组](https://leetcode-cn.com/problems/group-anagrams/)

class Solution {

    public List<List<String>> groupAnagrams(String[] strs) {

        HashMap<String, List<String>> ans = new HashMap<>();

        char[] cache = new char[26];

        for (String str : strs) {

            Arrays.fill(cache, 'a');

            for (char c : str.toCharArray()) cache[c - 'a']++;

            String key = String.valueOf(cache);

            if (!ans.containsKey(key)) ans.put(key, new ArrayList<>());

            ans.get(key).add(str);

        }

        return new ArrayList<>(ans.values());

    }

}

#### [94. 二叉树的中序遍历](https://leetcode-cn.com/problems/binary-tree-inorder-traversal/)

class Solution {

    public List<Integer> inorderTraversal(TreeNode root) {

        List<Integer> res = new ArrayList<Integer>();

        TreeNode predecessor = null;

        while (root != null) {

            if (root.left != null) {

                // predecessor 节点就是当前 root 节点向左走一步，然后一直向右走至无法走为止

                predecessor = root.left;

                while (predecessor.right != null && predecessor.right != root) {

                    predecessor = predecessor.right;

                }

                // 让 predecessor 的右指针指向 root，继续遍历左子树

                if (predecessor.right == null) {

                    predecessor.right = root;

                    root = root.left;

                }

                // 说明左子树已经访问完了，我们需要断开链接

                else {

                    res.add(root.val);

                    predecessor.right = null;

                    root = root.right;

                }

            }

            // 如果没有左孩子，则直接访问右孩子

            else {

                res.add(root.val);

                root = root.right;

            }

        }

        return res;

    }

}

#### [144. 二叉树的前序遍历](https://leetcode-cn.com/problems/binary-tree-preorder-traversal/)

class Solution {

  public List<Integer> preorderTraversal(TreeNode root) {

    LinkedList<TreeNode> stack = new LinkedList<>();

    LinkedList<Integer> output = new LinkedList<>();

    if (root == null) {

      return output;

    }

    stack.add(root);

    while (!stack.isEmpty()) {

      TreeNode node = stack.pollLast();

      output.add(node.val);

      if (node.right != null) {

        stack.add(node.right);

      }

      if (node.left != null) {

        stack.add(node.left);

      }

    }

    return output;

  }

}

#### [429. N叉树的层序遍历](https://leetcode-cn.com/problems/n-ary-tree-level-order-traversal/)

class Solution {

    public List<List<Integer>> levelOrder(Node root) {

        List<List<Integer>> ans = new ArrayList<>();

        Deque<Node> deque = new LinkedList<>();

        if (root == null) return ans;

        deque.add(root);

        while (!deque.isEmpty()) {

            List<Integer> level = new ArrayList<>();

            int size = deque.size();

            for (int i = 0; i < size; ++i) {

                Node node = deque.poll();

                level.add(node.val);

                deque.addAll(node.children);

            }

            ans.add(level);

        }

        return ans;

    }

}

#### [49. 丑数](https://leetcode-cn.com/problems/chou-shu-lcof/)

class Solution {

    public int nthUglyNumber(int n) {

        int p2=0,p3=0,p5=0;

        int[] dp=new int[n];

        dp[0]=1;

        for(int i=1;i<n;i++){

            dp[i]=Math.min(dp[p2]\*2,Math.min(dp[p3]\*3,dp[p5]\*5));

            if(dp[i]==dp[p2]\*2) p2++;

            if(dp[i]==dp[p3]\*3) p3++;

            if(dp[i]==dp[p5]\*5) p5++;

        }

        return dp[n-1];

    }

    // 一个十分巧妙的动态规划问题

    // 1.我们将前面求得的丑数记录下来，后面的丑数就是前面的丑数\*2，\*3，\*5

    // 2.但是问题来了，我怎么确定已知前面k-1个丑数，我怎么确定第k个丑数呢

    // 3.采取用三个指针的方法，p2,p3,p5

    // 4.index2指向的数字下一次永远\*2，p3指向的数字下一次永远\*3，p5指向的数字永远\*5

    // 5.我们从2\*p2 3\*p3 5\*p5选取最小的一个数字，作为第k个丑数

    // 6.如果第K个丑数==2\*p2，也就是说前面0-p2个丑数\*2不可能产生比第K个丑数更大的丑数了，所以p2++

    // 7.p3,p5同理

    // 8.返回第n个丑数

}

#### [347. 前 K 个高频元素](https://leetcode-cn.com/problems/top-k-frequent-elements/)

class Solution {

    public int[] topKFrequent(int[] nums, int k) {

        HashMap<Integer, Integer> eleTimesMap = new HashMap<>();

        for(int num : nums){

            eleTimesMap.put(num, eleTimesMap.getOrDefault(num, 0) + 1);

        }

        PriorityQueue<int[]> pq = new PriorityQueue<int[]>(new Comparator<int[]>(){

            public int compare(int[] a, int[] b){

                return a[1] - b[1];

            }

        });

        for(Map.Entry<Integer, Integer> eleTimesEntry : eleTimesMap.entrySet()){

            int ele = eleTimesEntry.getKey(), times = eleTimesEntry.getValue();

            if(pq.size() == k){

                if(pq.peek()[1] < times){

                    pq.poll();

                    pq.offer(new int[]{ele, times});

                }

            }else{

                pq.offer(new int[]{ele, times});

            }

        }

        int[] result = new int[k];

        for(int i = 0;i < k;i ++){

            result[i] = pq.poll()[0];

        }

        return result;

    }

}