算法第三次作业-10205101530-赵晗瑜

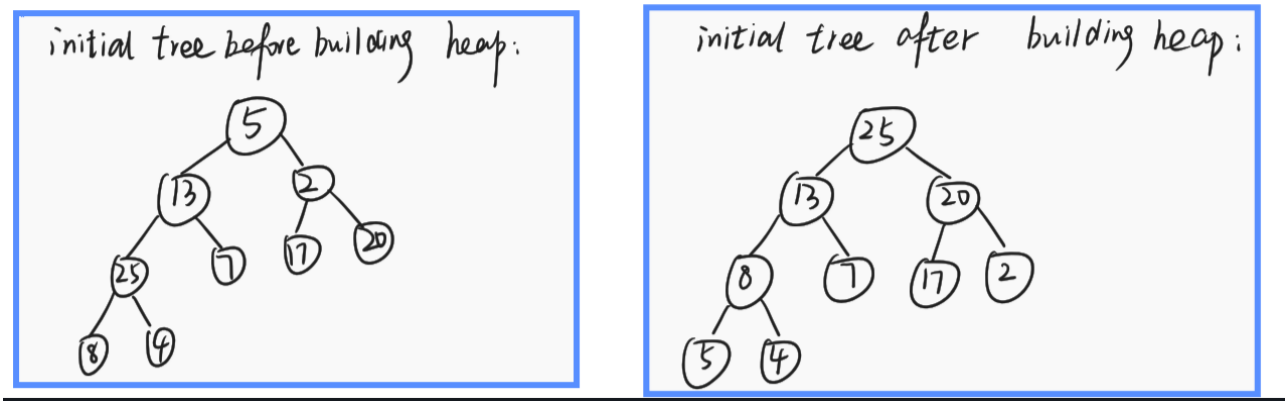
4.1-5

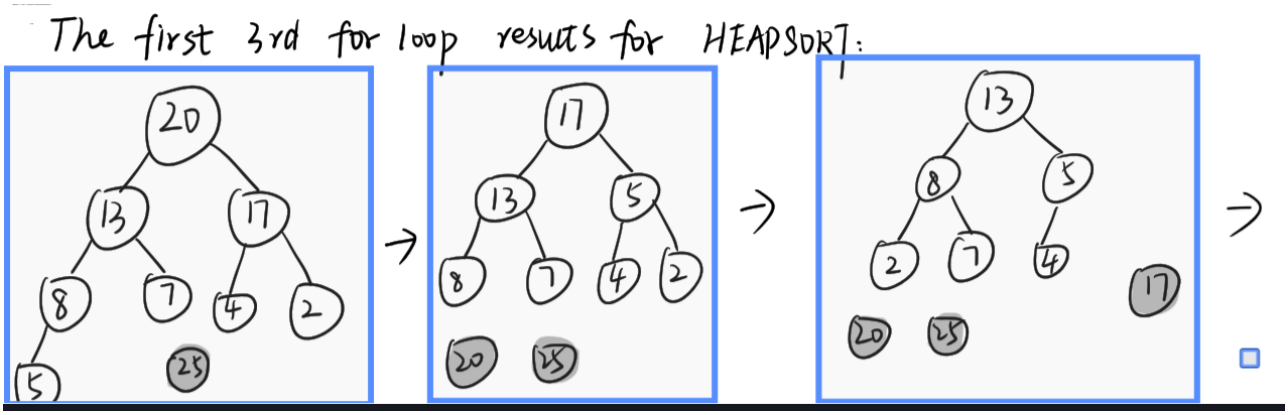
使用如下思想为最大子数组问题设计一个非递归的、线性时间的算法。从数组的左边界开始，由左至右处理，记录到目前为止已经处理过的最大子数组。若已知$A[1..j]$的最大子数组，基于如下性质将解扩展为$A[1..j+1]$的最大子数组：$A[1..j+1]$的最大子数组要么是$A[1..j]$的最大子数组，要么是某个子数组$A[i..j+1](1\leqslant i\leqslant j+1)$。在已知$A[1..j]$的最大子数组的情况下，可以在线性时间内找出形如$A[i..j+1]$的最大子数组。

