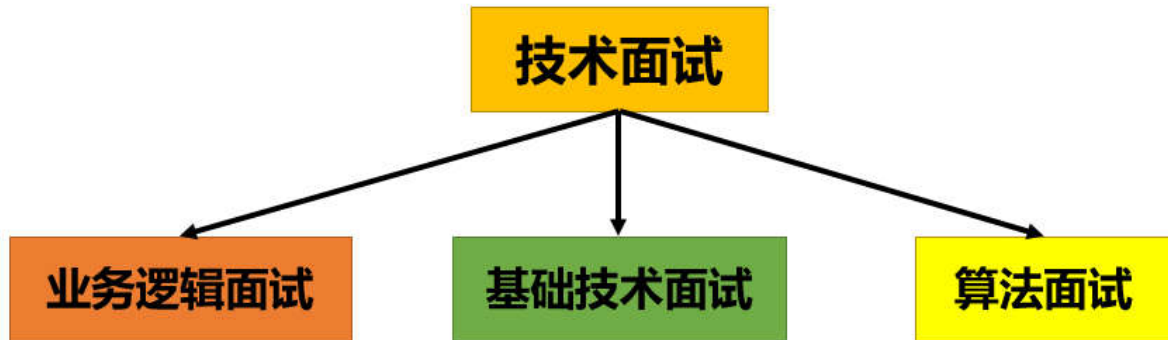


大家好，很高兴我们可以继续学习交流Java相关面试题目。本小节开始，我们主要进行高频算法题目的讲解。“手撕算法”应该算是技术岗位最通用的面试题目了。

在各大公司的面试中，有一个最基本的要求，那就是必须写点代码。技术面试一般情况下可以归纳为三大块，即业务逻辑面试，基础技术面试和算法面试。



牛客@我是祖国的花朵

业务逻辑面试就是让你讲述你的项目，并且进行针对性提问，考察你对项目是否足够熟悉与了解。基础技术面试就比较广了，所有涉及到的相关技术知识点都可以考察。一般情况下，面试官会留出20分钟左右的时间和我们一起研究探讨算法。

对于服务端开发同学来说，算法面试及其重要。在校招的面试中，一个算法题是否有思路并且可以完整的写出来，很多时候都直接决定了这轮面试的结果，因为校招毕竟是相当注重基础的考察。在社招的面试上，本轮的面试结果也会很大程度上受到算法题表现的影响。

### 为什么算法面试的重要性这么高？

- 首先，算法是一种通用的考察点，任何技术岗面试都可以进行考察。
- 其次，算法包含了太多的逻辑思维，可以考察应聘者思考问题的逻辑和解决问题的能力。
- 最后，连这么有难度的算法题你都可以搞定，那么其他只需要看看写写用用就可以掌握的基础知识和相关技术框架还怕学不会吗？

### 应该去哪里练习算法题呢？

- 当然是去牛客网啦，牛客网是一个专注于程序员的学习和成长的专业平台，集笔试面试系统、课程教育、社群交流、招聘内推于一体。我们可以在在线编程模块进行算法题的练习，对于找工作是一个不可多得的好帮手。
- 然后是LeetCode，这是一个美国的在线编程网站，上面主要收集了各大IT公司的笔试面试题，LeetCode的主要特点就是按照难易将题目分为了easy、medium和hard三种，并且在LeetCode上将题目进行了分类。在自己练习通过之后可以打开讨论区，看看别人解决问题的思路。

