



黑马程序员
www.itheima.com

传智播客旗下
高端IT教育品牌

数据结构与算法面试突击

讲师：唐僧

多年一线大厂研发经验，项目覆盖电商，银行，金融，办公自动化，在线教育等，有着非常丰富的项目开发经验，致力于研究大厂面试多年，有着丰富的大厂面经，另外在大数据，物联网等方面也有着深入的研究。



羡慕别人不如成就自己，20万优秀IT人的共同选择

如何评判代码写的好与坏？

```
mirror_mod = modifier_ob.  
    # Set mirror object to mirror.  
    mirror_mod.mirror_object =  
    operation == "MIRROR_X":  
    mirror_mod.use_x = True  
    mirror_mod.use_y = False  
    mirror_mod.use_z = False  
    operation == "MIRROR_Y":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = True  
    mirror_mod.use_z = False  
    operation == "MIRROR_Z":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = False  
    mirror_mod.use_z = True  
  
    # Select the mirror object.  
    if mirror_ob.select == 0:  
        bpy.context.scene.objects.active = mirror_ob  
        ("Selected" + str(modifier_ob.name) + " mirror")  
        mirror_ob.select = 1  
        = bpy.context.selected_objects[0].name  
        data.objects[one.name].select = 1  
  
    print("please select exactly one mirror object")  
  
-- OPERATOR CLASSES --  
  
    types.Operator):  
        X mirror to the selected object.  
        object.mirror_mirror_x  
        mirror X  
  
    def execute(self, context):  
        if context.object is not None:
```



如何评判代码写的好与坏？

时间复杂度 空间复杂度

时间复杂度表达了代码的执行时间（次数）随着数据规模增长的变化趋势

空间复杂度表达了算法占用的存储空间与数据规模之间的增长关系

空间换时间 OR 时间换空间？ 需要根据实际应用场景来权衡



常见时间复杂度

$O(1)$: 常量级复杂度

```
public void test(int n){
    int i=0;
    int sum=0;
    for(;i<10000000;i++){
        sum = sum+i;
    }
    System.out.println(sum);
}
```

$O(n)$: 线性复杂度

```
public void test(int n){
    int i=0;
    int sum=0;
    for(;i<n;i++){
        sum = sum+i;
    }
    System.out.println(sum);
}
```

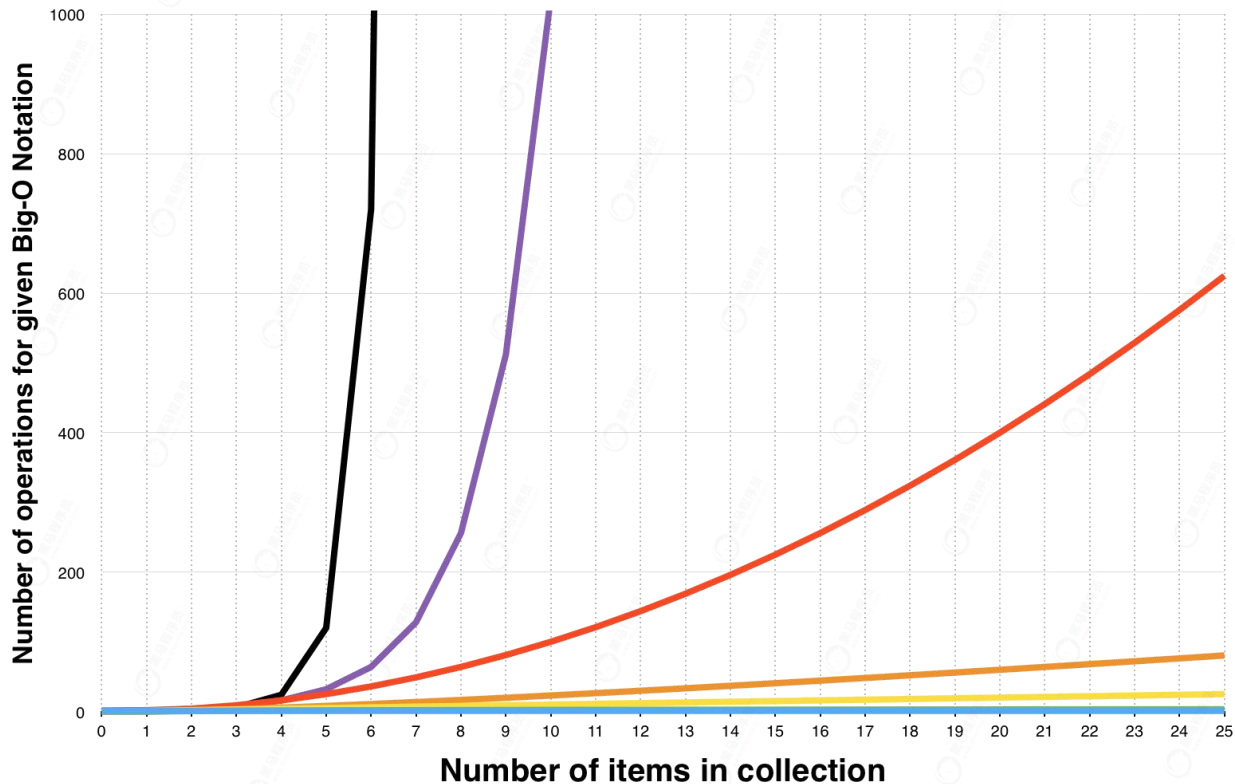
$O(\log n)$: 对数级复杂度

```
public void test(int n){
    int i=1;
    while(i<=n){
        i = i * 2;
    }
}
```