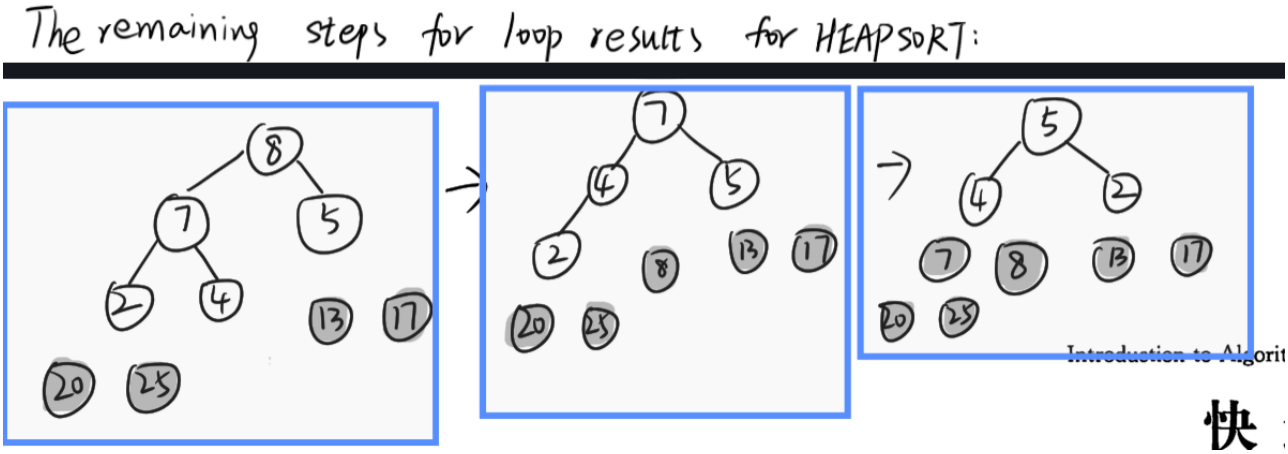
```
//如果所有的元素都是负数，则返回最大元素
FIND-MAXIMUM-SUBARRAY-LIMEAR(A)
    low=1       //要求的最大子数组的左下标
    high=1      //要求的最大子数组的右下标
    finalMaxsum=A[1]      //最大子数组各元素的和
    currentLow=1          //当前low
    currentMaxsum=A[1]    //当前high
    for j=2 to A.length
        //当前currentMaxsum是以A[j-1]结尾的子数组的和
        //如果currentMaxsum>0，我们实际需要的以A[j]结尾的子数组是
A[currentLow..j]
        //因为A[j]必须被包含并且A[currentMaxsum..j-1]代表了最大子数组
        if currentMaxsum<0
            currentMaxsum=A[j]
            currentLow=j
        else
            currentMaxsum=currentMaxsum+A[j]

        if currentSum>finalMaxsum
            finalMaxsum=currentSum
            low=currentLow
            high=j
    return (low,high,finalMaxsum);
```

6.4.1(Draw the result heaps of BUILD-MAX-HEAP(A) and the first 3 rounds of for loop.)

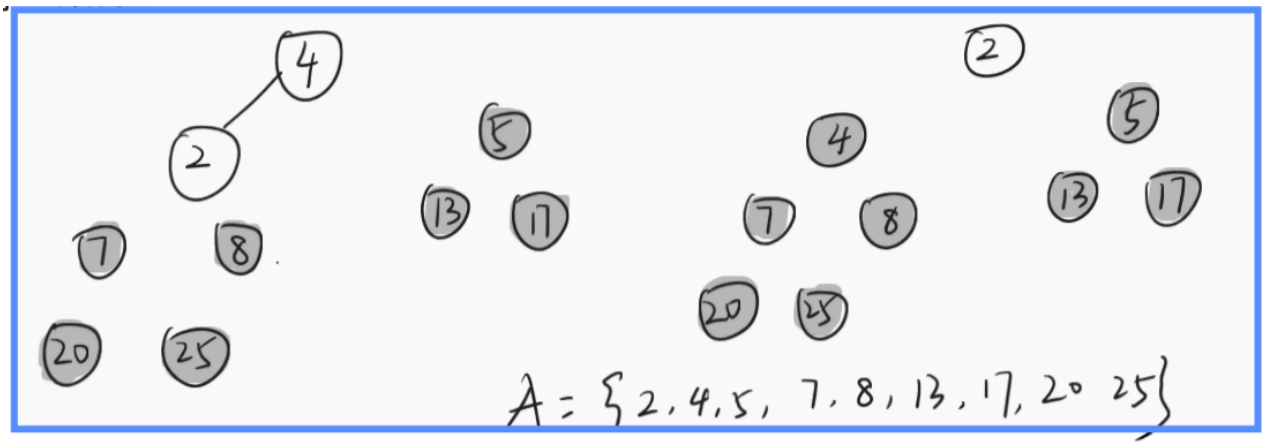
参照图 6-4 的方法，说明 HEAPSORT 在数组 $A=\langle 5, 13, 2, 25, 7, 17, 20, 8, 4\rangle$ 上的操作过程。







