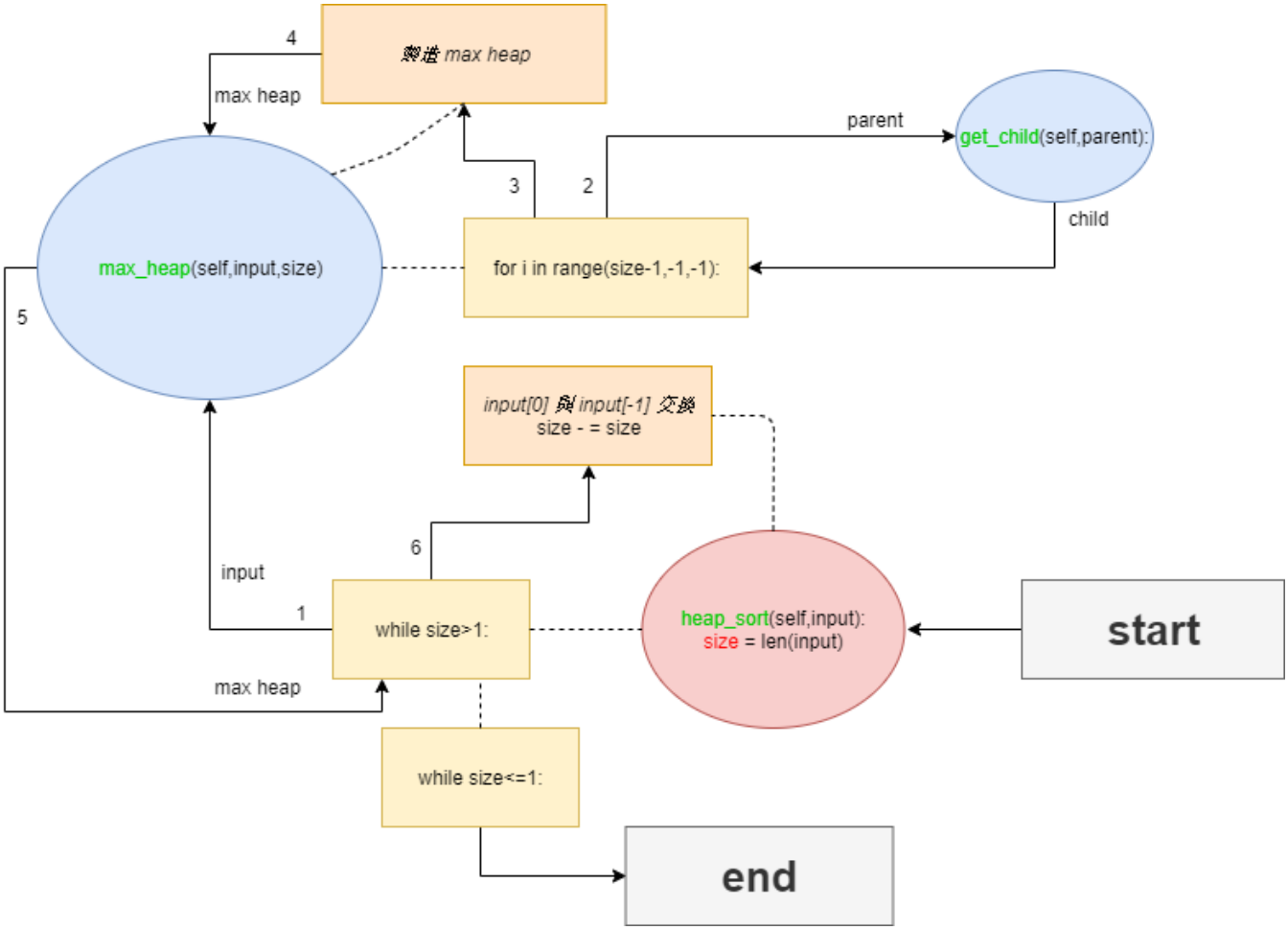
[HW2] Heap Sort

流程圖(打完 code 才畫的)



