Python树的遍历和堆排序

讲师：Wayne

从业十余载，漫漫求知路

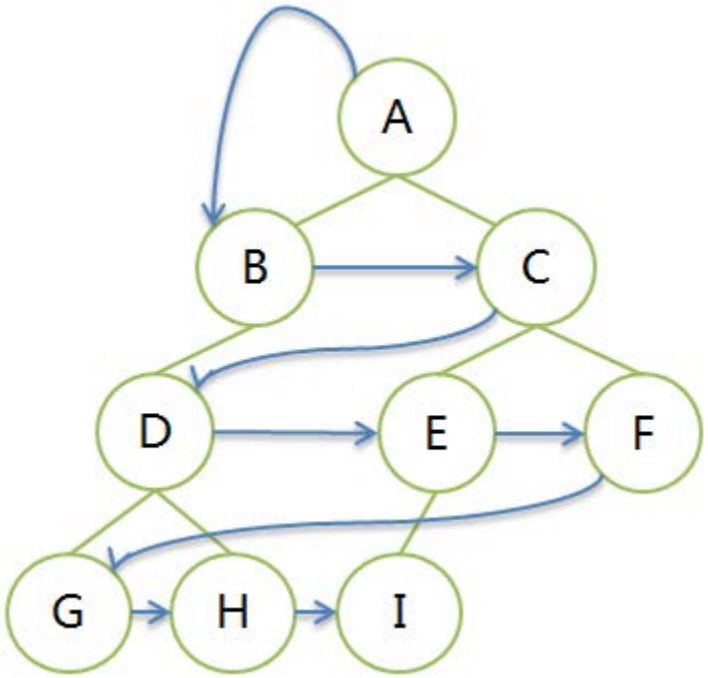
二叉树的遍历

- 遍历：迭代所有元素一遍
- 树的遍历：对树中所有元素不重复地访问一遍，也称作扫描

- 广度优先遍历
  - 层序遍历
- 深度优先遍历
  - 前序遍历
  - 中序遍历
  - 后序遍历
- 遍历序列：将树中所有元素遍历一遍后，得到的元素的序列。将层次结构转换成了线性结构

二叉树的遍历

- 层序遍历
  - 按照树的层次，从第一层开始，自左向右遍历元素
  - 遍历序列
    - ABCDEFGHI

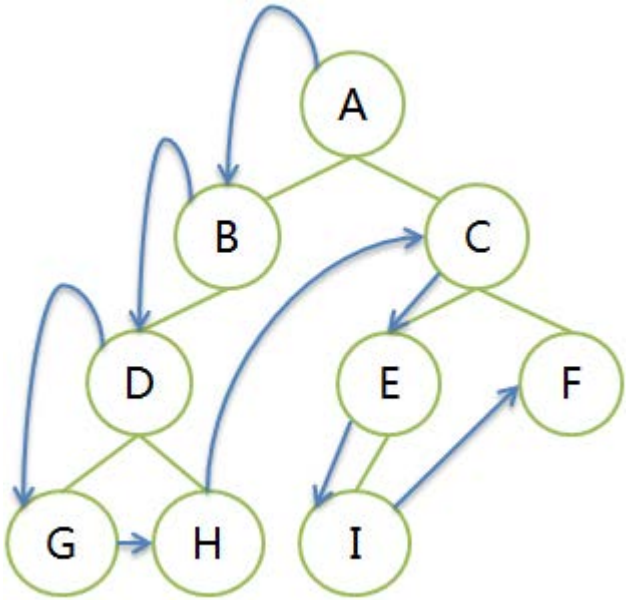


二叉树的遍历

- 深度优先遍历
  - 设树的根结点为D，左子树为L，右子树为R，且要求L一定在R之前，则有下面几种遍历方式：
  - 前序遍历，也叫先序遍历、也叫先根遍历，DLR
  - 中序遍历，也叫中根遍历，LDR
  - 后序遍历，也叫后根遍历，LRD

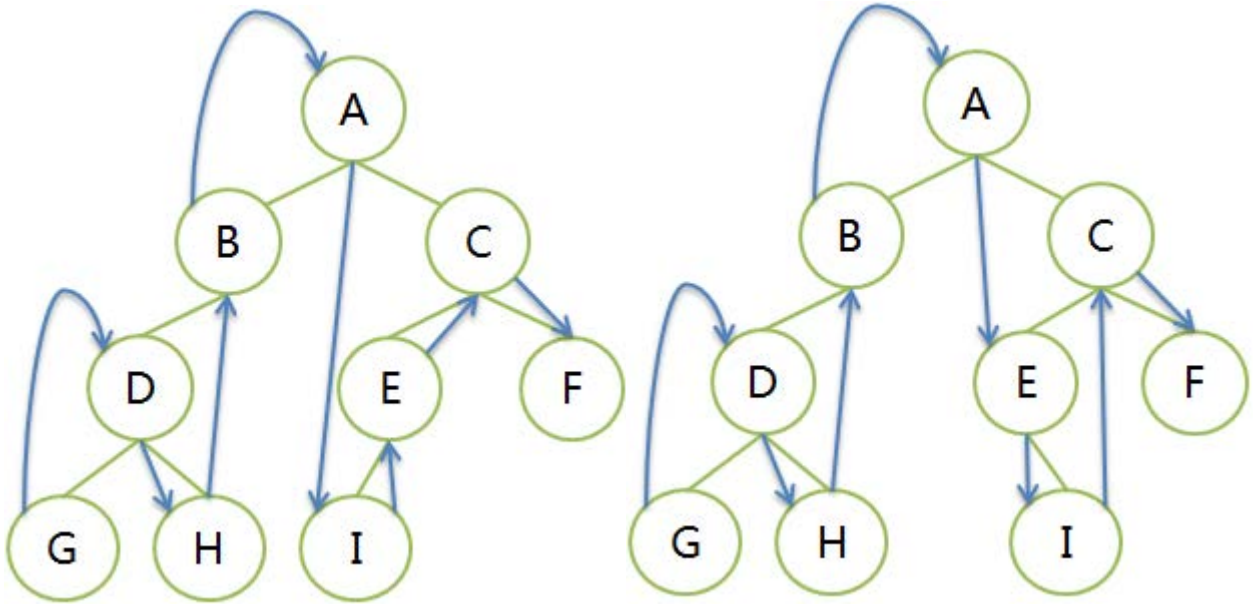
二叉树的遍历

- 前序遍历DLR
  - 从根结点开始，先左子树后右子树
  - 每个子树内部依然是先根结点，再左子树后右子树。递归遍历
  - 遍历序列
    - A BDGH CEIF



