Home work #2 B10815057 廖聖郝

1. Thanos Finger Snap

BST tree 的節點class，包含父節點指標與左右小孩的指標，並重載了一系列的大於小於符號，方便之後程式撰寫，因為整棵樹的節點都是動態配置記憶體出來的，所以用一個function來遞迴釋放記憶體。



接下來是BST tree的class，成員有根節點指標，與插入、刪除function…，為了保持物件封裝性，所以在class的public處都只有簡單的介面與防呆，真正的實作會放在private裡(重載函式)。

insert 資料進去BST內有一個點要注意的就是root為null時必須配置記憶體，而不能直接呼叫遞迴insert。



遞迴insert，這個function不能只傳入指標，而是要傳入指標的指標，否則配置記憶體時給予到的指標會是函式呼叫產生的新指標，而不是我們想要的節點內的指標。

根據BST的規則，新資料較小，就往左走，較大反之，一直走到leaf node後insert在leaf node之下，所以當遞迴走到nullptr時，就配置記憶體，產生新節點。



以下是幾個之後會用到的函式，都是用遞迴實作，原理已寫成註解。



前面的void erase(T data) { erase(search(data)); }，先使用search找到指標，然後交給下面的erase function處理。

刪除節點可分為2種情形處理:

1. 0或1個小孩，把替代指標(replacement)設為存在的小孩或null(0個小孩時)，判斷是否為root(沒爸爸)，是就特別處理後直接結束，一般狀況就將父節點指向自己的指標更新為替代指標，若小孩存在，就把小孩的父節點指標設為自己的爸爸。
2. 2邊都有小孩，就取左子樹中最大值，替代自己原本的值，然後刪除那個節點(遞迴)。



用一條水平線切開BST樹，切後的節點數要最接近原本的一半，一開始先建立儲存每層數量的陣列layer\_count，然後用遞迴function : get\_layer\_number\_of\_node，去找到每層的節點數，接下來，跑過每種水平線，算出哪條水平線最接近我們要的，找到後水平線後，建立一個森林，存放所有切開後下半部產生的樹，我把切開的實作與建立森林寫在同一個遞迴function: cut\_layer\_and\_create\_forest，做完後，輸出樹切開後的中序，與森林中每棵樹的前序，最後再將森林內的每棵樹釋放記憶體。



樹高函式:



遞迴計算BST樹中每層的節點數，結果存到layer\_count裡



切割與建立森林函式，會先遞迴到要切割的地方，指標存入森林，然後更改父節點內指到自己的指標為null(切除)



前中後，三序的差異就只是遞迴與輸出的先後順序不同，因為最後一個輸出不能加上空白，但這件事有點難在遞迴裡做到，所以我使用讓每個遞迴可共用的static變數，來記錄現在是不是第一次輸出，如果是就不在輸出data前加上空白。



處理輸入與主流程的main函式:



1. 2-3 Tree

因為2-3樹就只是order為3的B樹，所以我直接實作出B樹，以增加未來的擴充性，以下是B樹節點的class，有儲存父節點指標的father、child小孩指標陣列、data資料陣列、data\_num目前節點內資料數量，2個主要的陣列都多了一個位置，這是因為爆掉時的大小會比最大空間還多1格，有了這個空間，爆掉時才可以更好的整頓節點，不然會很難實作。



main函式，template的第二個參數填order，因為是2-3樹，所以填3



B樹的insert都是先插到leaf node裡，如果超過最大數量就會切割node，並將中間值往上丟，所以我實作了2種insert funciton，一種是leaf\_insert，專門處理新資料的插入，這個函式會遞迴的往正確的leaf node走，最後呼叫另一個insert function插入到leaf node裡，這個insert是對某個node去做插入資料用的，node爆掉時也會使用到。



