总的来说，这些算法题太细碎了。如果能够找到整理的比较好的就好了。

接下来要做的无非就是熟悉 树 ArrayList这种的基本的运算。以及变化。其次要总结思想以及解题方式。

不过不知道有没有专门总结这部分的书。曾经用的算法笔记，树也好， 图也好，基本的算法都总结到了。不过剑指offer这种题目要什么算法。只要熟练掌握各种用法，就可以了。Stack 等等的函数好好掌握就行了

要先总结各种常用的方法等等。

Import java.until.\*

ArrayList



List.size()

List.clear()

List.add()

List.contains()

List.get()

List.indexOf()

List.remove()

List.set() ???这个方法是 在括号里放什么？

总之有很多有用的功能。

ArrayList里似乎也要放封装类

数组list  
ArrayList<int[]> list = new ArrayList<int[]>();  
整数list  
ArrayList<Integer> list = new ArrayList<Integer>();

Stack<Integer> stack1 = new Stack<Integer>();

**1、注意 这个 stack1.size()**

**1 boolean empty()**

测试堆栈是否为空。

**2 Object peek( )**

查看堆栈**顶部**的对象，但不从堆栈中移除它。

**3 Object pop( )**

移除堆栈顶部的对象，并作为此函数的值返回该对象。

**4 Object push(Object element)**

把项压入堆栈顶部。

5 int search(Object element)

返回对象在堆栈中的位置，以 1 为基数。

6、 有 size（）方法

**HashMap**

**1 clear()**

**clear() 的作用是清空HashMap。它是通过将所有的元素设为null来实现的。**

2 containsKey()

containsKey() 的作用是判断HashMap是否包含key。

3 containsValue()

containsValue() 的作用是判断HashMap是否包含“值为value”的元素。

4 entrySet()、values()、keySet()

它们3个的原理类似，这里以entrySet()为例来说明。

entrySet()的作用是返回“HashMap中所有Entry的集合”，它是一个集合。实现代码如下：

**5 get()**

**get() 的作用是获取key对应的value，它的实现代码如下：**

**.6 put()**

**put() 的作用是对外提供接口，让HashMap对象可以通过put()将“key-value”添加到**HashMap中。

7 putAll()

putAll() 的作用是将"m"的全部元素都添加到HashMap中，它的代码如下：

**8 remove()**

**remove() 的作用是删除“键为key”元素**

HashMap<String, String> Map = new HashMap<String, String>(){

{

put("A", "a");

put("B", "b");

}

};

HashMap中 无论是key 还是value都不能是基本数据类型。

原始类型   封装类

boolean      Boolean

char           Character

byte           Byte

short          Short

int              Integer

long           Long

float           Float

double      Double

**deque（）**

https://www.cnblogs.com/bushi/p/6681543.html

https://blog.csdn.net/u013534893/article/details/53067946

**list()**

https://www.cnblogs.com/epeter/p/5648026.html

**queue()**

**https://blog.csdn.net/zhangjian15/article/details/77012687**

LinkedList 除了实现了List和Deque外，还实现了**Queue**接口(队列)。  
Queue是先进先出队列 **FIFO**，常用方法：  
**offer**在最后添加元素  
**poll**取出第一个元素  
**peek**查看第一个元素

Array

**1、Array.length**

新建int数组

新建Array集合

Return 数组？

ArrayList heros = new ArrayList(); 这东西也是数组，所以不用《integer》 而且还没有上限

StringBuffer newStr=new StringBuffer();

Stack<Integer> stack1 = new Stack<Integer>();; 这几个也不知道上限，

题目总结：

疑问：

**String 方法**

char charAt(int index) 返回index处的char值

int comparaTo(String str) 把这个字符按照字符串和另一个对象比较

char[] toCharArray() 将此字符串转换为一个新的字符数组。

注意：我们先以背题目为主，先确保能够像掌握题库一样掌握再说。 而由于算法题复习的成本过高了。所以我们之后复习的时候优先靠思路为主，没思路的看一下。过几天再看一遍，但是不要动手。 当看到一题，觉得有思路了后。才开始做。在做得过程中。 总结一些在代码过程中的遇到的动手时的难点，总结一下，并且作为问题写在这里。 除了少数像数据结构的不太好总结，其他的一般都是比较容易的。

