資結 練習五 報告

(一) 簡介:

任務一 : 建立二元搜尋樹

一開始判斷是否建過二元樹，如果有就先解構，之後用 fstream 將檔案讀進來。接著要畢業生數/學校名稱的二元樹，宣告三個節點 a、b 與 insert_node(要插入的節點)，先設定 a 為 root，只要 a 不為空 ，a 的值會存於 b，而且會進入判斷式，判斷 a 節點的畢業生人數是否大於 insert_node 的畢業生人數，或是 a 節點的學校名稱比較短或者筆畫較少，(1)如果是，代表為 leftchild 則往左走，(2)若不是則往右走。一直到 a 為空找到 insert_node 的插入位置。接著設 b 為 insert_node 的 parent，然後判斷(1)若 b 為空代表 insert_node 沒有 parent，則 insert_node 為 root，(2)insert_node 的畢業生人數小於 b 的，則 insert_node 為 b 的 leftchild，反之則為 rightchild，用這樣的方法以遞迴方式建立畢業生數/學校名稱的二元樹。再來要找樹高，我們用一個變數來找，(1)若變數為 0 則樹高為 0 (2)若不為 0，找左右子節點的樹高，看哪邊較高取之作為共同子節點樹高，用遞迴的方式找尋每個子節點的樹高，最後再加一(root 也要算)，就能找到總樹高。

任務二 : 使用二元搜尋樹

(1)找出鍵值不低於輸入的資料

先確認是否有任務一的二元樹，有則輸入一個正數 num，之後新增一個節點指

向 root 作為起點，如果節點不為空，判斷在此節點的畢業生人數是否大於或等於輸入的 num，是則依照格式印出。繼續判斷節點的畢業生人數是否小於輸入的 num，如果小於則往右走且回到此迴圈的開頭繼續遞迴，如果大於或等於則往左走回到迴圈的開頭再往右走回到迴圈的開頭繼續遞迴。

(2)找出名稱完全相同的資料

確認是否有任務一的二元樹，有則讓使用者輸入一個字串，之後新增一個節點指向root作為起點，如果節點不為空則判斷在此節點的字串是否等於輸入的字串，如果是則依照格式印出，再來判斷此節點的字串是否比使用者輸入的字串短或筆畫較少，如果是則往右走且回到此迴圈的開頭繼續遞迴，否則往左走回到迴圈的開頭遞迴再往右走回到迴圈的開頭遞迴。

任務三 ： 刪除二元搜尋樹節點及資料

確認是否有任務一的二元樹，有則讓使用者輸入一個數，之後新增一個節點指向 root 作為起點，(1)節點為空，直接刪除此 list 的資料 (2)節點不為空則判斷節點的畢業生人數是否小於等於使用者輸入的數，如果條件成立則將刪除的資料依照格式印出。接著進入刪除環節。刪除環節結束後，再新增一個節點指向 root 作為起點，設節點的左右小孩都是空的，如果此節點不為空，判斷此節點的學校名稱是否等於使用者輸入的字串，如果是再判斷系所、學生數、老師數、畢業人數是否也相等，(1)是就進入刪除環節，然後刪除此 list 的資料 (2)不是則判斷節點

的學校名稱是否比使用者輸入的短或筆畫較少，是的話往右走並回到迴圈開頭繼續遞迴，不是則往左走並回到迴圈開頭繼續遞迴再往右走回到迴圈的開頭遞迴。結束遞迴後刪除此 list 的資料。最後判斷使用者輸入的值是否大於或等於節點的畢業生人數，如果否則往左走後再遞迴，是就往右走並回到開頭遞迴再往左走回到開頭繼續遞迴，之後判斷使用者輸入的值是否大於或等於節點的畢業生人數且節點小孩是否都不為空，是的話回到開頭繼續遞迴，否則往左走後再遞迴。接著依任務一的方法印出樹高。

刪除環節 ：

新增 delete_node 與兩個節點 a、b，a 暫放要被刪除的節點的 child，b 為真正要被刪除的節點，接著判斷要刪的節點是否為 leaf 或是否只有一個 child，是的話就把 b 設為 delete_node，若不是則判斷 b 的 leftchild 是否為空，不是的話再去看 leftchild 的 rightchild 是否為空，若一直不是，就一直往右走。接著判斷如果 b 的 leftchild 不為空，則將 a 設成 b 的小孩，如果左邊為空則設成 rightchild 再來判斷 b 的 parent 是否為空，如果為空表示 b 為 root，則用 a 取代 b 當 root，如果 b 不是 root，判斷 b 為 leftchild 還是 rightchild 之後讓 a 取代 b。再來判定是否為 delete_node，如果 b 為 delete_node 代表有兩個 child，則讓 b 這個 delete_node 的小孩的資料取代 delete_node 的資料，最後釋放 b 位置的記憶體。

心得:

任務一讀檔建動態陣列的部分由於前面的練習都使用過很多次因此已經相當熟練所以沒什麼問題，接著將 vector 裡的資料建二元樹按照老師影片裏面詳細的說明去建立因此也沒遇到甚麼大問題，主要是新增節點進去的部分要特別釐清那個加入節點跟尋找節點的順序跟互相連接的順序，花了一點時間找 bug，不過後來也順利解決，到任務二的時候，由於前面先釐清過了因此，任務二很快便知道要如何去尋找要找到的節點。起初，沒看清楚題目，差點用走訪每個節點來找節點，後來又認真看過一次題目才發現! 搜尋資料禁止使用遍歷每個節點的暴力法，必須藉由比較鍵值來避免走訪過所有節點!因此趕快改掉，改成比較鍵值，如果比數字小就不往左走了。

