Heap sort

分成四頁

第一頁文字說明

第二-四頁流程圖

（製造成max heap 讓陣列由小到大排序）->最大的放在最後,剩下再次heapify,依序處理,最後排成一個由小到大的list

概念如下：

step1:是將陣列排序到完全二元樹裡面,

因為要製造max heap所以

step2-heapify:將每一個二元樹裡面的左子樹和右子樹和該二元樹的父節點比較,如果左子樹有大於該二元樹的父節點的話,則左子樹換到父節點（因為要滿足父節點大於子節點的條件）,同理,如果右節點也大於父節點的話,那麼就將右子樹也換到父節點,然後再繼續向下處理,最後排序成一個max heap(滿足所有父節點都大於子節點的條件）！

step3-heapsort:把heapify後的二元樹當中位在i=0（整個完全二元樹的最上面的節點)與i=len(nums)-1(整個完全二元樹的最後一個節點)交換位置

重複步驟二:繼續heapify除了最後一個節點以外的二元樹。

重複步驟三

依此類推,完成整個陣列有小到大的排序。

時間複雜度O(nlogn):

將陣列做maxheapify:需要花logn*n次->因為每次對調完,會再對剩下的node進行maxheapify所以要乘上n

將root與last node對調:n-1次

total:n-1+nlogn 省略常數,所以O(nlogn)

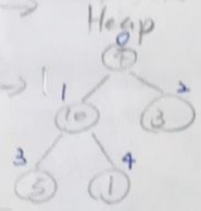
What: **Heap Sort**

```
def heap_sort(A, R)
    R = i
    l = 2i+1
    r = 2i+2
    if A[l] < A[R]:
        R = l
    if A[r] < A[R]:
        R = r
    if R != i:
        heapify(A, R)
```

Step 1 將 input data → Binary Tree → Heap

ex:

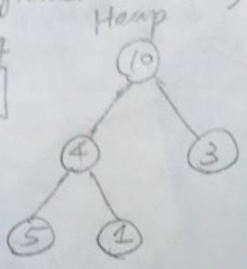
index	0	1	2	3	4
	4	10	3	5	1

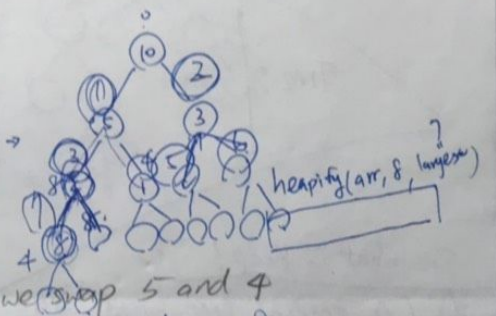


Step 2 Create Max Heap (Parent is greater than or equal to child nodes)

step 2-1 : 10 is greater than 4, So we swap 4 and 10

index	0	1	2	3	4
	10	4	3	5	1

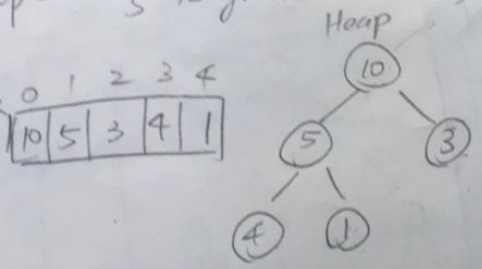




step 2-2 5 is greater than 4, So we swap 5 and 4

4 8

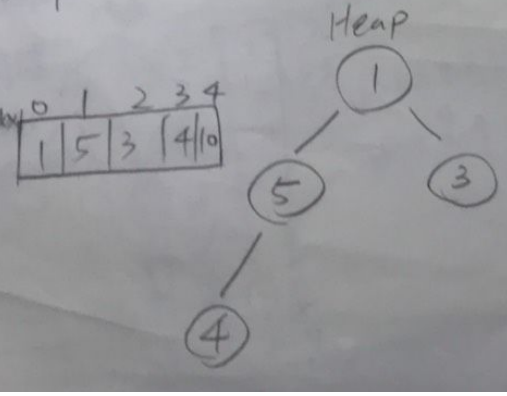
arr[3], arr[7] = arr[7], arr[3]



index	0	1	2	3	4
	10	5	3	4	1

(when all parent node is greater than or equal to child nodes)

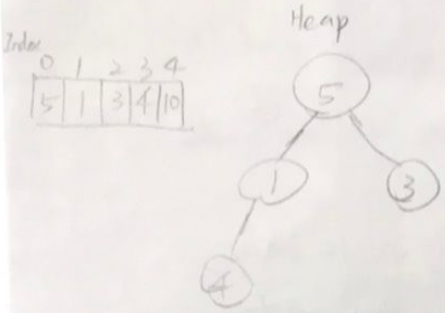
step 2-3 remove the node : Swap first and last node and delete the last node from heap



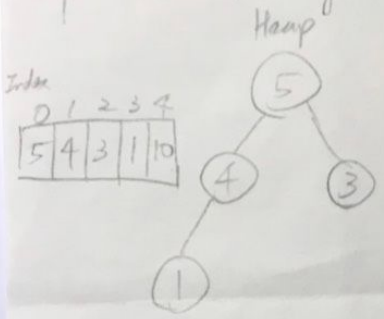
index	0	1	2	3	4
	1	5	3	4	10

Why swap first & last node?
→ return to the Parent node is not greater than or equal to child nodes's situation and Create Max Heap **AGAIN**!

Step 2-4 5 is greater than 1, so we swap 5 and 1

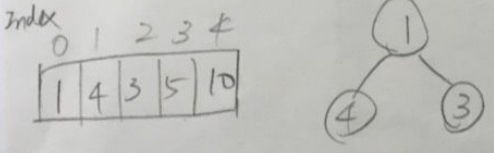


Step 2-5 4 is greater than 1, so we swap 4 and 1.

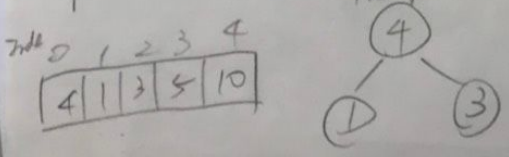


Step 2-6 remove the node: Swap first and last node and delete the last node from heap

Swap 1 and 5

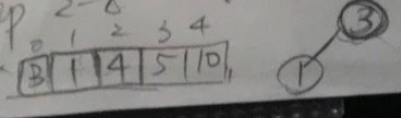


Step 2-7 4 is greater than 1, so swap 4 and 1

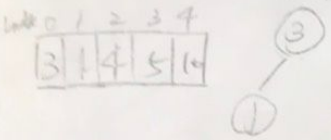


Step 2-8 remove the node: last node First node and delete last node

Swap 3 and 4 ⇒



Step 2-9 remove the node

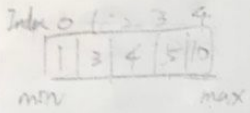


Heap

swap first node and last node

and delete the last node

Step 2-10 Only one element left in the heap, Algorithm ends.



Heap