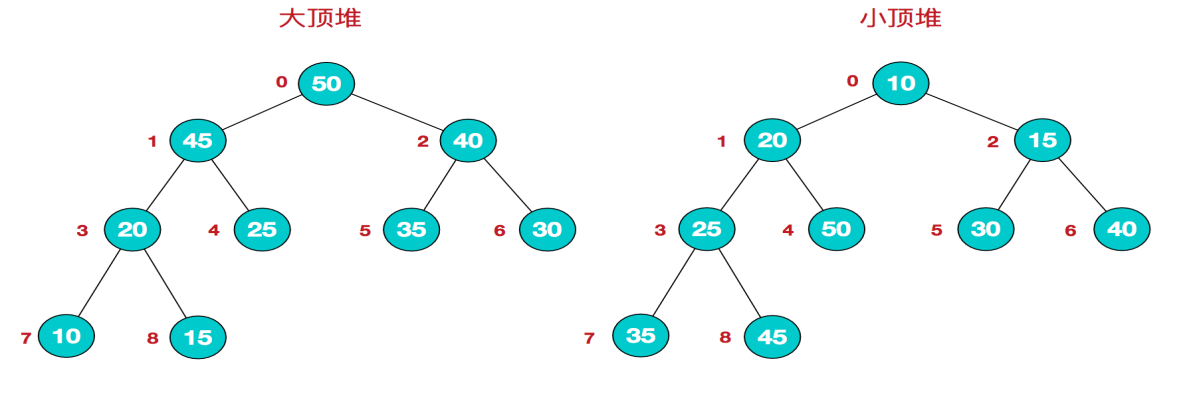
**堆排序**

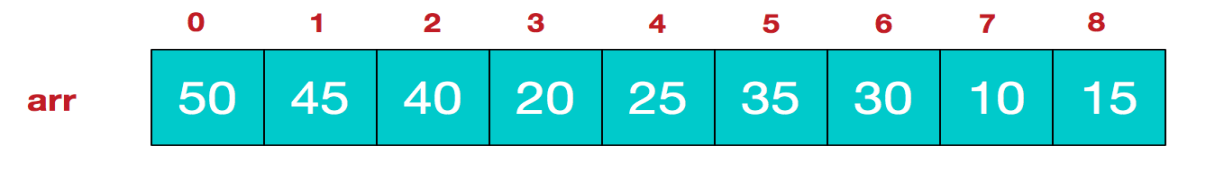
　　堆排序是利用**堆**这种数据结构而设计的一种排序算法，堆排序是一种**选择排序，**它的最坏，最好，平均时间复杂度均为O(nlogn)，它也是不稳定排序。首先简单了解下堆结构。