常见时间复杂度

$O(1)$ $O(\log n)$ $O(n)$ $O(n \log n)$ $O(n^2)$ $O(2^n)$ $O(n!)$





常见空间复杂度

$O(1)$

```
public void test(int n){
    int i=0;
    int sum=0;
    for(;i<n;i++){
        sum = sum+i;
    }
    System.out.println(sum);
}
```

$O(n)$

```
void print(int n) {
    int i = 0;
    int[] a = new int[n];
    for (i; i < n; ++i) {
        a[i] = i * i;
    }
    for (i = n-1; i >= 0; --i) {
        System.out.println(a[i]);
    }
}
```

A person in a grey suit and patterned tie is holding a white rectangular sign with the word "OFFER" written in large, dark, sans-serif capital letters. The background is a dark, blurred image of a person's legs and feet on a red carpet, with the word "Alibaba" visible in large white letters on the carpet.

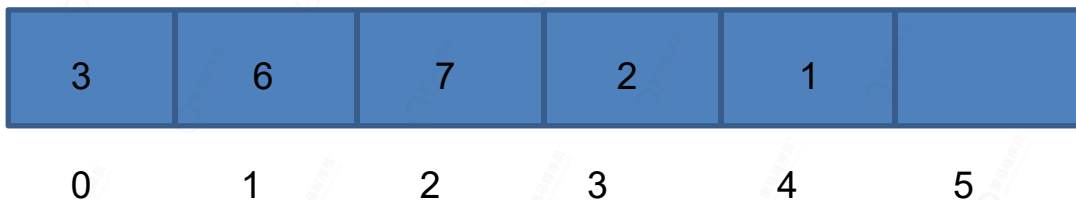
OFFER

数组与链表



> 数组 (Array)

用连续的内存空间存储相同类型的数据



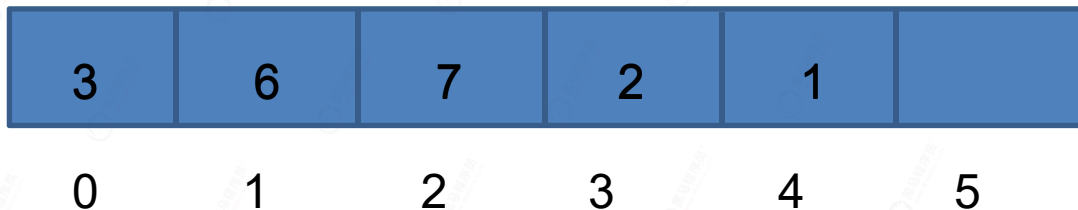
存储特点:

- 1: 每个存储空间占多大取决于存储什么类型的数据
- 2: 数组整体的存储空间大小在申请时已确定，后期无法修改
- 3: 数组的所有存储空间都是连续的，中间不间断

动态数组: 可以进行动态扩容，扩容--java.util.ArrayList

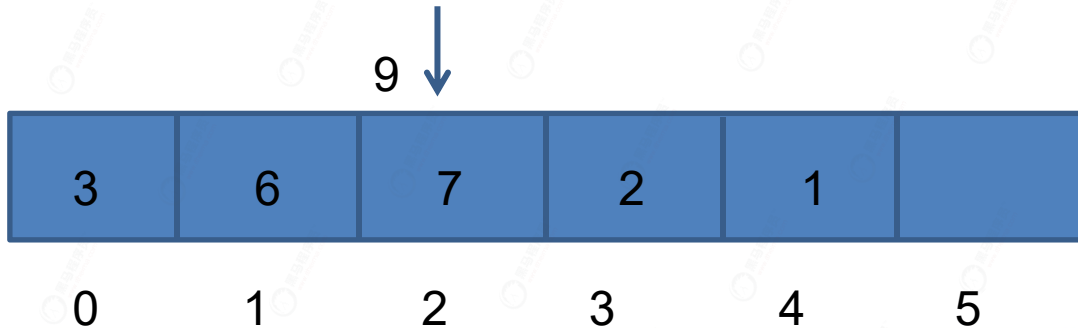


数组 (Array)