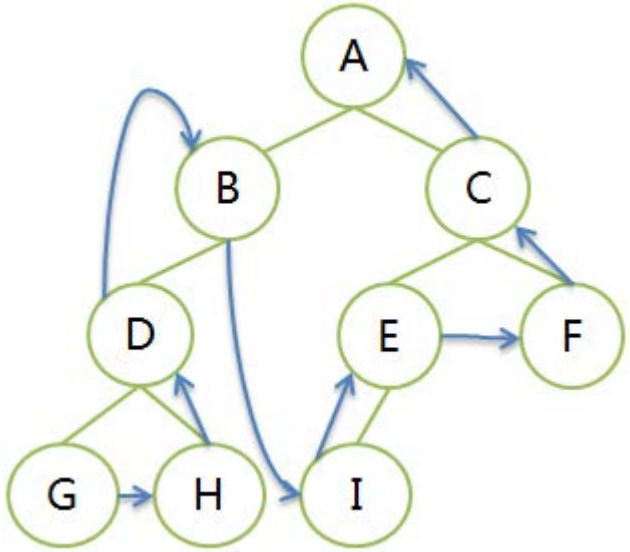
二叉树的遍历

- 中序遍历LDR
  - 从根结点的左子树开始遍历，然后是根结点，再右子树
  - 每个子树内部，也是先左子树，后根结点，再右子树。递归遍历
  - 遍历序列
    - 左图
      - GDHB A IECF
    - 右图
      - GDHB A EICF



二叉树的遍历

- 后序遍历LRD
  - 先左子树，后右子树，再根结点
  - 每个子树内部依然是先左子树，后右子树，再根结点。递归遍历
  - 遍历序列
    - GHDB IEFC A

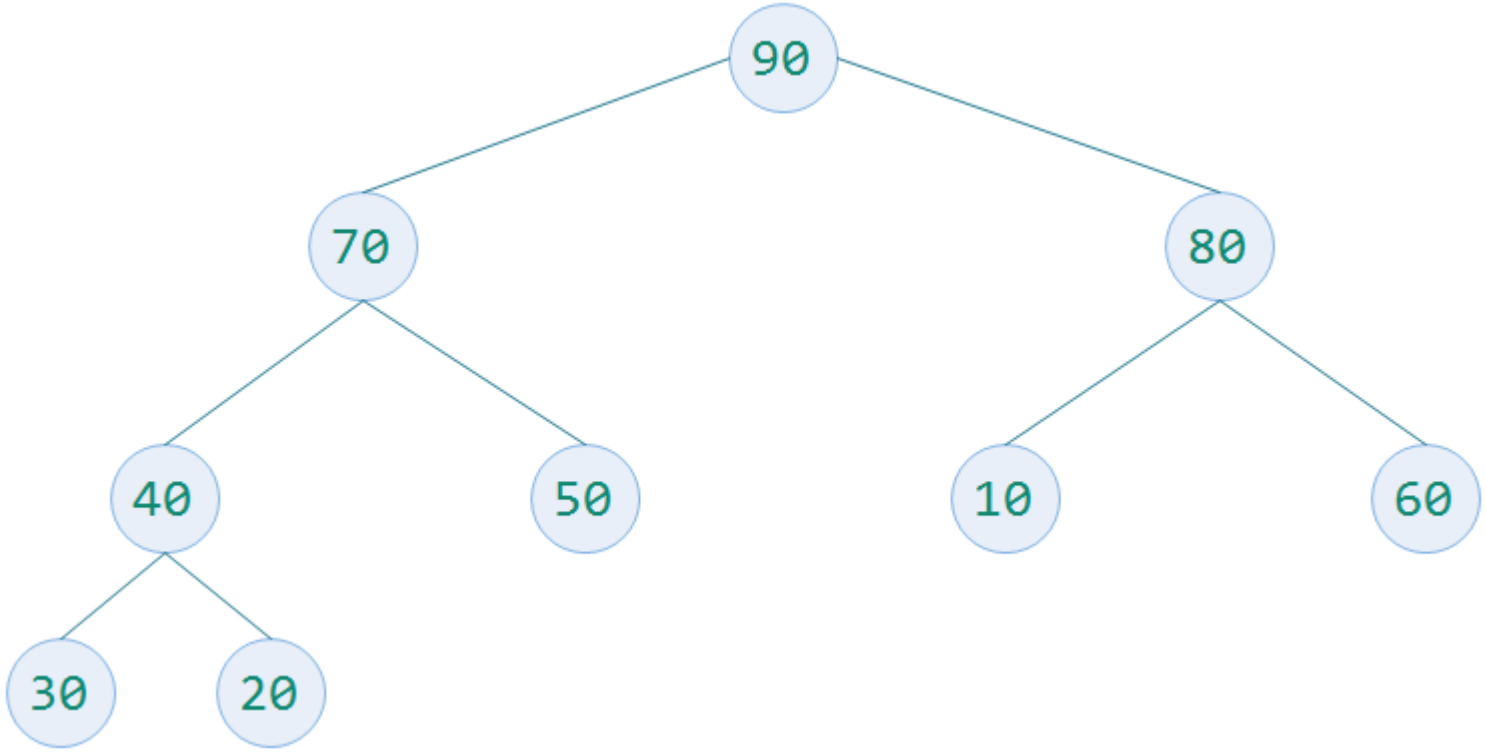


堆排序Heap Sort

- 堆Heap
  - 堆是一个完全二叉树
  - 每个非叶子结点都要大于或者等于其左右孩子结点的值称为大顶堆
  - 每个非叶子结点都要小于或者等于其左右孩子结点的值称为小顶堆
  - 根结点一定是大顶堆中的最大值，一定是小顶堆中的最小值