在本次的专刊中，我们主要从下边的七个考察点来交流，精选其中最高频的算法题目与大家进行交流。

- 排序和查找算法
- 单链表
- 二叉树
- 队列和栈
- 字符串
- 数组

- 其它算法

本小节中，我们主要交流探讨排序与查找算法，常见链表和二叉树的考察算法。好了，让我们一起来交流吧。

6

## (1) 排序与查找算法：

关于排序和查找算法的考察，在校招面试和社招面试中都是极其热门的，不外乎别的，仅仅是因为这就是最基础并且常用的算法。

### 排序算法分类：

常见的排序算法从排序方式上可以分为四种，即选择排序，交换排序，插入排序以及归并排序。

- 选择排序：直接选择排序和堆排序
- 交换排序：冒泡排序和快速排序
- 插入排序：直接插入排序，二分插入排序和希尔排序
- 归并排序：归并排序

关于排序算法的考察，考察点包括每一个排序算法的原理（排序方式），时间空间复杂度以及判断其是否稳定（得会分析）。这里我们给出最常见的快速排序的算法实现。

### 快速排序算法：

快速排序是一种分区交换排序算法。采用分治策略对两个子序列再分别进行快速排序，是一种递归算法。

#### 算法描述：

在数据序列中选择一个元素作为基准值，每趟从数据序列的两端开始交替进行，将小于基准值的元素交换到序列前端，将大于基准值的元素交换到序列后端，介于两者之间的位置则成为了基准值的最终位置。同时，序列被划分成两个子序列，再分别对两个子序列进行快速排序，直到子序列的长度为1，则完成排序。

#### 举例：


假设要排序的数组是a[6]，长度为7，首先任意选取一个数据（通常选用第一个数据）作为关键数据，然后将所有比它小的数都放到它前面，所有比它大的数都放到它后面，这个过程称为一趟快速排序。一趟快速排序的算法是：

- 设置两个变量i，j，排序开始的时候i=0；j=6；
- 以第一个数组元素作为关键数据，赋值给key，即key=a[0]；
- 从j开始向前搜索，即由后开始向前搜索（j--），找到第一个小于key的值，两者交换；
- 从i开始向后搜索，即由前开始向后搜索（i++），找到第一个大于key的值，两者交换；
- 重复第3、4步，直到i=j；此时将key赋值给a[i]；

例如：待排序的数组a的值分别是：（初始关键数据key=49）

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
	49	38	65	97	76	13	27
第一次交换后:	27	38	65	97	76	13	49
第二次交换后:	27	38	49	97	76	13	65
第三次交换后:	27	38	13	97	76	49	65
第四次交换后:	27	38	13	49	76	97	65

### 快速排序算法一趟循环示意图

 牛客@我是祖国的花朵

此时完成了一趟循环，将49赋值给a[3]，数据分为三组，分别为{27,38,13}{49}{76,96,65}，利用递归，分别对第一组和第三组进行排序，则可得到一个有序序列，这就是快速排序算法。

算法实现：

```

1  public void quickSort(int[] num, int left, int right) {
2      if (num == null)
3          return; //如果左边大于右边，则return，这里是递归的终点，需要写在前面。
4      if (left >= right) {
5          return;
6      }
7      int i = left;
8      int j = right;
9      int temp = num[i]; //此处开始进入遍历循环
10     while (i < j) { //从右往左循环
11         while (i < j && num[j] >= temp) { //如果num[j]大于temp值，则pass，比较下一个
12             j--;
13         }
14         num[i] = num[j];
15         while (i < j && num[i] <= temp) {
16             i++;
17         }
18         num[j] = num[i];
19         num[i] = temp; // 此处不可遗漏，将基准值插入到指定位置
20     }
21     quickSort(num, left, i - 1);
22     quickSort(num, i + 1, right);
23 }

```

### 查找算法：

关于查找算法的考察主要是二分查找算法。二分查找又称折半查找，优点是次数少，查找速度快，平均性能好；其缺点是要求待查表为有序表，且插入删除困难。因此，折半查找方法适用于不经常变动而查找频繁的有序列表。

#### 二分查找算法实现步骤：

- 首先，假设表中元素是按升序排列，将表中间位置记录的关键字与查找关键字比较，如果两者相等，则查找成功。

- 否则利用中间位置记录将表分成前、后两个子表，如果中间位置记录的关键字大于查找关键字，则进一步查找前面子表，否则进一步查找后面子表。
- 重复以上过程，直到找到满足条件的记录，使查找成功，或一直到子表不存在为止，此时查找失败。