學習歷程

- 參考網站
  - hackerearth (https://www.hackerearth.com/zh/practice/algorithms/sorting/heap-sort/tutorial/) >>> 主要看流程圖，但不太明白它的array 為何要空一個 index[0]。
  - geeksforgeeks (https://www.geeksforgeeks.org/heap-sort/) >>> 簡單說明，有影片，有範例程式碼(但是我先自己試試看)

Try

在看完 Heap Sort 的影片和一些 Heap Sort 的基本架構後，我想要先試著寫寫看，於是我造著在 geeksforgeeks 裡面提到三個步驟:

- **step 1** : 將 input 的 array 依造 heap 讀資料結構處理成 max heap 的形式。
- **step 2** : 這時候 root 值為 Max。root 與最後一個作交換 ， 然後把 heap 的大小-1(以 array 的角度，就是將限制 haep 的index往左移，將以排序值排除在 heap 外。
- **step 3** : 重複上面的步驟直到 heap 只剩一個 element 。

>>首先我要將 heap 中 父節點 & 左右子節點 的關係作出來

假設index 從0開始，父節點在 index=a 的位置則，Heap 與 array 的 index 有以下關係:

- 左子節點 = 2a + 1
- 右子節點 = 2a + 2

所以，我先弄一個可以拿到子節點index:

In [7]:

```
def get_child(parent):
        return {'left':2*parent+1,'right':2*parent+2}
```

In [8]:

```
# 測試一下
print('0 的左子樹: {}'.format(get_child(0)['left']))
print('0 的右子樹: {}'.format(get_child(0)['right']))
```

```
0 的左子樹: 1
0 的右子樹: 2
```

>>接著我要將 input 的 array，轉換成 max heap 的形式

透過一個迴圈遍歷input，確保每個父節點皆大於他的子節點，假如他有子節點的話。

In [9]:

```
def max_heap(input_):
    left = 0 #預設 left
    right = 0 #預設 right
    for i in range(len(input_)):
        child = get_child(i)
        left = child['left']
        right = child['right']
        if (len(input_) > left)&(len(input_)>right): # left & right 兩者的 index 沒超出
 array
            #print(i)
            if (input_[i]>input_[left]) & (input_[i]>input_[right]): # 當負節點最大時，直
接跳過
                continue
            elif (input_[right]<input_[left])&(input_[left]>input_[i]): #當左節點最大，將
其與父節點交換。
                input_[i],input_[left] = input_[left],input_[i]
            else:
                input_[i],input_[right] = input_[left],input_[right]
```

In [10]:

```
a=[1,5,3,7,2]
max_heap(a)
a
```

Out[10]:

```
[5, 7, 3, 1, 2]
```

跟原本想的不太一樣...

Try again

之後我發現，我這樣的做法在同一層可行，但是如果跨兩層就有問題了。因為有可能極大值在 heap 的偏下面沒辦法一次就換到 root，必須重最下面開始確保父節點已經大於子節點 於是我換成從最後一層開始排，也就是倒著遍歷。

In [15]:

```
def max_heap(input_,size):
    for i in range(size-1,-1,-1):
        child = get_child(i)
        left = child['left']
        right = child['right']
        #print(i,left,right) 檢查用

        if (size<=left): # left 的 index 超出 array
            continue
        elif (size<right): # right 的 index 超出 array，但是left 沒有
            if input_[left]>input_[i]:
                input_[i],input_[left] = input_[left],input_[i]
        else:
            if (input_[i]>input_[left]) & (input_[i]>input_[right]): # 當負節點最大時，直接跳過
                continue
            elif (input_[right]<input_[left])&(input_[left]>input_[i]): #當左節點最大，將其與父節點交換。
                input_[i],input_[left] = input_[left],input_[i]
            else:
                input_[i],input_[right] = input_[right],input_[i]
```

