Exercises

12.3-1
Give a recursive version of the TREE-INSERT procedure.

12.3-2
Suppose that we construct a binary search tree by repeatedly inserting distinct values into the tree. Argue that the number of nodes examined in searching for a value in the tree is one plus the number of nodes examined when the value was first inserted into the tree.

12.3-3
We can sort a given set of n numbers by first building a binary search tree containing these numbers (using TREE-INSERT repeatedly to insert the numbers one by one) and then printing the numbers by an inorder tree walk. What are the worst-case and best-case running times for this sorting algorithm?

12.3-4
Is the operation of deletion "commutative" in the sense that deleting x and then y from a binary search tree leaves the same tree as deleting y and then x? Argue why it is or give a counterexample.

12.3-5
Suppose that instead of each node x keeping the attribute $x.p$, pointing to x's parent, it keeps $x.succ$, pointing to x's successor. Give pseudocode for SEARCH, INSERT, and DELETE on a binary search tree T using this representation. These procedures should operate in time $O(h)$, where h is the height of the tree T. (*Hint:* You may wish to implement a subroutine that returns the parent of a node.)

12.3-6
When node z in TREE-DELETE has two children, we could choose node y as its predecessor rather than its successor. What other changes to TREE-DELETE would be necessary if we did so? Some have argued that a fair strategy, giving equal priority to predecessor and successor, yields better empirical performance. How might TREE-DELETE be changed to implement such a fair strategy?

⋆ 12.4 Randomly built binary search trees

We have shown that each of the basic operations on a binary search tree runs in $O(h)$ time, where h is the height of the tree. The height of a binary search tree varies, however, as items are inserted and deleted. If, for example, the n items are inserted in strictly increasing order, the tree will be a chain with height $n - 1$. On the other hand, Exercise B.5-4 shows that $h \geq \lfloor \lg n \rfloor$. As with quicksort, we can show that the behavior of the average case is much closer to the best case than to the worst case.

Unfortunately, little is known about the average height of a binary search tree when both insertion and deletion are used to create it. When the tree is created by insertion alone, the analysis becomes more tractable. Let us therefore define a ***randomly built binary search tree*** on n keys as one that arises from inserting the keys in random order into an initially empty tree, where each of the $n!$ permutations of the input keys is equally likely. (Exercise 12.4-3 asks you to show that this notion is different from assuming that every binary search tree on n keys is equally likely.) In this section, we shall prove the following theorem.

Theorem 12.4

The expected height of a randomly built binary search tree on n distinct keys is $O(\lg n)$.

Proof We start by defining three random variables that help measure the height of a randomly built binary search tree. We denote the height of a randomly built binary search tree on n keys by X_n, and we define the ***exponential height*** $Y_n = 2^{X_n}$. When we build a binary search tree on n keys, we choose one key as that of the root, and we let R_n denote the random variable that holds this key's ***rank*** within the set of n keys; that is, R_n holds the position that this key would occupy if the set of keys were sorted. The value of R_n is equally likely to be any element of the set $\{1, 2, \ldots, n\}$. If $R_n = i$, then the left subtree of the root is a randomly built binary search tree on $i - 1$ keys, and the right subtree is a randomly built binary search tree on $n - i$ keys. Because the height of a binary tree is 1 more than the larger of the heights of the two subtrees of the root, the exponential height of a binary tree is twice the larger of the exponential heights of the two subtrees of the root. If we know that $R_n = i$, it follows that

$$Y_n = 2 \cdot \max(Y_{i-1}, Y_{n-i}) .$$

As base cases, we have that $Y_1 = 1$, because the exponential height of a tree with 1 node is $2^0 = 1$ and, for convenience, we define $Y_0 = 0$.

Next, define indicator random variables $Z_{n,1}, Z_{n,2}, \ldots, Z_{n,n}$, where

$$Z_{n,i} = \mathrm{I}\{R_n = i\} .$$

Because R_n is equally likely to be any element of $\{1, 2, \ldots, n\}$, it follows that $\Pr\{R_n = i\} = 1/n$ for $i = 1, 2, \ldots, n$, and hence, by Lemma 5.1, we have

$$\mathrm{E}[Z_{n,i}] = 1/n , \tag{12.1}$$

for $i = 1, 2, \ldots, n$. Because exactly one value of $Z_{n,i}$ is 1 and all others are 0, we also have

$$Y_n = \sum_{i=1}^{n} Z_{n,i}\,(2 \cdot \max(Y_{i-1}, Y_{n-i})) \ .$$

We shall show that $\mathrm{E}[Y_n]$ is polynomial in n, which will ultimately imply that $\mathrm{E}[X_n] = O(\lg n)$.

We claim that the indicator random variable $Z_{n,i} = \mathrm{I}\{R_n = i\}$ is independent of the values of Y_{i-1} and Y_{n-i}. Having chosen $R_n = i$, the left subtree (whose exponential height is Y_{i-1}) is randomly built on the $i - 1$ keys whose ranks are less than i. This subtree is just like any other randomly built binary search tree on $i - 1$ keys. Other than the number of keys it contains, this subtree's structure is not affected at all by the choice of $R_n = i$, and hence the random variables Y_{i-1} and $Z_{n,i}$ are independent. Likewise, the right subtree, whose exponential height is Y_{n-i}, is randomly built on the $n - i$ keys whose ranks are greater than i. Its structure is independent of the value of R_n, and so the random variables Y_{n-i} and $Z_{n,i}$ are independent. Hence, we have