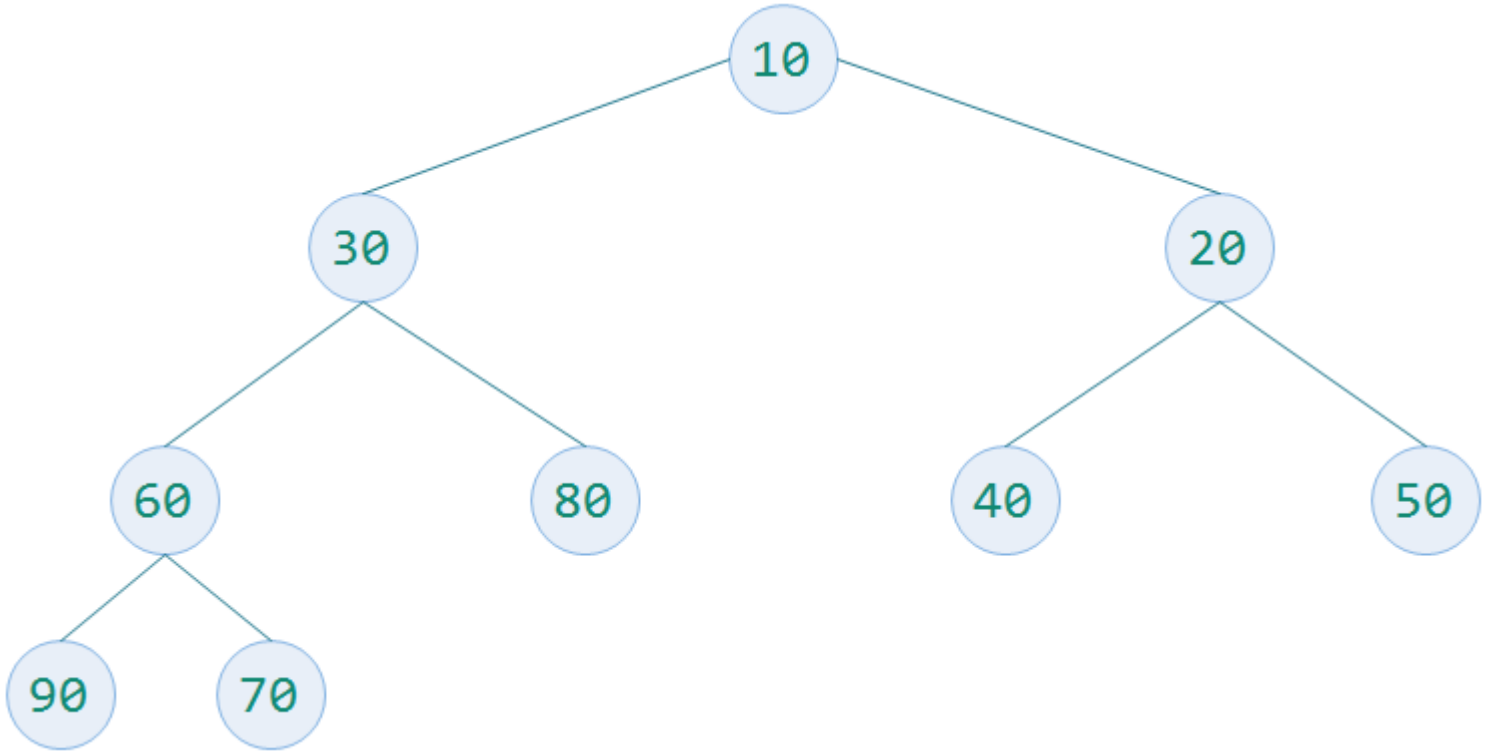
堆排序Heap Sort

- 大顶堆
  - 完全二叉树的每个非叶子结点都要大于或者等于其左右孩子结点的值称为大顶堆
  - 根结点一定是大顶堆中的最大值



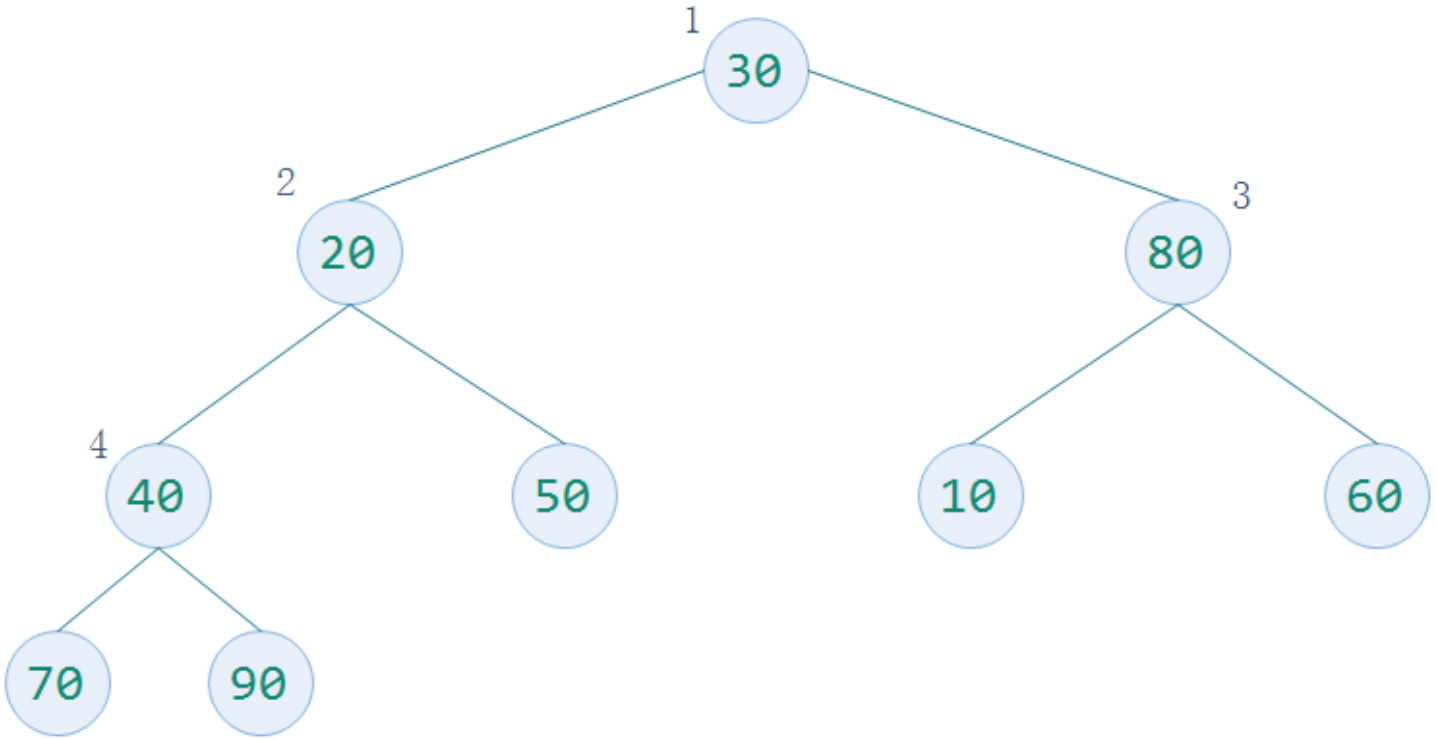
堆排序Heap Sort

- 小顶堆
  - 完全二叉树的每个非叶子结点都要小于或者等于其左右孩子结点的值称为小顶堆
