An application of generating functions to AVL trees

1. Definitions

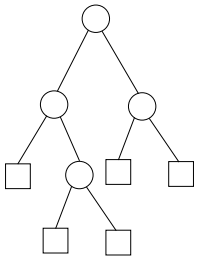
1.1 Definition (*Internal and External path lengths*). For an *extended* tree of n *internal* nodes we define two quantities, *external path length*, E_n, and *internal path length*, I_n, as follows. First off, set the root level at 0.

$$E_n = \sum_{\substack{\lambda \text{ is the level} \\ \text{of an external node}}} \lambda$$

$$I_n = \sum_{\substack{\lambda \text{ is the level} \\ \text{of an internal node}}} \lambda$$

□

1.2 Example. In the following extended tree we have $n = 3$ and $E_3 = 2+2+2+3+3 = 12$ while $I_3 = 0+1+1 = 2$.



It is clear that if the above tree were to be used for *search*, then the *unsuccessful* exits would involve $2, 2, 2, 3, 3$ probes (a total of 12, if we had each case of unsuccessful exit exactly once) and the successful exits would have $1, 2, 2$ probes (that is, $0+1, 1+1, 1+1$, or $1+$ *the level number of the node where success occurred.*)

It is easy to see, in general, that E_n is the *total* number of probes towards obtaining *each unsuccessful exit once*, while $I_n + n$ is the total number of probes towards having *each successful exit once.* □

The above motivates the following definition, *on the assumption that, when searching a tree of n internal nodes, each of the n successful exits are equally likely, and each of the $n+1$ unsuccessful exits are equally likely.*

1.3 Definition. Given an extended tree of n internal nodes, the quantities

$$U_n = \frac{E_n}{n+1}$$

and

$$S_n = \frac{I_n + n}{n} = \frac{I_n}{n} + 1$$

are termed *average unsuccessful number of probes* and *average successful number of probes* respectively. □

As any search of a table of n entries that is based on *comparisons only* gives rise to an extended ("decision") tree of n internal nodes, the above concepts of average "search times" apply to *any* comparison-only-based search, not just to tree-search.

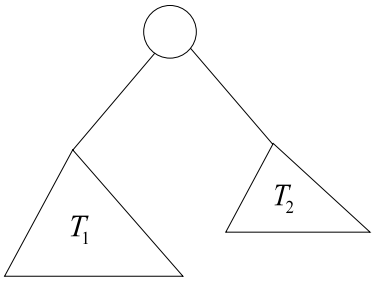
It turns out that E_n and I_n are connected by a simple formula, and hence so are U_n and S_n. So if we know one, we know the other.

1.4 Theorem. *$E_n = I_n + 2n$ for all $n \geq 0$.*

Proof. We do induction on extended trees.

The *Basis* considers the tree □. For this tree, $E_0 = 0$, $I_0 = 0$, hence $E_0 = I_0 + 2 \cdot 0$ is true.

We proceed to the "big" tree below, on the assumption (I.H.) that the claim is true for "small" trees (such as T_1 and T_2 below).



Let T_1 have l internal nodes and T_2 have r internal nodes. Then

$$E_l = I_l + 2l$$

and

$$E_r = I_r + 2r$$

Note that $n = l + r + 1$ (total number of internal nodes of the whole tree), thus

$$\begin{aligned} E_n - I_n &= (E_l + l + 1 + E_r + r + 1) - (I_l + l + I_r + r) \\ &= E_l - I_l + E_r - I_r + 2 \\ &= 2(l + r + 1) = 2n \end{aligned}$$

□

1.5 Corollary. *U_n and S_n are related by*

$$U_n = \frac{nS_n + n}{n + 1}$$

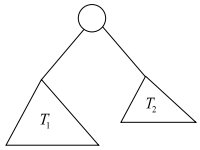
Proof.

$$\begin{aligned} U_n &= \frac{E_n}{n+1} \\ &= \frac{I_n + 2n}{n+1} \\ &= \frac{nS_n + n}{n+1}, \text{ since } nS_n = I_n + n \end{aligned}$$

□

1.6 Definition. An *AVL* or *balanced* (*extended*) tree is defined recursively by

(Basis) □ is an AVL tree. The following tree



is an AVL tree iff *both* T_1 and T_2 are AVL trees, and $|h_1 - h_2| \in \{0, 1\}$, where h_i is the height of T_i. □

"Iteratively", one can say that a tree is AVL iff *at every node* the heights of left and right subtrees differ by no more than 1.

