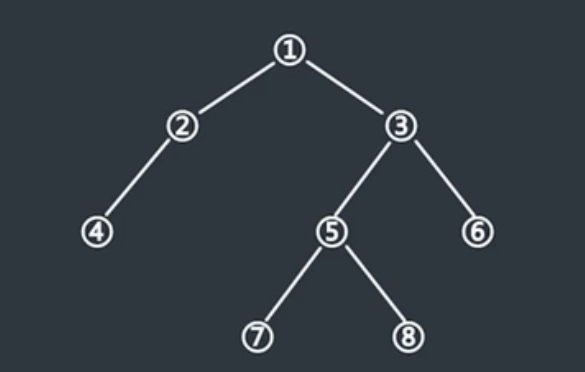
**Test1.java**

给定一颗二叉树的头节点 **head**，请按照看到的这种格式打印出来。



* 实现递归遍历
* 实现非递归遍历
* 实现层次遍历

层次遍历要求打印成：

***1***

***2 3***

***4 5 6***

***7 8***

**Test2.java**

**二叉树的序列化与反序列化**，把二叉树记录成文件的过程称为二叉树的序列化（持久化）过程。

* 序列化：二叉树==》字符串
* 反序列化：字符串==》二叉树

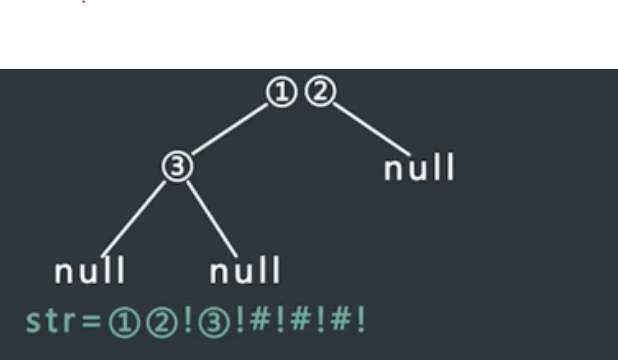
**序列化的方式**

* 先序序列化
* 中序序列化
* 后序序列化
* 层次序列化

**实例**：二叉树被记录成文件的过程被称为序列化，通过文件内容重建二叉树的过程称为二叉树的反序列化。给定一棵二叉树头节点为**head**，已知二叉树的节点值类型为32位整数类型。请设计一种二叉树序列化与反序列化方案，并用代码实现。

**分析**：**1、**先序遍历对二叉树进行遍历

* 假设序列化结果为str，初始时str为空字符串。
* 先序遍历二叉树是，如果遇到空节点，记为“#!”
* 如果遇到不为空的节点，假设节点值为3，记作“3!”
* 选择用什么样的方式进行序列化，就再用什么方式反序列化。
* 一棵树序列化的结果是唯一的，唯一的结果生成的二叉树结果也是唯一的。



2、按层遍历的方式进行序列化

* 用队列来进行二叉树的按层遍历，即宽度优先遍历
* 除了访问节点的顺序是按层之外，对结果字符串的处理与之前介绍的处理方式一样。
* 反序列化过程同理。

**Test3.java**

**时间复杂度O( n^2 )**

* 冒泡排序 bubbleSort
* 选择排序 SelectSort

是一种不稳定的排序算法。

* 插入排序

**时间复杂度O( N\*log(N) )**

* **归并排序**

将若干个有序序列逐步归并，最终归并为一个有序序列。

* **快速排序**

小的在做大的再有。

* **堆排序**

**堆排序**是简单选择排序的一种改进，改进夫人着眼点是：如何减少关键码的比较次数。简单选择排序在在一趟排序中今选出最小的关键码，没有把一趟的比较结果保存下来，因而记录的比较次数较多。堆排序在选出关键码的同时也找出了较小关键码，减少了在后面选择中的比较次数，从而提高了整个排序的效率。

**堆的定义**：具有下列性质的完全二叉树：每个节点的值都小于或者等于其左右孩子节点的值（称为小根堆）；或者每个节点的值都大于或等于其左右孩子节点的值（称为大根堆）。

虽然堆的典型实现方法是数组，但是从逻辑角度看堆实际上是一种树结构。

* **希尔排序**

是对直接插入排序的一种改进，改进的着眼点是：①若待排序记录按关键码基本有序，直接插入排序的效率很高。②由于直接插入排序算法简单，则在待排序记录个数较少的时效率也很高。

选择排序、快速排序、希尔排序和堆排序不是稳定的排序算法；

冒泡排序、插入排序、归并排序和基数排序是稳定的排序算法。