操作特点:

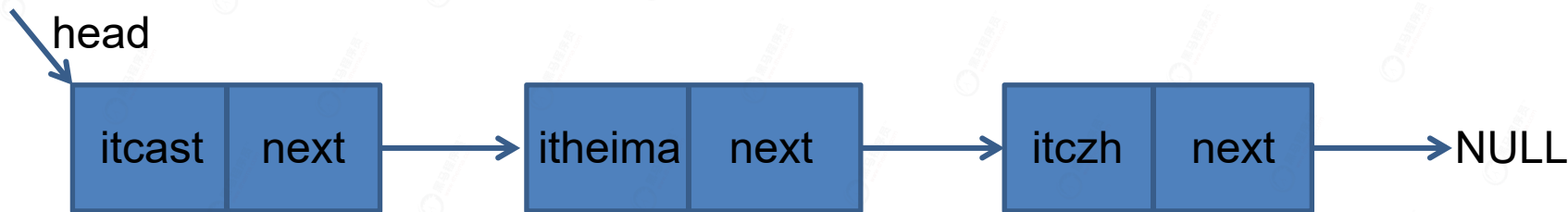
- 1: 数据的查询 (获取) 时间复杂度是 $O(1)$ 的
- 2: 数据的插入和删除操作时间复杂度是 $O(n)$ 的





> 链表 (Linked List)

物理存储单元上非连续、非顺序的存储结构，链表中的每一个元素称之为结点 (Node)，结点之间用指针连接起来。



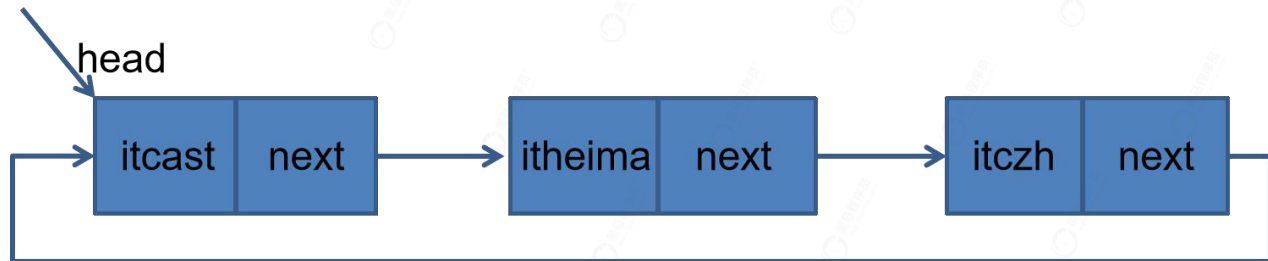
存储特点:

- 1: 存储空间非连续，是零散的
- 2: 链表结点是动态生成的，且天生具备动态扩容的特点

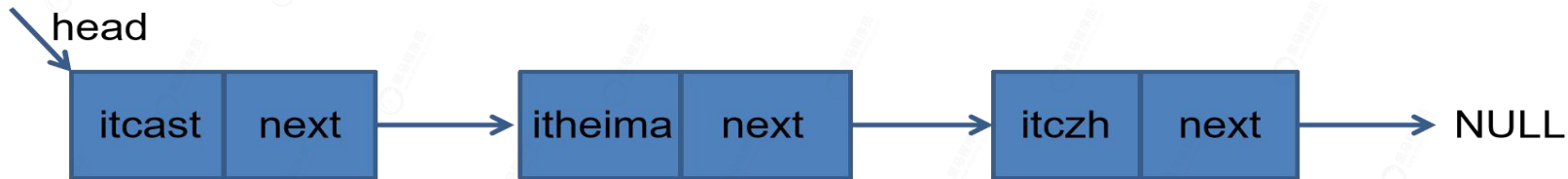


> 链表 (Linked List)

