1 Solutions to Tutorial Sheet 7

Questions **4.1** - **4.3** are straight forward.

Q. 4.4 Show that in a binary tree of N nodes, there are $N+1$ $NULL$ pointers.
Solution Every node has 2 outgoing pointers. Therefore there are $2N$ pointers. Each node, except the root node, has one incoming pointer from its parent. This accounts for $N-1$ pointers. The remaining $N+1$ pointers are $NULL$ pointers.

Q. 4.5 Show that the maximum number of nodes in a binary tree of height h is $2^{h+1}-1$.

Solution Proof by induction.
Base Case: $h = 0$**.** A binary tree of height 0 has one node. $2^{h+1}-1$ equals one for $h = 0$. Therefore true for $h = 0$.

Inductive Hypothesis Assume that the maximum number of nodes in a binary tree of height h is $2^{h+1}-1$, for $h = 1, 2, ..., k$.

Now consider a tree T of height $k+1$. The root of T has a left subtree and a right subtree each of which has height at most k. These can have at most $2^{k+1}-1$ nodes each by the induction hyptohesis. Adding the root node gives the maximum number of nodes in a binary tree of height $k+1$,

$$2(2^{k+1}-1)+1$$
$$2*2^{k+1}-2+1$$
$$2^{(k+1)+1}-1$$

Q 4.6 A *full node* is a node with two children. Prove that the number of full nodes plus one is equal to the number of leaves in a nonempty binary tree.

Solution Let N = the number of nodes, F = number of full nodes, L = the number of leaves, and H = the number of nodes with one child (or half nodes). The total number of nodes in a binary tree equals $N = F + H + L$.

Because each full node is incident on two outgoing edges, each half node is incident on one outgoing edge, and each leaf is incident on no outgoing edge it follows that the total number of edges in a binary tree equals $2F + H$. It is also true that the total number of edges in a tree equals $N - 1$. Thus,

$$\begin{aligned} 2F + H &= N - 1 \\ H &= N - 1 - 2F \end{aligned}$$

Subbing this value of H into $N = F + H + L$ gives,

$$\begin{aligned} N &= F + N - 1 - 2F + L \\ N - N + 1 &= -F + L \\ F + 1 &= L \end{aligned}$$

