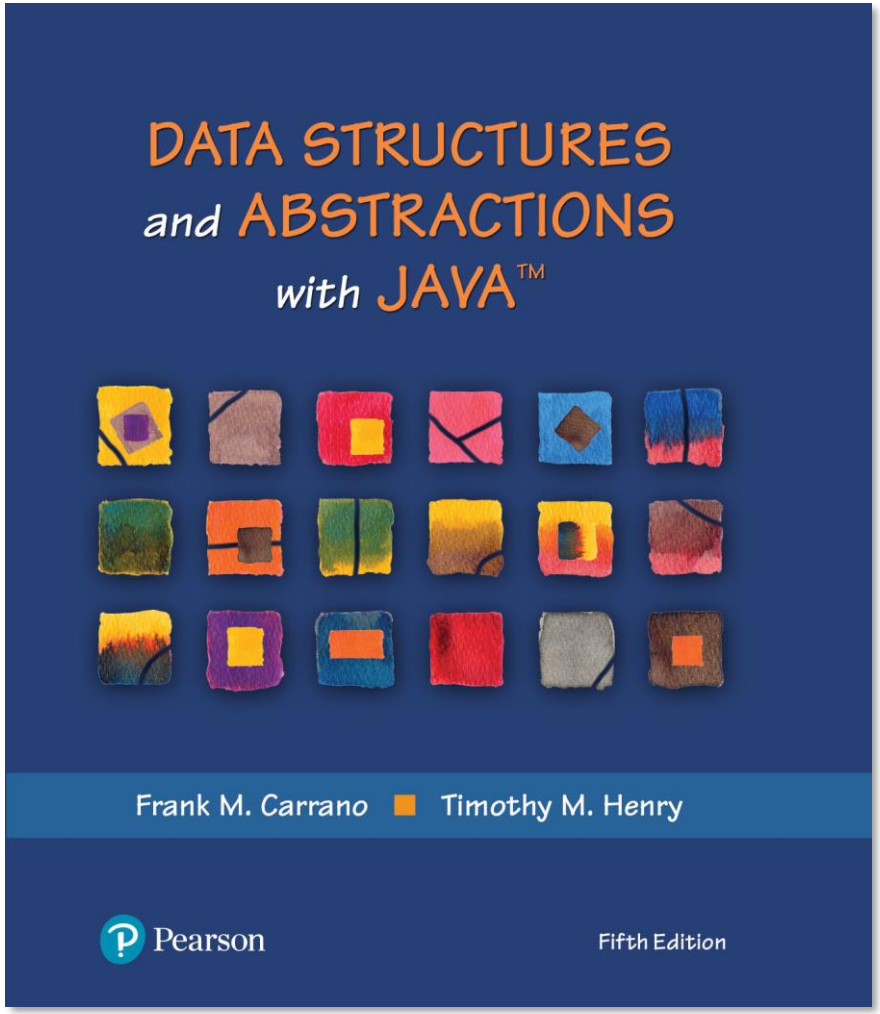
Data Structures and Abstractions with Java™