  - 根结点一定是小顶堆中的最小值



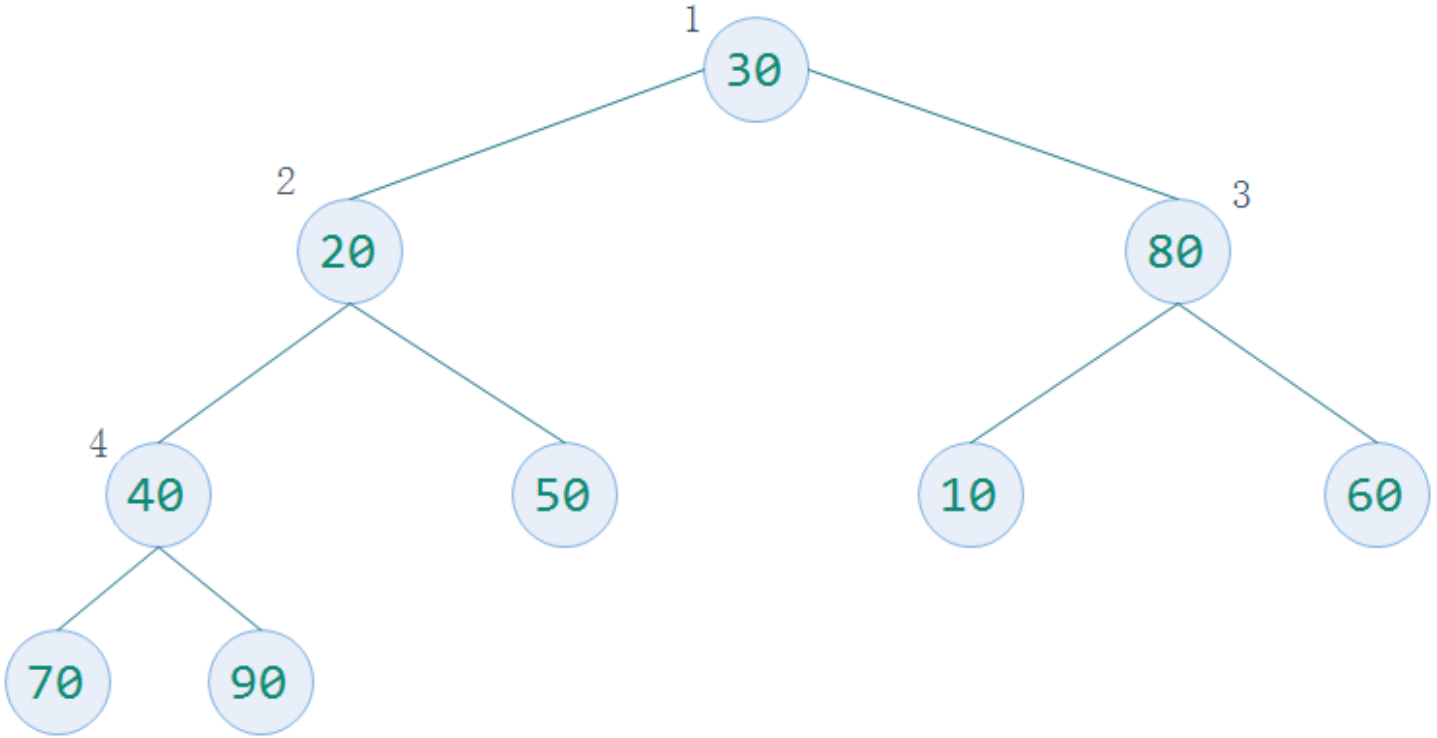
堆排序Heap Sort

- 1、构建完全二叉树
  - 待排序数字为 30,20,80,40,50,10,60,70,90
  - 构建一个完全二叉树存放数据，并根据性质5对元素编号，放入顺序的数据结构中
  - 构造一个列表为[0,30,20,80,40,50,10,60,70,90]