代码题的手感和熟练度还是重要的。在我们掌握之后，我们也最好循环天天刷。

不需要经常复习，因为这样时间上不划算，因为只要考前能够迅速过一遍即可。

1 二维数组中的查找：

2 替换空格的方法 考察StringBuffer的方法。重做。

http://www.runoob.com/java/java-stringbuffer.html

3 从尾到头打印链表 无非是反转一下。

4 重建二叉树 方法会，但是却没能够灵活运用。

5 用两个栈实现队列。 栈的 判空方法 st1.empty()；

6 旋转数组的最小数字

7 斐波那契数列 记住用递归做的典型即可

8 跳台阶， 同样的

9\* 变态跳台阶 属于那种，很难想到的一种解法

10 矩形覆盖 类似的递归

11-- 二进制中1的个数 负数的部分不会。难点1 取不同范围的数

难点2 负数正数的变化关系？ 答案是一个很难想到的巧妙的方法。

12-- 数值的整数次平方 其他类，先放着

13 调整数组顺序使奇数位于偶数前面 两种方法，你好好做一下把，太烂了 2暴力法。

14 链表中倒数第k个结点

15 反转链表 记住固定的算法即可

16 合并两个排序的链表

17 树的子结构

18 二叉树的镜像

19--顺时针打印矩阵 又难又偏

20 包含min函数的栈 迭代器法以及 辅助栈法都值得一看，我的比较赖皮。

21 栈的压入、弹出序列， 太典型了，所以你背下来解法吧

22 从上往下打印二叉树 **队列的插入，弹出，判空的方法**。

23 二叉搜索树的后序遍历序列 这题是蛮难的

24 二叉树中和为某一值的路径 PAT甲级中的水题

25 复杂链表的复制 我第一遍读没理解意思，

26 二叉搜索树与双向链表 嘀嘀打车，当时被这道题搞死了，记住怎么做即可

27 字符串的排列 好难啊。S级难度。 看都看不懂解法

28 数组中出现次数超过一半的数字

29 最小的k个数

30 连续子数组的最大和 很经典的动态规划，当然要掌握。

31-- 整数中1出现的次数 我愿意给它S级的难度，但是答案貌似挺容易

32 把数组排成最小的数。 虽然用sort工具不是最佳，但确实很方便。

<https://blog.csdn.net/lx_nhs/article/details/78871295> comparator

33-- 丑数， 所有其他类型的题目都这么难吗？

34 第一个只出现一次的字符 HashMap的使用

35 数组中的逆序对 经典的归并法，用暴力法的时间完全不够。应该是很重要的题目。嗯，而且很难看懂

36 两个链表，找出第一个公共结点

37 数字在排序数组中出现的次数

38 二叉树的深度

39 平衡二叉树

40 数组中只出现一次的数字， 用的是散列表的方法。很好的想法。值得与前面的HashMap对照。 但是这道题目实质上想考察的是异或的性质。 是很经典的一种做法。

41 和为S的连续正数序列

42 和为S的两个数字 以上两题都是使用同一种方法，很经典的方法。

43 左旋转字符串 这里有关反转的string的方法我需要掌握。

http://how2j.cn/k/number-string/number-string-manipulate/325.html#nowhere

44 翻转字符串顺序列 也是关于string的方法应用的。

45 扑克牌顺子

// 如果数字出现了一次

if((flag >> curNum) & 1 == 1){

return false;