至於任務三的部分，前一天有猜到應該會是 delete 節點，因此前一天便寫好了，但當天看了題目開始修改過後，發現雖然數量能跟老師一樣但順序不管怎麼改仍然不一樣，花了很多時間 debug。

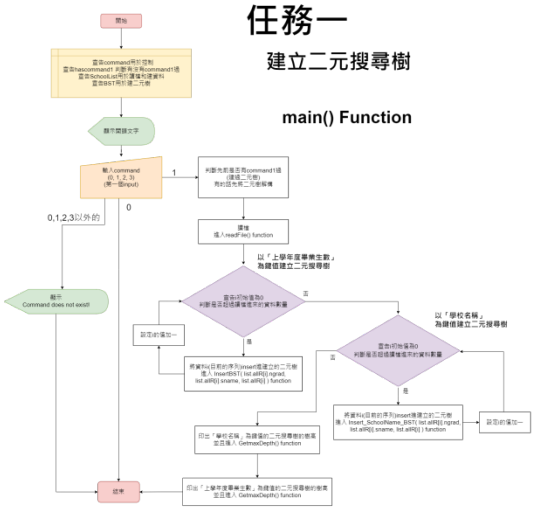
(二) 圖示:

任務一:

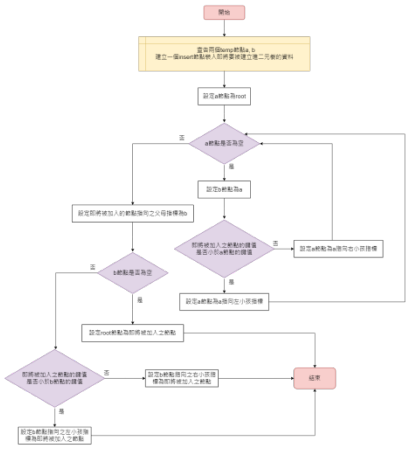
任務一

建立二元搜尋樹

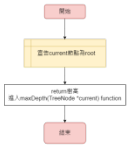
main() Function



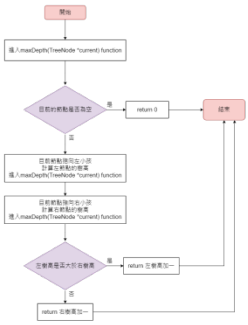
以「上學年度畢業生數」為鍵值建立二元搜尋樹
InsertBST(int key, string stringkey, SchoolList::schoolType data) Function



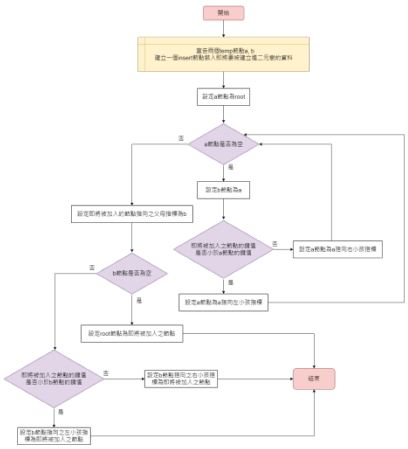
GetmaxDepth() Function



maxDepth(TreeNode *current) Function

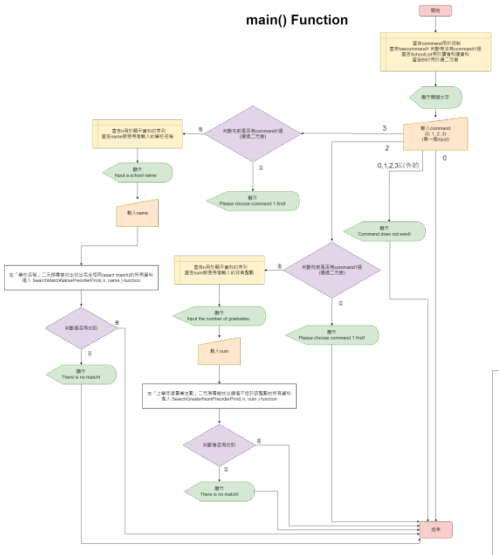


以「學校名稱」為鍵值建立二元搜尋樹
Insert_SchoolName_BST(int key, string stringkey, SchoolList::schoolType data) Function



任務二:

main() Function



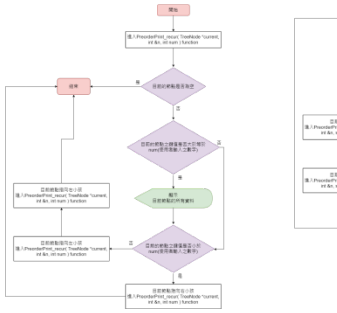
任務二

使用二元搜尋樹

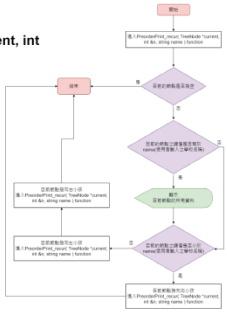
SearchMatchNamePreorderPrint(int &n, string name) Function

SearchGreaterNumPreorderPrint(int &n, int num) Function

PreorderPrint_recur(TreeNode *current, int &n, int num) Function



PreorderPrint_recur(TreeNode *current, int &n, string name) Function



(三) 解說

影片連結: