Building Efficient Binary Search Trees

In searching for an item in Binary Search Tree we want to reduce both Average and Worst case number of Comparisons. To achieve this we build a tree with minimal height. This can be achieved by filling all levels in the tree (except maybe the leaf level).

Perfectly Balanced Trees.

A perfectly balanced tree has minimal height.

Defn.

A tree t is perfectly balanced for each node the number of nodes in the left and right subtrees differ by at most one, i.e. for (sub)tree t,

$$| \#\ t.left\ - \#\ t.right\ | \leq 1 \quad \text{where } \#\ t \text{ is the number of nodes in t.}$$

We can build a perfectly by distributing n nodes as follows:

Use one node for the root, t,

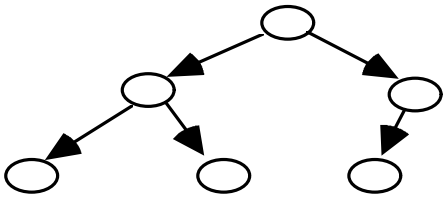
build a left subtree with # t.left = # t div 2

build right subtree with # t.right = (#t - # t.left - 1)

Height of a perfectly balanced tree $= \lceil (\log_2 (\#t+1) \rceil$

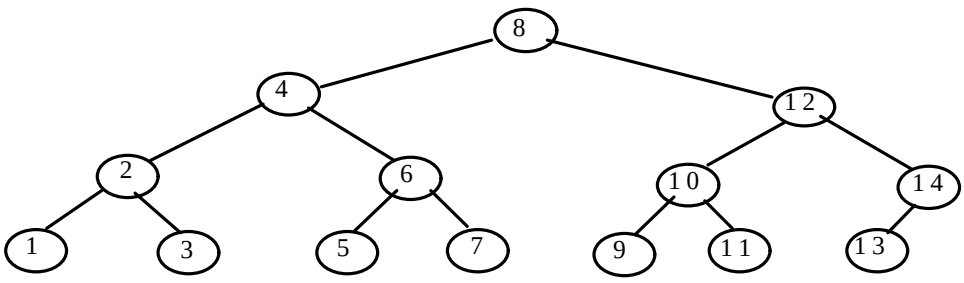
where $\lceil x \rceil$ is "ceiling x", the least integer $\geq x$

e.g. Perfectly Balanced Tree of # t = 6

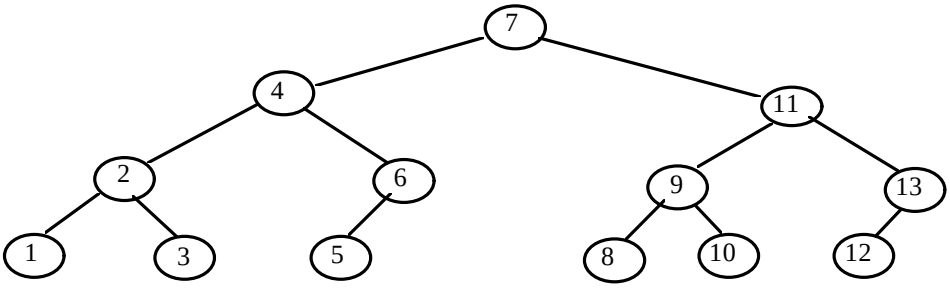


While a perfectly balanced tree is optimal for searching, it can have a poor worst case for insertion and deletion. In inserting or deleting an item the tree may have to be rebalanced.

Consider deleting 14 in the following;



After deletion the whole tree has to be reorganised to be again perfectly balanced.



The operations of Inserting and Deleting are too complex.

If there is no Insertion or Deletion, i.e. the tree is intially constructed and not changed, then we have a Static tree which provides efficient search operations.

For Searching, a Binary Search Tree is on average about 40% worse that a Perfectly Balanced Tree which is O(log n), but the Worse case performance of a Binary Search Tree is O(n).

Height Balanced Trees
AVL Trees *(Adelson-Velski and Landis, 1962)*

We can achieve performance very close to that of Perfectly Balanced trees if we consider Height Balnacing rather than Perfect Balancing. In a Height Balanced Tree, all the operations Insert, Delete and Search are O(log n), where n is the number of items (nodes) in the tree.

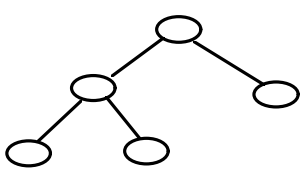
Defn. Height Balanced Tree -- AVL Tree

A (Binary Search) Tree T is Height (AVL) Balanced if for any node k, the heights of the left and right subtrees of k differ by at most one.

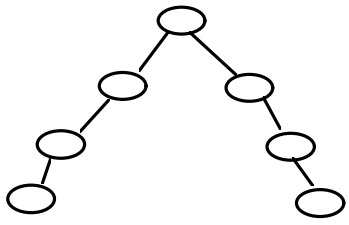
i.e. | height(k.left) - height(k.right) | ≤ 1

Height / Depth of Tree = Max level of all the nodes

(level of root = 1)



Height Balanced



Not Height Balanced

Perfectly Balanced Trees are always Height Balanced.