}

// 按位保存数字出现次数，比如0110表示，0出现0次，1出现1次，2出现1次，3出现0次。

flag |= 1 << curNum; 主要是这个看不懂？？

46-- 孩子们的游戏

47-- 1+2+…+n 做题有两种思考方向，其中一种是递归

48-- 不用加减乘除的加法 ？？？？

49 把字符串转换成整数 A~Z,a~z,0~9的ASCII码

https://zhidao.baidu.com/question/67956528.html

50 数组中重复的数字 这题乖乖的， 关于boolean数组越界的问题？？

51 构建乘积数组 虽然是单独完成的，但是可以浏览一遍自己的思路。

实际上有更高效的算法。

52\* 正则表达式匹配 这题是难题中的战斗机，赌不会考把，我也不掌握。

53-- 表示数值的字符串 这题实际上很没意义，就是考分类，分类。。。因为没有考察任何数据结构算法

54 字符流中第一个不重复的字符 这题应该算是前面字符串题目中的加强版

55 链表中环的入口结点

56 删除链表中重复的结点

57 二叉树的下一个结点

58 对称的二叉树

59 按之字形顺序打二叉树

60 把二叉树打印成多行

61 序列化二叉树

62 二叉搜索树的第k个结点

63 数据流中的中位数

64 滑动窗口的最大值 我第一时间想到的是暴力法，但是最好的貌似是双端队列

65 矩阵中的路径

66 机器人的运动范围

**数组(11道**)：

剑指Offer（一）：二维数组中的查找 思路最重要，会思路即可，本身想到的方法太笨重了 实践：

剑指Offer（六）：旋转数组的最小数字

剑指Offer（十三）：调整数组顺序使奇数位于偶数前面 两种方法，你好好做一下把，太烂了 2暴力法。 实践：就是写一遍，不然你根本做不出。 实际上唯一的疑问就是一轮下来哪边就不用排了。 这个用排序数组试一遍就知道了。（往的地方）

剑指Offer（二十八）：数组中出现次数超过一半的数字

剑指Offer（三十）：连续子数组的最大和 很经典的动态规划，当然要掌握。

实践：OK，不熟练。

\*\*剑指Offer（三十二）：把数组排成最小的数 虽然用sort工具不是最佳，但确实很方便。

<https://blog.csdn.net/lx_nhs/article/details/78871295> comparator

好难。。

\*\*剑指Offer（三十五）：数组中的逆序对 经典的归并法，用暴力法的时间完全不够。应该是很重要的题目。嗯，而且很难看懂

好难。。

背：剑指Offer（三十七）：数字在排序数组中出现的次数 重要的，典型的二分法题目。重点：1 mid如何计算？ 2 二分法如何分别得到最前和最后的。？ 比leetcode的写得好

实践：会写二分法的其中一边即可。

剑指Offer（四十）：数组中只出现一次的数字 用的是散列表的方法。很好的想法。值得与前面的HashMap对照。 但是这道题目实质上想考察的是异或的性质。 是很经典的一种做法。 异或 ^= 相同取0，相异取1 & 逻辑与 0&1=0； 1&1=1；0&0=0

实践：1 如何找一个数第几位为0 ？

剑指Offer（五十）：数组中重复的数字 这题乖乖的， 关于boolean数组越界的问题。。这题的话，一定不会越界，因为给出的数字必定为0 –n-1。 所以用boolean数组就不会存在以往的那种不知道设多大的数字的问题。