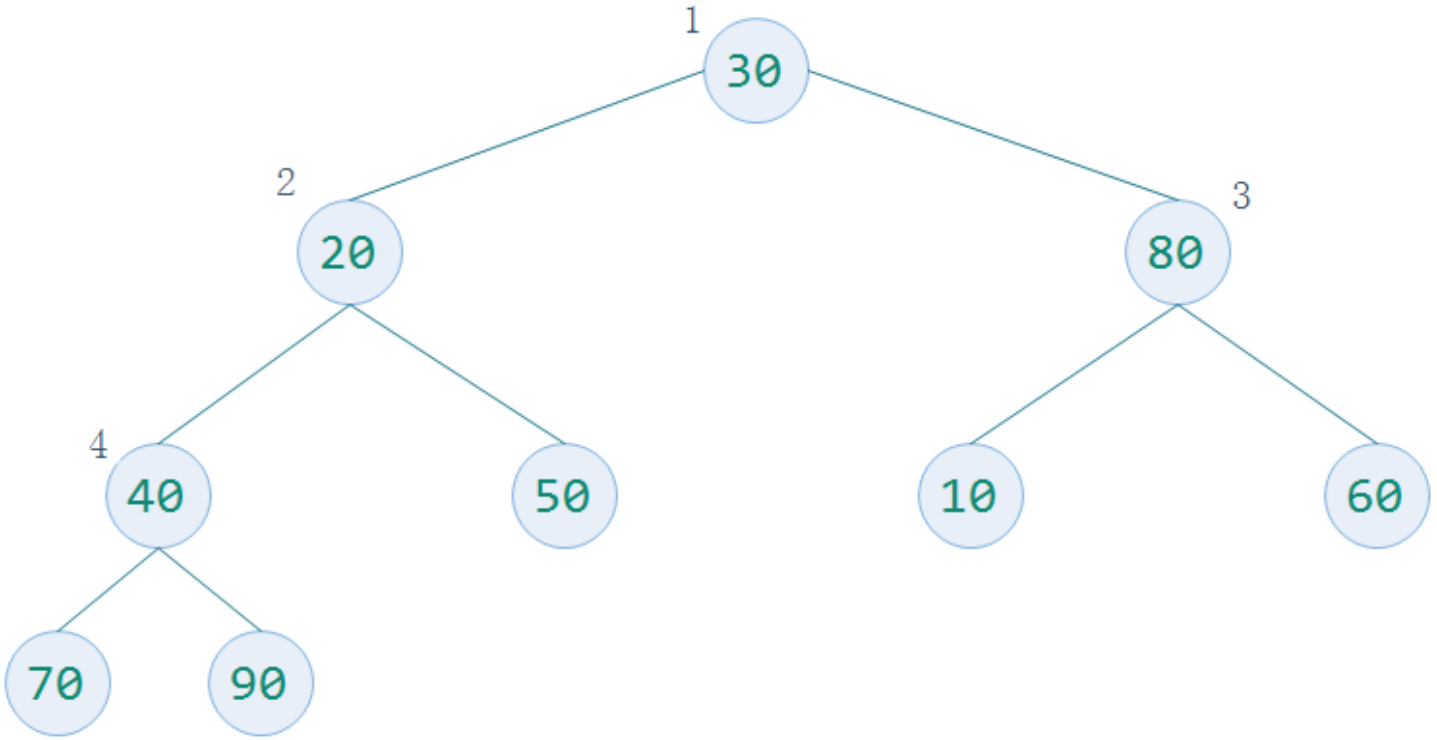
堆排序Heap Sort

- 2、构建大顶堆——核心算法
  - 度数为2的结点A，如果它的左右孩子结点的最大值比它大的，将这个最大值和该结点**交换**
  - 度数为1的结点A，如果它的左孩子的值大于它，则**交换**
  - 如果结点A被交换到新的位置，还需要和其孩子结点重复上面的过程



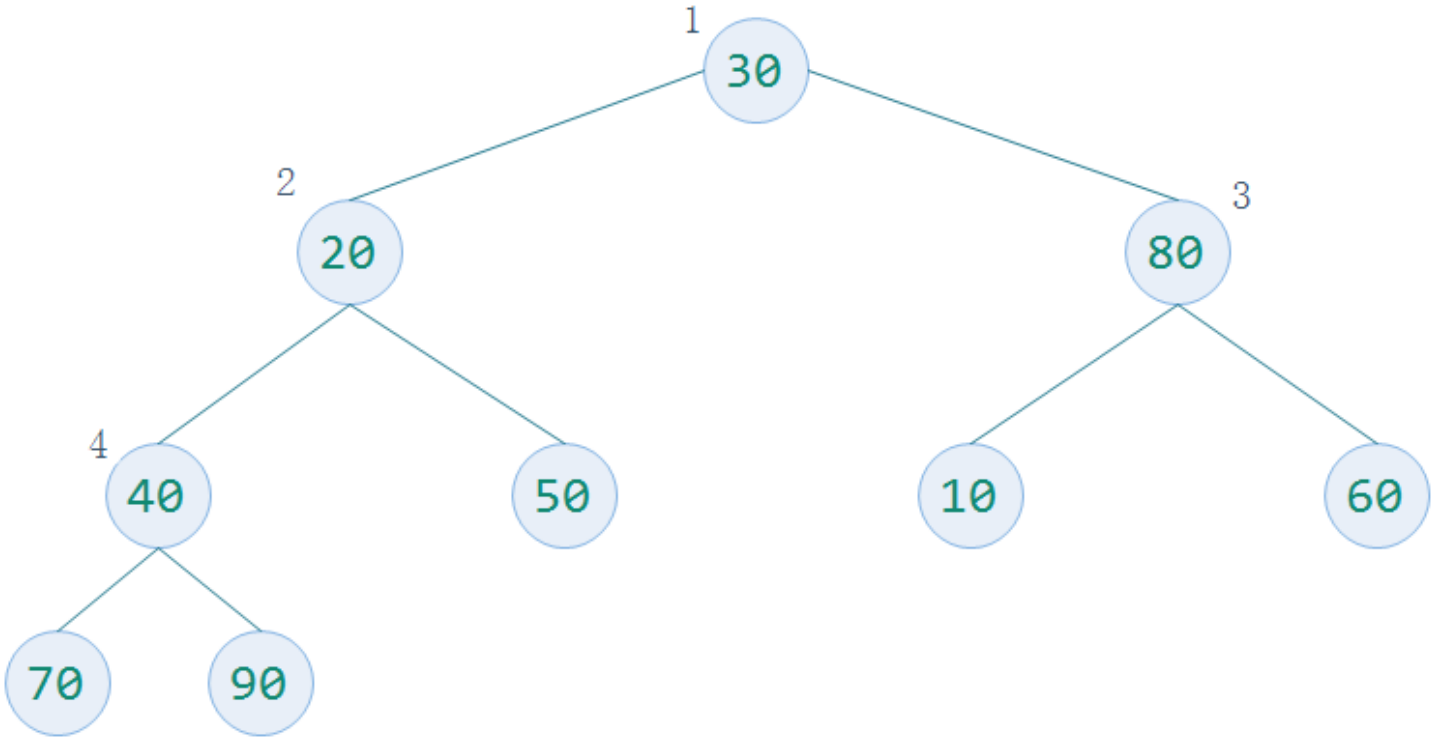
堆排序Heap Sort

- 2、构建大顶堆——起点结点的选择
  - 从完全二叉树的最后一个结点的双亲结点开始，即最后一层的最右边叶子结点的父结点开始
  - 结点数为n，则起始结点的编号为n//2（性质5）



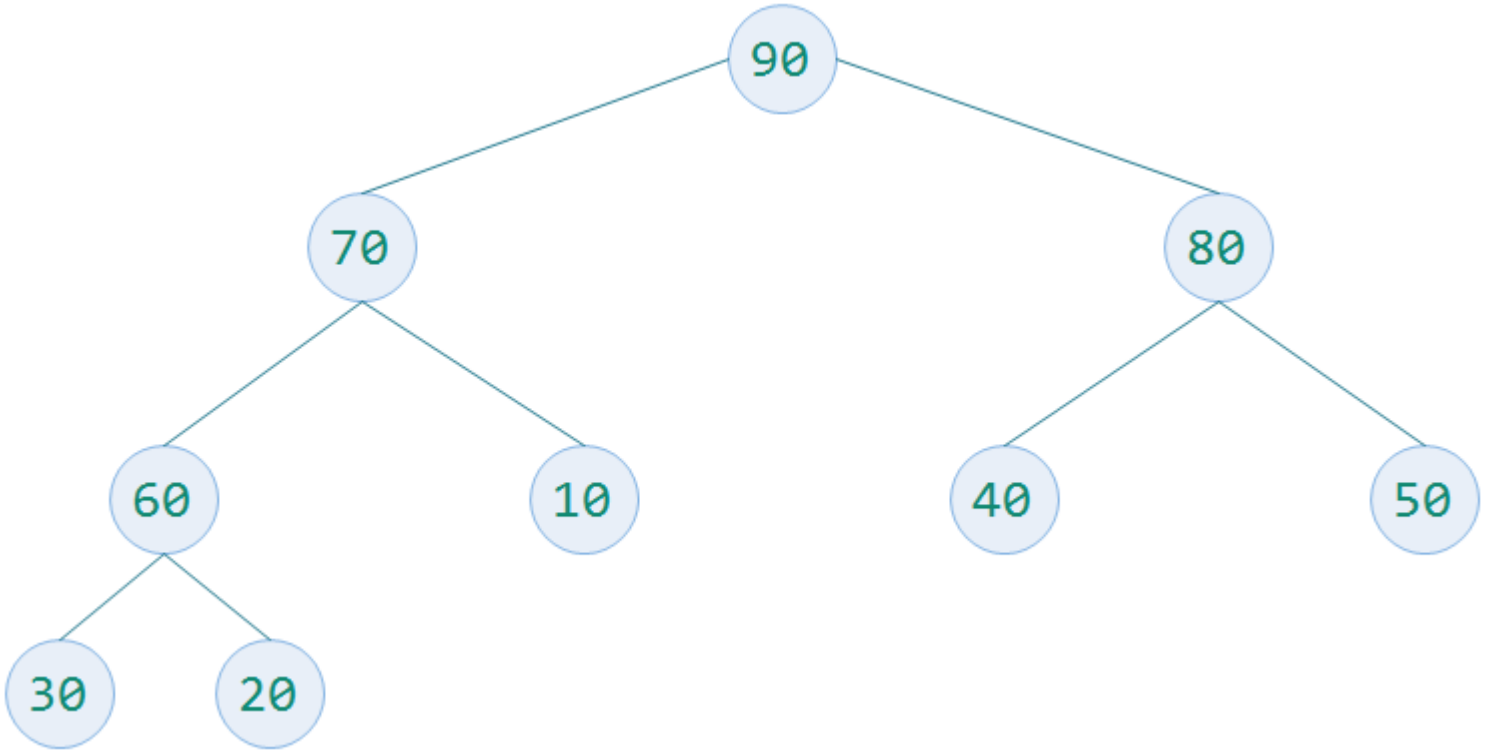
堆排序Heap Sort

- 2、构建大顶堆——下一个结点的选择
  - 从起始结点开始向左找其同层结点，到头后再从上一层的最右边结点开始继续向左逐个查找，直至根结点



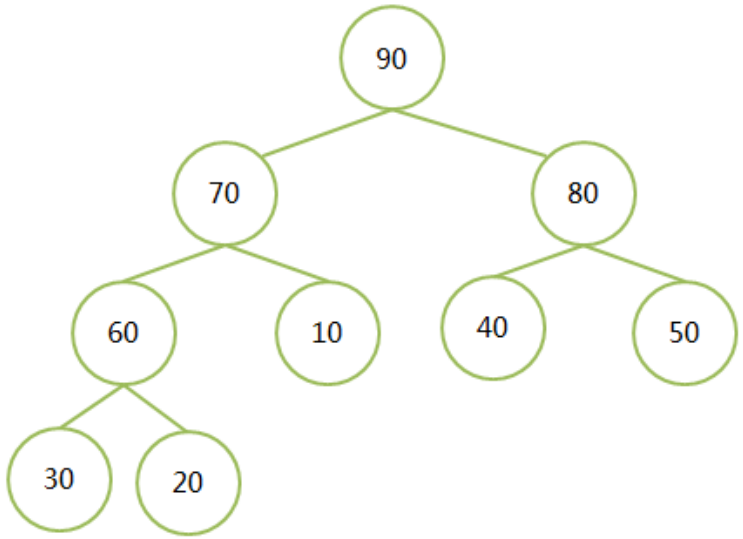
堆排序Heap Sort

- 2、大顶堆的目标
  - 确保每个结点的都比左右结点的值大



堆排序Heap Sort

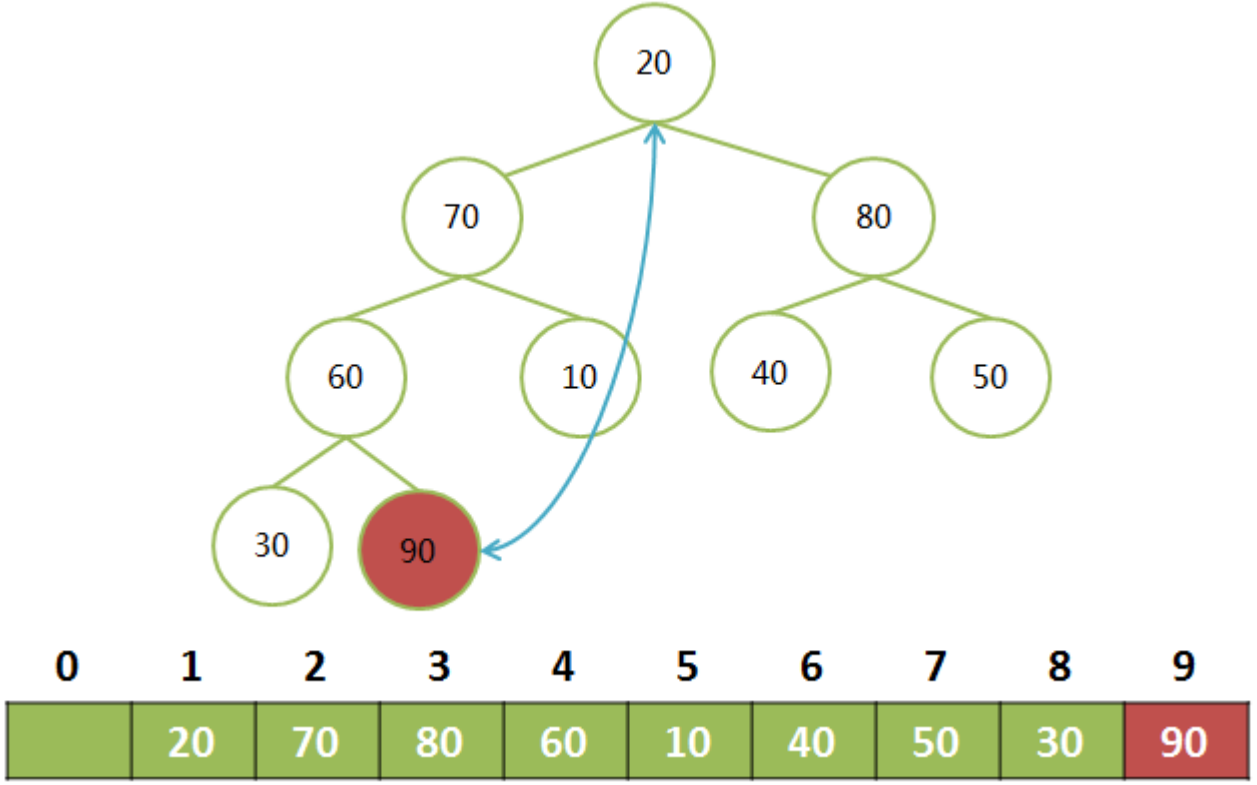
- 3、排序
  - 将大顶堆根结点这个最大值和最后一个叶子结点交换，那么最后一个叶子结点就是最大值，将这个叶子结点排除在待排序结点之外
  - 从根结点开始（新的根结点），重新调整为大顶堆后，重复上一步