In [16]:

```
a=[1,5,3,7,2]
max_heap(a,len(a))
a
```

Out[16]:

```
[7, 1, 3, 5, 2]
```

In [17]:

```
a=[2,1,3,5]
max_heap(a,len(a))
a
```

```
---------------------------------------------------------------------------
-
IndexError                                Traceback (most recent call las
t)
<ipython-input-17-082f17d13321> in <module>
      1 a=[2,1,3,5]
----> 2 max_heap(a,len(a))
      3 a

<ipython-input-15-53f142e200dc> in max_heap(input_, size)
     12                 input_[i],input_[left] = input_[left],input_[i]
     13         else:
---> 14             if (input_[i]>input_[left]) & (input_[i]>input_[right]
): # 當負節點最大時，直接跳過
     15                 continue
     16             elif (input_[right]<input_[left])&(input_[left]>input_
[i]): #當左節點最大，將其與父節點交換。

IndexError: list index out of range
```

Try again and again

第一次對了，可是試第二個較短的 array 居然錯了!!!

後來我發現是在 `elif` 那邊條件少一個 `=` ，導致 heap 的最尾端只有左子節點的話會 miss 掉跑到else 去。

In [20]:

```
def max_heap(input_,size):
    for i in range(size-1,-1,-1):
        child = get_child(i)
        left = child['left']
        right = child['right']
        #print(i,left,right) 檢查用

        if (size<=left): # left 的 index 超出 array
            continue
        elif (size<=right): # right 的 index 超出 array，但是left 沒有
            if input_[left]>input_[i]:
                input_[i],input_[left] = input_[left],input_[i]
        else:
            if (input_[i]>input_[left]) & (input_[i]>input_[right]): # 當負節點最大時，直
接跳過
                continue
            elif (input_[right]<input_[left])&(input_[left]>input_[i]): #當左節點最大，將
其與父節點交換。
                input_[i],input_[left] = input_[left],input_[i]
            else:
                input_[i],input_[right] = input_[right],input_[i]
```

In [21]:

```
a=[1,5,3,7,2]
max_heap(a,len(a))
a
```

Out[21]:

```
[7, 1, 3, 5, 2]
```

In [22]:

```
a=[2,1,3,5]
max_heap(a,len(a))
a
```

Out[22]:

```
[5, 2, 3, 1]
```

In [23]:

```
a=[1,2,3]
max_heap(a,len(a))
a
```

Out[23]:

```
[3, 2, 1]
```

In [24]:

```
a=[1,2]
max_heap(a,len(a))
a
```

Out[24]:

```
[2, 1]
```

終於成功了!!! 中間因為邏輯不太對，改了好幾次。怕後面還有bug 所以我多試幾次。

我覺得之後會需要限制 heap， 所以我將 `len(input)` 換成可變動的 `size` ， 也就是呼叫heapsort 我可以把長度限制在 size下。

>> 緊接著就是把已經為最大值的 root 掉到最後面，然後限制 heap 的大小(其實已排序的值還在array，只是限制了index)

In [25]:

```
def heap_sort(input):
    size = len(input)
    while size>1:
        max_heap(input,size)
        input[0],input[size-1] = input[size-1],input[0]
        size-=1
```

In [26]:

```
input = [1,5,3,7,2]
heap_sort(input)
input
```

Out[26]:

```
[1, 2, 3, 5, 7]
```

這邊比較容易，一次就達成了!!!

>> 最後把所有零件都組起來~ 在寫成 class 的形式

In [27]:

```
class Solution(object):
    def get_child(self,parent):
            return {'left':2*parent+1,'right':2*parent+2}
    def max_heap(self,input_,size):
        for i in range(size-1,-1,-1):
            child = self.get_child(i)
            left = child['left']
            right = child['right']
            #print(i,left,right) 檢查用

            if (size<=left): # left 的 index 超出 array
                continue
            elif (size<=right): # right 的 index 超出 array，但是left 沒有
                if input_[left]>input_[i]:
                    input_[i],input_[left] = input_[left],input_[i]
            else:
                if (input_[i]>input_[left]) & (input_[i]>input_[right]): # 當負節點最大時，直接跳過
                    continue
                elif (input_[right]<input_[left])&(input_[left]>input_[i]): #當左節點最大，將其與父節點交換。
                    input_[i],input_[left] = input_[left],input_[i]
                else:
                    input_[i],input_[right] = input_[right],input_[i]
    def heap_sort(self,input):
        size = len(input)
        while size>1:
            self.max_heap(input,size)
            input[0],input[size-1] = input[size-1],input[0]
            size-=1
        return input
```