剑指Offer（五十一）：构建乘积数组 虽然是单独完成的，但是可以浏览一遍自己的思路。

实际上有更高效的算法。 实践：1会画对角矩阵，2 知道比起暴力算法优势在利用前面已知的数而不是每个数字重新乘积，这两点后就能做出来。

从尾到头打印链表 旋转数组的最小数字 奇数到偶数前 出现超一半的数 连续子数组的最大和

20 -50 28 超过数组的一半 29最小的k个数，堆排序 32数组排成最小的数目

35 逆序对 41 和为S的连续正整数序列 43 54找出字符流只出现一次的

56 去除重复节点 （要考虑到头的情况）

对称的二叉树 实践：在做得时候存在问题，重做。一时之间会反应不过来。

滑动窗口：难但是未必容易考

**字符串(8道)**：

剑指Offer(二)：替换空格 考察StringBuffer的方法。重做。

StringBuffer 长度？ 添加元素？ 某位的字符？ 删除？ 有反转吗？

<http://www.runoob.com/java/java-stringbuffer.html>

反转方法 https://www.yiibai.com/java/stringbuffer\_reverse.html

\*\*剑指Offer（二十七）：字符串的排列 好难啊。S级难度。 看都看不懂解法

实践：全排列就是要死记

剑指Offer（三十四）：第一个只出现一次的字符 LinkedHashMap的使用

Character类 http://www.runoob.com/java/java-character.html

**实践**： 包含方法(有key，有value的) LinkedMap构造 str第n位的字符

Char的类类型？ 另外注意这种题目都是要遍历原数组根据key得到value的。

剑指Offer（四十三）：左旋转字符串 这里有关反转的string的方法我需要掌握。

<https://www.yiibai.com/java/stringbuffer_reverse.html>

**实践**：如何截取String？ 如何根据StringBuffer如何直接初始化成为字符串？ StringBuffer是如何合并在一起的？

剑指Offer（四十四）：翻转单词顺序序列 也是关于string的方法应用的

实践:信息量有点大，还是做一遍的好。 反转的话，一般是直接用string还是用StringBuffer?用StringBuffer要怎么添加字符串？

Equal() 和== 的区别 https://zhidao.baidu.com/question/176201312243104404.html

剑指Offer（四十九）：把字符串转换成整数 A~Z,a~z,0~9的ASCII码

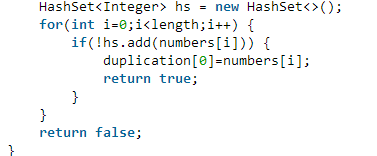
<https://zhidao.baidu.com/question/67956528.html>

**实践**：1 String似乎不能直接取得某一位的字符？ 可以的，转为char只是方便些。2 要写出简洁的代码，多用 **？：** 的符号。 3 char[] a=str.toCharArray() 不用新建字符数组，虽然这个过程实际上就是新建字符数组

\*\*剑指Offer（五十二）：正则表达式匹配 这题是难题中的战斗机，赌不会考把，我也不掌握。

--剑指Offer（五十三）：表示数值的字符串 这题实际上很没意义，就是考分类，分类。。。因为没有考察任何数据结构算法

总结： 重复数字有三种做法：1 HashMap LinkedHashMap 2 原来的数组排序，然后遍历，前后两个相等就是 3 异或的方法 4 HashSet法， 5 设置数组，boolean或者int， 但是这种原始的方法缺点在于空间过大。而且不知道应该设多少大小的数组。



**栈(3道)**：

剑指Offer（五）：用两个栈实现队列

剑指Offer（二十）：包含min函数的栈 迭代器法以及 辅助栈法都值得一看，我的比较赖皮。效率低下

**实践**：只要能想到辅助栈是用来做什么的，就能随手做出来。 记录最小值的变量，如果不能让它初始化为0（影响计数），该如何初始化最合适呢？ Min栈应当只存放最小值。

Pop()函数应当怎么写？

剑指Offer（二十一）：栈的压入、弹出序列

**实践**： 知道原理最重要。 注意要判空！！这个暂时想不明白

**递归(4道**)：

剑指Offer（七）：裴波那契数列

剑指Offer（八）：跳台阶

剑指Offer（九）：变态跳台阶

**实践**：很难想到的解法，记住即可。 不过能用数学归纳为什么不用呢？

剑指Offer（十）：矩形覆盖

**回溯法(2道)**：这特么不是我以前做得图里经常会有的一种题目么？

剑指Offer（六十五）：矩阵中的路径

剑指Offer（六十六）：机器人的运动范围 **实践**：会思想，大致就会了。

逻辑数组的初始值是false.

