

Lecture 21: Problem Solving with Recursion

CS 0445: Data Structures

Constantinos Costa

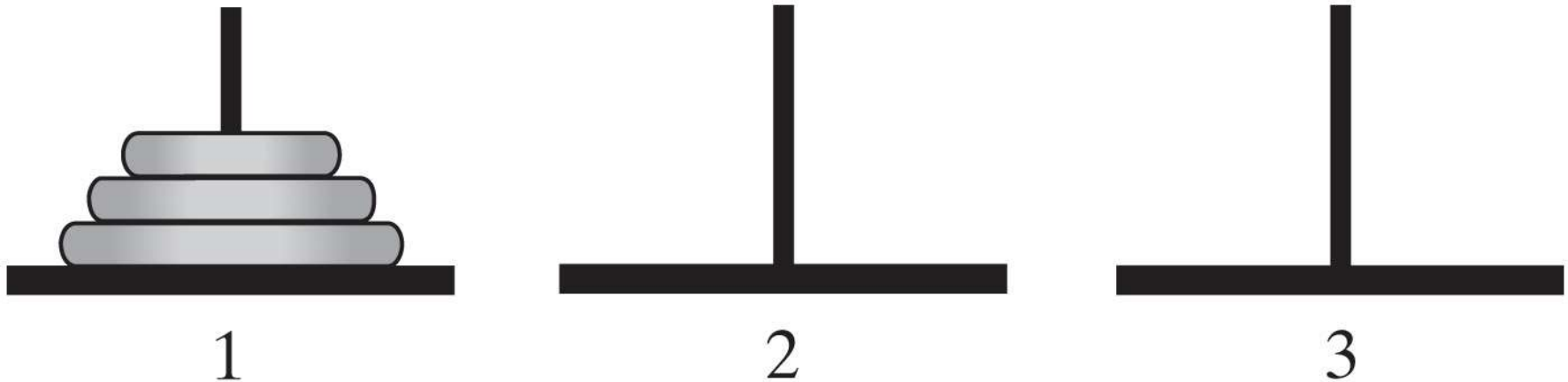
<http://db.cs.pitt.edu/courses/cs0445/current.term/>

Oct 23, 2019, 8:00-9:15
University of Pittsburgh, Pittsburgh, PA



Simple Solution to a Difficult Problem

- The initial configuration of the Towers of Hanoi for three disks



© 2019 Pearson Education, Inc.



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.

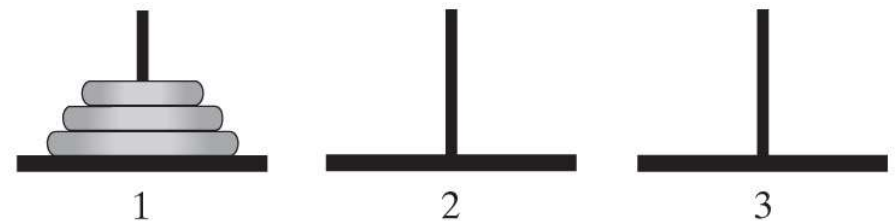


Simple Solution to a Difficult Problem (Part 1)

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

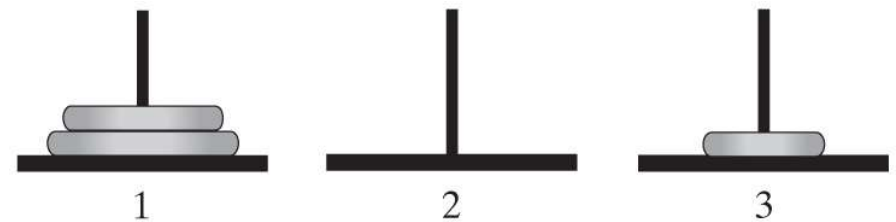
(a) The beginning configuration

© 2019 Pearson Education, Inc.



