**基数排序**

Step1：预先设置10个桶，对应0-9；

Step2：将待排序数组元素根据个位数字依次放入对应的桶中，将桶中元素按顺序放回原数组;

Step3：将待排序数组元素根据十位数字依次放入对应的桶中，将桶中元素按顺序放回原数组;

……

Step4：将待排序数组元素根据最高位数字依次放入对应的桶中，将桶中元素按顺序放回原数组，排序完毕;

**归并排序**

Step1：向左递归分解数组；

Step2：向右递归分解数组；

Step3：合并(先将左右两边数据按规则写入数组，如果一边的数据有剩余，再将这些数据依次写入数组)；

**快速排序**

Step1：以最左边元素为基准值；

Step2：用两个指针从右往左找到比基准值小的元素，从左往右找到比基准值大的元素，将二者互换，直到两个指针指向同一元素后，将该元素与基准值元素互换位置；

Step3：此时，基准值左边元素都小于基准值，右边元素都大于基准值，对基准值左边继续排序，对基准值右边继续排序；

**中序线索化二叉树**

Step1：线索化左子树

Step2：线索化当前节点

1. 若当前节点的左指针为空，则指向前驱节点
2. 若前驱节点不为空，且前驱节点的右指针为空，则指向当前节点
3. 更新前驱节点，将其指针指向当前节点

Step3：线索化右子树

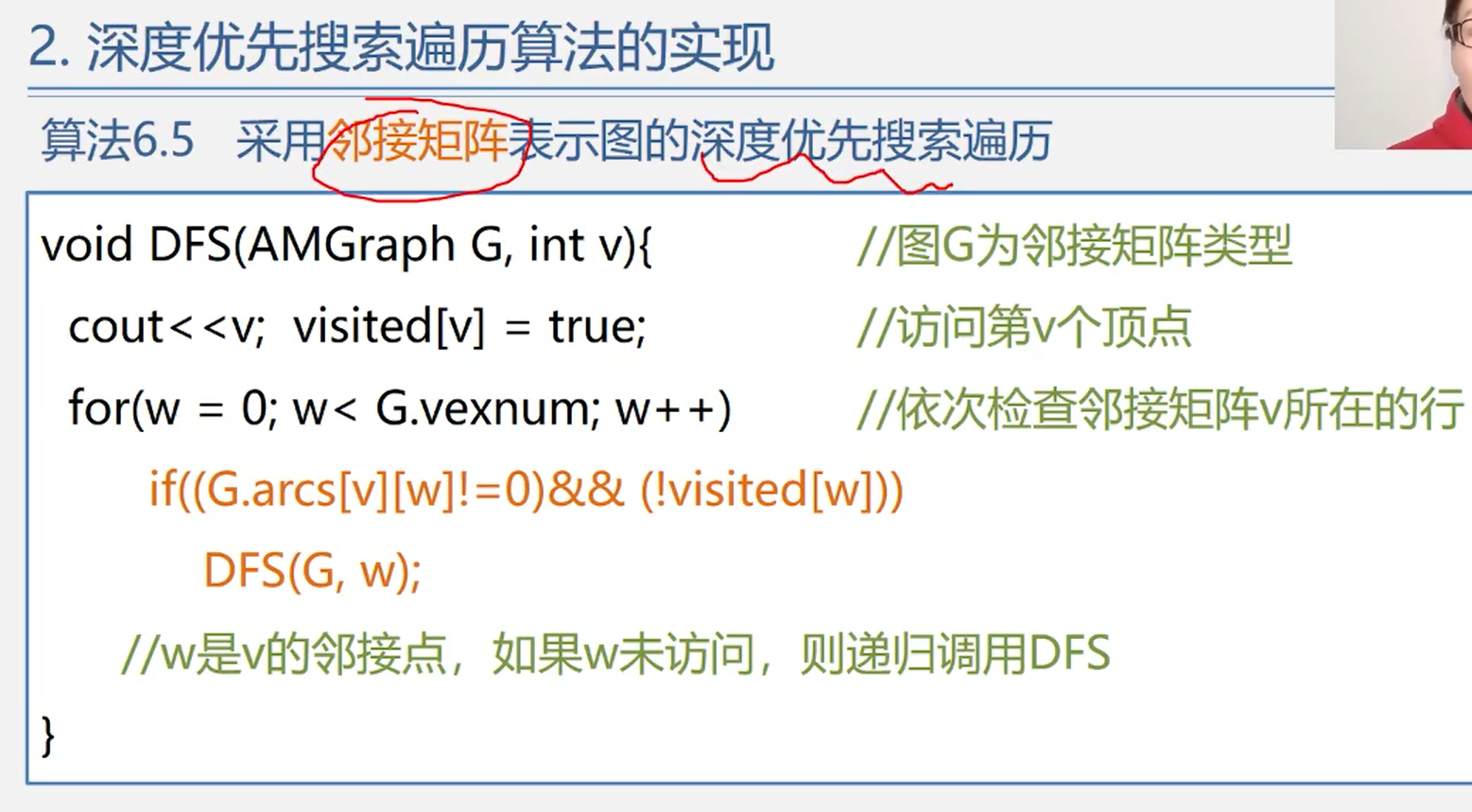
**堆排序**

Step1：从下往上，从最后一个非叶子节点开始调整堆的顺序，如果左右子节点最大的元素比父节点大，则将其位置互换；调整结束后变成大顶堆，此时根节点元素最大；

Step2：将根节点元素和最末元素位置互换，互换后从上往下，从左至右调整堆的位置，直到调整为大顶堆，将根节点元素与倒数第二个元素位置互换；

Step3：重复第二步，一共需要调整array.length-1次。

图的深度优先遍历



**单调栈**

1.单调栈是用来解决这一类问题：寻找数组中元素其左或者右第一个大于或小于该元素的值。

2.首先，如果要寻找其右侧，要从右向左遍历。反之，从左向右遍历。

3.如果是寻找第一个大于该元素的值，则要在栈顶保存较小的值。即从栈底到栈顶是递减的（栈顶到栈底就是递增的）。如果寻找第一个小于该元素的值，则要在栈顶保存较大的值。从栈底到栈顶是递增的。

**为什么重写equals方法，还必须要重写hashcode方法？**

1.使用hashcode方法提前校验，可以避免每一次比对都调用equals方法，提高效率

2.保证是同一个对象，如果重写了equals方法，而没有重写hashcode方法，会出现equals相等的对象，hashcode不相等的情况，重写hashcode方法就是为了避免这种情况的出现。