**算法前提：**必须采用顺序存储结构；必须按关键字大小有序排列。

6

**实现方式：**包含递归实现和非递归实现两种方式。

**递归实现方式如下：**

```
1 private int halfSearch(int[] a,int left,int right,int target) {
2     int mid=left+(right-left)/2; // 防止整型溢出
3     int midValue=a[mid];
4     if(left<=right){
5         if(midValue>target){
6             return halfSearch(a, left, mid-1, target);
7         }else if(midValue<target) {
8             return halfSearch(a, mid+1, right, target);
9         }else {
10            return mid;
11        }
12    }
13    return -1;
14 }
```

**非递归实现方式如下：**

```
1 private int halfSearch(int[] a,int target){
2     int i=0;
3     int j=a.length-1;
4     while(i<=j){
5         int mid=i+(j-i)/2;
6         int midValue=a[mid];
7         if(midValue>target){
8             j=mid-1;
9         }else if(midValue<target){
10            i=mid+1;
11        }else {
12            return mid;
13        }
14    }
15    return -1;
16 }
```

## 排序与查找算法总结：

关于排序与查找算法，必须熟练完成常见算法的手写。包括快速排序，冒泡排序以及选择排序等。另外，排序算法的时间和空间复杂度也需要掌握。不要觉得这些都很简单，那么我来提个问题吧~

选择排序和冒泡排序的区别是什么？

小伙伴们可以在文章下边评论，我们一起交流学习~

## (2) 链表相关算法考察：

链表是一种在物理存储单元上非连续，非顺序的存储结构，由一系列结点组成。每个结点包括两个部分：一个是存储数据元素的**数据域**，另一个是存储下一个结点地址的**指针域**。

### 一条链表：



6

### 单个Node：



牛客@我是祖国的花朵

链表是一种常见的数据结构，由于对链表的操作往往涉及到了指针的移动，所以也是面试中常见的算法考察知识点。链表分为了单向链表，双向链表和循环链表。在面试中，由于时间有限，一般情况下都在考察单向链表。

我们可以定义如下：

```
1 class Node{
2     int val;
3     Node next;
4     public Node(int val){
5         this.val=val;
6     }
7 }
```

**链表考察点：**链表的操作主要就是对指针的操作，常见面试题目都在考察指针的操作是否合理与正确。

### 链表常见算法题：

- 单链表反转
- 合并有序单链表
- 找出单链表的中间节点
- 判断单链表相交或者有环
- 找出进入环的第一个节点
- 求单链表相交的第一个节点

### 典型例题分析：

鉴于面试时间有限，一道算法题目的时间通常不会超出20分钟，所以都会挑着最高频的题目进行考察，那么我们就一起来看两道常见链表算法题目吧。

**单链表反转：**比如1→2→3→4→5，反转之后返回5→4→3→2→1

### 反转步骤：

- 从头到尾遍历原链表，每遍历一个结点
- 将其摘下放在新链表的最前端。

- 注意链表为空和只有一个结点的情况。

#### 算法实现：

```
1 public static Node reverseNode(Node head){
2     // 如果链表为空或只有一个节点，无需反转，直接返回原链表表头
3     if(head == null || head.next == null)
4         return head;
5
6     Node reHead = null;
7     Node cur = head;
8     while(cur!=null){
9         Node reCur = cur;    // 用reCur保存住对要处理节点的引用
10        cur = cur.next;       // cur更新到下一个节点
11        reCur.next = reHead; // 更新要处理节点的next引用
12        reHead = reCur;      // reHead指向要处理节点的前一个节点
13    }
14    return reHead;
15 }
```

6

**合并两个有序的单链表：**给出两个分别有序的单链表，将其合并成一条新的有序单链表。

举例：1→3→5和2→4→6合并之后为1→2→3→4→5→6

#### 步骤：

- 首先，我们通过比较确定新链表的头节点，然后移动链表1或者链表2的头指针。
- 然后通过递归来得到新的链表头结点的next

#### 算法实现：

```
1 public static Node mergeList(Node list1 , Node list2){
2     if(list1==null)
3         return list2;
4     if(list2==null)
5         return list1;
6     Node resultNode;
7     if(list1.val<list2.val){ // 通过比较大小，得到新的节点
8         resultNode = list1;
9         list1 = list1.next;
10    }else{
11        resultNode = list2;
12        list2 = list2.next;
13    }
14    // 递归得到next
15    resultNode.next = mergeList(list1, list2);
16    return resultNode;
17 }
```

