A decision tree argument can also be used to establish a lower bound on the worst-case number of comparisons required to sort a list of n elements. As we did for the search problem, let us see what we can say about a decision tree for sorting based on comparisons, regardless of the algorithm it represents. The leaves of such a tree represent the final outcomes, that is, the various ordered arrangements of the n items. There are $n!$ such arrangements, so if p is the number of leaves in the decision tree, then $p \geq n!$. The worst case will equal the depth of the tree. But it is also true that if the tree has depth d, then $p \leq 2^d$ (Exercise 43 of Section 6.2). Taking the base 2 logarithm of both sides of this inequality, we get $\log p \leq d$ or, because d is an integer, $d = \lceil \log p \rceil$. Finally, we obtain

$$d = \lceil \log p \rceil \geq \lceil \log n! \rceil$$

This result proves the following theorem.

THEOREM ON THE LOWER BOUND FOR SORTING
Any algorithm that sorts an n-element list by comparing pairs of items from the list must do at least $\lceil \log n! \rceil$ comparisons in the worst case.

It can be shown (Exercise 23) that $\log n! = \Theta(n \log n)$. Therefore we have proved that sorting n elements by comparing pairs of list items is bounded below by $\Theta(n \log n)$, whereas searching by comparing the target element to the list items is bounded below by $\Theta(\log n)$. As expected, it takes more work to sort than to search.

SECTION 6.3 REVIEW

TECHNIQUES

- Ⓦ Draw decision trees for sequential search and binary search on n-element lists.
- Ⓦ Create a binary search tree.

MAIN IDEAS

- Decision trees represent the sequences of possible actions for certain algorithms.
- Analysis of a generic decision tree for algorithms that solve a certain problem may lead to lower bounds on the minimum amount of work needed to solve the problem in the worst case.
- The task of searching an n-element list for a target value x, if done by comparing x to elements in the list, requires at least $\lfloor \log n \rfloor + 1$ comparisons in the worst case.
- The task of sorting an n-element list, if done by comparing pairs of list elements, requires at least $\lceil \log n! \rceil$ comparisons in the worst case.

EXERCISES 6.3

1. Draw the decision tree for sequential search on a list of three elements.
2. Draw the decision tree for sequential search on a list of six elements.
3. Draw the decision tree for binary search on a sorted list of seven elements. What is the depth of the tree?
4. Draw the decision tree for binary search on a sorted list of four elements. What is the depth of the tree?

5. Consider a search algorithm that compares an item with the last element in a list, then the first element, then the next-to-last element, then the second element, and so on. Draw the decision tree for searching a six-element sorted list. What is the depth of the tree? Does it appear that this is an optimal algorithm in the worst case?
6. Consider a search algorithm that compares an item with an element one-third of the way through the list; based on that comparison, it then searches either the first one-third or the second two-thirds of the list. Draw the decision tree for searching a nine-element sorted list. What is the depth of the tree? Does it appear that this is an optimal algorithm in the worst case?
7. a. Given the data

$$9, 5, 6, 2, 4, 7$$

construct the binary search tree. What is the depth of the tree?

b. Find the average number of comparisons done to search for an item that is known to be in the list using binary tree search on the tree of part (a). (*Hint*: Find the number of comparisons for each of the items.)

8. a. Given the data

$$g, d, r, s, b, q, c, m$$

construct the binary search tree. What is the depth of the tree?

b. Find the average number of comparisons done to search for an item that is known to be in the list using binary tree search on the tree of part (a). (*Hint*: Find the number of comparisons for each of the items.)

9. a. For a set of six data items, what is the minimum worst-case number of comparisons a search algorithm must perform?

b. Given the set of data items $\{a, d, g, i, k, s\}$, find an order in which to enter the data so that the corresponding binary search tree has the minimum depth.

10. a. For a set of nine data items, what is the minimum worst-case number of comparisons a search algorithm must perform?

b. Given the set of data items $\{4, 7, 8, 10, 12, 15, 18, 19, 21\}$, find an order in which to enter the data so that the corresponding binary search tree has the minimum depth.