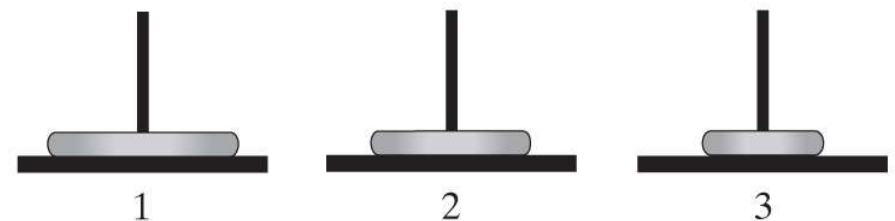
(b) After moving a disk from pole 1 to pole 3

© 2019 Pearson Education, Inc.



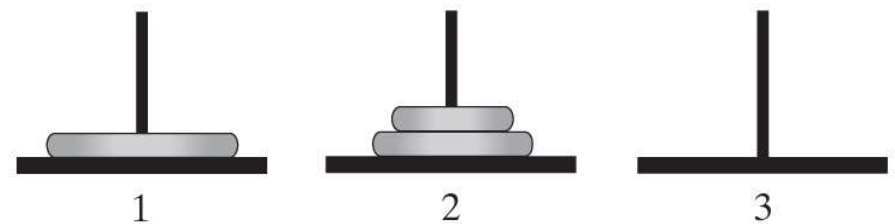
(c) After moving a disk from pole 1 to pole 2

© 2019 Pearson Education, Inc.



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

© 2019 Pearson Education, Inc.

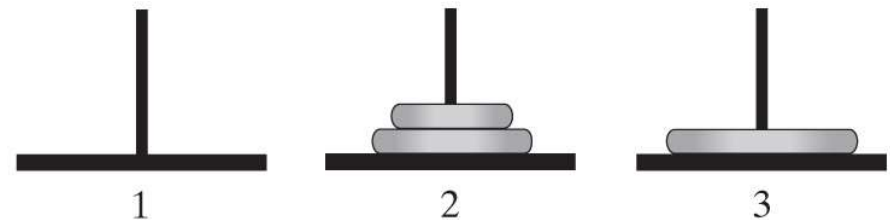


Simple Solution to a Difficult Problem (Part 2)

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

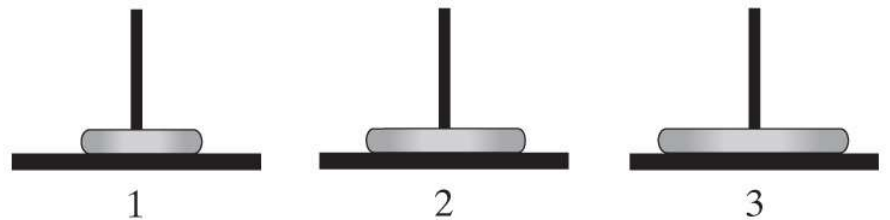
(e) After moving a disk from pole 1 to pole 3

© 2019 Pearson Education, Inc.



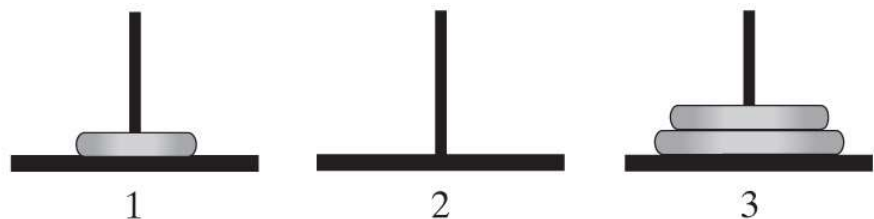
(f) After moving a disk from pole 2 to pole 1

© 2019 Pearson Education, Inc.