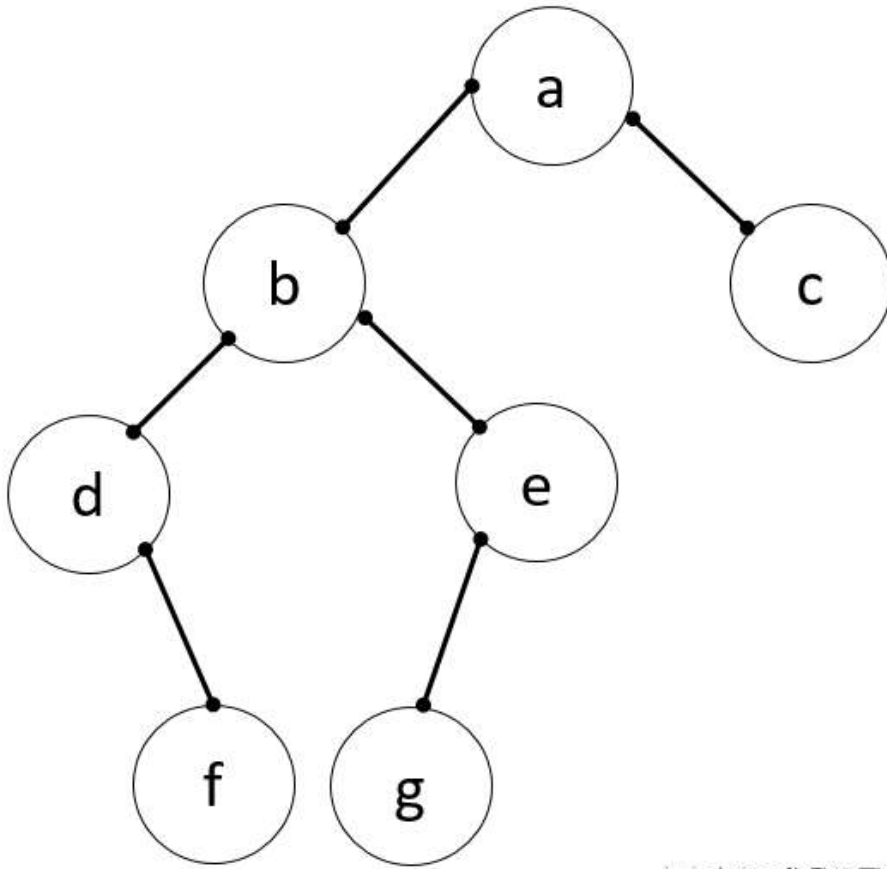
#### 链表总结：

关于链表这块的算法题目，我们一般采用多个指针即可搞定。由于面试时间有限，我们重点掌握单链表的反转，更进一步，面试中会考察单链表的分段反转。单链表的多段反转核心其实还是单链表的反转，这个算法留给小伙伴自行学习。

### (3) 二叉树相关算法考察：

二叉树是每个结点最多有两个子树的树结构。关于树的相关术语和性质，我们这里就不详细介绍了，请自行查阅相关资料。我们来解释下树的遍历方式吧。树的遍历方式分为深度优先方式和广度优先方式。假设，现在有这么一颗二叉树，我们来看看其遍历方式。

6



牛客@我是祖国的花朵

#### 深度优先遍历：

深度优先遍历方式分为了前序遍历，中序遍历和后序遍历。前中后指的是根root的遍历顺序。

- 前序遍历：根节点-左子树-右子树的遍历方式，a-b-d-f-e-g-c
- 中序遍历：左子树-根节点-右子树的遍历方式，d-f-b-g-e-a-c
- 后序遍历：左子树-右子树-根节点的遍历方式，f-d-g-e-b-c-a