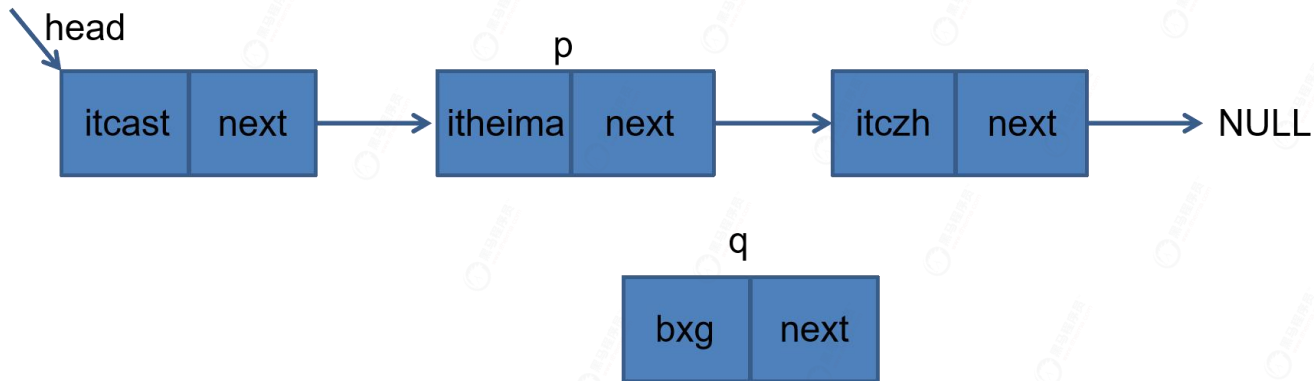
链表根据其结点之间的连接形式分单向链表，双向链表，循环链表，双向循环链表。



> 链表 (Linked List)



- 操作特点:**
- 1 查询操作的时间复杂度为 $O(n)$
 - 2 添加和删除操作的时间复杂度为 $O(1)$



$q.next = p.next$
 $p.next = q$

A person is seen from behind, holding a pen and looking at a document, likely a resume, which is the background of the image. The document has some text and a small blue logo. The overall scene is dimly lit, with the person's hands and the document being the primary focus.

快速领略面试中的数据结构与算法



2020年 - 秋招 - 顺丰

选择题：以下关于链表和数组说法正确的是()

- A.数组从栈中分配空间，链表从堆中分配空间
- B.数组插入或删除元素的时间复杂度 $O(n)$ ，链表的时间复杂度 $O(1)$
- C.数组利用下标定位，时间复杂度为 $O(1)$ ，链表定位元素时间复杂度 $O(n)$
- D.对于add和remove，ArrayList要比LinkedList快

答案：BC



2020年 - 秋招 - 招商银行

选择题：以下选项，对于线性链表的描述中正确的是()

- A.存储空间必须连续，且前件元素一定存储在后件元素的前面
- B.存储空间必须连续，且各元素的存储顺序是任意的
- C.存储空间不一定是连续，且各元素的存储顺序是任意的
- D.存储空间不一定是连续，且前件元素一定存储在后件元素的前面

答案：C



2020年 - 秋招 - 小米

选择题：关于链表，正确的是（）

- A. 无需事先估计空间
- B. 支持随机访问
- C. 增删不必挪动元素
- D. 所需空间与线性表长度成正比，并且地址连续
- E. 插入一个元素所需挪动元素的平均个数为 $n/2$

答案：AC

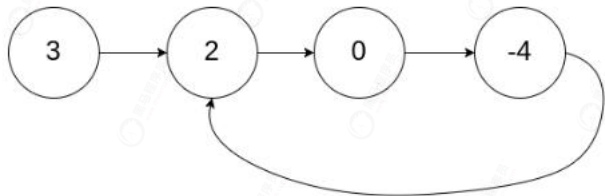


2020年 - 秋招 - 阿里，美团

141. 环形链表

给定一个链表，判断链表中是否有环。

示例 1:



142. 环形链表 II



2020年 - 秋招 - 字节，滴滴

206. 反转链表

反转一个单链表。

示例:

输入: 1->2->3->4->5->NULL

输出: 5->4->3->2->1->NULL

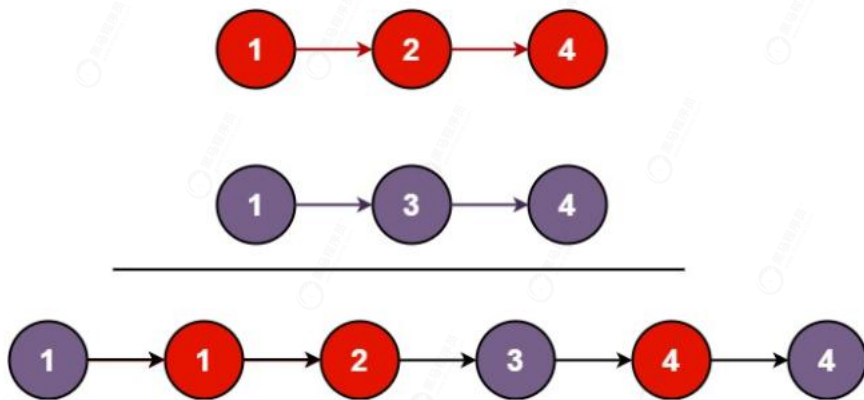


> 2020年 - 秋招 - 字节，小米

21. 合并两个有序链表

将两个升序链表合并为一个新的 **升序** 链表并返回。新链表是通过拼接给定的两个链表的所有节点组成的。

示例 1:



进阶

23. 合并K个升序链表



2020年 - 秋招 - 字节，滴滴

剑指 Offer 22. 链表中倒数第k个节点

输入一个链表，输出该链表中倒数第k个节点。为了符合大多数人的习惯，本题从1开始计数，即链表的尾节点是倒数第1个节点。

例如，一个链表有 6 个节点，从头节点开始，它们的值依次是 1、2、3、4、5、6。这个链表的倒数第 3 个节点是值为 4 的节点。

示例：

给定一个链表：1->2->3->4->5，和 $k = 2$ 。

返回链表 4->5。



2020年 - 秋招 - 字节，腾讯，美团，阿里

1. 两数之和

给定一个整数数组 `nums` 和一个整数目标值 `target`，请你在该数组中找出 **和为目标值** 的那 **两个** 整数，并返回它们的数组下标。

你可以假设每种输入只会对应一个答案。但是，数组中同一个元素不能使用两遍。

你可以按任意顺序返回答案。



2020年 - 秋招 - 字节，华为，爱奇艺

15. 三数之和

给你一个包含 n 个整数的数组 `nums`，判断 `nums` 中是否存在三个元素 a, b, c ，使得 $a + b + c = 0$ ？请你找出所有和为 0 且不重复的三元组。

注意：答案中不可以包含重复的三元组。

进阶

18. 四数之和

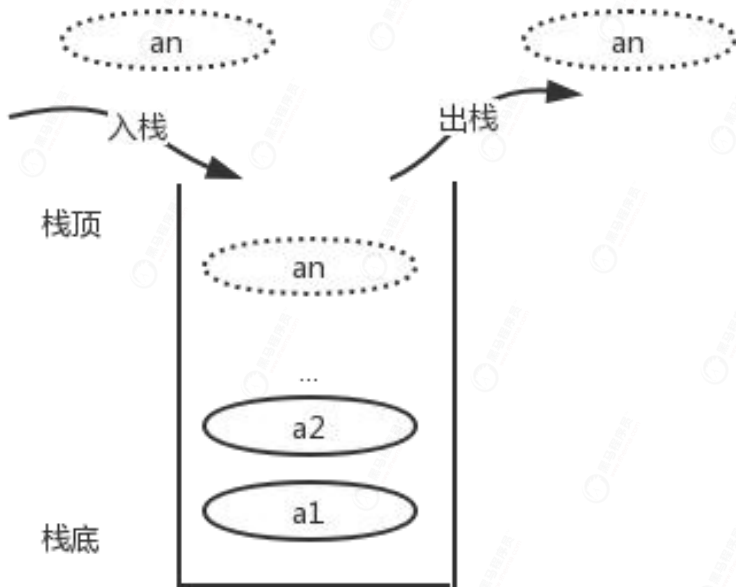
A person in a suit is holding a white sign with the word "OFFER" written on it. The background is a blurred image of a person's legs and feet walking on a red carpet with the "Alibaba" logo.

OFFER

栈，队列，哈希表



> 栈 (Stack)



底层实现： 数组或链表

操作特点：

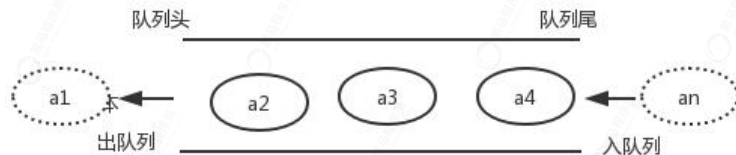
- 1: 数据入栈(push)的时间复杂度是 $O(1)$
- 2: 数据出栈(pop)的时间复杂度是 $O(1)$

面试题目：

- 1: 20. 有效的括号
- 2: 155. 最小栈



> 队列 (Queue)