**堆**

**堆是具有以下性质的完全二叉树：每个结点的值都大于或等于其左右孩子结点的值，称为大顶堆；或者每个结点的值都小于或等于其左右孩子结点的值，称为小顶堆。如下图：**



同时，我们对堆中的结点按层进行编号，将这种逻辑结构映射到数组中就是下面这个样子



该数组从逻辑上讲就是一个堆结构，我们用简单的公式来描述一下堆的定义就是：

**大顶堆：arr[i] >= arr[2i+1] && arr[i] >= arr[2i+2]**

**小顶堆：arr[i] <= arr[2i+1] && arr[i] <= arr[2i+2]**

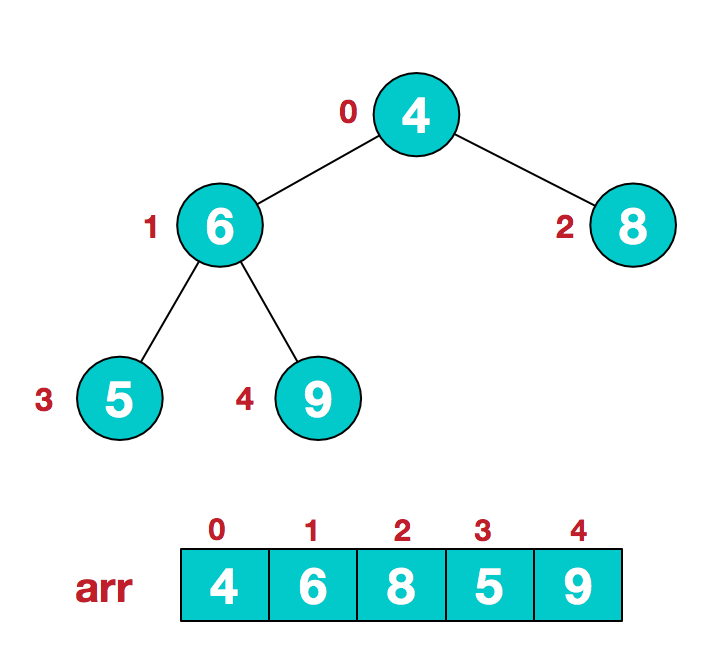
ok，了解了这些定义。接下来，我们来看看堆排序的基本思想及基本步骤：

**堆排序基本思想及步骤**

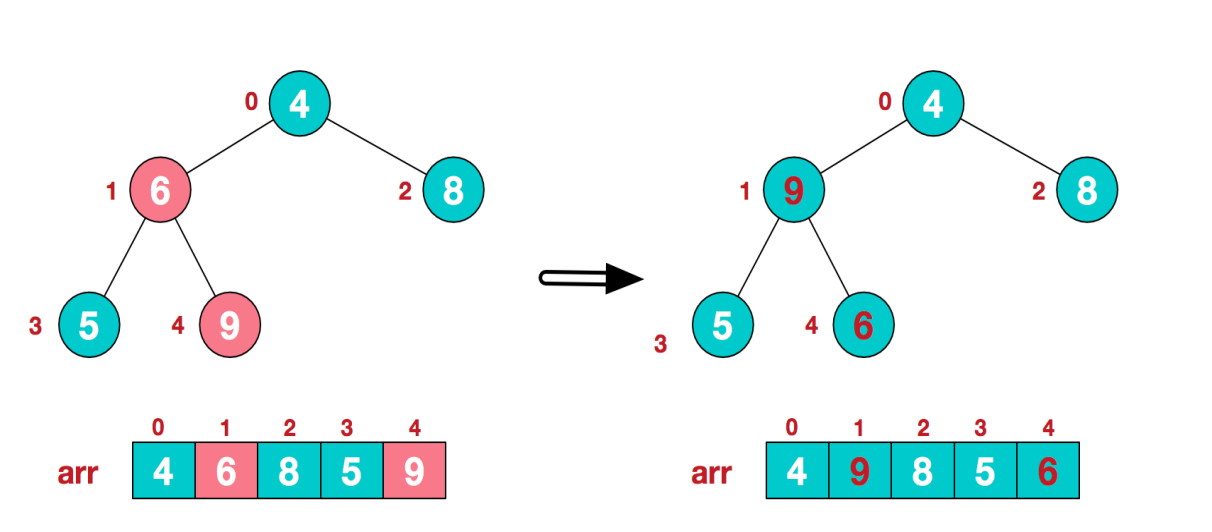
**堆排序的基本思想是：将待排序序列构造成一个大顶堆，此时，整个序列的最大值就是堆顶的根节点。将其与末尾元素进行交换，此时末尾就为最大值。然后将剩余n-1个元素重新构造成一个堆，这样会得到n个元素的次小值。如此反复执行，便能得到一个有序序列了**

