Assessment 3

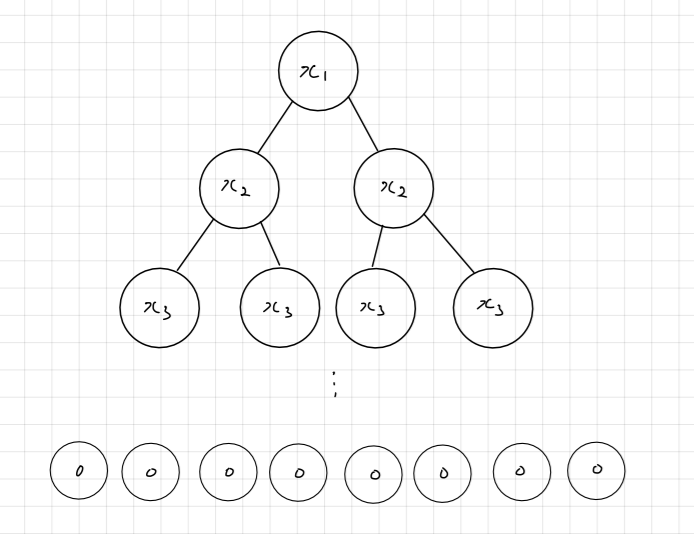
Kihyun Kim (<CID:01734683>)

ELEC40004 – Programming for Engineers

**Section1 – Algorithm**

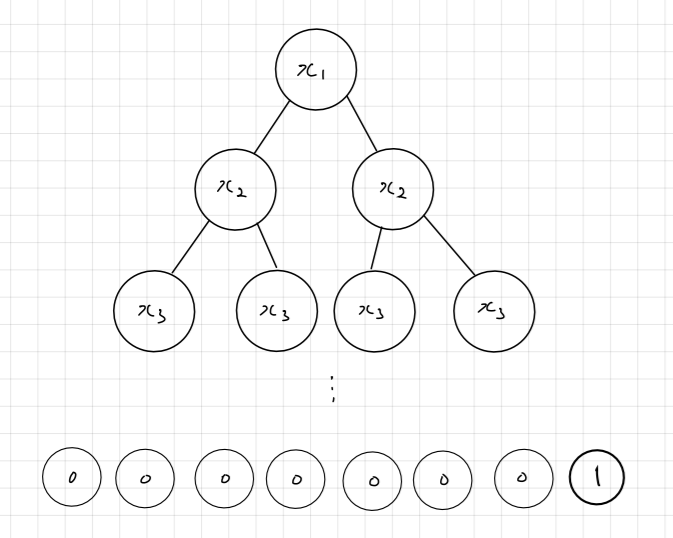
Constructing Tree

1. Constructing tree and filling each node as x1, x2, x3 and so on depending on the size of the tree. The final nodes values are only zeros.



1. Depending on the values of input, change the node value to 1 where appropriate.

For example, when the input is equal to ‘111’, the corresponding node’s value becomes ‘1’.

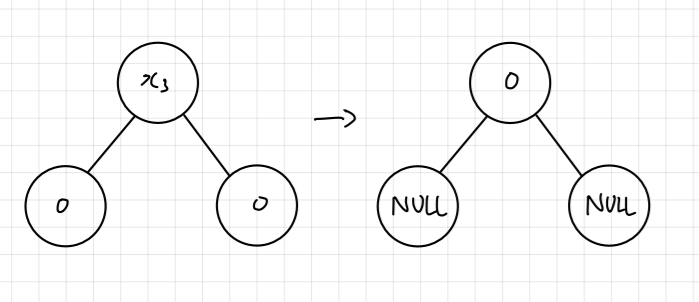


Simplifying Tree

1. The simplification process can be divided into two steps. The first step simplifies tree when the certain node satisfies following conditions:

* The value of t->left is equal to that of t->right
* The values of t->left and t->right are either zero or one

