def sift(dataSet ,low, high):  
 *"""* ***:param*** *dataSet: 列表* ***:param*** *low: 堆的根节点位置* ***:param*** *high: 堆的最后一个元素位置* ***:return****:  
 """* i = low #i最开始指向的根节点  
 j = 2 \* i + 1 #j开始左叶子  
 temp = dataSet[low] #存堆顶  
 while j <= high: #只要j位置有数  
 if j + 1 <= high and dataSet[j + 1] > dataSet[i]:  
 j = j + 1 #指向右叶子  
 if dataSet[j] > temp:  
 dataSet[i] = dataSet[j]  
 i = j #往下看一层  
 j = 2 \* i + 1  
 else:  
 break  
 dataSet[i] = temp #temp放到某一个根节点  
  
def heapSort(dataSet):  
 length = len(dataSet)  
 for i in range((length-2)//2 , -1, -1): #调整根部分的下标  
 sift(dataSet ,i , length - 1) #建堆完成  
 print(dataSet)

建堆

向下调整

找最大值

**推排序-topk问题**

**1//**取列表前k个元素建立一个小根堆

**2//**依次向后遍历原列表，

如果小于堆顶，忽略

如果大于堆顶，更换堆顶元素，并调整

**3//**遍历列表所有元素后弹出堆顶

def topk(dataSet , k):  
 heap = dataSet[0:k]  
 #前k个元素  
 for i in range((k - 2)//2 , -1, -1):  
 sift(heap , i, k - 1)

#1.建堆  
 for i in range(k , len(dataSet) - 1):  
 if dataSet[i] > heap[0]:  
 heap[0] = dataSet[i]  
 sift(heap , 0, k-1)  
 #2.遍历  
 for i in range(k-1 , -1, -1):  
 heap[0], heap[i] = heap[i], heap[0]  
 sift(heap , 0, i - 1)  
 #3.出数  
 return heap

**归并排序：**

**分解+合并**