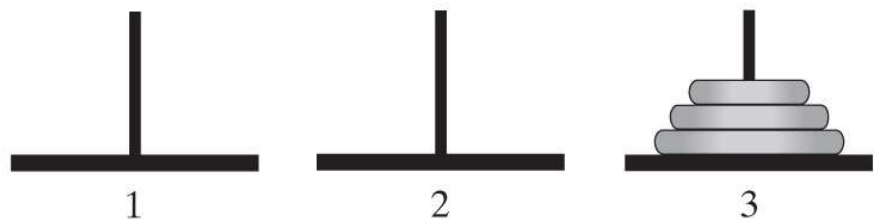
(g) After moving a disk from pole 2 to pole 3

© 2019 Pearson Education, Inc.



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

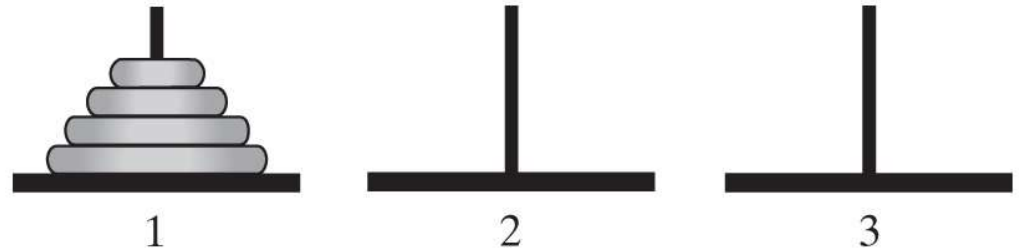
© 2019 Pearson Education, Inc.



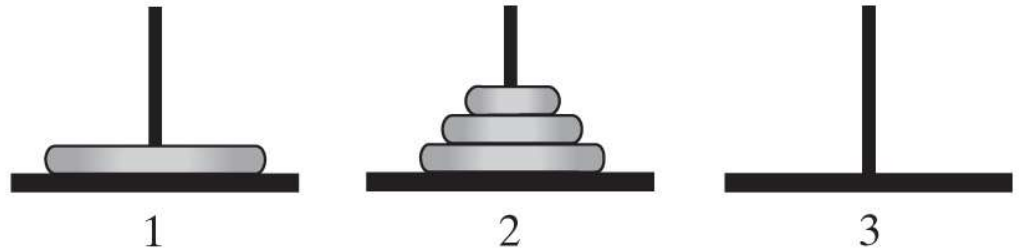
A Smaller Problem

- The smaller problems in a recursive solution for four disks

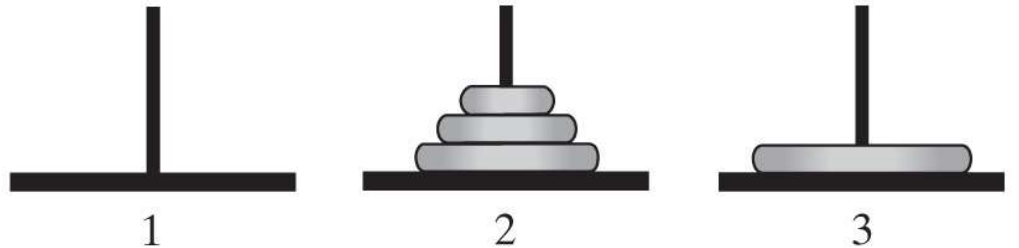
(a) The original configuration



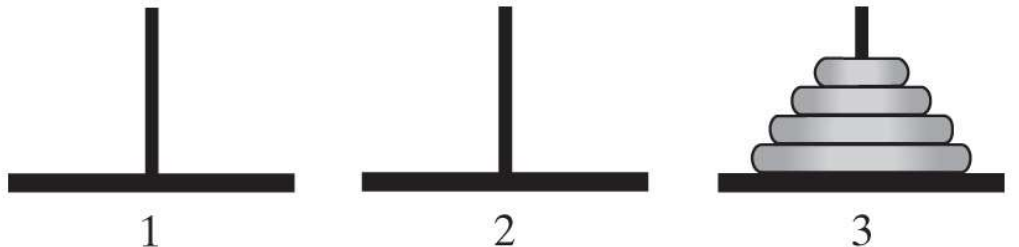
(b) After your friend moves three disks from pole 1 to pole 2



(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



Solutions

- Recursive algorithm to solve any number of disks.