2. Some theorems

A balanced tree has the nice property that its height is bounded by the logarithm of the number of its nodes. This is good, because the height of a (search) tree gives the *worst case* number of probes for search.

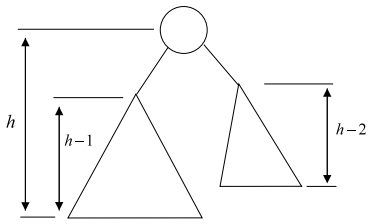
2.1 Theorem. *Any AVL of n internal nodes and of height h satisfies $h = O(\log n)$.*

Proof. For any AVL tree of height h, let

$$B(h) = \text{ the } \textit{minimum} \text{ number of internal nodes for this height.}$$

Clearly, $B(0) = 0$ (the tree is just $\square$) and $B(1) = 1$.

Look now at an AVL tree of height h (below) which has the *minimum possible* number of nodes, $B(h)$.



It is impossible for both left and right subtrees to have height $(h-1)$, because if they do, we can reduce one's height (say, the right's) by *taking away internal nodes*. That would contradict the assumption that the whole tree has $B(h)$ internal nodes (that is, minimum). So, without loss of generality, we adopted the above picture.

Finally, the left tree must have $B(h-1)$ (internal) nodes (that is, the *minimum possible*) otherwise reducing it down to $B(h-1)$ would reduce the number of nodes of the overall tree; impossible. Similarly, the right tree has $B(h-2)$ nodes. Thus, we arrive at:

$$\begin{aligned} &B(0) = 0 \\ &B(1) = 1 \\ &B(h) = B(h-1) + B(h-2) + 1, \text{ for } h > 1 \end{aligned} \tag{1}$$

We can solve this recurrence easily using generating functions, but we can do better:

For $h \geq 0$, $B(h) + 1 = F_{h+2}$, where F_n is the Fibonacci sequence with $F_0 = 0, F_1 = 1$ as starting conditions.†

† No magic in this observation. It follows directly, more or less, if we add 1 at both sides of (1).

The contention can be proved by induction. For the *Basis* ($h = 0$) note that $F_2 = 1 = B(0) + 1$.

Assume the claim (I.H.) for all indices (strictly) below h, and proceed to h (of course, whatever you do next, it must be valid for any $h > 0$).

Now we have two cases:

Case $h > 1$. Then both $h-1$ and $h-2$ are *valid indices* (arguments) for B. By I.H., $F_h = B(h-2) + 1$ and $F_{h+1} = B(h-1) + 1$. Thus

$$\begin{aligned} B(h) + 1 &= B(h-1) + 1 + B(h-2) + 1 \\ &= F_{h+1} + F_h = F_{h+2} \end{aligned}$$

Case $h = 1$. By direct verification, $F_3 = 2 = B(1) + 1$.

Now, we know that $F_h \sim (1/\sqrt{5})\phi^h$, where $\phi = (1+\sqrt{5})/2$. Thus, in view of the connection between F and B, for *very large* h we may write (*approximately*)

$$B(h) + 1 = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^{h+1} \qquad (i)$$

Recall that our AVL tree of height h, *really* has n internal nodes. Thus, since $n \geq B(h)$,

$$n + 1 \geq \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^{h+1}$$

hence

$$\log_2(n+1) \geq \log_2(\frac{1}{\sqrt{5}}) + (h+1)\log_2 \left(\frac{1+\sqrt{5}}{2} \right)$$

or

$$\log_2(n+1) \geq -1.16 + (h+1)0.69$$

and finally,

$$1.44 \log_2(n+1) + .68 \geq h$$

More fuzzily, $h = O(\log n)$. □

2.2 Corollary. *For very large* h, $B(h)/B(h-1) = \phi$ (approximately!) *and* $B(h)/B(h-2) = \phi^2$.

Proof. It follows from (i) above if we omit the $+1$ in the left hand side, a legitimate action due to the overwhelming size of the right hand (and hence left hand) side. □

2.3 Theorem. *The average performance of search in AVL trees is about* 30% *better than the worst case performance.*

We work here under the simplifying assumption that all the AVL trees are "skewed" like the minimum-node ones (an inaccurate assumption, but one that allows the analysis to proceed, and yields results close to experimental ones).