底层实现： 数组或链表

操作特点：

入队列和出队列操作的时间复杂度均为 $O(1)$

面试题目：

- 1: 622. 设计循环队列
- 2: 703. 数据流中的第K大元素

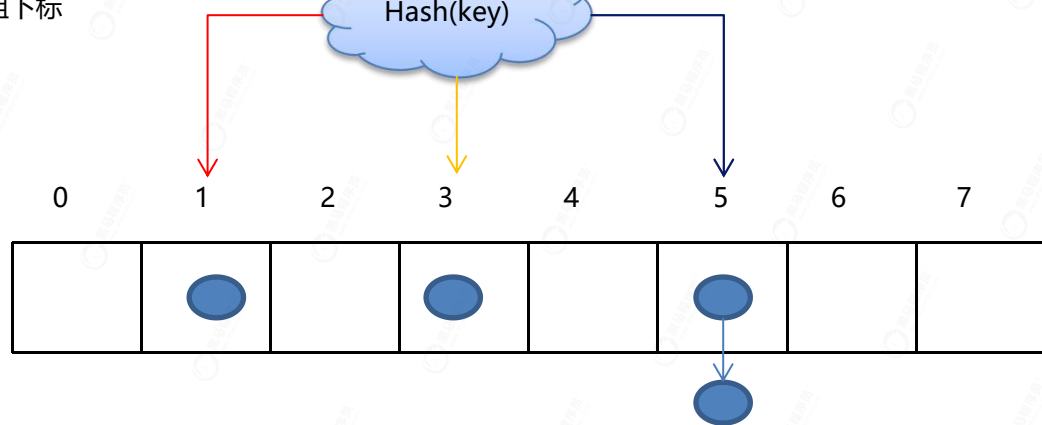
> 哈希表

> 哈希函数

key



hash函数：将key映射到数组下标



哈希冲突

开放寻址法

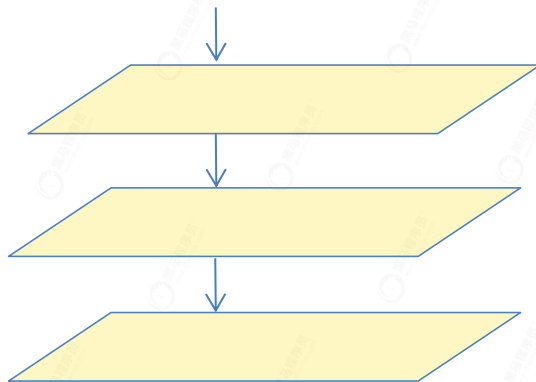
拉链法

A person in a grey suit and patterned tie is holding a white rectangular sign with the word "OFFER" printed in large, dark, sans-serif capital letters. The background is a blurred office or public space with a large "Alibaba" logo on the floor and the lower legs of other people.

OFFER

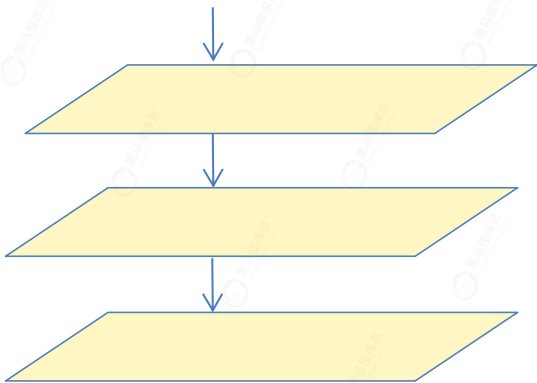
递归

> 递归(Recursion)





递归(Recursion)



在编程语言中对递归可以简单理解为：
方法自己调用自己，只不过每次调用时参数不同而已

```
int f(int n) {  
    if (n == 1){  
        return 1;  
    }  
    //do something  
    int a = f(n-1)  
    //do something  
    return a + 1;  
}
```

```
public void recur(int level,int param) {  
    //part1: terminal 递归终止条件  
    if (level > MAX_LEVEL) {  
        //处理结果数据  
        .....  
        return;  
    }  
    //part2: process current logic  
    process(level,param);  
    .....  
    //part3: drill down  
    recur(level+1,param);  
  
    //part4: clean  
    .....  
    return xxx;  
}
```



