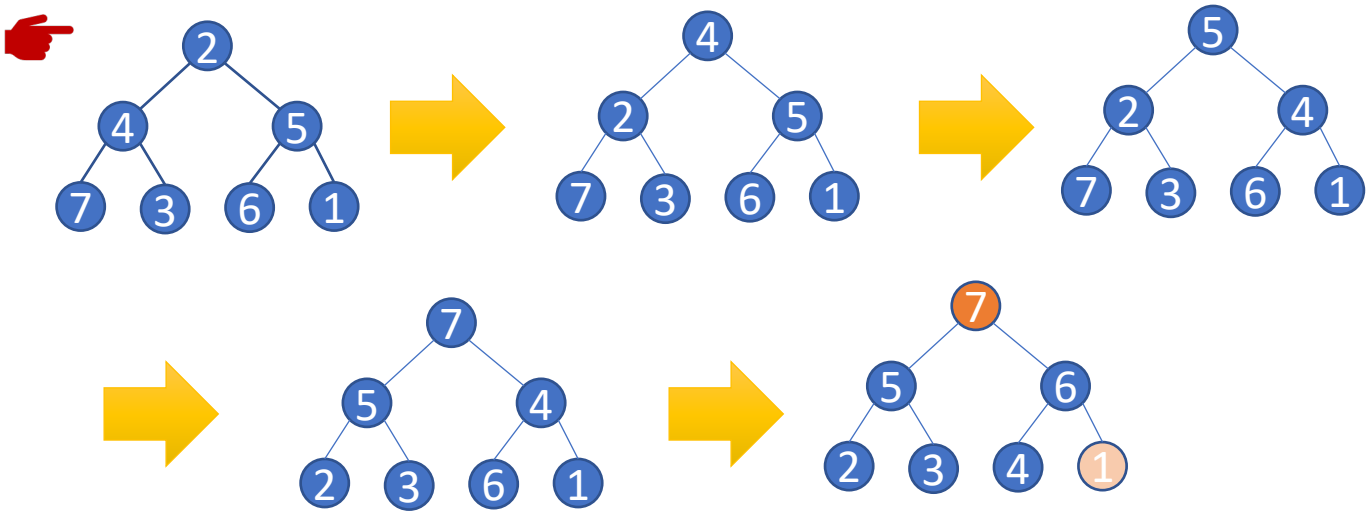
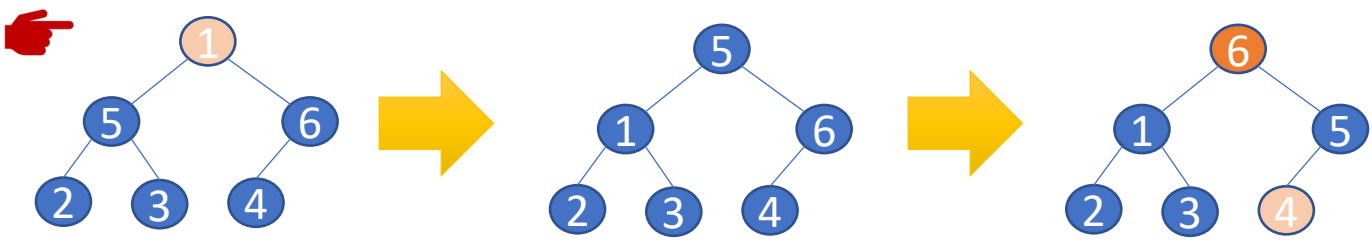
Heap Sort

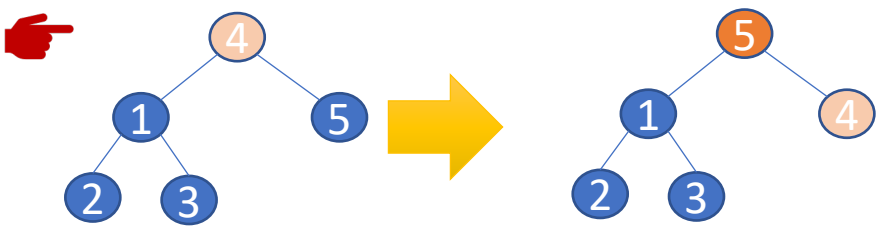
05113009
哲學四 吳家瑩

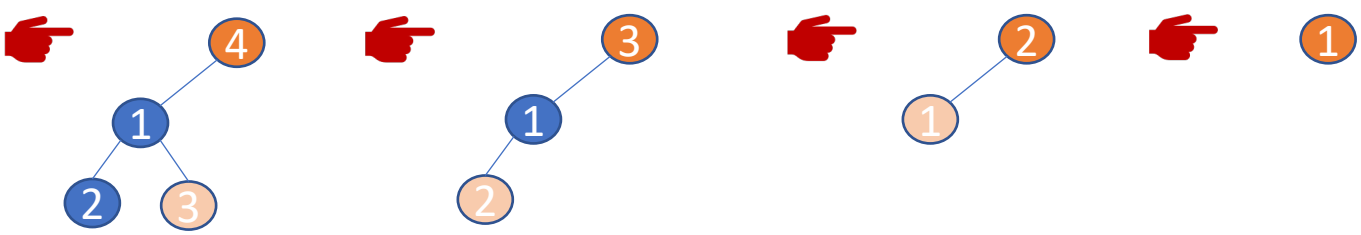
Index: 1. 2. 3. 4. 5. 6. 7.