快

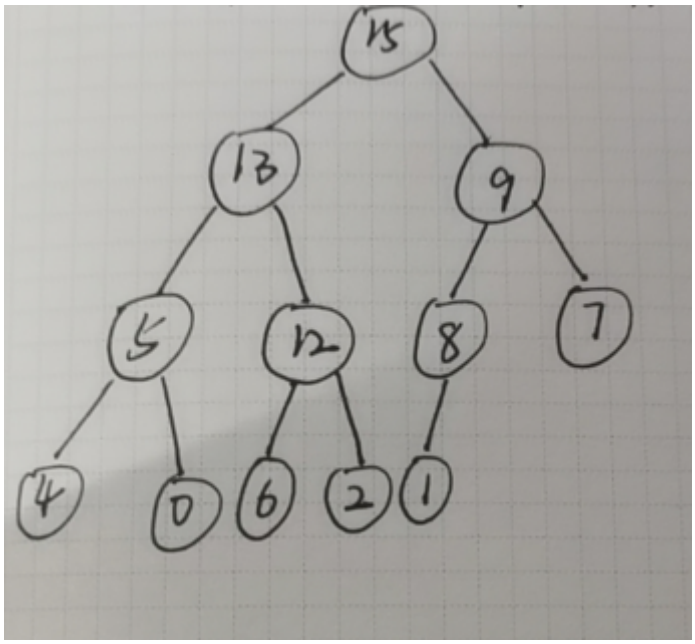


6.5-2

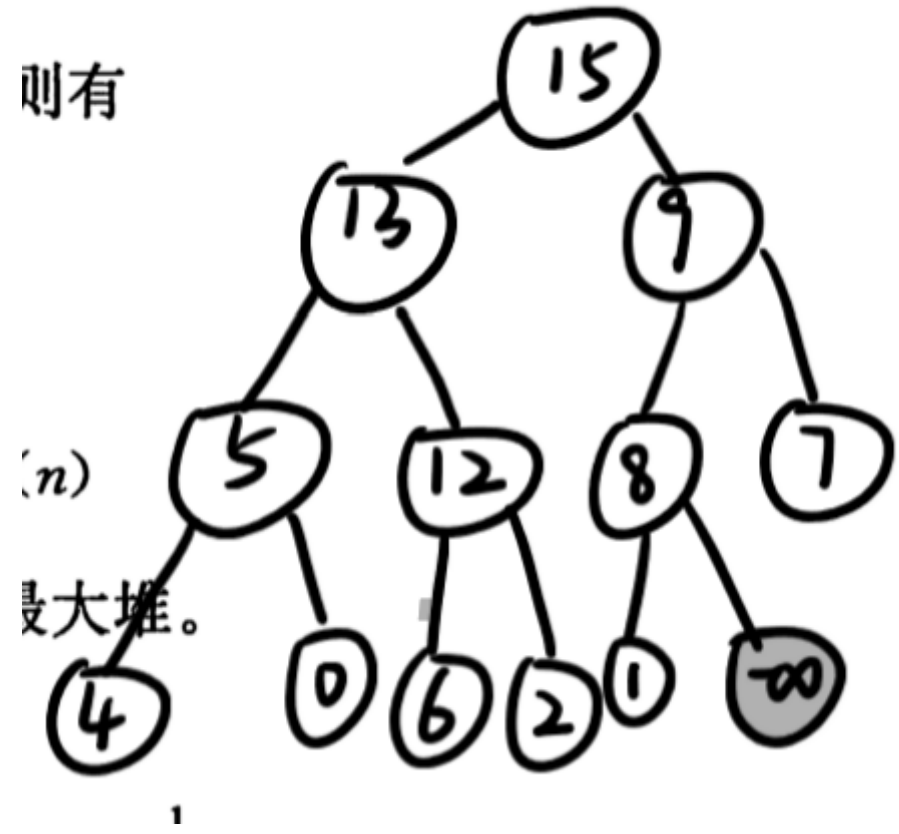
试说明 MAX-HEAP-INSERT(A，10)在堆 A=〈15，13，9，5，12，8，7，4，0，6，2，1〉上的操作过程。