Note: for n disks, solution will be $2^n - 1$ moves

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

}



Solution

```
// Java recursive program to solve tower of hanoi puzzle
class HanoiPuzzle {

    static String toColor(int i) {
        switch (i) {...}
    }

    // Java recursive function to solve tower of hanoi puzzle
    static void towerOfHanoi(int n, char startPole, char endPole, char tempPole) {
        if (n == 1) {
            System.out.println("Move disk (1) from pole " + startPole + " to pole " + endPole);
            return;
        }
        towerOfHanoi(n - 1, startPole, tempPole, endPole);
        System.out.println("Move disk (" + n + ") from pole " + startPole + " to pole " + endPole);
        towerOfHanoi(n - 1, tempPole, endPole, startPole);
    }

    // Driver method
    public static void main(String args[]) {
        int n = 4; // Number of disks
        towerOfHanoi(n, 'A', 'C', 'B'); // A, B and C are names of poles
    }
}
```



Poor Solution to a Simple Problem

- Algorithm to generate Fibonacci numbers.
- Why is this inefficient?

$$F_0 = 1$$

$$F_1 = 1$$

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

Algorithm **Fibonacci(n)** if ($n \leq 1$)

return 1

else

return Fibonacci(n - 1) + Fibonacci(n - 2)

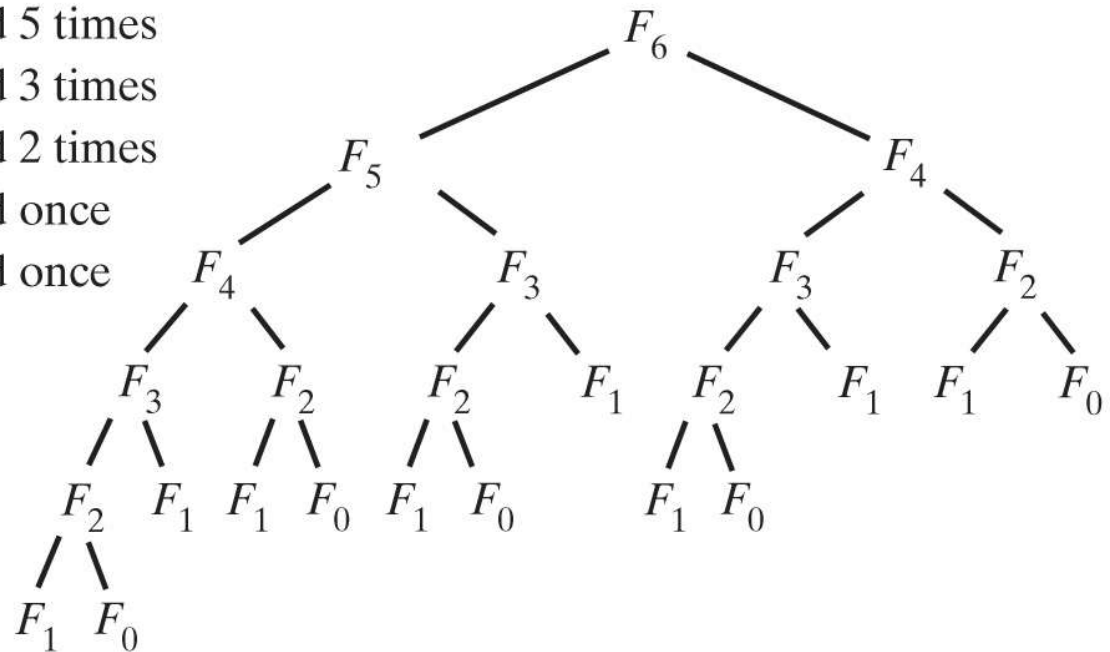


Poor Solution to a Simple Problem

- The computation of the Fibonacci number F_6

(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



© 2019 Pearson Education, Inc.