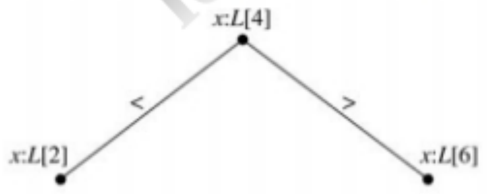
11. An inorder tree traversal of a binary search tree produces a listing of the tree nodes in alphabetical or numerical order. Construct a binary search tree for "To be or not to be, that is the question," and then do an inorder traversal.
12. Construct a binary search tree for "In the high and far-off times the Elephant, *O* Best Beloved, had no trunk," and then do an inorder traversal. (See Exercise 11.)
13. Use the theorem on the lower bound for sorting to find lower bounds on the number of comparisons required in the worst case to sort lists of the following sizes:

a. 4 b. 8 c. 16

14. Contrast the number of comparisons required for selection sort and merge sort in the worst case with the lower bounds found in Exercise 13 (see Exercise 23 in Section 3.3). What are your conclusions?

Exercises 15–20 concern the problem of identifying a counterfeit coin (one that is two heavy or too light) from a set of *n* coins. A balance scale is used to weigh a group of any number of coins from the set against a like number of coins from the set. The outcome of such a comparison is that group A weighs less than, the same as, or more than group *B*. A decision tree representing the sequence of comparisons done will thus be a ternary tree, where an internal node can have three children.

15. One of five coins is counterfeit and is lighter than the other four. The problem is to identify the counterfeit coin.
    a. What is the number of final outcomes (the number of leaves in the decision tree)?
    b. Find a lower bound on the number of comparisons required to solve this problem in the worst case.
    c. Devise an algorithm that meets this lower bound (draw its decision tree).
16. One of five coins is counterfeit and is either too heavy or too light. The problem is to identify the counterfeit coin and determine whether it is heavy or light.
    a. What is the number of final outcomes (the number of leaves in the decision tree)?
    b. Find a lower bound on the number of comparisons required to solve this problem in the worst case.
    c. Devise an algorithm that meets this lower bound (draw its decision tree).
17. One of four coins is counterfeit and is either too heavy or too light. The problem is to identify the counterfeit coin but not to determine whether it is heavy or light.
    a. What is the number of final outcomes (the number of leaves in the decision tree)?
    b. Find a lower bound on the number of comparisons required to solve this problem in the worst case.
    c. Devise an algorithm that meets this lower bound (draw its decision tree).
18. One of four coins is counterfeit and is either too heavy or too light. The problem is to identify the counterfeit coin and determine whether it is heavy or light.
    a. What is the number of final outcomes (the number of leaves in the decision tree)?
    b. Find a lower bound on the number of comparisons required to solve this problem in the worst case.
    c. Prove that no algorithm exists that can meet this lower bound. (*Hint*: The first comparison can be made with either two coins or four coins. Consider each case.)
19. Devise an algorithm to solve the problem of Exercise 18 using three comparisons in the worst case.
20. One of eight coins is counterfeit and is either too heavy or too light. The problem is to identify the counterfeit coin and determine whether it is heavy or light.
    a. What is the number of final outcomes (the number of leaves in the decision tree)?
    b. Find a lower bound on the number of comparisons required to solve this problem in the worst case.
    c. Devise an algorithm that meets this lower bound (draw its decision tree).
21. In the decision tree for the binary search algorithm (and the binary tree search algorithm), we have counted each internal node as one comparison. For example, the top of Figure 6.53 looks like this:



To get to either of the child nodes of the root, we have assumed that one comparison has been done. However, the outcome of the comparison at each internal node is really a three-way branch:

$$x = \text{node element}$$
$$x < \text{node element}$$
$$x > \text{node element}$$

Think about how this three-way branch would be implemented in most programming languages, and write a more accurate expression than $1 + \lfloor \log n \rfloor$ for the number of comparisons in the worst case.