**步骤一 构造初始堆。将给定无序序列构造成一个大顶堆（一般升序采用大顶堆，降序采用小顶堆)。**

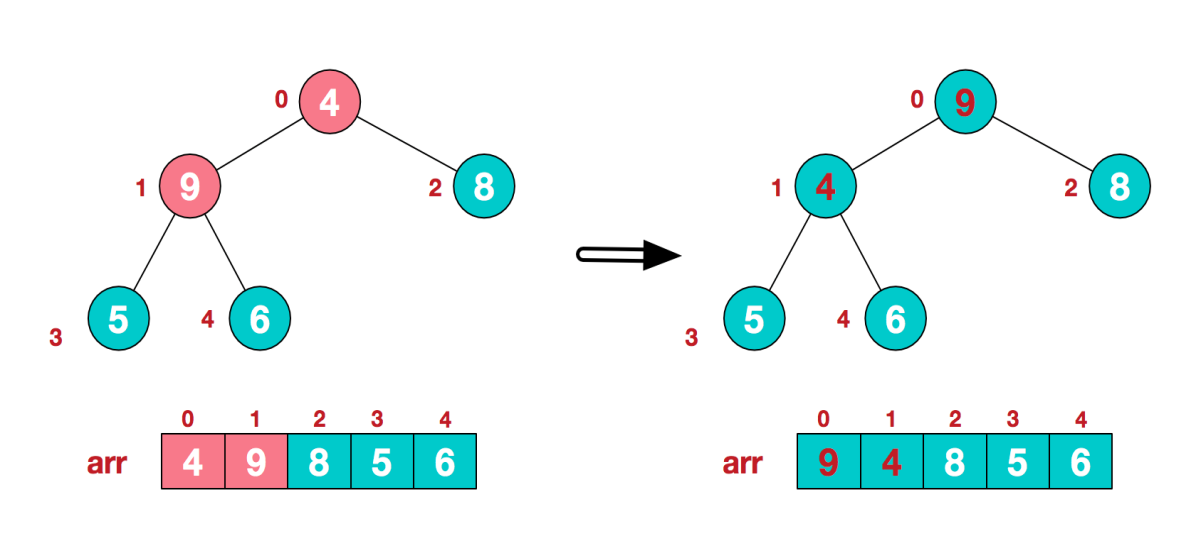
　　a.假设给定无序序列结构如下



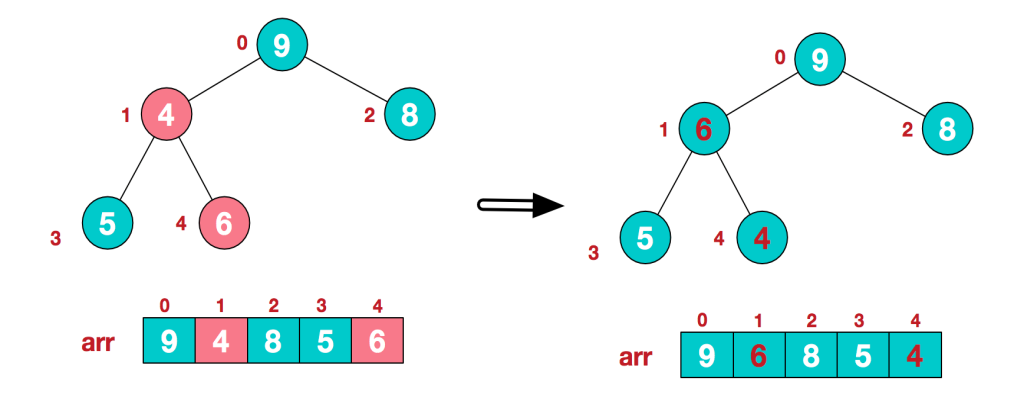
2.此时我们从最后一个非叶子结点开始（叶结点自然不用调整，第一个非叶子结点 arr.length/2-1=5/2-1=1，也就是下面的6结点），从左至右，从下至上进行调整。



3.找到第二个非叶节点4，由于[4,9,8]中9元素最大，4和9交换。



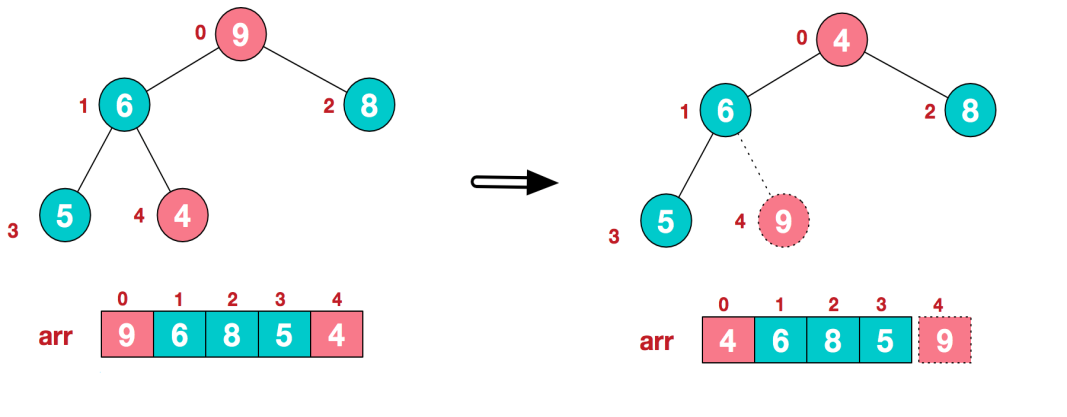
这时，交换导致了子根[4,5,6]结构混乱，继续调整，[4,5,6]中6最大，交换4和6。



