Contents

Tree Drawing

Vincent La

April 18, 2018

1 Motivation

Trees with at most two child nodes are used widely in computer science. The simplicity of their structure easily lends to mathematical analysis about algorithms operating on binary trees. Given the vast utility of this tree, many have tried to define algorithms which draw binary trees.

2 Reingold-Tillford (1981)

One classic algorithm used to layout binary trees is described by Reingold and Tillford.

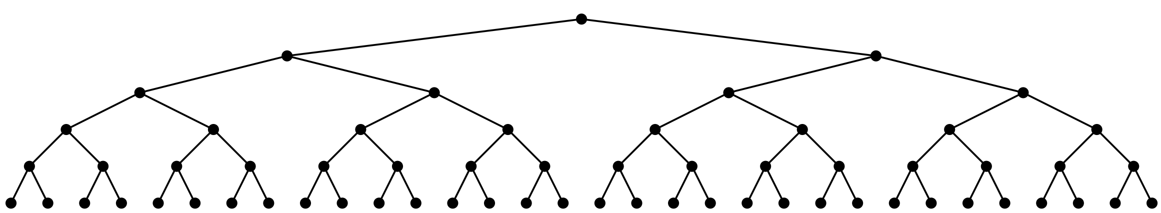
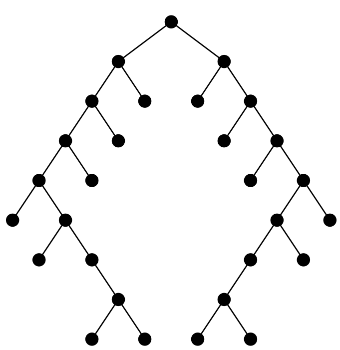
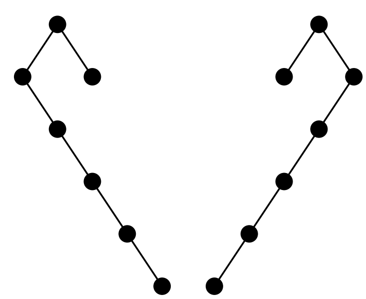


Figure 1: A complete binary tree



(a) An example of a tree generated by RT 81



(b) Unlike other algorithms, RT 81 draws a tree and its mirror symetrically

Figure 2: A reproduction of some figures from RT's original paper

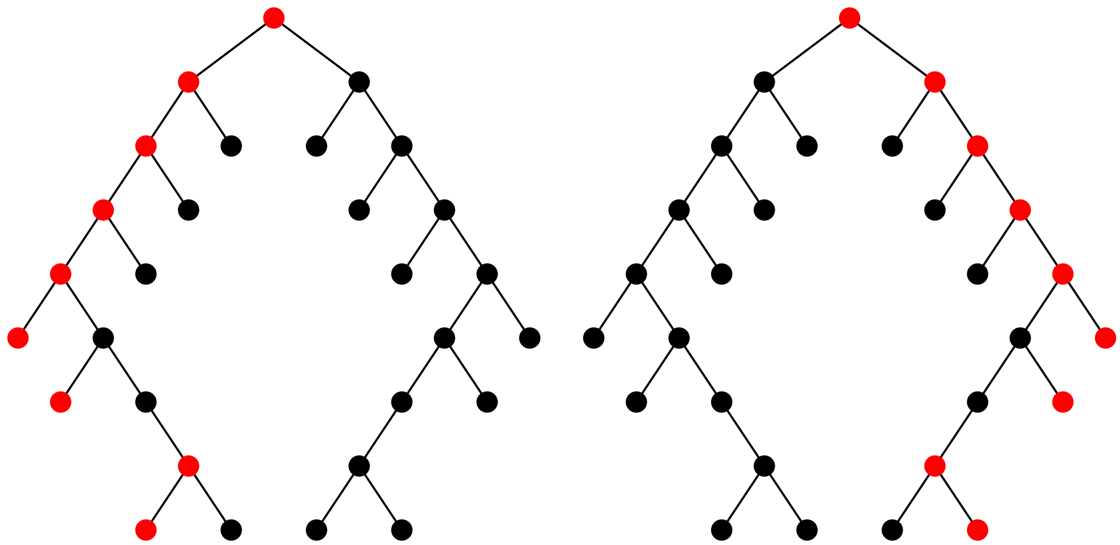


Figure 3: The left contour and right contour of the same tree

2.1 Algorithm Description

2.1.1 Informal Description

First, we calculate the displacements of the nodes relative to each other.

1. Base Case: Trivial
2. Apply this algorithm to subtrees via a postorder traversal
3. For each subtree, merge them horizontally such that they are two units apart horizontally

2.1.2 Formal Description

Node Type To implement the tree in a programming language, we will need to creature a structure that represents each node in the tree.

left Pointer to left subtree

right Pointer to right subtree

offset Offset relative to parent

By using offsets relative to a node's parent, as opposed to absolute offsets, we avoid having to reposition all of the nodes in a subtree when the root of that subtree gets moved.