5th Edition



Chapter 14

Problem Solving with Recursion

Simple Solution to a Difficult Problem

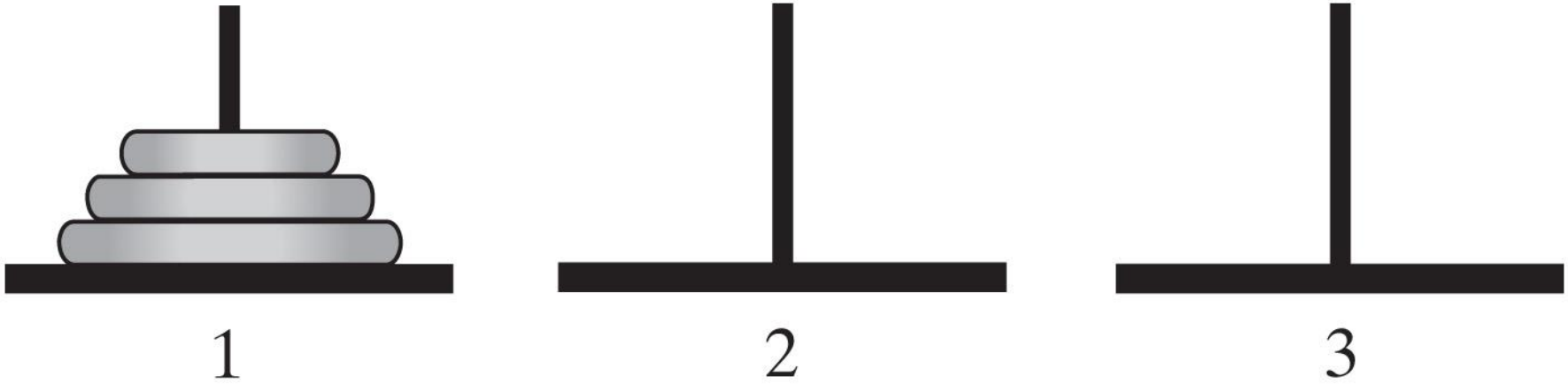


FIGURE 14-1 The initial configuration of the Towers of Hanoi for three disks

Simple Solution to a Difficult Problem

- Rules:
  - Move one disk at a time. Each disk moved must be the topmost disk.
  - No disk may rest on top of a disk smaller than itself.
  - You can store disks on the second (extra) pole temporarily, as long as you observe the previous two rules.

Simple Solution to a Difficult Problem (Part 1)

(a) The beginning configuration

(b) After moving a disk from pole 1 to pole 3

(c) After moving a disk from pole 1 to pole 2

(d) After moving a disk from pole 3 to pole 2

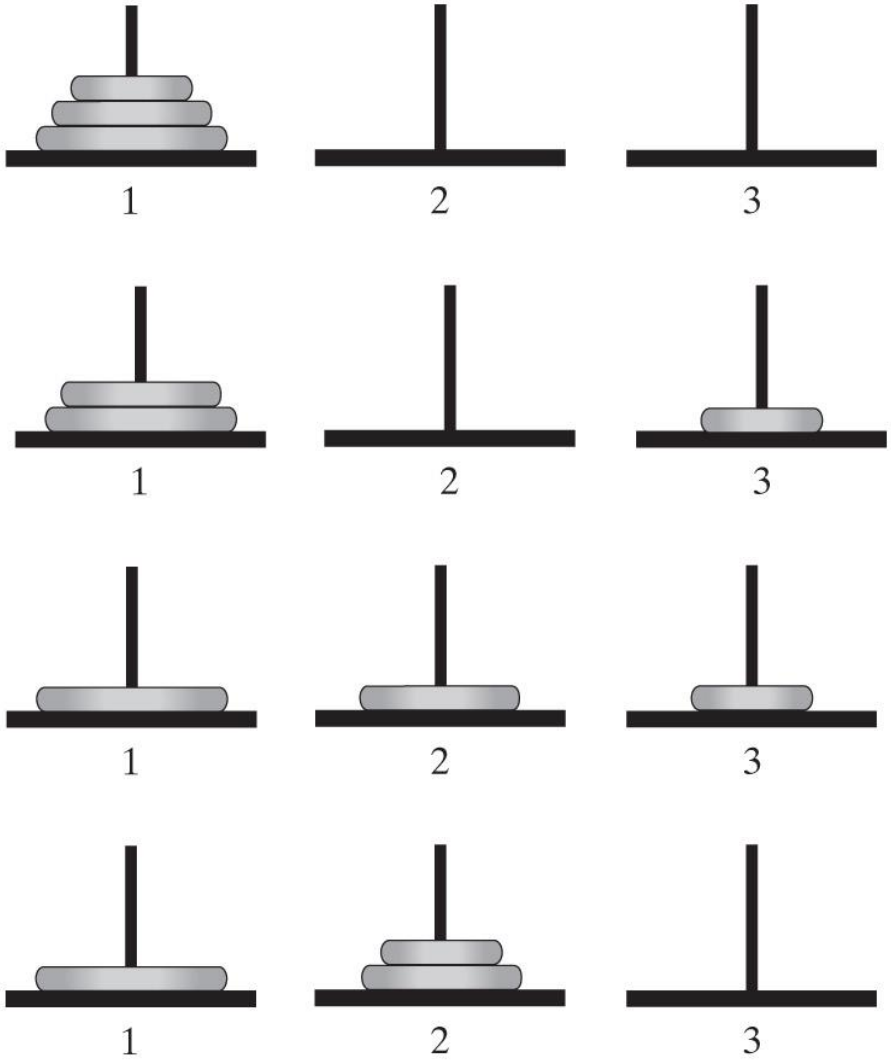


FIGURE 14-2 Sequence of moves for solving Towers of Hanoi problem with 3 disks

Simple Solution to a Difficult Problem (Part 2)

