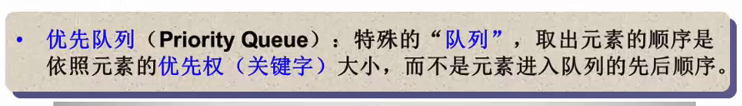
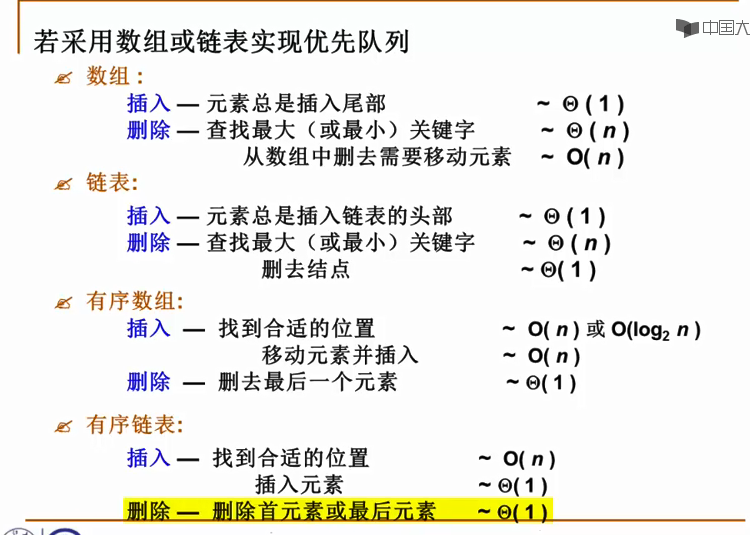
# 什么是堆：