2	4	5	7	3	6	1









1	2	3	4	5	6	7

學習歷程與程式說明

```
def heapsort(list):
    if len(list) <= 1:
        done = False

    else:
        done = True
        i = 0

    while done is True:
        if list[2*i+1] > list[i] :
            temp = list[i]
            list[i] = list[2*i+1]
            list[2*i+1] = temp
            heapsort(list)

        if len(list)>=3:
            if list[2*i+2] > list[i] :
                temp = list[i]
                list[i] = list[2*i+2]
                list[2*i+2] = temp
                d1 = True
                heapsort(list)

        i += 1

        if 2*i+1 > len(list)-1 :
            done = False

        if 2*i+2 > len(list)-1 :
            done = False
```

先判斷此數列的長度是否大於1，如果小於等於1，則我們就不需要進行排序。

左邊的子節點index:2i+1，右邊的子節點index:2i+2。

若數列長度大於1，則進行排序。
若子節點>父節點，則進行換位，換位後需要再向新的父節點比較，並依照index的順序進行比較。

若數列長度>=3，才會有右邊的子節點。

若是此節點已經沒有子節點，則不需要再進行排序。
但是問題來了，此處的判斷式可能會使最後一個數字沒有進行排序。

```
        if 2*i+2 <= len(list)-1:
            if list[2*i+2] > list[i] :
                temp = list[i]
                list[i] = list[2*i+2]
                list[2*i+2] = temp
                d1 = True
                change = True

        i += 1

        if 2*i+1 > len(list)-1 :
            done = False

    if change == True:
        heapsort(list)
```

我們將判斷式改成這個，就能避免發生error。

新增此程式碼，可增加程式的效率，減少進行無意義的排序。
判斷是否有換位，若是有才會再進行一次比較，
若是沒有，則不須再進行比較

In [128]:
```
def heap(list):
    templist = []
    heapsort(list)
    helper(list,templist)
    return templist
```

In [129]:
```
def helper(list,templist):
    templist.insert(0,list[0])
    list[0] = list[-1]
    list = list[:-1]
    heapsort(list)
```

為了要讓數列進行多次比較進行排列，所以我就開始想要怎麼寫，所以就寫出錯的程式碼，不過沒有失敗哪來的成功，所以經過幾次的努力與修改，我終於成功了！

In [254]:
```
def heap_sort(list):
    heapsort(list)
    templist=[]
    templist.insert(0,list[0])
    for i in range(len(list)-1):
        list[0] = list[-1]
        list = list[:-1]
        heapsort(list)
        templist.insert(0,list[0])
    return templist
```

首先我們需要先進行一次排序，將最大值取出後放入templist的第一個位置，再將最後一個值移到第一個位置，並進行第二次排序，依此類推，直到不能再進行排序，我們就會得到排序好的數列。

In [258]:
```
list = [7,4,6,5,2,8,1]
heap_sort(list)
```

成功了！！！

Out[258]: [1, 2, 4, 5, 6, 7, 8]

花了整整一天，經歷了無數次的失敗後，想出了這個函式，雖然一開始只能夠完整的排序一次，將最大值取出後，我就不知道要怎麼讓它進行第二次排序，但是當想出如何進行第二次排序的時候，心中真的無比感動，沒想到自己憑著老師上課教的記憶，沒有參考任何人的程式，靠自己的努力就想出來了，雖然速度不是很快，程式寫的也不是很好，但相信經歷多次的訓練，一定能寫出更好更簡潔的程式。

In [259]:
```
list = [7,4,6,5,2,8,1]
print('list:',list)
a = []
heapsort(list)
a.insert(0,list[0])
print('sort:',list)
print('output:',a)
print()
for i in range(6):
    list[0] = list[-1]
    list = list[:-1]
    print('list:',list)
    heapsort(list)
    a.insert(0,list[0])
    print('sort:',list)
    print('output:',a)
    print()
```

```
list: [7, 4, 6, 5, 2, 8, 1]
sort: [8, 5, 7, 4, 2, 6, 1]
output: [8]

list: [1, 5, 7, 4, 2, 6]
sort: [7, 4, 5, 1, 2, 6]
output: [7, 8]

list: [6, 4, 5, 1, 2]
sort: [6, 4, 5, 1, 2]
output: [6, 7, 8]

list: [2, 4, 5, 1]
sort: [5, 2, 4, 1]
output: [5, 6, 7, 8]

list: [1, 2, 4]
sort: [4, 1, 2]
output: [4, 5, 6, 7, 8]

list: [2, 1]
sort: [2, 1]
output: [2, 4, 5, 6, 7, 8]

list: [1]
sort: [1]
output: [1, 2, 4, 5, 6, 7, 8]
```

原數列
排序後
取出的數列

為了讓自己更加了解排序的過程，
所以我將每一次的排序都印出來。

參考資料

- http://alrightchiu.github.io/SecondRound/comparison-sort-heap-sortdui-ji-pai-xu-fa.html
- 老師上課講解