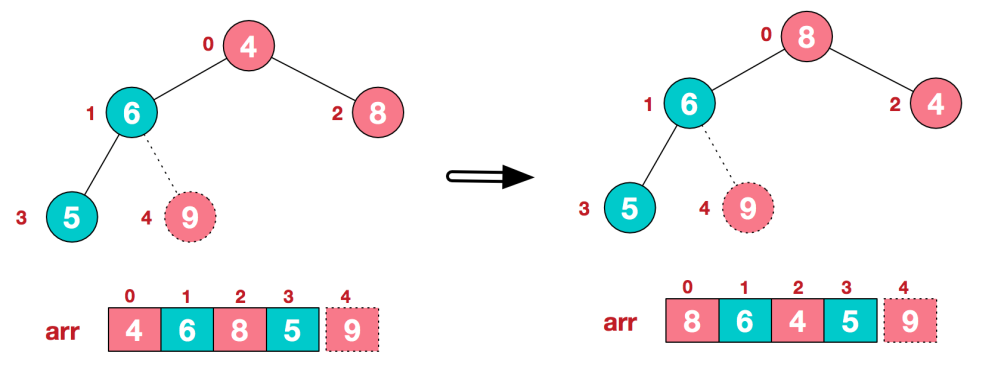
此时，我们就将一个无需序列构造成了一个大顶堆。

**步骤二 将堆顶元素与末尾元素进行交换，使末尾元素最大。然后继续调整堆，再将堆顶元素与末尾元素交换，得到第二大元素。如此反复进行交换、重建、交换。**

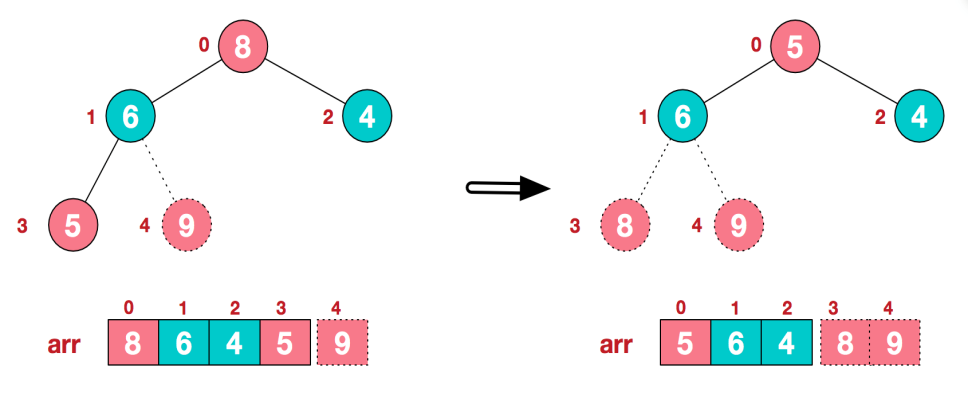
a.将堆顶元素9和末尾元素4进行交换



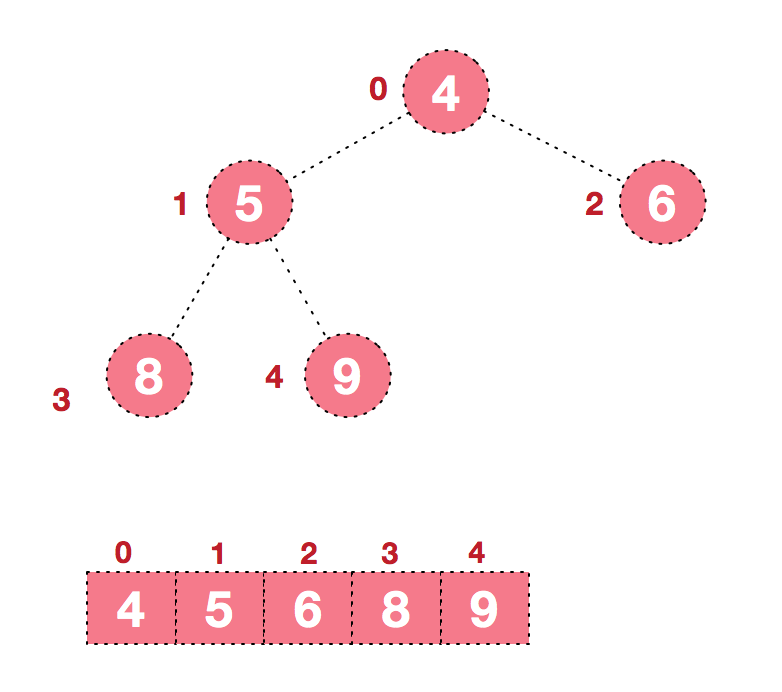
b.重新调整结构，使其继续满足堆定义



c.再将堆顶元素8与末尾元素5进行交换，得到第二大元素8.



后续过程，继续进行调整，交换，如此反复进行，最终使得整个序列有序



再简单总结下堆排序的基本思路：

**a.将无需序列构建成一个堆，根据升序降序需求选择大顶堆或小顶堆;**

**b.将堆顶元素与末尾元素交换，将最大元素"沉"到数组末端;**

**c.重新调整结构，使其满足堆定义，然后继续交换堆顶元素与当前末尾元素，反复执行调整+交换步骤，直到整个序列有序。**

**代码实现**

**package** facehandjava.sort;  
  
**public class** HeapSort {  
 **public static void** main(String[] args) {  
 **int**[] arrays = {10, 5, 36, 78,56,2, 5, 8, 9, 9};  
  
 **int** min = 0;  
 **int** max = arrays.**length**-1;  
 System.***out***.print(**"原来的："**);  
 **for**(**int** i =0;i<=max;i++) {  
 System.***out***.print(arrays[i]+**","**);  
 }  
 System.***out***.println();  
 *HeapSort*(arrays,max);  
 System.***out***.print(**"排序后："**);  