<https://blog.csdn.net/duyiwuerluozhixiang/article/details/85797745>

**其他(15道)**：

剑指Offer（十一）：二进制中1的个数 **实践**：手画一遍都明白了。

--剑指Offer（十二）：数值的整数次方 **实践**： 实践什么啊，这种题出出来我吃屎。而且不算难

背：剑指Offer（十九）：顺时针打印矩阵

剑指Offer（二十九）：最小的K个数

剑指Offer（三十一）：整数中1出现的次数（从1到n整数中1出现的次数）

我愿意给它S级的难度，但是答案貌似挺容易 。 答案的话要死记，还是很困难。

剑指Offer（三十三）：丑数 吐槽好难 ，看都看不懂，之后再看一遍。

剑指Offer（四十一）：和为S的连续正数序列 重要的还是思路，只要能演示一遍就能明白怎么做。 实践： 只要知道怎么做，就一定能做出来。 双指针法也不一定要从两边开始。。。。。！！！

剑指Offer（四十二）：和为S的两个数字 同样是重视思路的一题。

**实践**：只要有知道怎么做，就能够做出来。

剑指Offer（四十五）：扑克牌顺子 这题实际上读懂题目后就不太难。

**实践**：读懂题目后很好想

剑指Offer（四十六）：孩子们的游戏（圆圈中最后剩下的数） 注重原理，知道原理后就简单。 **实践**：

剑指Offer（四十七）：求1+2+3+…+n 做题有两种思考方向，其中一种是递归！！

剑指Offer（四十八）：不用加减乘除的加法 ？？？？

剑指Offer（五十四）：字符流中第一个不重复的字符 前面字符串题目的加强版

但是实质上思路还是最重要的，并不算太难。 **实践**：

剑指Offer（六十三）：数据流中的中位数 暴力法固然能够解决，但是有什么其他的方法，让新入的数字有序。 **实践**：这题既需要知道用什么容器，也要清楚该容器的使用方法。

