資結 練習六 報告

(一) 簡介:

任務一 : 建立二元搜尋樹

確認是否有建過二元樹，如果有就先刪除，之後用 fstream 讀檔，如果檔案有成功打開，把資料的六個欄位一行行讀進來並把它們存於動態陣列 vector 裡，如果檔案是空的就顯示 Get nothing from the file，接著依照格式一行行印出。再來要以「生命值 HP」為鍵值建立二元搜尋樹，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 HP 是否大於 insert_node 的 HP，(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 HP 小於 b 的，則 insert_node 為 b 的 leftchild，反之則為 rightchild。找尋樹高的方式，我們是用一個變數來找，(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。最後要找最左邊跟最右邊的節點，首先新增一個節點指向 root 當作起點，若節點的左子節點不為空，就往左走，一直走到左子節點為空，(找最右邊節點也是一樣的方法)，然後判斷我現在的節點是否為空，若不是就照格式印出所在節點的資訊。

任務二 ： 建立最大堆積

確認是否有任務一的二元樹，如果有則開始建 Heap，設一個最大值紀錄 children 比較大的數值，再設一個變數起始設為長度的 (1 / 2 -1 ) 然後去跑 (HeapRebuid 的演算法，節點有一半為 leaf 可忽略)，先判斷此節點是否有 left child，有的話再判斷此 left child 的 HP 是否大於最大值，如果有就將 left child 的 HP 設為最大值，並將此子節點設定為要交換的位置，再來判斷 right child，方式跟 left child 一樣，如果節點有 right child 且 HP 大於最大值，就設為最大值然與要交換的位置。最後判斷最大值(小孩的 HP)是否大於自己的 HP，有就照之前設定好的位置將他們做交換，之後依照格式將 Heap 一行行印出。樹高的方面我們用 ceil()函數與樹高的性質 log(n+1)找出。最後要找最左邊與底部的節點，找最左邊節點的方法是如果目前節點還有 left child 就往左走，一直走到沒有，找到後依照格式印出此節點資料。若要找底部的節點就直接帶入 Heap 的總長度 -1 的數，找到後依照格式印出。

任務三 ： 從最大堆積刪除樹根

確認是否有任務一的二元樹，如果有則判斷任務二的 Heap 是否為空，(1)如果是請使用者先輸入任務二，(2)如果 Heap 長度為一，則按格式印出且刪除最後一個元素。(3)若不是(1)與(2)則按格式印出，接著將 Heap 的最後一個元素丟到第一個，然後將最後的元素刪掉，然後將 Heap 按照任務二的方式重新整理成 MaxHeap。處理好後去判斷 Heap 是否為空，若不是就用任務二的方式印出

Heap 的內容、找出樹高、最左邊的節點與底部的節點的資料。

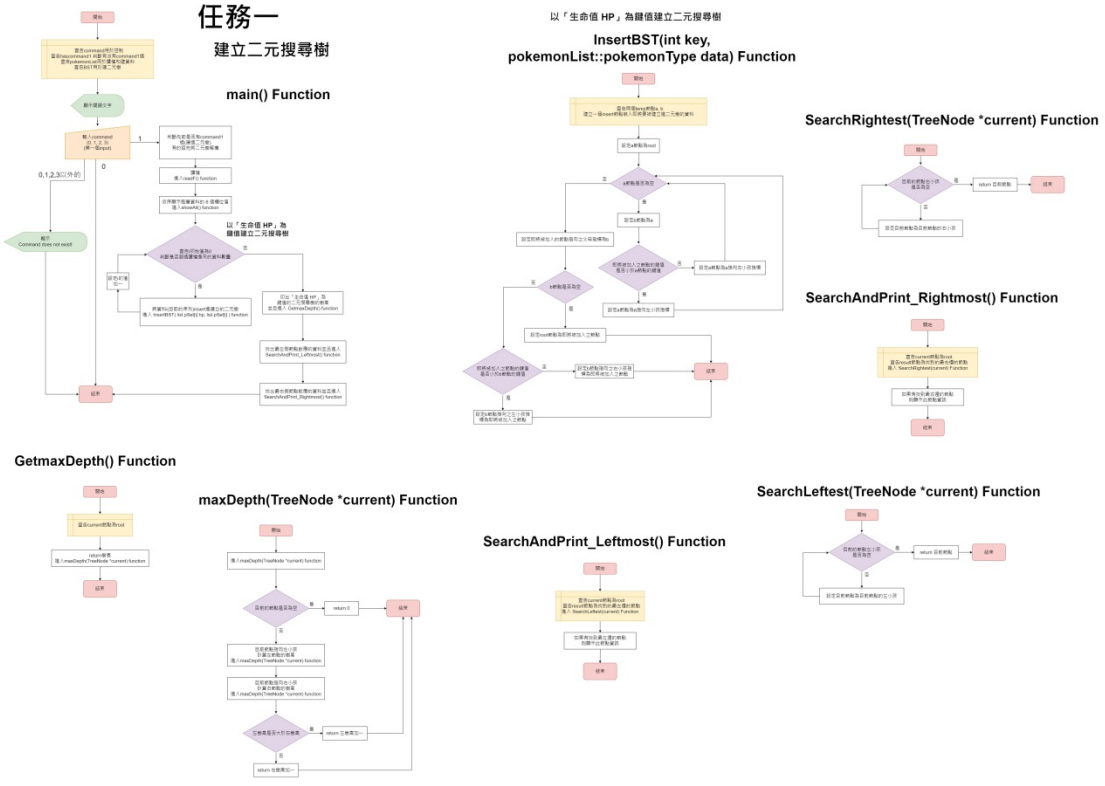
心得:

這次練習六可以說是我們這整個學期最順利的一份作業( 也可能是我們組員之間已經非常熟悉彼此，這次作業兩人都完成的非常有效率 )，當初看到題目還以為會相當難相當花時間，看完了老師的影片後便完成的非常的順利。

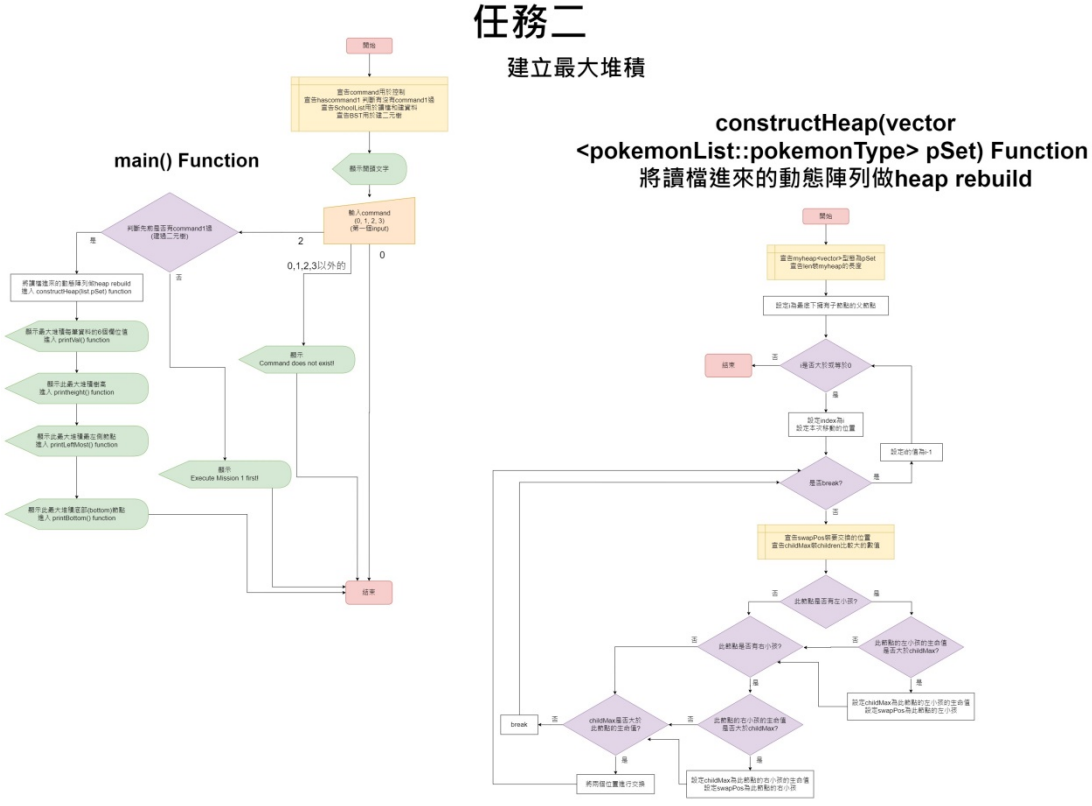
任務一的部分由於是建二元樹其實就是練習五的內容，除了有些地方需要稍微修改跟增加找最左邊節點跟最右邊節點部分之外，其他地方內容都差不多，因此任務一沒有問題的發生，而在寫任務二時，我們有點誤會題目的意思，以為是要把二元樹轉成 Heap，一直在思考要先將二元樹的哪個節點丟到 Heap 中，後來再仔細看了一次題目才發現用任務一的陣列就行了。清楚理解後，我們就很順利的把任務二寫出來。最後，任務三的部分，由於上次練習五的挑戰題老師以不一樣的方式刪除節點我耗時了很多時間研究，因此這次在前幾天也是先猜了一下挑戰題，想說大概是要做刪除，所以這次任務三也事先寫好了，很高興有猜到！因為有成功猜到，我們任務三完成的相當迅速！讓資料結構課程有一個完整的結尾！可喜可賀。

(二) 圖示:

任務一:


任務一
建立二元搜尋樹
main() Function
以「生命值 HP」為鍵值建立二元搜尋樹
InsertBST(int key, pokemonList::pokemonType data) Function
SearchRightest(TreeNode *current) Function
SearchAndPrint_Rightmost() Function
GetmaxDepth() Function
maxDepth(TreeNode *current) Function
SearchAndPrint_Leftmost() Function
SearchLeftest(TreeNode *current) Function

任務二:


任務二
建立最大堆積
main() Function
constructHeap(vector <pokemonList::pokemonType> pSet) Function
將讀檔進來的動態陣列做heap rebuild

(三) 解說

## 3.解說影片