Proof. We assume height h and number of internal nodes $B(h)$. The height, h, is, of course, the worst case performance for search (same for successful and unsuccessful). The *average* successful performance is

$$S_{B(h)} = \frac{I_{B(h)}}{B(h)} + 1 \tag{1}$$

We will abuse notation in favour of the simpler

$$S_h = \frac{I_h}{B(h)} + 1 \tag{2}$$

Most of the work will be to estimate

$$\frac{I_h}{B(h)}$$

for *very large* h.

Now,

$$I_h = I_{h-1} + I_{h-2} + B(h) - 1$$

thus

$$\begin{aligned}
\frac{I_h}{B(h)} &= \frac{I_{h-1}}{B(h)} + \frac{I_{h-2}}{B(h)} + 1 - \frac{1}{B(h)} \\
&= \frac{I_{h-1}}{B(h-1)}\frac{1}{\phi} + \frac{I_{h-2}}{B(h-2)}\frac{1}{\phi^2} + 1 - \frac{1}{B(h)} \qquad (3) \\
&= \frac{I_{h-1}}{B(h-1)}\frac{1}{\phi} + \frac{I_{h-2}}{B(h-2)}\frac{1}{\phi^2} + 1 \qquad (4)
\end{aligned}$$

where in (3) and (4) above use was made of "h is very large!" (using Corollary 2.2) to obtain the *approximate* equalities (due to the size of $B(h)$, $1/B(h)$ is practically 0 for large h, so it was omitted in going from (3) to (4)).

Let us write A_h for $I_h/B(h)$ to simplify notation. So we have

$$\begin{aligned}
A_H &= K \\
A_{H+1} &= L \\
A_h &= A_{h-1}\frac{1}{\phi} + A_{h-2}\frac{1}{\phi^2} + 1, \text{ for } h \geq H+2 \qquad (5)
\end{aligned}$$

where H is a *very large positive integer* (and K, L the values of $A_h = I_h/B(h)$ at $h = H$ and $h = H+1$ respectively) *from which onward the approximation* (5) *is good*.

We now solve (5), to find how A_h relates to h. We can shift indices to start from 0 (e.g., setting $C_h = A_{h+H}$, but this is not necessary).

The generating function for A_h is

$$G(z) = A_H + A_{H+1}z + A_{H+2}z^2 + \cdots + A_{H+h}z^h + \cdots$$

By (5), and the familiar technique,

$$\begin{aligned} G(z)(1 - \frac{z}{\phi} - \frac{z^2}{\phi^2}) &= K + Lz - \frac{1}{\phi}Kz + z^2(1 + z + z^2 + \cdots) \\ &= K + Sz + \frac{z^2}{1-z} \end{aligned}$$

where $S = L - (1/\phi)K$. The exact values of K, L (and hence S) are unimportant in our analysis, as it will become clear.

The equation

$$1 - \frac{z}{\phi} - \frac{z^2}{\phi^2} = 0$$

has roots $z = 1$ and $z = -\phi^2$ (verify!), hence

$$1 - \frac{z}{\phi} - \frac{z^2}{\phi^2} = -\frac{1}{\phi^2}(z-1)(z+\phi^2) = (1-z)(1 + \frac{z}{\phi^2})$$

Thus,

$$G(z) = \frac{K}{(1-z)(1+\frac{z}{\phi^2})} + \frac{Sz}{(1-z)(1+\frac{z}{\phi^2})} + \frac{z^2}{(1-z)^2(1+\frac{z}{\phi^2})} \tag{6}$$

Splitting (6) into partial fractions we get

$$G(z) = \frac{P}{1-z} + \frac{Q}{(1-z)^2} + \frac{R}{1+\frac{z}{\phi^2}} \tag{7}$$

for appropriate P, Q, R, which yields

$$A_{H+h} = P + Q(h+1) + R(-\frac{1}{\phi^2})^h$$

or $A_{H+h} = O(h)$ since

$$(-\frac{1}{\phi^2})^h$$

goes to 0 as h goes to infinity. For our "numerical" answer we only need to find Q. From (6) and (7), multiplying out by $(1-z)^2$ and setting $z \leftarrow 1$, we obtain $Q = 1/(1 + 1/\phi^2) = .72$, hence, for large h, $A_h = .72h$ (shouldn't I have said "$A_{H+h} = .72h$" instead?), therefore also $S_h = A_h + 1 = .72h$ and we are done! □