CompareTo

https://blog.csdn.net/wxzjn1027/article/details/80678578

compare 反正记住一点，只要compare 方法返回值是1 ，那么就从小到大正序排序？

http://how2j.cn/k/collection/collection-comparator-comparable/693.html#nowhere

剑指Offer（六十四）：滑动窗口的最大值 我第一时间想到的是暴力法，但是最好的貌似是双端队列

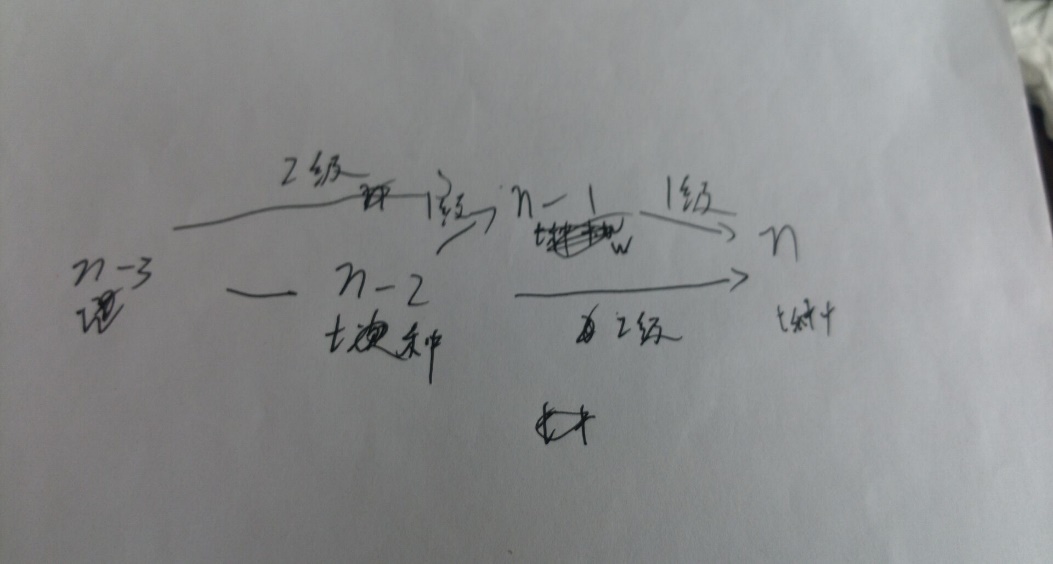
**跳台阶**

记录当时的弄清楚前后的关系。 n-1和 n的关系是什么？

到达n有两种方式 一种是 n-2直接两级 一种是n-1一级过来

所以 n的次数 为 n-2次数 和n-1次数之和吗？ n-2的那么多次 和n-1的那么多次，有重合吗

n-1是从n-2过来的。也可以n-3过来。



**变态跳台阶**

额难到我了，不过我找到了规律。

public class Solution {

public int JumpFloorII(int target) {

return (int) Math.pow(2,target-1);

}

}

正规解法我放到牛客上了。

public class Solution {

    public int JumpFloorII(int target) {

        return 1<<--target;