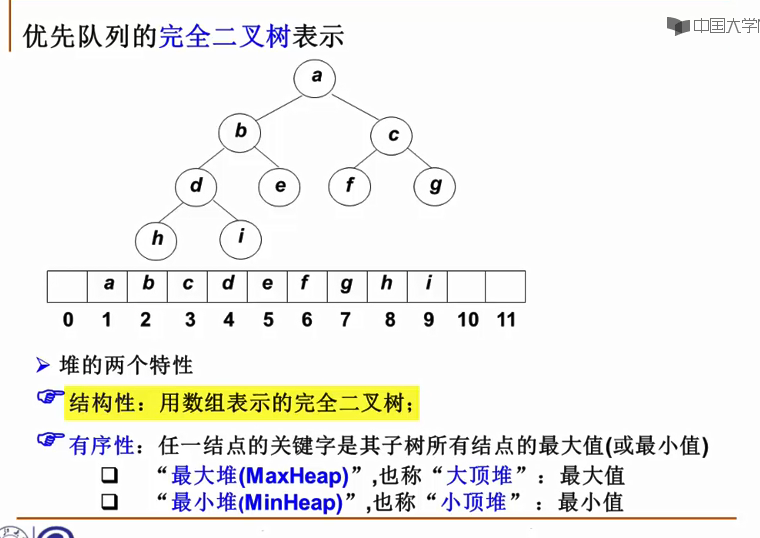
## 优先队列：



## 优先队列的实现思路：



## 优先序列完全二叉树的表示：



## 堆的举例：

### 完全二叉树的判定：

**若设二叉树的深度为h，除第 h 层外，其它各层 (1～h-1) 的结点数都达到最大个数，第 h 层所有的结点都连续集中在最左边，这就是完全二叉树。【判定方法】**

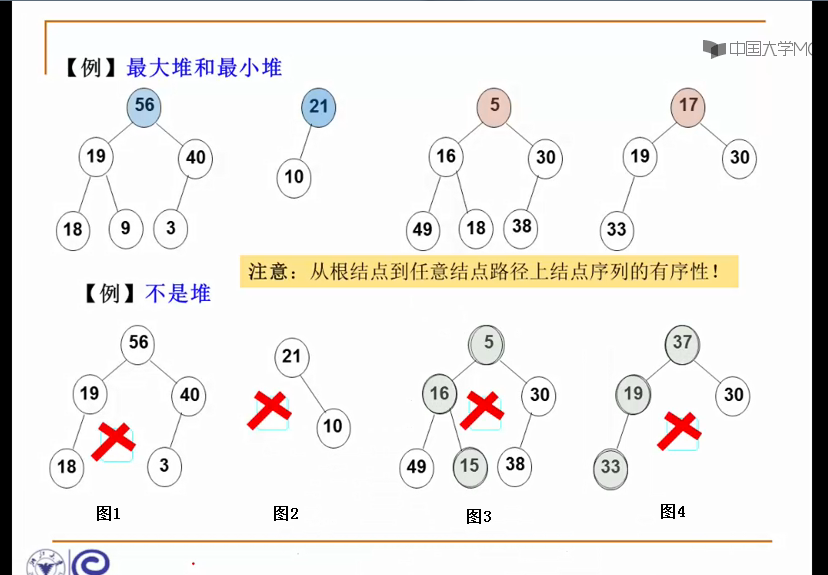


图1，不是堆得原因，不符合完全二叉树的定义。最后一层的节点没有连续集中在最左边。如果向19的右子树添加一个节点 或者 将值为3的节点删除，就是堆了。

图2，不是堆的原因：不符合完全二叉树的定影。最后一层的节点没有连续集中在最左边。

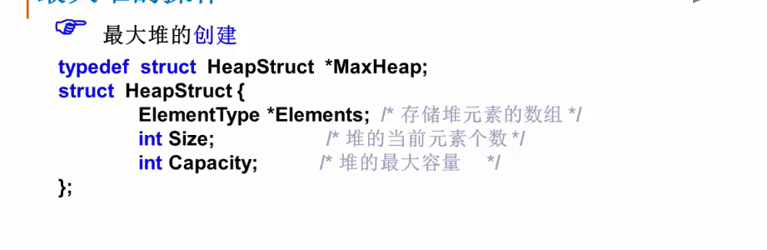
图3、4不是堆的原因：尽管他们都是完全二叉树，但是不符合有序性：任一节点的关键字是子树所有节点中的最大值（或最小值）