Test

In [28]:

```
import random

for _ in range(10):
    l = random.randint(2,30)
    array = [random.randint(-50,100) for _ in range(l)]
    print('input:',array)
    out=Solution().heap_sort(array)
    print('sorted:',out,'\n')
```

```
input: [76, 89, 74, 83, -45, 70, -26, 56, 99, 74, 50, 80, 85, -6, 8, 50, 5
9, 37, 63, -48, 86, -6, -2, -41, 34, 88, 92]
sorted: [-48, -45, -41, -26, -6, -6, -2, 8, 34, 37, 50, 50, 56, 59, 63, 7
0, 74, 74, 76, 80, 83, 85, 86, 88, 89, 92, 99]

input: [-11, 23]
sorted: [-11, 23]

input: [49, 53, 69, 86, 43, -17, -25, 38, 91, -11, 59, 46, 80, 42, 15, -2
2, -20, 99, 93, -36, -2, 1, 64, 58, -2, 81, -3, -38, -47, -23]
sorted: [-47, -38, -36, -25, -23, -22, -20, -17, -11, -3, -2, -2, 1, 15, 3
8, 42, 43, 46, 49, 53, 58, 59, 64, 69, 80, 81, 86, 91, 93, 99]

input: [77, 78, 35, 70, 31, 53, -38, -9, 56, 79, 36, 90, -34, -35, 99, 22,
46, 52, 20, 27, -23, 61, -45, -45, -22, 9, 93, 72, 1, 22]
sorted: [-45, -45, -38, -35, -34, -23, -22, -9, 1, 9, 20, 22, 22, 27, 31,
35, 36, 46, 52, 53, 56, 61, 70, 72, 77, 78, 79, 90, 93, 99]

input: [80, -44, 54, -33, -15, -4, 56, 40, -36, 90, -49, 82]
sorted: [-49, -44, -36, -33, -15, -4, 40, 54, 56, 80, 82, 90]

input: [-43, 83, 72, 100, 34, 69, 38, 6, 76, 10, 66, 32, 42, -34, 100, -3
6]
sorted: [-43, -36, -34, 6, 10, 32, 34, 38, 42, 66, 69, 72, 76, 83, 100, 10
0]

input: [41, 50, -17, -41, 92, -11, -4, 20, 37, 70, 41, -46, -30, 3, 11, 5
3, 21, 30, 16, -24, 45, -23, -19, 27, 27]
sorted: [-46, -41, -30, -24, -23, -19, -17, -11, -4, 3, 16, 20, 27, 27, 2
1, 30, 37, 41, 41, 11, 45, 50, 53, 70, 92]

input: [52, 9, 93, 0, 59, 69, 70, -12, 53, 86, 20, -39, 10, -19, 23, 70, -
28, -1, 92, 93, 88, 86, -48, 6, -1, 90, 59, 79, -30, -17]
sorted: [-48, -39, -30, -28, -19, -17, -12, -1, -1, 0, 6, 9, 10, 20, 23, 5
2, 53, 59, 59, 69, 70, 70, 86, 86, 79, 88, 90, 92, 93, 93]

input: [91, 89, -47]
sorted: [-47, 89, 91]

input: [55, 9, 11, -11, 78, 99, 58, 4, -31, -19, 0, 85, -8, -49, 35, 41, -
6, -11, -19]
sorted: [-19, -19, -49, -11, -11, -31, -8, -6, 0, 4, 9, 11, 35, 41, 55, 5
8, 78, 85, 99]
```