

F74076108_蔡秉睿_Hw6_Readme

1. screen shot:

The image displays two screenshots of a terminal window, likely from a Linux system, showing a file comparison process.

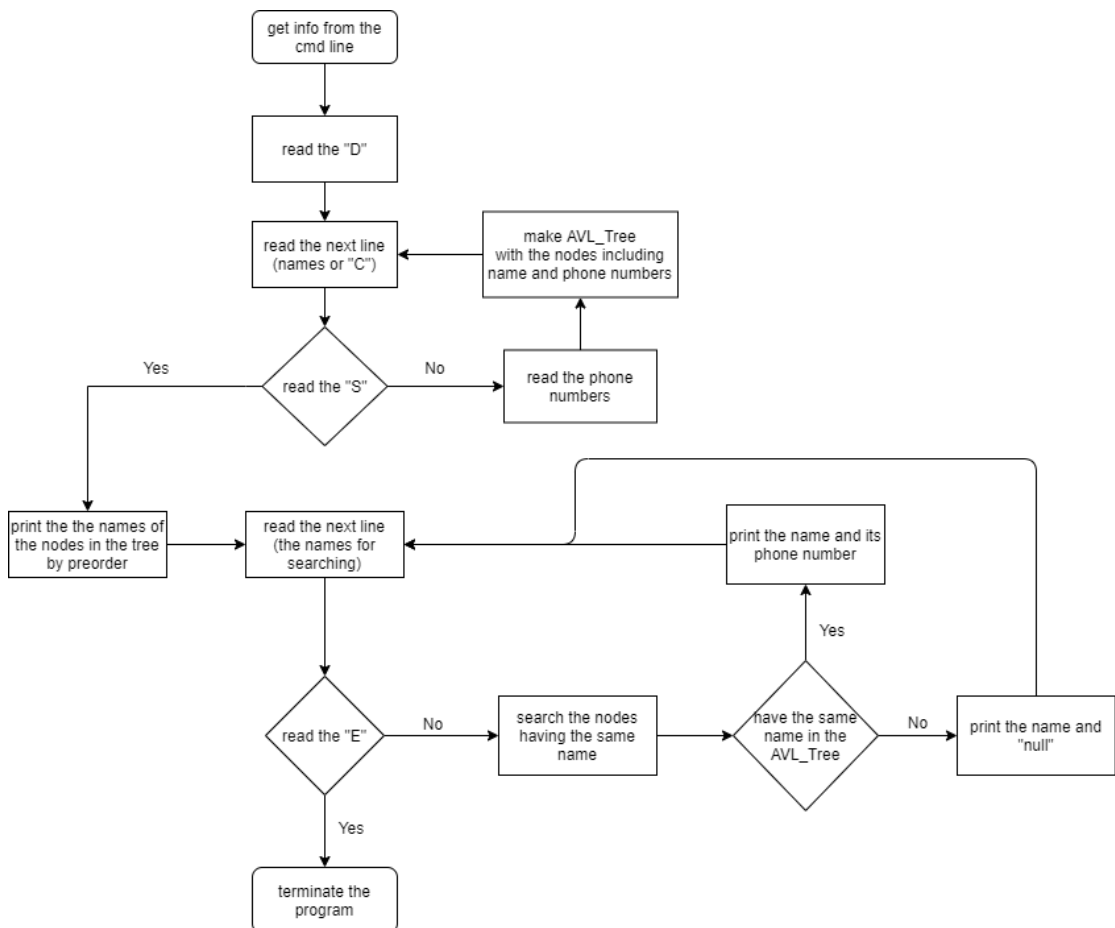
Top Screenshot:

- The terminal title bar shows tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", and "TERMINAL". The "TERMINAL" tab is active, and the window title is "1: vim".
- The terminal content shows a list of files: "David Bob Alice Charlie Paul Ruby", "Paul 0900000002", "Amy null", and "Ruby 0900000004".
- The cursor is positioned at the end of the "Ruby 0900000004" line.

Bottom Screenshot:

- The terminal title bar shows the same tabs, but the window title is now "1: bash".
- The terminal content shows the following commands and output:
 - `teddy@LAPTOP-MGLSTEMM:/mnt/c/Users/Asus/Desktop/寶結/Hw6_2020_11_19$./avl_tree < input.txt > output.txt`
 - `teddy@LAPTOP-MGLSTEMM:/mnt/c/Users/Asus/Desktop/寶結/Hw6_2020_11_19$ vim output.txt`
 - `teddy@LAPTOP-MGLSTEMM:/mnt/c/Users/Asus/Desktop/寶結/Hw6_2020_11_19$ diff output.txt output_windows.txt`
- The cursor is positioned at the end of the last command line.

2. program architecture:



3. program function:

- i. `node_t *newNode(char name[], char phone[]):`

新建一個 node，node 的 name 和 phone number 為讀入的值。

a. char name[]:

node 的 name

b. char phone[]:

node 的 phone number

return new_Node(回傳新的 node)

ii. int nodeHeight(int height_a, int height_b):

計算 node 的高度(比較 left、right subtree)。

a. int height_a:

left subtree 的 height

b. int height_b:

right_subtree 的 height

return 最大的 height (height_a or height_b)

iii. int height(node_t *root):

回傳當前 node 的 height；當 node == NULL，return 0。

一定要有這個 function 判斷是否為 NULL，不然會

segmentation fault。

a. node_t *root:

要取得高度的樹(node)

return root->height

iv. node_t *rightRotate(node_t *root):

向右旋轉要被移動的子樹。

a. node_t *root:

要被移動的子樹的 root 。

return 新的 root(l_sub => 原 root 的左子樹)

v. node_t *leftRotate(node_t *root):

向左旋轉要被移動的子樹 。

a. node_t *root:

要被移動的子樹的 root 。

return 新的 root(r_sub => 原 root 的右子樹)

vi. int getBF(node_t *root):

取得當前 node 的 balance factor 。

a. node_t *root:

要計算 bf 的 node 。

return bf(左子樹高 - 右子樹高)

vii. node_t *insert_node(node_t *node, char name[], char phone[]):

把新增的 node 插入樹中。一開始先用 Binary search tree

的方法插入，然後再判斷 bf 來決定需不需要 rotate 。

a. node_t *node:

要被插入新的 node 的樹的 root 。

b. char name[]:

新的 node 的 name 。

c. char phone[]:

新的 node 的 phone number 。

return 樹的 root

viii. void preorder(node_t *node):

依照 preorder 把 tree 的 name 印出來。

a. `node_t *node`: 要印出來的 tree 的 root。

ix. `void search(node_t *node, char name[])`:

搜尋樹中是否有和輸入的 name 一樣的 node，有的話連

同 node 的 phone number 一起印出來；沒有的話，

phone number 的位子印 null。

a. `node_t *node`: 要搜尋的樹的 root。

b. `char name[]`: 要比對的 name。

4. program design:

利用 scanf 讀取 D 到 C 之間的 names 和 phone numbers，然後利

用 AVL_Tree 的方式組成樹；接著在 C 到 E 之間的 names 用來搜

尋樹裡的 nodes 是否有一樣的 names，有的話 name 和 phone

number 一起印出來；沒有的話 name 和 “null”一起印出來。

這次作業困難的地方我覺得就是 AVL tree 本身，雖然我懂 avl tree

的演算法，在紙上也能畫出來，但是打成 code 就沒頭緒。最後

研究好久才打出一個雛形，結果執行時出現 segmentation fault，

搞到快崩潰，上網看了別人寫的才發現原來我 leaf node 是 null

的情況沒考慮到，leaf node 本身是 null，但是我卻直接用

`node->height` 來計算，才導致 segmentation fault。我覺得平衡樹

的 code 都有點難寫，希望之後用到會寫的比較順。