Poor Solution to a Simple Problem

- The computation of the Fibonacci number F_6

(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

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

© 2019 Pearson Education, Inc.



Indirect Recursion

- Example
 - Method A calls Method B
 - Method B calls Method C
 - Method C calls Method A
- Difficult to understand and trace
 - But does happen occasionally



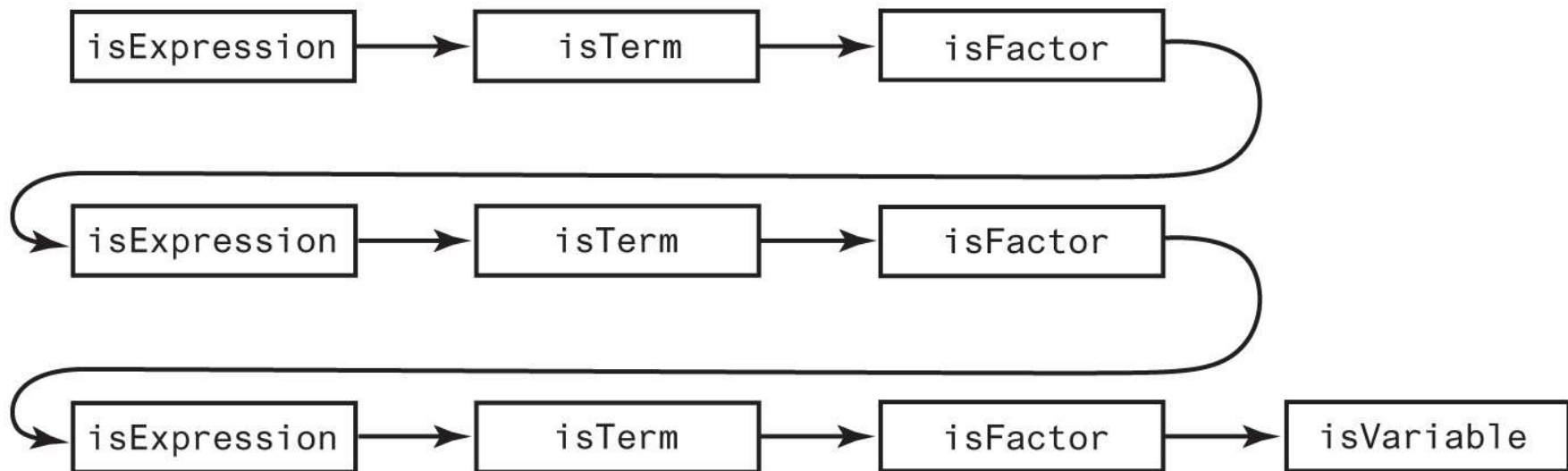
Indirect Recursion

- Consider evaluation of validity of an algebraic expression
 - Algebraic expression is either a term or two terms separated by a $+$ or $-$ operator
 - Term is either a factor or two factors separated by a $*$ or $/$ operator
 - Factor is either a variable or an algebraic expression enclosed in parentheses
 - Variable is a single letter



Indirect Recursion

- An example of indirect recursion



© 2019 Pearson Education, Inc.