0	1	2	3	4	5	6	7	8	9
	90	70	80	60	10	40	50	30	20

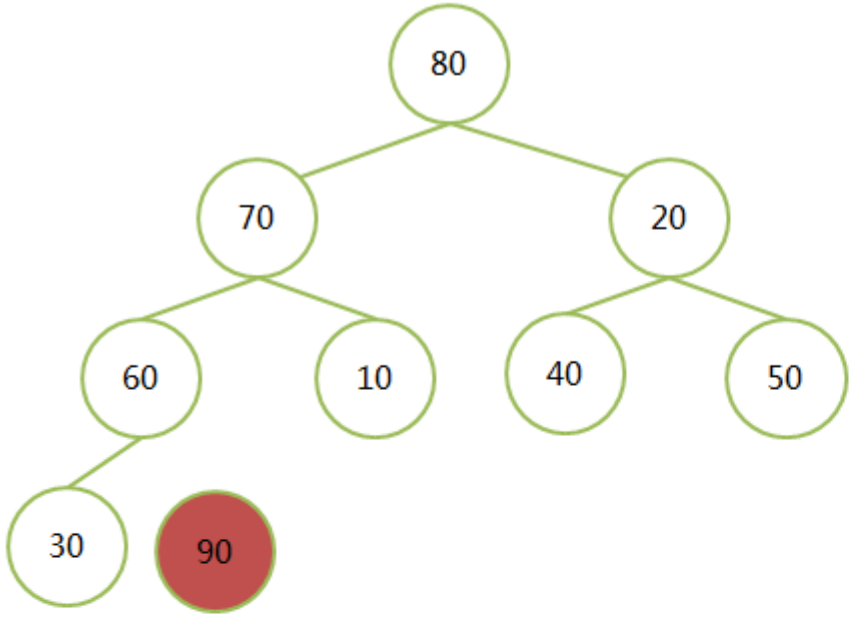
堆排序Heap Sort

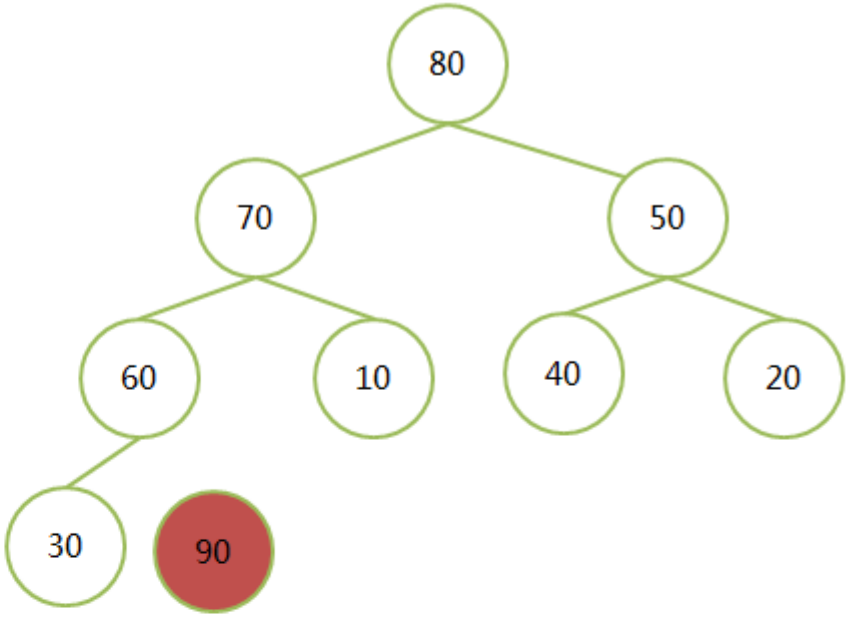
- 3、排序
  - 堆顶和最后一个结点交换，并排除最后一个结点

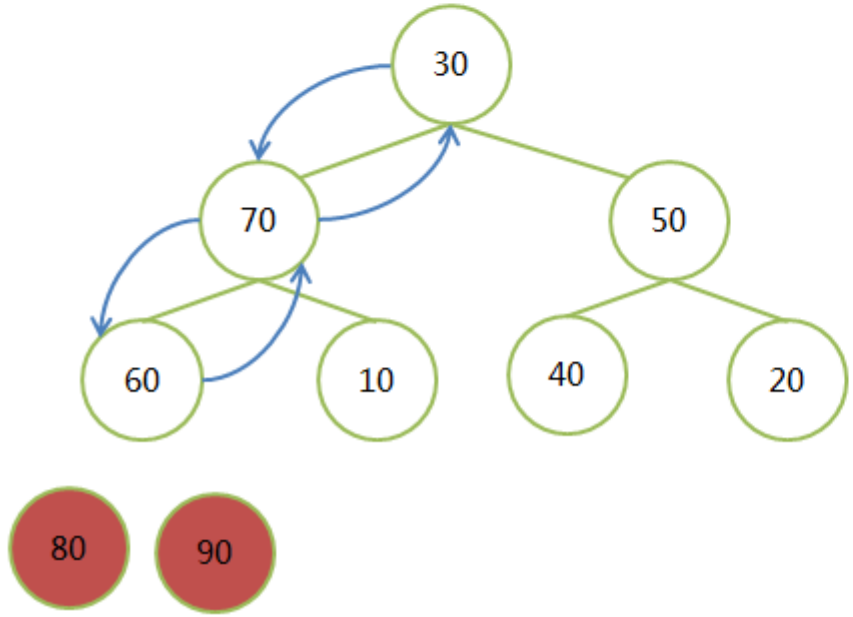


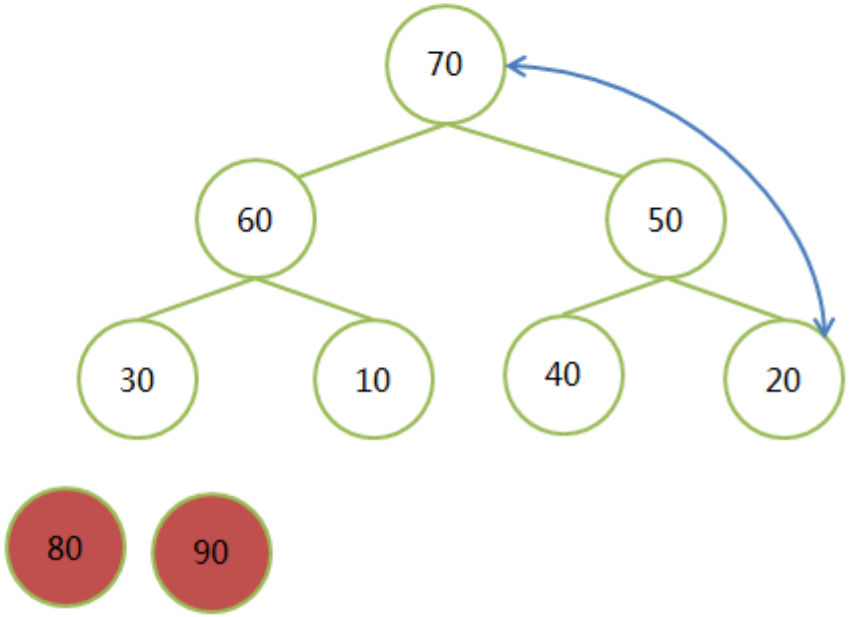
堆排序

- 3、排序


80
70
20
60
10
40
50
30
90


80
70
50
60
10
40
20
30
90


30
70
50
60
10
40
20