(e) After moving a disk from pole 1 to pole 3

(f) After moving a disk from pole 2 to pole 1

(g) After moving a disk from pole 2 to pole 3

(h) After moving a disk from pole 1 to pole 3

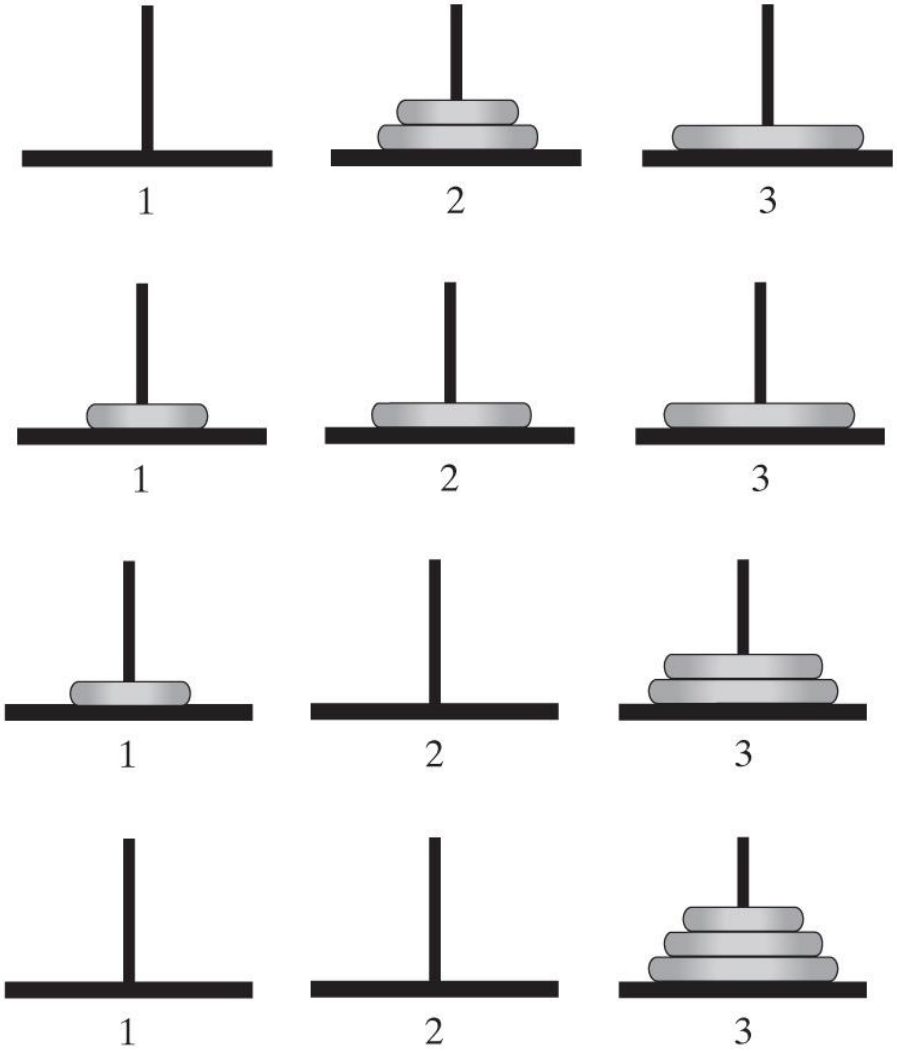


FIGURE 14-2 Sequence of moves for solving Towers of Hanoi problem with 3 disks

A Smaller Problem

(a) The original configuration

(b) After your friend moves three disks from pole 1 to pole 2

(c) After you move one disk from pole 1 to pole 3

(d) After your friend moves three disks from pole 2 to pole 3

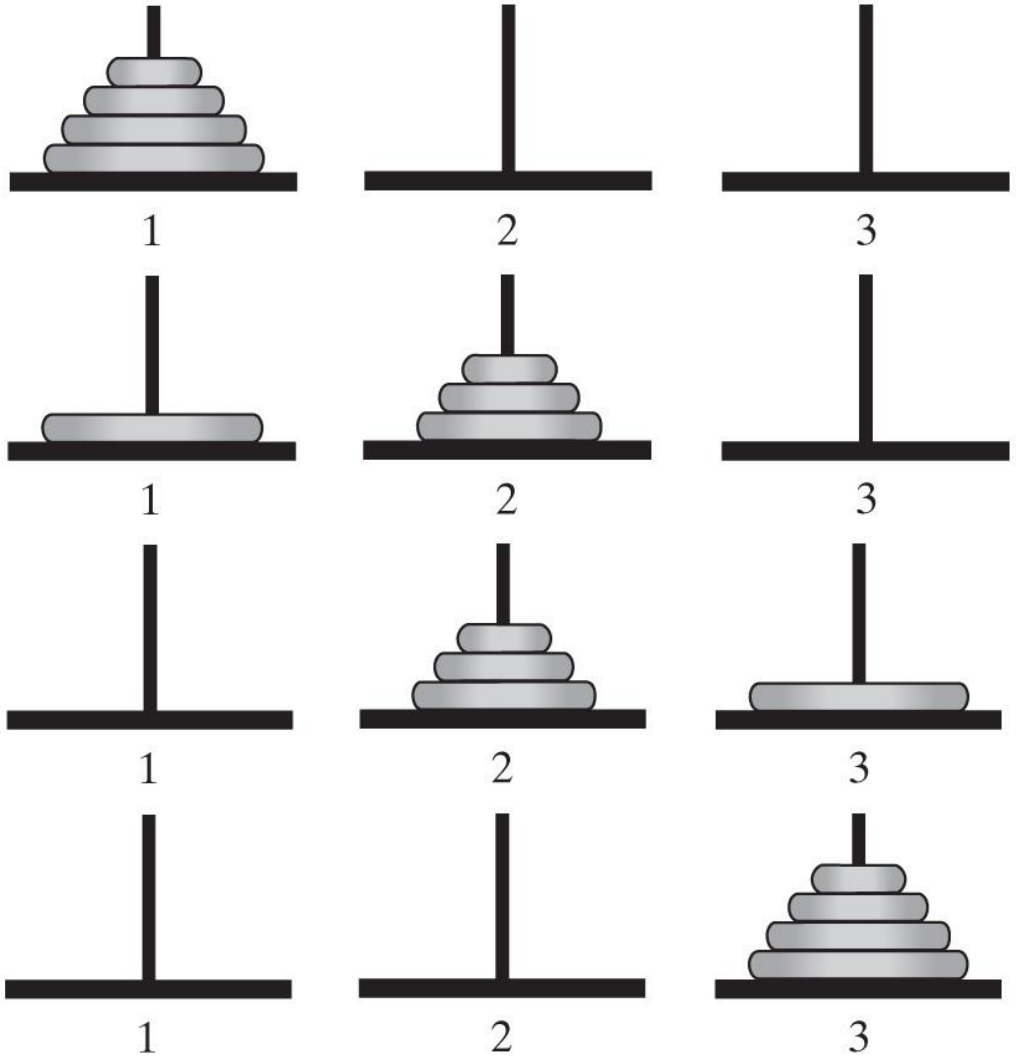


FIGURE 14-3 The smaller problems in a recursive solution for four disks

Solutions

Algorithm to move **numberOfDisks** *disks from* **startPole** *to* **endPole** *using* **tempPole**
as a spare according to the rules of the Towers of Hanoi problem

```
if (numberOfDisks == 1)
   Move disk from startPole to endPole
else
{
   Move all but the bottom disk from startPole to tempPole
   Move disk from startPole to endPole
   Move all disks from tempPole to endPole
}
```

Recursive algorithm to solve any number of disks.
Note: for n disks, solution will be $2^n - 1$ moves

Poor Solution to a Simple Problem

$$F_0 = 1$$
$$F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2} \text{ when } n \geq 2$$

```
Algorithm Fibonacci(n) if (n <= 1)
  return 1
else
  return Fibonacci(n − 1) + Fibonacci(n − 2)
```

Algorithm to generate Fibonacci numbers.
Why is this inefficient?