The following sequence of pictures shows how 10 is inserted into the heap,then swapped with parent nodes until the max-heap property is restored. The node containing the new key is heavily shaded.

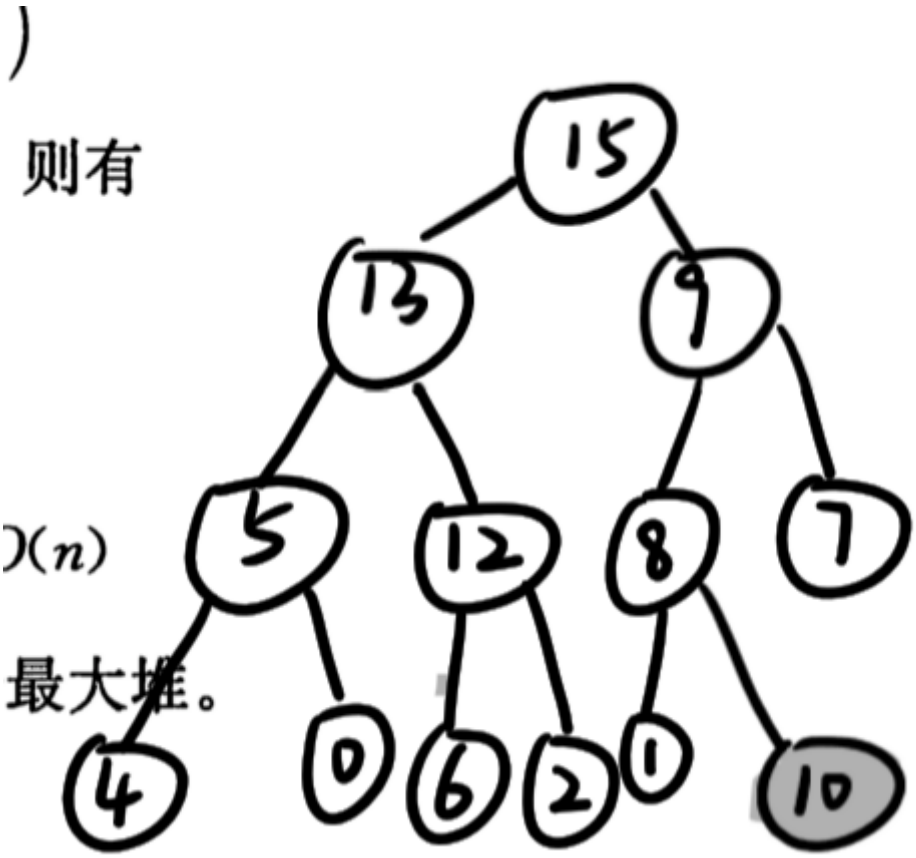
1.original heap



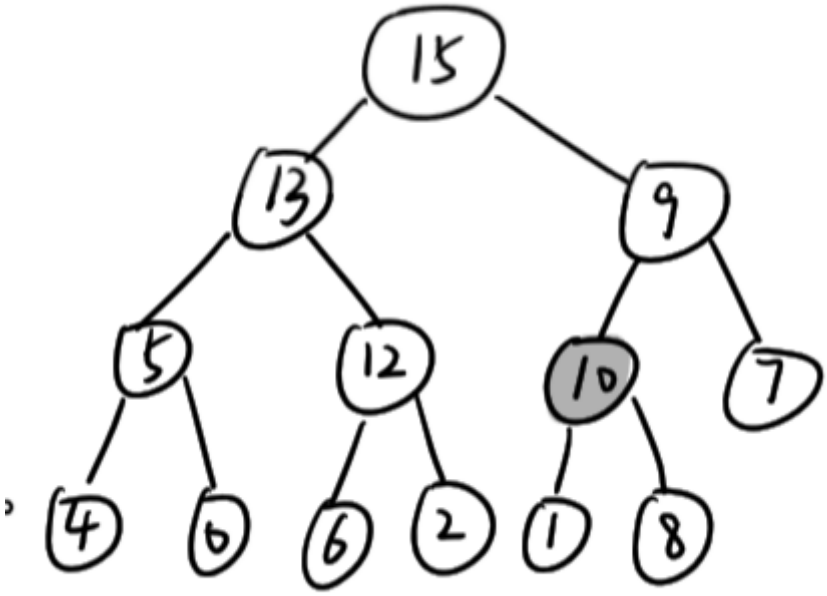
2. MAX-HEAP-INSERT(A,10) 被调用，所以我们首先附加一个节点被赋值为-∞的节点：



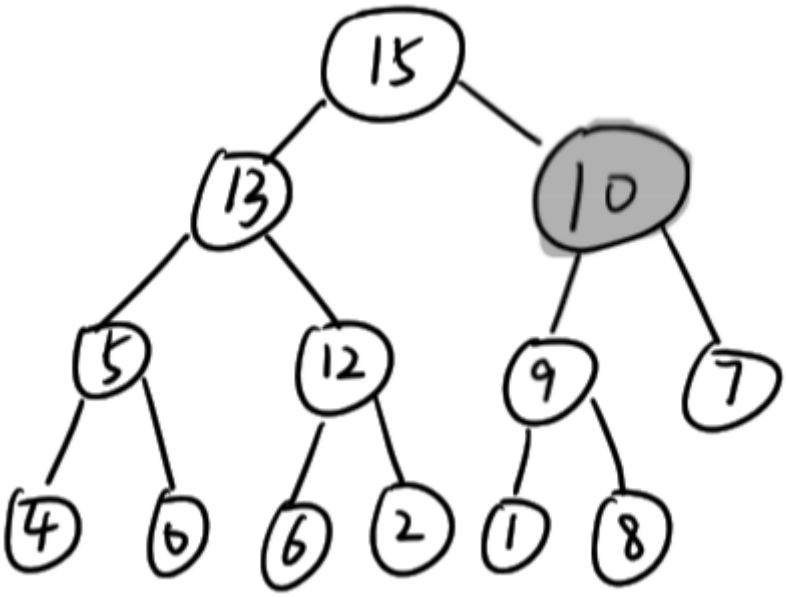
3.进行新添加的节点的健值更新：



4.由于父键小于 **10**，因此交换节点：



5.由于父节点小于**10**，节点交换：



6.5-9

请设计一个时间复杂度为 $O(n \lg k)$ 的算法，它能够将 k 个有序链表合并为一个有序链表，这里 n 是所有输入链表包含的总的元素个数。（提示：使用最小堆来完成 k 路归并。）

```
MERGE-SORTED-LISTS(lists)
    let heapList be an empty array
    //选取每个list的头部(head)构造一个最小堆(min heap)
    for i=1 to lists.length
        push(lists[i][0],i,0) into heapList
    BUILD-MIN-HEAP(heapList)

    let mergedlist be an empty array
    while not EMPTY(heapList)
        //为了找到已排序数组的下一个元素，我们提取出最小元素(在Olg(k)时间内)
        elem,li,ei=HEAP-EXTRACT-MIN(heapList)
        push elem into mergedlist
        //将提取的元素最初来自的更短的列表中的下一个元素添加到堆中(也是O(lg(k))
时间)
        if ei<lists[li].length
            MIN-HEAP-INSERT(heapList,(lists[li][ei+1],li,ei+1))
    return mergedlist
```

步骤：

1. 选取每个list的头部(head)构造一个最小堆(min heap)；
2. 然后，为了找到已排序数组的下一个元素，我们提取出最小元素(在Olg(k)时间内)；
3. 然后，将提取的元素最初来自的更短的列表中的下一个元素添加到堆中(也是O(lg(k))时间)。
4. 因为找到排序列表中的下一个元素最多只需要O(lg(k))个时间，所以要找到整个列表，总共需要O(n lg(k))个步骤