#### 宽度优先遍历：

宽度优先遍历其实就是分层遍历方式，也就是按照二叉树的每一层依次遍历输出。对应我们的示例树，那就是a-b-c-d-e-f-g。

#### 二叉树的考察点：

二叉树的考察，主要是对指针的考察。出现最多的就是深度优先和宽度优先遍历方式的考察。在各个遍历方式中，不光涉及到了指针的移动，还涉及到了辅助数据结构，比如说队列和堆栈的使用。

#### 二叉树常见算法题：

- 分层遍历（宽度优先遍历）
- 前序遍历，中序遍历，后序遍历
- 求二叉树中的节点个数
- 求二叉树的深度
- 求二叉树第K层的节点个数
- 求二叉树中叶子节点的个数
- 判断两颗二叉树是否是相同的树
- 求二叉树中两个节点的最低公共祖先节点

6

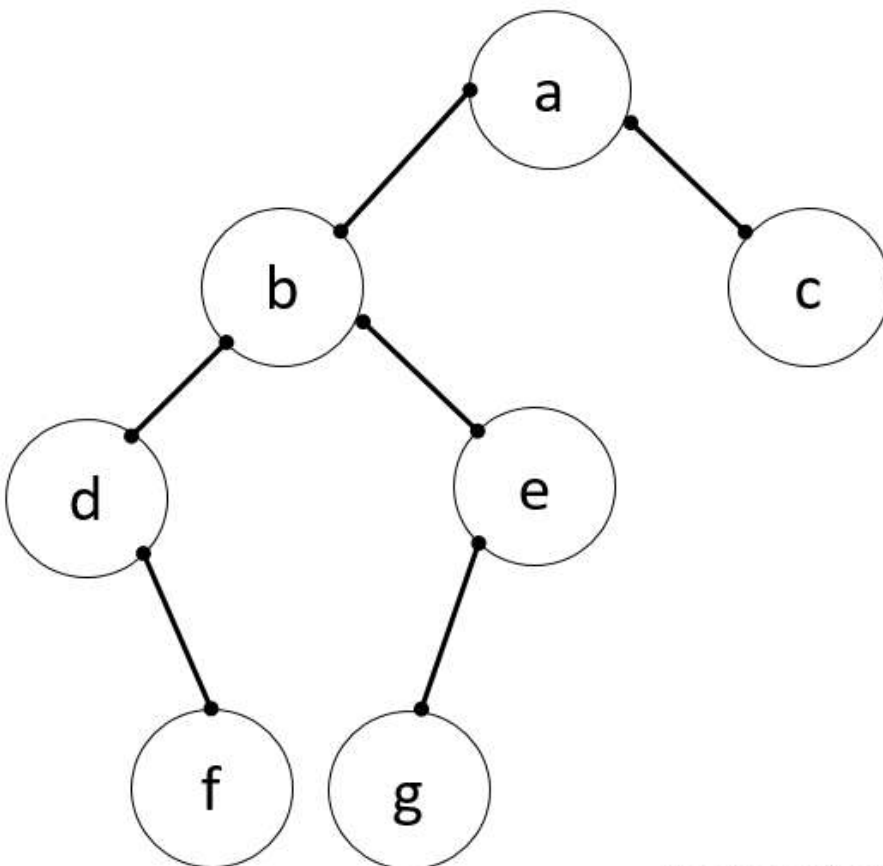
### 典型例题分析：

同样的，限于面试时间有限，我们这里挑选最高频的二叉树面试题目进行分析。在例题开始之前，我们先来定义如下的二叉树结构：

```
1 class TreeNode {  
2     int val;  
3     TreeNode left;  
4     TreeNode right;  
5     public TreeNode(int val) {  
6         this.val = val;  
7     }  
8 }
```

