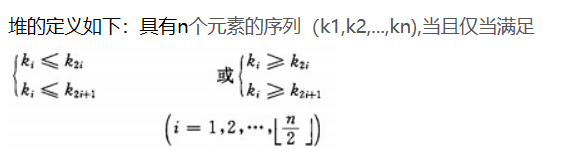
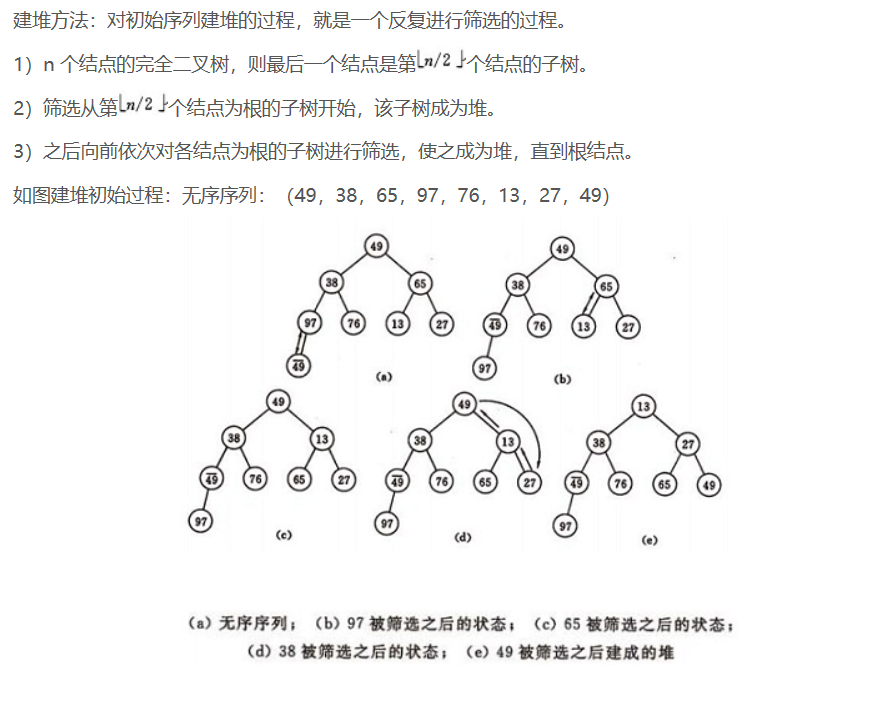
HeapSort(堆排序)

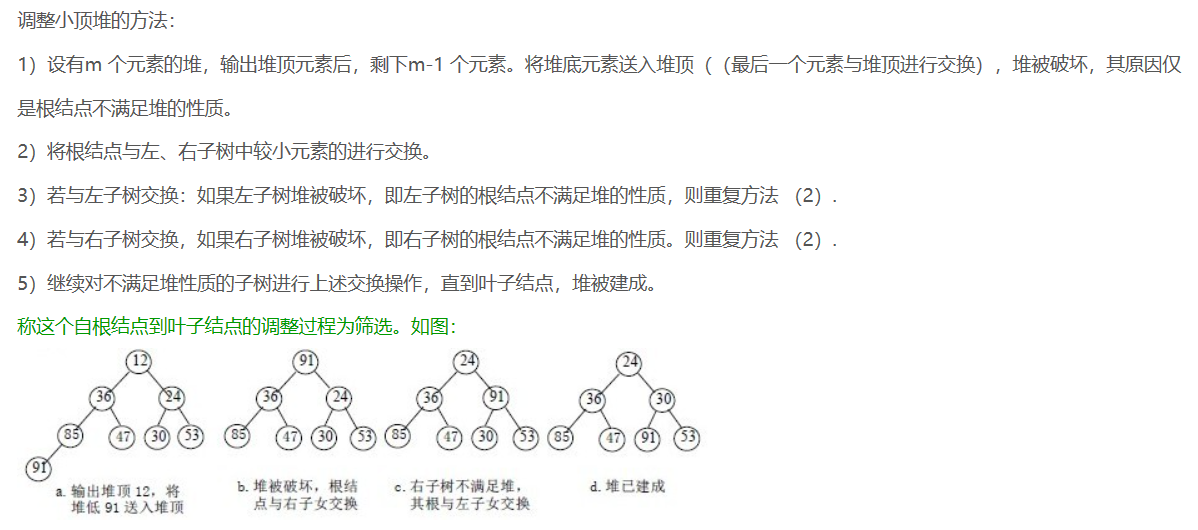
1. 概念：堆排序是一种树形选择排序，是对直接选择排序的有效改进。堆分为最大堆和最小堆，堆其实是一种二叉树结构，最大堆即每个父节点都大于其左右子节点，最小堆即为给个父节点都小于左右子节点。



1. 方法：先对初始数组进行创建最大或最小堆，然后在取堆顶元素（0号元素）与n-1号元素交换，将0号取出，得到数组最大或最小值，之后对剩下n-1个元素进行继续创建堆。在取堆顶元素，直到全部排好序。
2. 举个例子：在堆排序中，由于是先将一个数组按最大或最小堆建成，然后去顶，在对剩下的n-1个元素进行建堆。
3. 创建堆的方法：



1. 对剩下n-1个元素建堆：



1. 注意：在用最大最小堆排序后，如果要按原理输出，即最大堆降序，最小堆升序，则需要从n-1开始输，因为每次去掉的那个元素会放在数组的最后。
2. 代码：
3. 元素下滤，最大堆将大值放到父节点，最小堆将小值放到父节点。（以最小堆为例）

**public** **static** **void** PercolateDown(**int**[] array, **int** index, **int** length) {

**int** min;

**while** (index \* 2 + 1 < length) {

min = index \* 2 + 1;//左节点

**if** (index \* 2 + 2 < length) {

**if** (array[min] > array[index \* 2 + 2]) {

min = index \* 2 + 2;//右节点

}

}

**if** (array[index] < array[min]) {//如果左右节点中有一个值大于其根节点则退出

**break**;

} **else** {

*Swap*(array, index, min);//把小的子节点与大的（根）交换

index = min;//下移一层继续(或退出)，直到全部完成

}

}

1. 建堆：

**public** **static** **void** BuildHeap(**int**[] array, **int** length) {

**for** (**int** i = length / 2 - 1; i >= 0; i--) {

*PercolateDown*(array, i, length);

}

}

1. 堆排序：

//堆排序，就是把最上面的根元素取出，剩下的元素再次构建堆，然后在取出最上面的，直到i=1,取出的序列即为排好序的

**public** **static** **void** HeapSort(**int**[] array, **int** length) {

*BuildHeap*(array, length);

**for** (**int** i = length - 1; i >= 1; i--) {

*Swap*(array, 0, i);

length--;

*PercolateDown*(array, 0, length);

}

}

1. 时间复杂度分析：

