

1. Result screenshot

```
MINGW64/c/Users/user/desktop/Data Structure/HW4
user@LAPTOP-47I2GHM0 MINGW64 ~/desktop/Data Structure/HW4
$ cd
user@LAPTOP-47I2GHM0 MINGW64 ~/desktop/Data Structure/HW4
$ cd desktop
user@LAPTOP-47I2GHM0 MINGW64 ~/desktop/Data Structure
$ cd HW4
user@LAPTOP-47I2GHM0 MINGW64 ~/desktop/Data Structure/HW4
$ gcc -std=c11 ./hw4.c -o hw4
user@LAPTOP-47I2GHM0 MINGW64 ~/desktop/Data Structure/HW4
$ ./hw4.exe < input0_windows.txt > ans_output0_windows.txt
user@LAPTOP-47I2GHM0 MINGW64 ~/desktop/Data Structure/HW4
$ diff ./output0_windows.txt ./ans_output0_windows.txt
user@LAPTOP-47I2GHM0 MINGW64 ~/desktop/Data Structure/HW4
$
```

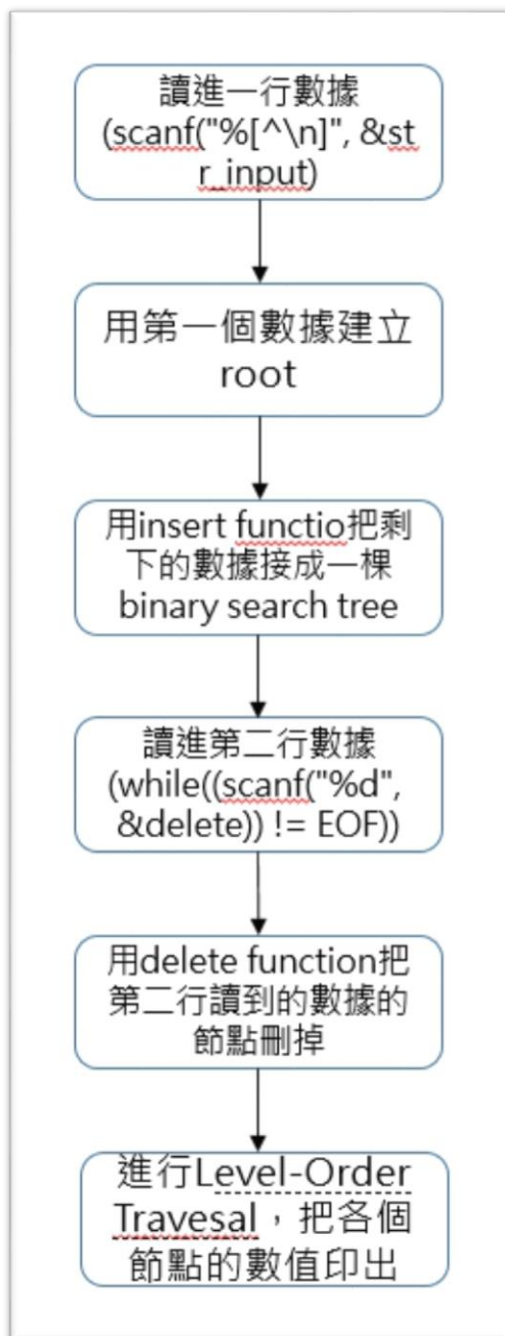
input0_windows.txt - 記事本

```
1 3 5 7 2 4
5 7
```

ans_output0_windows.txt - 記事本

```
1
3
2
4
```

2. Program structure



3. Program functions

(1)

`tree_pointer max(tree_pointer node)`:用來尋找存放整個樹最大值的
`node`

`node` – 用來不斷指向右邊的 `node`(最右邊最下面的 `node` 存放的值
即是整個數最大的值)

`return` – 整個樹最右邊最下面的 `node`

(2)

`tree_pointer insert(tree_pointer node, int data)`:用來新增 `node`

`node` – 如果此 `node` 為 `NULL` 則新增為新的 `node`，如果不是就
用來指向下一個 `node`(用 `data` 進行大小比對來看要放在右邊還左邊)

`data` – 要存放在 `node` 裡的值

`return` – 要新增或進行下一次比對的 `node`

(3)

`tree_pointer delete_node(tree_pointer node, int data)`:用來刪除
`node`

`node` – 用來尋找要刪除的 `node`

`data` – 存放在要刪除的 `node` 裡的值

`return` – 透過比對不斷回傳 `node`，最後找到要刪除的 `node`

(4)

`void traversal(tree_pointer root)`:用來把整棵樹印出來(Level-order)

`root` – 整棵樹的 `root`