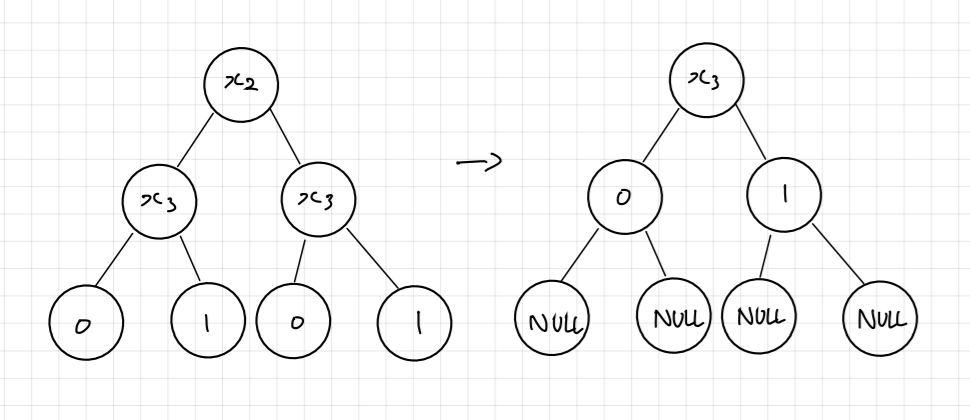
Once these conditions are met, t value(t->val) now changes to that of t->left or t->right and both t->left and t->right nodes get empty (t->left = NULL & t->right = NULL). The first step is repeated several times so that the tree is fully simplified before it undergoes the second step.



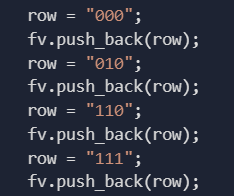
1. The second step simplifies tree when the certain node satisfies following conditions:

* The value of t->left->left is equal to that of t->right->left
* The value of t->left->right is equal to that of t->right->right
* The values of t->left->left, t->left->right, t->right->left and t->right->right is either zero or one

Once these conditions are met, t value now changes to that of t->left or t->right and the value of t->left changes to that of t->left->left and t->right value changes to that of t->right->right. Finally, t->left->left, t->left->right, t->right->left and t->right->right nodes get empty (t->left->left = NULL and so on). The second step is also repeated several times so that the tree is fully simplified.



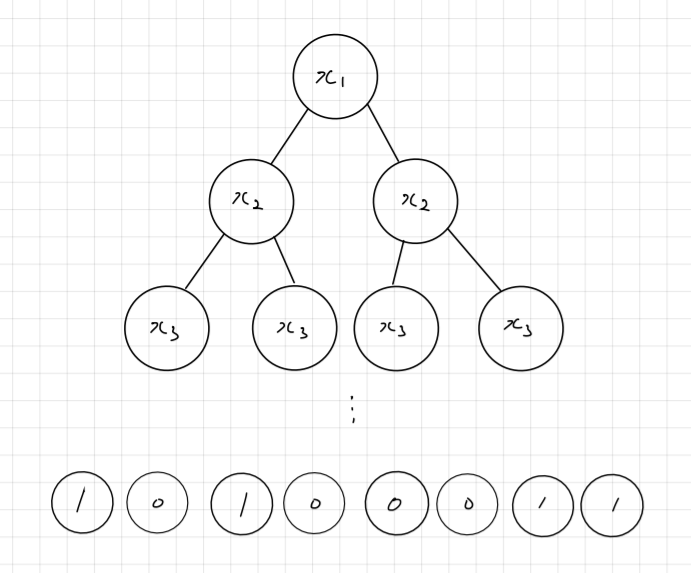
Worked example



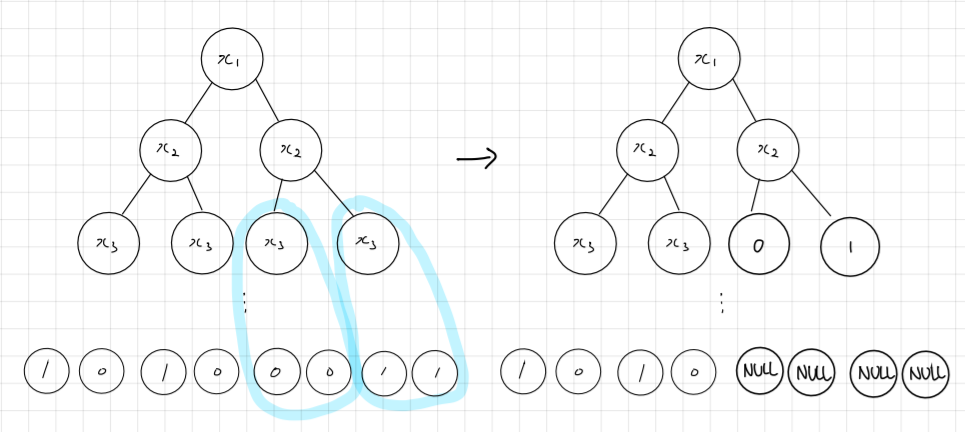
Assume this is the input vector, that gives result 1 for a certain Boolean function, we can draw a truth table:

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | X3 | F (x1, x2, x3) |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

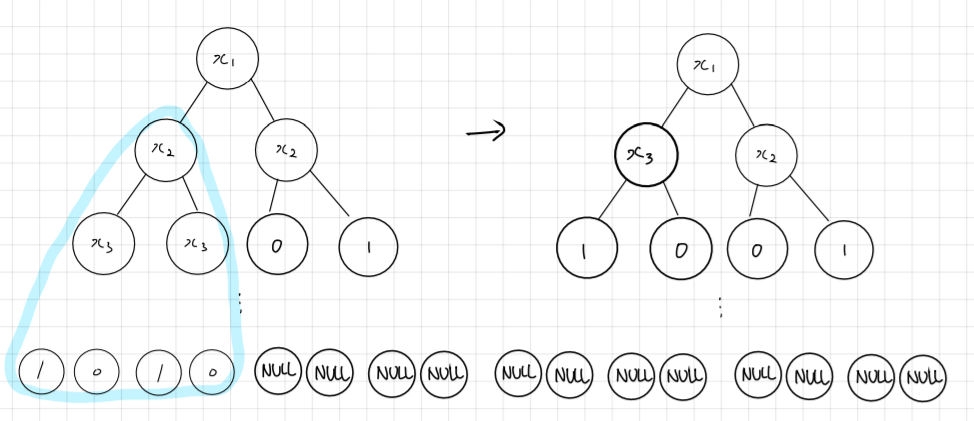
The original tree is only filled with zeros. However, using the truth table, a tree can be filled with ‘1’s where needed.



After the complete construction of tree, first simplification occurs.



Second simplification occurs.



As a result, the number of nodes reduced from 15 to 7 after simplification process.

**Section 2 – Testing and evaluation**

Using the same example from section 1, a tree with 15 nodes is formed at first since there are 3 variables x1, x2 and x3. According to the worked example in the section one, these nodes must be reduced to 7 by removing total 8 nodes.

The maximum number of nodes that can be removed is 12 nodes since 3 nodes structure is the smallest possible simplification.

In order to find the pattern, some values are measured as followed:

|  |  |
| --- | --- |
| Number of variables | Maximum number of reducible nodes |
| 1 | 0 |
| 2 | 4 |
| 3 | 12 |
| 4 | 28 |
| 5 | 60 |
| 6 | 126 |

Although these collected data is simple and basic, some patterns are observed:

1. The maximum number of reducible nodes are divisible by the number of variables
2. The number of reducible nodes is all even numbers (This is due to the algorithms)

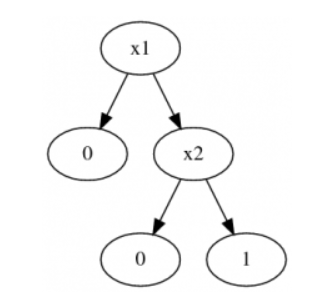
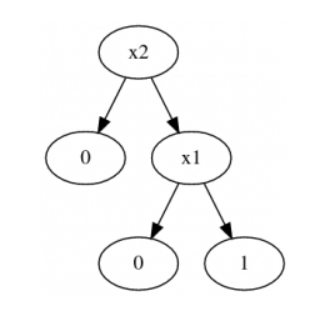
Does your final version perform better than your initial one?

Yes, it does due to the following reasons:

1. Simplification process removes the overlapping nodes and minimizes the tree size which leads to the efficient use of storage inside the tree.
2. “label\_to\_idx” function converts the string “x3” to integer 2, “x2” to integer 1 and so on. Using the “label\_to\_idx” function, “eval\_bt” function identifies the string value of the corresponding index of input string value. Hence, unlike the initial one, the comparison of entire string input is now unnecessary once the tree simplification is completed correctly. In other words, final version is more efficient.

However, there are some weaknesses in my final version.

1. Some trees can be simplified as following:



This is an example shown in the third assessment page. (Specific “AND” Boolean function’s simplification). However, my final version didn’t manage to swap the nodes including “x”. In other words, “x1” and “x2” can’t be swapped in my final version and the nodes including “x” should be ordered ascending power in the tree.