### 二叉树的分层遍历：

给出如下的二叉树，输出其分层遍历结果a-b-c-d-e-f-g。



牛客@我是祖国的花朵



**步骤:**

分层遍历符合先进先出的规律，可以使用队列做为辅助数据结构。

- 队列初始化，将根节点压入队列。
- 当队列不为空，进行如下操作：弹出一个节点，访问，若左子节点或右子节点不为空，将其压入队列。

6

a						
---	--	--	--	--	--	--

将root根节点压入队列

a	b	c				
---	---	---	--	--	--	--

当队列不为空时，弹出第一个节点a，然后将a节点的左右节点b和c分别压入队列

a	b	c	d	e		
---	---	---	---	---	--	--

弹出节点b，然后将b节点的左右节点d和e分别压入队列

## 二叉树分层遍历示意图

 牛客@我是祖国的花朵

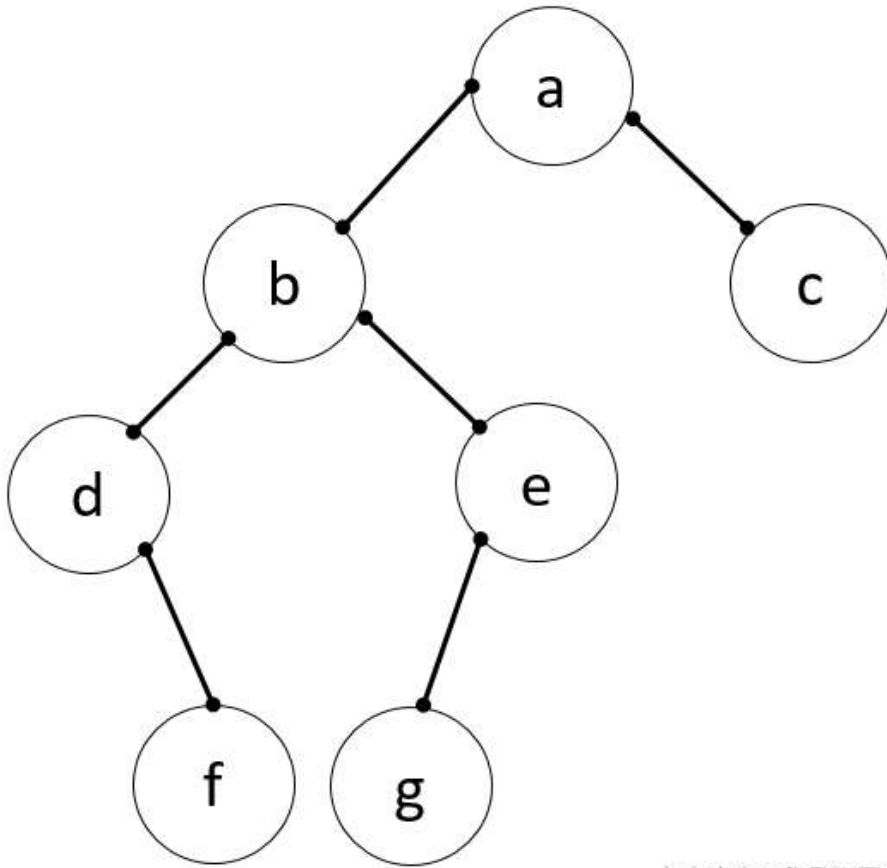
**算法实现:**

```

1 public static void levelTraversal(TreeNode root){
2     if(root==null)
3         return ;
4     LinkedList<TreeNode> queue = new LinkedList<TreeNode>();
5     queue.add(root); // 队列初始化，将根节点加入队列
6     while(!queue.isEmpty()){
7         TreeNode cur = queue.remove();
8         System.out.print(cur.val+" ");
9         if(cur.left!=null)
10            queue.add(cur.left);
11        if(cur.right!=null)
12            queue.add(cur.right);
13    }
14 }
```