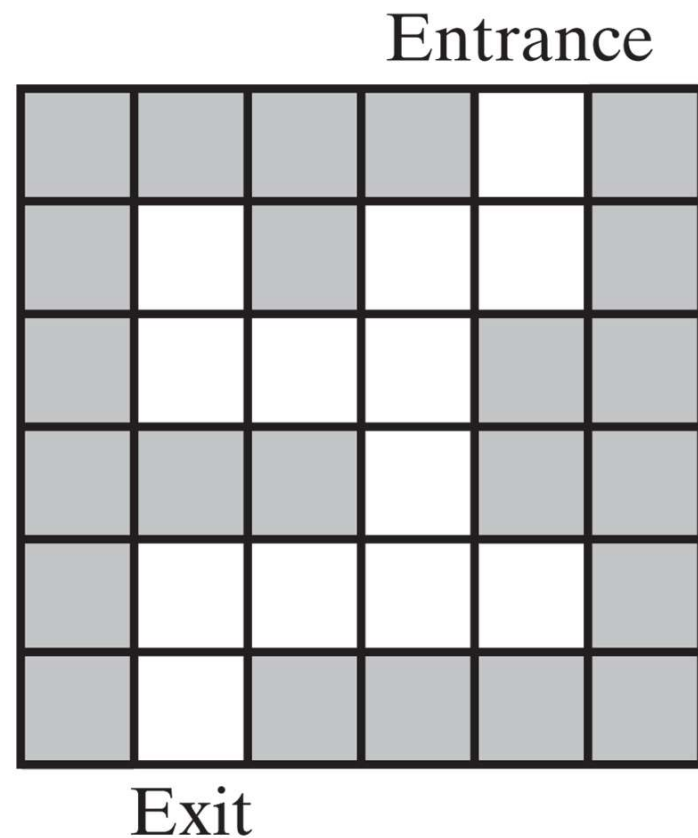
Backtracking

- A two-dimensional maze with one entrance and one exit

(a)



(b)



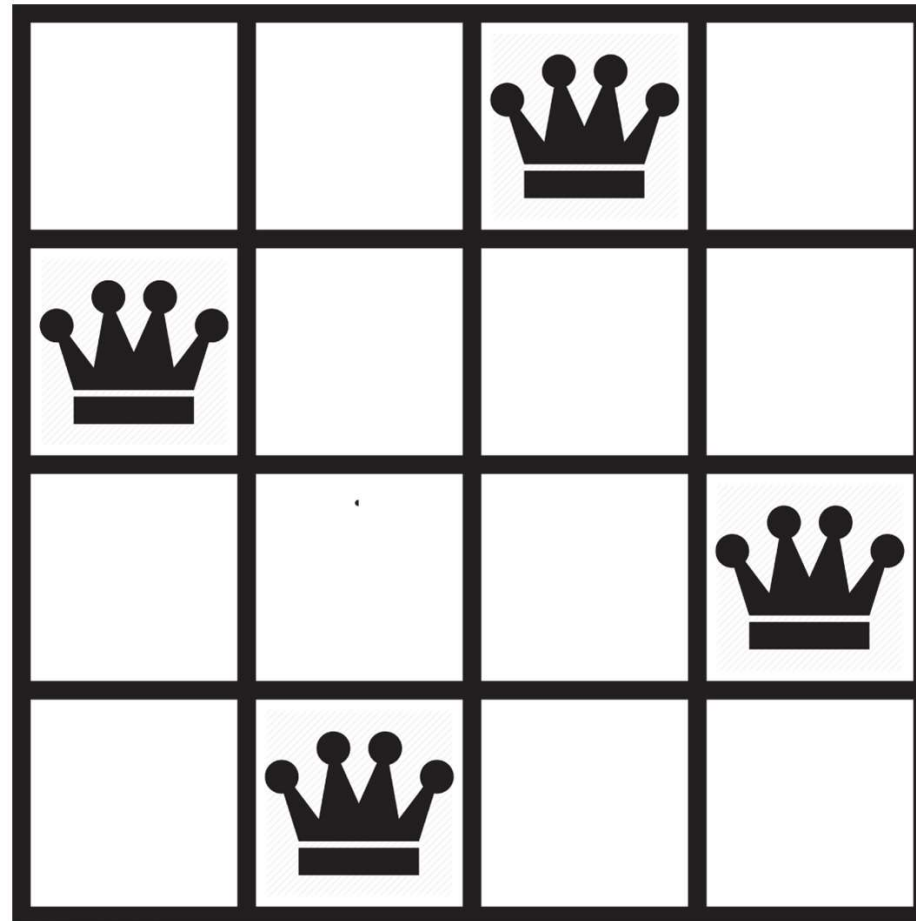
© 2019 Pearson Education, Inc.