 **for**(**int** i =0;i<=max;i++) {  
 System.***out***.print(arrays[i]+**","**);  
 }  
  
 }  
  
 **public static void** HeapSort(**int**[] arrays, **int** max) {  
 *//构造堆* **for** (**int** i = max / 2 - 1; i >= 0; i--) {  
 *headAdjust*(arrays, i, max);  
 }  
 *//调换最后一个，再重新构造堆* **while** (max >= 0) {  
 *swap*(arrays, 0, max);  
 max--;  
 *headAdjust*(arrays, 0, max);  
 }  
  
 }  
  
 **public static void** headAdjust(**int**[] arrays, **int** i, **int** max) {  
  
 **int** k ;  
 **int** left;  
 **int** right;  
 **while** ((left= i \* 2 + 1) <= max) {  
 right = left + 1;  
 k = left;  
 *//k<max，证明有right。如果右边大，则指针指向右边。* **if** (k < max && arrays[k] < arrays[right]) {  
 k++;  
 }  
 *//父比子小，交换。并且i要指向子，进行下一步的判断，如果还继续有子节点，则要继续判断和交换，直到i没有子节点* **if** (arrays[i] < arrays[k]) {  
 *swap*(arrays, i, k);  
 i = k;  
 } **else** {  
 **break**;  
 }  
 }  
  
 }  
  
 **public static void** swap(**int**[] arrays,**int** a, **int** b) {  
 **int** temp = arrays[a];  
 arrays[a] = arrays[b];  
 arrays[b] = temp;  
 }  
}