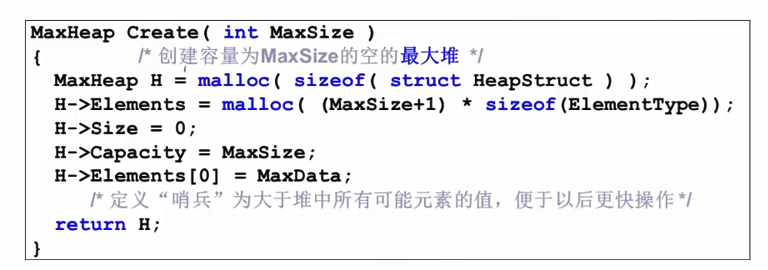
# 堆的抽象数据类型描述：



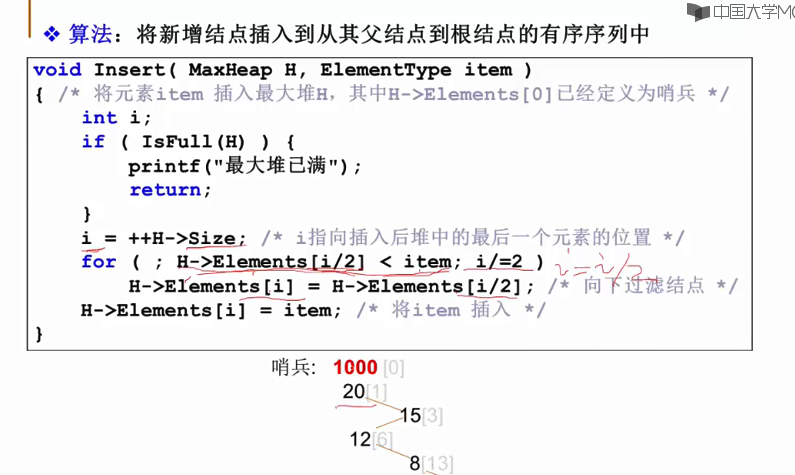
# 最大堆的操作：

## 最大堆得创建





## 插入，算法



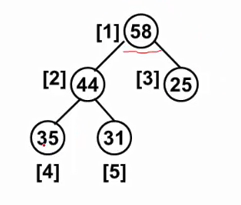
### 说明：

i/2 代表的是当前节点的 父节点的下标。

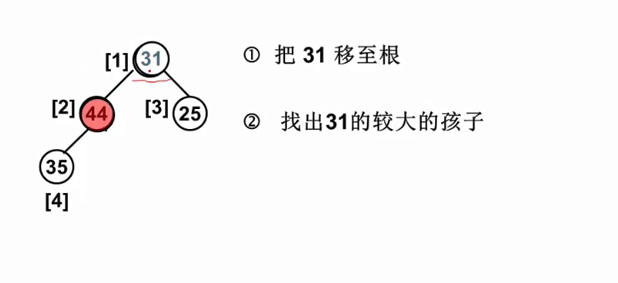
由于设置了哨兵，这里是设置了数组下标为0 的值 是一个最大值，只要你插入的节点的值是在范围内就不可能大过这个最大值，所以如果你 插入的 值是大于根节点，不需要再设置一个判断条件来防止数组越界。

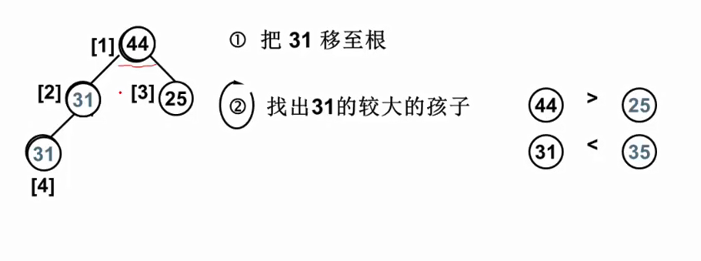
## 最大堆的删除

### 思路：

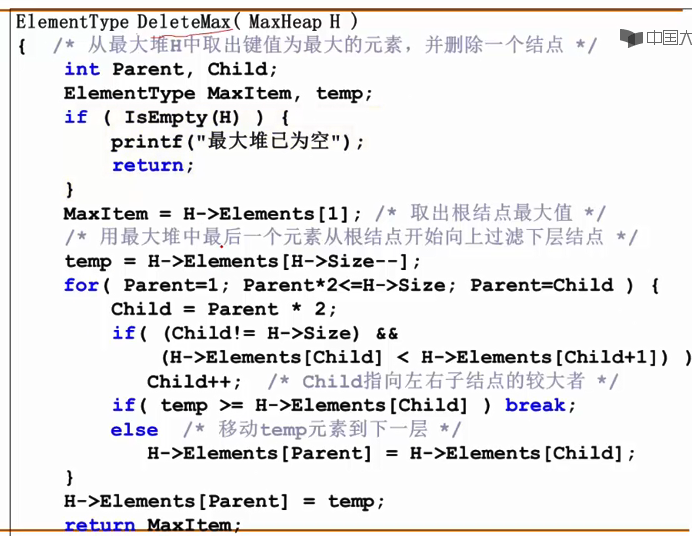


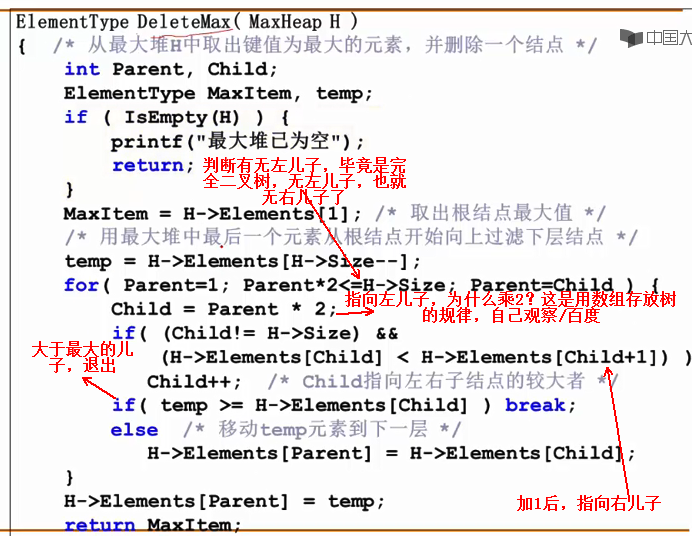
1. 将最后一个元素替代根节点
2. 接着找出现在根元素的较大的孩子
3. 进行有序性调整





### 实现代码





## 最大堆的建立

### 方式