2020年 - 秋招 - 头条，美团，滴滴，

70. 爬楼梯

假设你正在爬楼梯。需要 n 阶你才能到达楼顶。

每次你可以爬 1 或 2 个台阶。你有多少种不同的方法可以爬到楼顶呢？

注意：给定 n 是一个正整数。

进阶

- 1: 傻递归的问题
- 2: 动态规划



那就是棵树而已，
儿子…

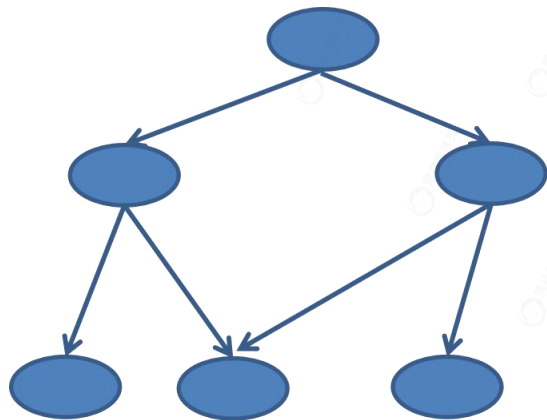
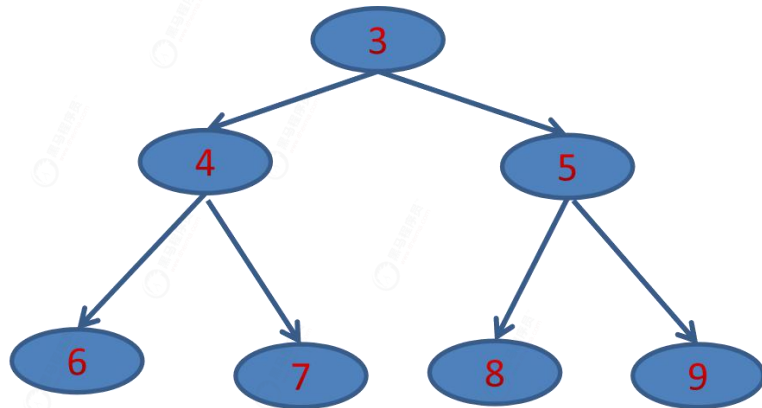
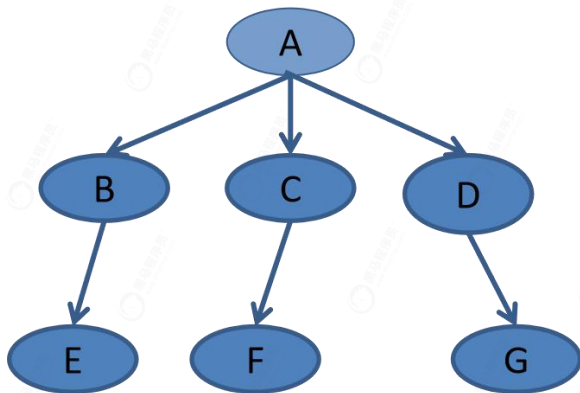
掌握高级数据结构：二叉树，二叉搜索树

可是，那是棵二叉树…





> 树 (Tree)

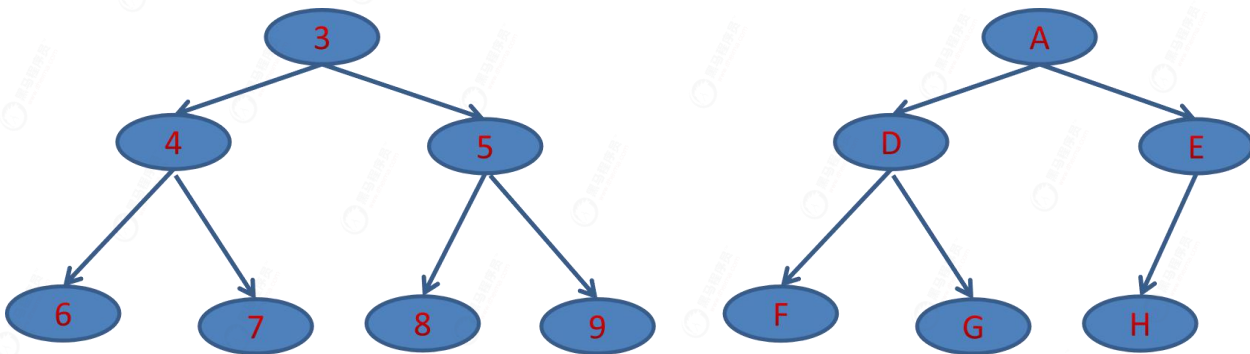


树的特点:

1. 每个节点都只有有限个子节点或无子节点;
2. 没有父节点的节点称为根节点;
3. 每一个非根节点有且只有一个父节点;
4. 除了根节点外，每个子节点可以分为多个不相交的子树;
5. 树里面没有环路(cycle)

> 二叉树 (Binary Tree)

