On Random Walks in Binary Trees

by

Ian Munro

Research Report CS-76-33

Department of Computer Science

University of Waterloo
Waterloo, Ontario, Canada

June 1976

Keywords: average path length binary tree combinatorics

The problem of determining the average path length in various classes of rooted binary trees has been studied in considerable depth ([1]), primarily as a result of interest in binary search trees. In this note we turn our attention to a variation on this question, perhaps best described as determining the expected length of a random walk in binary trees.

By a binary tree we will mean a rooted tree, oriented away from the root, with each node of outdegree at most 2, and such that for any edge $\overrightarrow{xy}$, y is distinguished as either the (unique) right son or left son of x. We may, therefore, think of the nodes of any such tree as having the unique labelling of the integers 1 through n such that the label of a node follows those on its left subtree and precedes those on its right. The usual notion of average path length in such a tree is the average length of the paths from the root to each of the nodes of the tree (counting the distance of the root from itself as 0). Obviously the average path length is minimized at $\sum_{i=1}^{n} \lfloor \log i \rfloor / n$ (about $\log n - 1$) if the tree is balanced. It can be shown that the average path length over the set of all binary trees is approximately $\sqrt{\pi n}$ (see for example [1] p. 590).

Our present interest lies not in the average distance from a node to the root, but in what we shall call the average length of a random walk. Suppose we start at the root of a binary tree and randomly (with equal probabilities) choose to follow the left or right branch. How many edges will be taken on the average, over all binary trees on n nodes, before a chosen edge is found not to be present? The answer is about 2, indeed:

Theorem - The average length of a random walk over the set of n node binary trees is $2 - 6/(n+2)$.

The following lemmata are useful in demonstrating the theorem.

Lemma 1 - The average walk length over the set of all n node binary trees is the same for every left-right sequence.

Proof - Follows immediately since interchanging the left and right subtrees at any fixed point (end of a finite left-right sequence) in the set of n node binary trees simply maps the set into 1-1 correspondence with itself.

∎

We are now free to prove our result for any particular path, we choose the path in which the right branch is always taken.

Lemma 2 - There exists a correspondence between each n node binary tree, T, and a set of

$$2 + (\text{length rightmost path in } T)$$

trees each containing $(n+1)$ nodes. Furthermore these sets form a partition of the set of $(n+1)$ node binary trees.

Proof - Assume the left to right labelling convention noted above is adopted. We can construct an $n+1$ node tree by appending a node labelled $(n+1)$ to an n node binary tree by

1) Making the new node the root of the tree and the original root its left son.

or 2) For any node, x, on the rightmost path, make the new $(n+1)$ node the right son of x and the former right son (if any) of x, the left son of the new node (see figure 1).

The rest of the tree remains unaltered.

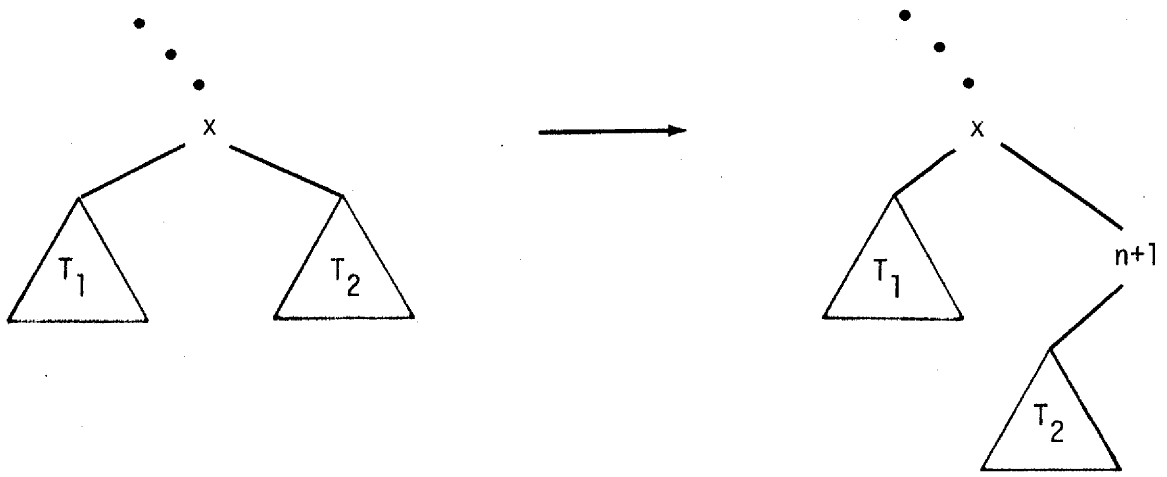

Figure 1 - Addition of a node to an n-node tree

Note that any (n+1) node tree can be constructed in this manner from exactly one n node tree and "position" in the rightmost path.

■

Proof of the Theorem - From Lemma 2 we see that

$$\# \text{ (n+1) node binary trees} = \sum_{\substack{\text{all n}\\ \text{node}\\ \text{binary}\\ \text{trees}}} \text{(2+ length rightmost path)}$$

$$= (\#\ n \text{ node trees}) \times (2+A)$$

Where A denotes the average length of the rightmost path. However the number of n node binary trees is given by the nth Catalan number, $\binom{2n}{n}/(n+1)$.

Hence

$$\binom{2n+2}{n+1} / (n+2) = (2+A) \binom{2n}{n} / (n+1)$$

so $$A+2 = 2(2n+1) / n+2$$

or $$A = 2-6/(n+2)$$

The theorem now follows from Lemma 1.

■

Reference

[1] Knuth, D.E., The Art of Computer Programming, Volume I, Fundamental Algorithms, Addison Wesley, Don Mills, 1968.