接下來這個遞迴函式就是對node插入資料，第二個參數是在節點爆掉時產生的新節點指標，會這樣做是因為中間值與新節點指標是相鄰的，一起插入比較方便，這個函式中，會需要處理一個特例，也就是root node爆掉時產生的新root node， 因為這時候裡面甚麼資料都沒有，所以要特別去處理，還有一個狀況就是新資料大於現有的所有資料，此時就要插到最後面，插入完資料後就檢查是否爆掉，如果爆掉的話就找爸爸來分割，沒爸爸(root node)，就產生一個新root node，這種狀況就是上面提到的特例，下面就是這個function的實作程式碼。



以下是呼叫父節點分割爆掉子節點的function，記憶體配置出一個新節點，然後把爆掉節點右半邊的資料與指標搬到新節點裡，被搬動的小孩指標也需要將此小孩的父節點指標設為新節點，因為指標比資料多一個，所以會額外處理最後一個指標的小孩，新節點的資料數就是剛剛搬的數量，最後將爆掉節點的中間值插入到自己裡面，遞迴下去。



middle回傳中間值，output輸出這個節點的資料(最後不加空白)



以下是幾個內建在node class裡的funciton，find\_root回傳root node指標，只要一直往上走就能找到，height回傳節點的高度，要判斷是不是leaf node只需判斷第一個小孩存不存在就好，del是釋放記憶體用的function，會遞迴刪除所有小孩，最後再刪除自己。

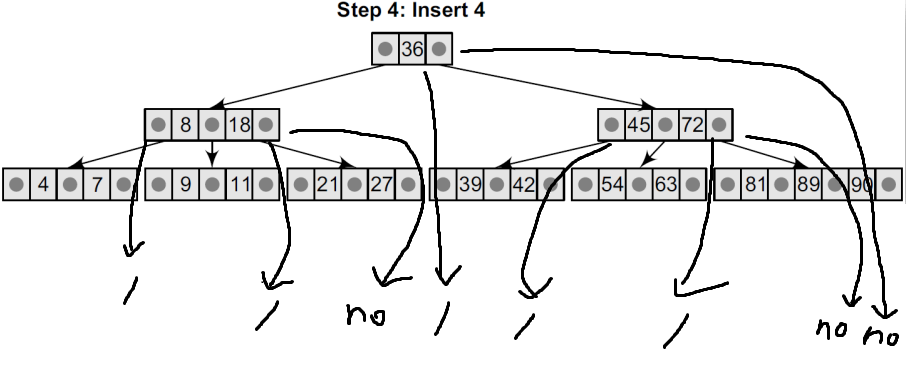


以下是B樹的class，有root node指標，insert function加入新資料會從這裡開始呼叫node內的leaf\_insert，加入新資料完後也要更新root node。



處理輸出的函式，先算出樹高，然後用for迴圈跑過每層，每層用output\_layer遞迴函式輸出該層所有node，每個遞迴結束後都要輸出**/**區隔，但最後一個遞迴則不用。

每個箭頭都是遞迴結束後的輸出，例如8、18節點輸出4、7後結束遞迴，然後輸出斜線，但輸出完21、27後則不輸出，因為是最後一個遞迴，但這節點並不是最該層最後一節點(81、89、90才是)，所以21、27節點與39、42節點之間必須有斜線輸出，這時候，上一層的遞迴就派上用場了，36節點遞迴到8、18節點完後就會輸出一個斜線，這個斜線恰好解決了剛剛的疑問，所以不管要輸出哪層，只要最後一個遞迴出去後不輸出斜線就可以成功。





Part B: 插入20, 45, 30, 50, 100, 70, 40, 10, 87到2-3樹裡

1. 插入20

由於此時root node是null，因此會執行這行:

if (root == nullptr) root = new B\_tree\_node<T, order>(new\_data);

樹內的資料:  data\_num == 1

1. 插入45

進入leaf\_insert root->leaf\_insert(45);

節點insert: insert(45,nullptr);

找到正確位置插入，45大於節點內所有資料，所以放到最後，資料數+1



檢查有無爆掉->沒有

樹內資料: 

1. 插入30

進入leaf\_insert root->leaf\_insert(30);

節點insert: insert(30,nullptr);