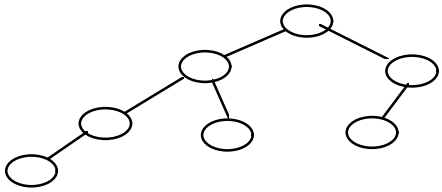
If a Perfectly Balanced Tree has n nodes then its height is $\lceil \log (n+1) \rceil$;

if $n = 2^h - 1$ then height = h (= log (n+1))

To compare the performance of Height Balanced Trees with Perfectly Balanced trees we consider the worse case of a Height Balanced Tree (AVL Tree).

Given n nodes how high can an AVL Tree grow,
(this will be the worse case).

e.g. n = 7, height = 4



Defn. Fibonacci Tree (Worse case AVL tree)

An AVL balanced tree of height h with the minimum number of nodes is a Fibonacci tree.

For h = 4, min. #nodes = 7 (any less, either it would not be an AVL tree or h = 3)

Given an AVL tree of height h,

Maximum #nodes = $2^h - 1$, a perfectly balanced tree

Minimum #nodes = ??, a Fibonacci tree.

#Nodes in a Fibonacci Tree with height h

h = 0 Min(h) = 0 i.e. Min(0) = 0

h = 1, Min(h) = 1 -- only one tree with h=1

i.e. Min(1) = 1

If an AVL tree is to have a minimal #nodes then the left and right subtrees will also be minimal, i.e. a fibonacci tree will have left and right fibonacci trees.

The left could have height h -1 with Min(h-1) nodes and
the right have heigth h-2 with Min(h-2) nodes
then minimum #nodes in tree is

Min(h) = Min(h-1) + Min(h-2) + 1

This is similar to the fibonacci formula,

fib(n) = fib(n-1) + fib(n-2), with fib(0) = 0 & fib(1) = 1

By (straight forward) induction we can show,

Min(h) = fib(h+2) - 1

From fibonacci theory, we get that

$$\text{fib}(n) = \frac{g^n - \gamma^n}{\sqrt{5}}$$

where g = $\frac{1+\sqrt{5}}{2}$ and $\gamma = \frac{-1}{g}$

This can be shown by induction.

Note:

g and γ are solutions to the quadratic equation

$$x^2 - x - 1 = 0$$

$g = \frac{1+\sqrt{5}}{2}$ and $\gamma = \frac{1-\sqrt{5}}{2}$; g is the Golden Ratio

Also, $\gamma = 1 - g$

as $g^2 - g - 1 = 0$

tf. $g = 1 + \frac{1}{g}$

$$\frac{-1}{g} = 1 - g$$

i.e. $\gamma = 1 - g$

g is a irrational number with approx. value ≈ 1.618..

and γ has approx. value ≈ -0.618..

Let [x] be round(x), the nearest integer to x

e.g. [1.5] = 2, [-1.4] = -1

end Note

Show fib(n) = $\left[\frac{g^n}{\sqrt{5}}\right]$

Pf.

$$\text{fib}(n) = \frac{g^n}{\sqrt{5}} - \frac{g^n}{\sqrt{5}}$$

but $\left|\frac{g^n}{\sqrt{5}}\right| < 0.5$, all n, as $\gamma \approx -0.618..$ and $\sqrt{5} \approx 2.24$

tf. fib(n) = $\left[\frac{g^n}{\sqrt{5}}\right]$

end Pf.

Performance of Searching an AVL Tree

We get from above,

$$\text{Min}(h) = \text{fib}(h+2) - 1$$

$$= \left[\frac{g^{h+2}}{\sqrt{5}}\right] - 1, \quad g \approx 1.618$$

The worst case AVL tree of height h can have a minimum of the Min(h) nodes.

For any AVL of height h we have

$$\text{Min}(h) \leq \#\text{nodes} \leq \text{Max}(h) = 2^h - 1$$

Worst case Best case

(fibonacci tree) (Complete tree)

The height gives the measure of efficiency so we want the hieight in terms of #nodes

Let n = #nodes

tf. $\log(n+1) \leq h$ -- as $n \leq \text{Max}(h) = 2^h - 1$

Also, since $n \geq \text{Min}(h)$

$$n \geq \left[\frac{g^{h+2}}{\sqrt{5}}\right] - 1$$

$$> \frac{g^{h+2}}{\sqrt{5}} - 2, \quad \text{as } [x] > x-1$$

$$n + 2 > \frac{g^{h+2}}{\sqrt{5}}$$

tf.

$$\log(n+2) > (h+2)\log(g) - \log(\sqrt{5})$$

$$\frac{\log(n+2)+\log(\sqrt{5})}{\log(g)} - 2 > h$$

(log (g) = 0.694, $\frac{1}{\log(g)}$ = 1.44 and $\log(\sqrt{5}) = 1.161$)

tf. $1.44 \log(n+2) - 0.328 > h$

Conclusion

Let n = #nodes , h = height, then

$$\log(n+1) < h < 1.44 \log(n+2) - 0.328$$

i.e. AVL trees require at most 45% more comparisons than Perfectly Balanced Trees

i.e. Given n nodes, we can always build an AVL (Height Balanced) tree which at worst will be a Fibonacci tree or at best a Perfectly Balanced tree so that the height h will be

$$\log(n+1) < h < 1.44 \log(n+2) - 0.328$$