**二叉树的前序遍历:**

还是同样的一颗二叉树，要求输出其前序遍历a-b-d-f-e-g-c。



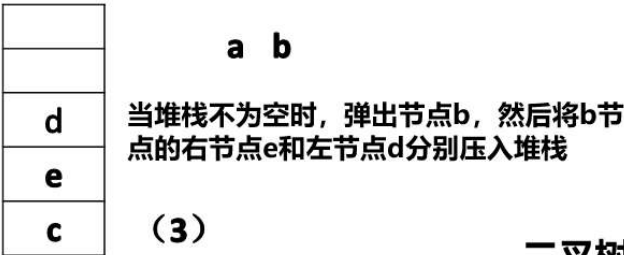
牛客@我是祖国的花朵

### 分析：

在二叉树的前序遍历中，遍历顺序为根左右，也就是左节点在右节点之前，可以考虑使用堆栈这种后进先出的数据结构来实现。

### 步骤：

- 使用一个辅助堆栈，初始时将根节点加入堆栈。
- 当堆栈不为空时，弹出一个节点，分别将其不为空的右子节点和左子节点加入堆栈。



二叉树前序遍历示意图

牛客@我是祖国的花朵

算法实现：

```
1 public static void preorderTraversal(TreeNode root){
2     if(root==null)
3         return ;
4     Stack<TreeNode> stack = new Stack<TreeNode>(); // 辅助stack
5     stack.push(root);
6     while(!stack.isEmpty()){
7         TreeNode cur = stack.pop(); // 出栈栈顶元素
8         System.out.print(cur.val+" ");
9         // 关键点：要先压入右孩子，再压入左孩子，这样在出栈时会先打印左孩子再打印右孩子
10        if(cur.right!=null)
11            stack.push(cur.right);
12        if(cur.left!=null)
13            stack.push(cur.left);
14    }
15 }
```

二叉树总结：

关于二叉树的面试题准备中，我们一定要装备好常见的分层遍历和前中后序遍历的实现方式，这些遍历中使用的辅助数据结构为先进先出的队列以及后进先出的堆栈。宽度优先和深度优先遍历做为二叉树考察中最高频的算法题目，希望大家一定加以理解与掌握。说白了，如果面试中考察你遍历方式，那就应该是送分题目。

本节总结：

在本小节中，我们较为详细的给大家阐述了排序与查找算法，链表以及二叉树的相关高频面试题。由于，我们并不是在进行算法与数据结构的入门，所以挑选了面试中最常出现的高频题目来进行了讲解与分析。这些题目，说白了就是送分题目，希望大家可以有效理解与掌握，把握住机会。

限于作者水平，文章中难免会有不妥之处。大家在学习过程中遇到我没有表达清楚或者表述有误的地方，欢迎随时在文章下边指出，我会及时关注，随时改正。另外，大家有任何话题都可以在下边留言，我们一起交流探讨。



周黑字

1#

什么时候更新??

发表于 2020-01-08 23:48:48

赞(0) 回复(1)

6

我是祖国的花朵 (作者) : 一周三更

2020-01-08 23:59:06

赞(1) 回复(0)

回复



还年轻多学习

2#

问老哥点赞好吧，当初5块钱真的值！

发表于 2020-01-11 00:45:17

赞(0) 回复(2)

我是祖国的花朵 (作者) : 第一波红利真是买到就是赚到呀!!!

2020-01-11 08:27:11

赞(0) 回复(0)

L1nxi : 50元大洋买的(☹)，不过是真的值，感谢老哥分享。

2020-04-10 14:56:33

赞(2) 回复(0)

回复



牧水s

3#

打卡

发表于 2020-02-16 21:13:36

赞(0) 回复(0)



我不会neralNetwork

4#

我擦，我花了19

发表于 2020-02-28 15:32:10

赞(0) 回复(0)



现在的菜鸡，未来的顶尖技术总监

5#

请问 中序遍历和后序遍历也能用栈实现吗 以前都是用递归实现

发表于 2020-05-27 11:22:12

赞(0) 回复(1)

我是祖国的花朵 (作者) : 必须可以呀

2020-06-07 17:48:21

赞(0) 回复(0)

回复



惊弦 N

6#

之前讲HashMap的时候有提到过红黑树，这个对于不同等级的面试者来说需要掌握到什么程度？

6

发表于 2020-06-07 16:24:29

赞(0) 回复(1)

我是祖国的花朵 N 作者： 可以这么说吧，基本上不需要掌握！！头条手撕红黑树只是大家玩的一个梗而已啦，有研究红黑树的时间和精力，还不如去看看好的开源项目源码 😊

2020-06-07 17:49:32

赞(0) 回复(0)

请输入你的观点

回复