找到正確位置插入，用for迴圈跑，找到第一個大於30的資料=>data[1] 45 ，先把45以後(含45)的資料與指標往後搬，然後把30放到45原本的位置



此時節點內的資料:20,30,45

判斷是否爆掉->是if (data\_num == order) {//資料數 == order 代表node滿了，node爆掉

父節點指標是否為空(是否為root node)->是，new一個新爸爸出來，並指派自己成為第一個小孩

然後呼叫父節點分割爆掉的節點，分割後的資料數除以2



父節點分割子節點函式，new一個新節點，並將45與指標(現在是null)搬到新節點，最後將中間值30 insert到自己內部



在insert function內，新的root node屬於特例，特別處理它



樹內資料: 

1. 插入50

進入leaf\_insert root->leaf\_insert(50);

遞迴一次後到達leaf ndoe: 

leaf ndoe節點insert: insert(50,nullptr);

因為50大於45，所以放到最後面就好，該節點資料數+1



判斷是否爆掉->否

樹內資料: 

1. 插入100

跟前面的步驟一樣，遞迴到節點後，呼叫插入

插入完後節點內資料:45,50,100

判斷是否爆掉->是

是否為root node->否

呼叫父節點分割爆掉的節點，分割後的資料數除以2

父節點: new出節點，把100搬過去，50插入到節點。

判斷是否爆掉->否

樹內資料: 

1. 插入70

跟前面的步驟一樣，遞迴到節點後，呼叫插入

插入完後節點內資料:70,100

判斷是否爆掉->否

樹內資料: 

1. 插入40

跟前面的步驟一樣，遞迴到節點後，呼叫插入

插入完後節點內資料:40,45

判斷是否爆掉->否

樹內資料: 

1. 插入10

跟前面的步驟一樣，遞迴到節點後，呼叫插入

插入完後節點內資料:10,20

判斷是否爆掉->否

樹內資料: 

1. 插入87

跟前面的步驟一樣，遞迴到節點後，呼叫插入

插入完後節點內資料:70,87,100

判斷是否爆掉->是

是否為root node->否

呼叫父節點分割爆掉的節點，分割後的資料數除以2

父節點: new出節點，把100搬過去，87插入到節點。

插入完後節點內資料:30,50,87

判斷是否爆掉->是

是否為root node->是

new出新root node，並指派自己成為第一個小孩，呼叫新爸爸分割節點

新root node: new出新node，把87後(含)的指標與資料，搬到新node，50插到新root node，插入完後的新root node: 

判斷是否爆掉->否

樹內資料: 

1. 插入1

跟前面的步驟一樣，遞迴到節點後，呼叫插入

插入完後節點內資料:1,10,20

判斷是否爆掉->是

是否為root node->否

呼叫父節點分割爆掉的節點，分割後的資料數除以2

父節點: new出節點，把20搬過去，10插入到節點。

插入完後節點內資料:10,30

判斷是否爆掉->否

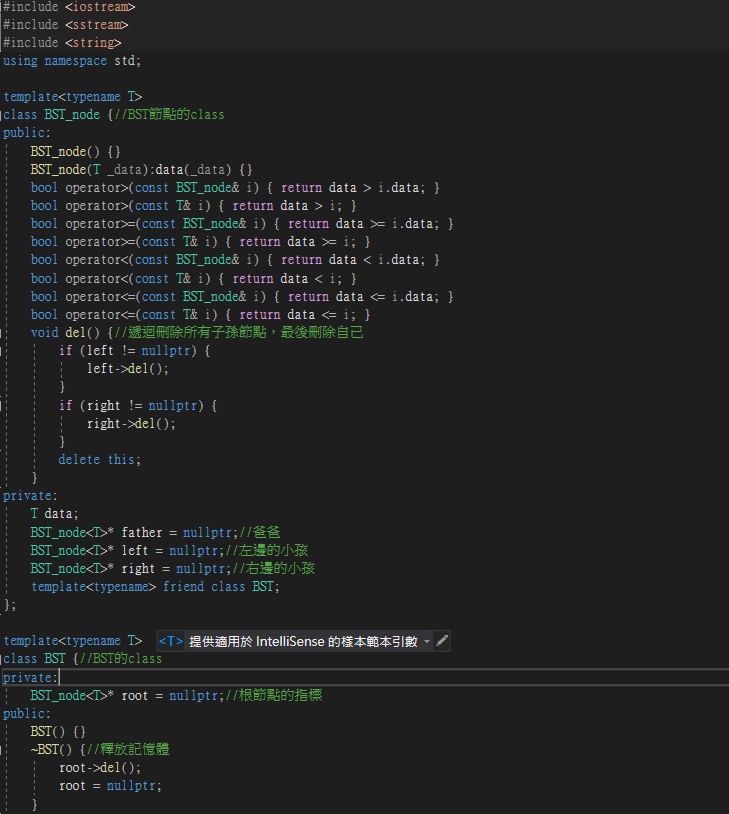
樹內資料: 