Q. 4.9 a Show the result of inserting $3, 1, 4, 6, 9, 2, 5, 7$ into an initially empty binary search tree.
Solution

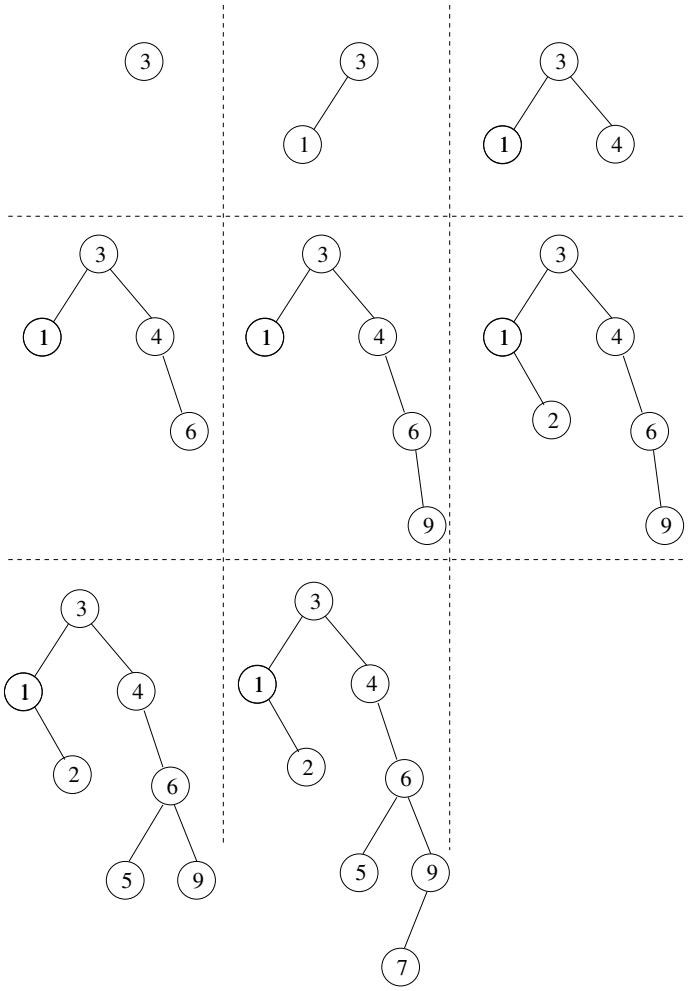

Fig. 1. A step by step illustration of all eight insertions

Q. 4.9 b Show the result deleting the root.
Solution

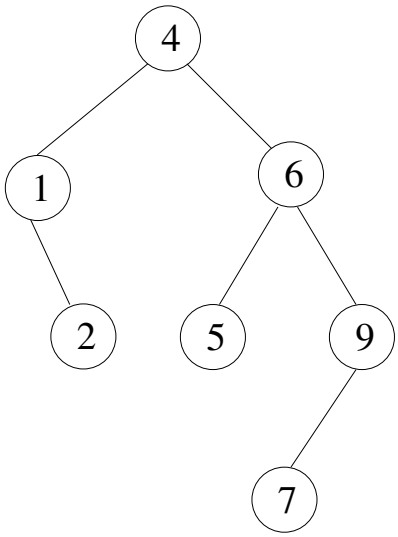

Fig. 2. The result of deleting the root node

Q. 4.14 Suppose you want to perform an experiment to verify the problems that can be caused by random `insert/remove` pairs. Here is a strategy that is not perfectly random, but close enough. You build a tree with N elements by inserting N elements chosen at random from the range 1 to $M = \alpha N$. You then perform N^2 pairs of insertions followed by deletions. Assume the existence of a routine, `randomInt(a,b)`, which returns a uniform random integer between a and b inclusive.

a. Explain how to generate a random integer between 1 and M that is not already in the tree (so a random insertion can be performed). In terms of N and α, what is the running time of this operation?

b. Explain how to generate a random integer between 1 and M that is already in the tree (so a random deletion be performed). What is the running time of this operation?

c. What is a good choice of α? Why?

Solution

a.

Strategy.

Keep a bit array (an array of 1's and 0's) B of size $M + 1$. If i is in the tree then $B[i]$ is set to 1; if i is not in the tree then $B[i]$ is 0. Repeatedly generate random numbers until you generate a number that is not in the tree.

Running time analysis.

Out of the M possible numbers that may be randomly generated, N are already in the tree and $M - N$ are not already in the tree. Thus the probability of a randomly generated number not being in the tree is $\frac{M-N}{M}$. It follows that the expected number of random numbers that need to be generated before getting a number that is not in the tree is

$$\frac{1}{\frac{M-N}{M}} = \frac{M}{M-N}$$

b.

Strategy.

Same as in part a.

Running time analysis.

The probability of a randomly generated number being in the tree is $\frac{N}{M}$. It follows that the expected number of random numbers that need to be generated before getting a number that is in the tree is

$$\frac{1}{\frac{N}{M}} = \frac{M}{N}$$

c.

The expected number of random numbers that need to be generated for each `insert/remove` pair is:

$$\begin{aligned}
&\frac{M}{M-N} + \frac{M}{N} \\
&= \frac{MN + M(M-N)}{(M-N)N} \\
&= \frac{MN + M^2 - MN)}{(M-N)N} \\
&= \frac{M^2}{(M-N)N}
\end{aligned}$$

Substituting αN for M gives

$$\begin{aligned}
&\frac{\alpha^2 N^2}{(\alpha N - N)N} \\
&= \frac{\alpha^2 N}{(\alpha N - N)} \\
&= \frac{\alpha^2 N}{(\alpha - 1)N} \\
&= \frac{\alpha^2}{(\alpha - 1)}
\end{aligned}$$

You must now find the value of α that minimizes the value of this function, with the restriction that $\alpha > 1$ (because if $\alpha \leq 1$ then $M \leq N$ are it will be impossible to generate a number that is not already in the tree). The value of $\alpha > 1$ that minimizes this function is 2.

Q. 4.19 Show the result of inserting $2, 1, 4, 5, 9, 3, 6, 7$ into an initially empty AVL-tree.

Solution

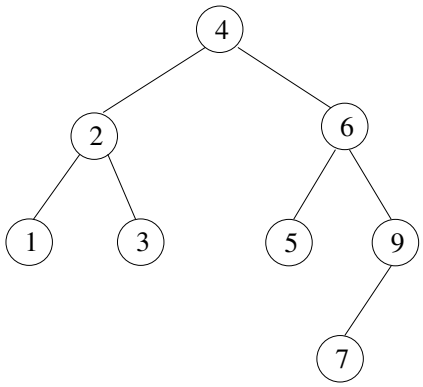

Fig. 3. The resulting AVL-tree.

Question 4.25
(a) How many bits are required per node to store the height of a node in an N-node *AVL* tree?
(b) What is the smallest *AVL* tree that overflows an 8-bit height counter?

Solution
(a) The depth of an *AVL* tree is $O(\log N)$. It takes $O(\log \log N)$ bits to store a number of size $O(\log N)$.

(b) With 8 bits one can represent all numbers in the range 0 to 255. So the answer is the smallest AVL tree with height 256.