Poor Solution to a Simple Problem

(a) Recursively

F_2 is computed 5 times
F_3 is computed 3 times
F_4 is computed 2 times
F_5 is computed once
F_6 is computed once

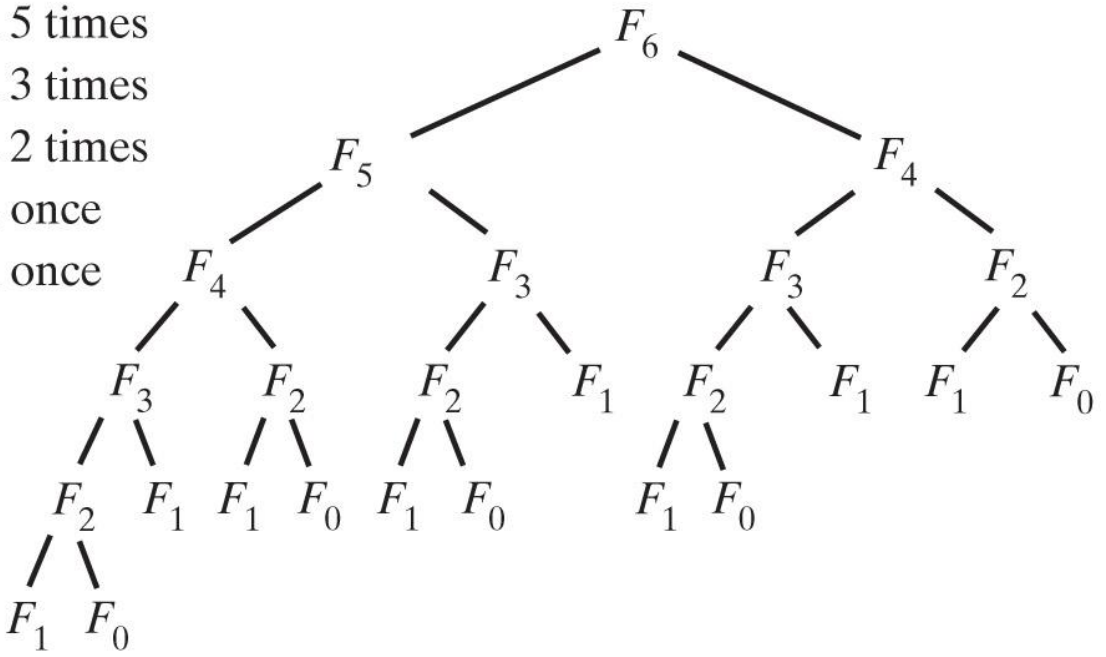


FIGURE 14-4a The computation of the Fibonacci number F6

Poor Solution to a Simple Problem

(a) Recursively

F_2 is computed 5 times
F_3 is computed 3 times
F_4 is computed 2 times
F_5 is computed once
F_6 is computed once

(b) Iteratively

$$F_0 = 1$$
$$F_1 = 1$$
$$F_2 = F_1 + F_0 = 2$$
$$F_3 = F_2 + F_1 = 3$$
$$F_4 = F_3 + F_2 = 5$$
$$F_5 = F_4 + F_3 = 8$$
$$F_6 = F_5 + F_4 = 13$$