Algorithm 1 Reingold and Tilford's Algorithm

Constants:
minsep: The smallest distance any two subtrees can be separated by [units]

Input:
t: A binary tree

```
procedure SETUP(t)
    cursep = minsep                                  ▷ Used to keep track of how far apart subtrees are
    setup(t→left)                                    ▷ Post-order traversal on tree
    setup(t→right)
    left ← t → left
    right ← t → right
    while left is not NULL right is not NULL do      ▷ We only have to traverse as deep as the shortest subtree
        if cursep < minsep then                      ▷ Trees too close, so push them apart
            left_dist ← left_dist + (minsep − cursep)/2
            right_dist ← right_dist + (minsep − cursep)/2
            cursep = minsep
        end if
        if left → right not null then                ▷ Traverse left subtree
            left ← left → right
            cursep ← cursep − left.offset
        else
            left ← left → right
            cursep ← cursep − left.offset
        end if
        if right → left not null then                ▷ Traverse right subtree
            right ← right → left
            cursep ← cursep − right.offset
        end if
        if left → right not null then
            right ← right → right
            cursep ← cursep − right.offset
        end if
    end while
    if left then                                     ▷ The left subtree was taller
        Insert a thread from the right-most item of the right subtree to right
    end if
    if right then                                    ▷ The right subtree was taller
        Insert a thread from the left-most item of the left subtree to right
    end if
end procedure
```

2.1.3 Threading

For every node in a given tree, the algorithm traverses through the left contour of the right subtree, and the right contour of the left subtree. In this traversal, the goal is to find the appropriate number units to separate the trees by. However, for any given node, its right contour may not be contained entirely within the same subtree. Hence, we require "threads", or connections between nodes in different subtrees in order to follow a tree's contour. Per the visualization below, we may think of threads as temporary edges.

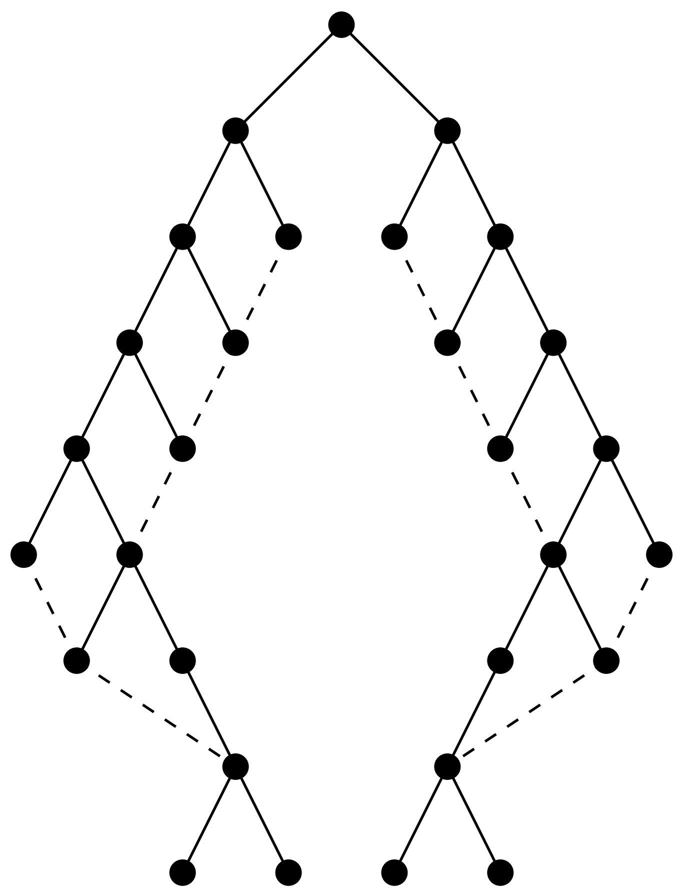


Figure 4: Threads (dashed) created by the algorithm while traversing the tree in Figure 2a

2.2 Algorithm Trace for Complete Binary Trees

The figures below show the displacements set by the algorithm for each node. From these figures, we can see how the algorithm achieves symmetry between subtrees.