1. 程式碼

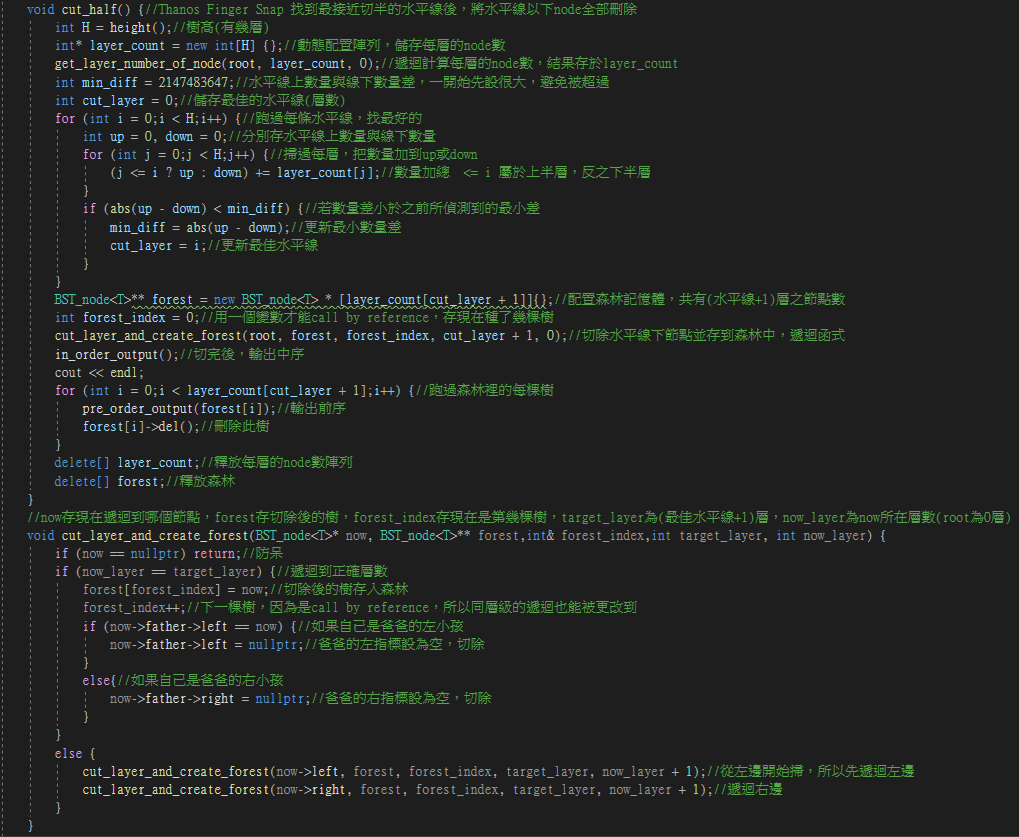
[2-1 Thanos Finger Snap gist](https://gist.github.com/frakw/903cab8f10d66b2c0bf0083fcf971b42) [imgur圖片](https://imgur.com/a/kwKQOot)