FIGURE 14-4b The computation of the Fibonacci number F6

Backtracking

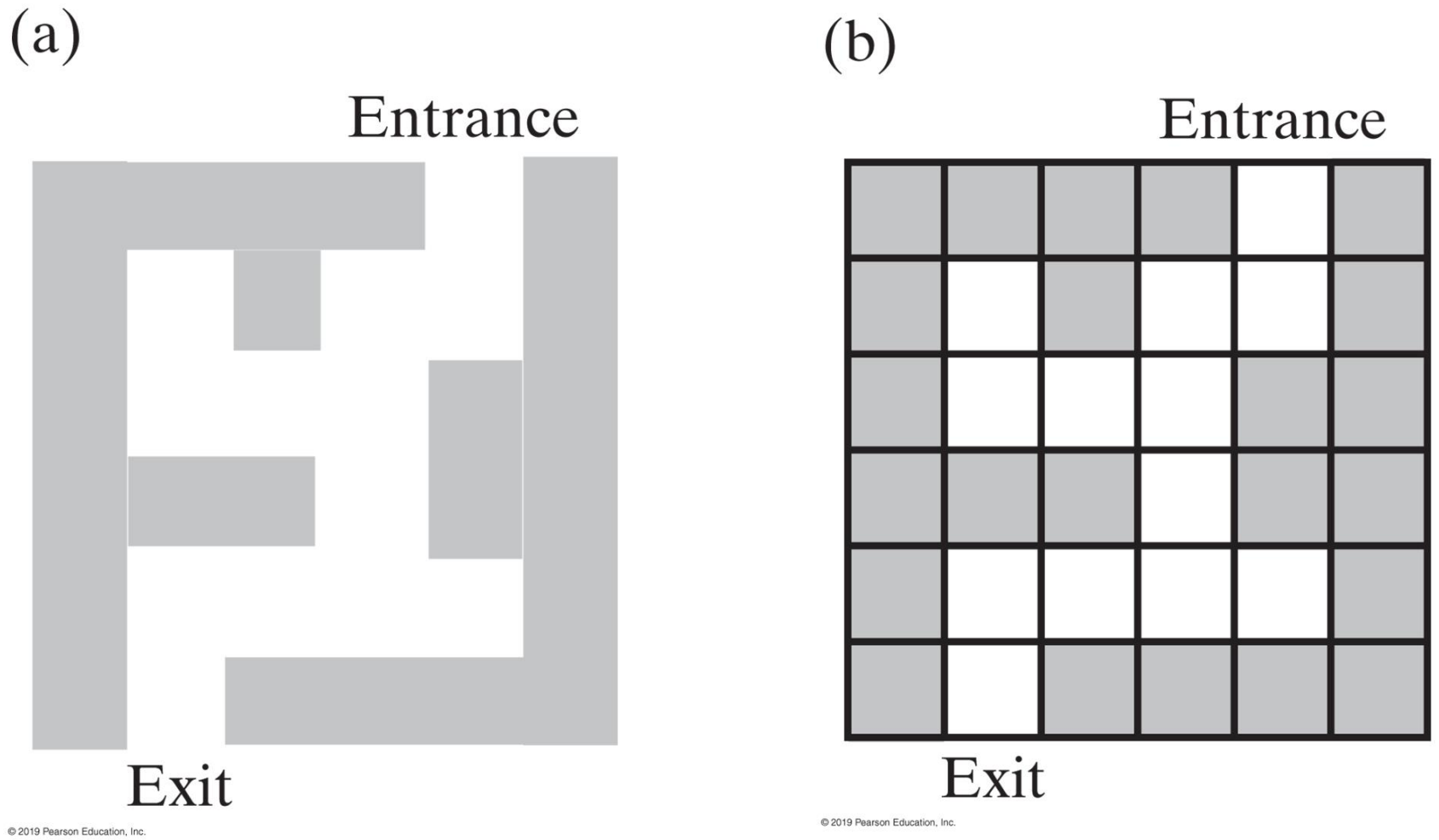


FIGURE 14-6 A two-dimensional maze with one entrance and one exit

Backtracking

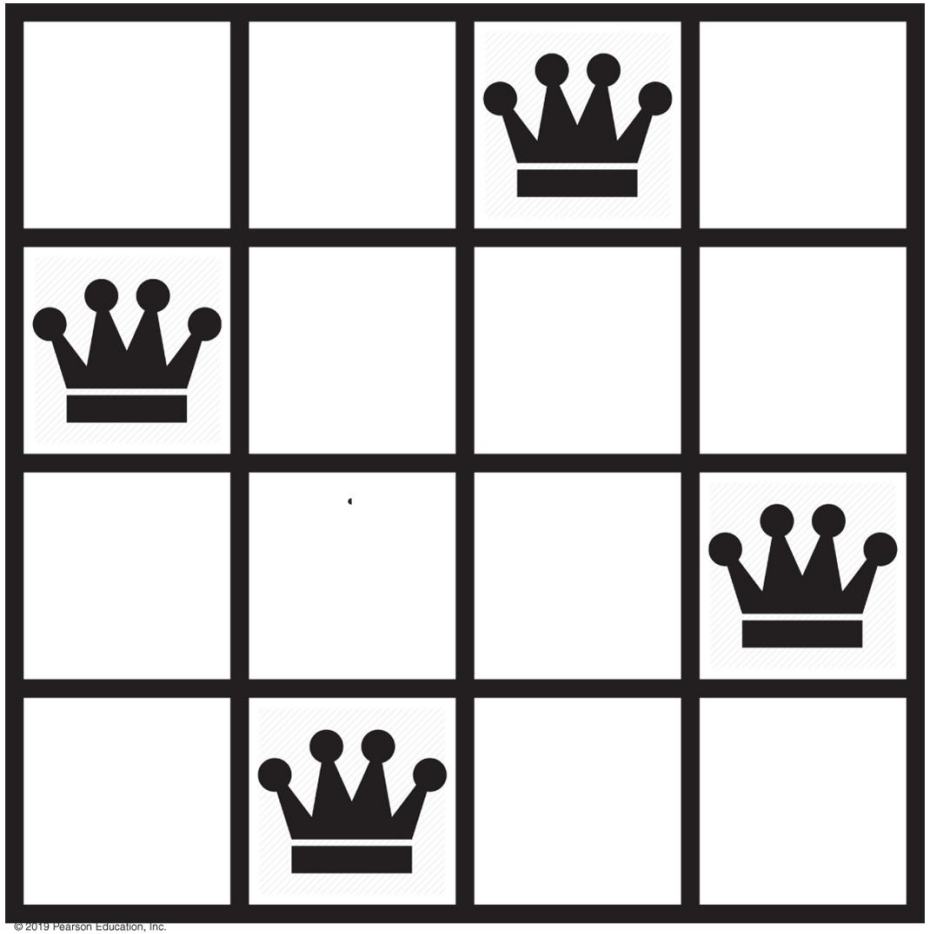
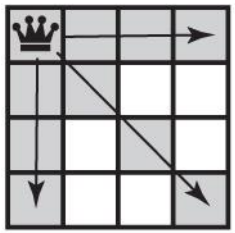