80
90

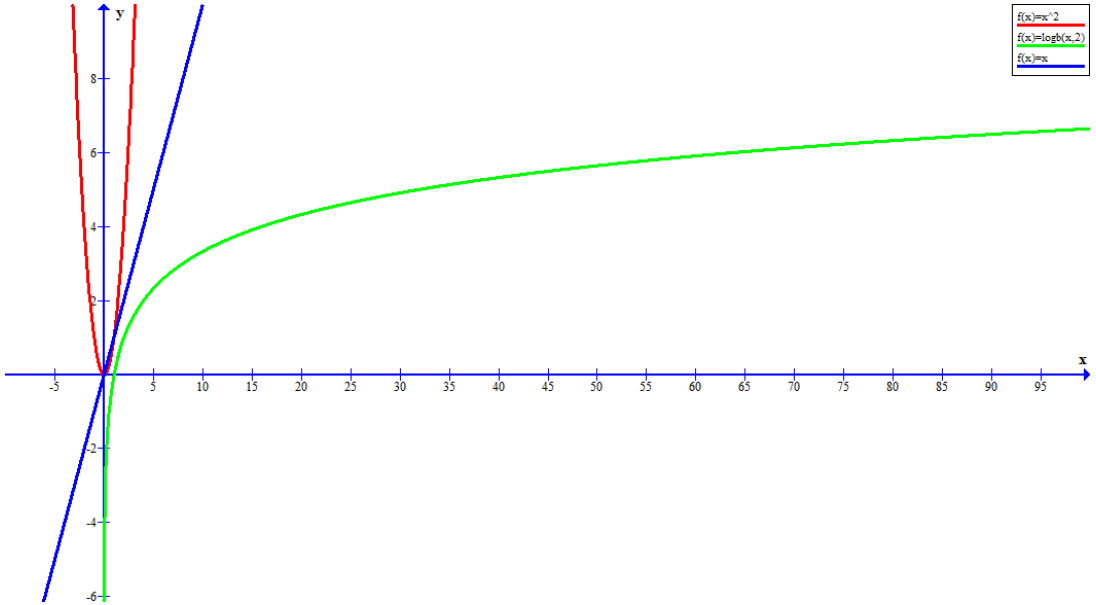

70
60
50
30
10
40
20
80
90

堆排序Heap Sort

- 算法实现
  - 见代码

堆排序Heap Sort

- 总结
  - 是利用堆性质的一种选择排序，在堆顶选出最大值或者最小值
  - 时间复杂度
    - 堆排序的时间复杂度为O(nlogn)
    - 由于堆排序对原始记录的排序状态并不敏感，因此它无论是最好、最坏和平均时间复杂度均为O(nlogn)



堆排序Heap Sort

- 总结
  - 空间复杂度
    - 只是使用了一个交换用的空间，空间复杂度就是O(1)
  - 稳定性
    - 不稳定的排序算法

谢谢

咨询热线 400-080-6560