$$\begin{aligned}
\mathrm{E}[Y_n] &= \mathrm{E}\left[\sum_{i=1}^{n} Z_{n,i}\,(2 \cdot \max(Y_{i-1}, Y_{n-i}))\right] \\
&= \sum_{i=1}^{n} \mathrm{E}[Z_{n,i}\,(2 \cdot \max(Y_{i-1}, Y_{n-i}))] && \text{(by linearity of expectation)} \\
&= \sum_{i=1}^{n} \mathrm{E}[Z_{n,i}]\,\mathrm{E}[2 \cdot \max(Y_{i-1}, Y_{n-i})] && \text{(by independence)} \\
&= \sum_{i=1}^{n} \frac{1}{n} \cdot \mathrm{E}[2 \cdot \max(Y_{i-1}, Y_{n-i})] && \text{(by equation (12.1))} \\
&= \frac{2}{n} \sum_{i=1}^{n} \mathrm{E}[\max(Y_{i-1}, Y_{n-i})] && \text{(by equation (C.22))} \\
&\leq \frac{2}{n} \sum_{i=1}^{n} (\mathrm{E}[Y_{i-1}] + \mathrm{E}[Y_{n-i}]) && \text{(by Exercise C.3-4)} \ .
\end{aligned}$$

Since each term $\mathrm{E}[Y_0], \mathrm{E}[Y_1], \ldots, \mathrm{E}[Y_{n-1}]$ appears twice in the last summation, once as $\mathrm{E}[Y_{i-1}]$ and once as $\mathrm{E}[Y_{n-i}]$, we have the recurrence

$$\mathrm{E}[Y_n] \leq \frac{4}{n} \sum_{i=0}^{n-1} \mathrm{E}[Y_i] \ . \qquad (12.2)$$

Using the substitution method, we shall show that for all positive integers n, the recurrence (12.2) has the solution

$$\mathrm{E}[Y_n] \le \frac{1}{4}\binom{n+3}{3} .$$

In doing so, we shall use the identity

$$\sum_{i=0}^{n-1}\binom{i+3}{3} = \binom{n+3}{4} . \tag{12.3}$$

(Exercise 12.4-1 asks you to prove this identity.)

For the base cases, we note that the bounds $0 = Y_0 = \mathrm{E}[Y_0] \le (1/4)\binom{3}{3} = 1/4$ and $1 = Y_1 = \mathrm{E}[Y_1] \le (1/4)\binom{1+3}{3} = 1$ hold. For the inductive case, we have that

$$\begin{aligned}
\mathrm{E}[Y_n] &\le \frac{4}{n}\sum_{i=0}^{n-1}\mathrm{E}[Y_i] \\
&\le \frac{4}{n}\sum_{i=0}^{n-1}\frac{1}{4}\binom{i+3}{3} && \text{(by the inductive hypothesis)} \\
&= \frac{1}{n}\sum_{i=0}^{n-1}\binom{i+3}{3} \\
&= \frac{1}{n}\binom{n+3}{4} && \text{(by equation (12.3))} \\
&= \frac{1}{n}\cdot\frac{(n+3)!}{4!\,(n-1)!} \\
&= \frac{1}{4}\cdot\frac{(n+3)!}{3!\,n!} \\
&= \frac{1}{4}\binom{n+3}{3} .
\end{aligned}$$

We have bounded $\mathrm{E}[Y_n]$, but our ultimate goal is to bound $\mathrm{E}[X_n]$. As Exercise 12.4-4 asks you to show, the function $f(x) = 2^x$ is convex (see page 1199). Therefore, we can employ Jensen's inequality (C.26), which says that

$$\begin{aligned}
2^{\mathrm{E}[X_n]} &\le \mathrm{E}\left[2^{X_n}\right] \\
&= \mathrm{E}[Y_n] ,
\end{aligned}$$

as follows:

$$2^{\mathrm{E}[X_n]} \le \frac{1}{4}\binom{n+3}{3}$$

$$= \frac{1}{4} \cdot \frac{(n+3)(n+2)(n+1)}{6}$$
$$= \frac{n^3 + 6n^2 + 11n + 6}{24} .$$

Taking logarithms of both sides gives $\mathrm{E}[X_n] = O(\lg n)$. ∎

Exercises

12.4-1
Prove equation (12.3).

12.4-2
Describe a binary search tree on n nodes such that the average depth of a node in the tree is $\Theta(\lg n)$ but the height of the tree is $\omega(\lg n)$. Give an asymptotic upper bound on the height of an n-node binary search tree in which the average depth of a node is $\Theta(\lg n)$.

12.4-3
Show that the notion of a randomly chosen binary search tree on n keys, where each binary search tree of n keys is equally likely to be chosen, is different from the notion of a randomly built binary search tree given in this section. (*Hint:* List the possibilities when $n = 3$.)

12.4-4
Show that the function $f(x) = 2^x$ is convex.

12.4-5 ⋆
Consider RANDOMIZED-QUICKSORT operating on a sequence of n distinct input numbers. Prove that for any constant $k > 0$, all but $O(1/n^k)$ of the $n!$ input permutations yield an $O(n \lg n)$ running time.

Problems

12-1 Binary search trees with equal keys
Equal keys pose a problem for the implementation of binary search trees.

a. What is the asymptotic performance of TREE-INSERT when used to insert n items with identical keys into an initially empty binary search tree?

We propose to improve TREE-INSERT by testing before line 5 to determine whether $z.key = x.key$ and by testing before line 11 to determine whether $z.key = y.key$.