FIGURE 14-7 A solution to the four-queens problem

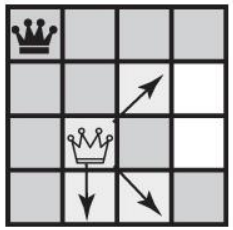
Backtracking - Queens Solution (Part 1)

= Can be attacked by existing queens = Can be attacked by the newly placed queen = Rejected during backtracking

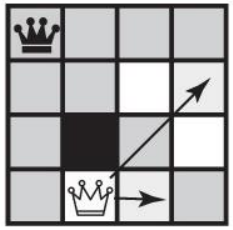
(a) The first queen in column 1.



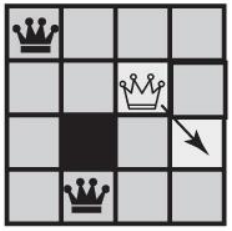
(b) The second queen in column 2. All of column 3 is under attack.



(c) Backtrack to column 2 and try another square for the queen.



(d) The third queen in column 3. All of column 4 is under attack.



(e) Backtrack to column 3, but the queen has no other move.

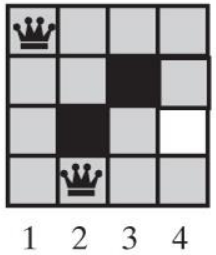
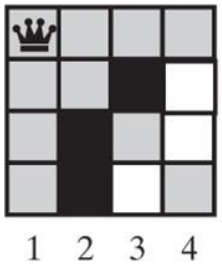


FIGURE 14-8 Solving the four-queens problem by placing one queen at a time in each column

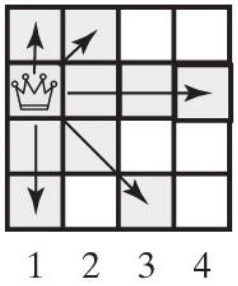
Backtracking - Queens Solution (Part 2)

= Can be attacked by existing queens = Can be attacked by the newly placed queen = Rejected during backtracking

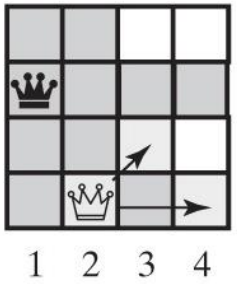
(f) Backtrack to column 2, but the queen has no other move.



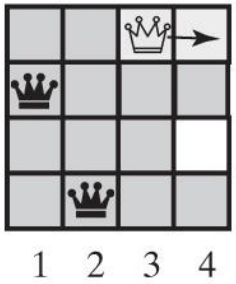
(g) Backtrack to column 1 and try another square for the queen.



(h) The second queen in column 2.



(i) The third queen in column 3.



(j) The fourth queen in column 4. Solution!

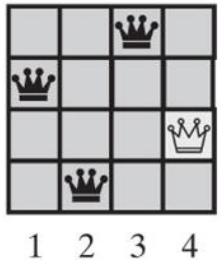


FIGURE 14-8 Solving the four-queens problem by placing one queen at a time in each column

End

Chapter 14