© 2019 Pearson Education, Inc.

Backtracking

- A solution to the four-queens problem



© 2019 Pearson Education, Inc.



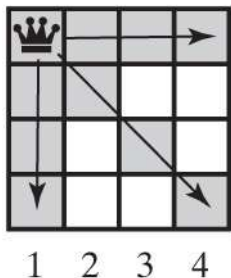
Backtracking - Queens Solution (Part 1)

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

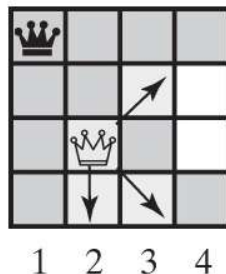
■ = Can be attacked by existing queens ■ = Can be attacked by the newly placed queen ■ = Rejected during backtracking

© 2019 Pearson Education, Inc.

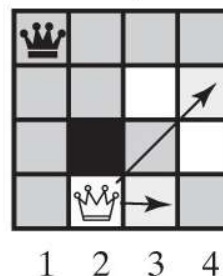
(a) The first queen in column 1.



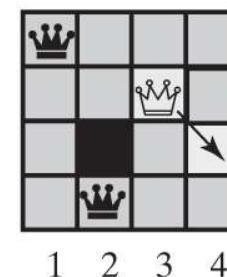
(b) The second queen in column 2. All of column 3 is under attack.



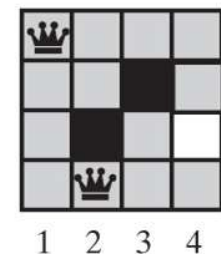
(c) Backtrack to column 2 and try another square for the queen.



(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.



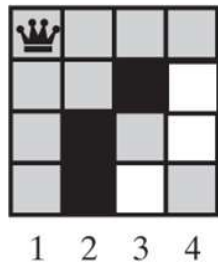
Backtracking - Queens Solution (Part 2)

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

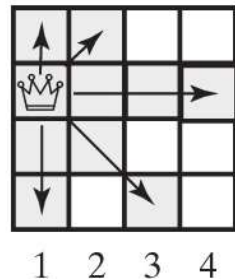
 = Can be attacked by existing queens
  = Can be attacked by the newly placed queen
  = Rejected during backtracking

© 2019 Pearson Education, Inc.

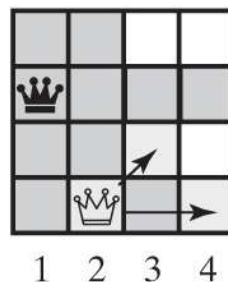
(f) Backtrack to column 2,
but the queen has
no other move.



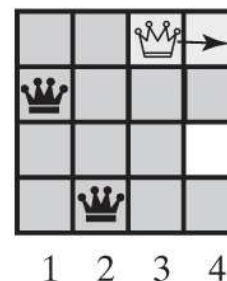
(g) Backtrack to column 1
and try another square
for the queen.



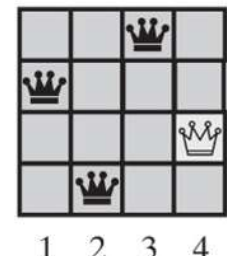
(h) The second queen
in column 2.



(i) The third queen
in column 3.



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



Happy Halloween!

**WHAT IS A GHOST'S
FAVORITE DATA TYPE?**



BOO - LEAN