二叉树，顾名思义，每个节点最多有两个子节点，分别是左子节点和右子节点

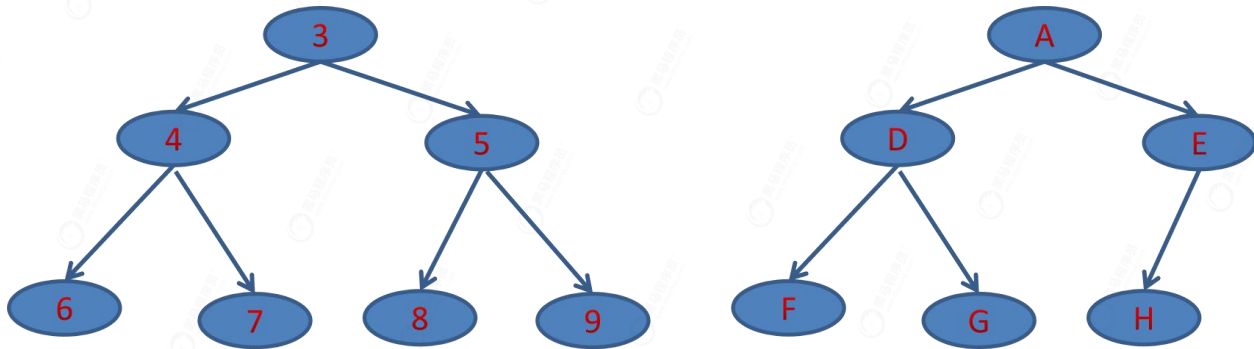


存储特点：

很多其他高级数据结构都是基于二叉树，他们的操作特点各有不同，但从存储上来说底层无外乎就是两种：数组存储，链式存储。



> 二叉树 (Binary Tree)



二叉树的遍历：前序遍历，中序遍历，后序遍历

前序遍历：对于树中的任意节点来说，先打印这个节点，然后再打印它的左子树，最后打印它的右子树。

3467589

中序遍历：对于树中的任意节点来说，先打印它的左子树，然后再打印它本身，最后打印它的右子树。

6473859

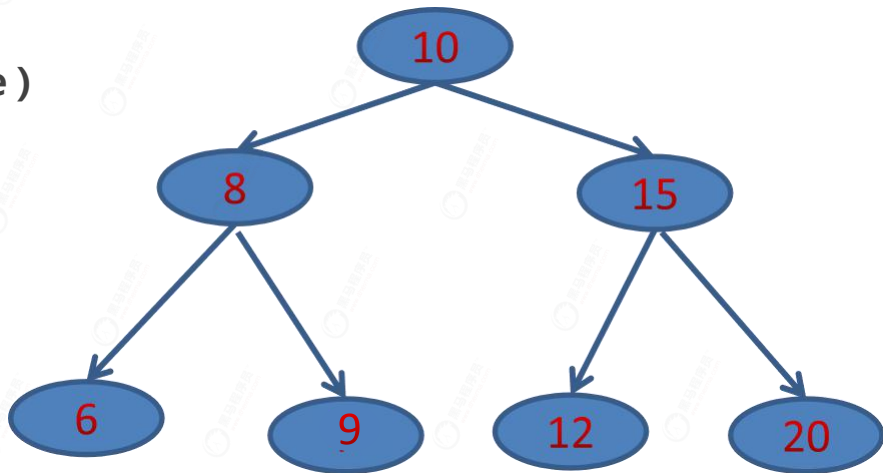
后序遍历：对于树中的任意节点来说，先打印它的左子树，然后再打印它的右子树，最后打印这个节点本身。

6748953



> 二叉搜索树 (Binary Search Tree)

二叉查找树要求，在树中的任意一个节点，其左子树中的每个节点的值，都要小于这个节点的值，而右子树节点的值都大于这个节点的值：



1. 若任意节点的左子树不空，则左子树上所有节点的值均小于它的根节点的值；
2. 若任意节点的右子树不空，则右子树上所有节点的值均大于它的根节点的值；
3. 任意节点的左、右子树也分别为二叉查找树；
4. 没有键值相等的节点。

二叉搜索树的中序遍历结果是递增序列



2020年 - 秋招 - 腾讯

94. 二叉树的中序遍历

给定一个二叉树，返回它的中序遍历。

示例:

```
输入: [1,null,2,3]
```

```
  1
   \
    2
   /
  3
```

```
输出: [1,3,2]
```



102. 二叉树的层序遍历



黑马程序员

www.itheima.com

传智播客旗下高端IT教育品牌

THANKS

www.itheima.com