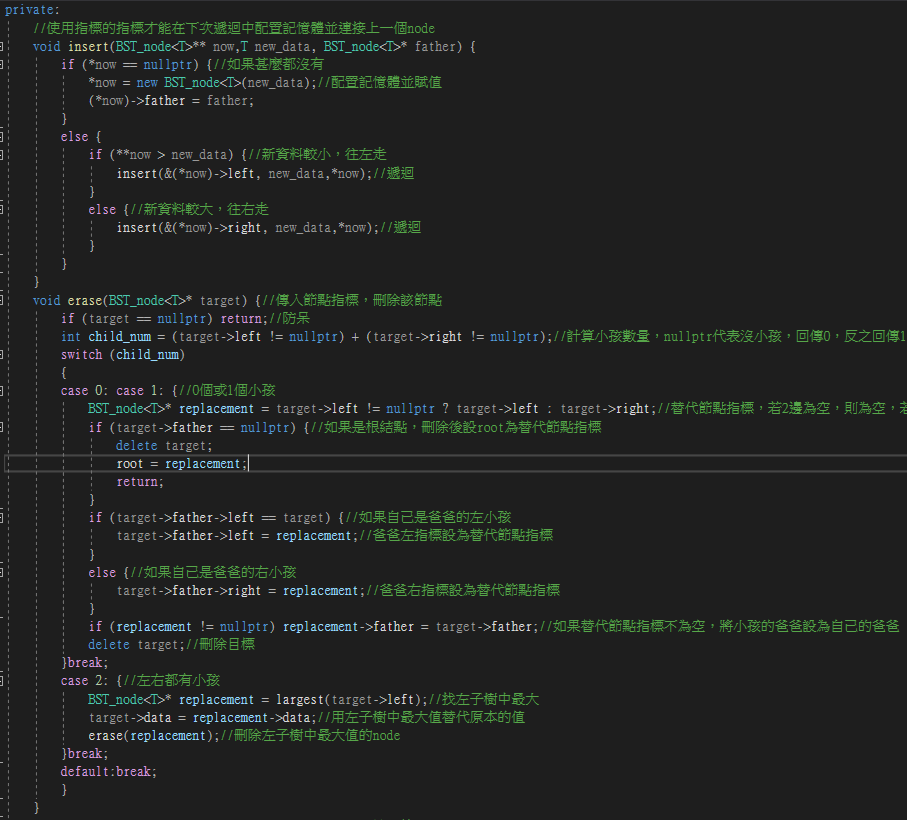
[2-2 2-3 Tree gist](https://gist.github.com/frakw/d81997657cadc25db1fdc750b699ca51) [imgur圖片](https://imgur.com/a/KOptpXJ)

2-1:

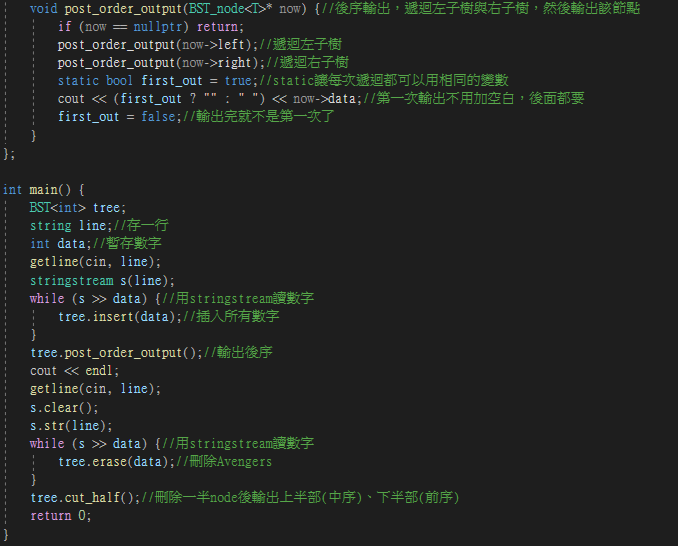












2-2:

