ECE368 Project3 Report
Yinuo Li

	run_time	width	height	xcoord	ycoord
r6_flr.txt	0	1.100000e+01	1.500000e+01	9.000000e+00	0.000000e+00
r3_flr.txt	0	9.793506e+06	2.573651e+06	9.637800e+05	2.320255e+06
r4_flr.txt	0	9.653391e+06	1.018544e+07	7.987561e+06	3.441761e+06
r5_flr.txt	0	2.745456e+07	1.320092e+07	2.447639e+07	1.683258e+06

I used postorder to generate the width and height for every node.
I used preorder to generate the xcoord and ycoord for every node.

In postorder traversal , the root is visited after both subtrees . Postorder traversal is defined as follows ,
1. Traverse the left subtree in postorder .
2. Traverse the right subtree in postorder 3. Visit the root .
3. Visit the root .
Time complexity : O(n)
Space complexity : O(n)

In preorder traversal , each node is processed before (pre) either of its sub trees . This is the simplest traversal to understand . However , even though each node is processes before the subtrees , it still requires that some information must be maintained while moving down the tree . In the example above , the 1 is processed first , then the left sub tree followed by the right subtree .Therefore , processing must return to the right sub tree after finishing the processing left sub tree we must maintain root information . The obvious ADT for such information is a stack .Because of its "Last in First Out " Structure it is possible to get the information about the right subtrees back in the reverse order .
Preorder traversal is defined as follows ,
1. Visit the root.
2. Traverse to the left subtree in preorder.
3. Traverse the right subtree in preorder .
Time complexity :O(n)
Space complexity : O(n)

Therefore, for my algorithm,
Time complexity :O(n)
Space complexity : O(n)