    }

}位移法不是特别看得懂

**数值的整数次方**

有以下几个陷阱：1、正负

2、超过double的最大值，不过要完成这个陷阱就要知道double的极限。

**调整数组顺序使得奇数位于偶数前面**。

排是排出来了，但是不知道怎么输出整数数组。

这一题的题目设计就是不用输出的，只要把array原数组改变了就行了。系统会自动检查你的array的。

**输入一个链表，输出该链表中倒数第k个结点。**

**Java的链表有没有空表头的啊？**

**树的子结构**

这里还是很难的。

**从上往下打印二叉树**

如何用java专属的队列来完成遍历。

**合并两个排序的链表**

**应该是难题。 哦不对，你想不出来就再想想，实际上不难的。**

**第二页**

**数组中出现次数超过一半的数字。**

**发现不太擅长使用数组的方法以及 不太擅长定义数据结构。 像c的vector<node\*>不知道怎么弄个类似的。**

**二叉搜索树的后序遍历**

**根本不会做，初看。这里涉及的主要是后序遍历的性质这一思想。利用这一点来做题。**

**最小的k个数**

**学会使用工具类，并且要注意所谓的非法数组的问题!! 这题我们还是用了取巧的方法。可以看看其他方法是怎么做得。比如基本的排序我们还是要掌握的。**

**字符串的排序，归为难题**

**基本确定是用递归法做得，可是str字符串数组的用法还不熟悉，思想回溯法，我也遇到过类似的，但是我忘记当时作为全局变量的数组path【】 是怎么让自己回到分岔口的状态的了。**

**把数组排成最小的数**

**还是不会。 利用利用的是collections的工具类，同时用上了compare来规定sort的用法。**

**Java compareTo() 方法http://www.runoob.com/java/number-compareto.html**

**整数中1出现的次数**

**奥数题，智力题，去你妈的。 这个实际上是利用字符类型的函数来求得答案的，在不熟悉这些常用方法的前提下，我根本不可能能够利用这种方法来完成题目。**

**平衡二叉树**

**应该是很经典的一道递归的题目。 非递归的做法爱看不看。我没做出来，太久没接触了。这里面有逻辑上很奇怪的点，再做一遍把。**

**数组中只出现一次的数字**

**Int【】 数组创建出来，在JAVA中初始就是0。然后我们的方法估计不太正规，太耗费空间了。正规方法看一下。**

**\*丑数**

**这种题目从1开始往上，找到第n个 具有某种性质的数，这种题目考验的本质是如何防止超时。不过看答案的话，算是难题。**

**最大子序列**

**关于动态规划，最好把简单的几种全部在算法笔记中弄懂。**

**第三页**

**二叉树的下一个结点**

**这题是道难题。或者是一种固定题型。直接硬考想，是很难得，但是可以背下来这种做法。**

**对称的二叉树**

**我想在这道题寻找树的固有解法。所以偷个懒不想了。**

**Java刷题注意的要点：**

1. **注意想到所有的边界以及方法。我已经不止一次没有考虑输入数组的非法性而爆炸了。**

**疑问：**

1. **在java中如何 像C中，利用二维数组来储存树或者。。的某项属性呢？**

**在间隔时间里，思考算法是绝佳的利用时间的方式，因为容易想，而且也容易模拟面试的环境。并且算法比我们想象中的要更有趣一些。**

**在面试中问得算法，一般要么是排序，要么是涉及到数据结构的。而且还是很容易讲清楚的数据结构。 一两句话能够说清楚的，比如反转链表等等。我们就专门挑这些数据结构搞。**

**做题技巧：1思考的话有两个方向，一个是递归，另一个是非递归。很多时候你根本没想着用递归来完成。有着多个重复过程的，用递归的概率大大增加。比如说 链表这种。**

**2 实际上一个想法很难从头到尾都想清楚，都是先去写，满足一个要求，然后再次基础上删删改改就可以完成了。这个就是做题技巧，人的脑容量有限。**

**3 不要忘记了边界。**

**第一页 从尾巴到头返回一个ArrayList（自己的思路也不错） 重建二叉树（这个有可能）**

**反转链表（死记）**

**合并两个排序的链表（这个偏南一些，死记递归，理解非递归） 实践：只要知道原理就能写**

**树的子结构（所以说递归这种，还是要想清楚这个递归方法是具体做什么的，这个清晰了才能做下去） 在这道题中减枝是什么样子的，想象一下？ 实践：前一部分写的不够严谨。可以再做改进。 为什么一定要分成两个递归? 两个递归的含义根本不一样。第一个是看是否是子结构。左边不行右边再看。并且存在减枝。 第二个，默认第一个相等。中途有一个不对应就GG，并且最后return && 也是这个道理。**

**二叉树的镜像**

**非数据结构：两个栈实现一个队列 实践:Stack的判空？**

**第二页：从上往下打印二叉树的每个结点 Queue的创建， Queue的添加 ，出队，判空**

**二叉搜索树的后序遍历序列 实践：**

**二叉树中和为某一值的路径 如何把路径list加入到大List中？ 如何解决用于临时记录的list数组在递归中的处理？**

**实践：注意，不能把集合加入大集合中，必须要重新new一个才行**

**二叉搜索树与双向链表（） 实践：知道思想应该可以做出。不过再此基础上最好熟记大致解法。**

**两个链表第一个公共结点 这个有更高效的算法的。**

**import java.lang.Math;**

**二叉树的深度（经典）**

**平衡二叉树（这题目原本的方法有误） 实践：能不能背下来，真的是很好的一段代码，堪称艺术。 此外还有一种后序遍历的方法也很好。**

**第三页： 链表中环的入口结点 双指针法更高效！！ 删除链表中重复的结点**

**二叉树的下一个结点 实践：有思路就会做。思路看一下。 对称的二叉树 常见的递归。 实践：有思想即可。 按之字形顺序打印二叉树**

**把二叉树打印成多行**

**第四页： \*序列化二叉树 二叉搜索树的第k个结点 ！！**

**太难不太可能考的： 复杂链表的复制**

**全排列**

**链表头插法 尾插法**

**Stack常用方法**

[**https://www.cnblogs.com/guweiwei/p/6516601.html**](https://www.cnblogs.com/guweiwei/p/6516601.html)

**st.isEmpty()**

**741**

**852**

**963**

**123 147**

**456 258**

**789 369**

**For(int i=0;i<length;i++){**

**For(int j=I+1;j<length;j++**

**Swap(a,i,j)**

**}**

**Array**

**1 这题减少时间复杂度的方式很值得参考 实践：看看则好，看样子某一种写的方式是很不鼓励在答案中出现的。**

2