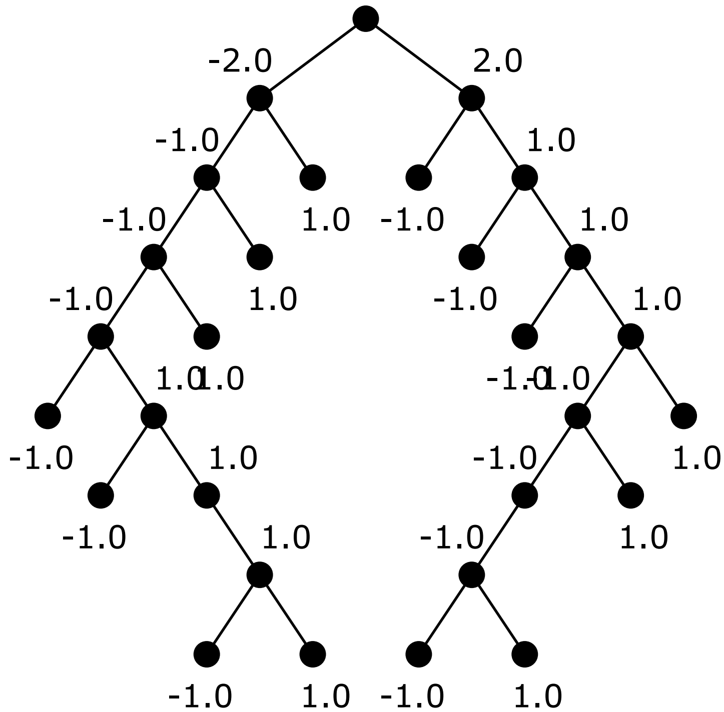


Figure 5: Algorithm trace for example tree

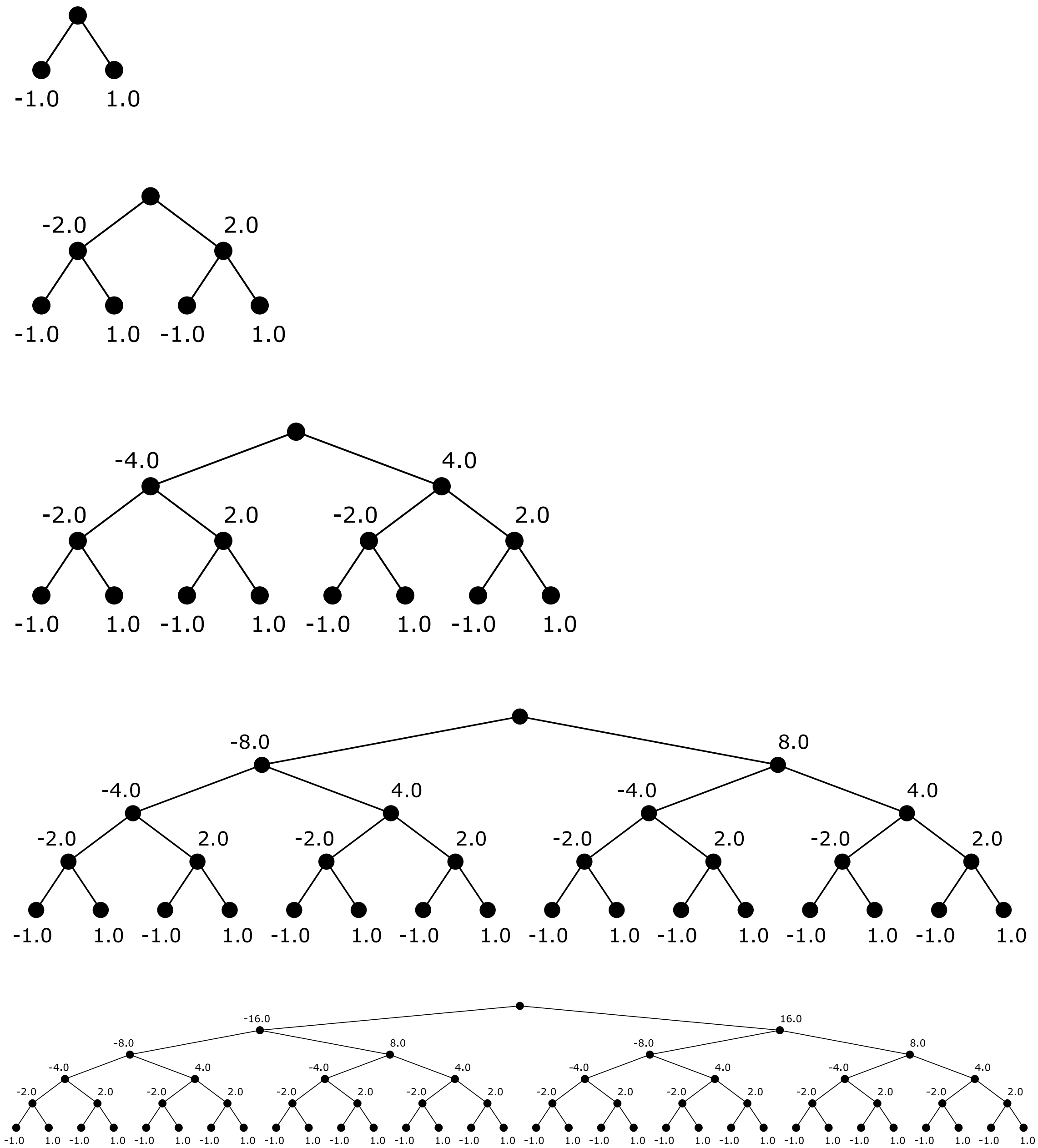


Figure 6: Algorithm trace for complete binary trees of heights 2 through 6

2.3 N-Ary Tree Drawing

The algorithm above can also be generalized to trees with n roots.